École Normale Supérieure de Lyon
Laboratoire de l'Informatique du Parallélisme

THÈSE

présentée et soutenue publiquement le 11 mai 2007 par

Vincent Nesme

pour l'obtention du grade de

Docteur de l'École Normale Supérieure de Lyon
spécialité : Informatique

au titre de l'École doctorale de mathématiques et d'informatique fondamentale de Lyon

Complexité en requêtes et symétries

Directeurs de thèse : Pascal Koiran
Natacha Portier

Après avis de : Peter Høyer
Sophie Laplante

Devant la commission d'examen formée de :

Peter Høyer	Membre/Rapporteur
Philippe Jorrand	Membre
Pascal Koiran	Membre
Sophie Laplante	Membre/Rapporteur
Natacha Portier	Membre
Jean-Pierre Tillich	Membre

Table des matières

Introduction **5**

1 Préliminaires **7**
1.1 Problèmes 7
1.2 Modèle de calcul déterministe 11
1.3 Modèle de calcul probabiliste 21
1.4 Modèle de calcul quantique 25
1.4.1 De la pertinence du modèle de calcul quantique 29

2 Le problème du sous-groupe caché **35**
2.1 Définition 35
2.1.1 Automorphisme et isomorphisme de graphe 35
2.1.2 Le sous-groupe caché 37
2.2 Calcul quantique 39
2.2.1 Algorithme standard 39
2.2.2 Un soupçon de théorie des groupes 41
2.2.3 Complexité 45

3 Bornes inférieures quantiques **53**
3.1 La méthode par adversaire 53
3.2 La méthode polynomiale 55
3.2.1 Le théorème principal 55
3.2.2 Les lemmes d'approximation 59
3.3 Le problème du sous-groupe caché dans $(\mathbb{Z}_p)^n$ 65
3.3.1 Définitions et énoncé du résultat 65
3.3.2 Preuve de la proposition 3.9 66

4 Symétries et complexité probabiliste **75**
4.1 Symétries 75
4.1.1 Définitions 75
4.1.2 Symétrisation 77
4.1.3 Convexité 81
4.1.4 Problèmes 83
4.2 Non-adaptativité 84

4.2.1 Définition 84
4.2.2 Programmation linéaire 85
4.2.3 Formule 87
4.2.4 Point de vue algorithmique 89
4.3 Quand l'adaptativité ne sert à rien 91
4.3.1 Le Problème 91
4.3.2 Propriétés générales 93
4.3.3 Problèmes totalement symétriques 94
4.3.4 Problèmes de collision 96
4.4 Exemples 97
4.4.1 Recherche dans un tableau non trié 97
4.4.2 Problème de Simon 99
4.4.3 One-to-one versus two-to-one 101
4.4.4 Translation cachée 104
4.5 Quelques comparaisons 106
4.5.1 La distance en variation 106
4.5.2 Sensibilité par blocs 109
4.5.3 Distance de Hellinger 111

Conclusion **115**

Remerciements **117**

Notations **119**

Bibliographie **120**

Introduction

Les ramifications de la théorie de la complexité montrent assez qu'il s'agit là d'un sujet fort « complexe ». Suffisamment complexe pour que, face au but généralement proclamé de résoudre le problème « **P=NP** », un nombre considérable de pistes se soient ouvertes pour contourner ce qui constitue l'obstacle majeur : il est fort difficile de prouver des bornes inférieures sur la complexité en temps dans le modèle classique des machines de Turing. Les théoriciens ont au fil des ans développé de nouveaux modèles, étudié des questions subtilement différentes, progressivement enrichi le bestiaire des classes de complexité[1] et des techniques mathématiques.

Nous discuterons principalement de deux modèles de calcul, qui sont le calcul probabiliste et le calcul quantique. Le premier est bien sûr connu de tous les théoriciens ; il s'impose avec évidence comme la formalisation de la programmation pratique faisant intervenir une source aléatoire parfaite, de la même manière que les machines de Turing traditionnelles peuvent formaliser la programmation pratique sans aléa. Le calcul quantique, malgré l'entrée fracassante qu'il fit dans le domaine de l'informatique théorique dans les années 90, reste moins connu, et c'est bien normal, car bien qu'il se veuille au départ une formalisation des possibilités effectives de calcul qu'offre la mécanique quantique, les applications pratiques en sont encore au stade embryonnaire.

La mesure de complexité que nous étudierons, dans l'un et l'autre modèles de calcul, est la complexité en requêtes, qui mesure le nombre de questions qu'un algorithme doit poser pour résoudre un problème posé sous forme de boîte noire. Cela signifie que l'entrée de l'algorithme, au lieu de lui être donnée tout d'un bloc, n'est dévoilée que petit à petit, au rythme des questions posées par l'algorithme, comme dans ce jeu où il faut reconnaître le visage d'une célébrité que l'on fait apparaître en enlevant case par case le masque qui recouvre la photographie. Le premier chapitre est consacré à définir des modèles de calcul adaptés à l'étude de la complexité en requêtes.

Le problème du sous-groupe caché est introduit dans le deuxième chapitre. Il s'agit en réalité d'une classe de problèmes sur lesquels l'attention des chercheurs en calcul quantique se porte particulièrement. Nous y présen-

[1] Voir [Wik].

terons l'algorithme quantique standard pour les groupes commutatifs, et en analyserons la complexité en requêtes, pour la comparer dans le troisième chapitre avec une borne inférieure. Cette borne inférieure est obtenue par l'utilisation judicieuse d'une méthode bien connue, la *méthode polynomiale*. Une étape importante de ce processus est la *symétrisation*, qui consiste à utiliser les symétries du problème considéré pour simplifier l'expression de la borne inférieure recherché. Nous expliquerons plus en profondeur dans le quatrième et dernier chapitre ce que sont les symétries d'un problème et ce qu'est la symétrisation, avant d'en montrer l'application au calcul exact, en calcul probabiliste, de la complexité en requêtes des problèmes vérifiant certaines hypothèses de symétrie.

Chapitre 1

Préliminaires

Il est essentiel dans toute discussion scientifique, au moins en théorie, d'avoir au préalable l'assurance que les bases du discours sont communes à toutes les parties, et que leur compréhension ne fait pas de doute.

Bien entendu, l'intuition, la mise au point d'images mentales appropriées, l'imagination surtout jouent un rôle crucial dans la découverte et la transmission du savoir. Mais il est également important de toujours pouvoir, à la demande, dénuder ces constructions aériennes, pour en révéler le noyau logique prenant racine dans la base de fondamentaux admis par tous, ou bien le cas échéant, au contraire, admettre que l'intuition n'est pas totalement justifiable et est susceptible d'être remise en question.

Nous avons à plusieurs reprises entendu exprimer l'opinion selon laquelle l'informatique théorique serait un domaine peu rigoureux, trop peu pour être considéré comme une discipline mathématique. Nous voulons croire qu'il ne s'agit que de l'expression d'une méconnaissance de la réalité du terrain.

Nous avons tenté d'être le plus précis et rigoureux possible, dans les définitions comme dans l'énoncé des résultats et leur démonstration. Le formalisme des définitions pourra paraître inutilement lourd ; il n'est pas là pour alourdir ou ralentir la lecture. Nous n'avons fait que reprendre le vocabulaire communément utilisé, à cette exception près que nous utilisons le terme de « problème élémentaire », qui ne devrait pas présenter de difficulté.

1.1 Problèmes

Une notion fondamentale en matière de théorie de la complexité est celle de problème. Une manière de voir un problème est comme la spécification d'un algorithme : on décrit, pour chaque entrée, le résultat attendu. Les entrées comme les résultats peuvent être a priori codés de différentes manières.

Traditionnellement, les entrées sont codées sous forme de mots, c'est-à-dire de suites finies de lettres, les lettres étant choisies dans un alphabet fini fixé. Pour une raison qui apparaîtra plus tard, nous allons coder les entrées

sous une forme proche, mais légèrement différente : comme des applications. Si l'on veut garder en tête qu'une entrée est un mot, on peut appliquer la correspondance suivante pour rentrer dans le cadre de notre formalisation :

$$w = u_1 u_2 \dots u_n \quad \longrightarrow \quad f : i \mapsto u_i$$

Quant au format de sortie, nous allons rester encore plus vague, pour l'instant, et nous contenter de dire que le résultat doit appartenir à un certain ensemble... Une catégorie importante de problèmes est celle des problèmes « de décision », où le résultat attendu est simplement un booléen (« vrai » ou « faux »). Mais on verra dans la suite que le résultat peut prendre des formes beaucoup plus incongrues : entiers, fonctions, groupes, graphes, etc.

Pour des raisons techniques, nous allons d'abord définir une version restreinte des problèmes, les « problèmes élémentaires ». Si l' les entrées comme des mots, les problèmes élémentaires sont ceux pour lesquels tous les mots en entrée sont supposés avoir la même taille.

Définition 1.1
Un **problème élémentaire** $\mathscr{P}$ *est la donnée d'ensembles I, J, R, d'une partie $\mathscr{S} \subseteq J^I$ et d'une application f de $\mathscr{S}$ dans R. Formellement, on peut écrire $\mathscr{P}$ sous la forme du quintuplet*[1] *$(I, J, R, \mathscr{S}, f)$.*

Les éléments de $\mathscr{S}$ sont nommés les **instances** *de $\mathscr{P}$. I est l'ensemble des* **entrées**, *J l'ensemble des* **indices** *et R l'ensemble des* **résultats**.

Si $|J| = 2$, on dit que $\mathscr{P}$ est un problème élémentaire binaire.

Si $R = \mathbb{B} = \{\top, \bot\}$, on dit que $\mathscr{P}$ est un problème de décision élémentaire.

Cette définition à l'air barbare, quelques exemples sont sans doute nécessaires pour l'illustrer. Commençons par un grand classique, la primalité.

Exemple 1.2
On code les entiers naturels non nuls n par les fonctions

$$\tilde{n} : \left(\begin{array}{rcl} [\lfloor \log_2 n \rfloor + 1] & \to & [2] \\ i & \mapsto & \lfloor 2^{-i} \cdot n \rfloor \mod 2 \end{array} \right)$$

Autrement dit, $\tilde{n}(i)$ est le i-ème bit de la représentation binaire n — en considérant que le bit de poids le plus faible porte le numéro 0 ; de sorte que

$$n = \sum_{i=0}^{\lfloor \log_2 n \rfloor} 2^i \cdot \tilde{n}(i).$$

Si l'on représente ces fonctions par la suite des valeurs qu'elles prennent sur $0, 1, \dots, \lfloor \log_2 n \rfloor$, 42 est par exemple représenté par (0 1 0 1 0 1). Il y a

[1] Il semble que pour des raisons obscures toute notion un tant soit peu importante d'informatique théorique se formalise comme un quintuplet d'objets divers. Il s'agit là sans doute d'une loi de la nature ou quelque chose comme ça.

ainsi une correspondance biunivoque entre les entiers non nuls et les suites finies de 0 et de 1 terminant par un 1 ; ainsi on associe, à une fonction w « convenable », l'entier $\hat{w}$ tel que $\tilde{\hat{w}} = w$. La **taille** d'un entier naturel, pour ce codage, est $\lfloor \log_2 n \rfloor + 1$. Si l'on fixe une taille t, la question « l'entier n de taille t est-il premier ? » peut se formaliser sous la forme d'un problème élémentaire. Remarquons que n est un entier de taille t si et seulement si sa fonction associée $\tilde{n}$ a pour domaine $[t]$ et vérifie $\tilde{n} = 1$. On va donc formaliser la primalité en choisissant ces paramètres : $I = [t]$, $J = [2]$, $\mathscr{S} = \left\{ w \in J^I / w\,(t-1) = 1 \right\}$, $R = \{\top, \bot\}$ — on va donc avoir à faire à un problème de décision élémentaire binaire — et bien entendu

$$f(w) = \begin{cases} \top & \text{si } \hat{w} \text{ est un nombre premier} \\ \bot & \text{sinon} \end{cases} .$$

Déjà dans ce premier exemple $\mathscr{S}$ est un sous-ensemble strict de J^I. On dit dans ce cas que le problème est « **à promesse** » — dans le cas contraire on dit qu'il est **total** — car on considère que $\mathscr{S}$ représente la promesse que les entrées de l'algorithme sont, justement, dans $\mathscr{S}$. Dans le cas de notre exemple, le fait que I et J vaillent respectivement $[t]$ et $[2]$ nous dit que les mots considérés sont de taille t, sur l'alphabet $\{0, 1\}$. Là-dessus $\mathscr{S}$ nous donne ce renseignement supplémentaire, la *promesse* que ces mots terminent par 1, promesse dont pourront faire usage les algorithmes chargés de résoudre le problème ; mais nous n'en sommes pas encore là.

En attendant, on peut d'ores et déjà constater que le choix de la représentation d'un entier sous la forme de son écriture binaire n'est pas anodin, puisque cela intervient dans la définition de la taille d'un entier. On peut définir plus généralement la taille d'un problème élémentaire, ou de manière équivalente la taille de ses instances, comme étant le cardinal de l'ensemble I de la définition 1.1. Ainsi, dans notre exemple, la taille de l'entier n était $\lfloor \log_2 n \rfloor + 1$, mais si l'on choisit de représenter l'entier n par son écriture en base 3, sa taille devient alors $\lfloor \log_3 n \rfloor + 1$. Comme les instances d'un problème élémentaire ont par définition toutes la même taille, les problèmes élémentaires reposant sur ces codages des entiers ne sont pas les mêmes.

On pourrait vouloir ne s'intéresser qu'aux nombres impairs, c'est-à-dire ceux dont le bit de poids faibles vaut 1. Le problème est naturellement plus « facile », puisqu'on a de l'information supplémentaire : on dit qu'on a ainsi défini un *sous-problème* élémentaire du problème de la primalité. Voici la définition :

Définition 1.3
$\mathscr{P}' = (I', J', R', \mathscr{S}', f')$ est un **sous-problème** *de $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ si $I' = I$, $J' = J$, $R' = R$, $\mathscr{S}' \subseteq \mathscr{S}$ et $f' = f|_{\mathscr{S}'}$.*

Il existe diverses autres manières d'affaiblir un problème élémentaire. Une autre manière que nous emploierons couramment consiste, à partir d'un problème $\mathscr{P} = (I, J, R, \mathscr{S}, f)$, à considérer que toute la gamme des réponses

possibles, c'est-à-dire les des éléments de R, n'est pas nécessaire, et qu'on se contenterait fort bien de seulement savoir dans quelle « catégorie » la réponse se trouve. Par exemple, imaginons qu'au lieu du problème de la primalité nous ayons défini un problème élémentaire correspondant à la question « combien l'entier n (d'une taille binaire donnée) a-t-il de diviseurs positifs ? » Le problème de la primalité s'obtiendrait à partir de celui-ci en considérant que seule la réponse « 2 » est importante, et que pour les autres il nous suffit de savoir que ce n'est pas 2. Formellement, on passe d'un problème élémentaire $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ à un problème élémentaire $\mathscr{P}' = (I, J, R', \mathscr{S}, f')$ tel qu'il existe $\pi : R \to R'$ tel que $f' = \pi \circ f$. Cette opération sera particulièrement utilisée dans le cas où $\mathscr{P}'$ est un problème de décision élémentaire, c'est-à-dire quand $R' = \mathbb{B}$; on dira alors que $\mathscr{P}'$ est un problème de décision (élémentaire) associé à $\mathscr{P}$, la plupart du temps que $\mathscr{P}'$ est *le* problème de décision associé quand il n'y aura pas d'ambiguïté.

La définition de l'isomorphisme entre problèmes élémentaires s'impose d'elle-même. Deux problèmes élémentaires sont isomorphes s'ils sont identiques via une identification de leurs entrées, de leurs indices et de leurs résultats. Autrement dit :

Définition 1.4
Deux problèmes élémentaires $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ et $\mathscr{P}' = (I', J', R', \mathscr{S}', f')$ sont dits **isomorphes** *s'il existe des bijections α, β, γ, respectivement de I dans I', de J dans J' et de R dans R', telles que*

- $\mathscr{S}' = \left\{\beta \circ x \circ \alpha^{-1} / x \in \mathscr{S}\right\}$ *et*
- *pour tout $x \in \mathscr{S}$, $f'\left(\beta \circ x \circ \alpha^{-1}\right) = \gamma\left(f(x)\right)$.*

Nous donnons maintenant la définition habituelle d'un problème, où les instances ne sont pas contraintes à avoir toutes la même taille.

Définition 1.5
Un **problème** *$\mathscr{P}$ est une application définie sur un ensemble S, qui à chaque $s \in S$ associe un problème élémentaire $\mathscr{P}_s$.*

Exemple 1.6
Reprenons notre exemple 1.2. Nous souhaitons maintenant formaliser le problème de la primalité sur tous les entiers non nuls. La chose évidente et bonne à faire est la suivante. D'abord on définit, pour t un entier non nul, le problème élémentaire $\mathscr{P}_t$ de la même manière que dans l'exemple 1.2, qui formalise la question « l'entier n de taille t est-il premier ? ». Le problème de la primalité est alors simplement la fonction qui à t associe $\mathscr{P}_t$. Rappelons que, non seulement $\mathscr{P}$ formalise le problème de la primalité, mais en plus il contient une information sur la manière dont les entiers sont codés — dans ce cas, en binaire.

Pourquoi choisir de définir les problèmes en deux temps, en commençant par les problèmes « élémentaires » ? Cela est motivé par le fait que la plupart

des grandeurs associées à un problème — en particulier celle qui nous intéressera le plus et que nous définirons par la suite, la complexité en requêtes — sont étudiées d'un point de vue asymptotique, en fonction de la taille des instances. Ainsi, si l'on considère un problème $\mathscr{P} : s \in S \mapsto \mathscr{P}_s$, la quantité étudiée sur $\mathscr{P}$ peut se modéliser par une fonction λ définie sur S. Á partir de là, on peut vouloir étudier le comportement de λ d'une manière ou d'une autre, le plus souvent donc en s'intéressant uniquement à la taille des instances, mais cela peut être un peu plus subtil. Concrètement, pour anticiper sur la suite, mentionnons le cas du problème défini par la définition 2.6, où le rôle de S est tenu par un ensemble de groupes, et de la proposition 2.27, qui énonce un résultat sur la complexité en requêtes de ce problème, non pas en fonction de la taille de ces groupes mais de leur rang[2].

Il faut préciser ce point en ce qui concerne l'usage des comparateurs asymptotiques standard que sont $\mathcal{O}$, Ω et Θ. Quand on écrira, pour un problème $\mathscr{P} : s \in S \mapsto \mathscr{P}_s$, quelque chose comme « $\lambda(s) = \mathcal{O}(\mu(s))$ », cela signifiera, sauf mention explicite du contraire, qu'il existe une constante $C > 0$ telle que pour presque tous les $s \in S$ — c'est-à-dire pour tous sauf un nombre fini d'entre eux — on ait $|\lambda(s)| \leq C\,|\mu(s)|$. Il ne s'agit donc pas, comme on pourrait le penser à tort, d'une inégalité vraie seulement lorsque $\mu(s)$ est suffisamment grand, ce qui pourrait s'écrire « $\lambda(s) = \mathcal{O}_{\mu(s)\to+\infty}(\mu(s))$ », mais bien d'une égalité vraie lorsque le problème élémentaire lui-même est « suffisamment grand ». Cette remarque s'applique en particulier à l'énoncé du théorème 3.16.

1.2 Modèle de calcul déterministe

Il est temps justement d'introduire les définitions qui vont nous permettre de parler de la complexité en requêtes. Reprenons notre exemple du problème de la primalité. La question que l'on se pose habituellement serait : « quel est le temps de calcul d'un algorithme capable de déterminer avec une probabilité satisfaisante si un entier est premier ? ». Mais nous ne nous intéresserons pas à la question générale de la complexité en temps dans ce mémoire. Nous allons plutôt nous intéresser à la **complexité en requêtes**. Considérons le problème de la primalité et commençons par décrire le fonctionnement d'un algorithme déterministe. On représente en général les algorithmes déterministes par des arbres de décision comme celui de la figure 1.1.

D'abord, on peut remarquer que, comme l'algorithme va peut-être demander la valeur de $x(4)$, l'entier passé en entrée de l'algorithme devrait être de longueur au moins 5. En fait, un arbre de décision n'est habilité à résoudre qu'un problème élémentaire. Disons donc que notre arbre de décision est associé au problème élémentaire de la primalité pour les entiers de longueur 6, c'est-à-dire ceux compris entre 32 et 63 inclus. Cet algorithme a

[2] Voir la définition 2.21.

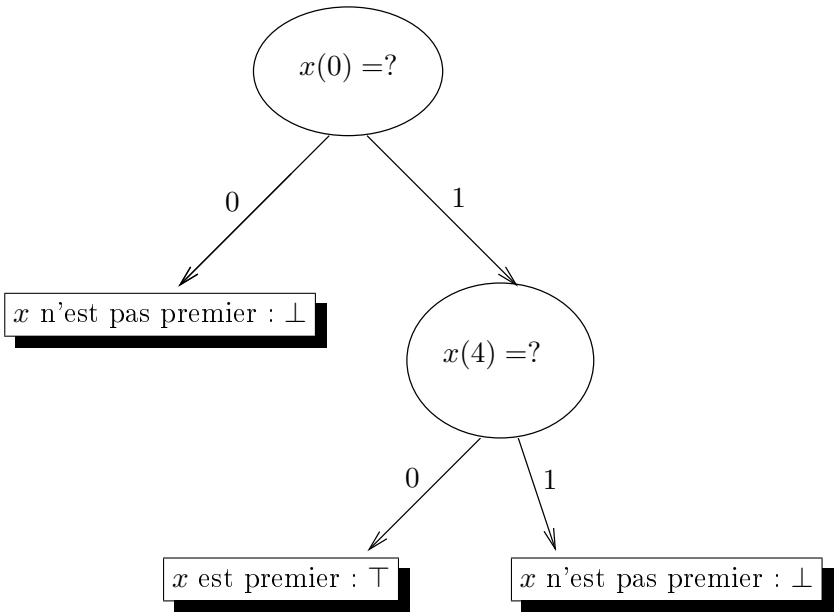

FIG. 1.1 – Un arbre de décision pour le problème de la primalité

toujours raison quand on lui donne en entrée un nombre pair, mais ensuite il raconte un peu n'importe quoi. Par exemple, pour 37, qui s'écrit 100101 en binaire, il a raison ; mais pour 39, qui s'écrit 100111 en binaire, il se trompe.

La complexité en requêtes d'un arbre de décision est sa hauteur moins 1, en fait le nombre de questions que l'algorithme va poser dans le pire cas. Dans notre exemple, la complexité en requêtes est de 2, complexité atteinte dès que l'entier passé en entrée est impair.

Les arbres de décision conviennent parfaitement pour formaliser la notion de complexité en requêtes dans le cas déterministe, mais comme nous voulons le faire aussi pour le calcul probabiliste et le calcul quantique, nous allons donner une formulation plus alambiquée, collant au formalisme hilbertien de la mécanique quantique.

On s'intéresse à l'évolution de l'**état** de l'environnement de l'algorithme, c'est-à-dire l'ensemble des valeurs des variables. L'état de l'environnement est décrit par un vecteur de base d'un espace vectoriel de dimension finie. La dimension de cet espace représente donc le nombre d'états possibles de l'environnement. On peut découper le fonctionnement de l'algorithme en deux phases qui se succèdent alternativement, appelons-les les phases 1 et 2. Dans la phase 1, l'algorithme ne pose aucune question à la boîte noire mais effectue de son côté autant de calculs qu'il le souhaite. Dans la phase 2, représentée par la figure 1.2, il envoie une question à la boîte noire, sous la forme d'un élément i de I s'il traite le problème élémentaire $(I, J, R, \mathscr{S}, f)$, et reçoit une réponse $j \in J$, qu'il lui est alors loisible de stocker dans une

variable prévue à cet effet. Puis on reprend la phase 1, etc., jusqu'à ce que l'algorithme décide qu'il a terminé et propose une réponse sous la forme d'un élément de R, dans une variable dédiée.

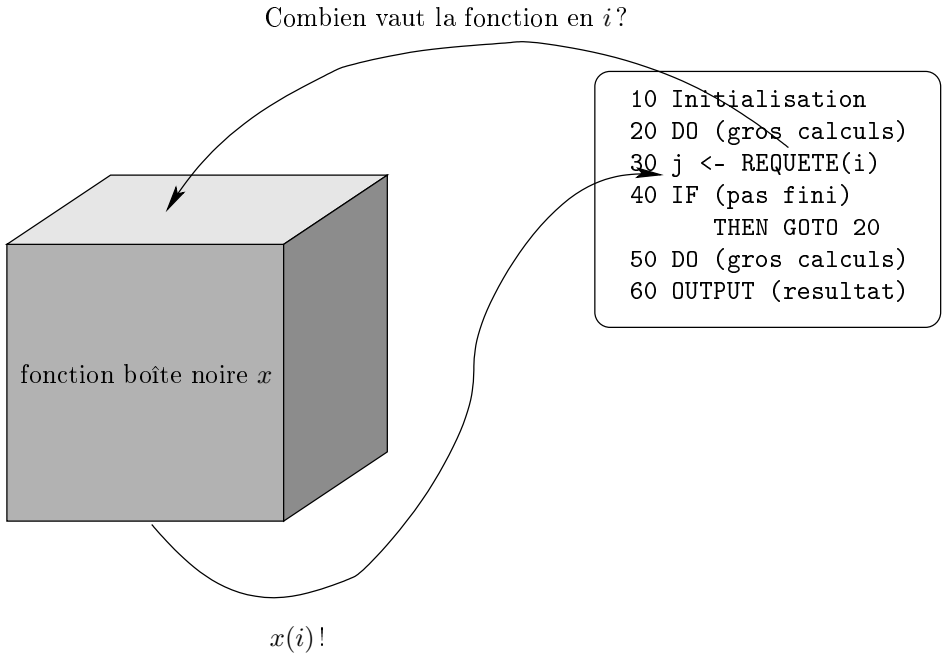

FIG. 1.2 – L'algorithme dialogue avec la boîte noire

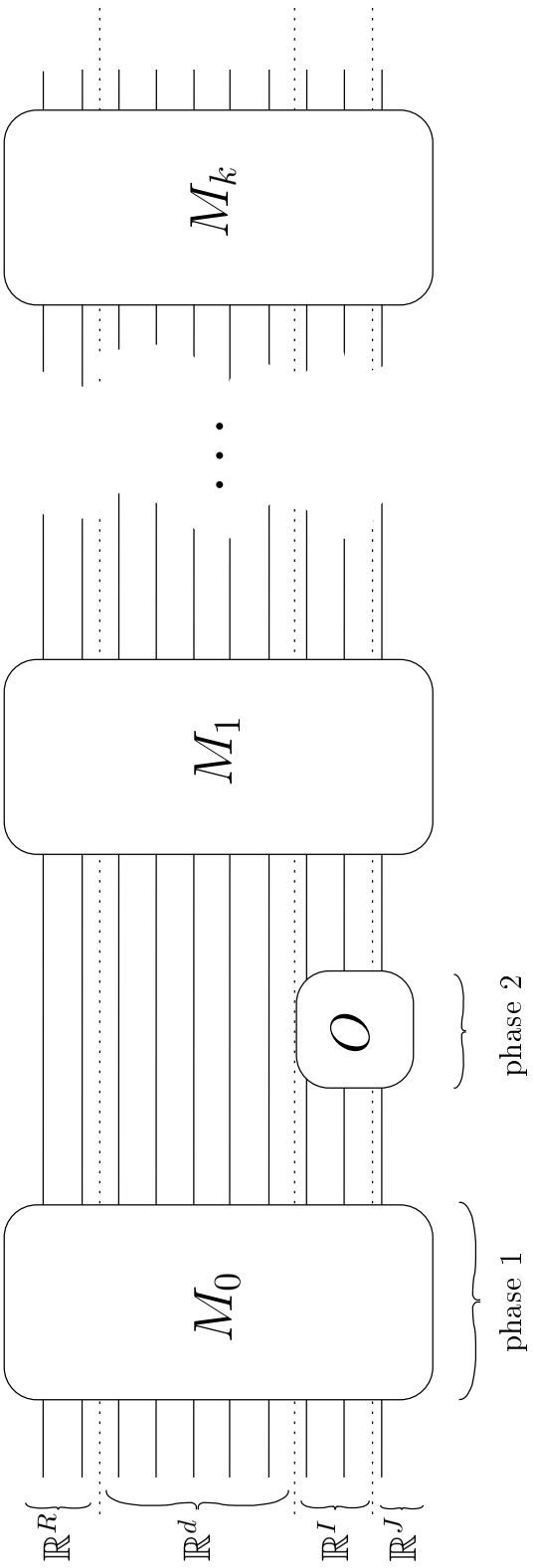

FIG. 1.3 – Un algorithme déterministe

Que se passe-t-il quand on juxtapose deux environnements ? Mettons qu'un algorithme utilise deux catégories de variables. Le premier ensemble de variables peut être dans k différents états globaux : son état est codé par un des vecteurs $e_1, \ldots, e_k$ qui forment la base d'un espace E. De même le second ensemble de variables peut être dans l différents états globaux et son état est donc codé par un des vecteurs $f_1, \ldots, f_l$ qui forment la base d'un espace F. Lorsque ces deux ensembles de variables sont réunis, on obtient un système plus complexe dont l'état est décrit comme un couple (e_i, f_j). Or quel est l'espace dont la base est naturellement le produit cartésien des bases de E et de F ? Il s'agit du produit tensoriel $E \otimes F$.

Au lecteur non familier de cette notion de produit tensoriel, nous offrons la brève parenthèse suivante, qui a pour simple but de faire passer le minimum nécessaire à la compréhension de l'usage que nous allons en faire. Le lecteur qui n'a pas besoin de ces éclaircissements pourra sauter la définition et l'exemple qui suivent.

Définition 1.7
Étant donné deux $\mathbb{K}$-espaces vectoriels E et F munis respectivement des bases $(e_i)_{i\in I}$ et $(f_j)_{j\in J}$, leur produit tensoriel $E \otimes F$ est par définition un $\mathbb{K}$-espace vectoriel, qui vient conjointement avec une application bilinéaire $\cdot \otimes \cdot : E \times F \to E \otimes F$ telle que $(e_i \otimes f_j)_{i\in I, j\in J}$ forme une base de $E \otimes F$.

Si E et F sont munis de produits scalaires $\langle \cdot, \cdot \rangle$, $E \otimes F$ possède également un produit scalaire défini par $\langle e_i \otimes f_j, e_{i'} \otimes f_{j'} \rangle = \langle e_i, e_{i'} \rangle \langle f_j, f_{j'} \rangle$.

Si E et F sont munis d'une structure d'algèbre, on en définit une sur $E \otimes F$ par $(e_i \otimes f_j) \cdot (e_{i'} \otimes f_{j'}) = (e_i \cdot e_{i'}) \otimes (f_j \cdot f_{j'})$.

De manière générale, quelles que soient les structures imposées à E et F, leur produit tensoriel $E \otimes F$ est tel que pour tout autre espace G, les morphismes linéaires de $E \otimes F$ dans G s'identifient aux morphismes bilinéaires de $E \times F$ dans G.

Exemple 1.8
Un exemple courant et que nous allons utiliser très prochainement est celui des algèbres de matrices. Considérons $\mathscr{M}_n(\mathbb{K})$ et $\mathscr{M}_m(\mathbb{K})$. La question est : $\mathscr{M}_n(\mathbb{K}) \otimes \mathscr{M}_m(\mathbb{K})$ est-elle isomorphe à une algèbre connue que l'on pourrait exprimer plus simplement ? Pour y répondre, il faut décrire sa structure d'algèbre. Pour commencer, en tant que $\mathbb{K}$-espaces vectoriels, $\mathscr{M}_n(\mathbb{K})$ et $\mathscr{M}_m(\mathbb{K})$ sont munis d'une base naturelle, celle des $E_{i,j}$, les matrices composées de 0 à l'exclusion d'un 1 dans la case (i, j). Les $E_{i,j} \otimes E_{k,l}$, pour $i, j \in \{1, \ldots, n\}$ et $k, l \in \{1, \ldots, m\}$, forment donc une base de $\mathscr{M}_n(\mathbb{K}) \otimes \mathscr{M}_m(\mathbb{K})$, qui est par conséquent isomorphe à $\mathbb{K}^{nm}$ en tant que $\mathbb{K}$-espace vectoriel.

Regardons maintenant la structure d'algèbre. Par définition,

$$\begin{array}{rcl}(E_{i,j}\otimes E_{k,l})\left(E_{i',j'}\otimes E_{k',l'}\right) & = & \left(E_{i,j}E_{i',j'}\right)\otimes\left(E_{k,l}E_{k',l'}\right)\\ & = & \left(\delta_{ji'}E_{i,j'}\right)\otimes\left(\delta_{l,k'}E_{k,l'}\right)\\ & = & \delta_{(j,l)(i',k')}\left(E_{i,j'}\otimes E_{k,l'}\right)\\ (E_{i,j}\otimes E_{k,l})\left(E_{i',j'}\otimes E_{k',l'}\right) & = & \delta_{(mj+l)(mi'+k')}\left(E_{i,j'}\otimes E_{k,l'}\right).\end{array}$$

La dernière égalité tient simplement au fait que, comme l et k' sont confinés à $\{1,\ldots,m\}$, les couples (j,l) et (i',k') sont égaux si et seulement si $mj+l=mi'+k'$. Ceci nous mène à la remarque suivante : l'application linéaire λ définie par

$$\lambda:\left(\begin{array}{rcl}\mathscr{M}_n(\mathbb{K})\otimes\mathscr{M}_m(\mathbb{K}) & \to & \mathscr{M}_{nm}(\mathbb{K})\\ E_{i,j}\otimes E_{k,l} & \mapsto & E_{m(i-1)+k,m(j-1)+l}\end{array}\right)$$

est un isomorphisme d'algèbres. La vérification en est simple, il suffit de vérifier que λ est injective, ce qui est évident par construction, et qu'elle respecte la multiplication interne d'algèbre. Cela est vrai car, dans $\mathscr{M}_{nm}(\mathbb{K})$, on a

$$\begin{array}{rl}& E_{m(i-1)+k,m(j-1)+l}E_{m(i'-1)+k',m(j'-1)+l'}\\ = & \delta_{(m(j-1)+l)(m(i'-1)+k')}E_{m(i-1)+k,m(j'-1)+l'}\end{array}.$$

On en déduit ceci :

$$\lambda\left[(E_{i,j}\otimes E_{k,l})\left(E_{i',j'}\otimes E_{k',l'}\right)\right]=\lambda(E_{i,j}\otimes E_{k,l})\lambda\left(E_{i',j'}\otimes E_{k',l'}\right).$$

Ainsi le produit tensoriel de deux matrices A et B s'obtient en remplaçant chaque case (i,j) de A par la matrice $A[i,j].B$. Par exemple, on a

$$\begin{pmatrix}1&2\\0&-1\end{pmatrix}\otimes\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}=\begin{pmatrix}1&2&3&2&4&6\\4&5&6&8&10&12\\7&8&9&14&16&18\\0&0&0&-1&-2&-3\\0&0&0&-4&-5&-6\\0&0&0&-7&-8&-9\end{pmatrix}.$$

On peut définir aussi plus généralement le produit tensoriel de deux matrices non nécessairement carrées. Pour $E_{i,j}\in\mathscr{M}_{n,m}$ et $E_{k,l}\in\mathscr{M}_{p,q}$, on pose $E_{i,j}\otimes E_{k,l}=E_{pi+k,qj+l}$, ce qui revient encore à dire que pour multiplier A et B on remplace la case (i,j) de A par la matrice $A[i,j]B$.

Revenons à présent à nos algorithmes déterministes. Dans la figure 1.3, chaque « fil » représente ce que l'on appelle un **registre**, c'est-à-dire informellement une variable interne du programme, et formellement un espace vectoriel. Comme par définition tous les états sont des vecteurs de base, on

a la propriété que tout état global s'écrit comme produit tensoriel d'états sur chaque registre — ce qui ne sera plus vrai ni en calcul probabiliste ni en calcul quantique.

Une transformation déterministe en phase 1 change un état en n'importe quel autre état. Traduisons cela dans notre formalisme : il s'agit d'une application qui laisse la base canonique de l'espace vectoriel E stable. On peut donc l'étendre linéairement, et ce de manière unique, à tout l'espace E — cela semble présenter peu d'intérêt pour l'instant mais en aura plus tard pour les autres procédés de calcul. Au final, nous pouvons définir une transformation déterministe comme une application linéaire M de E dans E laissant la base canonique de E stable. Autrement dit, si l'on écrit la matrice de M dans la base E, elle est composée de « 1 » et de « 0 » et comprend exactement un « 1 » par colonne ; par exemple convient

$$M = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Comme nous ferons référence plusieurs fois à des matrices de cette forme particulière, nous allons leur donner un nom :

Définition 1.9
Une telle matrice, composée de « 0 » et de « 1 » et comprenant exactement un « 1 » par colonne, est appelée une **matrice déterministe**.

Par exemple, les matrices de permutations sont des cas particuliers de matrices déterministes, qui correspondent aux transformations réversibles. Reprenons ; une itération de la phase 1 peut se résumer en « on applique au vecteur courant une matrice M de la forme susdite[3] indépendante de la fonction contenue dans la boîte noire », fonction que nous noterons x.

La phase 2, représentée en figure 1.2 consiste à demander combien vaut $x(i)$ pour un certain i. En pratique, l'algorithme réserve deux variables, i et j, et au moment où il décide de lancer la phase 2, la valeur de j devient $x(i)$.

Il est temps d'introduire de nouvelles notations. Nous marquerons par $|i)$ le vecteur associé à l'état i de l'environnement, et la simple juxtaposition notera le produit tensoriel :

$$|i)\,|j) =_{\text{def}} |i) \otimes |j)$$

Ainsi on peut définir l'application linéaire correspondant à la phase 2 ainsi sur la base canonique :

$$O'_x\,|z)\,|i)\,|j) = |z)\,|i)\,|x(i))\,.$$

[3] C'est-à-dire composée uniquement de 1 et de 0 et contenant exactement un 1 par colonne.

z représente tout simplement l'état de la partie de l'environnement qui n'est pas concerné par la requête, ce qui fait que l'on peut également écrire

$$O'_x = \mathbf{I} \otimes O_x,$$

où O_x est simplement définie par $O_x\,|i)\,|j) = |i)\,|x(i))$. Par exemple, si $I = [4]$, $J = [7]$, que l'un et l'autre sont représentés par ordre croissant et que l'on considère la fonction x définie par $x(0) = 6$, $x(1) = 6$, $x(2) = 1$ et $x(3) = 4$, on peut écrire

$$O_x = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

Exemple 1.10
Réexaminons l'arbre de décision de la figure 1.1. On peut le traduire informellement en programme séquentiel sous la forme suivante.

```
* initialisation *
resultat <- 0
aux <- 0
i <- 0
j <- 0

* phase 1 *

* phase 2 *
j <- x(i)

* phase 1 *
si j = 0 alors aux <- 1
i <- 4

* phase 2 *
j <- x(i)

* phase 1 *
si aux = 0 et j = 0 alors resultat <- 1

*fin*
renvoyer resultat
```

Les variables `resultat`, `aux` et `j` peuvent prendre chacune deux valeurs ; la variable `i` quant à elle peut prendre six valeurs différentes, même si dans ce programme elle n'en prend effectivement que deux : rappelons en effet que ce programme est censé résoudre le problème élémentaire de la primalité des entiers de taille six. La valeur de l'environnement est donc représentée par un vecteur de la base canonique d'un espace à $6 \times 2 \times 2 \times 2 = 48$ dimensions. Mettons que l'on écrive toujours les registres dans le même ordre : `i`, `j`, `aux` puis `resultat`, ordre que l'on a choisi pour la commodité de l'écriture des matrices sous forme de produits tensoriels. Alors les différentes étapes se formalisent sous forme matricielle ainsi :

`* initialisation *` On part du vecteur

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

puis on applique successivement les transformations suivantes :

`* phase 1 *`

$$\mathbf{I}_{48}$$

`* phase 2 *`

$$O_x \otimes \mathbf{I}_4$$

`* phase 1 *`

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \otimes \mathbf{I}_2$$

`* phase 2 *`

$$O_x \otimes \mathbf{I}_4$$

`* phase 1 *`

$$\mathbf{I}_6 \otimes \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

`* fin *` Si le vecteur obtenu est de la forme $|z) \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ l'algorithme rejette l'entier — c'est-à-dire qu'il annonce qu'il est composé —, sinon c'est qu'il est de la forme $|z) \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ et alors l'algorithme l'accepte.

Outre les registres servant à la communication avec la boîte noire, un algorithme doit avoir un registre désigné pour accueillir la réponse finale qu'il va donner. On dit ainsi qu'un algorithme $\mathscr{A}$ donne la réponse $r \in R$ sur l'entrée $x \in J^I$ lorsque son état final s'écrit sous la forme $|z) \otimes |r)$, où z est quelconque.

Définition 1.11
Étant donné un problème élémentaire $\mathscr{P} = (I, J, R, \mathscr{S}, f)$, un algorithme déterministe $\mathscr{A}$ pour $\mathscr{P}$ est la donnée d'un entier d et d'une suite $(M_0, \ldots, M_T)$ d'endomorphismes de $\mathbb{R}^R \otimes \mathbb{R}^d \otimes \mathbb{R}^I \otimes \mathbb{R}^J$ donnés par des matrices déterministes[4] dans la base canonique.

On dit que $\mathscr{A}$ résout le problème élémentaire $\mathscr{P}$ si, pour tout $x \in \mathscr{S}$,

$$\pi_R M_T O_x M_{T-1} O_x \cdots O_x M_0 \, |0) = |f(x)) \, ,$$

où

- $|0) = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$,
- *O_x est l'endomorphisme de $\mathbb{R}^I \otimes \mathbb{R}^J$ défini par $O_x \, |i) \, |j) = |i) \, |x(i))$,*
- *$(|r))_{r \in R}$ est la base canonique de de $\mathbb{R}^R$ et*
- *π_R est le projecteur perpendiculaire de $\mathbb{R}^R \otimes \mathbb{R}^d \otimes \mathbb{R}^I \otimes \mathbb{R}^J$ sur $\mathbb{R}^R$, c'est-à-dire défini par $\pi_R \, |r) \, |z) \, |i) \, |j) = |r)$.*

On dit qu'un algorithme $\mathscr{A}$ résout un problème $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ lorsque, pour tout $x \in \mathscr{S}$, $\mathscr{A}$ répond $f(x)$ sur l'entrée x.

La **complexité en requêtes déterministe** *d'un problème élémentaire $\mathscr{P}$ est le minimum des complexités en requêtes des algorithmes déterministes résolvant $\mathscr{P}$.*

Exemple 1.12
« Vous avez douze billes d'apparence identique, dont l'une a toutefois un poids différent des onze autres. On vous fournit une balance à deux plateaux, sans graduation ni poids. De combien de pesées avez-vous besoin pour déterminer la bille particulière ? »

On peut formaliser ce problème bien connu comme un calcul de complexité en requêtes déterministe. Ce n'est pourtant pas immédiat : ici I serait par exemple l'ensemble des couples de parties disjointes de $\{1, \ldots, 12\}$.

[4] Voir la définition 1.9

1.3 Modèle de calcul probabiliste

Nous être apesantis sur le modèle de calcul déterministe, en avoir donné une description aussi peu naturelle, va nous permettre d'exposer plus rapidement les modèles suivants, à commencer par le modèle de calcul probabiliste. En effet, le principe est simplement que l'on remplace les matrices du modèle déterministe par des matrices stochastiques. Voyons cela.

L'état initial de l'environnement est toujours fixé arbitrairement à une valeur notée $|0)$, qui est le premier vecteur de la base canonique. La succession des phases 1 et 2 est respectée de la même manière, et la phase 2 consiste toujours à poser une question à la boîte noire. Mais maintenant, en phase 1, l'algorithme peut procéder aléatoirement. Quand on écrit un programme dans notre langage de programmation préféré, on invoque le grand aléa à chaque fois qu'on le souhaite, mais bien entendu comme tous ces tirages sont indépendants, on pourrait très bien les faire tous d'un coup au début, avant de commencer les calculs, du moins si l'on savait de combien de tirages on va avoir besoin. Ou alors simplement au début de chaque phase 1, c'est-à-dire que chaque phase 1 consiste à appliquer une transformation déterministe T_ω avec probabilité $p(\omega)$, parmi un ensemble prédéterminé de transformations $\{T_\omega, \omega \in \Omega\}$. Si l'on tient compte que les $p(\omega)$ sont positifs et que $\sum_{\omega\in\Omega} p(\omega) = 1$, on s'aperçoit que cela revient en réalité à dire que l'on applique une matrice stochastique, pour la définition suivante de stochastique, que nous précisons car elle peut varier légèrement selon les sources :

Définition 1.13
Une **matrice stochastique** *est une matrice réelle à coefficients positifs et telle que la somme des coefficients sur chaque colonne vaut* 1.

Peut-être est-il utile de justifier que cette définition d'un algorithme probabiliste correspond bien à celle, plus intuitive, que l'on peut donner sous forme d'arbres de décision. Pour cela, commençons par faire la remarque suivante.

Fait 1.14
Une matrice A est stochastique si et seulement si elle s'écrit sous la forme $A = \sum_{i\in I} p(i)M_i$, où les M_i sont des matrices déterministes[5] *et p est une distribution de probabilité sur I.*

Preuve : Par récurrence sur le nombre d'entrées non nulles de la matrice A de taille $n \times k$. Le minimum de cases non nulles pour une matrice stochastique de taille $n \times k$ est k, et dans ce cas il s'agit d'une matrice déterministe.

[5] Voir la définition 1.9.

Si le nombre d'entrées non nulles de A est strictement supérieur à k, notont m le minimum des entrées non nulles de A ; $m < 1$. Dans chaque colonne j, on choisit arbitrairement une ligne i_j telle que $A_{i_j,j} \geq m$. Soit M la matrice de taille $n \times k$ définie par

$$M_{i,j} = \begin{cases} 1 & \text{si } i = i_j \\ 0 & \text{sinon} \end{cases} .$$

On a, et c'est peu étonnant, $A = mM + (1-m)\frac{1}{1-m}(A - mM)$. Or M est une matrice déterministe et $\frac{1}{1-m}(A - mM)$ une matrice stochastique dont le nombre d'entrées non nulles est strictement inférieur à celui de A, ce qui clôt la récurrence. □

Ainsi, appliquer une transformation stochastique $A = \sum_{i \in I} p(i) M_i$, c'est comme appliquer avec probabilité $p(i)$ la transformation déterministe M_i. On retrouve bien la notion habituelle d'algorithme probabiliste : à chaque nœud de l'arbre on peut raccorder autant de branches que l'on veut, chacune menant à un sous-arbre de décision probabiliste, et chaque branche étant choisie lors du calcul avec une certaine probabilité fixée. Sur la figure 1.4, on a representé un choix aléatoire par un hexagone, et les probabilités correspondantes sont indiquées sur les arêtes qui en partent.

L'état de l'environnement est donc maintenant représenté par un vecteur stochastique — qui n'est rien d'autre qu'une matrice stochastique n'ayant qu'une colonne. La grande nouveauté par rapport au cas déterministe est que tous les états ne s'écrivent plus nécessairement comme des produits tensoriels. Par exemple, l'état

$$\frac{1}{2}(|0)\,|0) + |1)\,|1))$$

ne peut pas s'écrire sous la forme $|\varphi)\,|\psi)$. C'est ce que l'on appelle dans le cas quantique un état enchevêtré, mais cette appellation est impropre dans le cas probabiliste, nous verrons pourquoi dans la section 1.4. En l'occurrence, cela traduit simplement le fait qu'il peut exister des événements A et B, se produisant chacun avec probabilité $\frac{1}{2}$, et totalement dépendants, en ce sens que A se produit si et seulement si B se produit.

Ainsi, nous pouvons reprendre quasiment mot pour mot la définition d'un algorithme déterministe.

Définition 1.15
Étant donné un problème élémentaire $\mathscr{P} = (I, J, R, \mathscr{S}, f)$, un algorithme probabiliste $\mathscr{A}$ pour $\mathscr{P}$ est la donnée, d'une part d'un entier d, et d'autre part d'une suite $(M_0, \ldots, M_T)$ d'endomorphismes stochastiques de $\mathbb{R}^R \otimes \mathbb{R}^d \otimes \mathbb{R}^I \otimes \mathbb{R}^J$.

On dit que $\mathscr{A}$ résout le problème élémentaire $\mathscr{P}$ avec une probabilité d'erreur au plus ε si, pour tout $x \in \mathscr{S}$,

$$\left\|\pi_{f(x)} M_T O_x M_{T-1} O_x \cdots O_x M_0 \left|0\right)\right\|_1 \geq 1 - \varepsilon,$$

où

- $\left|0\right) = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$,
- *O_x est l'endomorphisme de $\mathbb{R}^I \otimes \mathbb{R}^J$ défini par $O_x \left|i\right) \left|j\right) = \left|i\right) \left|x(i)\right)$,*
- *π_r, pour $r \in R$, est le projecteur perpendiculaire de $\mathbb{R}^R \otimes \mathbb{R}^d \otimes \mathbb{R}^I \otimes \mathbb{R}^J$ sur $\left|r\right) \otimes \mathbb{R}^d \otimes \mathbb{R}^I \otimes \mathbb{R}^J$, c'est-à-dire défini par $\pi_r \left|r'\right) \left|z\right) \left|i\right) \left|j\right) = \delta_{r,r'} \left|r\right) \left|z\right) \left|i\right) \left|j\right)$, et*
- *$(\left|r\right))_{r \in R}$ est la base canonique de de $\mathbb{R}^R$.*

La valeur par défaut de ε, quand celle-ci n'est pas précisée, est de $\frac{1}{3}$. La **probabilité d'erreur** *de $\mathscr{A}$ est le plus petit ε tels que $\mathscr{A}$ résout $\mathscr{P}$ avec une probabilité d'erreur au plus ε.*

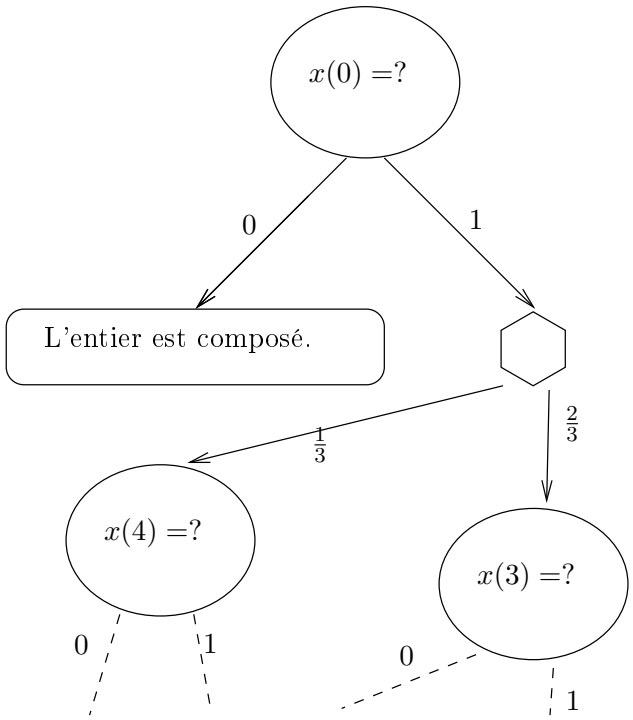

FIG. 1.4 – Un bout d'arbre de décision probabiliste pour le problème de la primalité

Quand on fixe à $x \in \mathscr{S}$ la fonction boîte noire, la probabilité qu'un chemin particulier, compatible avec les choix faits quand x est la fonction

boîte noire, soit suivi lors du calcul, est le produit des probabilités rencontrées le long des arêtes du chemin. On pourrait cependant tout aussi bien décider que toutes les bifurcations aléatoires de l'arbre de décision sont décidées avant même que l'algorithme ne débute. En ce sens un algorithme probabiliste peut être présenté tout simplement comme une distribution de probabilité sur des algorithmes déterministes. Nous préciserons cette affirmation dans le fait 1.19.

Notons $\mathbb{P}_{\mathscr{A}}(x, A)$ la probabilité que l'algorithme $\mathscr{A}$ effectue la suite de requêtes $A \in I^*$ lorsque la fonction boîte noire est x. Cette définition informelle est sans doute suffisante, mais par acquit de conscience en voici une plus rigoureuse, reposant sur le fait que nous voulons définir par récurrence $\mathbb{P}_{\mathscr{A}}(x, A.(i))$ comme étant $\mathbb{P}_{\mathscr{A}}(x, A)$ multiplié par la probabilité que $\mathscr{A}$ effectue la requête i sachant qu'il a auparavant effectué la suite de requêtes A.

Définition 1.16
On part de la définition 1.15 d'un algorithme probabiliste. $\mathscr{P}$ est un quintuplet $(I, J, R, \mathscr{S}, f)$ et $\mathscr{A}$ est défini par la suite $(M_0, \ldots, M_T)$ d'endomorphismes stochastiques de $\mathbb{R}^R \otimes \mathbb{R}^d \otimes \mathbb{R}^I \otimes \mathbb{R}^J$. Notons π_i la projection orthogonale sur $\mathbb{R}^R \otimes \mathbb{R}^d \otimes |i) \otimes \mathbb{R}^J$. Alors

$$\mathbb{P}_{\mathscr{A}}(x, (i_0, \ldots, i_{r-1})) = \left\| \pi_{i_{r-1}} M_{r-1} O_x \pi_{i_{r-2}} \cdots O_x \pi_{i_0} M_0 |0) \right\|_1$$

Cette définition se passe probablement de commentaire : simplement, avant chaque communication avec la boîte noire, la projection Π_i s'assure qu'il s'agisse bien de la requête i. On définit pareillement $\mathbb{P}^r_{\mathscr{A}}(x)$ comme étant la probabilité que l'algorithme $\mathscr{A}$ renvoie au final la réponse $r \in R$ quand la fonction boîte noire est x. Formellement, on a, en utilisant la notion π_r de la définition 1.15,

Définition 1.17

$$\mathbb{P}^r_{\mathscr{A}}(x) = \left\| \pi_r M_T O_x M_{T-1} O_x \cdots O_x M_0 |0) \right\|_1 .$$

Il nous faut également préciser une notion pour éviter les malentendus, celle d'équivalence entre algorithmes. Intuitivement, nous dirons que deux algorithmes sont équivalents si, quand on cache leur fonctionnement interne, que l'on ne se donne accès qu'à leurs entrées et sorties, ils deviennent tout à fait indistinguables l'un de l'autre, du moins quand la boîte noire est du type autorisé, c'est-à-dire quand $x \in \mathscr{S}$. Nous insistons sur ce point : si x n'est pas un élément de $\mathscr{S}$, les deux algorithmes, tout en étant équivalents, pourraient malgré cela très bien adopter des comportements tout à fait différents.

Définition 1.18
Les algorithmes $\mathscr{A}$ et $\mathscr{A}'$, de complexité en requêtes T, pour le problème $\mathscr{P} = (I, J, R, \mathscr{S}, f)$, sont **équivalents** *si, pour tout $x \in \mathscr{S}$,*

- *pour tout* $A \in I^*$ *de taille au plus* T, $\mathbb{P}_{\mathscr{A}}(x, A) = \mathbb{P}_{\mathscr{A}'}(x, A)$, *et*
- $\mathbb{P}^r_{\mathscr{A}}(x) = \mathbb{P}^r_{\mathscr{A}'}(x)$.

Nous pouvons maintenant préciser ce que nous disions, à propos de l'identification des algorithmes probabilistes avec les distributions de probabilité sur les algorithmes déterministes.

Fait 1.19
Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ *un problème élémentaire. Pour tout algorithme probabiliste* $\mathscr{A}$ *de complexité en requêtes* T, *il existe une distribution de probabilité* $p : \Omega \to [0; 1]$, *où* Ω *est un ensemble fini d'algorithmes déterministes de complexité en requêtes* T, *qui est équivalente à* $\mathscr{A}$ *en ce sens que l'on a, pour tout* $x \in \mathscr{S}$,

- *pour tout* $A \in I^*$ *de taille au plus* T, $\mathbb{P}_{\mathscr{A}}(x, A) = \sum_{\mathscr{D} \in \Omega} p(\mathscr{D}) \mathbb{P}_{\mathscr{D}}(x, A)$, *et*
- $\mathbb{P}^r_{\mathscr{A}}(x) = \sum_{\mathscr{D} \in \Omega} p(\mathscr{D}) \mathbb{P}^r_{\mathscr{D}}(x)$.

Nous exploitons le fait que les algorithmes déterministes sont des algorithmes probabilistes en utilisant les notations $\mathbb{P}_{\mathscr{D}}(x, A)$ et $\mathbb{P}^r_{\mathscr{D}}(x)$, mais il ne faut pas perdre de vue que ces probabilités valent toujours 0 ou 1 dans le cas d'algorithmes déterministes. Quant à la preuve de ce fait, sans entrer dans le détail, c'est comme nous l'avons dit : les choix aléatoires présents dans un algorithme probabiliste peuvent aussi bien être fixés une fois pour toutes au début du « programme », le reste de l'algorithme étant déterministe — mais dépendant de l'aléa choisi.

Cette remarque est particulièrement importante pour le principe de Yao, que nous verrons dans la section 4.2.2. Disons dès à présent qu'il n'existe pas de propriété équivalente pour les algorithmes quantiques, que nous allons définir dans un instant. Les algorithmes quantiques ne sont pas des « superpositions quantiques » d'algorithmes déterministes, quel que soit le sens que l'on puisse donner à cette phrase ; en tout cas personne ne sait le faire. Il en découle que le principe de Yao n'a, lui non plus, pas d'équivalent quantique connu, même si des tentatives ont été faites dans ce sens — voir par exemple [dGR02].

1.4 Modèle de calcul quantique

Nous ne nous préoccuperons pas de justifier le paradigme du calcul quantique à partir de la célébrissime théorie physique du même nom. Toutefois, si le lecteur désirait en savoir plus sur cet aspect des choses, nous pourrions lui recommander *An Introduction to Quantum Computing for Non-Physicits* d'Eleanor Rieffel et Wolfgang Polak [RP98]. Pour avoir un aperçu de l'historique du modèle de calcul quantique, nous pouvons citer Feynman [Fey82] qui est considéré comme le premier à avoir indiqué la possibilité théorique

d'augmenter la puissance de calcul des ordinateurs à l'aide des propriétés quantiques de la matière, et Deutsch [Deu85] qui le premier proposa un modèle théorique effectif de calcul quantique.

Par rapport au point de vue formel adopté dans les définitions 1.11 et 1.15, il est très simple d'introduire la notion d'algorithme quantique. Toutes les transformations doivent être non plus stochastiques mais unitaires. Rappelons donc s'il en était besoin ce qu'est une matrice unitaire.

Définition 1.20
Une matrice carrée U à coefficients complexes est dite **unitaire** *si $U^*U = I$, où $U^* = {}^t\bar{U}$ est la matrice adjointe de U.*

Comme les transformations du système sont maintenant unitaires, l'état du système n'est plus décrit par un vecteur stochastique, mais par un vecteur unitaire, c'est-à-dire un vecteur à coefficients complexes — et non plus nécessairement réels positifs — dont la norme 2 vaut 1. Pour marquer cette différence, et retrouver les notations standard de la mécanique et de l'informatique quantiques, nous n'utiliserons plus la notation $|\cdot)$, qui sera réservée aux vecteurs stochastiques, mais plutôt $|\cdot\rangle$.

Comme il s'agit d'une notion bien connue de tous, nous n'avons pas pris la peine dans les sections précédentes de définir le bit, mais il nous faut préciser ce qu'est son équivalent quantique, le **qubit**, même si nous n'utiliserons que rarement ce terme. Étant donné la façon dont on a défini les algorithmes déterministes et probabilistes, on comprend qu'un bit, déterministe ou probabiliste, est representé par un vecteur vivant dans un espace vectoriel réel de dimension 2. Ajouter un bit à l'espace de travail se traduit dans cette description par le doublement de la dimension de l'espace de travail, puisque techniquement on prend le produit tensoriel avec un espace de dimension 2. De manière générale, un espace de dimension d décrit un système portant $\log_2(d)$ bits. Il n'en va pas autrement pour les qubits, si ce n'est que l'on considère non plus des espaces réels, mais des espaces complexes. Quand un bit probabiliste est un vecteur stochastique de $\mathbb{R}^2$, un qubit est un vecteur unitaire de $\mathbb{C}^2$. En général, la première réaction, et la plus naturelle, est de penser qu'un qubit « contient plus d'information » qu'un bit car il « vit dans un espace plus gros ». C'est faux. Un qubit ne code pas plus d'information qu'un bit probabiliste, de même qu'un bit probabiliste ne code pas plus d'information qu'un bit déterministe. Il s'agit là d'un résultat que l'on peut énoncer et démontrer tout à fait rigoureusement, et qui est connu sous le nom de « borne de Holevo » ; on peut par exemple en trouver un énoncé et une démonstration sous la forme du théorème 12.1 de [NC00]. Nous le verrons, cela n'empêche cependant pas les algorithmes quantiques de résoudre certains problèmes en effectuant significativement moins de requêtes que les algorithmes probabilistes, de même que pour certains problèmes les algorithmes probabilistes sont significativement plus efficaces que les algorithmes déterministes – de manière toutefois moins drastique.

Le seul petit problème avec la règle que nous nous sommes fixée de n'appliquer que des transformations unitaires est que la matrice O_x, définie par $O_x |i\rangle |j\rangle = |i\rangle |x(i)\rangle$, ne l'est pas nécessairement, unitaire. En effet, la première propriété d'une matrice unitaire U est d'être inversible, et O_x ne l'est que si x est injective. Essentiellement, n'importe quelle convention raisonnable fait l'affaire. Á l'heure actuelle, il n'existe pas de système de calcul quantique suffisamment perfectionné pour qu'une définition naturelle de O_x se détache particulièrement. Nous allons pour notre part adopter la définition suivante :

$$O_x^\circ |i\rangle |j\rangle = |i\rangle |j \circ x(i)\rangle ,$$

où l'application $\begin{pmatrix} J \times J & \to & J \\ (j,j') & \mapsto & j \circ j' \end{pmatrix}$ est bijective en chacune de ses variables. La nature exacte de $\circ$ n'a pas d'importance, du moment que cette opération est indépendante de x ; à la rigueur, on pourrait même choisir un $\circ$ différent par requête. La plupart du temps, $J = [|J|]$, et on choisit pour $\circ$ l'addition modulo $|J|$, notée $\oplus$. Quoi qu'il en soit, $\circ$ n'ayant pas de réelle importance, on écrira la plupart du temps simplement O_x à la place de O_x°. Voici maintenant comment on défini un algorithme quantique :

Définition 1.21
Étant donné un problème élémentaire $\mathscr{P} = (I, J, R, \mathscr{S}, f)$, un algorithme quantique $\mathscr{A}$ pour $\mathscr{P}$ est la donnée d'un entier d et d'une suite $(M_0, \ldots, M_T)$ d'endomorphismes unitaires de $\mathbb{C}^R \otimes \mathbb{C}^d \otimes \mathbb{C}^I \otimes \mathbb{C}^J$.

On dit que $\mathscr{A}$ résout le problème élémentaire $\mathscr{P}$ avec une probabilité d'erreur au plus ε si, pour tout $x \in \mathscr{S}$,

$$\mathbb{P}_{\mathscr{A}}^{f(x)}(x) \geq 1 - \varepsilon,$$

où par définition $\mathbb{P}_{\mathscr{A}}^{r}(x)$ vaut $\|\pi_r M_T O_x M_{T-1} O_x \cdots O_x M_0 |0\rangle\|_2^2$, où

- $|0\rangle = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$,
- *O_x est l'endomorphisme de $\mathbb{R}^I \otimes \mathbb{R}^J$ défini par $O_x |i\rangle |j\rangle = |i\rangle |j \oplus x(i)\rangle$,*
- *$(|r\rangle)_{r \in R}$ est la base canonique de de $\mathbb{R}^R$ et*
- *π_r est le projecteur perpendiculaire de $\mathbb{R}^R \otimes \mathbb{R}^d \otimes \mathbb{R}^I \otimes \mathbb{R}^J$ sur $|r\rangle \otimes \mathbb{R}^d \otimes \mathbb{R}^I \otimes \mathbb{R}^J$, c'est-à-dire défini par*

$$\pi_r |s\rangle |z\rangle |i\rangle |j\rangle = \begin{cases} |r\rangle |z\rangle |i\rangle |j\rangle & \text{si } s = r \\ 0 & \text{sinon} \end{cases} .$$

De la même manière que dans la définition 1.15, la valeur par défaut de ε, quand celle-ci n'est pas précisée, est de $\frac{1}{3}$; la **probabilité d'erreur** *de $\mathscr{A}$*

est toujours le plus petit ε tels que $\mathscr{A}$ résout $\mathscr{P}$ avec une probabilité d'erreur au plus ε.

On pourra remarquer, et c'est même nécessaire de le faire, que dans la définition ci-dessus, l'application $\begin{pmatrix} R & \to & R \\ r & \mapsto & \mathbb{P}^r_{\mathscr{A}}(x) \end{pmatrix}$ est bien une distribution de probabilité sur R, du fait que $M_T O_x M_{T-1} O_x \cdots O_x M_0 |0\rangle$ est un vecteur unitaire et que les π_r sont des projecteurs orthogonaux deux-à-deux orthogonaux[6] tels que $\sum_{r\in R} \pi_r = 1$. En effet, sous ces conditions, en notant $|\psi_x\rangle = M_T O_x M_{T-1} O_x \cdots O_x M_0 |0\rangle$, on a

$$\begin{aligned} \sum_{r\in R} \mathbb{P}^r_{\mathscr{A}}(x) &= \sum_{r\in R} \|\pi_r |\psi_x\rangle\|_2^2 \\ &= \left\| \left(\sum_{r\in R} \pi_r \right) |\psi_x\rangle \right\|_2^2 \\ &= \|\mathbf{I}\psi_x\|_2^2 \\ \sum_{r\in R} \mathbb{P}^r_{\mathscr{A}}(x) &= 1 \end{aligned}$$

Il existe pour les problèmes élémentaires *binaires* une définition alternative des portes de requête, que nous noterons différemment, soit O'_x, par souci de clarté. O'_x agit seulement sur $\mathbb{C}^I$, de la façon suivante :

$$O'_x |i\rangle = (-1)^{x(i)} |i\rangle .$$

La première remarque qui peut être faite à propos de cette définition est qu'elle semble curieusement insuffisante. Considérons en effet $x \in \mathscr{S}$ et supposons qu'il existe $\bar{x}$ tel que pour tout $i \in I$, on ait $x(i) = 1 - \bar{x}(i)$. Si l'on considère un algorithme quantique où l'on remplace les portes O_x par les portes O'_x, on a alors, par linéarité des opérateurs, $\mathbb{P}^r_{\mathscr{A}}(x) = \mathbb{P}^r_{\mathscr{A}}(\bar{x})$. Autrement dit, l'utilisation exclusive de la porte O'_x comme source de renseignement sur x ne nous donne accès au mieux qu'à $\{x, \bar{x}\}$: il nous sera tout à fait impossible de faire la distinction entre x et $\bar{x}$.

Une façon de simuler O_x par O'_x serait de tricher allègrement en n'utilisant O'_x non pas de manière isolée, mais de manière *contrôlée*, c'est-à-dire en utilisant en réalité une application A_x définie par

$$A_x |\varepsilon\rangle |i\rangle = |\varepsilon\rangle \left[{O'_x}^{\varepsilon} |i\rangle \right] = (-1)^{\varepsilon \cdot x(i)} |\varepsilon\rangle |i\rangle ,$$

où $\varepsilon \in \{0; 1\}$. On a alors $A_x \frac{|0\rangle+|1\rangle}{\sqrt{2}} |i\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + (-1)^{x(i)} |1\rangle \right) |i\rangle$, et il ne reste plus qu'à tester si le premier registre contient $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ ou bien $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$;

[6] C'est-à-dire tels que $\pi_r \circ \pi'_r = 0$ quand $r \neq r'$, soit, ce qui est équivalent pour des projecteurs orthogonaux, que leurs images sont deux à deux orthogonales, de sorte que pour tout vecteur $|\psi\rangle$ on a $\|(\pi_r + \pi_{r'}) |\psi\rangle\|_2^2 = \|\pi_r |\psi\rangle\|_2^2 + \|\pi_{r'} |\psi\rangle\|_2^2$ simplement d'après le théorème de Pythagore.

la chose est faisable puisque ces vecteurs sont orthogonaux, alors qu'elle était infaisable directement avec $|i\rangle$ et $-|i\rangle$. Autrement dit, par un simple changement de base unitaire, on peut traiter $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ et $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ respectivement comme $|0\rangle$ et $|1\rangle$, et agir en conséquence comme on l'aurait fait avec la porte O_x, modulo ce changement de base. Pour être plus précis, il existe une transformation unitaire U telle que

$$U|\psi\rangle|i\rangle|j\rangle = \begin{cases} |\psi\rangle|i\rangle|j\rangle & \text{si } |\psi\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}} \\ |\psi\rangle|i\rangle|1-j\rangle & \text{si } |\psi\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}} \end{cases}$$

et alors on a $UA_x\frac{|0\rangle+|1\rangle}{\sqrt{2}}|i\rangle|j\rangle = \frac{|0\rangle+(-1)^{x(i)}}{\sqrt{2}}(O_x|i\rangle|j\rangle)$, ce qui montre que la porte O_x peut être effectivement simulée en appliquant la porte O'_x de manière contrôlée, au prix de l'utilisation d'un qubit supplémentaire pour chaque appel de O_x.

La porte O'_x, quant à elle, est tout à fait simulable par la porte O_x. Il suffit en effet de remarquer la relation suivante :

$$O_x\left[|i\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] = (-1)^{x(i)}|i\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}} = \left[O'_x|i\rangle\right]\frac{|0\rangle-|1\rangle}{\sqrt{2}}.$$

Comme le registre additionnel portant $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ est inchangé dans cette opération, on peut utiliser le même pour toutes les requêtes, et donc un algorithme quantique utilisant pour les requêtes les portes O'_x est simulable par un algorithme utilisant les portes O_x au coût minime de l'ajout d'un qubit auxiliaire, ce qui revient à multiplier le d de la définition 1.21 par 2.

1.4.1 De la pertinence du modèle de calcul quantique

De la même manière que le vocabulaire employé couramment pour décrire un algorithme probabiliste colle rarement avec la définition qui en a été donnée dans la section 1.3, un algorithme quantique peut être décrit en des termes qui peuvent sembler *a priori* ne pas s'insérer dans le formalisme de la définition 1.21. Nous pourrons ainsi faire référence au concept de **mesure**. La mesure est un concept fondamental en mécanique quantique. Celle-ci affirme en effet que les objets quantiques sont décrits[7], comme nous l'avons fait, par des vecteurs unitaires, dits *vecteurs d'état*. Mais à nous humains, qui sommes des machines trop grosses pour que des effets quantiques s'y appliquent, ces vecteurs ne sont pas accessibles en tant que tels. Il se trouve que l'interaction apparemment très compliquée qui se produit quand un système quantique entre en contact avec un système classique, ici l'observateur, est très bien

[7] Selon l'interprétation métaphysique que l'on fait de la mécanique quantique, on peut même considérer qu'un système quantique n'est pas seulement *décrit* par un vecteur, mais *est* un vecteur.

décrite par ce processus de la mesure. On peut donc dire que la mesure est le procédé nécessaire à toute observation sensible d'un système quantique : il nous est impossible de connaître *directement* un état quantique, mais si l'on est capable de reproduire ce système en plusieurs exemplaires, on pourra en tirer plus d'informations, en faisant plus de mesures, de la même manière que l'on peut reconstituer une forme tridimensionnelle à partir de clichés bidimensionnels, avec d'autant plus de précision que l'on a de clichés ; c'est ce que l'on appelle la tomographie.

Nous allons commencer par décrire un cas particulier simple du processus de mesure, avant d'expliquer comment cela se généralise. Concrètement, dans le cas simple donc, mesurer un système quantique representé par un vecteur d'état $|\psi\rangle$ vivant dans un espace de Hilbert $\mathscr{V}$, c'est commencer par choisir une base orthonormée $|e_1\rangle, |e_2\rangle, \ldots, |e_n\rangle$ de $\mathscr{V}$. Comme $|\psi\rangle$ est un vecteur unitaire, on a $|\psi\rangle = \sum_{i=1}^{n} \lambda_i |e_i\rangle$, avec $\sum_{i=1}^{n} |\lambda_i|^2 = 1$. Le résultat de la mesure, c'est-à-dire ce qui est effectivement observé, est un certain $i \in \{1, \ldots, n\}$, qui d'après les lois de la mécanique quantique est aléatoire, chaque i pouvant être observé avec probabilité $\left|\lambda_i^2\right|$. Après cela le système quantique se retrouve lui-même dans l'état $|e_i\rangle$. Par analogie avec notre définition des algorithmes déterministes et probabilistes (définitions 1.11 et 1.15), on peut constater que le système quantique une fois mesuré se retrouve dans un état classique, puisque son état $|e_i\rangle$, ou $|e_i)$ comme nous le notions alors, est dans une base orthonormée fixée de $\mathscr{V}$, et non pas n'importe quel vecteur unitaire. Cependant, comme l'état $|e_i)$ se produit avec probabilité $|\lambda_i|^2$, on peut tout aussi bien considérer qu'une fois la mesure effectuée, l'état du système peut être décrit dans le cadre probabiliste par le vecteur stochastique $\sum_{i=1}^{n} |\lambda_i|^2 |e_i)$. Ainsi la mesure, outre l'information qu'elle apporte à l'observateur, transforme le système quantique en un système classique ; c'est pour cette raison que l'on dit que l'observateur perturbe le système, et ce phénomène qui affecte le système observé pour en faire un système probabiliste classique s'appelle la **décohérence**. De plus, cette opération est, d'après les lois de la physique, fondamentalement irréversible, ce qui signifie qu'un système quantique ne peut être mesuré qu'une seule fois. On pourrait alors penser à en faire une copie avant de le mesurer, pour pouvoir effectuer par la suite une autre mesure sur cette copie si le besoin s'en fait sentir ; mais copier, ou *cloner* un système quantique est également interdit par les lois de la physique. Dans notre formalisme, cela se traduit par le fait qu'il n'existe pas d'endomorphisme unitaire U de $\mathscr{V} \otimes \mathscr{V}$ tel que pour tout vecteur unitaire $|\psi\rangle$ de $\mathscr{V}$, on ait $U |\psi\rangle |0\rangle = |\psi\rangle |\psi\rangle$. En effet, on aurait

$$\begin{aligned} U\left(|\psi\rangle + |\varphi\rangle\right)|0\rangle &= \left(|\psi\rangle + |\varphi\rangle\right)\left(|\psi\rangle + |\varphi\rangle\right) \\ &= |\psi\rangle|\psi\rangle + |\varphi\rangle|\varphi\rangle + |\psi\rangle|\varphi\rangle + |\varphi\rangle|\psi\rangle \\ &= U|\psi\rangle|0\rangle + U|\varphi\rangle|0\rangle + |\psi\rangle|\varphi\rangle + |\varphi\rangle|\psi\rangle \\ U\left(|\psi\rangle + |\varphi\rangle\right)|0\rangle &= U\left(|\psi\rangle + |\varphi\rangle\right)|0\rangle + |\psi\rangle|\varphi\rangle + |\varphi\rangle|\psi\rangle\,. \end{aligned}$$

La dernière égalité est absurde car dans le cas général $|\psi\rangle|\varphi\rangle + |\varphi\rangle|\psi\rangle$ n'est pas nul. On peut remarquer que le point essentiel de cette preuve d'impossibilité n'est pas l'unitarité de U, qui ne sert en aucun point, mais son additivité et le fait que l'on puisse choisir des états $|\psi\rangle$ et $|\varphi\rangle$ suffisamment librement pour faire en sorte que $|\psi\rangle|\varphi\rangle + |\varphi\rangle|\psi\rangle$ soit non nul. Par conséquent, il existe un résultat similaire en calcul probabiliste : on ne peut pas cloner un système stochastique. Autrement dit, il n'existe pas de procédure universelle qui à partir d'une observation permette de reproduire tout un tirage aléatoire.

Il semble donc qu'un système quantique dont le vecteur d'état vit dans un espace de Hilbert de dimension N ne permette pas de coder plus d'information qu'un système probabiliste vivant dans un espace de même dimension, c'est-à-dire $\log_2(N)$ bits d'information, puisque essentiellement le mieux que l'on puisse faire est de mesurer le système dans une base bien choisie, après quoi l'état initial est irrémédiablement perdu. Nous avons déjà mentionné qu'il s'agit là d'un résultat connu sous le nom de « borne de Holevo », et dont on peut trouver une formalisation sous l'incarnation du théorème 12.1 de [NC00] — la démonstration est fournie.

Revenons à la question de la mesure. Nous avons vu que les descriptions d'un système, au départ quantique, avant et après la mesure, appartiennent à deux paradigmes bien distincts. Il faudrait imaginer un système de description plus élaboré, englobant les aspects quantiques et les aspects probabilistes des systèmes physiques, si l'on voulait pouvoir inclure la mesure comme une opération aussi élémentaire que les transformations unitaires ou stochastiques. Une telle description existe, sous la forme de *matrices de densité*, mais nous n'allons pas la développer, pour la simple raison qu'elle ne nous serait pas très utile. En effet, le modèle de la définition 1.21, est complet au sens où il permet de simuler la notion intuitive de ce qu'est un algorithme quantique, mais n'inclut pas de description de la mesure. La définition naturelle et intuitive d'un algorithme quantique pourrait être par exemple consister en un arbre de décision probabiliste, où à chaque nœud on peut appliquer une opération unitaire quelconque, faire un requête, ou mesurer quelques qubits. Essentiellement il faut donc s'assurer que l'on peut mesurer, et que l'on peut faire des choix aléatoires.

Mettons que nous voulions mesurer un état $|\psi\rangle$, vivant dans un espace de Hilbert $\mathscr{V}$, dans la base $(|e_i\rangle)_{i=1,\dots,n}$. L'idée est d'incorporer un observateur dans le système quantique, c'est-à-dire d'ajouter un système quantique « observant », qui va modifier son état en fonction de la décomposition de

$|\psi\rangle$ dans la base des $|e_i\rangle$. On pourrait penser qu'il faut nécessairement que le système observant soit suffisamment « gros », suffisamment « classique », pour que la décohérence se produise. Ce n'est pas du tout le cas. En effet, si l'on écrit $|\psi\rangle = \sum_{i=1}^{n} \lambda_i |e_i\rangle$, alors il suffit de produire l'état

$$\sum_{i=1}^{n} \lambda_i |e_i\rangle |e_i\rangle \in \mathscr{V} \otimes \mathscr{V}.$$

Cela peut se faire par exemple en partant de l'état $|\psi\rangle |e_1\rangle$ — le point important étant que $|e_1\rangle$ est un vecteur fixé indépendamment de $|\psi\rangle$ — et en lui appliquant la transformation unitaire U définie par $U |e_i\rangle |e_j\rangle = |e_i\rangle \left|e_{(i+j-2 \mod n)+1}\right\rangle$.

Le système observant est précisément de la même taille que le système observé : pour plus de clarté renommons les espaces hilbertiens $\mathscr{V}_1$ et $\mathscr{V}_2$, de sorte que l'état que nous venons de construire soit dans $\mathscr{V}_1 \otimes \mathscr{V}_2$. Les lois de la mécanique quantique nous assurent que cette opération est tout à fait équivalente à une mesure — ce qui ne signifie pas, loin de là, que toute opération de mesure puisse se décrire de cette façon-là — du moment que l'on n'accède plus à $\mathscr{V}_2$ après cela, de sorte que l'on puisse considérer $\mathscr{V}_2$ comme tout à fait inaccessible. Par exemple, si l'on considère ce que l'on appelle une *paire EPR*, qui est un système quantique à deux qubits pouvant être décrit par l'état $\frac{1}{\sqrt{2}} (|0\rangle |0\rangle + |1\rangle |1\rangle)$, que l'on garde pour soi un des qubits et que l'on abandonne l'autre dans la nature, on se retrouve avec un système qui est tout à fait indiscernable, par quelque opération physique qui soit, d'une superposition probabiliste uniforme des états $|0\rangle$ et $|1\rangle$, ce que l'on peut noter $\frac{1}{2} (|0) + |1))$.

Techniquement, cela signifie qu'après cela toutes les opérations unitaires effectuées par notre algorithme devront être de la forme $U \otimes \mathrm{id}_{\mathscr{V}_2}$. Bien entendu nous écrivons ici les choses de manière simplifiée car en général $\mathscr{V}_1$ ne rassemble que quelques qubits du système et $\mathscr{V}_2$ n'est pas nécessairement « à droite » dans le produit tensoriel. De plus, on peut souhaiter recueillir moins d'information sur le système quantique mesuré, l'intérêt étant de moins le perturber. On peut ainsi mesurer non pas dans une base orthonormale, mais par rapport à une décomposition de $\mathscr{V}_1$ en somme directe de sous-espaces deux-à-deux orthogonaux : $\mathscr{V}_1 = \bigoplus_{i \in I} E_i$. Un vecteur unitaire $|\psi\rangle$ de $\mathscr{V}_1$ se décompose sous la forme $|\psi\rangle = \sum_{i \in I} \lambda_i |\psi_i\rangle$, où pour tout $i \in I$, $|\psi_i\rangle$ est vecteur unitaire appartenant à E_i ; comme les E_i sont deux à deux orthogonaux, on a $\sum_{i \in I} |\lambda_i|^2 = 1$. Cette fois l'information mesurée est un certain $i \in I$. Avec probabilité $|\lambda_i|^2$, on mesure i et le système se retrouve dans l'état $|\psi_i\rangle$; on a alors une information partielle — l'état du système après mesure appartient à E_i — mais dans le cadre de E_i il n'y a pas eu de décohérence. Par exemple, si l'on code des entiers de cette façon, plutôt que de déterminer précisément

quel est l'entier représenté, ce qui après la mesure conduirait à se retrouver avec un système purement probabiliste, on peut se contenter de demander, par exemple, si l'entier est pair, ce qui correspond à une décomposition de $\mathscr{V}_1$ en deux sous-espaces orthogonaux, le sous-espace engendré par les vecteurs représentant entiers pairs et le sous-espace engendré par les vecteurs représentant entiers impairs ; après la mesure on se retrouve projeté sur l'un de ces sous-espaces, mais le vecteur d'état n'appartient pas nécessairement à une base orthonormale fixée, ce qui fait qu'il peut rester des propriétés quantiques exploitables.

Pour clôre ce bref aperçu de la mesure, mentionnons-en un autre raffinement. On peut vouloir effectuer une mesure aléatoire, pas obligatoirement toujours la même. En réalité, ce n'est qu'un particulier d'un procédé général : on peut vouloir appliquer non pas toujours la même transformation unitaire, mais plutôt se donner une répartition de probabilité sur les transformations unitaires et tirer au sort. En fait, nous avons déjà vu comment faire. En effet, il suffit de produire d'abord le nombre de bits aléatoires nécessaires à la simulation de la répartition de probabilité voulue, puis d'appliquer les transformations unitaires de manière contrôlée. Dans le détail, supposons que l'on souhaite appliquer la transformation U_i avec probabilité p_i, pour $i \in I$. On commence par produire un état s'écrivant sous la forme $\sum_{i\in I} \sqrt{p_i}\,|e_i\rangle$, où les $|e_i\rangle$ forme une base orthonormée d'un espace de Hilbert de dimension $|I|$. Ensuite on mesure cet état, ou plutôt on effectue une simulation de mesure par la méthode décrite précédemment. Puis on applique une transformation unitaire U agissant ainsi :

$$U\,|e_i\rangle\,|\psi\rangle = |e_i\rangle\,(U_i\,|\psi\rangle)\,.$$

On constate donc que cet ajout de probabilités dans le processus de la mesure ne change rien, dans le fond. A-t-on fait le tour de la mesure, alors ? Pas tout à fait. On peut remarquer que les procédés de mesure que nous avons considérés jusqu'à présent vérifient tous la propriété qu'il est tout à fait équivalent de mesurer deux fois de suite un système quantique avec le même procédé qu'une seule : cela donne la même répartition de probabilité sur le résultat de la mesure, et le système qui a été mesuré deux fois de suite n'est pas discernable du système qui n'a été mesuré qu'une fois — quoique si on le mesure une troisième fois il manifeste parfois de l'énervement. Les mesures vérifiant cette propriété sont appelées des mesures *projectives*, car telles les projections elles sont idempotentes. Or cela n'est pas toujours le cas en pratique : même pour un système classique, le procédé de mesure peut être destructeur, voire peut désintégrer tout à fait l'objet mesuré, de sorte qu'il n'est pas mesurable une seconde fois. Un exemple quantique « classique » est celui du filtre polarisant, qui effectue d'une certaine manière une mesure de la polarisation des photons, en les détruisant avec une certaine probabilité. Plus proche de l'expérience quotidienne, on peut citer le cas des bandes

magnétiques dans la série *Mission impossible*, qui ne peuvent être écoutées qu'une seule fois. Il existe donc une formalisation plus large de la notion de mesure, que nous n'avons pas l'intention de développer ici car notre but n'était que de donner les idées générales de la manière dont on peut effectivement mêler calcul purement quantique et calcul probabiliste classique dans le modèle de calcul quantique donné dans la définition 1.21. Le lecteur souhaitant plus d'information sur le sujet pourra par exemple se reporter au chapitre 2 de [NC00].

Chapitre 2

Le problème du sous-groupe caché

2.1 Définition

2.1.1 Automorphisme et isomorphisme de graphe

Nous ne parlerons que de graphes non-orientés. Pour nous un graphe $G = (V, E)$ est donc la donnée d'un ensemble de sommets V et d'un ensemble d'arêtes E qui est une partie de ${}^2V$, l'ensemble des paires de sommets.

Étant donné deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$, un **isomorphisme** de G_1 dans G_2 — ou *entre* G_1 et G_2 — est une bijection $\sigma : V_1 \to V_2$ telle que

$$\forall \{i, j\} \in {}^2V \quad \{i, j\} \in E_1 \iff \{\sigma(i), \sigma(j)\} \in E_2.$$

Deux graphes sont dits isomorphes s'il existe un isomorphisme de l'un dans l'autre ; autrement dit, s'ils ne diffèrent que par un renommage de leurs sommets. Quand on s'intéresse aux graphes, la plupart du temps on considère en réalité non pas vraiment les graphes au sens ensembliste où l'on pourrait considérer comme important le « nom » des sommets mais on s'intéresse plutôt à leur classe d'isomorphisme.

Définition 2.1
Une **propriété de graphe** *est une propriété des graphes qui ne dépend que de leur classe d'isomorphisme.*

Exemple 2.2
Comme nous l'avons dit, la plupart des propriétés usuelles sur les graphes sont des propriétés de graphe. Il en va ainsi de la connexité, de la planarité, de la « perfection », etc. On peut citer comme contre-exemple la propriété d'être un arbre binaire équilibré, pour laquelle il faut spécifier lequel des nœuds du graphe forme la racine.

Pour formaliser les problèmes sur les graphes, on considère qu'une requête consiste à demander s'il existe une arête entre deux sommets donnés, et donc on représente un graphe $G = (V, E)$ par la fonction

$$\mu_G : \left(\begin{array}{ccc} {}^2V & \to & \mathbb{B} \\ \{i,j\} & \mapsto & \left\{ \begin{array}{ll} \top & \text{si } \{i,j\} \in E \\ \bot & \text{sinon} \end{array} \right. \end{array} \right)$$

Inversement, à toute fonction $\mu : {}^2V \to \mathbb{B}$ correspond un graphe $G_\mu = (V, E_\mu)$ défini par

$$\{i,j\} \in E_\mu \iff \mu(\{i,j\}) = \top.$$

Le problème d'isomorphisme consiste à déterminer, étant donné deux graphes, s'ils sont isomorphes ; ce qui, formellement, donne la définition suivante.

Définition 2.3
Le problème d'isomorphisme de graphes de taille n est un problème élémentaire défini par le quintuplet $\left([2] \times {}^2[n], \mathbb{B}, \mathbb{B}^{[2]\times{}^2[n]}, f\right)$, *où* $f(\lambda) = \top$ *si et seulement si* $G_{x\mapsto\lambda(0,x)}$ *et* $G_{x\mapsto\lambda(1,x)}$ *sont isomorphes.*

Le problème d'isomorphisme de graphes est alors l'application qui à un entier (non nul) n associe le problème d'isomorphisme de graphes de taille n.

Le problème d'isomorphisme de graphes occupe une place particulière en théorie de la complexité. Bien qu'il soit dans la classe **NP**, il n'est connu ni pour être dans **P**, ni pour être **NP**-complet. À tel point que l'on a donné un nom à la classe des problèmes qui s'y réduisent polynomialement, il s'agit de la classe **GI**, dont on conjecture en général qu'elle est strictement comprise entre les classes **P** et **NP**. On comprend que ce problème soit l'objet de certaines attentions ; on peut citer par exemple [KST93] pour une revue générale du problème, ou [AK06] pour un article plus récent et spécifique.

Un isomorphisme d'un graphe dans lui-même est appelé un **automorphisme**. Il est clair que l'ensemble des automorphismes d'un graphe $G = (V, E)$ forme un groupe pour la loi de composition, en fait un sous-groupe de $\mathfrak{S}_V$. Le problème d'automorphismes de graphe consiste à calculer, étant donné un graphe, son groupe d'automorphismes.

La formalisation de ce problème ne pose pas de difficulté.

Définition 2.4
Le problème d'automorphisme de graphe de taille n est un problème élémentaire défini par le quintuplet $\left({}^2[n], \mathbb{B}, S\left(\mathfrak{S}_{[n]}\right), \mathbb{B}^{\binom{2}{[n]}}, f\right)$, *où $f(\mu)$ est le groupe des automorphismes de G_μ.*

Le problème d'automorphismes de graphe est alors l'application qui à un entier n associe le problème d'automorphismes de graphe de taille n.

On considère généralement plutôt le problème de décision associé, qui consiste à simplement déterminer si un graphe donné possède un automorphisme non trivial. La classe des problèmes qui se réduisent polynomialement à ce problème de décision est notée **GA** ; il est connu que **GA** est inclus dans **GI** — on pourra encore à ce sujet consulter [KST93].

2.1.2 Le sous-groupe caché

Pour définir le problème du sous-groupe caché, nous avons besoin d'un bref rappel de théorie des groupes.

Définition 2.5
Soit G un groupe et H un sous-groupe de G. On définit sur G la relation suivante : $g \sim_H g'$ si et seulement s'il existe $h \in H$ tel que $g' = g \cdot h$. La relation $\sim_H$ est une équivalence sur G. Ses classes d'équivalence sont nommées les **classes à gauche de H (dans G)**. *La classe d'équivalence contenant un élément g est $g \cdot H = \{g \cdot h / h \in H\}$.*

On se donne un groupe fini G. La fonction boîte noire γ est définie sur G, à valeurs dans un ensemble X. On suppose qu'il existe un sous-groupe H de G tel que

$$\forall g, g' \in G \qquad \left(\gamma(g) = \gamma(g') \iff g \sim_H g'\right).$$

Ainsi, une telle fonction est constante sur les classes à gauche de H dans G, c'est-à-dire que $\gamma(g)$ ne dépend que de $g \cdot H$. On peut ainsi factoriser γ en une application $\tilde{\gamma}$, définie sur G/H, telle que $\tilde{\gamma}(g \cdot H) = \gamma(g)$, et $\tilde{\gamma}$ est alors injective. Autrement dit, la propriété 2.1.2 est équivalente à dire qu'il existe $\tilde{\gamma}$ injective rendant ce diagramme commutatif :

Nous reprendrons la terminologie de [Lom04] en disant d'une telle fonction γ qu'elle **sépare** les classes à gauche de H dans G, ou bien plus simplement qu'elle **cache** le sous-groupe H. Étant ainsi assuré que γ cache un certain sous-groupe H de G, le problème du sous-groupe caché consiste naturellement à retrouver de quel sous-groupe il s'agit. Cela est théoriquement possible, puisque une application $\gamma : G \to X$ cache au plus un sous-groupe de G : cela se vérifie aisément en considérant $\tilde{\gamma}$. Par exemple, une application de G dans X cache le sous-groupe trivial si et seulement si elle est injective ; elle cache G lui-même si et seulement si elle est constante.

Comme on peut le constater, X n'est pas muni de la moindre structure. Il s'agit juste d'un ensemble, qui fait partie des données du problème. Tout au plus peut-on supposer qu'il est de taille au moins $|G|$, sinon cela éliminerait d'emblée la possibilité que la fonction boîte noire cache le sous-groupe trivial. Il est plus généralement évident que le problème est plus facile quand X est plus petit. En réalité, cela n'a pas grande importance, mais pour fixer les choses nous allons imposer que X et G soient de même cardinal.

Tout ceci nous amène à donner la définition formelle suivante :

Définition 2.6
Soit G un groupe fini. Le **problème du sous-groupe caché dans** G *est un problème élémentaire, noté* $\mathbf{HSP}_G$*, qui est défini par le quintuplet* $(G, [|G|], S(G), \mathscr{S}, f)$*, où*

- $S(G)$ *désigne l'ensemble des sous-groupes de* G*,*
- $\mathscr{S}$ *est l'ensemble des applications de* G *dans* $[|G|]$ *vérifiant la propriété 2.1.2, c'est-à-dire cachant un sous-groupe de* G*, et*
- $f(\gamma)$ *est le sous-groupe* H *de* G *qui est caché par* γ*.*

Bien entendu, le **problème du sous-groupe caché**, *noté* **HSP**, *est l'application qui à un groupe fini* G *associe le problème du sous-groupe caché dans* G*.*

Nous considérerons également des restrictions de **HSP**, notamment aux groupes abéliens.

Nous pouvons maintenant vérifier que le problème d'automorphismes de graphes se « déduit » assez naturellement du problème du sous-groupe caché — il s'agit en fait d'une notion de réduction que nous n'avons pas définie car elle est de peu d'importance pour la complexité en requêtes de ces problèmes. En effet, pour $G = (V, E)$ un graphe et $\sigma \in \mathfrak{S}_V$, on définit $G^\sigma = (V, E^\sigma)$ par

$$\{i, j\} \in E^\sigma \iff \left\{\sigma^{-1}(i), \sigma^{-1}(j)\right\} \in E.$$

Alors, par définition, $G = G^\sigma$ si et seulement si σ est un automorphisme de G. Mieux, $G^\sigma = G^\tau$ si et seulement s'il existe un automorphisme π de G tel que $\tau = \sigma \circ \pi$. En effet, $E^\sigma = E^\tau$ si et seulement si, pour tout $\{i, j\} \in {}^2X$, $\left\{\sigma^{-1}(i), \sigma^{-1}(j)\right\} \in E \iff \left\{\tau^{-1}(i), \tau^{-1}(j)\right\} \in E$. En effectuant le changement de variable $i = \tau(i')$ et $j = \tau(j')$, on obtient que $E^\sigma = E^\tau$ est équivalent à

$$\forall \left\{i', j'\right\} \in {}^2X \quad \left\{\sigma^{-1} \circ \tau(i'), \sigma^{-1} \circ \tau(j')\right\} \in E \iff \left\{i', j'\right\} \in E,$$

ce qui signifie que $\sigma^{-1} \circ \tau$ est un automorphisme de G, c'est-à-dire qu'il existe un automorphisme π tel que $\tau = \sigma \circ \pi$.

Autrement dit, la fonction qui à une permutation $\sigma \in \mathfrak{S}_V$ associe le graphe G^σ cache le sous-groupe des automorphismes de G. Il suffit donc, en un certain sens, de savoir résoudre le problème du sous-groupe caché sur les groupes symétriques pour savoir résoudre le problème d'automorphisme de graphe. Comme de plus le calcul de G^σ est fort simple, une algorithme efficace pour le problème du sous-groupe caché sur les groupes symétriques donnerait un algorithme presque aussi efficace pour le problème d'automorphisme de graphe. C'est une des raisons pour lesquelles le problème de sous-groupe caché est considéré comme intéressant — la raison principale étant évidemment qu'il présente une certaine beauté théorique et que son étude est intéressante

per se. Cela dit, le problème d'automorphisme de graphe n'est pas si populaire et on pourra préférer savoir en quoi le problème du sous-groupe caché aide à la compréhension du problème d'isomorphisme de graphes.

Supposons donnés deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ dont on veut savoir s'ils sont isomorphes. Pour des raisons techniques qui apparaîtront très bientôt, on supposera que chacun d'eux est connexe ; si ce n'était pas le cas, on pourrait toujours les rendre abruptement connexes en ajoutant à chacun un point relié à tous les autres. Juxtaposons ces graphes pour former le graphe $G_1 \sqcup G_2$, dont l'ensemble des sommets est $V_1 \sqcup V_2$, l'union disjointe de V_1 et V_2, et qui a pour ensemble d'arêtes $E_1 \sqcup E_2$, soit les arêtes de G_1 plus celles de G_2. Quels sont les automorphismes de $G_1 \sqcup G_2$? Nécessairement ils doivent envoyer une composante connexe sur une composante connexe. La question est alors : existe-t-il un automorphisme de $G_1 \sqcup G_2$ échangeant ses deux composantes connexes que forment G_1 et G_2 ? Cela se produit si et seulement si G_1 et G_2 sont isomorphes. Connaître le groupe d'automorphisme de $G_1 \sqcup G_2$ permet donc de déterminer si G_1 et G_2 sont isomorphes, et par transitivité, savoir résoudre le problème du sous-groupe caché sur les groupes symétriques le permet aussi.

2.2 Calcul quantique

On peut trouver beaucoup d'informations à propos du problème du sous-groupe caché dans le livre de Michael A. Nielsen et Isaac L. Chuang [NC00]. En particulier, on y trouvera un historique du problème en page 246 et l'art et la manière de ramener divers problèmes, comme la recherche de l'ordre d'un élément dans un groupe ou le logarithme discret, à un problème de sous-groupe caché, et ce en page 241.

2.2.1 Algorithme standard

Quand on présente un aperçu historique du calcul quantique, parler de l'algorithme de Grover [Gro96] pour la recherche dans un tableau non trié (**RTNT**, voir la définition 4.2) et de celui de Shor pour la factorisation (voir [Sho97]), est en général un passage obligé. Il y a un troisième résultat, antérieur aux deux sus-cités, et tout aussi fondamental concernant notre propos dans ce chapitre, c'est celui que Daniel R. Simon a publié en 1994 sous le titre *On the Power of Quantum Computation* [Sim94]. Le problème que Simon résout est un cas particulier du problème du sous-groupe caché ; il est cependant historiquement douteux de présenter les choses ainsi, car il faut bien comprendre qu'à l'époque la problématique du sous-groupe caché n'était pas d'actualité, mais qu'à l'inverse cette problématique s'est dégagée suite aux succès de Simon [Sim94] et de Shor [Sho97]. Ceci précisé, nous allons présenter brièvement l'algorithme de Simon d'une manière complètement anachronique, mais assumée.

Une caractéristique importante de l'algorithme standard est que l'on se sert des requêtes toujours exactement de la même façon. Reprenons les notations de la section 2.1.2 : G est un groupe fini, et la fonction boîte noire est notée γ.

On commence par préparer, à l'aide d'une matrice unitaire appropriée, un état s'écrivant $\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle\,|0\rangle$, puis on applique la porte de requête O_γ :

On mesure ensuite le second registre. La valeur mesurée n'a aucune importance et peut être oubliée aussitôt, ce qui nous intéresse est qu'après la mesure le système se trouve dans un état particulier, que l'on appelle « coset state », ce qui se traduirait par « état de classe à gauche », mais nous dirons « état coset », et qui s'écrit

$$\frac{1}{\sqrt{|H|}} \sum_{h \in c \cdot H} |h\rangle\,.$$

Il s'agit de la superposition quantique des éléments d'une classe à gauche $c \cdot H$ de H dans G, les classes à gauche étant les parties maximales de G sur lesquelles γ est constante. L'élément c, ou, de manière équivalente, $c \cdot H$, est aléatoire, avec une distribution de probabilité uniforme : sa valeur dépend de ce que l'on a observé en mesurant le registre $|\gamma(g)\rangle$. De manière générale, lorsque X est un ensemble d'éléments x représentés par des vecteurs $|x\rangle$ deux-à-deux orthogonaux, on note $|X\rangle$ le vecteur unitaire superposant les $|x\rangle$ pour x appartenant à X :

$$|X\rangle = \frac{1}{\sqrt{|X|}} \sum_{x \in X} |x\rangle\,.$$

Ainsi, la première étape de l'algorithme standard consiste à produire $|c \cdot H\rangle$ pour un c aléatoire uniformément réparti dans G, et ce à l'aide d'une requête. Si l'on pouvait contrôler ce c d'une manière ou d'une autre, pour par exemple parvenir à produire l'état $|H\rangle$, le problème serait résolu, car il suffirait de mesurer cet état pour obtenir un élément aléatoire de H, et comme nous le verrons plus tard, cela serait suffisant pour résoudre le problème du sous-groupe caché en temps polynomial. Ce c qui agit comme une translation sur les éléments du groupe, il serait par conséquent souhaitable d'en changer la signification par une transformation appropriée ; nous allons voir qu'on peut effectivement en faire un coefficient multiplicateur, un simple scalaire. Cette transformation est analogue à la transformée de Fourier des fonctions réelles, qui change les fréquences en amplitudes ; en effet, il s'agit bien là d'une transformée de Fourier que va subir l'état $|c \cdot H\rangle$, mais d'une transformée de Fourier un peu particulière, qui nécessitera une petite mise au point en matière de théorie des groupes au lecteur peu familier de cette notion.

2.2.2 Un soupçon de théorie des groupes

Définition 2.7
Un **caractère** *d'un groupe G est un morphisme (de groupes) de G dans $\mathbb{U}$, le groupe des nombres complexes de module 1. L'ensemble des caractères de G, muni de la loi de multiplication terme à terme, c'est-à-dire*

$$\left(\begin{array}{ccc} \hat{G} \times \hat{G} & \to & \hat{G} \\ (\chi, \chi') & \mapsto & \chi \cdot \chi' : \left(\begin{array}{ccc} G & \to & \mathbb{U} \\ g & \mapsto & \chi(g)\chi'(g) \end{array} \right) \end{array} \right),$$

forme un groupe pour la composition, nommé le **groupe dual de** *G, et noté $\hat{G}$.*

Exemple 2.8
Un caractère de $\mathbb{Z}$ est un morphisme $\chi : \mathbb{Z} \to \mathbb{U}$. Un tel morphisme étant déterminé par $\chi(1)$, qui peut lui-même être quelconque, il en résulte que $\hat{\mathbb{Z}}$ est isomorphe à $\mathbb{U}$.

Exemple 2.9
Soit N un entier supérieur ou égal à 2. Un caractère de $\mathbb{Z}/N\mathbb{Z}$ est un morphisme $\chi : \mathbb{Z}/N\mathbb{Z} \to \mathbb{U}$. χ est toujours déterminé par $\chi(1)$, mais cette fois $\chi(1)$ est astreint à être une racine N^{e} de l'unité. Le groupe dual de $\mathbb{Z}/N\mathbb{Z}$ est donc isomorphe au groupe des racines N^{es} de l'unité, autrement dit à $\mathbb{Z}/N\mathbb{Z}$ lui-même.

Exemple 2.10
Considérons le groupe $\mathfrak{S}_{[3]}$, et χ un caractère de ce groupe. χ envoie les transpositions sur 1 ou -1 et les cycles de taille 3 sur 1, j ou $\bar{j}$. Comme le produit d'une transposition et d'un cycle de taille 3 est une transposition, χ vaut 1 sur les cycles de taille 3. Comme le produit de deux transpositions distinctes est un cycle de taille 3, χ vaut soit 1 sur toutes les transpositions, soit -1 sur toutes les transpositions. En conclusion, le dual de $\mathfrak{S}_{[3]}$ possède deux éléments : le caractère trivial, et celui qui vaut -1 sur les transpositions et 1, c'est-à-dire qui à une permutation associe son signe.

Plus généralement, comme $\mathbb{U}$ est un groupe abélien, les homomorphismes d'un groupe G dans $\mathbb{U}$ se relèvent en un unique homomorphisme de l'abélianisé G_{ab} de G dans $\mathbb{U}$. On en déduit que G et G_{ab} ont le même dual. L'abélianisé de $\mathfrak{S}_3$ étant isomorphe à $\mathbb{Z}_2$, son dual est donc également isomorphe à $\mathbb{Z}_2$, d'après l'exemple 2.9.

Fait 2.11
Soit G et H deux groupes. À un caractère χ de G et un caractère χ' de H on peut associer le caractère $\rho_{\chi,\chi'}$ de $G \times H$ défini par

$$\rho_{\chi,\chi'}(g,h) = \chi(g)\chi'(h).$$

Comme $G \times \{0\}$ et $\{0\} \times H$ engendrent à eux deux le groupe $G \times H$, réciproquement, tous les caractères de $G \times H$ sont de la forme $\rho_{\chi,\chi'}$, où χ et χ' sont respectivement des caractères de G et H.

Par conséquent, le dual de $G \times H$ est isomorphe à $\hat{G} \times \hat{H}$.

Fait 2.12
Soit G un groupe, H un sous-groupe distingué de G et χ un caractère de G/H. On peut élaborer à partir du χ un caractère $\chi^G_{G/H}$ de G, trivial sur H, simplement défini par $\chi^G_{G/H}(g) = \chi([g])$, où $[g]$ désigne la classe de g dans G/H.

Théorème 2.13
Tout groupe fini abélien est isomorphe à son dual.

Preuve : Selon le théorème de structure des groupes abéliens finis — voir par exemple les théorèmes 8.1 et 8.2 de [Lan65] —, tout groupe abélien fini est isomorphe à un produit direct de groupes cycliques. Or, d'après l'exemple 2.9, tout groupe cyclique est isomorphe à son dual, et d'après le fait 2.11, cette propriété est conservée par le produit direct ; on en déduit donc le résultat. □

Fait 2.14
Si g est un élément non neutre d'un groupe fini abélien G, alors il existe $\chi \in \hat{G}$ tel que $\chi(g) \neq 1$.

Preuve : D'après le théorème de structure des groupes abéliens finis, G s'écrit, à isomorphisme près, sous la forme $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \cdots \times \mathbb{Z}_{m_k}$; g est alors le k-uplet $(g_1, \ldots, g_k)$. Comme g n'est pas le neutre de G, il existe c tel que $g_c \neq 0$. Alors, le caractère χ_c défini par

$$\chi_c : \begin{pmatrix} G & \to & \mathbb{U} \\ (x_1, \ldots, x_k) & \mapsto & e^{2\,\mathbf{i}\,\pi\frac{x_c}{m_c}} \end{pmatrix}$$

est bien tel que $\chi_c(g_c) \neq 1$. □

Définition 2.15
*Soit G un groupe et H un sous-groupe de G. L'***orthogonal de** *H, noté $H^\perp$, est le sous-groupe de $\hat{G}$ constitué des caractères χ tels que pour tout $h \in H$, $\chi(h) = 1$.*

Commençons par une remarque élémentaire, qui ne nous sera pas réellement utile, mais qu'il peut être approprié de garder présente à l'esprit. Pour un groupe G et un sous-groupe distingué H de G, l'orthogonal de H est isomorphe au dual de G/H. En effet, l'application $\chi \mapsto \chi^G_{G/H}$ (voir la notation introduite dans le fait 2.12) est de manière évidente un isomorphisme

du dual de G/H dans l'orthogonal de H. D'après le théorème 2.13, $H^\perp$ est donc isomorphe à G/H.

Voici maintenant une propriété qui nous sera, elle, vraiment utile :

Proposition 2.16
Soit G un groupe, H un sous-groupe de G, et $\chi \in \hat{G}$ un caractère. La moyenne de χ sur H, c'est-à-dire $\frac{1}{|H|}\sum_{h\in H}\chi(h)$, vaut 1 si $\chi \in H^\perp$, 0 sinon.

Preuve : Si χ est dans l'orthogonal de H, par définition, $\chi(h) = 1$ pour tout $h \in H$, donc la moyenne de χ sur H vaut 1. Si χ n'est pas dans l'orthogonal de H, cela signifie qu'il existe $h_c \in H$ tel que $\chi(h_c) \neq 1$. Or, par changement de variable, on a :

$$\frac{1}{|H|}\sum_{h\in H}\chi(h) = \frac{1}{|H|}\sum_{h\in H}\chi(h_c \cdot h) = \chi(h_c)\,\frac{1}{|H|}\sum_{h\in H}\chi(h).$$

Par conséquent, la moyenne de χ sur H doit être nulle. □

Nous définissons maintenant un morphisme d'un groupe G dans son bidual $\hat{\hat{G}}$, que nous noterons μ_G :

$$\mu_G : \left(\begin{array}{ccc} G & \to & \hat{\hat{G}} \\ g & \mapsto & \left(\begin{array}{ccc} \hat{G} & \to & \mathbb{U} \\ \chi & \mapsto & \chi(g) \end{array} \right) \end{array} \right).$$

μ_G est effectivement un morphisme, pour les raisons suivantes. D'une part, $\mu_G(1_G)$ est le caractère de $\hat{G}$ qui a $\chi \in \hat{G}$ associe $\chi(1_G) = 1$, donc le caractère trivial ; c'est-à-dire qu'on a $\mu_G(1_G) = 1_{\hat{\hat{G}}}$. D'autre part, $\mu_G(g \cdot h)$ est le caractère de $\hat{G}$ qui a $\chi \in \hat{G}$ associe $\chi(g \cdot h)$; ceci est à égal $\chi(g)\chi(h)$ car χ est un morphisme, et vaut donc bien $(\mu_G(g)\,\mu_G(h))(\chi)$.

Proposition 2.17
Si G est un groupe fini abélien, alors μ_G est un automorphisme.

Preuve : Comme nous savons déjà que G et son bidual sont isomorphes, il suffit de montrer que μ_G est injectif. Soit donc $g \in G$ tel que μ_G soit le caractère trivial de $\hat{G}$, c'est-à-dire que pour tout $\chi \in \hat{G}$, $\chi(g) = 1$. D'après le fait 2.14, g est nécessairement le neutre de G, ce qui termine — déjà — la preuve. □

On peut remarquer que μ_G étant défini de manière canonique, un groupe abélien fini est canoniquement isomorphe à son bidual, alors qu'il est en général non canoniquement isomorphe à son dual ; une situation fréquente dans le petit monde de la dualité.

À présent nous allons définir ce qui est l'outil principal des algorithmes résolvant les problèmes de sous-groupe caché : la transformée de Fourier.

Dorénavant, G est supposé être un groupe fini abélien. On note $\mathscr{E}_G$ l'espace de Hilbert muni de la base orthonormale $(|g\rangle)_{g\in G}$.

Définition 2.18
La transformée de Fourier de G, notée $\mathbf{F}_G$, est l'application linéaire de $\mathscr{E}_G$ dans $\mathscr{E}_{\hat{G}}$ définie par

$$\mathbf{F}_G\,|g\rangle = \frac{1}{\sqrt{|G|}}\sum_{\chi\in\hat{G}}\chi(g)\,|\chi\rangle\,.$$

Si les mathématiciens modernes ont l'outrecuidance d'appeler ce morphisme « transformée de Fourier », c'est que si, pour reprendre l'exemple 2.9, on considère un groupe cyclique $G=\mathbb{Z}_N$, on obtient

$$\mathbf{F}_{\mathbb{Z}_N}\,|k\rangle = \frac{1}{\sqrt{N}}\sum_{l=0}^{N-1} e^{2\,\mathbf{i}\,\pi\frac{kl}{N}}\,|\chi_l\rangle\,,$$

formule qui rappelle en effet fortement la transformée de Fourier habituelle sur les signaux discrets.

Comme le but avoué est d'utiliser la transformée de Fourier dans un algorithme quantique, il serait pratique que cette transformation soit unitaire. Par chance, et aussi un peu parce l'on a anticipé la chose en introduisant le facteur normalisant $\frac{1}{\sqrt{|G|}}$ dans la définition de $\mathbf{F}_G$, c'est le cas.

Proposition 2.19
$\mathbf{F}_G$ est un isomorphisme unitaire.

Preuve : Pour prouver cela, il nous suffit de vérifier que $\mathbf{F}_G$ envoie la base canonique de $\mathscr{E}_G$ sur une base orthonormée de $\mathscr{E}_{\hat{G}}$. Soit g et h des éléments de G. Il faut montrer que le produit scalaire de $\mathbf{F}_G\,|g\rangle$ et $\mathbf{F}_G\,|h\rangle$ vaut 1 si $g=h$, 0 sinon. On a :

$$\begin{aligned}
\langle \mathbf{F}_G\,|g\rangle\,,\mathbf{F}_G\,|h\rangle\rangle &= \tfrac{1}{|G|}\left\langle \sum_{\chi\in\hat{G}}\chi(g)\,|\chi\rangle\,,\sum_{\chi'\in\hat{G}}\chi'(h)\,|\chi'\rangle\right\rangle\\
&= \tfrac{1}{|G|}\sum_{\chi,\chi'\in\hat{G}}\bar{\chi}(g)\chi'(h)\,\langle|\chi\rangle\,,|\chi'\rangle\rangle\\
&= \tfrac{1}{|G|}\sum_{\chi\in\hat{G}}\bar{\chi}(g)\chi(h)\\
&= \tfrac{1}{|\hat{G}|}\sum_{\chi\in\hat{G}}\chi\left(g^{-1}h\right)\\
\langle \mathbf{F}_G\,|g\rangle\,,\mathbf{F}_G\,|h\rangle\rangle &= \tfrac{1}{|\hat{G}|}\sum_{\chi\in\hat{G}}\mu_G\left(g^{-1}h\right)(\chi)\,.
\end{aligned}$$

D'après la proposition 2.16, ce produit scalaire vaut 1 si le caractère $\mu_G\left(g^{-1}h\right)$ est dans l'orthogonal de $\hat{G}$, qui d'après le

fait 2.14 est réduit au sous-groupe trivial, 0 sinon. Autrement dit, si $\mu_G\left(g^{-1}h\right)$ est le neutre du bidual de G, alors le produit scalaire vaut 1, sinon il vaut 0. Or, d'après la proposition 2.17, μ_G est un automorphisme, donc $\mu_G\left(g^{-1}h\right)$ est neutre si et seulement si $g = h$, ce qui est bien ce que l'on voulait. □

$\mathbf{F}_G$ étant unitaire, on peut donc s'en servir dans notre algorithme. Bien sûr, *stricto sensu*, cela n'a aucun sens puisque $\mathbf{F}_G$ n'est pas un endomorphisme. Il suffit cependant de décider arbitrairement d'une bijection entre G et $\hat{G}$ — on peut même demander qu'il s'agisse d'un isomorphisme, même si cela n'est pas nécessaire — pour pouvoir identifier $\mathscr{E}_G$ et $\mathscr{E}_{\hat{G}}$, et par conséquent $\mathbf{F}_G$ comme un automorphisme unitaire de cet espace. Voyons donc ce qui ce passe lorsque l'on applique la transformée de Fourier à l'état $|c \cdot H\rangle$.

$$\begin{aligned}
\mathbf{F}_G\,|c\cdot H\rangle &= \tfrac{1}{\sqrt{|H||G|}} \sum_{\chi\in\hat{G}} \sum_{h\in H} \chi\left(c.h\right)|\chi\rangle \\
&= \tfrac{1}{\sqrt{|H||G|}} \sum_{\chi\in\hat{G}} \chi\left(c\right) \sum_{h\in H} \chi\left(h\right)|\chi\rangle \\
&= \tfrac{1}{\sqrt{|H||G|}} \sum_{\chi\in\hat{G}} \chi\left(c\right) \left(\sum_{h\in H} \chi\left(h\right)\right)|\chi\rangle \\
\mathbf{F}_G\,|c\cdot H\rangle &= \sqrt{\tfrac{|H|}{|G|}} \sum_{\chi\in H^\perp} \chi\left(c\right)|\chi\rangle
\end{aligned}$$

Nous avons ce que nous voulions : maintenant c n'est plus un problème. Si l'on mesure $\mathbf{F}_G\,|c\cdot H\rangle$ dans la base des caractères, on trouvera un élément aléatoire de $H^\perp$, de façon totalement indépendante de c, puisque $\chi(c)$ est toujours de module 1. Il reste à vérifier que la connaissance de $H^\perp$ est suffisante pour retrouver H. Pour cela, il suffit de remarquer que H est isomorphe à son « bi-orthogonal » $\left(H^\perp\right)^\perp$.

Proposition 2.20
$\mu_G\left(H\right) = \left(H^\perp\right)^\perp$.

Preuve : C'est un bel exemple de propriété essentiellement triviale. Il suffit pour la prouver de dérouler les définitions. D'abord, $(\mu_G)^{-1}\left(\left(H^\perp\right)^\perp\right)$ est le sous-groupe des $g \in G$ tels que pour tout $\chi \in H^\perp$, $\chi(g) = 1$. Par définition de $H^\perp$, il est donc immédiat que H est inclus dans ${\mu_G}^{-1}\left(\left(H^\perp\right)^\perp\right)$. De plus, d'après les faits 2.14 et 2.12, si $g \in G \setminus H$, il existe un caractère χ de G tel que $\chi \in H^\perp$ mais $\chi(g) \neq 1$, ce qui signifie précisément qu'on a aussi l'inclusion inverse. □

2.2.3 Complexité

On a ainsi dégagé un schéma d'algorithme :

- Commencer par répéter ces opérations $n(G)$ fois :
  1. Produire un état $|c \cdot H\rangle$, où c est aléatoire, au moyen d'une requête.
  2. Appliquer la transformée de Fourier et mesurer. Le résultat est un élément aléatoire, uniformément réparti, de $H^{\perp}$.
- Poser K le sous-groupe de $\hat{G}$ engendré par les $n(G)$ caractères mesurés, et renvoyer $(\mu_G)^{-1}\left(\hat{K}\right)$.

On pourra objecter que cela ne ressemble en rien à un algorithme quantique. Il est vrai que si les étapes individuelles peuvent être quantiques, la structure globale de l'algorithme ne l'est pas. Il s'agit d'un trait commun à la plupart des algorithmes quantiques conçus à ce jour : il s'agit essentiellement d'algorithmes au sens usuel faisant appel à des routines quantiques. Ce n'est dû, probablement, qu'à la difficulté qu'a encore l'esprit humain à penser dans le cadre de la mécanique quantique ; peut-être verra-t-on apparaître dans un avenir proche des structures algorithmiques proprement quantiques. Quoi qu'il en soit, pour ce qui est de la formalisation, une présentation sous forme d'algorithme itératif n'est pas plus gênante dans le cas quantique que dans le cas probabiliste, et nous avons déjà tenté d'expliquer à grands traits, dans la section 1.4.1, pourquoi ce genre de description informelle se traduit bien dans notre modèle.

Il reste maintenant à déterminer quel est le nombre de passages dans la boucle, $n(G)$, qu'il suffit d'effectuer pour que K soit égal à $H^{\perp}$ avec une probabilité satisfaisante. Habituellement, dans la « littérature », par exemple dans [Lom04], [Hø97], [Hal02], [Bea97] et [Joz98], on signale que l'on peut choisir $n(G)$ pour être $\mathcal{O}(\log|G|)$, et c'est amplement suffisant pour montrer que le calcul quantique résout efficacement le problème du sous-groupe caché abélien, mais nous voulons effectuer ici une analyse un peu plus fine. Dès l'origine, Peter W. Shor remarquait dans [Sho97] que $n(G)$ peut être majoré par une constante, lorsque G est supposé être un groupe cyclique. Pour exprimer une majoration uniforme de $n(G)$ pour tous les groupes abéliens, nous avons besoin d'introduire la définition suivante, qui pour être peu courante, peut tout de même être considérée comme standard (voir par exemple [KS04]).

Définition 2.21
Le **rang** *d'un groupe G, noté $r(G)$ est le cardinal minimal d'un ensemble de générateurs de G.*

Nous allons montrer que l'on peut choisir $n(G) = \mathcal{O}(r(G))$. Cela va passer par l'étude d'une certaine quantité e_k^G dont voici la définition.

Définition 2.22
Pour un groupe fini abélien G et un entier naturel k, e_k^G est la probabilité que k éléments aléatoires de G — non nécessairement deux-à-deux distincts — engendrent G.

Nous allons exploiter une fois de plus la structure particulière des groupes finis abéliens pour déterminer e_k^G. Commençons par cette remarque :

Fait 2.23
Si G et H sont deux groupes (finis abéliens) d'ordre premier entre eux, alors $e_k^{G\times H} = e_k^G \cdot e_k^H$.

Preuve : Soit (g_1, h_1), (g_2, h_2), …, (g_k, h_k) des éléments aléatoires uniformément répartis dans $G \times H$. On notera X l'ensemble $\{(g_1, h_1), \ldots, (g_k, h_k)\}$ et X_G et X_H, respectivement, la projection de X sur G et H. Clairement, les événements « X_G engendre G » et « X_H engendre H » sont indépendants, puisque les g_i sont indépendants des h_i. De plus, si X engendre $G \times H$, X_G et X_H engendrent respectivement G et H. Rien de ce que nous avons dit jusqu'à présent n'utilisait le fait que G et H sont d'ordre premier entre eux, mais nous allons en avoir besoin pour prouver la réciproque.

Supposons que X_G et X_H engendrent respectivement G et H et soit $(g, h) \in G \times H$. On a $g = \prod_{i=1}^{k} g_i^{\alpha_i}$ et $h = \prod_{i=1}^{k} h_i^{\beta_i}$. D'après le théorème des restes chinois, comme $|G|$ et $|H|$ sont premiers entre eux, il existe, pour chaque $i \in \{1, \ldots, k\}$, un entier r_i tel que $r_i \equiv \alpha_i \,[|G|]$ et $r_i \equiv \beta_i \,[|H|]$. D'après le théorème de Lagrange, l'ordre d'un élément d'un groupe divise l'ordre du groupe ; on a donc $g_i^{r_i} = g^{\alpha_i}$ et $h_i^{r_i} = h^{\beta_i}$. On en déduit

$$(g, h) = \prod_{i=1}^{k} (g_i, h_i)^{r_i} .$$

Comme (g, h) est quelconque parmi les éléments des $G \times H$, ceci montre que X engendre $G \times H$. Résumons : d'une part, X engendre $G \times H$ si et seulement si X_G et X_H engendrent respectivement G et H, d'autre part ces deux derniers événements sont indépendants. On en déduit alors que la probabilité que X engendre $G \times H$ est égale au produit des probabilité que X_G engendre G et que X_H engendre H, c'est-à-dire $e_k^{G\times H} = e_k^G \cdot e_k^H$.
□

Rappelons que d'après le théorème de structure des groupes abéliens finis, G est isomorphe à $G_1 \times G_2 \times \cdots \times G_k$, où, pour chaque i, G_i est un p_i-groupe — c'est-à-dire est d'ordre une puissance du nombre premier p_i. Il nous suffit

donc de calculer e_k^G quand G est un p-groupe pour en déduire simplement sa valeur sur tous les groupes abéliens finis par le fait 2.23. Les deux faits qui suivent on pour but de montrer que l'on n'a même pas besoin d'étudier tous les p-groupes finis abéliens, mais seulement ceux de la forme $(\mathbb{Z}_p)^r$. Rappelons d'abord que pour un groupe G, une partie $X \subseteq G$ et un entier k, on désigne par X^k l'ensemble des x^k pour $x \in X$. Rappelons également que pour un groupe G, un sous-groupe H de G et un élément g de G, $[g]$ désigne la classe de g dans G/H, du moment qu'il n'y a pas d'ambiguïté sur ce que sont G et H.

Fait 2.24
Soit G et H deux p-groupes abéliens et f un morphisme de G dans H. Il existe un unique morphisme $\tilde{f}$ de G/G^p dans H/H^p agissant comme f, c'est-à-dire tel que pour tout $g \in G$, $\tilde{f}([g]) = [f(g)]$. Si $\tilde{f}$ est surjectif, alors f l'est aussi.

Preuve : Un tel $\tilde{f}$ existe car pour g et g' dans G on a $f(gg'^p) = f(g)f(g')^p$, donc $f(g)$ et $f(gg'^p)$ sont dans la même classe de G/G^p. Son unicité est immédiate par la relation $\tilde{f}([g]) = [f(g)]$. De manière plus générale, pour entier naturel k, il existe un unique morphisme f_k de G/G^{p^k} dans H/H^{p^k} tel que pour tout $g \in G$, $f_k([g]) = [f(g)]$, et alors $f_1 = \tilde{f}$. Comme G et H sont des p-groupes finis, à partir d'un certain rang $G/G^{p^k} = G$ et $H/H^{p^k} = H$, et donc $f_k = f$. Il nous suffit donc de prouver par récurrence sur k que tous les f_k sont surjectifs.

Pour commencer, par hypothèse, f_1 l'est. Ensuite, supposons f_k surjectif. Soit $h \in H$; il s'agit de montrer qu'il existe $g \in G$ et $h \in H$ tel que $f(g) = hh'^{p^{k+1}}$. Comme f_k est surjectif, il existe $g \in G$ et $h' \in H$ tels que $f(g) = hh'^{p^k}$. De plus, f_1 étant surjectif, il existe $g' \in G$ et $h'' \in H$ tels que $f(g') = h'h''^p$. Alors $f\left(gg'^{-p^k}\right) = hh''^{-p^{k+1}}$, ce qui finit la preuve. □

Fait 2.25
Soit G un p-groupe abélien fini, X une partie de G. X engendre G si et seulement si son image dans G/G^p engendre G/G^p.

Preuve : Le sens direct est le plus facile. En effet, par définition du quotient, si $g_1 g_2 \cdots g_n = h$, alors $[g_1][g_2]\cdots[g_n] = [h]$.

Pour la réciproque, posons H le sous-groupe de G engendré par X, et f le morphisme d'inclusion de H dans G. Supposons que $[X]$ engendre G/G^p, autrement dit que $H/H^p = G/G^p$. D'après le fait 2.24, on en conclut que f est surjectif, c'est-à-dire que X est une partie génératrice de G. □

Que déduit-on de cela ? Posons G un p-groupe abélien fini. Il faut remarquer que, pour tout sous-groupe H de G, la projection sur G/H de la distribution uniforme sur G est la distribution uniforme sur G/H. Cette propriété, appliquée à $H = G^p$, en conjonction avec le fait 2.25 que nous venons de démontrer, implique que l'on a, pour tout $k \in \mathbb{N}$, $e_k^G = e_k^{G/G^p}$. G/G^p est un p-groupe abélien fini, dont tous les éléments non neutres sont d'ordre p : il est donc isomorphe à $(\mathbb{Z}_p)^k$, pour un certain k qui se trouve être en réalité le rang de G. C'est donc le cas de ce type de groupes qu'il convient de traiter et le calcul de e_k^G pour tous les groupes abéliens finis en découlera.

Il faut commencer par remarquer que $(\mathbb{Z}_p)^n$ peut être vu comme un espace vectoriel sur le corps $\mathbb{F}_p$; les deux structures sont tout à fait équivalentes, au sens où les sous-espaces vectoriels sont les sous-groupes, les parties engendrées par un ensemble donné d'éléments sont les mêmes dans les deux structures, etc. Nous aurons par conséquent librement recours au vocabulaire des espaces vectoriels quand cela nous paraîtra utile. Pour calculer $e_k^{(\mathbb{Z}_p)^n}$, nous allons d'abord effectuer le simple dénombrement suivant, qui nous servira aussi ultérieurement dans le chapitre 3 pour calculer une borne inférieure sur la complexité en requêtes de ce même problème du sous-groupe caché.

Fait 2.26
Le nombre de n-uplets libres de $(\mathbb{Z}_p)^k$ est

$$\alpha_p(k,n) = \prod_{i=0}^{n-1} \left(p^k - p^i\right).$$

Preuve : Pour qu'un n-uplet $(v_0, \ldots, v_n)$ soit libre, il faut dans un premier temps que v_0 soit non nul, ce qui donne $p^k - 1$ possibilités. Pour v_1, on peut choisir n'importe quel vecteur qui n'est pas dans le sous-espace engendré par v_0 : il reste $p^k - p$ possibilités. Pour v_2, on doit maintenant s'interdire tout le sous-espace engendré par v_0 et v_1, qui est de dimension 2, donc de cardinal p^2 ; et ainsi de suite. □

Le nombre de n-uplets libres de $(\mathbb{Z}_p)^k$, c'est également le nombre de matrices $k \times n$ à coefficients dans $\mathbb{Z}_p$ qui sont de rang n ; en considérant la transposée de ces matrices, on peut encore dire qu'il s'agit du nombre de k-uplets de $(\mathbb{Z}_p)^n$ qui engendrent tout l'espace. Comme il y a exactement $(p^n)^k$ k-uplets d'éléments de $(\mathbb{Z}_p)^n$, on en déduit ceci :

$$e_k^{(\mathbb{Z}_p)^n} = \prod_{i=0}^{n-1} \left(1 - p^{i-k}\right).$$

On en déduit que dans le cas plus général où G est un p-groupe[1] abélien, on a $e_k^G = \prod_{i=0}^{r(G)-1} \left(1 - p^{i-k}\right)$. Mieux, dans le cas tout à fait général où G est juste supposé abélien et fini, comme G est un produit de p-groupes de rang au plus $r(G)$, on a

$$e_k^G \geq \prod_{i=0}^{r(G)-1} \prod_{p} \left(1 - p^{i-k}\right),$$

étant entendu que le produit sur p est pris sur tout les p premiers. Le cas $k < r(G)$ étant clairement inintéressant, puisqu'on a alors $e_k^G = 0$, nous allons supposer $k \geq r(G)$, auquel cas on a la relation $\prod_p \left(1 - p^{i-k}\right) = \frac{1}{\zeta(k-i)}$, où ζ est bien la fonction ζ de Riemann — avec la convention $\zeta(1) = +\infty$.

On a ainsi

$$e_k^G \geq \prod_{i=0}^{r(G)-1} \frac{1}{\zeta\left(k-i\right)}.$$

Étant donné la manière dont nous l'avons obtenue, cette minoration est la meilleure qui soit : il s'agit de la borne inférieure des e_k^G à $r(G)$ fixé. Elle permet de démontrer que notre schéma d'algorithme fonctionne avec $\mathcal{O}\left(r(G)\right)$ requêtes. Plus précisément, on peut démontrer ceci :

Proposition 2.27
La complexité en requêtes quantique du problème du sous-groupe caché dans un groupe fini abélien G, pour une probabilité d'erreur au plus ε, est au plus

$$\max\left(r(G) + 4, r(G) + 1 - \log_2 \varepsilon\right).$$

Preuve : La fonction ζ est définie sur les entiers strictement supérieurs à 1 par

$$\zeta(n) = \sum_{j=1}^{+\infty} j^{-n}.$$

On peut écrire $\zeta(n) = 1 + 2^{-n} + 3^{-n}\xi(n)$, avec $\xi(n) = \sum_{j=3}^{+\infty} \left(\frac{j}{3}\right)^{-n}$. ξ étant décroissante, on en déduit que l'on a $\zeta(n) = 1 + 2^{-n} + \mathcal{O}\left(3^{-n}\right)$. Disons que pour $n \geq n_0$, on a $\zeta(n) \leq 1 + 2^{-n} + C \cdot 3^{-n}$. Supposons $k > r(G) + n_0$. Alors

[1] Rappelons qu'un p-groupe est un groupe fini d'ordre une puissance de p.

$$\begin{aligned} -\ln e_k^G &\leq \sum_{i=0}^{r(G)-1} \ln \zeta(k-i) \\ &\leq \sum_{i=0}^{r(G)-1} \ln\left(1 + 2^{i-k} + C \cdot 3^{i-k}\right) \\ &\leq \sum_{i=0}^{r(G)-1} \left(2^{i-k} + C \cdot 3^{i-k}\right) \\ -\ln e_k^G &\leq 2^{r(G)-k} + \tfrac{C}{2} \cdot 3^{r(G)-k} \end{aligned}$$

Remarquons que si $n \geq \frac{\ln C}{\ln \frac{3}{2}}$, alors $2^{-n} + C \cdot 3^{-n} \leq 2^{-(n-1)}$. On en déduit que si $k \geq r(G) + n_1$, où $n_1 = \max\left(n_0 + 1, \frac{\ln C}{\ln \frac{3}{2}}\right)$, alors on a

$$-\ln e_k^G \leq 2^{r(G)-k+1}.$$

La complexité en requêtes du problème du sous-groupe caché dans G pour une probabilité d'erreur au plus ε est majorée par le plus petit k tel que $e_k^G \geq 1 - \varepsilon$, que nous noterons k_ε^G.
Sous la condition $k \geq r(G) + n_1$, pour avoir $e_k^G \geq 1 - \varepsilon$, il suffit que l'on ait $2^{r(G)-k+1} \leq \ln \frac{1}{1-\varepsilon}$, soit $k \geq r(G) + 1 - \log_2 \ln \frac{1}{1-\varepsilon}$. Par conséquent,

$$k_\varepsilon^G \leq \max\left(r(G) + n_1, r(G) + 1 - \log_2 \ln \frac{1}{1-\varepsilon}\right).$$

Cela dit, le terme $\log_2 \ln \frac{1}{1-\varepsilon}$ paraît bien incongru. Il apparaît cependant qu'il est supérieur à $\log_2 \varepsilon$ tout en en étant très proche ; en effet la fonction $f : \varepsilon \mapsto \log_2 \ln \frac{1}{1-\varepsilon} - \log_2 \varepsilon$ est croissante sur $\left[0; \frac{1}{2}\right]$, équivalente à $\frac{\varepsilon}{2}$ en 0 et vaut environ $0,47$ en $\frac{1}{2}$. En conséquence, non seulement on peut écrire de manière plus simple $k_\varepsilon^G \leq \max\left(r(G) + n_1, r(G) + 1 - \log_2 \varepsilon\right)$, mais de plus on perd très peu, pour ne pas dire rien, à faire de la sorte.
Enfin, on peut choisir $n_0 = 2$ et $C = 4$, ce qui donne $n_1 \leq 4$. □

L'énoncé de ce résultat, plutôt plus précis qu'il est de coutume, ne doit pas laisser penser qu'il ait une quelconque prétention à l'optimalité. Par exemple, pour le groupe $\mathbb{Z}_p$, où p est un nombre premier, la complexité en requêtes vaut clairement au plus 2 du moment que la probabilité d'erreur demandée est comprise dans l'intervalle $\left[0; \frac{1}{2}\right[$. En effet, comme $\mathbb{Z}_p$ a pour seul sous-groupes le sous-groupe trivial et lui-même, il suffit pour décider dans quel cas on se situe de demander la valeur de fonction boîte noire en 0 et en 1. Plus généralement, sur un groupe G, la complexité en requêtes est clairement bornée par $|G|$, soit le nombre de requêtes suffisant à connaître la valeur de la fonction boîte noire sur toutes les entrées.

Chapitre 3

Bornes inférieures quantiques

Les deux principales méthodes connues pour prouver des bornes inférieures sur la complexité en requêtes quantique sont la méthode par adversaire, que l'on peut attribuer à Andris Ambainis [Amb02], et la méthode polynomiale, qui a été introduite dans le cadre quantique par Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca et Ronald de Wolf [BBC+98]. Nous allons brièvement commenter le principe de la méthode par adversaire, et après cela, plus nous apesantir sur la méthode polynomiale, car nous avons l'intention d'en faire bon usage. En matière d'introduction à ces méthodes, nous ne saurions toutefois faire mieux que [Phi03], qui présente méthodes, preuves et exemples de manière tout à fait claire.

3.1 La méthode par adversaire

Deux enfants jouent dans la cour. Alceste choisit de tête un nombre de 1 à 100, et Bérénice tente de le trouver en posant des questions du type « plus petit, plus grand ou égal ? ». Alceste pense en secret à 23. Bérénice commence :

« Est-ce que c'est 60 ?

– Non, c'est plus petit.

– Est-ce que c'est 34 ?

– Non, c'est plus petit.

– Est-ce que c'est 23 ?

– Euh... non, pas du tout ! »

Car Alceste vient de perdre plusieurs parties particulièrement rapidement et en a assez, alors il triche. Il réfléchit rapidement. S'il veut rester cohérent, il doit choisir un nouvel entier entre 1 et 33. Comme il y a plus de possibilités entre 1 et 22 qu'entre 24 et 33, il annonce :

« C'est plus petit ! »

Et le jeu se poursuit.

Pour la petite histoire, Alceste a choisi 6 comme nouveau nombre. Mais cela n'a pas d'importance. Alceste n'est pas obligé de réellement choisir un entier, finalement, il lui suffit de rester cohérent dans ses réponses. Et c'est bien là le cœur de la « méthode par adversaire » puisque Alceste, au lieu de jouer franc jeu, devient *de facto* un adversaire de Bérénice et choisit une stratégie de réponse destinée à pointer du doigt les lenteurs de son algorithme.

Nous supposerons à présent avoir affaire à un problème de décision élémentaire $\mathscr{P} = (I, J, \mathbb{B}, \mathscr{S}, f)$, puisque essentiellement la méthode par adversaire telle qu'elle est exprimée habituellement ne s'occupe que de problèmes de décision. En pratique, l'idée générale est, pour montrer une borne inférieure sur la complexité en requêtes, de montrer que, si Alceste choisit un état particulier réparti sur les différentes fonctions boîte noire possibles — une distribution de probabilité dans le cas d'algorithmes probabilistes, un état quantique dans le cas d'algorithmes quantiques — alors quels que soient les efforts de Bérénice elle ne parviendra pas à démêler le vrai du faux sans faire un certain nombre de requêtes.

L'historique des méthodes de bornes inférieures par adversaire quantiques est intéressant, puisque pendant un certain temps, sous l'impulsion principale d'Andris Ambainis [Amb99, Amb02], ces méthodes ont fleuri [BSS03, Amb03, Aar04, LM04], avant d'avoir été prouvées, pour la plupart, équivalentes [ŠS06], ce qui n'empêche pas de continuer d'explorer de nouvelles pistes dans cette voie. Nous donnons ici l'expression de la méthode telle qu'elle est élaborée dans [LM04], et formulée plus explicitement dans [ŠS06].

Théorème 3.1
Soit $\mathscr{P} = (I, J, \mathbb{B}, \mathscr{S}, f)$ un problème de décision élémentaire et $\varepsilon \in \left[0; \frac{1}{2}\right[$. Dans les formules qui vont suivre, p dénote une famille $(p_x)_{x \in \mathscr{S}}$ de distributions de probabilité sur I ; lorsque l'on considère le minimum sur p, il est entendu que p parcourt l'ensemble — infini — de ces familles.

La complexité en requêtes probabiliste de $\mathscr{P}$ pour une probabilité d'erreur au plus ε est supérieure ou égale à $(1 - 2\varepsilon)\, LM^r(\mathscr{P})$, où

$$LM^r(\mathscr{P}) = \min_p \max_{\substack{x, y \in \mathscr{S} \\ f(x) \neq f(y)}} \frac{1}{\sum\limits_{\substack{i \in I \\ x(i) \neq y(i)}} \min\left(p_x(i), p_y(i)\right)}.$$

La complexité en requêtes quantique de $\mathscr{P}$ pour une probabilité d'erreur au plus ε est supérieure ou égale à $\left(1 - 2\sqrt{\varepsilon(1-\varepsilon)}\right) LM^q(\mathscr{P})$, où

$$LM^q(\mathscr{P}) = \min_p \max_{\substack{x, y \in \mathscr{S} \\ f(x) \neq f(y)}} \frac{1}{\sum\limits_{\substack{i \in I \\ x(i) \neq y(i)}} \sqrt{p_x(i) p_y(i)}}.$$

Nous reviendrons sur ce théorème dans la section 4.1.3.

3.2 La méthode polynomiale

Par contraste avec la méthode par adversaire, la méthode polynomiale paraît bien peu algorithmique. Elle ne repose pas sur l'étude d'un jeu d'interactions entre l'algorithme et le problème qu'il cherche à résoudre, mais se contente de poser un principe algébrique général sur le comportement moyen de l'algorithme. Son intérêt le plus évident, du fait qu'il s'agit d'une méthode bien différente, est qu'elle n'a pas les mêmes défauts que la méthode par adversaire, et peut donc être utilisée de façon complémentaire, quand celle-ci fait défaut, par exemple pour les problèmes de collision. Plus particulièrement, pour le problème de collision, consistant à trouver des indices i et j tels que $x(i) = x(j)$, et pour le problème consistant à déterminer si n entiers donnés sont distincts, seule la méthode polynomiale, à l'heure actuelle, permet de donner des bornes inférieures satisfaisantes [AS04, Kut05] ; nous verrons un autre exemple dans la section 3.3.

Pas de miracle cependant, la méthode a ses défauts, au premier rang desquels sans doute la difficulté apparente d'application. Elle se présente sous un aspect un peu abrupt, et il faut généralement faire usage des *symétries* du problèmes pour se ramener à un résultat utilisable. On comprendra, par conséquent, qu'elle ne soit en pratique utile que sporadiquement : elle requiert du problème traité qu'il soit suffisamment structuré pour que l'exploitation de ses symétries produise quelque chose d'intéressant, et aussi, peut-être, un peu de chance.

3.2.1 Le théorème principal

Ce que l'on appelle **méthode polynomiale** repose entièrement sur un résultat, exposé dans [BBC+98], que nous allons présenter de suite en reprenant le formalisme de [AS04]. Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire. Pour s une fonction partielle de I dans J et x une application de I dans J, on définit

$$I_s(x) = \begin{cases} 1 & \text{si } x \text{ étend } s \\ 0 & \text{sinon} \end{cases} .$$

On dit qu'une fonction x **étend** une fonction s si le domaine de s est inclus dans celui de x et que x coïncide avec s sur ce domaine. C'est tout ce dont nous avons besoin pour énoncer le...

Théorème 3.2 (Beal, Buhrman, Cleve, Mosca et de Wolf)
Soit $\mathscr{A}$ un algorithme (probabiliste ou quantique) pour le problème $\mathscr{P} = (I, J, R, \mathscr{S}, f)$, de complexité en requêtes T et soit $r \in R$. Il existe un ensemble S de fonctions partielles de I dans J et une famille de nombres réels $(\alpha_s)_{s \in S}$ tels que :

1. *pour tout $s \in S$,*

- $|\mathrm{dom}(s)| \leq T$ *si* $\mathscr{A}$ *est probabiliste,*
- $|\mathrm{dom}(s)| \leq 2T$ *si* $\mathscr{A}$ *est quantique, et*

2. *pour toute application* x *de* I *dans* J, $\mathscr{A}$ *répond* r *sur l'entrée* x *avec probabilité*

$$\sum_{s \in S} \alpha_s I_s(x).$$

Pourquoi appelle-t-on ceci la « méthode polynomiale » ? Pour la raison suivante. Pour i dans I et j dans J, notons $\Delta_{i,j}$ l'application qui à une application $x : I \to J$ associe 1 si $x(i) = j$, 0 sinon. I_s s'écrit alors comme un nomôme en les $\Delta_{i,j}$:

$$I_s = \prod_{i \in \mathrm{dom}(s), j = s(i)} \Delta_{i,j}.$$

La somme $\sum_{s \in S} \alpha_s I_s$ qui apparaît dans le théorème 3.2 est donc un polynôme en les $\Delta_{i,j}$, qui plus est de degré au plus T dans le cas probabiliste, $2T$ dans le cas quantique.

Avant de donner la preuve, remarquons un point important de la méthode polynomiale. Elle ne se contente pas de parler du comportement de $\mathscr{A}$ sur les entrées $x \in \mathscr{S}$, mais sur toutes les applications de J^I. Cela peut avoir son importance : voir par exemple [AS04], [Kut05] ou l'application que nous en ferons nous-mêmes dans la proposition 3.9. Nous allons maintenant présenter une preuve calquée sur celle de [BBC+98], puis nous reviendrons sur la façon d'appliquer cette méthode.

> *Preuve du théorème 3.2 :* Notons comme d'habitude $|\psi_k\rangle$ l'état de l'algorithme juste avant la requête numéro k — étant entendu que l'on commence la numérotation à 0. $|\psi_{k+1}\rangle = M_k O_x |\psi_k\rangle$, où O_x est la porte effectuant la requête, x désignant la fonction boîte noire, et M_k la k^{e} transformation linéaire globale ne faisant pas intervenir d'appel à la boîte noire. Les coefficients de $|\psi_0\rangle$ ne dépendant pas de x, on peut considérer qu'il s'agit de polynômes de degré 0 en les $\Delta_{i,j}(x)$. Nous allons montrer par récurrence sur k que les coefficients du vecteur $|\psi_k\rangle$ sont des polynômes en les $\Delta_{i,j}$ de degré au plus k.
>
> Comme les M_k sont des applications linéaires ne dépendant pas de x, les coefficients de $|\psi_{k+1}\rangle$ ne sont qu'une combinaison linéaire des coefficients de $O|\psi_k\rangle$; par conséquent le degré des polynômes ne peut que diminuer.
>
> Divisons le système en trois registres : le permier contient la requête envoyée à la boîte noire, le deuxième l'endroit où l'on stocke la réponse par le procédé habituel, le troisième tout le reste, de sorte que l'application de requête O agit ainsi :

$$O_x \left|i,j,z\right\rangle = \left|i,j\circ x(i),z\right\rangle .$$

Rappelons que $\circ$ est une quelconque application de $J\times J$ dans J, simplement supposée bijective en chacune des variables. Écrivons $\left|\psi_k\right\rangle = \sum_{i,j,z} \beta^k_{i,j,z}(x)\left|i,j,z\right\rangle$. Par hypothèse de récurrence, chaque $\beta^k_{i,j,z}(x)$ est un polynôme de degré au plus k en les $\Delta_{i,j}(x)$. On a alors

$$O_x \left|\psi_k\right\rangle = \sum_{i,j,z} \beta^k_{i,j,z}(x)\left|i,j\oplus x(i),z\right\rangle .$$

Autrement dit,

$$O_x \left|\psi_k\right\rangle = \sum_{i,j,j',z/j'=j\circ x(i)} \beta^k_{i,j,z}(x)\left|i,j',z\right\rangle .$$

C'est à ce point qu'entrent en jeu les monômes $\Delta_{i,j}$. Par définition, $\Delta_{i,j''}(x)$ vaut 1 si $x(i)=j''$, 0 sinon ; on en déduit que si l'on note μ l'application de $J\times J$ dans J vérifiant $j_1 = j_2\circ\mu\left(j_1,j_2\right)$, on a $\Delta_{i,\mu(j',j'')}(x)=1$ si et seulement si $x(i)=\mu(j',j'')$, c'est-à-dire $j''\circ x(i)=j'$. Pour i, j et j' tels que $j'=j\circ x(i)$, on peut donc écrire

$$\beta^k_{i,j,z}(x) = \sum_{j''} \beta^k_{i,j'',z}(x)\Delta_{i,\mu(j',j'')}(x).$$

Au final, on obtient

$$O_x \left|\psi_k\right\rangle = \sum_{i,j',z}\left(\sum_{j''} \beta^k_{i,j'',z}(x)\Delta_{i,\mu(j',j'')}(x)\right)\left|i,j',z\right\rangle .$$

Comme, par hypothèse de récurrence, chaque $\beta^k_{i,j,z}(x)$ est un polynôme de degré au plus k en les $\Delta_{i,j}(x)$, les coefficients de $O_x\left|\psi_k\right\rangle$ sont des polynômes de degré au plus $k+1$ en ces mêmes variables. Ainsi, si l'on note $\left|\psi_T\right\rangle$ l'état du système à la fin du calcul et que l'on divise cette fois le système en deux registres, le premier contenant le résultat renvoyé par l'algorithme, le second contenant tout le reste, on a

$$\left|\psi_T\right\rangle = \sum_{r,z} \gamma_{r,z}(x)\left|r,z\right\rangle ,$$

où les $\gamma_{r,z}$ sont des polynômes de degré au plus T en les $\Delta_{i,j}$. La probabilité que l'algorithme $\mathscr{A}$ réponde r est alors

- $\sum_z \gamma_{r,z}(x)$ si $\mathscr{A}$ est probabiliste,
- $\sum_z |\gamma_{r,z}(x)|^2$ si $\mathscr{A}$ est quantique.

Ainsi, si $\mathscr{A}$ est probabiliste, le résultat du théorème 3.2 suit immédiatement. Cela paraît *a priori* moins évident si $\mathscr{A}$ est quantique. Pourtant, il suffit de se rappeler que pour un nombre complexe $z = a + \mathbf{i}\, b$, $|z|^2 = a^2 + b^2$, ce qui fait que, les $\gamma_{r,z}$ étant des polynômes de degré au plus T en les $\Delta_{i,j}$, la probabilité que $\mathscr{A}$ réponde r est bien un polynôme de degré au plus $2T$.

□

Comment peut-on utiliser ce théorème pour prouver des bornes inférieures sur la complexité en requêtes d'un problème ? Imaginons que nous ayons un problème de décision élémentaire $\mathscr{P} = (I, J, \mathbb{B}, \mathscr{S}, f)$, et $\mathscr{A}$ un algorithme le résolvant avec probabilité d'erreur au plus $\frac{1}{3}$. Notons $\mathbb{P}_{\mathscr{A}} : J^I \to [0;1]$ la fonction qui à une fonction boîte noire — quelconque, pas nécessairement dans $\mathscr{S}$ — associe sa probabilité d'acceptation par $\mathscr{A}$. Cette fonction a les propriétés importantes suivantes :

- pour tout $x \in J^I$, $\mathbb{P}_{\mathscr{A}}(x) \in [0;1]$,
- pour tout $x \in f^{-1}(\top)$, $\mathbb{P}_{\mathscr{A}}(x) \in \left[\frac{2}{3};1\right]$, et
- pour tout $x \in f^{-1}(\bot)$, $\mathbb{P}_{\mathscr{A}}(x) \in \left[0;\frac{1}{3}\right]$.

On peut aussi voir cette probabilité comme une fonction partielle sur $\{0;1\}^{I\times J}$, qu'un peu abusivement nous noterons encore $\mathbb{P}_{\mathscr{A}}$: en ce sens elle a $|I \times J|$ variables que l'on notera $(\delta_{i,j})_{(i,j)\in I\times J}$. À une application $x \in J^I$ on associe l'élément de $\{0;1\}^{I\times J}$ défini par $\delta_{i,j} = \Delta_{i,j}(x)$. $\mathbb{P}_{\mathscr{A}}$ est alors définie sur l'ensemble des éléments correspondant à des fonctions x. Et, d'après le théorème 3.2, $\mathbb{P}_{\mathscr{A}}$ est prolongeable sur tout $\{0;1\}^{I\times J}$ par un polynôme de degré au plus T ou $2T$.

On est donc dans cette situation : on sait qu'il existe un polynôme de degré au plus T ou $2T$ qui doit vérifier certaines inégalités connues en quelques points de contrôle. On s'attend à ce que ces inégalités donnent une borne inférieure sur le degré du polynôme en question, et partant, sur T. Le problème, c'est qu'il s'agit là de polynômes à $|I| \cdot |J|$ variables. C'est beaucoup trop. Nous verrons dans la section 3.2.2 quelques lemmes techniques qui permettent d'expliciter des bornes inférieures dans quelques cas particuliers, mais il s'agit toujours de polynômes à *une seule* variable.

Le point délicat de la méthode polynomiale est donc de se ramener à un polynôme univarié. La méthode de base consiste à définir une fonction partielle $\mathscr{V} : J^I \to \mathbb{R}$, fonction que l'on va naturellement choisir de manière à ce qu'elle colle au problème, qu'elle respecte ses symétries, etc. ; pour cette raison, on appelle cette technique la **symétrisation**. On définit ensuite, pour $a \in \mathbb{R}$, $P(a)$ comme étant la moyenne des $\mathbb{P}_{\mathscr{A}}(x)$ pour $x \in \mathscr{V}^{-1}(a)$. Il n'est pas difficile de choisir $\mathscr{V}$ de telle manière que P vérifie encore une série

d'inégalités en des points de contrôle. Ce qui est *a priori* plus ardu, c'est de faire en sorte que P reste un polynôme.

Bien sûr, il ne s'agit que d'un schéma général, ouvert à toutes sortes d'arrangements. L'application la plus astucieuse et dévoyée de la méthode polynomiale que nous connaissions est l'œuvre de Scott Aaronson, où il se ramène à un ensemble d'applications qui ont non pas une mais deux variables, et qui ne sont pas des polynômes mais s'en rapprochent (cf le lemme 2 de [Aar02]). On peut presque regretter qu'il ait plus tard, avec Yaoyun Shi, prouvé un résultat plus fort avec une méthode plus simple [AS04].

3.2.2 Les lemmes d'approximation

La plupart des applications de la méthode polynomiale utilisent en fin de compte le lemme de Paturi [Pat92] ou celui de Nisan-Szegedy [NS94], qui sont issus de la théorie de l'approximation polynomiale. Nous allons donc les présenter pour donner une idée de ce qui est appliqué usuellement, puis nous démontrerons un troisième lemme, dans la même veine, mais plus adapté à notre situation.

Commençons par le lemme de Nisan-Szegedy. Il fut énoncé clairement par Noam Nisan et Mario Szegedy dans [NS94], mais on peut en trouver des traces antérieures, en filigrane, notamment dans un article de Ehlich et Zeller [EZ64], et dans un autre de Rivlin et Cheney [RC66].

Lemme 3.3 (Nisan-Szegedy)
Soit P un polynôme vérifiant les propriétés suivantes :

1. *pour tout $i \in [n+1]$, $|P(i)| \leq M$, et*
2. *il existe un réel $x \in [0;n]$ tel que $|P'(x)| \geq c$.*

Alors $\deg(P) \geq \sqrt{\frac{cn}{c+2M}}$.

Le preuve étant relativement simple, nous la reproduisons ici. Elle utilise le théorème suivant de Markov :

Théorème 3.4 (Markov)
Soit P un polynôme réel à une variable de degré d tel que pour $x \in [-A;A]$, $|P(x)| \leq M$. Alors pour tout $x \in [-A;A]$, $|P'(x)| \leq \frac{M}{A}d^2$.

Preuve du lemme 3.3 : Soit $c' = \max_{x\in[0;n]} |P'(x)| \geq c$. Étant donné la borne sur $|P|$ sur les entiers, pour tout $x \in [0;n]$, on a $|P(x)| \leq M + \frac{c'}{2}$. Par l'inégalité de Markov, on a alors $c' \leq \frac{M+\frac{c'}{2}}{\frac{n}{2}} \deg(P)^2 = \frac{2M+c'}{n} \deg(P)^2$, c'est-à-dire

$$\deg(P)^2 \geq \frac{c'n}{c'+2M}.$$

Comme $c' \geq c$, on en déduit la conclusion du lemme 3.3. □

Le lemme suivant a été démontré par Paturi dans [Pat92], où il n'apparaît pas tel quel, mais est présent dans la preuve du théorème 4. Quoi qu'il en soit, voilà un autre exemple de lemme d'approximation utilisé.

Lemme 3.5 (Paturi)
Soit P un polynôme réel à une variable, a et b des entiers tels que $a < b$, ξ un réel appartenant à $[a;b]$, $c > 0$ une constante. Supposons que l'on a les deux propriétés suivantes :

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

Alors $\deg(P) = \Omega\left(\sqrt{(\xi - a + 1)(b - \xi + 1)}\right)$.

Il est plus précis que le lemme 3.3 puisqu'il laisse la possibilité de donner des bornes inférieures strictement meilleures que $\Omega\left(\sqrt{b-a}\right)$; en effet, quand ξ est environ à la moitié de l'intervalle, on obtient une borne inférieure en $\Omega(b-a)$. Comme on peut le constater dans [Pat92], ce lemme est utile pour l'étude des fonctions symétriques, à ce point que la borne inférieure fournie est optimale — à un facteur constant près.

Malheureusement, pour l'application que nous avons en tête, les points de contrôle du polynôme P — c'est-à-dire les points où l'on connaît une borne sur ses valeurs — ne sont pas espacés selon une progression arithmétique, mais selon une progression géométrique. Qu'à cela ne tienne, ce lemme-ci, apparu dans [KNP05a], fera l'affaire.

Lemme 3.6
Soit $c > 0$ et $\xi > 1$ des constantes et P un polynôme réel vérifiant les propriétés suivantes :

1. *pour tout entier $i \in [n]$, $\left|P(\xi^i)\right| \leq 1$, et*
2. *il existe un réel $x_0 \in [1;\xi]$ tel que $|P'(x_0)| \geq c$.*

Alors $\deg(P) = \Omega(n)$, *soit plus précisément :*

$$\deg(P) \geq \min\left(\frac{n}{2}, \frac{\log_2\left(\xi^{n+3}c\right) - 1}{\log_2\left(\frac{\xi^3}{\xi-1}\right) + 1}\right).$$

Preuve : Soit d le degré de P ; supposons $d \leq \frac{n}{2}$, puisque dans le cas contraire il n'y a rien à prouver.

Comme les polynômes P' and P'' sont respectivement de degré $d-1$ et $d-2$, il existe un entier $a \in [n-2d+2; n-1]$ tel que P'' n'ait pas de racine réelle dans l'intervalle $\left]\xi^a; \xi^{a+1}\right[$, et que P' n'ait pas de racine dont la partie réelle soit dans ce même intervalle. Les bornes choisies pour a font en particulier que $\xi^a \geq \xi^2$.

Dans la première partie de cette preuve, nous allons donner une borne supérieure sur $\left|P'\left(\frac{1+\xi}{2}\xi^a\right)\right|$, grâce au fait que $\frac{1+\xi}{2}\xi^a$ est au milieu de l'intervalle $\left(\xi^a;\xi^{a+1}\right)$, où P' est monotone.
Dans la seconde partie de la preuve nous donnerons une borne inférieure sur $\left|P'\left(\frac{1+\xi}{2}\xi^a\right)\right|$ faisant intervenir d avec un exposant négatif, en utilisant le fait que P' n'a pas de racine dont la partie réelle soit dans $\left(\xi^a;\xi^{a+1}\right)$. Les deux parties combinées donneront directement le résultat attendu.
Commençons donc par prouver ceci :

$$\left|P'\left(\frac{1+\xi}{2}\xi^a\right)\right| \leq \frac{4}{\xi^a(\xi-1)}$$

Le polynôme P, sur l'intervalle $\left]\xi^a;\xi^{a+1}\right[$, est soit convexe, soit concave, puisque P'' n'y a pas de racine. Les deux cas étant bien entendu similaires, il suffit d'en traiter un seul : supposons pour l'exemple que P est convexe, et considérons sa tangente t au milieu de l'intervalle (voir Figure 3.1).

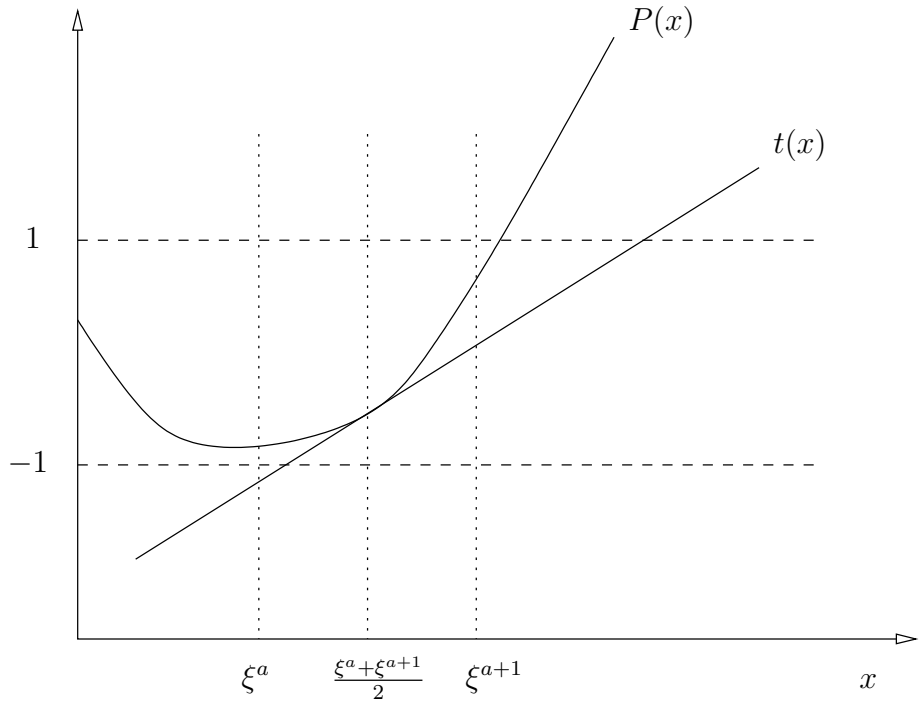

FIG. 3.1 – P et sa tangente t en $\frac{\xi^a+\xi^{a+1}}{2}$

Considérons la variation de t sur l'intervalle $\left[\xi^a;\xi^{a+1}\right]$. Là encore, il y a deux cas : soit la dérivée de P en $\frac{\xi^a+\xi^{a+1}}{2}$ est strictement

positive, soit elle est strictement négative, et dans chaque cas cela s'étend à tout l'intervalle $\left]\xi^a;\xi^{a+1}\right[$. Si elle est strictement positive, on doit avoir $P\left(\frac{\xi^a+\xi^{a+1}}{2}\right) \geq -1$. Comme de plus $t \leq P\left(\xi^{a+1}\right) \leq 1$, la variation de t sur l'intervalle $\left[\xi^a;\xi^{a+1}\right]$ vaut au plus 4. De même, si P' est négative sur $\left]\xi^a;\xi^{a+1}\right[$, on doit avoir $P\left(\frac{\xi^a+\xi^{a+1}}{2}\right) \leq 1$ et $t(\xi^a) \geq -1$, dont la variation en valeur absolue de t sur l'intervalle $\left[\xi^a;\xi^{a+1}\right]$ vaut au plus 4. Dans tous les cas, on a donc

$$\left|P'\left(\frac{1+\xi}{2}\xi^a\right)\right| \leq \frac{4}{\xi^a(\xi-1)}$$

On en déduit une borne supérieure sur la valeur absolue du quotient $\frac{P'\left(\frac{1+\xi}{2}\xi^a\right)}{P'(x_0)}$:

$$\left|\frac{P'\left(\frac{1+\xi}{2}\xi^a\right)}{P'(x_0)}\right| \leq \frac{4}{c\xi^a(\xi-1)} \leq \frac{4}{c\xi^{n-2d+2}(\xi-1)}. \tag{3.1}$$

Nous allons maintenant établir cette borne inférieure :

$$\left|\frac{P'\left(\frac{1+\xi}{2}\xi^a\right)}{P'(x_0)}\right| \geq \left(\frac{\xi-1}{2\xi}\right)^{d-1} \tag{3.2}$$

Pour cela, commençons par écrire la dérivée de P sous la forme

$$P'(X) = \lambda\prod_{i=1}^{d-1}(X-\alpha_i),$$

où les α_i sont des nombres complexes. On a l'égalité suivante :

$$\left|\frac{P'\left(\frac{1+\xi}{2}\xi^a\right)}{P'(x_0)}\right| = \prod_{i=1}^{d-1}\left|\frac{\frac{1+\xi}{2}\xi^a-\alpha_i}{x_0-\alpha_i}\right| \tag{3.3}$$

Il s'agit maintenant de borner inférieurement la fonction f ainsi définie :

$$f : \left(\begin{array}{rcl} \mathbb{R}\setminus\left(\{x_0\}\cup\left(\xi^a;\xi^{a+1}\right)\right) & \to & \mathbb{R} \\ x & \mapsto & \left|\frac{\frac{1+\xi}{2}\xi^a-x}{x_0-x}\right| \end{array}\right).$$

Pour x dans $\mathbb{R}\setminus\left(\{x_0\}\cup\left(\xi^a;\xi^{a+1}\right)\right)$, on a, la figure 3.2 est là pour nous en convaincre,

$$f(x) \geq \min(1, f(\xi^a), f(\xi^{a+1})) \geq \frac{\xi - 1}{2\xi}.$$

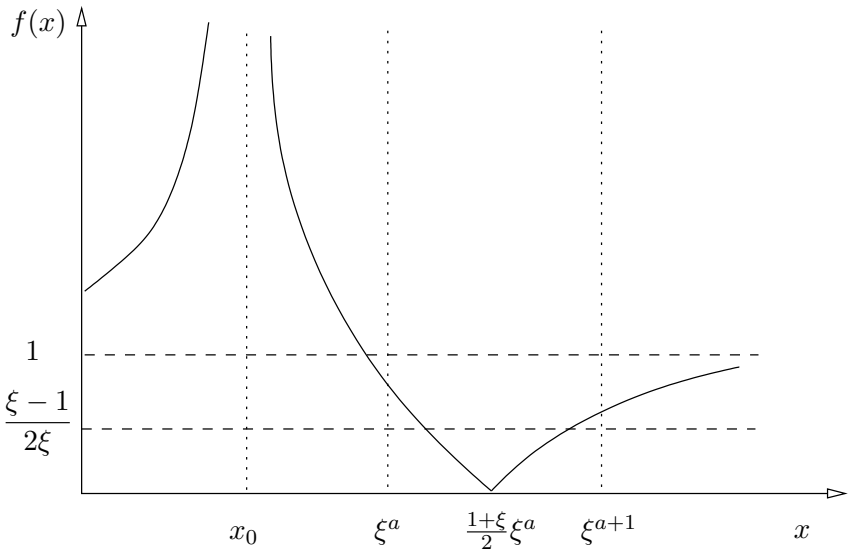

FIG. 3.2 – Tableau de variation de f

Souvenons-nous que, par hypothèse, aucun des a_i n'a sa partie réelle dans $\left]\xi^a; \xi^{a+1}\right[$, ce qui fait que l'on a, pour tout i,

$$f(\Re(\alpha_i)) \geq \frac{\xi - 1}{2\xi}. \tag{3.4}$$

Pour en déduire une borne inférieure sur $f(\alpha_i)$ il suffit d'être conscient de ce fait géométrique élémentaire :
Soit MBC un triangle, M' le projeté orthogonal de M sur (BC), et (d) la médiatrice de $[BC]$.
Si M est « à gauche de » (d), autrement dit si $MC \geq MB$, alors, bien sûr, $\frac{MC}{MB} \geq 1$. Considérons maintenant le cas où M est « à droite de » (d), c'est-à-dire où $MC \leq MB$ (voir figure 3.3). Comme C est plus près de la droite (MM') que ne l'est B,

$$\tan\alpha = MM'/BM' \leq \tan\beta = MM'/CM'.$$

Ainsi, $\alpha \leq \beta$, donc $\cos\alpha \geq \cos\beta$, ce qui signifie $\frac{MC}{MB} \geq \frac{M'C}{M'B}$. Finalement :

$$\frac{MC}{MB} \geq \min\left(1, \frac{M'C}{M'B}\right) \tag{3.5}$$

En appliquant cette inégalité aux points $M = \alpha_i$, $M' = \Re(\alpha_i)$, $B = x_0$ et $C = \frac{1+\xi}{2}\xi^a$, on obtient ceci :

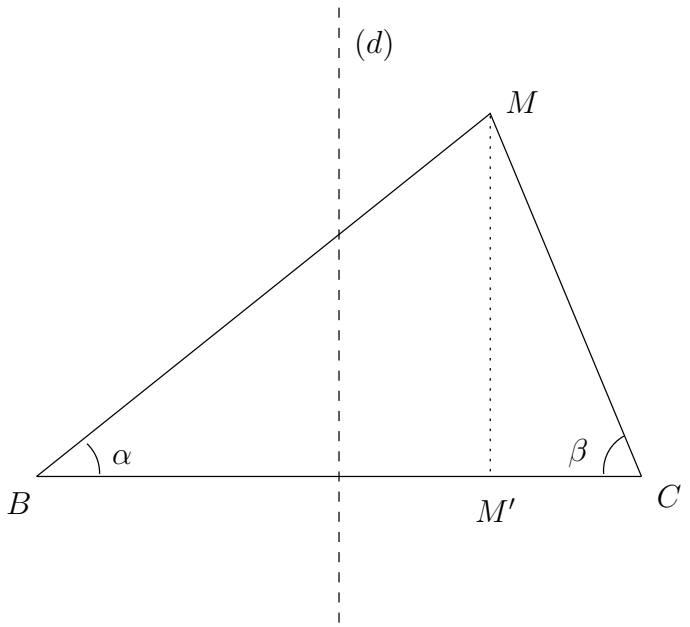

FIG. 3.3 – La trigonométrie pour les nuls

$$\left|\frac{\frac{1+\xi}{2}\xi^a-\alpha_i}{x_0-\alpha_i}\right| \geq \min\left(1, \left|\frac{\frac{1+\xi}{2}\xi^a-\Re(\alpha_i)}{x_0-\Re(\alpha_i)}\right|\right),$$

et donc $f(\alpha_i) \geq \frac{\xi-1}{2\xi}$, d'après (3.4). En ajoutant (3.3), on conclut que

$$\left|\frac{P'\left(\frac{1+\xi}{2}\xi^a\right)}{P'(x_0)}\right| \geq \left(\frac{\xi-1}{2\xi}\right)^{d-1}.$$

Enfin, en prenant en compte (3.1), on obtient l'inégalité

$$\left(\frac{\xi-1}{2\xi}\right)^{d-1} \leq \frac{4}{c\xi^{n-2d+2}(\xi-1)},$$

soit

$$d \geq \frac{\log_2\left(\xi^{n+3}c\right)-1}{\log_2\left(\frac{\xi^3}{\xi-1}\right)+1}.$$

□

3.3 Le problème du sous-groupe caché dans $(\mathbb{Z}_p)^n$

3.3.1 Définitions et énoncé du résultat

Nous allons en fait considérer non pas exactement le problème du sous-groupe caché dans $(\mathbb{Z}_p)^n$, mais un affaiblissement de celui-ci.

Définition 3.7
Pour un nombre premier p et un entier positif n, $\mathbf{wHSP}_{(\mathbb{Z}_p)^n}$ est défini par le quintuplet $((\mathbb{Z}_p)^n, [p^n], \mathbb{B}, \mathscr{S}, f)$, où

- *$\mathscr{S}$ est l'ensemble des applications de G dans $[|G|]$ cachant un sous-groupe de G d'ordre 1 ou p, et*
- *pour $\gamma \in \mathscr{S}$, $f(\gamma)$ vaut $\top$ si γ cache le sous-groupe trivial, c'est-à-dire est injective, $\bot$ sinon.*

$\mathbf{wHSP}_{(\mathbb{Z}_p)^n}$ étant par définition isomorphe à un sous-problème élémentaire de $\mathbf{HSP}_{(\mathbb{Z}_p)^n}$, pour toute probabilité d'erreur ε, sa complexité en requêtes est inférieure ou égale à celle de $\mathbf{HSP}_{(\mathbb{Z}_p)^n}$. Comme nous comptons prouver une borne inférieure sur la complexité en requêtes de $\mathbf{wHSP}_{(\mathbb{Z}_p)^n}$, la même borne sera valable *a fortiori* pour $\mathbf{HSP}_{(\mathbb{Z}_p)^n}$. La preuve que nous allons donner, qui inclut le lemme 3.6, reprend les résultats développés successivement dans [KNP], [KNP05a] et [KNP05b]. Toutes les idées sont dans [KNP], qui ne traite cependant que le cas $p = 2$. La généralisation au cas où p est un nombre premier quelconque ne nécessite pas de réflexion particulière, simplement des calculs quelque peu plus pénibles.

Cette version affaiblie du problème du sous-groupe caché est plus proche du problème de Simon original que l'on trouve dans le papier originel [Sim97], où il s'agissait, dans $(\mathbb{Z}_2)^n$, de déterminer le sous-groupe caché en ayant la promesse qu'il était d'ordre 2. On peut remarquer au passage que Daniel Simon donne dans cet article une version de type « Las Vegas » de son algorithme, dont la complexité en requêtes moyenne est en $\mathcal{O}(n)$ — nous ne nous apesantirons pas sur cette remarque car elle s'exprime mal dans le formalisme que nous avons défini, qui n'a que faire de la complexité en requêtes « moyenne ». Mieux, Gilles Brassard et Peter Høyer ont donné dans [BH97] un algorithme quantique exact, c'est-à-dire qui a une probabilité d'erreur nulle, qui fonctionne en temps polynomial — en particulier sa complexité en requêtes est polynomiale.

Fixons dorénavant une valeur $\varepsilon \in \left[0; \frac{1}{2}\right[$ représentant la probabilité d'erreur maximale des algorithmes considérés.

Définition 3.8
Soit $T_\varepsilon(p, n)$ la complexité en requêtes quantique de $\mathbf{wHSP}_{(\mathbb{Z}_p)^n}$ pour une probabilité d'erreur au plus ε.

Le résultat que nous entendons prouver est le suivant :

Proposition 3.9

$$T_\varepsilon(p,n) \geq \min\left(\frac{n}{4}, \frac{\log_2\left((2-4\epsilon)\frac{p^{n+3}}{p-1}\right)-1}{2\log_2\left(\frac{p^3}{p-1}\right)+2}\right).$$

Ce n'est pas joli, et le lecteur préférera sans doute cet énoncé, moins précis mais plus esthétique :

Proposition 3.10
$T_\varepsilon(p,n) = \Omega(n)$.

Preuve : Étant donné l'allure peu avenante de l'énoncé de la proposition 3.9, il n'apparaîtra peut-être pas immédiat à tout à chacun que la proposition 3.10 en est un corollaire ; pourtant c'est bien le cas. En effet, commençons par remarquer que l'on a ces quatretrois inégalités : $\frac{p^{n+3}}{p-1} \geq p^{n+2}$, $\frac{p^3}{p-1} \leq 2p^2$, $1+\log_2 p \leq 2\log_2 p$ et $\log_2(2-4\varepsilon) \leq 1$. On déduit alors de la proposition 3.9 que l'on a

$$\begin{aligned}
T_\varepsilon(p,n) &\geq \min\left(\tfrac{n}{4}, \tfrac{\log_2\left((2-4\epsilon)p^{n+2}\right)-1}{2\log_2(2p^2)+2}\right)\\
&\geq \min\left(\tfrac{n}{4}, \tfrac{\log_2(2-4\epsilon)+(n+2)\log_2 p-1}{4(1+\log_2 p)}\right)\\
&\geq \min\left(\tfrac{n}{4}, \tfrac{\log_2(2-4\epsilon)+(n+2)\log_2 p-1}{8\log_2 p}\right)\\
&\geq \min\left(\tfrac{n}{4}, \tfrac{n}{8}+\tfrac{\log_2(2-4\varepsilon)-1}{8\log_2 p}\right)\\
T_\varepsilon(p,n) &\geq \min\left(\tfrac{n}{4}, \tfrac{n}{8}+\tfrac{\log_2(2-4\varepsilon)-1}{8}\right).
\end{aligned}$$

$T_\varepsilon(p,n)$ est donc asymptotiquement supérieur à $\frac{n}{8}+C(\varepsilon)$, et ce de manière indépendante de p — pas de ε, bien entendu, mais rappelons que nous avons fixé sa valeur en début de section. □

3.3.2 Preuve de la proposition 3.9

Soit maintenant $\mathscr{A}$ un algorithme quantique pour le problème élémentaire $\mathbf{wHSP}_{(\mathbb{Z}_p)^n}$ ayant une probabilité d'erreur au plus ε et T sa complexité en requêtes. Pour x une application de $(\mathbb{Z}_p)^n$ dans $[p^n]$, soit $\mathbb{P}_{\mathscr{A}}(x)$ la probabilité que $\mathscr{A}$ réponde « $\top$ » sur l'entrée x.

D'après le théorème 3.2, il existe un ensemble S de fonctions partielles de $(\mathbb{Z}_p)^n$ dans $[p^n]$ et une famille de nombres réels $(\alpha_s)_{s\in S}$ tels que :

- pour toute application x de $(Z_p)^n$ dans $[p^n]$, $\mathbb{P}_{\mathscr{A}}(x) = \sum_{s\in S} \alpha_s I_s(x)$ et
- pour tout $s \in S$, $|\mathrm{dom}(s)| \leq 2T$.

Notre but à présent est d'appliquer une transformation idoine à la fonction $\mathbb{P}_{\mathscr{A}}$ afin d'en faire une fonction plus simple à analyser qu'un polynôme

en $(p^n)^{p^n}$ variables. Pour cela, nous allons la *symétriser* — un terme qui prendra toute sa signification dans la section 4.1.2.

Dans un abus de notation patent mais bien pratique, d désignera dans la suite un entier de $[n+1]$, et D sera p^d ; réciproquement, d désignera donc $\log_p D$.

Soit Q l'application de $\{p^0, p^1, \ldots, p^n\}$ dans $[0;1]$ qui à D associe

$$Q(D) = \frac{1}{|X_D|} \sum_{\gamma \in X_D} \mathbb{P}_{\mathscr{A}}(\gamma) = \frac{1}{|X_D|} \sum_{\gamma \in X_D} \sum_{s \in S} \alpha_s I_s(\gamma),$$

où X_D est l'ensemble des applications de $(\mathbb{Z}_p)^n$ dans $[p^n]$ cachant un sous-groupe d'ordre D. Bien sûr, Q n'est définie que sur quelques puissances de p, et peut être étendu à tout le domaine réel de bien des façons. Nous dirons que Q est un polynôme de degré δ s'il est la restriction d'un polynôme réel de degré δ. L'intérêt immédiat d'avoir défini Q de cette manière, outre le fait que le nombre de variables a été drastiquement réduit par rapport à $\mathbb{P}_{\mathscr{A}}$, est qu'il y a un saut entre $Q(1)$ et $Q(p)$.

Plus précisément, comme $\mathscr{A}$ est un algorithme pour $\mathbf{wHSP}_{(\mathbb{Z}_p)^n}$ de probabilité d'erreur au plus ε, pour tout γ cachant le sous-groupe trivial, c'est-à-dire pour tout γ injectif, on a $\mathbb{P}_{\mathscr{A}}(\gamma) \geq 1 - \varepsilon$, et pour tout γ cachant un sous-groupe d'ordre p, on a $\mathbb{P}_{\mathscr{A}}(\gamma) \leq \varepsilon$. On en déduit respectivement ces deux inégalités : $Q(1) \geq 1 - \varepsilon$ et $Q(p) \leq \varepsilon$. De plus, comme Q est défini comme une moyenne de probabilités, on doit avoir $Q(D) \in [0;1]$ pour tout $D \in \{p^0; p^1; \ldots; p^n\}$.

Reprenons l'expression de Q en fonction des I_s :

$$Q(D) = \frac{1}{|X_D|} \sum_{\gamma \in X_D} \sum_{s \in S} \alpha_s I_s(\gamma).$$

On peut aussi écrire cela de cette façon :

$$Q(D) = \sum_{s \in S} \alpha_s Q^s(D),$$

où, pour $s \in S$, $Q^s(D) = \frac{1}{|X_D|} \sum_{\gamma \in X_D} I_s(\gamma)$ est la proportion d'applications de $(\mathbb{Z}_p)^n$ dans $[p^n]$ étendant s parmi celles cachant un sous-groupe d'ordre D. La fonction Q est alors combinaison linéaire des Q^s, et nous allons borner le degré de Q en bornant le degré de chacun des Q^s, c'est-à-dire en montrant que chacune d'elles est la restriction d'un polynôme de degré borné.

Lemme 3.11
Soit n et k des entiers naturels, p un nombre premier. Le nombre de sous-groupes de $(\mathbb{Z}_p)^n$ d'ordre p^k vaut précisément

$$\beta_p(n,k) = \prod_{0 \leq i < k} \frac{p^{n-i} - 1}{p^{k-i} - 1}.$$

Preuve : Considérons l'espace d'un instant $(\mathbb{Z}_p)^n$ comme un espace vectoriel sur le corps $\mathbb{Z}_p$. De ce point de vue, les sous-groupes sont les sous-espaces vectoriels. D'après le fait 2.26, le nombre de k-uplets libres de $(\mathbb{Z}_p)^n$ vaut

$$\alpha_p(n,k) = \prod_{0 \leq i < k} \left(p^n - p^i\right).$$

Comme chaque sous-espace vectoriel de dimension k peut être engendré par $\alpha_p(k,k)$ différents k-uplets, le nombre de sous-espaces de dimension k est

$$\frac{\alpha_p(n,k)}{\alpha_p(k,k)} = \prod_{0 \leq i < k} \frac{p^{n-i} - 1}{p^{k-i} - 1}.$$

On peut remarquer que cette formule reste correcte même quand $k > n$, auquel cas $\alpha_p(n,k) = 0$. □

Proposition 3.12
Q est de degré au plus $2T$.

Comme $Q(D) = \sum_{s \in S} \alpha_s Q^s(D)$, il suffit de montrer que pour toutes les fonctions partielles $s : (\mathbb{Z}_p)^n \to E$ telles que $|\operatorname{dom}(s)| \leq 2T$, $Q^s(D)$ est (la restriction d'un polynôme) de degré au plus $2T$. Soit donc s une telle fonction partielle. Nous allons procéder en trois étapes. Nous allons d'abord examiner le cas où s est une fonction constante, puis celui où elle est injective, et enfin le cas général.

Lemme 3.13
Si la fonction partielle $s : (\mathbb{Z}_p)^n \to E$ est constante sur son domaine, alors Q^s est de degré au plus $|\operatorname{dom}(s)|$.

Preuve : Rappelons que, selon nos notations, $D = p^d$. Désignons par $a_0, \ldots, a_{k-1}$ les éléments constituant le domaine de s, les a_i étant bien entendu deux-à-deux distincts. Alors une fonction γ cachant un sous-groupe H étend s si et seulement si $\gamma(a_0) = s(a_0)$ et $\{a_i - a_0 / i = 1 \ldots k\} \subseteq H$.
On a donc $Q^s(D) = Q^{s'}(D)$, où s' est la fonction partielle de $(\mathbb{Z}_p)^n$ dans E définie par $s'(x) = s(x - a_0)$. On peut donc supposer, sans perte de généralité, que a_0 vaut 0. Comme E, l'ensemble des valeurs potentiellement prises par x, est de cardinal p^n, on a

$$Q^s(D) = \frac{\lambda(D,s)}{p^n},$$

où $\lambda(D,s)$ is la proportion, parmi les sous-groupes d'ordre D, de ceux qui contiennent $\operatorname{dom}(s)$. Soit H' le sous-groupe engendré

par $\mathrm{dom}(s)$ et $D' = p^{d'}$ son ordre, d' étant donc la dimension de H' en tant que sous-espace vectoriel de $(\mathbb{Z}_p)^n$. Comme $a_0 = 0$, on a $d' \leq |\mathrm{dom}(s)| - 1$. Le nombre de sous-groupes de $(\mathbb{Z}_p)^n$ d'ordre D contenant H' est égal au nombre de sous-groupes d'ordre $\frac{D}{D'}$ de $(\mathbb{Z}_p)^n / H'$, qui est isomorphe à $(\mathbb{Z}_p)^{n-d'}$; d'après le lemme 3.11, il y en a donc $\beta(n-d', d-d')$. Par conséquent,

$$\begin{aligned} Q^s(D) &= \tfrac{1}{p^n} \tfrac{\beta(n-d',d-d')}{\beta(n,d)} \\ &= \tfrac{1}{p^n} \prod_{0 \leq i < d'} \tfrac{p^{d-i}-1}{p^{n-i}-1} \\ Q^s(D) &= \tfrac{1}{p^n} \prod_{0 \leq i < d'} \tfrac{\frac{D}{p^i}-1}{p^{n-i}-1}. \end{aligned}$$

Q^s est donc un polynôme en D de degré $d' < |\mathrm{dom}(s)|$. □

Lemme 3.14
Si la fonction partielle $s : (\mathbb{Z}_p)^n \to E$ est injective, alors Q^s est de degré au plus $|\mathrm{dom}(s)|$.

Preuve : Nous continuerons d'utiliser les mêmes notations : $D = p^d$ et $\mathrm{dom}(s) = \{a_i / i \in [k]\}$ avec $k = |\mathrm{dom}(s)|$. Une application $x : (\mathbb{Z}_p)^n \to [p^n]$ cachant un sous-groupe H est une extension de s si et seulement si les a_i se trouvent dans des classes à gauche deux-à-deux distinctes de H et x prend des valeurs appropriées en les a_i. Or, les a_i sont dans des classes à gauches deux-à-deux distinctes de H si et seulement si H ne contient aucun des $a_i - a_j$ pour $i \neq j$. On a donc $Q^s(D) = \nu^s(D)\lambda^s(D)$, où

- $\lambda^s(D)$ est la proportion de sous-groupes de G, ne contenant aucun des $a_i - a_j$ pour $i \neq j$, parmi ceux qui sont d'ordre D, et
- $\nu^s(D)$ est la proportion des applications étendant s parmi celles cachant un sous-groupe d'ordre D ne contenant aucun des $a_i - a_j$ pour $i \neq j$.

Commençons par calculer $\nu^s(D)$. Pour chaque sous-groupe H d'ordre D, le nombre d'applications de $(\mathbb{Z}_p)^n$ dans $[p^n]$ cachant H se monte à $p^n(p^n-1)\cdots(p^n-p^{n-d}+1)$. En effet, cela revient à choisir librement une valeur pour chaque classe à gauche de H, avec la seule contrainte qu'elles doivent être deux à deux distinctes. Supposons que H ne contiennent aucun des $a_i - a_j$ pour $i \neq j$. Alors le nombre d'applications de $(\mathbb{Z}_p)^n$ dans $[p^n]$ cachant H et dont s est une restriction vaut $(p^n-k)(p^n-k-1)\ldots(p^n-p^{n-d}+1)$, puisque cette fois-ci on ne choisit librement la valeur de l'application que sur les classes à gauche de H ne

contenant aucun des a_i. Il en résulte

$$\nu^s(D) = \frac{(p^n - k)!}{(p^n)!}.$$

$\lambda^s(D)$ vaut $1-\mu^s(D)$, où $\mu^s(D)$ est la proportion de sous-groupes contenant au moins un des $a_i - a_j$ pour $i \neq j$, parmi ceux qui sont d'ordre D. C'est un truisme, mais un truisme utile, car par la classique formule d'inclusion-exclusion, on en déduit que l'on peut développer $\lambda^s(D)$ de la façon suivante :

$$\lambda^s(D) = 1 - \left(\begin{array}{ll} & \displaystyle\sum_{i \neq j} \Pr(a_i - a_j \in H) \\ - & \displaystyle\sum_{\substack{i_1 \neq j_1 \\ i_2 \neq j_2 \\ \{i_1;j_1\} \neq \{i_2;j_2\}}} \Pr\left(\begin{array}{ll} & a_{i_1} - a_{j_1} \in H \\ \wedge & a_{i_2} - a_{j_2} \in H \end{array} \right) \\ + & \cdots \\ - & \cdots \\ & \vdots \\ + & \Pr(\forall i \neq j \; a_i - a_j \in H) \end{array} \right)$$

D'après le lemme 3.13, dans cette somme, chaque terme est un polynôme en D de degré au plus d', où l'on définit d' en posant que le sous-groupe de $(\mathbb{Z}_p)^n$ engendré par les $a_i - a_j$ est d'ordre $p^{d'}$. Comme, pour tout $i, j \in [k]$, $a_i - a_j$ appartient au sous-groupe engendré par les $a_l - a_0$, on a $d' < |\mathrm{dom}(s)|$. □

Nous avons maintenant réuni suffisamment d'éléments pour prouver la proposition 3.12.

Preuve de la proposition 3.12 :

En tout généralité, une fonction partielle s est définie par un système d'égalités de la forme

$$\left\{ \begin{array}{l} s\left(a_0^0\right) = s\left(a_1^0\right) = \cdots = s\left(a_{k_1-1}^0\right) = b_0 \\ s\left(a_0^1\right) = s\left(a_1^1\right) = \cdots = s\left(a_{k_2-1}^1\right) = b_1 \\ \vdots \\ s\left(a_0^{l-1}\right) = s\left(a_1^{l-1}\right) = \cdots = s\left(a_{k_{l-1}-1}^{l-1}\right) = b_{l-1} \end{array} \right.$$

où les $b_0, \ldots, b_{l-1}$ sont deux-à-deux distincts. Comme précédemment, nous pouvons supposer sans perte de généralité que l'on a $a_0^1 = 0$, et nous allons le faire sans vergogne. De plus, comme $\gamma(a_i^j) = \gamma(a_0^j)$ est équivalent à $\gamma(a_i^j - a_0^j) = \gamma(0)$ — les deux étant équivalents à « a_i^j est a_0^j sont dans la même classe à gauche de

H », on peut retirer chacun des a_i^j, pour $i, j > 0$, du domaine de s, pour les remplacer par $a_i^j - a_0^j$, où l'on fixe la valeur de s à b_0. En procédant de la sorte on n'augmente pas le domaine de s. Il peut cependant arriver que l'on définisse alors s de manière contradictoire, en imposant à s de prendre la valeur b_0 sur l'entrée $a_i^j - a_0^j$, où s était peut-être déjà déffinie. Le cas échéant, cela signifie tout simplement que s n'est la restriction d'aucune application cachant un sous-groupe, et que par conséquent Q^s est nulle sur les points où elle est définie, et est donc de degré $-\infty$, ce qui est bien conforme à l'énoncé de la proposition 3.12. Nous pouvons donc évacuer ce cas l'esprit tranquille, et supposer que nous pouvons définir s de cette nouvelle façon de manière non contradictoire. On se ramène alors à un système de conditions de cette forme :

$$\begin{cases} s(0) = s\left(a_1^0\right) = \cdots = s\left(a_{k_1-1}^0\right) = b_0 \\ s(a^1) = b_1 \\ \vdots \\ s\left(a^{l-1}\right) = b_{l-1} \end{cases}$$

avec $k_1 + l \leq |\operatorname{dom}(s)|$. $Q^s(D)$ est alors la proportion $P_1(D)$, parmi les applications $\gamma : (\mathbb{Z}_p)^n \to [p^n]$ cachant un sous-groupe d'ordre D, de celles vérifiant $\gamma(0) = \gamma(a_1^0) = \cdots = \gamma\left(a_{k_1-1}^0\right) = b_0$, multipliée par la proportion $P_2(D)$, parmi celles vérifiant ces égalités, de celles étendant s. Or nous avons déjà calculé la première de ces proportions dans le lemme 3.13 puisqu'il s'agit tout bêtement du cas où s est une fonction constante. Soit H' le sous-groupe de $(\mathbb{Z}_p)^n$ engendré par les a_i^0. H' est d'ordre $D' = p^{d'}$, où $d' < k_0$, et d'après le lemme 3.13 on a

$$P_1(D) = \frac{1}{p^n} \prod_{0 \leq i < d'} \frac{p^{d-i} - 1}{p^{n-i} - 1}.$$

On définit s' sur G/H' comme étant le quotient de s, c'est-à-dire qu'on a $\operatorname{dom}(s') = \{a \cdot H' / a \in \operatorname{dom}(s)\}$ et pour tout $a \in \operatorname{dom}(s)$, $s'(a \cdot H') = s(a)$. Encore une fois, s'il n'est pas possible de définir s' de cette façon, c'est que Q^s est le polynôme nul et donc que la preuve se termine bien abruptement. Supposons donc que s' peut bien être défini de cette manière. Si une application γ vérifie $\gamma(0) = \gamma\left(a_1^0\right) = \cdots = \gamma\left(a_{k_1-1}^0\right) = b_0$, alors on peut définir γ' sur G/H' comme étant le quotion de γ, de la même manière que l'on vient de définir s' à partir de s.

L'assertion « γ est une extension de s et cache un sous-groupe d'ordre D » est alors équivalente à « γ' est une extension de s'

et cache un sous-groupe d'ordre D/D' ». Puisque s' est définie par les égalités

$$s'(H') = b_0, s'(a^1 + H') = b_1, \ldots, s'\left(a^{l-1} + H'\right) = b_{l-1}$$

et est injective, le lemme 3.14 montre que $P_2(D) = Q^{s'}(D/D')$ est la restriction d'un polynôme en D de degré au plus $l-1$. Par conséquent, Q^s est de degré au plus $(k_1 - 1) + (l-1) < |\operatorname{dom}(s)| \leq 2T$. □

Il ne reste plus qu'à recoller les bouts pour réaliser que l'on a effectivement démontré la proposition 3.9. Résumons donc ce qui a été fait à ce point.

Preuve de la proposition 3.9 :

Nous avons commencé par choisir un algorithme quantique $\mathscr{A}$ pour le problème élémentaire $\mathbf{wHSP}_{(\mathbb{Z}_p)^n}$ de complexité en requêtes T ayant une probabilité d'erreur au plus ε. Cela nous permet de construire une fonction Q définie sur $\left\{p^i / i \in [n+1]\right\}$, qui vérifie les propriétés suivantes :

- pour tout entier $i \in [n+1]$, $0 \leq Q\left(p^i\right) \leq 1$,
- $Q(1) \geq 1 - \varepsilon$,
- $Q(p) \leq \varepsilon$ et
- Q peut être interpolée par un polynôme de degré au plus $2T$ (d'après la proposition 3.12).

Soit P un polynôme de degré au plus $2T$ interpolant Q. Comme $P(1) \geq 1-\varepsilon$, il existe $x_0 \in [1;p]$ tel que $|P'(x_0)| \geq \frac{1-2\varepsilon}{p-1}$. D'après le lemme 3.6, on a

$$\deg(P) \geq \min\left(\frac{n}{2}, \frac{\log_2\left((1-2\varepsilon)\frac{p^{n+3}}{p-1}\right) - 1}{\log_2\left(\frac{p^3}{p-1}\right) + 1}\right).$$

On déduit ensuite immédiatement le résultat voulu de l'inégalité $\deg(P) \leq 2T$. □

Avant de conclure, nous allons écrire noir sur blanc la constatation très naturelle et très simple du fait que le problème du sous-groupe caché est de difficulté croissante : il est plus difficile sur un groupe G que sur ses sous-groupes.

Fait 3.15

*Soit G un groupe fini, H un sous-groupe de G, et $\mathscr{A}$ un algorithme déterministe (*resp. probabiliste, quantique*) pour le problème élémentaire $\mathbf{HSP}_G$ ayant une probabilité d'erreur au plus ε. Alors il existe un algorithme déterministe (*resp. probabiliste, quantique*) pour $\mathbf{HSP}_H$ de même complexité en requêtes que $\mathscr{A}$ et ayant une probabilité d'erreur au plus ε.*

Preuve : Choisissons un ensemble $\{t_i / i \in [|G/H|]\}$ de représentants dans G de G/H, de sorte que $G/H = \{t_i \cdot H / i \in [|G/H|]\}$ et les $t_i \cdot H$ sont deux à deux distincts, avec $t_0 = 0$. À toute application $\gamma : H \to [|H|]$ on associe $\gamma' : G \to [|G|]$ définie par $\gamma'(t_i \cdot h) = \gamma(h) + i\,|H|$ pour $h \in H$ et $i \in [|G/H|]$. Si γ cache un sous-groupe K de H, alors γ' cache ce même sous-groupe K, mais dans G.

En effet, supposons que l'on ait $\gamma'(g) = \gamma'(g')$. Il existe une unique décomposition $g = t_i \cdot h$ et $g' = t_{i'} \cdot h'$ vérifiant $h, h' \in H$; on doit donc avoir $i = i'$ et $\gamma(h) = \gamma(h')$, ce qui signifie qu'il existe $k \in K$ tel que $h' = h \cdot k$. Par suite, on a $g' = g \cdot k$. La réciproque se montre plus aisément encore.

Ainsi, l'algorithme $\mathscr{A}$ peut être utilisé quasiment tel quel pour résoudre le problème du sous-groupe caché dans H. Il suffit pour cela d'intercepter chacun de ces appels à la fonction boîte noire, et de les modifier comme suit : quand $\mathscr{A}$ lance une requête sur l'entrée $g \in G$, on décompose g sous la forme $t_i \cdot h$ avec $h \in H$, on lance plutôt la requête h, puis on intercepte le résultat renvoyé par la boîte noire, qui est $\gamma(h)$, pour lui ajouter $i\,|H|$. Ainsi $\mathscr{A}$ croit bien être mis en contact avec une boîte noire recelant la fonction γ', alors qu'il ne s'agit que de γ, mais après tout ce n'est qu'une machine. L'essentiel est que, par hypothèse, $\mathscr{A}$ va correctement identifier le sous-groupe caché par γ' avec une probabilité d'erreur au plus ε, et comme ce sous-groupe est aussi celui caché par γ, nous n'avons plus besoin d'intervenir. Pour résumer, il nous a suffit de modifier légèrement $\mathscr{A}$ en ajoutant un petit module avant et après chaque porte de requête. □

Il est enfin temps de réunir les résultats des propositions 2.27 et 3.9 en une forme moins informative mais plus compacte et élégante. Rappelons d'abord que $r(G)$ désigne le rang du groupe G (voir la définition 2.21).

Théorème 3.16
La complexité en requêtes quantique du problème du sous-groupe caché sur un sous-groupe abélien G est en $\Theta(r(G))$.

Rappelons la remarque faite à la fin de la section 1.1. La grandeur asymptotique $\Theta(r(G))$ doit être comprise comme étant atteinte lorsque G est assez grand, et non seulement lorsque $r(G)$ est assez grand. Comme il existe un nombre fini de groupes d'une taille inférieure à un entier donné et que le seul groupe sur lequel le problème du sous-groupe caché se résoud sans faire aucune requête est le groupe trivial, ce théorème est équivalent à : « il existe des constantes $0 < m < M$ telles que pour tout groupe abélien G non trivial, la complexité en requêtes quantique du problème du sous-groupe caché sur G est comprise entre $m \cdot r(G)$ et $M \cdot r(G)$ ».

Preuve du théorème 3.16 : Rappelons qu'en l'absence de précision sur la probabilité d'erreur, on suppose que l'on borne celle-ci à $\frac{1}{3}$ (voir la définition 1.21). La borne supérieure $O(r(G))$ est une conséquence immédiate de la proposition 2.27. La borne inférieure quant à elle, est une conséquence immédiate de la proposition 3.9 et du fait 3.15 une fois remarqué que pour tout groupe abélien fini G il existe un nombre premier p tel que l'un des sous-groupes de G est isomorphe à $(\mathbb{Z}_p)^{r(G)}$. □

Il pourrait sembler à première vue que ce résultat ne dit rien sur les groupes abéliens. Pas explicitement, c'est vrai. Mais, via le fait 3.15, il permet tout de même de donner une borne en $\Omega(\tilde{r}(G))$, où, pour un groupe fini, $\tilde{r}(G)$ est défini comme le rang maximal des sous-groupes commutatifs de G. Par exemple, le groupe symétrique $\mathfrak{S}_n$ contient $\lfloor n/2 \rfloor$ transpositions à supports deux-à-deux disjoints, qui engendrent un sous-groupe isomorphe à $(\mathbb{Z}_2)^{\lfloor n/2 \rfloor}$; on peut dès lors en déduire que la complexité en requêtes quantique du problème de décision associé à $\mathbf{HSP}_{\mathfrak{S}_n}$ — c'est-à-dire déterminer si le sous-groupe caché est trivial ou non — est en $\Omega(n)$, sachant que l'on sait par ailleurs qu'elle est en $\mathcal{O}(n \log(n))$ d'après un résultat de Ettinger, Høyer et Knill. En effet, il est démontré dans [EHK04] que la complexité en requêtes du problème de décision associé au problème du sous-groupe caché dans G est en $\mathcal{O}(\log |G|)$. Pour le cas de $\mathfrak{S}_n$, la marge d'incertitude n'est donc pas catastrophique, puisque le rapport de la meilleure borne supérieure connue sur la meilleure borne inférieure connue est de l'ordre du double logarithme de la taille du groupe. On peut faire bien pire, la palme étant décernée aux groupes diédraux, pour lesquels aucune borne inférieure non constante n'est connue à ce jour.

Chapitre 4

Symétries et complexité probabiliste

4.1 Symétries

4.1.1 Définitions

Soit I, J, R trois ensembles fixés. $\mathfrak{S}_I \times \mathfrak{S}_J$ agit sur J^I de la façon suivante :

$$(\sigma, \tau) \cdot x = \tau \circ x \circ \sigma^{-1} =_{\text{def}} x_\sigma^\tau$$

Dans la suite, on omettra souvent, pour alléger les notations, le symbole de composition, pour noter $\tau x \sigma^{-1}$, ou bien encore x_σ^τ. Il s'agit en effet d'une action de groupe puisque

$$\left(\sigma', \tau'\right) \cdot \left((\sigma, \tau) \cdot x\right) = \tau' \left(\tau x \sigma^{-1}\right) \sigma'^{-1} = \left(\tau' \tau\right) x \left(\sigma' \sigma\right)^{-1} = \left(\sigma' \sigma, \tau' \tau\right) \cdot x.$$

$\mathfrak{S}_I \times \mathfrak{S}_J \times \mathfrak{S}_R$ agit sur l'ensemble des problèmes élémentaires de la forme $(I, J, R, \mathscr{S}, f)$ via la loi

$$(\sigma, \tau, \pi) \cdot (I, J, R, \mathscr{S}, f) = \left(I, J, R, (\sigma, \tau) \cdot \mathscr{S}, \pi \circ f_\sigma^\tau\right),$$

où pour tout $x \in (\sigma, \tau) \cdot \mathscr{S}$, $f_\sigma^\tau(x) = f\left(x_{\sigma^{-1}}^{\tau^{-1}}\right)$.

On notera $\mathscr{P}_{\sigma,\tau}^{\pi}$ ce problème élémentaire $(I, J, R, \mathscr{S}', f')$.

Définition 4.1
Un **automorphisme** *d'un problème élémentaire $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ est un élément (σ, τ, π) de $\mathfrak{S}_I \times \mathfrak{S}_J \times \mathfrak{S}_R$ tel que $\mathscr{P}_{\sigma,\tau}^{\pi} = \mathscr{P}$.*

Les automorphismes d'un problème élémentaire $\mathscr{P}$ forment un groupe que l'on note $\mathrm{Aut}(\mathscr{P})$.

Exemple 4.2 (recherche dans un tableau non trié)
Le problème de la recherche dans un tableau non trié de taille N, noté $\mathbf{RTNT}_N$, est le problème élémentaire $([N], [2], \mathbb{B}, \mathscr{S}, f)$, où

- $f(x) = \begin{cases} \bot & \text{si } \left|x^{-1}(0)\right| = N \\ \top & \text{si } \left|x^{-1}(0)\right| = N-1 \text{ et } \left|x^{-1}(1)\right| = 1 \end{cases}$ et
- $\mathscr{S} = \mathrm{dom}\,(f)$.

  Le problème de la recherche dans un tableau non trié, qui à $N \in \mathbb{N}^*$ associe $\mathbf{RTNT}_N$ est bien sûr sobrement noté **RTNT**.

On est donc face à un tableau dont les entrées contiennent toutes « 0 », sauf peut-être une qui contiendrait « 1 », et il faut déterminer si c'est le cas.

Quant il s'agit de chercher un élément dans un tableau non trié de taille N, peu importe l'ordre des éléments du tableau puisque celui-ci n'est justement pas supposé trié. Autrement dit, $\mathrm{Aut}\,(\mathbf{RTNT}_N)$ contient $\mathfrak{S}_{[N]} \times \{\mathrm{id}\} \times \{\mathrm{id}\}$. Mais supposons que $\mathbf{RTNT}_N$ possède un automorphisme (σ, τ, π) tel que $\pi \neq \mathrm{id}$. Cela signifie que l'on peut passer d'un tableau ne contenant que des « 0 » à un tableau contenant un « 1 » au milieu de $N-1$ « 0 » par permutation des éléments de chaque tableau, et éventuellement en s'autorisant à remplacer tous les « 0 » d'un tableau par des « 1 » et vice-versa : cela est clairement impossible si $N > 1$. On réfute de même l'idée que $\mathbf{RTNT}_N$ puisse posséder un automorphisme (σ, τ, π) avec un τ non trivial, si $N > 2$. On en déduit que si $N > 2$,

$$\mathrm{Aut}\,(\mathbf{RTNT}_N) = \mathfrak{S}_{[N]} \times \{\mathrm{id}\} \times \{\mathrm{id}\}\,.$$

Exemple 4.3 (sous-groupe caché)
Considérons le problème du sous-groupe caché dans le groupe G , $\mathbf{HSP}_G = (G, [|G|]\,, S(G), \mathscr{S}, f)$ — voir pour rappel la définition 2.6.

Tout d'abord, il est évident que permuter les valeurs renvoyées par la boîte noire n'a aucune incidence. Autrement dit, $\mathrm{Aut}\,(\mathbf{HSP}_G)$ contient $\{0\} \times \mathfrak{S}_{[|G|]} \times \{0\}$.

Ensuite, soit σ un automorphisme de G et $\gamma \in \mathscr{S}$ cachant le sous-groupe H_γ. Clairement, $\gamma \circ \sigma^{-1}$ cache le sous-groupe $\sigma\,(H_\gamma)$. En effet,

$$\begin{array}{rcl} \gamma \circ \sigma^{-1}\,(g) = \gamma \circ \sigma^{-1}\,(g') & \iff & \exists h \in H_\gamma \quad \sigma^{-1}\,(g') = \sigma^{-1}\,(g) \cdot h \\ & \iff & \exists h \in H_\gamma \quad g' = g \cdot \sigma(h) \\ & \iff & \exists h \in \sigma\,(H_\gamma) \quad g' = g \cdot h \end{array}$$

Cela signifie que $(\sigma, \mathrm{id}, \tilde{\sigma})$ est un automorphisme de $\mathbf{HSP}_G$, où $\tilde{\sigma}$ est naturellement ainsi défini :

$$\tilde{\sigma} : \begin{pmatrix} S(G) & \to & S(G) \\ H & \mapsto & \sigma(H) \end{pmatrix}.$$

Maintenant, pour $g \in G$, soit d_g la multiplication à gauche par g, c'est-à-dire

$$d_g : \begin{pmatrix} G & \to & G \\ h & \mapsto & g \cdot h \end{pmatrix}$$

On remarque que $\gamma \circ (d_g)^{-1}$ cache le même sous-groupe γ. On a effet la série d'équivalences suivante :

$$\begin{array}{lll} & \gamma \circ (d_g)^{-1}(g_1) = \gamma \circ (d_g)^{-1}(g_2) \\ \iff & \exists h \in H_\gamma \quad (d_g)^{-1}(g_2) = (d_g)^{-1}(g_1) \cdot h \\ \iff & \exists h \in H_\gamma \quad g^{-1} \cdot g_2 = g^{-1} \cdot g_2 \cdot h \\ \iff & \exists h \in H_\gamma \quad g_2 = g_1 \cdot h \end{array}$$

$(d_g, \mathrm{id}, \mathrm{id})$ est donc aussi un automorphisme de $\mathbf{HSP}_G$.

Il paraît cependant difficile de donner une caractérisation plus précise de $\mathrm{Aut}(\mathbf{HSP}_G)$. Il peut exister dans le cas général de nombreux autres automorphismes que ceux engendrés par les trois classes d'automorphismes que nous avons déjà évoquées. Par exemple, si p est un nombre premier, pour toute permutation σ de $\mathbb{Z}_p$ telle que $\sigma(0) = 0$, $(\sigma, 0, 0)$ est un automorphisme de $\mathrm{Aut}\left(\mathbf{HSP}_{\mathbb{Z}_p}\right)$, car $\mathbb{Z}_p$ n'a que deux sous-groupes, le sous-groupe trivial et $\mathbb{Z}_p$ lui-même.

4.1.2 Symétrisation

Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire, et $\mathscr{A}$ un algorithme probabiliste pour $\mathscr{P}$. Selon ce que nous avons énoncé dans le fait 1.19, on peut considérer que $\mathscr{A}$ consiste à tirer un algorithme déterministe au hasard. Disons donc que $\mathscr{A}$, de complexité en requêtes T, est défini par une distribution de probabilité sur Ω et des applications $h_i : \Omega \times J^i \to J$, pour $i \in [T]$, et $O : \Omega \times J^T \to R$, de sorte que $\mathscr{A}$ puisse être décrit de cette manière :

- Choisir aléatoirement $\omega \in \Omega$.
- Calculer $i_1 = h_0(\omega)$, quérir i_1 et poser $j_1 = x(i_1)$.
- Calculer $i_2 = h_1(\omega, j_1)$, quérir i_2 et poser $j_2 = x(i_2)$.
- ...
- Calculer $i_T = h_{T-1}(\omega, j_1, \ldots, j_{T-1})$, quérir i_T et poser $j_T = x(i_T)$.
- Renvoyer $O(\omega, j_1, \ldots, j_T)$.

Nous allons définir son symétrisé $\tilde{\mathscr{A}}$. L'idée est que l'on commence par choisir un élément aléatoire (σ, τ, π) de $\mathrm{Aut}(\mathscr{P})$, puis qu'on leurre $\mathscr{A}$ en lui faisant croire que la boîte noire contient la fonction x_σ^τ au lieu de x, ce que l'on obtient en interceptant les communications entre $\mathscr{A}$ et la boîte noire, ainsi qu'il est montré dans la figure 4.1. En compensation, il faut bien sûr également intercepter la réponse de l'algorithme, r, pour la remplacer par $\pi(r)$.

Comme il n'y a clairement aucun besoin pour un algorithme d'effectuer la même requête deux fois[1], on supposera que les requêtes sont distinctes.

[1] Il faut comprendre par là que si un algorithme fait plusieurs fois la même requête lors d'une exécution, alors il existe un autre algorithme de même complexité en requêtes

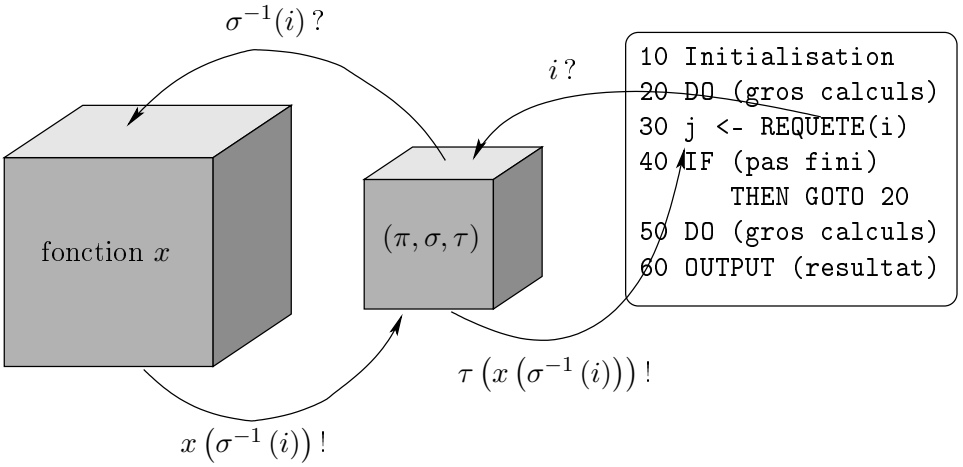

FIG. 4.1 – L'algorithme, la boîte noire et l'intercepteur

Formellement, cela signifie que l'on suppose que $h_k(\omega, j_1, \ldots, j_k)$ est toujours distinct de $h_0(\omega)$, $h_1(\omega, j_1)$,... et $h_{k-1}(\omega, j_1, \ldots, j_{k-1})$.

L'algorithme symétrisé $\tilde{\mathscr{A}}$ se déroule ainsi :

- Choisir aléatoirement $\omega \in \Omega$ et $(\sigma, \tau, \pi) \in \mathrm{Aut}(\mathscr{P})$.
- Calculer $i_1 = h_0(\omega)$, quérir $\sigma^{-1}(i_1)$ et poser $j_1 = \tau x \sigma^{-1}(i_1)$.
- Calculer $i_2 = h_1(\omega, j_1)$, quérir $\sigma^{-1}(i_2)$ et poser $j_2 = \tau x \sigma^{-1}(i_2)$.
- ...
- Calculer $i_T = h_{T-1}(\omega, j_1, \ldots, j_{T-1})$, quérir $\sigma^{-1}(i_T)$ et poser $j_T = \tau x \sigma^{-1}(i_T)$.
- Renvoyer $\pi^{-1} O(\omega, j_1, \ldots, j_T)$.

En réalité, il n'y a pas que les algorithmes probabilistes que l'on peut symétriser, la chose est faisable aussi avec les algorithmes quantiques, mais pour cela il faut reprendre la définition sous forme de circuit. La symétrisation consiste alors à ajouter un registre représentant un sous-espace de dimension $|\mathrm{Aut}(\mathscr{P})|$, où chaque (σ, τ, π) correspond à un vecteur de base $|\sigma, \tau, \pi)$ ou $|\sigma, \tau, \pi\rangle$ selon le cas. Ce registre est initialisé à une superposition uniforme de ses vecteurs de base, ce qui signifie dans le cas probabiliste $\frac{1}{|\mathrm{Aut}(\mathscr{P})|} \sum_{(\sigma,\tau,\pi)\in\mathrm{Aut}(\mathscr{P})} |\sigma, \tau, \pi)$, et $\frac{1}{\sqrt{|\mathrm{Aut}(\mathscr{P})|}} \sum_{(\sigma,\tau,\pi)\in\mathrm{Aut}(\mathscr{P})} |\sigma, \tau, \pi\rangle$ dans le cas

ne faisant jamais deux fois la même requête lors d'une même exécution et répondant la même chose : il suffit pour cela de quérir n'importe quel autre élément de I lorsqu'une collision doit se produire, et d'utiliser le résultat d'une requête précédente pour renseigner l'algorithme sur ce que la réponse de la boîte noire aurait été. Bien entendu, cette manipulation n'est réalisable que si $T \leq |I|$, mais de toute manière le cas $T \geq |I|$ n'a évidemment absolument aucun intérêt.

quantique. Nous utiliserons maintenant, temporairement, la notation $|\cdot\rangle$ dans les deux cas, pour éviter d'avoir à écrire les choses deux fois, puisque le principe de la symétrisation est le même dans les deux systèmes.

Chaque requête est effectuée par le truchement de l'endomorphisme O_x défini par $O_x |i\rangle |j\rangle = |i\rangle |j \oplus x(i)\rangle$. Pour effectuer la symétrisation, on ajoute avant O_x l'endomorphisme défini par $|i\rangle |\sigma, \tau, \pi\rangle \mapsto \left|\sigma^{-1}(i)\right\rangle |\sigma, \tau, \pi\rangle$, et après, l'endomorphisme défini par $|i\rangle |j\rangle |\sigma, \tau, \pi\rangle \mapsto |\sigma(i)\rangle |\tau(j)\rangle |\sigma, \tau, \pi\rangle$ — il faut en effet appliquer σ à i après la requête afin de « masquer » la symétrisation à $\mathscr{A}$. Enfin, on ajoute une dernière porte, à la toute fin de l'algorithme, définie par $|r\rangle |\sigma, \tau, \pi\rangle \mapsto \left|\pi^{-1}(r)\right\rangle |\sigma, \tau, \pi\rangle$, où $|r\rangle$ est le vecteur du registre de réponse correpondant à la réponse $r \in R$. Voici donc la définition du symétrisé $\tilde{\mathscr{A}}$ dans le cas général.

La conséquence la plus immédiate de la symétrisation est la suivante :

Fait 4.4
Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire et $\mathscr{A}$ un algorithme pour $\mathscr{P}$. Pour tous $x \in J^I$ et $r \in R$, on a, en reprenant les notations de la définition 1.17,

$$\mathbb{P}^r_{\tilde{\mathscr{A}}}(x) = \frac{1}{|\mathrm{Aut}(\mathscr{P})|} \sum_{(\sigma,\tau,\pi)\in \mathrm{Aut}(\mathscr{P})} \mathbb{P}^{\pi(r)}_{\mathscr{A}}\left(x^{\tau}_{\sigma}\right).$$

Preuve : Intuitivement, la chose est claire, et si elle est fausse c'est que la définition doit être changée, car le but de la symétrisation est précisément d'obtenir ce genre de propriétés. Par souci de complétude, voici une ébauche de preuve un peu rigoureuse.

Notons respectivement $|\psi^x_t\rangle$ et $\left|\tilde{\psi}^x_t\right\rangle$ l'état du système sur lequel agit l'algorithme $\mathscr{A}$ (respectivement $\tilde{\mathscr{A}}$) à l'instant t, lorsque la fonction de la boîte noire est x. La notion de temps est introduite uniquement pour avoir une notion de coordination entre $\mathscr{A}$ et son symétrisé : les transformations unitaires se font toutes instantanément, mais le temps correspond entre $\mathscr{A}$ et $\tilde{\mathscr{A}}$. Ainsi, si au temps t, l'algorithme $\mathscr{A}$ a effectué 3 requêtes, il en est de même pour $\tilde{\mathscr{A}}$, qui a effectué ces 3 requêtes symétrisées. La symétrisation finale impliquant π, dans $\tilde{\mathscr{A}}$, est effectuée à la toute fin, en un temps que nous noterons t_f, et que $\mathscr{A}$ n'atteint pas.

On peut montrer par récurrence sur le nombre de requêtes que l'on a $\left|\tilde{\psi}^x_t\right\rangle = K \sum_{(\sigma,\tau,\pi)\in \mathrm{Aut}(\mathscr{P})} \left|\psi^{x^{\tau}_{\sigma}}_t\right\rangle |\sigma, \tau, \pi\rangle$ pour $t < t_f$, où K est un facteur normalisant, valant $\frac{1}{|\mathrm{Aut}(\mathscr{P})|}$ dans le cas probabiliste, $\frac{1}{\sqrt{|\mathrm{Aut}(\mathscr{P})|}}$ dans le cas quantique. C'est en effet bien le cas après initialisation, pour la raison que l'on a alors $|\psi^x_t\rangle = |\psi^y_t\rangle$, pour tous $x, y \in J^I$.

Supposons maintenant que l'égalité soit vraie au temps t, juste avant une requête, et soit t' un temps situé juste après la requête. On peut écrire $|\psi_t^x\rangle = \sum_{i\in I, j\in J} \left|\phi_{t,i,j}^x\right\rangle |i\rangle\,|j\rangle$. Alors on a, après la requête, $\left|\psi_{t'}^x\right\rangle = \sum_{i\in I, j\in j} \left|\phi_{t,i,j}^x\right\rangle |i\rangle\,|j\oplus x(i)\rangle$. D'autre part, on a

$$\left|\tilde{\psi}_t^x\right\rangle = K \sum_{\substack{(\sigma,\tau,\pi)\,\in\,\mathrm{Aut}\,(\mathscr{P}) \\ i\,\in\,I,\,j\,\in\,j}} \left|\phi_{t,i,j}^{x_\sigma^\tau}\right\rangle |i\rangle\,|j\rangle\,|\sigma,\tau,\pi\rangle\,.$$

Par définition de $\tilde{\mathscr{A}}$, on a donc, après requête,

$$\left|\tilde{\psi}_{t'}^x\right\rangle = K \sum_{\substack{(\sigma,\tau,\pi)\,\in\,\mathrm{Aut}\,(\mathscr{P}) \\ i\,\in\,I,\,j\,\in\,j}} \left|\phi_{t,i,j}^{x_\sigma^\tau}\right\rangle |i\rangle\,\left|j\oplus\tau x\sigma^{-1}\,(i)\right\rangle |\sigma,\tau,\pi\rangle\,.$$

Comme $x_\sigma^\tau = \tau x\sigma^{-1}$, on peut aussi écrire cet état ainsi :

$$K \sum_{(\sigma,\tau,\pi)\in\mathrm{Aut}(\mathscr{P})} \left|\psi_{t'}^{x_\sigma^\tau}\right\rangle |\sigma,\tau,\pi\rangle\,.$$

C'est précisément ce qu'il fallait prouver pour conduire la récurrence à l'étape suivante.

Il reste à vérifier que le comportement de l'algorithme à la fin est correct. Pour t proche de t_f mais inférieur, on a donc $\left|\tilde{\psi}_t^x\right\rangle = K \sum_{(\sigma,\tau,\pi)\in\mathrm{Aut}(\mathscr{P})} \left|\psi_t^{x_\sigma^\tau}\right\rangle |\sigma,\tau,\pi\rangle$. On peut, similairement à ce que nous venons de faire pour les requêtes, écrire cette fois $|\psi_t\rangle = \sum_{r\in R} |\phi_{t,r}\rangle\,|r\rangle$; on en déduit alors

$$\left|\tilde{\psi}_{t_f}^x\right\rangle = K \sum_{\substack{(\sigma,\tau,\pi)\,\in\,\mathrm{Aut}\,(\mathscr{P}) \\ r\,\in\,R}} \left|\phi_{t_f,\pi(r)}^{x_\sigma^\tau}\right\rangle |r\rangle\,|\sigma,\tau,\pi\rangle\,.$$

D'où le résultat escompté. □

Exemple 4.5

Dans la section 3.3.2, nous définissions une distribution de probabilité comme une moyenne de probabilités de réussite d'un certain algorithme en affirmant qu'il s'agissait d'une symétrisation. On peut effectivement le voir comme le résultat de la symétrisation d'un algorithme. En effet, partant d'un algorithme $\mathscr{A}$ pour $\mathbf{wHSP}_{(\mathbb{Z}_p)^n}$, on peut construire son symétrisé $\tilde{\mathscr{A}}$. Nous avons vu dans l'exemple 4.3 qu'à tout automorphisme σ de $(\mathbb{Z}_p)^n$ correspond un

automorphisme $(\sigma, \mathrm{id}, \tilde{\sigma})$ de $\mathbf{wHSP}_{(\mathbb{Z}_p)^n}$. Or, si H et H' sont deux sous-groupes de $(\mathbb{Z}_p)^n$ de même cardinal, il existe un automorphisme σ de $(\mathbb{Z}_p)^n$ tel que $\sigma(H) = \sigma(H')$ — il ne s'agit que d'un changement de base. Comme, de plus, pour toute bijection τ de $[p^n]$, $(\mathrm{id}, \tau, \mathrm{id})$ est aussi un automorphisme de $\mathbf{wHSP}(\mathbb{Z}_p)^n$, on en déduit que $\mathbb{P}_{\mathscr{A}}(\gamma)$ ne dépend que de l'ordre du sous-groupe caché par γ, qui est ce que nous voulions dans la section 3.3.2 ; comme nous n'avions pas spécifiquement besoin de $\mathscr{A}$, il nous a cependant suffi de prendre directement la moyenne sans nous encombrer d'automorphismes de problèmes élémentaires.

4.1.3 Convexité

Les symétries permettent de simplifier un certain nombre de techniques. Nous allons voir l'exemple des techniques de bornes inférieures vues dans la section 3.1.

Commençons par cette remarque d'ordre général.

Fait 4.6
Soit C un convexe compact de $\mathbb{R}^n$ et $g : C \to \mathbb{R}$ une application convexe. Soit G un groupe fini[2] d'automorphismes linéaires de $\mathbb{R}^n$ tel que C et g soient invariants sous l'action de G, c'est-à-dire

$$\forall x \in C \; \forall \mu \in G \quad (\mu(x) \in C \wedge g(\mu(x)) = g(x)) .$$

Alors il existe $x \in C$, fixé par tous les éléments de G, tel que $g(x) = \min_C g$.

Preuve : Soit y tel que $g(y) = \min_C g$ — un tel y existe car C est compact.

Alors $x = \frac{1}{|G|} \sum_{\mu \in G} \mu(y)$ convient. En effet, par définition x est fixé par tous les éléments de G et par convexité de g, $g(x) \leq g(y)$. □

Reprenons les notations du théorème 3.1. On a $\frac{1}{LM^r(\mathscr{P})} = \max_p \mathscr{F}^r(p)$, où

$$\mathscr{F}^r(p) = \min_{\substack{x, y \in \mathscr{S} \\ f(x) \neq f(y)}} \sum_{i / x(i) \neq y(i)} \min\left(p_x(i), p_y(i)\right) .$$

On peut considérer p comme un vecteur de $\mathbb{R}^{\mathscr{S} \times I}$. Alors le domaine de définition de $\mathscr{F}^r$, qui est l'ensemble des p tels que chaque p_x est une distribution de probabilité sur I, est convexe. Et $\mathscr{F}^r$ étant définie par minimums et sommes, est concave. D'après le fait 4.6, elle atteint donc son maximum en un point de symétrie. Certes, mais quel groupe faisons nous agir ? Eh bien, très

[2] Il n'est pas nécessaire de supposer G fini. On peut se contenter de le supposer compact, la démonstration est la même en remplaçant la distribution de probabilité uniforme sur G par la mesure de Haar.

naturellement, $\mathrm{Aut}(\mathscr{P})$; en effet, ce groupe agit sur p par $(\sigma,\tau,\pi)\cdot p = p_\sigma^\tau$, où par définition $(p_\sigma^\tau)_x(i) = p_{x_\sigma^\tau}(\sigma(i))$. Il nous faut, pour justifier cela, vérifier que $\mathscr{F}^r((\sigma,\tau,\pi)\cdot p) = \mathscr{F}^r(p)$ pour tout $(\sigma,\tau,\pi) \in \mathrm{Aut}(\mathscr{P})$. Par définition, on a

$$\mathscr{F}^r((\sigma,\tau,\pi)\cdot p) = \min_{\substack{x,y\in\mathscr{S}\\ f(x)\neq f(y)}} \sum_{i/x(i)\neq y(i)} \min\left(p_{x_\sigma^\tau}(\sigma(i)), p_{y_\sigma^\tau}(\sigma(i))\right).$$

Par les changements de variable $x' = x_\sigma^\tau$ et $y' = y_\sigma^\tau$, on obtient cette expression :

$$\min_{\substack{x,y\in(\sigma,\tau,\pi)\cdot\mathscr{S}\\ f\left(x_{\sigma^{-1}}^{\tau^{-1}}\right)\neq f\left(y_{\sigma^{-1}}^{\tau^{-1}}\right)}} \sum_{i/x_{\sigma^{-1}}^{\tau^{-1}}(i)\neq y_{\sigma^{-1}}^{\tau^{-1}}(i)} \min\left(p_x(\sigma(i)), p_y(\sigma(i))\right).$$

Comme $f\left(x_{\sigma^{-1}}^{\tau^{-1}}\right) \neq f\left(y_{\sigma^{-1}}^{\tau^{-1}}\right)$ est équivalent à $f(x)\neq f(y)$, cette expression vaut

$$\min_{\substack{x,y\in(\sigma,\tau,\pi)\cdot\mathscr{S}\\ f(x)\neq f(y)}} \sum_{i/x_{\sigma^{-1}}^{\tau^{-1}}(i)\neq y_{\sigma^{-1}}^{\tau^{-1}}(i)} \min\left(p_x(\sigma(i)), p_y(\sigma(i))\right).$$

Effectuons maintenant le changement de variable $i' = \sigma(i)$. On obtient alors

$$\mathscr{F}^r((\sigma,\tau,\pi)\cdot p) = \min_{\substack{x,y\in(\sigma,\tau,\pi)\cdot\mathscr{S}\\ f(x)\neq f(y)}} \sum_{i/\tau^{-1}x(i)\neq\tau^{-1}y(i)} \min\left(p_x(i), p_y(i)\right).$$

Comme $\tau^{-1}x(i) \neq \tau^{-1}y(i)$ est équivalent à $x(i)\neq y(i)$ et, on a donc bien $\mathscr{F}^r((\sigma,\tau,\pi)\cdot p) = \mathscr{F}^r(p)$. On peut par conséquent se contenter de chercher le maximum de $\mathscr{F}^r$ sur les p tels que pour tous $x\in\mathscr{S}$, $i\in I$ et $(\sigma,\tau,\pi)\in\mathrm{Aut}(\mathscr{P})$, $p_{x_\sigma^\tau}(\sigma(i)) = p_x(i)$.

Il en va de même dans le cas quantique. On a $\frac{1}{LM^q(\mathscr{P})} = \max_p \mathscr{F}^q(p)$, où

$$\mathscr{F}^q(p) = \min_{\substack{x,y\in\mathscr{S}\\ f(x)\neq f(y)}} \sum_{i/x(i)\neq y(i)} \sqrt{p_x(i)p_y(i)}.$$

Il faut juste constater que $(x,y)\mapsto\sqrt{xy}$ est concave sur $[0;1]^2$, ce que l'on peut sans doute démontrer de façon astucieuse, mais qui se vérifie tout aussi bien en étudiant sa hessienne.

Exemple 4.7
Prenons l'exemple de $\mathbf{RTNT}_N$, la recherche dans un tableau non trié de taille N, introduite dans la définition 4.2. Notons $\tilde{0}$ la fonction constante

égale à 0 sur $[N]$ et, pour $i \in [N]$, x^i la fonction de $[N]$ telle que $x^i(j) = \delta_{ij}$. Supposons $N > 1$. Rappelons que l'on a dans ce cas $\mathrm{Aut}\,(\mathbf{RTNT}_N) = \mathfrak{S}_{[N]} \times \{\mathrm{id}\} \times \{\mathrm{id}\}$; pour alléger les notations nous nous contenterons de noter σ l'automorphisme $(\sigma, \mathrm{id}, \mathrm{id})$. Tâchons maintenant de calculer $LM^r\,(\mathbf{RTNT}_N)$ et $LM^q\,(\mathbf{RTNT}_N)$. Nous venons d'affirmer qu'il nous suffit de considérer les p symétriques. Quels sont-ils ?

Pour commencer, pour tout $\sigma \in \mathfrak{S}_{[N]}$, on a $\tilde{0}_\sigma = \tilde{0}$, donc pour tout $i, j \in [N]$, $p_{\tilde{0}}\,(i) = p_{\tilde{0}}\,(j)$, d'où l'on déduit que $p_{\tilde{0}}$ doit être la fonction constante égale à $\frac{1}{N}$, puisqu'il s'agit d'une distribution de probabilité sur $[N]$. Ensuite, il faut remarquer que $x^i_\sigma = x^{\sigma(i)}$. On a donc, pour tous $i, j \in [N]$ et $\sigma \in \mathfrak{S}_{[N]}$, $p_{x^i}\,(j) = p_{x^{\sigma(i)}}\,(\sigma(j))$. Or $\mathfrak{S}_{[N]}$ agit 2-transitivement sur $[N]$, c'est-à-dire que pour tous couples (i, j) et (i', j') d'éléments de $[N]$ vérifiant $i \neq j$ et $i' \neq j'$, il existe $\sigma \in \mathfrak{S}_{[N]}$ tel que $\sigma(i) = i'$ et $\sigma(j) = j'$. On a donc, $p_{x^i}\,(j) = p_{x^{i'}}\,(j')$, pour tous i,j,i' et j' dans $[N]$ vérifiant $i \neq j$ et $i' \neq j'$; d'autre part on a aussi, par simple transitivité, $p_{x^i}\,(i) = p_{x^{i'}}\,(i')$ pour tous i, i' dans $[N]$; soit α cette valeur. Du moment que p est symétrique, il est complètement déterminé par α ; nous le noterons alors p^α. Nous avons en effet déjà vu que $p^\alpha_{\tilde{0}}$ est la fonction constante égale à $\frac{1}{N}$. De plus, par définition, $p^\alpha_{x^i}\,(i) = \alpha$. Comme $p^\alpha_{x^i}$ est une distribution de probabilité sur $[N]$ et est constante sur $[N] \setminus \{i\}$, on a $p^\alpha_{x^i}\,(j) = \frac{1-\alpha}{N-1}$ pour $j \neq i$. Notons que la seule contrainte sur la valeur d'α est d'être dans l'intervalle $[0; 1]$. Maintenant, $\mathscr{F}^r$ et $\mathscr{F}^q$ sont fort simples à calculer :

$$\mathscr{F}^r\,(p^\alpha) = \min\left(\alpha, \frac{1}{N}\right),$$

$$\mathscr{F}^q\,(p^\alpha) = \sqrt{\frac{\alpha}{N}}.$$

Dans les deux cas, on atteint le maximum dans le cas $\alpha = 1$, ce qui donne $LM^p\,(\mathbf{RTNT}_N) = N$ et $LM^q\,(\mathbf{RTNT}_N) = \sqrt{N}$.

4.1.4 Problèmes

Soit G un sous-groupe de $\mathrm{Aut}\,(\mathscr{P})$. Nous allons nous intéresser à l'action de G sur $\mathscr{S}$, action qui est simplement définie comme précédemment en attribuant un rôle trivial à la permutation des éléments de R :

$$(\sigma, \tau, \pi) \cdot x = x^\tau_\sigma.$$

Ce qui va particulièrement nous intéresser sont les orbites de $\mathscr{S}$ sous l'action de G. Nous noterons $\mathscr{O}_G\,(\mathscr{P})$ l'ensemble de ces orbites. Nous allons définir dans peu de temps la notion d'action transitive de G sur $\mathscr{P}$, mais avant cela il nous paraît avisé d'expliquer brièvement pourquoi nous distinguons cette notion de celle d'action transitive de G sur $\mathscr{S}$, qui se produit quand $\mathscr{O}_G\,(\mathscr{P}) = \{\mathscr{S}\}$.

Bien sûr Il faut voir que les problèmes élémentaires $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ tels que l'action de $\mathrm{Aut}(\mathscr{P})$ sur $\mathscr{S}$ est transitive ne sont pas nécessairement très intéressants. Que l'on songe aux propriétés qu'un tel problème vérifierait. Pour chaque x et chaque y de $\mathscr{S}$, il devrait exister σ et τ tels que $y = x^{\tau}_{\sigma}$. À renommage des entrées et des indices près, toutes les fonctions de $\mathscr{S}$ sont identiques ! Autrement dit, il existe des entiers strictement positifs $\lambda_1, \ldots, \lambda_k$ vérifiant $\sum_{i=1}^{k} \lambda_i = |I|$ tels que pour tout $x \in \mathscr{S}$, il existe $j_1, \ldots, j_k \in J$ deux-à-deux distincts tels que pour tout $i \in \{1, \ldots, k\}$, $\left|x^{-1}(j_i)\right| = \lambda_i$. Par exemple, dans le cas de la recherche dans un tableau non trié, si l'on se fixe toujours pour but de repérer les tableaux remplis de « 0 » à l'exception d'un « 1 » quelque part, on ne peut pas utiliser comme témoin le tableau nul. On peut décider que les témoins seront les complémentaires des tableaux recherchés, soit les tableaux remplis de « 1 » à l'exception d'une case qui contient un « 0 ». On peut bien entenu imaginer des exemples moins triviaux, mais nous ne voyons aucun cas où cette propriété de transitivité « totale » ne réduirait pas l'intérêt des problèmes à la portion congrue, comparé à la propriété de transitivité que nous définissons maintenant.

Définition 4.8
Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire. Un sous-groupe G de $\mathrm{Aut}(\mathscr{P})$ est dit **agir transitivement** *sur $\mathscr{P}$ si pour chaque $r \in R$, $f^{-1}(r)$ est contenu tout entier dans une même classe de transitivité de $\mathscr{S}$ sous l'action de G, c'est-à-dire*

$$\mathscr{O}_{G \cap (\mathfrak{S}_I \times \mathfrak{S}_J \times \{0\})}(\mathscr{P}) = \left\{ f^{-1}(r) / r \in R \right\}.$$

On dit que $\mathscr{P}$ est **transitivement symétrique** *si $\mathrm{Aut}(\mathscr{P})$ agit transitivement sur $\mathscr{P}$.*

Lorsque l'on considèrera tous les automorphismes d'un problème élémentaire, on omettra la mention du groupe ; ainsi $\mathscr{O}(\mathscr{P})$ sera utilisé en lieu et place de $\mathscr{O}_{\mathrm{Aut}(\mathscr{P})}(\mathscr{P})$.

4.2 Non-adaptativité

Nous allons désormais nous intéresser uniquement aux algorithmes probabilistes. Il existe une notion d'algorithme quantique non-adaptatif, mais on ne sait en dire que peu de choses ; on pourra tout de même se reporter à [BD99] ou [NY04] pour plus d'information.

4.2.1 Définition

Informellement, un algorithme est dit non-adaptatif lorsque il ne tient pas compte du résultat des requêtes avant la toute fin, au moment de rendre

son verdict.

Définition 4.9
Soit $\mathscr{A}$ un algorithme probabiliste de complexité en requêtes T pour un problème élémentaire $\mathscr{P} = (I, J, R, \mathscr{S}, f)$. Si, pour tout $A \in I^*$ de taille au plus T, $\mathbb{P}_{\mathscr{A}}(x, A)$ ne dépend pas de $x \in \mathscr{S}$, alors on dit que $\mathscr{A}$ est un algorithme **non-adaptatif**. *On note alors fort logiquement $\mathbb{P}_{\mathscr{A}}(A) = \mathbb{P}_{\mathscr{A}}(x, A)$.*

Cette définition justifie que l'on puisse présenter un algorithme non-adaptatif de complexité en requêtes $T \leq |I|$ pour un problème élémentaire $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ de la sorte :

- Choisir aléatoirement $\omega \in \Omega$
- Quérir les image $j_1, \ldots, j_T$ des éléments de A_ω par x.
- Renvoyer $O(\omega, j_1, \ldots, j_T)$.

Il apparaît clairement que l'on a une propriété équivalente à celle du fait 1.19 : un algorithme probabiliste non-adaptatif n'est rien d'autre qu'une distribution de probabilité sur des algorithmes déterministes non-adaptatifs.

Fait 4.10
Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire. Pour tout algorithme probabiliste non-adaptatif $\mathscr{A}$ de complexité en requêtes T, il existe une distribution de probabilité $p : \Omega \to [0; 1]$, où Ω est un ensemble fini d'algorithmes déterministes non-adaptatifs de complexité en requêtes T, qui est équivalente à $\mathscr{A}$ en ce sens que l'on a, pour tout $x \in \mathscr{S}$,

- *pour tout $A \in I^*$ de taille au plus T, $\mathbb{P}_{\mathscr{A}}(A) = \sum_{\mathscr{D} \in \Omega} p(\mathscr{D})\, \mathbb{P}_{\mathscr{D}}(A)$, et*
- $\mathbb{P}^r_{\mathscr{A}}(x) = \sum_{\mathscr{D} \in \Omega} p(\mathscr{D})\, \mathbb{P}^r_{\mathscr{D}}(x)$.

Analyser la complexité des algorithmes non-adaptatifs est, comme on peut s'y attendre, particulièrement facile ; nous allons voir comment procéder, à l'aide de la programmation linéaire.

4.2.2 Programmation linéaire

Un **problème de programmation linéaire** n'a rien à voir avec ce que nous appelons « problème » de manière générale dans ce mémoire. Il ne s'agit pas d'un problème impliquant une boîte noire. C'est un problème d'*optimisation*, plus précisément on cherche en programmation linéaire à trouver le maximum d'une fonction linéaire sur un domaine lui-même défini par des inéquations affines.

La programmation linéaire est un domaine très étudié. De nombreux problèmes d'optimisation peuvent se ramener à un programme linéaire, et on a découvert des méthodes ingénieuses pour résoudre ces programmes. Nous

ne nous intéresserons cependant pas du tout à la résolution des problèmes de programmation linéaire, mais uniquement au théorème de dualité de cette théorie. Les preuves des affirmations qui vont suivre, ainsi que de plus amples renseignements sur les tenants et aboutissants de la programmation linéaire, peuvent être trouvés par exemple dans [Vaz06], au chapitre 12.

Voici la forme générale que prend un programme linéaire.

- Variables : $x_1, \ldots, x_n$.
- Constantes : $\left(a_i^j\right)_{i=1\ldots n}^{j=1\ldots k}, (b_j)_{j=1\ldots k}, (c_i)_{i=1\ldots n}$.
- Problème : maximiser $\sum_{i=1}^{n} c_i x_i$ sous les contraintes
  1. $\forall j = 1 \ldots k$, $\sum_{i=1}^{n} a_i^j x_i \leq b_j$, et
  2. $\forall i = 1 \ldots n$, $x_i \geq 0$.

Comme annoncé, la fonction à maximiser est linéaire en les variables x_i, et les inégalités 1 et 2 définissent un domaine convexe par inégalités affines, formant ce que l'on appelle un *polyèdre convexe*. Appelons ce programme linéaire le **primal**. Le théorème de dualité de la programmation linéaire énonce que l'on peut associer au primal un **dual** dont la solution est identique. Les constantes du dual sont les mêmes que celles du primal, même si elles ne sont pas utilisées aux mêmes emplacements, mais le nombre de variables change, et le dual est un problème de minimisation, alors que le primal était un problème de maximisation. Voici la forme du dual :

- Variables : $y_1, \ldots, y_k$.
- Constantes : $\left(a_i^j\right)_{i=1\ldots n}^{j=1\ldots k}, (b_j)_{j=1\ldots k}, (c_i)_{i=1\ldots n}$.
- Problème : minimiser $\sum_{j=1}^{k} b_j y_j$ sous les contraintes
  - $\forall i = 1 \ldots n$, $\sum_{j=1}^{k} a_i^j y_j \geq c_i$, et
  - $\forall j = 1 \ldots k$, $y_i \geq 0$.

Ainsi, à part la condition de positivité des variables, le dual a autant d'inégalités définissant le domaine que le primal a de variables, et vice-versa.

Un aspect intéressant, pour nous, des programmes linéaires, est qu'ils consistent à maximiser une fonction linéaire sur un convexe. Ils tombent donc tout à fait sous le coup du fait 4.6.

Avant de passer à la suite, nous nous devons de mentionner que cette dualité de la programmation linéaire, telle que nous l'avons utilisée pour parler d'algorithmes probabilistes en utilisant le fait 4.10, peut être également vue comme une application du polymorphe principe de Yao [Yao77],

qui est de fait une application de la dualité de la programmation linéaire à l'algorithmique probabiliste.

4.2.3 Formule

Pour énoncer la proposition qui suit, nous avons besoin de cette définition.

Définition 4.11
Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire, pour $A \in I^T$, $X \subseteq J^T$ et $r \in R$.

$\mathbb{P}_A^r(X)$ *est la proportion des éléments x de $f^{-1}(r)$ vérifiant $x(A) \in X$.*

Proposition 4.12
Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire transitivement symétrique. La probabilité de succès maximale pour un algorithme probabiliste non-adaptatif $\mathscr{A}$ de complexité en requêtes $T \leq |I|$ vaut

$$\beta = \min_{\substack{p_r \geq 0 \\ \sum_{r\in R} p_r = 1}} \max_{\substack{A \in I^T \\ \bigsqcup_{r\in R} X_r = J^T}} \sum_{r\in R} p_r \cdot \mathbb{P}_A^r(X_r)$$

Cette formule, dans le cas particulier des problèmes de décision et avec une notion plus faible d'automorphisme de problème élémentaire, est l'objet de [KNP06].

Preuve de la proposition 4.12 : Un algorithme déterministe non-adaptatif de complexité en requêtes T est défini par un ensemble $A \subseteq I$ de taille T et une partition de $J^A = \bigsqcup_{r\in R} X_r$. Selon cette description, l'algorithme consiste à effectuer les requêtes i pour $i \in A$, puis à répondre « r » si $x(A) \in X_r$; on peut représenter cet algorithme par le couple $\left(A, (X_r)_{r\in R}\right)$.
Supposons maintenant que, pour $\omega \in \Omega$, nous appliquions l'algorithme $\left(A, (X_r)_{r\in R}\right)$, avec probabilité $q\left(A, (X_r)_{r\in R}\right)$, où q est une distribution de probabilité sur l'ensemble des algorithmes déterministes non-adaptatifs de complexité en requêtes T. La probabilité de réussite de cette procédure, sur l'entrée $x \in \mathscr{S}$, vaut

$$\sum_{\left(A,(X_r)_{r\in R}\right)/x(A)\in X_{f(x)}} q\left(A, (X_r)_{r\in R}\right).$$

D'après le fait 4.10, un algorithme probabiliste non-adaptatif de complexité en requêtes T est une distribution de probabilité sur des algorithmes déterministses non-adaptif de complexité en requêtes T ; par conséquent, β, la probabilité de succès maximale, est la solution de ce programme linéaire :

- Variables : b et les $q\left(A,(X_r)_{r\in R}\right)$ tels que $|A|=T$.
- Problème : maximiser b sous les contraintes
  1. $\sum_{\left(A,(X_r)_{r\in R}\right)} q\left(A,(X_r)_{r\in R}\right)\leq 1$,
  2. $\forall x\in\mathscr{S}, b-\sum_{\left(A,(X_r)_{r\in R}\right)/x(A)\in X_{f(x)}} q\left(A,(X_r)_{r\in R}\right)\leq 0$,

     et
  3. $\forall\left(A,(X_r)_{r\in R}\right), q\left(A,(X_r)_{r\in R}\right)\geq 0$.

Cela fait un nombre considérable, mais bien déterminé, de variables. On pourra remarquer que, bien que q soit censé être une distribution de probabilité sur les $\left(A,(X_r)_{r\in R}\right)$, nous requérons seulement que la somme des $q\left(A,(X_r)_{r\in R}\right)$ soit inférieure ou égale à 1, et non nécessairement égale. Il s'agit simplement d'un allègement technique qui permet de simplifier l'écriture du dual. Cette altération ne modifie pas la solution du programme linéaire. En effet, si pour certaines valeurs les variables b et $q\left(A,(X_r)_{r\in R}\right)$ réalisent les contraintes, on peut toujours augmenter la valeur des $q\left(A,(X_r)_{r\in R}\right)$ pour faire en sorte que leur somme fasse 1, et les contraintes seront toujours respectées. Cela étant dit, voici maintenat le dual de ce programme linéaire :

- Variables : S et les α_x, pour $x\in\mathscr{S}$.
- Problème : minimiser S sous les contraintes
  1. $\sum_{x\in\mathscr{S}}\alpha_x\geq 1$,
  2. $\forall\left(A,(X_r)_{r\in R}\right), S-\sum_{x/x(A)\in X_{f(x)}}\alpha_x\geq 0$,
  3. $S\geq 0$ et
  4. $\forall x\in\mathscr{S}, \alpha_x\geq 0$.

Soit $x\in\mathscr{S}$, $(\sigma,\tau,\mathrm{id})\in\mathrm{Aut}(\mathscr{P})$ et $\left(A,(X_r)_{r\in R}\right)$ un algorithme déterministe non-adaptatif. On a

$$x_\sigma^\tau(A)\in X_{f(x)}\iff x\left(\sigma^{-1}(A)\right)\in\tau^{-1}\left(X_{f(x)}\right).$$

Fixons S et $(\alpha_x)_{x\in\mathscr{S}}$ respectant les contraintes du dual, ainsi qu'un automorphisme $(\sigma,\tau,\mathrm{id})$ de $\mathscr{P}$. (σ,τ) agit naturellement par permutation sur les variables du programme, par $(\sigma,\tau)\cdot\alpha_x=\alpha_{x_\sigma^\tau}$; il agit aussi sur les programmes déterministes non-adaptatifs, par

$$(\sigma,\tau)\cdot\left(A,(X_r)_{r\in R}\right)=\left(\sigma(A),(\tau(X_r))_{r\in R}\right).$$

L'équivalence que nous avons écrite dit simplement que l'algorithme $\mathscr{A} = \left(A, (X_r)_{r\in R}\right)$ ne fait pas d'erreur sur l'entrée x_σ^τ si et seulement si l'algorithme $\left(\sigma^{-1}, \tau^{-1}\right) \cdot \mathscr{A}$ ne fait pas d'erreur sur l'entrée x. Comme la permutation des variables peut être « compensée » par une permutation des algorithmes, les contraintes du programme linéaire dual sont globalement invariantes par l'action de (σ, τ). Comme de plus la fonction à maximiser, S, est inchangée par cette action, on en déduit, d'après le fait 4.6, que l'on peut supposer que les variables sont un point fixe de cette action. En l'occurrence, cela signifie que l'on peut supposer que l'on a, pour tout $x \in \mathscr{S}$ et tout $(\sigma, \tau, 0) \in \mathrm{Aut}\,(\mathscr{P})$, $\alpha_{x_\sigma^\tau} = \alpha_x$. $\mathscr{P}$ étant supposé transitivement symétrique, on peut aussi écrire β comme la solution de ce programme linéaire dual :

- Variables : S et les p_r, pour $r \in R$.
- Problème : minimiser S sous les contraintes
  1. $\sum_{r\in R} p_r \geq 1$,
  2. $\forall \left(A, (X_r)_{r\in R}\right)$, $S - \sum_{r\in R} p_r \cdot \mathbb{P}_A^r\,(X_r) \geq 0$,
  3. $S \geq 0$ et
  4. $\forall r \in R$, $p_r \geq 0$.

On retrouve ainsi précisément la formule exprimée dans la proposition 4.12.

□

4.2.4 Point de vue algorithmique

β s'exprime donc comme le minimum sur un certain convexe du maximum d'un certain nombre de fonctions convexes définies sur un espace de dimension $|R| - 1$. Dans ces conditions, on peut affirmer que, pour un problème donné, il suffit en réalité de considérer $|R|$ fonctions convexes particulières. Précisément, on a le résultat suivant :

Fait 4.13
Soit $n > 0$, C une partie convexe de $\mathbb{R}^n$ et $(f_i)_{i\in I}$ une famille finie de fonctions convexes de C dans R. Alors il existe $J \subseteq I$ de taille au plus $n + 1$ tel que

$$\min_{x\in C} \max_{i\in I} f_i\,(x) = \min_{x\in C} \max_{i\in J} f_i\,(x)\,.$$

Il s'agit là d'une conséquence immédiate d'un théorème démontré par Eduard Helly dans [Hel23], et que voici :

Théorème 4.14 (Helly)
Soit $(C_i)_{i\in I}$ une famille finie de convexes de $\mathbb{R}^n$ telle que $\bigcap_{i\in I} C_i = \emptyset$. Alors il existe $J \subseteq I$ de taille au plus $n+1$ tel que $\bigcap_{i\in J} C_i = \emptyset$.

> *Preuve du fait 4.13 :* Soit $m = \min_{x\in C} \max_{i\in I} f_i(x)$ et, pour $i \in I$, $C_i = \{x \in C / f_i(x) < m\}$. Les C_i sont convexes car les f_i le sont, et leur intersection est vide par définition de m. D'après le théorème de Helly, il existe donc $J \subseteq I$ de taille au plus $n+1$ tel que $\bigcap_{i\in J} C_i = \emptyset$, ce qui signifie $\min_{x\in C} \max_{i\in J} f_i(x) \geq m$; du fait que $J \subseteq I$ on a immédiatement l'inégalité inverse. □

Appliquons le fait 4.13 au calcul du paramètre β de la proposition 4.12. Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire. Au sens strict, la minimisation se fait sur $|R|$ variables, les p_r. Toutefois, ces variables étant liées par la contrainte $\sum_{r\in R} p_r = 1$, un simple changement de variable nous ramène à un problème de minimisation sur un convexe de $\mathbb{R}^{|R|-1}$. Il existe par conséquent, pour $s \in R$, $A^s \in I^T$ et $(X_r^s)_{r\in R}$, avec pour tout $s \in R$, $\bigsqcup_{r\in R} X_r^s = J^T$, tels que le paramètre β de la proposition 4.12 vaille

$$\min_{\substack{p_r \geq 0 \\ \sum_{r\in R} p_r = 1}} \max_{s\in R} \sum_{r\in R} p_r \cdot \mathbb{P}^r_{A^s}(X_r^s).$$

Le dual donne une expression de cette forme :

$$\beta = \max_{A^s,(X_r^s)} \min_{\substack{q_s \geq 0 \\ \sum_{s\in R} q_s = 1}} \sum_{s\in R} q_s \left[\sum_{r\in R} \mathbb{P}^r_{A^s}(X_r^s)\right].$$

Or $\sum_{r\in R} \cdot \mathbb{P}^r_{A^s}(X_r^s)$ est la probabilité de réussite du symétrisé de l'algorithme suivant :

- Effectuer les requêtes A^s.
- Si $x(A^s) \in X_r^s$ alors répondre « $f(x) = r$ ».

On en déduit donc qu'il existe une distribution de probabilité $(q_s)_{s\in R}$ telle que le symétrisé de l'algorithme suivant est optimal parmi les algorithmes non-adaptatifs de complexité en requêtes donnée :

- Choisir $s \in R$ avec probabilité q_s.
- Effectuer les requêtes A^s.
- Si $x(A^s) \in X_r^s$ alors répondre « $f(x) = r$ ».

Nous verrons dans la suite essentiellement des exemples où $|R| = 2$. Dans ce cas, un algorithme non-adaptatif optimal peut donc s'écrire comme le symétrisé d'une superposition probabiliste de deux algorithmes déterministes, pas plus.

4.3 Quand l'adaptativité ne sert à rien

4.3.1 Le Problème

À la lumière des résultats de la section 4.2, on peut naturellement se demander si la restriction à des algorithmes non-adaptatifs est particulièrement contraignante. Plusieurs questions émergent naturellement :

- Les algorithmes déterministes non-adaptatifs sont-ils aussi efficaces que les algorithmes déterministes adaptatifs effectuant autant de requêtes ?
- Les algorithmes probabilistes non-adaptatifs sont-ils exactement aussi efficaces que les algorithmes probabilistes adaptatifs effectuant autant de requêtes ?
- Les algorithmes probabilistes non-adaptatifs sont-ils presque aussi efficaces que les algorithmes probabilistes adaptatifs effectuant autant de requêtes ?

Toutes semblent ardues en général ; on peut citer en exemple une conjecture de Karp, qui semble être citée dans la littérature pour la première fois par Arnold L. Rosenberg dans [Ros73], et reste ouverte à ce jour. Comme nous pensons que cet exemple mérite d'être connu, nous allons développer brièvement ; pour cela il nous faut ces deux définitions.

Définition 4.15
Une propriété de graphe[3] $\mathcal{P}$ *est* **croissante** *si pour tous graphes* $G = (V, E)$ *et* $G' = (V, E')$ *avec* $E \subseteq E'$, *si* $\mathcal{P}(G)$ *alors* $\mathcal{P}(G')$.

Définition 4.16
Un problème $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ *est dit* **évasif** *si sa complexité en requêtes déterministe vaut* $|I|$.

Selon la conjecture de Karp donc, toute propriété de graphe croissante et non triviale[4] génère un problème évasif.[5] Comme il découle aisément de la proposition 4.12 — ou plus directement de trente secondes de réflexion — que la complexité en requêtes déterministe et non-adaptative de toute propriété de graphe non triviale sur les graphes de taille n est de $\frac{n(n-1)}{2}$, la conjecture de Karp équivaut à : « pour décider les propriétés de graphe croissantes et non triviales, les algorithmes non-adaptatifs sont optimaux ». On voit donc que même dans un cas particulier de propriétés de graphe, et

[3] Voir la définition 2.1.
[4] C'est-à-dire ni toujours vraie, ni toujours fausse.
[5] Pour la formalisation des problèmes de graphes, voir la section 2.1.1

en ne considérant que les algorithmes déterministes, la question de l'efficacité des algorithmes non-adaptatifs est difficile.

Remarquons que « croissante » est un mot important dans la conjecture. L'intuition peut laisser penser que toute propriété de graphe non triviale est évasive, mais si c'est le cas ce n'est qu'un mauvais coup de plus à mettre à son actif. En effet, des « contre-exemples » à la conjecture de Karp existent dans le cas de propriétés non-croissantes ; un tel exemple est fourni par la reconnaissances des graphes-scorpions. Un graphe G de taille n est un **graphe-scorpion** s'il contient un sommet b de degré $n-2$ et que le seul sommet qui ne soit pas adjacent à b est de degré 1 et relié à un sommet u lui-même de degré 2. Si cela n'est pas clair, une illustration est fournie avec la figure 4.2.

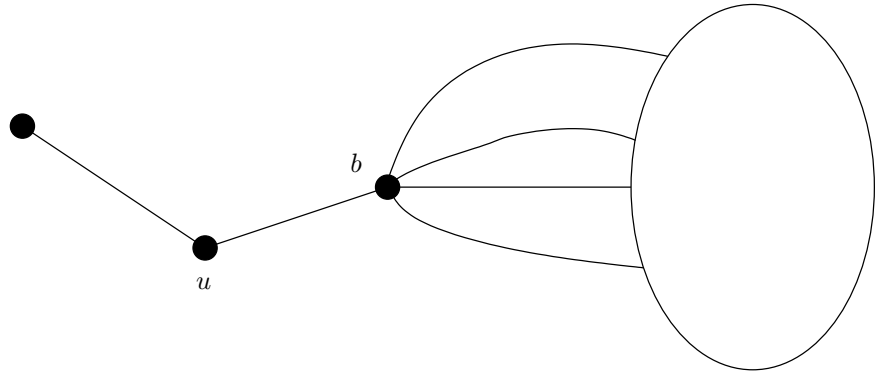

FIG. 4.2 – Un graphe-scorpion tout ce qu'il y a de plus typique

Il existe un algorithme déterministe reconnaissant les graphes-scorpions de taille n, et de complexité en requêtes seulement au plus $6n$. On peut trouver la preuve de ce fait dans le théorème 1.5 du chapitre VIII de [Bol04].

On peut citer également un exemple plus simple : distinguer un graphe-étoile[6] d'un graphe-étoile auquel une arête a été retirée. Il ne s'agit pas d'un problème de graphes à proprement parler, puisqu'il y a la promesse supplémentaire que le graphe est d'une forme ou d'une autre, mais de la même manière les algorithmes déterministes non-adaptatifs doivent effectuer $\frac{n(n-1)}{2}$ requêtes pour résoudre ce problème alors qu'il existe clairement un algorithme déterministe résolvant le problème et ayant une complexité en requêtes en $O(n)$.

Il y a donc bien quelques difficultés à dire quand l'adaptativité est inutile. Nous allons tout de même consacrer la suite de cette section à donner des exemples simples de classes de problème pour lesquels c'est le cas.

[6] Un graphe-étoile est un graphe à $n-1$ arêtes, qui connectent $n-1$ sommets « extérieurs » à un sommet « central ».

4.3.2 Propriétés générales

Nous allons dans cette section relever quelques propriétés générales de $\mathbb{P}_{\mathscr{A}}(x, A)$.

Fait 4.17
Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire, $\mathscr{A}$ un algorithme probabiliste pour $\mathscr{P}$ et $A \in I^*$. Si $x(A) = y(A)$ alors pour tout $i \in I$,

$$\mathbb{P}_{\mathscr{A}}(x, A.(i)) = \mathbb{P}_{\mathscr{A}}(y, A.(i)).$$

Preuve : Supposons dans un premier temps que $\mathscr{A}$ est un algorithme déterministe. Dans ce cas, $\mathbb{P}_{\mathscr{A}}(x, A)$ ne peut prendre que les valeurs 0 et 1, et la conclusion du fait 4.17 découle du fait que les requêtes faites dépendent de manière déterministe des résultats des requêtes précédentes.

Dans le cas général d'un algorithme probabiliste, le résultat découle du cas précédent et du fait qu'un algorithme probabiliste est une superposition probabiliste d'algorithmes déterministes, comme indiqué par le fait 1.19. □

Nous avons déjà associé à un algorithme $\mathscr{A}$ pour un problème élémentaire $\mathscr{P}$ son algorithme symétrisé $\tilde{\mathscr{A}}$: il s'agit de la définition de la section 4.1.2.

Fait 4.18
Pour tout $(\sigma, \tau, \pi) \in \mathrm{Aut}(\mathscr{P})$ et tout $A \in I^*$,

$$\mathbb{P}_{\tilde{\mathscr{A}}}(x, A) = \mathbb{P}_{\tilde{\mathscr{A}}}(x_\sigma^\tau, \sigma(A)).$$

Preuve : Par définition de la symétrisation, on a

$$\mathbb{P}_{\tilde{\mathscr{A}}}(x, A) = \frac{1}{|\mathrm{Aut}(\mathscr{P})|} \sum_{(\sigma,\tau,\pi)\in \mathrm{Aut}(\mathscr{P})} \mathbb{P}_{\mathscr{A}}(x_\tau^\sigma, \sigma(A)).$$

Soit $(\sigma, \tau, \pi) \in \mathrm{Aut}(\mathscr{P})$.

$$\begin{aligned}
\mathbb{P}_{\tilde{\mathscr{A}}}(x_\sigma^\tau, A) &= \tfrac{1}{|\mathrm{Aut}(\mathscr{P})|} \sum_{(\sigma',\tau',\pi')\in \mathrm{Aut}(\mathscr{P})} \mathbb{P}_{\mathscr{A}}\left((x_\sigma^\tau)_{\sigma'}^{\tau'}, \sigma'(A)\right) \\
&= \tfrac{1}{|\mathrm{Aut}(\mathscr{P})|} \sum_{(\sigma',\tau',\pi')\in \mathrm{Aut}(\mathscr{P})} \mathbb{P}_{\mathscr{A}}\left(x_{\sigma'\circ\sigma}^{\tau'\circ\tau}, \sigma'(A)\right) \\
&= \tfrac{1}{|\mathrm{Aut}(\mathscr{P})|} \sum_{(\sigma'',\tau'',\pi'')\in \mathrm{Aut}(\mathscr{P})} \mathbb{P}_{\mathscr{A}}\left(x_{\sigma''}^{\tau''}, \sigma''\sigma^{-1}(A)\right) \\
&\qquad \text{avec } (\sigma'', \tau'', \pi'') = (\sigma', \tau', \pi').(\sigma, \tau, \pi) \\
\mathbb{P}_{\tilde{\mathscr{A}}}(x_\sigma^\tau, A) &= \mathbb{P}_{\tilde{\mathscr{A}}}\left(x, \sigma^{-1}(A)\right)
\end{aligned}$$

□

4.3.3 Problèmes totalement symétriques

Dans certains cas, sans qu'on ait besoin d'aller chercher plus loin, la symétrisation suffit à construire un algorithme non-adaptatif, quel que soit $\mathscr{A}$; il s'agit donc là d'une propriété qui appartient en propre à $\mathscr{P}$.

Fait 4.19
Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire et supposons qu'il existe $H \leq \mathfrak{S}_I$ agissant k-transitivement[7] sur I et tel que $H \times \{0\} \times \{0\} \leq \mathrm{Aut}(\mathscr{P})$. Alors, pour tout $x \in J^I$ et $A \in I^*$ de taille au plus k,

$$\mathbb{P}_{\tilde{\mathscr{A}}}(x, A) = \frac{1}{|I|\,(|I|-1)\cdots(|I|-|A|-1)}.$$

Preuve : Fixons la valeur des bits aléatoires de $\mathscr{A}$ à ω, et agissons donc pour l'instant comme si $\mathscr{A}$ était déterministe.

Procédons par récurrence sur la taille de A. L'affirmation du fait 4.19 est triviale lorsque $|A| = 0$. Ensuite, la première requête effectuée par $\mathscr{A}$ est toujours la même ; appelons-la i_0. La première requête effectuée par $\tilde{\mathscr{A}}$ est donc $\sigma^{-1}(i_0)$, où (σ, τ, π) est choisi aléatoirement et de manière uniforme parmi les éléments de $\mathrm{Aut}(\mathscr{P})$. Ainsi on a, pour tout $x \in J^I$ et tout $A \in I^1$,

$$\mathbb{P}_{\tilde{\mathscr{A}}}(x, A) = \frac{1}{|I|}.$$

Après la première requête, $\mathscr{A}$ ne querra plus jamais i_0 ; on peut donc voir le reste de l'algorithme comme une procédure auxiliaire travaillant sur un problème élémentaire modifié où les fonctions de $\mathscr{S}$ ne serait plus définies que sur $I \setminus \{i_0\}$. Pour le présenter d'une autre manière, cela ne changerait rien — c'est-à-dire qu'on obtiendrait un algorithme équivalent à $\tilde{\mathscr{A}}$ — si, avant la deuxième requête, on remplaçait l'automorphisme (σ, τ, π) choisi par $\tilde{\mathscr{A}}$ par un nouvel automorphisme (σ', τ', π') choisi au hasard et de manière uniforme parmi les automorphismes de $\mathscr{P}$ vérifiant $\sigma'(i_0) = \sigma(i_0)$. Comme $\mathrm{Aut}(\mathscr{P})$ contient H qui agit 2-transitivement sur I, la deuxième requête lancée par $\tilde{\mathscr{A}}$ est aléatoire, et uniformément choisie dans $I \setminus \{i_0\}$. Le raisonnement se poursuit par récurrence et cela prouve donc le résultat dans le cas où $\mathscr{A}$ est déterministe. Comme un algorithme probabiliste peut s'écrire comme une superposition probabiliste d'algorithmes déterministes[8], le résultat reste vrai pour les algorithmes probabilistes. □

[7] On dit qu'un groupe G agit *k-transitivement* sur un ensemble X si, étant donné deux k-uplets $(x_1, \ldots, x_k)$ et $(y_1, \ldots, y_k)$ d'éléments distincts de X, il existe toujours $g \in G$ tel que pour tout $i = 1, \ldots, k$, $g \cdot x_i = y_i$. Pour $k = 1$ on retrouve la notion habituelle de transitivité.

[8] Voir le fait 1.19.

Il est malheureusement « bien connu » — voir par exemple la section 1.12 de [Cam99] — que si un groupe de permutations d'un ensemble à n éléments agit 7-transitivement, alors il agit $(n-2)$-transitivement. Le fait 4.19 n'est par conséquent utile que quand H se trouve être $\mathfrak{S}_I$, le groupe symétrique, ou $\mathfrak{A}_I$, le groupe alterné. Ainsi, si nous définissons ainsi la notion de problème élémentaire totalement symétrique :

Définition 4.20
$\mathscr{P} = (I, J, R, \mathscr{S}, f)$ *est* **totalement symétrique** *si* $\mathfrak{S}_I \times \{0\} \times \{0\} \leq \mathrm{Aut}(\mathscr{P})$.

Il s'agit en réalité de ce que l'on appelle en général simplement un « problème symétrique », mais nous préférons ici utiliser une terminologie un peu plus élaborée pour distinguer cette notion de celle de symétrie transitive. Quoi qu'il en soit, on a le corollaire suivant, que l'on peut aussi déduire immédiatement du lemme 9 de [BYKS01] :

Fait 4.21
Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ *un problème élémentaire totalement symétrique. Alors, pour* $\mathscr{P}$*, les algorithmes non-adaptatifs sont aussi efficaces que les algorithmes généraux.*

Preuve : Soit $\mathscr{A}$ un algorithme probabiliste pour $\mathscr{P}$. D'après le fait 4.19, $\tilde{\mathscr{A}}$ est équivalent à un algorithme non-adaptatif ; or sa probabilité d'erreur est la même que celle de $\mathscr{A}$. □

On pourrait penser au premier abord que les hypothèses du fait 4.19 sont trop restrictives, qu'il est suffisant de supposer que $\mathrm{Aut}(\mathscr{P})$ agit k-transitivement sur I — l'action étant définie par $(\sigma, \tau, \pi) \cdot i = \sigma(i)$. Ce n'est cependant pas possible de supposer seulement cela, car les éventuelles « intrications » entre les σ et les τ peuvent poser problème. Considérons par exemple le problème élémentaire $\mathscr{P} = (I, J, R, \mathscr{S}, f)$, où

- $I = J = [N]$,
- $R = \mathbb{B}$,
- $f(x) = \begin{cases} \top & \text{si } x \text{ est l'identité} \\ \bot & \text{si } x \text{ est une transposition.}^9 \end{cases}$, et
- $\mathscr{S} = \mathrm{dom}(f)$.

On vérifie aisément que $\mathrm{Aut}(\mathscr{P}) = \{(\sigma, \sigma) / \sigma \in \mathfrak{S}_{[N]}\} \times \{0\}$, qui sans conteste agit N-transitivement sur $[N]$. Mais soit $\mathscr{A}$ l'algorithme suivant, dont ne nous précisons que le début, la suite n'étant d'aucune importance et pouvant être complètement arbitraire :

- Quérir 1, 2, 3, etc., jusqu'à trouver i_s tel que $x(i_s) \neq i_s$.
- Quérir $x(i_s)$ si ce n'est pas déjà fait.

[9] Une transposition est une permutation se contentant de permuter deux éléments distincts.

Remarquons que si l'on arrive à la seconde étape c'est que x n'est pas l'identité ; s'il s'agit d'une transposition, alors en réalité la requête $x(i_s)$ n'a pas encore été effectuée puisque nécessairement $x(i_s) > i_s$. Voilà maintenant comment on peut décrire $\tilde{\mathscr{A}}$:

- Quérir des éléments aléatoires, distincts et uniformément répartis dans I jusqu'à trouver i_s tel que $x(i_s) \neq i_s$.
- Une fois que cela s'est produit, quérir $x(i_s)$, à condition que cela n'ait déjà été fait.

On voit bien que la conclusion du fait 4.19 est fausse dans cet exemple ; par exemple, si x est la transposition permutant 0 et 1 et σ la transposition permutant 0 et 2, alors $\mathbb{P}_{\tilde{\mathscr{A}}}(x, \sigma) = 0$.

4.3.4 Problèmes de collision

Certains problèmes, en particulier parmi ceux qui sont étudiés dans le cadre du calcul quantique, se ramènent à déterminer si la fonction dans la boîte noire est injective ou pas. C'est par exemple le cas de « one-to-one versus two-to-one », ainsi que des problèmes de décision associés aux problèmes de sous-groupes cachés. Nous appellerons « problèmes de collision » les problèmes de cette forme, puisqu'ils consistent à déterminer si la fonction dans la boîte noire a une *collision*, une collision pour une fonction $f : X \to Y$ étant une paire $\{x, y\}$ d'éléments distincts de X telle que $f(x) = f(y)$; voici une définition précise de cette classe de problèmes.

Définition 4.22
Un problème de décision élémentaire $\mathscr{P} = (I, J, \mathbb{B}, \mathscr{S}, f)$ est un **problème de collision** *si toute application de $f^{-1}(\top)$ est injective alors qu'aucune de $f^{-1}(\bot)$ ne l'est, ou vice-versa en échangeant $\top$ et $\bot$.*

Comme dans la section 4.3.3, nous allons pour l'essentiel symétriser un algorithme $\mathscr{A}$ pour en faire un algorithme non-adaptatif. Seulement, cette opération ne suffit plus tout à fait ; il faut aussi contrôler le comportement de $\tilde{\mathscr{A}}$ après qu'il a trouvé une collision, en prenant la place de la boîte noire, et en intervenant aussi au moment où $\tilde{\mathscr{A}}$ rend son verdict.

Fait 4.23
Soit $\mathscr{P} = (I, J, \mathbb{B}, \mathscr{S}, f)$ un problème de collision tel que $\{0\} \times \mathfrak{S}_J \times \{0\} \leq \mathrm{Aut}(\mathscr{P})$. Alors, pour $\mathscr{P}$, les algorithmes non-adaptatifs sont aussi efficaces que les algorithmes généraux.

Preuve : On supposera sans perte de généralité qu'une fonction x de $\mathscr{S}$ est injective si et seulement si $f(x) = \top$. Comme $\mathrm{Aut}(\mathscr{P})$ contient $\{0\} \times \mathfrak{S}_J \times \{0\}$, $f^{-1}(\top)$ est en fait l'ensemble des applications injectives de I dans J. Soit $\mathscr{A}$ un algorithme pour $\mathscr{P}$ de complexité en requêtes T, et $\tilde{\mathscr{A}}$ son symétrisé.

Soit $x \in \mathscr{S}$ injective et notons B la suite de réponses reçue par $\tilde{\mathscr{A}}$ quand la fonction boîte noire est x. Comme $\mathrm{Aut}(\mathscr{P})$ contient $\{0\} \times \mathfrak{S}_J \times \{0\}$, et selon un raisonnement similaire à celui de la preuve du fait 4.19, à chaque requête, $\tau x \sigma^{-1}(i)$ est uniformément distribué parmi les éléments de J qui ne sont pas les résultats de requêtes précédentes. B est donc uniformément distribué parmi les suites de taille T d'éléments distincts de J.

Nous définissons maintenant un algorithme $\mathscr{A}'$ en modifiant $\tilde{\mathscr{A}}$ de la façon suivante. Tant qu'aucune collision n'a été trouvée, on applique respectueusement $\tilde{\mathscr{A}}$. En revanche, dès qu'une collision se manifeste, on « trompe » $\tilde{\mathscr{A}}$ en lui faisant croire qu'il n'a pas trouvé de collision, puis à la fin on intercepte sa réponse pour annoncer à sa place que la fonction boîte noire possède une collision. Pour faire croire à $\tilde{\mathscr{A}}$ qu'il a affaire à une fonction injective, on répond tout simplement à ses requêtes par des éléments aléatoires de J qui ne sont pas le résultat de précédentes requêtes, et ce de manière uniformément répartie. Cela nous assure que, jusqu'à la toute fin, $\tilde{\mathscr{A}}$ adopte exactement le même comportement statistique face à toutes les fonctions de $\mathscr{S}$ — et même de J^I. En effet, quelle que soit la fonction boîte noire, les réponses aux requêtes de $\tilde{\mathscr{A}}$ sont totalement et uniformément aléatoires — avec seulement la restriction que les réponses sont distinctes les unes des autres. En particulier, $\mathbb{P}_{\mathscr{A}'}(x, A)$ est indépendant de $x \in J^I$, se qui fait que $\mathscr{A}'$ est équivalent a un algorithme non-adaptatif, ainsi qu'énoncé dans le fait 4.9. De plus, par construction, la probabilité d'erreur de $\mathscr{A}'$ vaut au plus celle de $\tilde{\mathscr{A}}$, qui elle-même est inférieure ou égale à celle de $\mathscr{A}$. □

4.4 Exemples

4.4.1 Recherche dans un tableau non trié

Reprenons $\mathbf{RTNT}_N$, le problème de la recherche dans un tableau non trié de taille N, défini dans l'exemple 4.2. D'après le fait 4.21, son groupe d'automorphismes étant $\mathfrak{S}_{[N]} \times \{0\} \times \{0\}$, pour déterminer la probabilité d'erreur minimale des algorithmes effectuant T requêtes, il suffit d'étudier les algorithmes non-adaptatifs. Comme $\mathbf{RTNT}_N$ est transitivement symétrique, on peut pour cela utiliser la proposition 4.12. Commençons par regarder aux algorithmes probabilistes de complexité en requêtes 1. À toute paire $(A, X) \in I^1 \times \mathfrak{P}\left(J^1\right)$, on associe dans le plan (p, e) la droite $\Delta_{A,X}$ d'équation

$$\Delta_{A,X} : e = p \cdot \mathbb{P}_A^{\perp}(X) + (1-p)\left(1 - \mathbb{P}_A^{\top}(X)\right).$$

En fait, quelle que soit la requête A, on obtient les quatre mêmes droites,

représentées sur la figure 4.3 :

- pour $X = \emptyset$: $e = 1 - p$,
- pour $X = \{0\}$: $e = p\left(1 - \frac{1}{N}\right) + \frac{1}{N}$,
- pour $X = \{1\}$: $e = (1-p)\left(1 - \frac{1}{N}\right)$, et
- pour $X = \{0;1\}$: $e = p$.

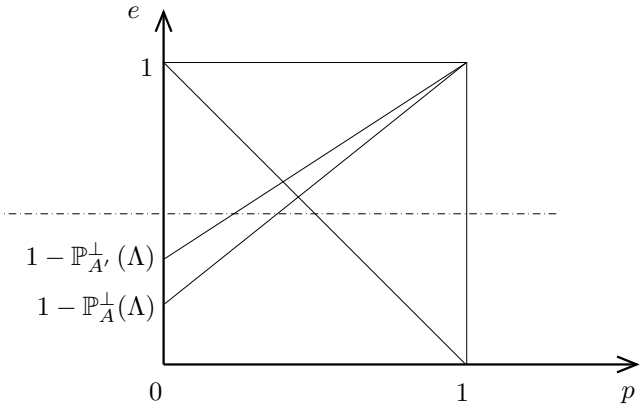

FIG. 4.3 – Une seule requête pour $\mathbf{RTNT}_N$

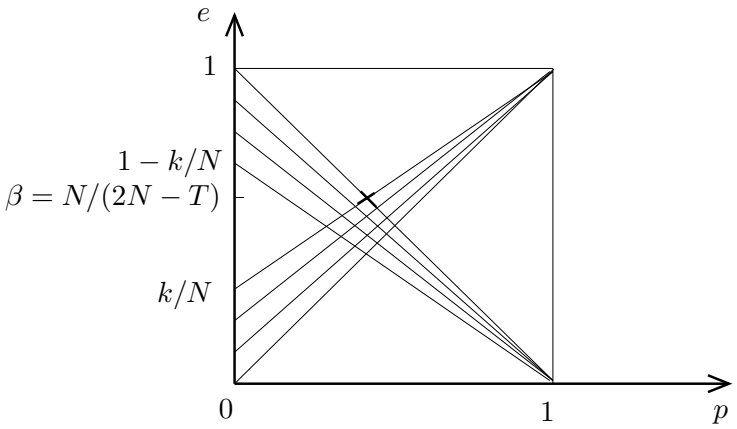

FIG. 4.4 – T requêtes pour $\mathbf{RTNT}_N$

On pourra noter la symétrie de cette figure par rapport à la droite d'équation $e = \frac{1}{2}$. C'est tout à fait naturel et généralisable, puisque les droites $\Delta_{A,X}$ et $\Delta_{A,\mathfrak{P}(J^T)\setminus X}$ sont symétriques l'une de l'autre. On peut calculer le paramètre β de la proposition 4.12 dans ce cas : il vaut $\frac{N}{2N-1}$.

Considérons maintenant le cas général d'un algorithme de complexité en requêtes T, que l'on va tout de même supposer inférieur à N pour d'évidentes raisons. Le calcul est simple, grâce à la propriété suivante : $\mathbb{P}^{\perp}_{A}(X)$ vaut toujours soit 0 soit 1, car la seule suite de réponses possible, à n'importe

quelle suite de requêtes à une fonction de $f^{-1}(\perp)$, est $(0,\ldots,0)$, car il n'y a qu'une seule fonction dans $f^{-1}(\perp)$, la fonction nulle. Plus précisément, $\mathbb{P}_A^{\perp}(X)$ vaut 1 si X contient $(0,\ldots,0)$, 0 sinon. Cela signifie — et pour s'en convaincre il n'est que de jeter un œil à la figure 4.3 — que pour trouver β il nous suffit de trouver $X_{\max}$ minimisant $\mathbb{P}_A^{\top}(X_{\max})$ sous la contrainte $(0,\ldots,0) \in X_{\max}$, puis de regarder l'intersection de la droite correspondante avec celle d'équation $e = 1 - p$. $X_{\max} = \{(0,\ldots,0)\}$ convenant clairement, un simple calcul montre que maintenant

$$\beta = \frac{N}{2N - T}.$$

Ceci nous permet de calculer exactement la complexité en requêtes de $\mathbf{RTNT}_N$ pour une probabilité d'erreur au plus ε :

$$\left\lceil \frac{1-2\varepsilon}{1-\varepsilon} N \right\rceil.$$

4.4.2 Problème de Simon

Nous allons étudier dans cette section le problème de Simon affaibli , soit $\mathbf{wHSP}_{(\mathbb{Z}_2)^n}$ — voir la définition 3.7. Soit A une suite de T requêtes distinctes, et B une suite de T réponses. Bien entendu, si B contient deux fois ou plus la même valeur, alors $\mathbb{P}_A^{\top}(B) = 0$. Il en résulte que, lorsque nous tentons de calculer β, il nous suffit de considérer les X contenant toutes les suites non-injectives. Cela traduit le fait que le problème de Simon est un problème de collision : une fois une collision trouvée, il n'y a plus de question quant à la classification de la fonction boîte noire. Soit donc Λ l'ensemble des suites injectives de taille T d'éléments de J.

Calculons $\mathbb{P}_A^{\perp}(\Lambda)$. Une fonction $\gamma : (\mathbb{Z}_2)^n \to [2^n]$ cachant le sous-groupe $H_\gamma = \{0, s_\gamma\}$ est injective sur A si et seulement si

$$\forall x, y \in A \quad x - y \neq s_\gamma.$$

On peut aussi l'exprimer, en notant, pour deux ensembles X et Y, $X - Y = \{x - y / x \in X, y \in Y\}$, l'exprimer ainsi :

$$s_\gamma \notin A - A.$$

Quand on choisit une fonction γ au hasard et uniformément parmi celles cachant un sous-groupe d'ordre 2, s_γ est uniformément réparti dans $(\mathbb{Z}_2)^n \setminus \{0\}$. Il en résulte ceci :

$$\mathbb{P}_A^{\perp}(\Lambda) = 1 - \frac{|A - A| - 1}{2^n - 1}.$$

Fixons maintenant A et remarquons que ni $\mathbb{P}_A^{\top}(\{B\})$ ni $\mathbb{P}_A^{\perp}(\{B\})$ ne dépend de B du moment que B se contente d'être injectif, c'est-à-dire une

suite d'éléments distincts ; par exemple $\mathbb{P}_A^\top(\{B\})$ vaut $\frac{(2^n-T)!}{2^n!}$, puisque $2^n!$ est le nombre total de fonctions injectives et $(2^n-T)!$ le nombre de fonctions injectives prenant les valeurs B sur A. Lorsque X ne contient que des suites injectives, $\mathbb{P}_A^\top(X)$ et $\mathbb{P}_A^\perp(X)$ sont donc linéaires en la taille de X. En prenant en compte le fait qu'il nous suffit de considérer les ensembles contenant Λ, cette remarque montre que toutes les droites $\Delta_{A,X}$ qui nous intéressent, pour un A fixé, passent par le même p oint, ainsi qu'on peut le voir sur la figure 4.5, où l'on a representé les droites $\Delta_{A,\Xi}$ et $\Delta_{A,\Xi'}$ correspondant respectivement à une proportion ξ et ξ' des fonctions injectives. Le point est en particulier à l'intersection des droites $\Delta_{A,\Lambda}$ et Δ_{A,J^T}, ce qui nous permet de calculer l'ordonnée de ce point d'intersection simplement :

$$\min_{0\leq p\leq 1}\max_X p\mathbb{P}_A^\top(X)+(1-p)\left(1-\mathbb{P}_A^\perp(X)\right)=\frac{1}{1+\mathbb{P}_A^\perp(\Lambda)}=\frac{1}{2-\frac{|A-A|-1}{2^n-1}}.$$

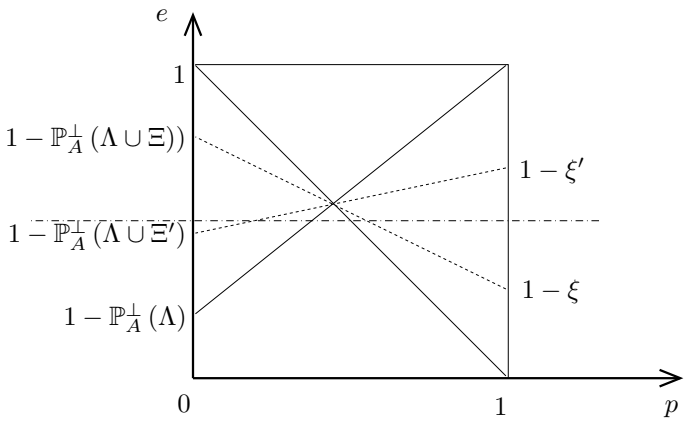

FIG. 4.5 – Les droites $\Delta_{A,X}$ quand X varie

La figure 4.6 devrait également rendre clair le fait que, comme tous ces min-max ont lieu sur la droite d'équation $e=p$, on peut inverser $\min_{0\leq p\leq 1}$ et $\max_A$ dans la définition de β, de sorte que l'on trouve

$$\beta=\max_A\frac{1}{2-\frac{|A-A|-1}{2^n-1}}.$$

Le meilleur algorithme probabiliste de complexité en requêtes T consiste donc à choisir A de taille T maximisant $|A-A|$, à effectuer les requêtes $\sigma^{-1}(A)$ où σ est un automorphisme linéaire aléatoire de $(\mathbb{Z}_2)^N$, puis à effectuer une de ces deux procédures, la première avec probabilité $1-\beta$, la seconde avec probabilité β :

1. Oublier les requêtes, les réponses, et déclarer que γ cache un sous-groupe d'ordre 2.

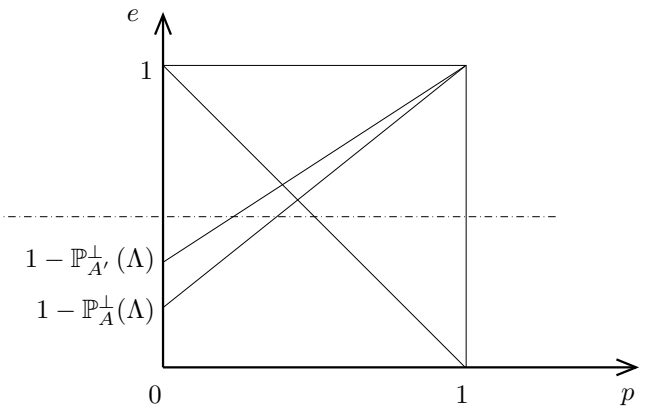

FIG. 4.6 – Les droites $\Delta_{A,\Lambda}$ quand A varie

2. Déclarer que γ cache un sous-groupe d'ordre 2 si et seulement si aucune collision n'a été trouvée.

On voit qu'il y a là un problème combinatoire non trivial : la maximisation de $|A-A|$, à taille de A fixée. Ce problème devient encore plus patent lorque l'on cherche à calculer la complexité en requêtes du problème de décision associé au problème de Simon, pour une probabilité d'erreur au plus ε, puisqu'il s'agit alors de trouver la taille minimale d'un ensemble $A \subseteq (\mathbb{Z}_2)^n$ vérifiant $|A-A| \geq (2^n-1)\left(2-\frac{1}{\varepsilon}\right)+1$. On peut toutefois aisément en déduire qu'il s'agit de $\Theta\left(\sqrt{2^n}\right)$. Premièrement, $|A-A|$ est toujours inférieur à $|A|^2$: ceci montre que la complexité en requêtes est en $\Omega\left(\sqrt{2^n}\right)$. Deuxièmement, on peut écrire $(\mathbb{Z}_2)^n = \mathbb{Z}_2^{\lceil \frac{n}{2} \rceil} \times \mathbb{Z}_2^{\lfloor \frac{n}{2} \rfloor}$ et choisir, en suivant cette décomposition, $A = \left(\mathbb{Z}_2^{\lceil \frac{n}{2} \rceil} \times \{0\}\right) \cup \left(\{0\} \times \mathbb{Z}_2^{\lfloor \frac{n}{2} \rfloor}\right)$. A est de taille $\mathcal{O}\left(\sqrt{2^n}\right)$, et $A+A = (\mathbb{Z}_2)^n$; CQFD.

4.4.3 One-to-one versus two-to-one

Commençons par expliquer le titre de cette section...

Définition 4.24
Une fonction $f : X \to Y$ est dite **two-to-one** *si, pour tout $y \in Y$, le cardinal de $f^{-1}(y)$ vaut 0 ou 2.*

Il s'agit maintenant, étant donné une application de $[2N]$ dans lui-même dont on est assuré qu'elle est soit injective, soit two-to-one, de déterminer dans lequel des deux cas on se trouve. Voici maintenant comme de coutume la définition formelle :

Définition 4.25
$\mathbf{OVT}_N = ([2N], [2N], \mathbb{B}, \mathscr{S}, f)$ *où*

- $f(x) = \begin{cases} \top & \textit{si } x \textit{ est injective} \\ \bot & \textit{si } x \textit{ est two-to-one} \end{cases}$, *et*
- $\mathscr{S} = \mathrm{dom}\,(f)$.

Signalons en passant que ce problème a été raisonnablement étudié dans le cadre quantique ; il s'agit d'un de ces cas, que nous avions mentionnés dans la section 3.2, de problèmes pour lesquels la méthode par adversaire est inefficace pour trouver des bornes inférieures sur la complexité en requêtes. On doit donc s'en remettre tant bien que mal à la méthode polynomiale [AS04, Amb05, Kut03].

$\mathrm{Aut}\,(\mathbf{OVT}_N) = \mathfrak{S}_{[2N]} \times \mathfrak{S}_{[2N]} \times \{0\}$, et $\mathbf{OVT}_N$ est transitivement symétrique. On peut donc calculer sa complexité en requêtes suivant la formule de la proposition 4.12. Il s'agit là encore d'un problème de collision, aussi les remarques préliminaires faites dans la section 4.4.2 restent encore valables, à savoir qu'il nous suffit de considérer les X contenant Λ, l'ensemble des fonctions injectives, car $\mathbb{P}_A^{\top}(\Lambda) = 1$.

Calculons donc $\mathbb{P}_A^{\bot}(\Lambda)$. C'est la probabilité qu'une fonction two-to-one aléatoire x possède une collision sur un ensemble fixé $A \subseteq [2N]$. Il y a $\binom{2N}{T}$ parties de $[2N]$ de taille T. Il faut maintenant compter le nombre de parties A de taille T telles que la restriction de x à A soit injective — x étant two-to-one sur $[2N]$. Pour ce faire, considérons la partition de $[2N]$ en parties à deux éléments sur lesquelles x est constante. Commençons par choisir T de ces parties qui constituent la partition : il y a $\binom{N}{T}$ possibilités. Pour chacune de ces parties il faut encore choisir quel élément est gardé : cela fait 2^T possibilités. Au final, on obtient donc

$$\mathbb{P}_A^{\bot}(\Lambda) = 1 - 2^{|A|}\frac{\binom{N}{|A|}}{\binom{2N}{|A|}}.$$

Toujours dans la même veine que dans la section 4.4.2, pour un A fixé, ni $\mathbb{P}_A^{\top}(\{B\})$ ni $\mathbb{P}_A^{\bot}(\{B\})$ ne dépend de B tant que celui-ci reste injectif, et l'on peut calculer β de la même manière, ce qui donne dans ce cas-ci

$$\beta = \max_{|A|=T} \frac{1}{2 - \mathbb{P}_A^X(\Lambda)} = \frac{1}{1 + 2^T \frac{\binom{N}{T}}{\binom{2N}{T}}}.$$

De cette formule on peut inférer que la complexité en requêtes probabiliste de « one-to-one versus two-to-one », pour une probabilité d'erreur au plus ε, où $\varepsilon \in \left]0; \frac{1}{2}\right]$ est fixé, est équivalente à $2\sqrt{N \ln\left(\frac{1}{\varepsilon} - 1\right)}$. Voici comment :

Soit $F(T,N) = 2^T \frac{\binom{N}{T}}{\binom{2N}{T}}$. On a vu dans la section 4.4.3 que le meilleur algorithme probabiliste de complexité en requêtes T a une probabilité d'erreur de $1 - \frac{1}{1+F(T,N)}$. Si nous voulons que celle-ci soit inférieure à ε, on doit donc avoir $F(T,N) \leq \frac{1}{1-\varepsilon} - 1$.

Il n'existe pas, *a priori*, d'expression plus simple de la complexité en requêtes probabiliste. Toutefois, on peut en donner un équivalent relativement simple pour les grandes valeurs de N. Nous allons pour cela calculer un équivalent de $F\left(c\sqrt{N},N\right)$, où c est une constante.

$$\begin{aligned}
F\left(c\sqrt{N},N\right) &= 2^{c\sqrt{N}}\frac{\binom{N}{c\sqrt{N}}}{\binom{2N}{c\sqrt{N}}}\\
&= 2^{c\sqrt{N}}\frac{N!\left(2N-c\sqrt{N}\right)!}{\left(N-c\sqrt{N}\right)!(2N)!}\\
&\sim 2^{c\sqrt{N}}\frac{\sqrt{2\pi N}\left(\frac{N}{e}\right)^N\sqrt{2\pi\left(2N-c\sqrt{N}\right)}\left(\frac{2N-c\sqrt{N}}{e}\right)^{2N-c\sqrt{N}}}{\sqrt{2\pi\left(N-c\sqrt{N}\right)}\left(\frac{N-c\sqrt{N}}{e}\right)^{N-c\sqrt{N}}\sqrt{4\pi N}\left(\frac{2N}{e}\right)^{2N}}\\
&\sim \frac{1}{2^{2N-c\sqrt{N}}N^N}\frac{\left(2N-c\sqrt{N}\right)^{2N-c\sqrt{N}}}{\left(N-c\sqrt{N}\right)^{N-c\sqrt{N}}}\\
&\sim \frac{1}{2^{2N-c\sqrt{N}}N^N}\left(\frac{2N-c\sqrt{N}}{N-c\sqrt{N}}\right)^{N-c\sqrt{N}}\left(2N-c\sqrt{N}\right)^N\\
&\sim \frac{1}{2^{2N-c\sqrt{N}}N^N}2^{N-c\sqrt{N}}\left(1+\frac{c}{2\sqrt{N}-2c}\right)^{N-c\sqrt{N}}\setminus\\
&\qquad\times(2N)^N\left(1-\frac{c}{2\sqrt{N}}\right)^N\\
F\left(c\sqrt{N},N\right) &\sim \left(1+\frac{c}{2\sqrt{N}-2c}\right)^{N-c\sqrt{N}}\left(1-\frac{c}{2\sqrt{N}}\right)^N=g(N)
\end{aligned}$$

Calculons maintenant un développement limité du logarithme népérien de $g(N)$.

$$\begin{aligned}
\ln\left(g(N)\right) &= \left(N-c\sqrt{N}\right)\ln\left(1+\frac{c}{2\sqrt{N}-2c}\right)+N\ln\left(1-\frac{c}{2\sqrt{N}}\right)\\
&= \left(N-c\sqrt{N}\right)\ln\left(1+\frac{c}{2\sqrt{N}}+\frac{c^2}{2N}+\mathcal{O}\left(\frac{1}{N^{\frac{3}{2}}}\right)\right)\setminus\\
&\qquad+N\left(-\frac{c}{2\sqrt{N}}-\frac{c^2}{8N}+\mathcal{O}\left(\frac{1}{N^{\frac{3}{2}}}\right)\right)\\
&= \left(N-c\sqrt{N}\right)\left(\frac{c}{2\sqrt{N}}+\frac{3c^2}{8N}+\mathcal{O}\left(\frac{1}{N^{\frac{3}{2}}}\right)\right)\setminus\\
&\qquad-\frac{c}{2}\sqrt{N}-\frac{c^2}{8}+\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)\\
\ln\left(g(N)\right) &= -\frac{c^2}{4}+\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)
\end{aligned}$$

On en déduit que, pour tout $c>0$,

$$\lim_{N\to+\infty}F\left(c\sqrt{N},N\right)=e^{-\frac{c^2}{4}}.$$

Si l'on définit $\theta_c(N)$ comme vérifiant $F\left(\theta_c\left(N\right),N\right)=\frac{1}{1-\varepsilon}-1$, on en déduit que $\theta_c(N)$ est asymptotiquement plus petit que $c\sqrt{N}$ lorsque $e^{-\frac{c^2}{4}}<\frac{1}{1-\varepsilon}-1$ et asymptotiquement plus grand que $c\sqrt{N}$ quand $e^{-\frac{c^2}{4}}<\frac{1}{1-\varepsilon}-1$; d'où :

$$\theta_c(N) \sim 2\sqrt{N \ln\left(\frac{1}{\varepsilon} - 1\right)}.$$

4.4.4 Translation cachée

Voilà encore un problème qui fait couler de l'encre dans la communauté quantique. Il s'agit en réalité d'un problème de sous-groupe caché — affaibli — sur les groupes diédraux. Il s'agit en quelque sorte du plus simple des problèmes de sous-groupe caché difficiles. Citons [Kup03], qui contient essentiellement tout ce que l'on sait faire. Actuellement, aucune borne inférieure non triviale n'est connue ; mais revenons à nos algorithmes probabilistes.

Le problème de la translation cachée n'est rien d'autre que le problème du sous-groupe caché dans un groupe diédral[10], si ce n'est qu'on ne s'intéresse qu'au sous-groupes d'ordre 2. Nous allons faire encore pire, en ne nous intéressant dans cette section qu'au problème de décision associé, afin d'obtenir encore une fois un problème de collision.

Pour de simples raisons de commodité, nous allons redéfinir le temps de cette section le problème de la translation cachée.

Définition 4.26
$\mathbf{Trans}_N = ([2] \times \mathbb{Z}_N, [2N], \mathbb{B}, \mathscr{S}, f)$, *où*

- $f(x) = \begin{cases} \top & \textit{si } x \textit{ est injective} \\ \bot & \textit{si } x \textit{ est injective sur } \{0\} \times \mathbb{Z}_N \textit{ et qu'il existe } t_x \in \mathbb{Z}_N \\ & \textit{tel que pour tout } k \in \mathbb{Z}_N,\ x\,(1, k + t_x) = x(0, k) \end{cases}$
  et
- $\mathscr{S} = \mathrm{dom}\,(f)$.

Soit H le sous-groupe des permutations σ de $[2] \times \mathbb{Z}_N$ telles qu'il existe j_σ vérifiant que pour tout $(b, i) \in [2] \times \mathbb{Z}_N$, $\sigma(b, i) = (b, i + b.j)$. Alors $H \times \mathfrak{S}_{[2N]} \times \{0\}$ est un sous-groupe de $\mathrm{Aut}\,(\mathbf{Trans}_N)$ agissant transitivement. Le problème élémentaire $\mathbf{Trans}_N$ est donc transitivement symétrique, et on peut là encore appliquer la proposition 4.12 pour analyser sa complexité en requêtes, car comme il s'agit d'un problème de collision dont le groupe d'automorphismes contient $\{0\} \times \mathfrak{S}_J \times \{0\}$, les algorithmes non-adaptatifs sont optimaux.

Pour la même raison, nous allons comme dans les sections 4.4.2 et 4.4.3 accorder une importance particulière à Λ, l'ensemble des suites injectives de taille T d'éléments de $[2N]$. Pour $A \subseteq [2]$, on va noter A_0 et A_1 les parties de $\mathbb{Z}_2$ telles que $A = (\{0\} \times A_0) \cup (\{1\} \times A_1)$.

[10] D_n, le groupe diédral d'ordre n, est par définition le groupe des isométries du plan conservant un polygone régulier convexe à n côtés. On peut le définir par générateurs et relations : $\langle r, s | r^n, s^2, rsrs \rangle$; on peut encore dire qu'il s'agit du produit semi-direct $\mathbb{Z}_n \rtimes \mathbb{Z}_2$. Contrairement à ce que son nom pourrait laisser penser le groupe diédral d'ordre n a $2n$ éléments.

Une fonction x cachant une translation t_x est injective sur A si et seulement si $A_1 - A_0$ ne contient pas t_x. Or quand x est uniformément distribué parmi $f^{-1}(\perp)$, t_x est uniformément distribué dans $\mathbb{Z}_N$; par conséquent,

$$\mathbb{P}_A^{\perp}(\Lambda) = 1 - \frac{|A_1 - A_0|}{N}.$$

Encore une fois, pour un A fixé, ni $\mathbb{P}_A^{\top}(\{B\})$ ni $\mathbb{P}_A^{\perp}(\{B\})$ ne dépend de B du moment que B est une suite d'éléments distincts. On a donc

$$\beta = \max_{|A|=T} \frac{1}{1 + \mathbb{P}_A^{\perp}(\Lambda)} = \max_{|A|=T} \frac{1}{2 - \frac{|A_1 - A_0|}{N}} = \frac{1}{2 - \frac{\max_{|A|=T} |A_1 - A_0|}{N}}.$$

Voilà que nous rencontrons une fois de plus un problème combinatoire non trivial : étant donné T, maximiser $A - B$ pour A et B inclus dans $\mathbb{Z}_N$ et $|A| + |B| = T$. On peut néanmoins prouver assez facilement qu'il résulte de cette expression que la complexité en requêtes probabiliste du problème de la translation cachée est bien en $\Theta\left(\sqrt{N}\right)$. On peut même calculer un équivalent relativement simple. En effet, si l'on pose A^T le sous-ensemble de $[2] \times [N]$ tel que

- $A_0^T = \left\{0, -\left\lfloor \frac{T}{2} \right\rfloor, -2\left\lfloor \frac{T}{2} \right\rfloor, \ldots, -\left\lceil \frac{T}{2} - 1 \right\rceil \left\lfloor \frac{T}{2} \right\rfloor\right\}$ et
- $A_1^T = \left\{0, 1, 2, \ldots, \left\lfloor \frac{T}{2} - 1 \right\rfloor\right\}$,

alors $\left|A^T\right| = \left\lceil \frac{T}{2} \right\rceil + \left\lfloor \frac{T}{2} \right\rfloor = T$. Par construction, on a

$$A_1^T - A_0^T = \left\{0, 1, \ldots, \left\lceil \frac{T}{2} \right\rceil \left\lfloor \frac{T}{2} \right\rfloor - 1\right\}.$$

Remarquons pour simplifier les notations que $\left\lceil \frac{T}{2} \right\rceil \left\lfloor \frac{T}{2} \right\rfloor = \left\lfloor \frac{T^2}{4} \right\rfloor$. On a donc $\left|A_1^T - A_0^T\right| = \max\left(N, \left\lfloor \frac{T^2}{4} \right\rfloor\right)$; comme A^T est clairement optimal, cela donne une expression plus simple de β :

$$\beta = \min\left(1, \frac{1}{2 - \frac{\left\lfloor \frac{T^2}{4} \right\rfloor}{N}}\right).$$

Il en résulte que la complexité en requêtes probabiliste de $\mathbf{Trans}_N$, pour une probabilité d'erreur au plus $\varepsilon \in \left[0; \frac{1}{2}\right]$, vaut exactement

$$\left\lceil 2\sqrt{\frac{1 - 2\varepsilon}{1 - \varepsilon} N} \right\rceil.$$

4.5 Quelques comparaisons

4.5.1 La distance en variation

Nous allons définir dans cette section, pour un problème de décision élémentaire et un nombre de requêtes fixé, un autre paramètre α, déjà utilisé par exemple dans [BYKS01]. Sa définition, basée sur la distance en variation totale, est plus simple que celle de β, mais α est néanmoins assez proche de β. Commençons par donner la définition de la distance en variation totale.

Définition 4.27
Soit p et q deux distributions de probabilité sur un ensemble fini E. Leur **distance en variation totale**, *notée $\partial(p,q)$, vaut*

$$\partial(p,q) = \max_{F \subseteq E} \sum_{x \in F} |p(x) - q(x)| .$$

Il est bien connu et à peu près immédiat que l'on peut également écrire $\partial(p,q)$ sous ces formes :

$$\partial(p,q) = \frac{1}{2} \sum_{x \in E} |p(x) - q(x)| = \sum_{x/p(x)>q(x)} p(x) - q(x).$$

Soit maintenant $\mathscr{P} = (I, J, \mathbb{B}, \mathscr{S}, f)$ un problème de décision élémentaire. Pour $A \subseteq I$, $\mathbb{P}_A^\top$ et $\mathbb{P}_A^\perp$ définissent des distributions de probabilité sur $[J]^T$. En ce sens on peut définir α_A comme étant la distance en variation totale entre $\mathbb{P}_A^\top$ et $\mathbb{P}_A^\perp$; étant donné notre définition de $\mathbb{P}_A^\top$ et $\mathbb{P}_A^\perp$, on a, outre les expressions déjà formulées dans le cas général de distributions de probabilité :

$$\alpha_A = \max_{X \subseteq J^T} \left| \mathbb{P}_A^\top(X) - \mathbb{P}_A^\perp(X) \right| .$$

Le paramètre α est alors simplement défini comme étant le maximum des α_A :

$$\alpha = \max_{A \in I^T} \alpha_A .$$

Nous allons maintenant montrer que α fournit une assez bonne approximation de β.

Proposition 4.28
$\frac{1}{2-\alpha} \leq \beta \leq \frac{1}{2} + \frac{\alpha}{2}$.

Preuve : Considérons la fonction convexe

$$\mathscr{E} : \left(\begin{array}{ccc} [0;1] & \to & \left[\frac{1}{2};1\right] \\ p & \mapsto & \max\limits_{\substack{A \in I^T \\ X \subseteq J^T}} \quad p \cdot \mathbb{P}_A^\top(X) + (1-p)\left(1 - \mathbb{P}_A^\perp(X)\right) \end{array} \right) .$$

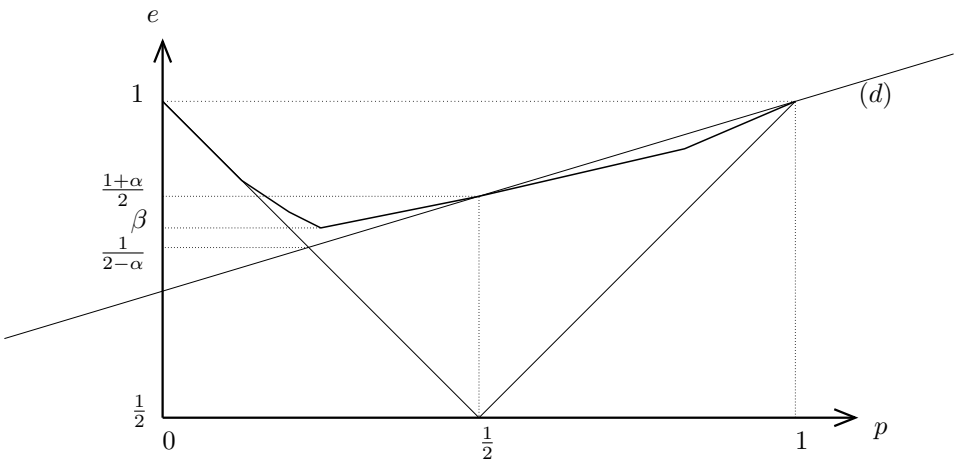

FIG. 4.7 – α et β sont dans une figure.

Quasiment par définition, $\mathscr{E}\left(\frac{1}{2}\right) = \frac{1+\alpha}{2}$. Comme $\beta = \min_{p\in[0;1]} \mathscr{E}(p)$, on doit avoir $\beta \leq \frac{1}{2} + \frac{\alpha}{2}$. Supposons, sans perte de généralité, que le minimum de $\mathscr{E}$ est atteint sur $\left[0; \frac{1}{2}\right]$ — l'autre cas est tout à fait symétrique. En considérant $X = \emptyset$, on obtient $\mathscr{E}(p) \geq 1 - p$. De plus, la convexité de $\mathscr{E}$ fournit une autre relation, ainsi qu'on peut le visualiser sur la figure 4.7. En effet, le graphe de $\mathscr{E}$ est « au-dessus » de la droite (d) passant passant par les points $\left(\frac{1}{2}; \frac{1+\alpha}{2}\right)$ et $(1; 1)$. β est donc plus grand que l'ordonnée du point d'intersection de (d) et de la droite d'équation $e = 1 - p$. L'équation de (d) étant

$$(d) : e = (1 - \alpha)\, p + \alpha,$$

on en déduit bien

$$\beta \geq \frac{1}{2 - \alpha}.$$

□

Ces inégalités sont optimales. Pour le prouver, choisissons λ et μ des rationnels quelconques de $[0; 1]$ vérifiant les relations de la proposition 4.28, c'est-à-dire $\frac{1}{2-\lambda} \leq \mu \leq \frac{1+\lambda}{2}$. Nous allons construire un problème élémentaire tel que pour une seule requête, les paramètre α et β valent respectivement λ et μ. Pour cela, commençons par choisir N un entier strictement positif tel que λN et $\frac{(2-\lambda)\mu-1}{\mu-\lambda} N$ soient tous deux entiers. Soit f la fonction partielle de $[3]^{[N]}$ dans $\mathbb{B}$ qui à x associe

$$f(x) = \begin{cases} \top & \text{si } n_{x,0} = \frac{(2-\lambda)\mu-1}{\mu-\lambda} N \quad \text{et } n_{x,2} = 0 \\ \bot & \text{si } n_{x,0} = 0 \qquad\qquad\quad \text{et } n_{x,2} = \lambda N \end{cases}$$

Vérifions que le problème élémentaire $\mathscr{P} = ([N], [3], \mathbb{B}, \mathrm{dom}(f), f)$, pour les algorithmes de complexité en requêtes 1, possède les paramètres α et β adéquats. La première des vérifications consiste à s'assurer que $\frac{(2-\lambda)\mu-1}{\mu-\lambda}$ est bien dans $[0;1]$. D'abord, comme $\frac{1}{2\lambda} \leq \mu$, le numérateur $(2-\lambda)\,\mu - 1$ est positif et, comme $\frac{1}{2-\lambda} \geq \lambda$ pour $\lambda \in [0;1]$, le dénominateur l'est aussi ; la fraction est donc positive. Ensuite, comme $\mu \leq 1$ et $1 - \lambda \geq 0$, on a $\mu\,(1-\lambda) \leq 1 - \lambda$, ce que l'on peut autrement écrire $(2-\lambda)\,\mu - 1 \leq \mu - \lambda$; la fraction est donc plus petite que 1. On peut remarquer en passant qu'à ce stade nous n'avons pas encore utilisé la relation $\mu \leq \frac{1+\lambda}{2}$.

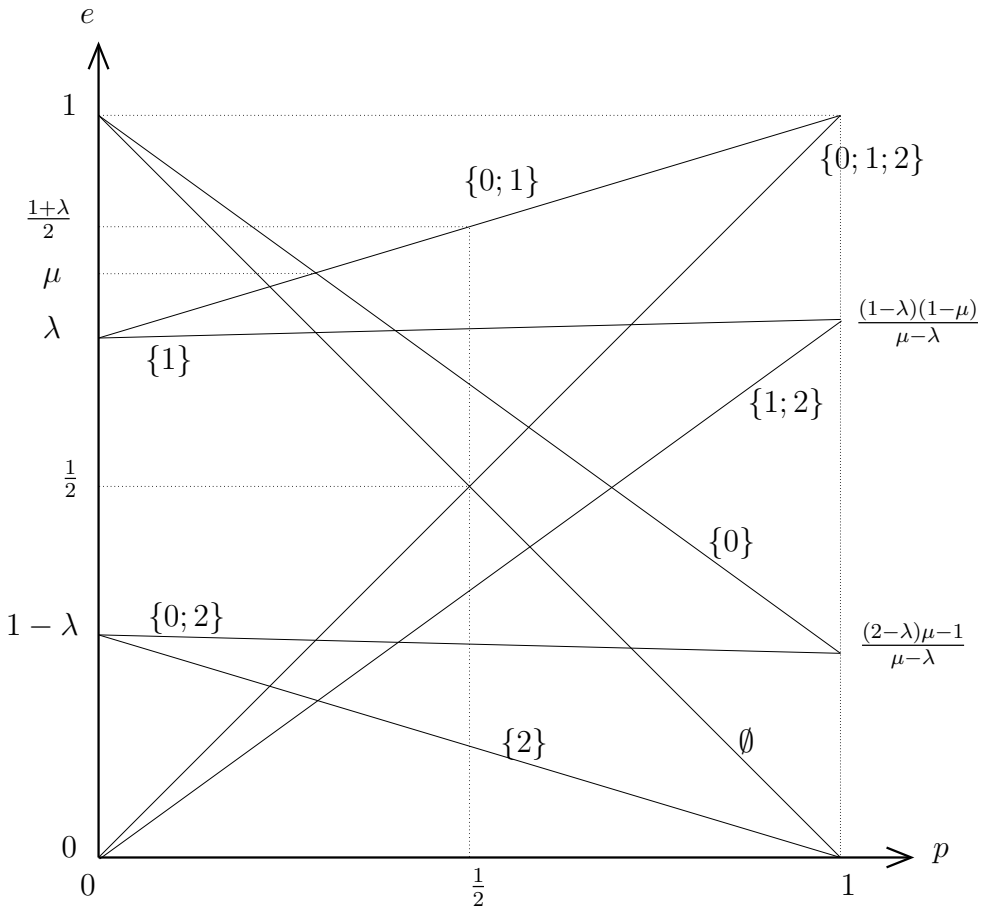

FIG. 4.8 – Optimalité de la proposition 4.28

Il faut ensuite s'appliquer à examiner chacune des droites $\Delta_{A,X}$ pour X prenant toutes les valeurs parmi $\mathfrak{P}([3])$. Au lecteur égaré par mégarde dans ce paragraphe nous épargnerons une étude de cas quelque peu fastidieuse, pour lui recommander plutôt une contemplation prolongée de la figure 4.8, qui nous l'espérons, parle d'elle-même avec assez de clarté. Elle nous dit en particulier l'usage qui est fait de l'inégalité $\mu \leq \frac{1+\lambda}{2}$: sans cela le paramètre

α, qui est la valeur du maximum de toutes ces droites en $\frac{1}{2}$, ne vaudrait point $\frac{1+\lambda}{\mu}$ mais se trouverait sur la droite associée à $\{0\}$, et aurait donc une valeur plus ésotérique en plus d'être plus élevée.

Alors, pourquoi ces inégalités sont-elles intéressantes ? D'abord parce que α est un paramètre certainement plus facile à calculer que β. Ensuite, si l'on peut encadrer la valeur de α, on encadre du même coup la valeur de β de manière assez efficace pour pouvoir donner l'ordre de grandeur de la complexité en requêtes du problème considéré. Par exemple, si $\alpha \in \left[\frac{1}{3}; \frac{2}{3}\right]$, alors $\beta \in \left[\frac{3}{5}; \frac{5}{6}\right]$. Voyons si le paramètre α des exemples de la section 4.4 sont faciles à calculer.

D'après l'analyse faite dans la section 4.4.1, pour $\mathbf{RTNT}_N$, avec T requêtes, on a $\alpha = \frac{T}{N}$. Rien à signaler ici, puisque β était déjà particulièrement facile à calculer ; tout au plus peut-on remarquer que β est minimal dans ce cas car il vaut exactement $\frac{1}{2-\alpha}$, ce qui est une conséquence du fait que le minimum de la fonction qui permet de déterminer β est atteint sur une des diagonales du carré — voir la figure 4.3. En fait, la même remarque est valable pour les trois autres problèmes, la translation cachée, « one-to-one versus two-to-one » et le problème de Simon ; le minimum est à chaque fois atteint sur une diagonale du carré, ce qui fait que l'on a $\alpha = 2 - \frac{1}{\beta}$, et que par conséquent α n'est pas plus beaucoup facile à calculer que β.

4.5.2 Sensibilité par blocs

Dans cette section et la suivante nous discuterons sommairement des relations qui peuvent être établies entre ce que nous avons raconté jusqu'à présent dans ce chapitre et ce qui l'est dans [BYKS01]. Commençons par parler de la notion sensibilité par blocs, notion introduite par Noam Nisan dans [Nis91].

Nous allons d'abord donner quelques définitions qui se veulent des versions simplifiées de celles que l'on peut trouver dans [BYKS01]. Soit $\mathscr{P} = (I, J, R, \mathscr{S}, f)$ un problème élémentaire.

Définition 4.29
Une **approximation** *de $\mathscr{P}$ est une fonction $C : \mathscr{S} \to \mathfrak{P}(R)$ telle que pour tout $x \in \mathscr{S}$, $f(x) \in C(x)$. Des fonctions x et y de $\mathscr{S}$ sont dites* **disjointes selon** *C si $C(f) \cap C(g) = \emptyset$.*

L'*approximation triviale* est par définition l'application $x \mapsto \{f(x)\}$.

Définition 4.30
$\mathscr{P}$ est **sensible selon** *C à un ensemble $X \subseteq I$ sur la fonction $x \in \mathscr{S}$ s'il existe une application $y \in \mathscr{S}$ coïncidant avec x sur $I \setminus X$ et disjointe de x selon C.*

La **sensibilité par blocs de $\mathscr{P}$ sur x selon** C*, notée* $bs_C(\mathscr{P}, x)$*, est le cardinal maximal d'une partition de I telle que $\mathscr{P}$ soit sensible selon C à chacune des parties de la partition, pour x.*

La **sensibilité par blocs de $\mathscr{P}$ selon** C *est le maximum des* $bs_C(\mathscr{P}, x)$ *pour $x \in \mathscr{S}$, et est notée* $bs_C(\mathscr{P})$*.*

La sensibilité par blocs permet de donner une borne inférieure sur la complexité en requêtes.

Définition 4.31
On dit qu'un algorithme probabiliste (C, ε)**-approxime** $\mathscr{P}$ *si, pour tout $x \in \mathscr{S}$, la probabilité qu'il réponde un élément $C(x)$ sur l'entrée x est au moins $1 - \varepsilon$. La* (C, ε)**-complexité en requêtes** *de $\mathscr{P}$, notée* $S_{C,\varepsilon}(\mathscr{P})$*, est alors naturellement le minimum des complexités en requêtes des algorithmes (C, ε)-approximant $\mathscr{P}$.*

Si C est l'approximation triviale, alors $S_{C,\varepsilon}(\mathscr{P})$ est simplement la complexité en requêtes probabiliste de $\mathscr{P}$. La proposition suivante elle, tient pour une approximation C quelconque.

Proposition 4.32
Pour toute approximation C de $\mathscr{P}$ et tout $\varepsilon \in \left[0; \frac{1}{2}\right]$,

$$S_{C,\varepsilon}(\mathscr{P}) \geq (1 - 2\varepsilon)\, bs_C(\mathscr{P}).$$

Cette proposition est prouvée dans [BYKS01], où il est cependant implicite que $\mathscr{P}$ est un problème total. Les problèmes à promesse étant *a priori* plus facile, on s'attendrait à ce que la borne inférieure soit plus basse dans le cas général ; il n'en est rien, car la proposition telle que nous l'avons énoncée reste vraie, même lorsque $\mathscr{P}$ n'est pas total. D'une part, la preuve donnée dans [BYKS01] n'utilise pas vraiment l'hypothèse de totalité de $\mathscr{P}$. D'autre part, on peut déduire simplement le résultat général de celui sur les problèmes totaux. En effet, on peut commencer par étendre $\mathscr{P}$ en un problème total $\mathscr{P}'$ d'une manière arbitraire, c'est-à-dire que l'on a $\mathscr{P}' = \left(I, J, R, J^I, f'\right)$ avec $f'|_{\mathscr{S}} = f$. On définit ensuite l'approximation C' de $\mathscr{P}'$ de la sorte :

$$C'(x) = \begin{cases} C(x) & \text{si } x \in \mathscr{S} \\ \mathfrak{P}(R) & \text{sinon} \end{cases}.$$

La première remarque qui s'impose est que x et y ne sont pas C'-disjoints si l'une des deux n'est pas dans $\mathscr{S}$; par conséquent $bs_{C'}(\mathscr{P}') = bs_C(\mathscr{P})$ — en fait un algorithme (C, ε)-approxime $\mathscr{P}$ si et seulement si il (C', ε)-approxime $\mathscr{P}'$. On a donc $S_{C,\varepsilon}(\mathscr{P}) = S_{C',\varepsilon}(\mathscr{P}')$, et voilà qui prouve la proposition dans le cas général.

Voyons maintenant ce que donne cette borne inférieure sur les exemples vus dans la section 4.4. Comme les seules approximations qui nous intéressent

sont les approximations triviales, nous omettrons dorénavant la mention de C et il sera entendu implicitement qu'il s'agit toujours de l'approximation triviale.

Recherche dans un tableau non-trié

La sensibilité par blocs de $\mathbf{RTNT}_N$ est facile à déterminer. Sur la fonction nulle elle vaut N, donc elle vaut N pour $\mathbf{RTNT}_N$, ce qui d'après la proposition 4.32 permet d'affirmer que la complexité en requêtes probabiliste de $\mathbf{RTNT}_N$, pour une erreur au plus ε, vaut au moins $(1-2\varepsilon)N$, ce qui est proche de la valeur exacte de $\left\lceil \frac{1-2\varepsilon}{1-\varepsilon} N \right\rceil$ établie dans la section 4.4.1.

One-to-one versus two-to-one

La sensibilité par blocs de « one-to-one versus two-to-one » est de 2. Or la sensibilité par blocs est une fonction croissante des problèmes, au sens la sensibilité par blocs d'un sous-problème élémentaire de $\mathscr{P}$ vaut au plus celle de $\mathscr{P}$, et la problème de Simon et le problème de décision associé à la translation cachée sont des sous-problèmes de « one-to-one versus two-to-one » ; leur sensibilité par blocs vaut donc également 2 tout rond. Pour ces trois problèmes, la proposition 4.32 ne propose en conséquence qu'une borne inférieure constante.

4.5.3 Distance de Hellinger

Soit $\mathscr{P} = \left(I, J, \mathbb{B}, J^I, f\right)$ un problème totalement symétrique[11]. Le théorème 8 de [BYKS01] donne dans ce cas une borne inférieure sur $S_C\left(\mathscr{P}, \varepsilon\right)$. Pour énoncer ce résultat nous devons introduire une nouvelle distance entre distributions.

Définition 4.33
Soit p et q deux distributions de probabilité sur un ensemble fini E. Leur **distance de Hellinger**, *notée $h(p,q)$, vaut*

$$h(p,q) = \sqrt{1 - \sum_{x \in E} \sqrt{p(x)q(x)}}.$$

Chaque fonction x de J^I induit une probabilité de distribution p_x sur J, définie par

$$p_x(j) = \left|x^{-1}(j)\right|.$$

Remarquons que, comme $\mathscr{P}$ est totalement symétrique, $f(x)$ ne dépend que de p_x. Notons, pour une approximation C de $\mathscr{P}$, $h_C\left(\mathscr{P}\right)$ le minimum

[11] Voir la définition 4.20

des distances de Hellinger entre P_x et P_y pour x et y dans $\mathscr{S}$ et disjointes selon C. D'après le théorème 8 de [BYKS01], on a, à supposer que $\varepsilon < \frac{1}{4}$, $h_C(\mathscr{P}) \leq \frac{1}{2}$ et $S_C(\mathscr{P}, \varepsilon) \leq \frac{N}{4}$, la borne inférieure suivante :

$$S_{C,\varepsilon}(\mathscr{P}) \geq \frac{1}{(2h_C(\mathscr{P}))^2} \ln \frac{1}{4\varepsilon + \mathcal{O}\left(\frac{1}{N}\right)}.$$

En réalité, cette borne n'est énoncée dans [BYKS01] qu'à propos des problèmes totaux, mais une manipulation élémentaire identique à celle vue dans la section 4.5.2 permet de la généraliser immédiatement à tous les problèmes totalement symétriques. Une restriction évidente de cette méthode, qui est dans le même temps sa grande force puisqu'elle simplifie grandement les calculs, est de ne prendre en compte que les distributions de probabilité p_x, ignorant volontairement tous les effets, potentiellement complexes, pouvant se produire lorsque les fonctions de $\mathscr{S}$ sont comparées sur plusieurs entrées à la fois.

Encore une fois, seules les approximations triviales nous intéressent, aussi allons-nous dorénavant omettre de préciser l'approximation. Faisons rapidement le tour des problèmes de la section 4.4.

Recherche dans un tableau non trié

Pour $\mathbf{RTNT}_N$, la distance de Hellinger entre x et y, lorsque $f(x) \neq f(y)$, vaut toujours $\sqrt{1 - \sqrt{1 - \frac{1}{N}}}$. On a donc

$$S_\varepsilon(\mathbf{RTNT}_N) \geq \frac{1}{4\left(1 - \sqrt{1 - \frac{1}{N}}\right)} \ln \frac{1}{4\varepsilon + \mathcal{O}\left(\frac{1}{N}\right)},$$

d'où l'on tire effectivement $S_\varepsilon(\mathbf{RTNT}_N) = \Omega(N)$.

One-to-one versus two-to-one

Il n'est pas difficile de vérifier que la distance de Hellinger entre deux fonctions x et y telles que $f(x) \neq f(y)$ vaut toujours $\sqrt{1 - \frac{1}{\sqrt{2}}}$. Comme il s'agit d'une constante, on n'obtient qu'une borne inférieure triviale par cette méthode. En fait, la méthode de la distance de Hellinger, comme celle de la sensibilité par blocs, se heurte là au cas typique où elle ne peut rien dire. Le problème « one-to-one versus two-to-one », bien que relativement difficile, présente la particularité que si $f(x) \neq f(y)$, alors x et y diffèrent sur une assez grande proportion de leurs entrées, ici au moins $\frac{1}{2}$. Une méthode qui se contente d'examiner les requêtes une à une s'arrête à ce constat et s'imagine que le problème ne doit pas être bien difficile puisque les fonctions de $f^{-1}(\top)$ sont si différentes des fonctions de $f^{-1}(\bot)$. Elles sont certes très différentes, mais le problème est qu'en réalité la boîte noire ne fournit aucune

autre information exploitable que « perdu, cherche encore » à l'algorithme du moment qu'aucune collision n'a été trouvée.

Conclusion

Les résultats présentés dans ce mémoire sont appelés à être étendus naturellement dans plusieurs directions.

Pour commencer, les algorithmes quantiques non-adaptatifs mériteraient probablement d'être plus étudiés qu'ils ne l'ont été jusqu'à présent. Il n'existe certes pas de principe de Yao quantique satisfaisant, mais il est possible d'utiliser la dualité de la programmation semi-définie de la même manière que l'on utilise la dualité de la programmation linéaire dans le cas probabiliste, avec un même usage des symétries. Cela aurait d'autant plus d'intérêt que la majorité des algorithmes quantiques pour les problèmes du sous-groupe caché sont en réalité non-adaptatifs ; c'est en particulier le cas de l'algorithme dit standard, qui peut être défini non pas seulement pour les groupes commutatifs comme nous l'avons fait dans la section 2.2.1, mais pour tous les groupes finis.

De manière plus générale, ce qui manque cruellement aux méthodes de bornes inférieures quantiques, c'est de pouvoir traiter de manière un tant soit peu systématique les problèmes de collision symétriques. En effet, la méthode par adversaire usuelle est souvent inefficace, et la méthode polynomiale qui semble plus naturelle parce qu'elle utilise justement les symétries du problème pour fonctionner, est très loin de la systématisation recherchée, car elle repose en général sur des résultats difficiles de la théorie de l'approximation, qu'il faut rechercher au cas par cas. Une telle méthode générique ne serait pas aussi simple que dans le cas probabiliste, pour plusieurs raisons. D'abord, si elle l'était, elle aurait déjà été découverte. Ensuite, il y a manifestement, parmi les complexités en requêtes quantiques des problèmes de collision, une disparité telle que l'on a peine à en imaginer une description simple. Qu'est-ce qui fait que les problèmes de sous-groupe caché abéliens sont de complexité logarithmique en la taille du groupe, tandis que la recherche dans un tableau non trié nécessite environ $\sqrt{n}$ requêtes pour un tableau de taille n ? Pour un problème de sous-groupe caché abélien, on peut expliquer que la complexité soit basse par le fait qu'il existe une transformation unitaire, exploitant la structure algébrique du problème, et rendant le problème facile à résoudre, j'ai nommé la transformée de Fourier. Avoir une explication à la fois aussi peu élémentaire et aussi adaptée — car il fait peu de doute que c'est la meilleure que l'on puisse donner, mathématiquement

parlant — est peu encourageant pour qui voudrait chercher une généralisation. Mettons donc de côté les problèmes de ce type, pour nous cantonner au problème possédant moins de structure algébrique, et pourquoi pas tout simplement aux problèmes totalement symétriques. Dans ce cas la méthode polynomiale permet de se ramener au moins à des problèmes d'approximation polynomiale multidimensionnelle relativement raisonnables, mais l'idéal serait d'avoir quelque chose de plus maniable, dans le goût des méthodes par adversaire existantes.

Enfin, le point qui nous semble à l'heure actuelle le plus obscur est la complexité en requêtes des problèmes de sous-groupe caché non abéliens. On ne connaît à ce jour aucune autre borne inférieure que celle découlant du théorème 3.16, bien qu'il existe des résultats intéressants allant dans ce sens, par exemple le récent [HMR$^+$06]. Il nous semble qu'il serait à la fois très intéressant en soi et très prometteur de développements ultérieurs de prouver une nouvelle borne inférieure pour un problème de sous-groupe caché non abélien.

Remerciements

Je remercie au premier chef, bien évidemment, mes directeurs de thèse Pascal Koiran et Natacha Portier, en la proverbiale absence desquels rien n'aurait été possible. Ils m'ont porté à bout de bras durant ces années de lutte. Ils ont été pour moi indispensables sur tous les plans, des aspects les plus pratiques du travail de recherche aux mises au point sur des résultats scientifiques, en passant bien entendu par la définition d'objectifs pertinents ; en un mot, ils m'ont guidé avec autant de nécessité qu'ils ont dû guider leurs jeunes enfants, que j'ai vu grandir durant cette période.

Je remercie Frédéric Magniez, qui un beau jour m'a présenté pour la première fois le principe et les problématiques du calcul quantique, en particulier les problèmes de sous-groupe caché. Il a ce jour-là conquis un étudiant indécis à la cause de ce domaine. Depuis lors il s'est toujours montré disponible pour l'aider et le conseiller.

Quand je cherche à comprendre quelque chose, je pose souvent beaucoup de questions, et pas des plus pertinentes, à beaucoup de gens, et de manière répétée. Pour leur patience et leurs éclaircissements, je remercie donc Xavier Caruso, Yves de Cornulier, Gábor Ivanyos, Emmanuel Jeandel, Frédéric Magniez, Cris Moore, Joël Riou, Miklos Santha, Pranab Sen et Yves Verhoeven, en espérant ne pas trop en oublier.

Sur un plan moins « professionnel », j'ai pu constater que la vie dans un laboratoire d'informatique peut être tout à fait agréable. Voici une petite liste non exhaustive d'humains peuplant ces lieux, et dont la compagnie s'est avérée néanmoins fort agréable : Florent B., Anne B., Thomas C., Marc K., Stéphane L.R., Laurent L., Emmanuel M., Guillaume M., Victor P., Damien R., Sylvain S., David T. Je remercie chacun d'eux, même lorsque leur principale action sur moi fut un franc encouragement à la paresse — cette personne se reconnaîtra.

Sans oublier Rachmaninov et tous les autres, bien entendu.

Notations

- $\top$ et $\bot$ représentent respectivement les valeurs booléennes « vrai » et « faux ».
- $\mathbb{B} = \{\top, \bot\}$.
- $\lfloor x \rfloor$ est la partie entière de x.
- $\lceil x \rceil$ est le plus petit entier supérieur ou égal à x, soit $\lceil x \rceil = -\lfloor -x \rfloor$.
- $\mathbb{B} = \{\top, \bot\}$.
- $\delta_{i,j} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{sinon} \end{cases}$ est le symbole de Kronecker.
- $|A|$ est le cardinal de A.
- F^E est l'ensemble des fonctions de E dans F.
- $\mathfrak{S}_X$ est le groupe des permutations des éléments de X.
- $S(G)$ est, pour un groupe G, l'ensemble des sous-groupes de G.
- ${}^2X$ est l'ensemble des paires d'éléments de X.
- $\mathrm{dom}(s)$ est le domaine de la fonction s.
- $f \circ g$ est la composition des applications f et g, dans le sens que $f \circ g(x) = f(g(x))$.
- $f(A)$ et $f^{-1}(A)$ sont respectivement l'image directe et l'image réciproque de l'ensemble A par l'application f.
- id et $\mathbf{I}$ sont respectivement l'application et la matrice ou le morphisme identité.
- $H \leq G$ signifie que H est un sous-groupe de G.
- d_1 est la distance en norme 1, ainsi définie :

$$d_1\left((x_1, \ldots, x_n), (y_1, \ldots, y_n)\right) = \sum_{i=1}^{n} |x_i - y_i| .$$

- $\mathbb{F}_l$, où l est une puissance d'un nombre premier, est le corps à l éléments, qui est unique à isomorphisme près.
- $\Re(z)$ et $\Im(z)$ dénotent respectivement la partie réelle et la partie imaginaire du nombre complexe z.
- $\oplus$ et $\ominus$ sont respectivement l'addition et la soustraction modulo.
- $\bigsqcup_{i \in I} X_i = X_i \sqcup \cdots \sqcup X_j$ est la réunion de la famille d'ensembles deux à deux disjoints $(X_i)_{i \in I}$.

- $X^* = \bigcup_{n\in\mathbb{N}} X^n$ est l'ensemble des suites finies d'éléments de X
- $\mathfrak{P}(X)$ est l'ensemble des parties de X.
- $X - Y = \{x - y / x \in X, y \in Y\}$ est la différence de Minkowski entre X et Y.
- $f|_X$ est la restriction de la fonction f à l'ensemble X.
- $\langle a, b, c, \ldots \rangle_E$ est la sous-structure de E engendrée par les éléments $a, b, c, \ldots$.

Bibliographie

[Aar02] Scott Aaronson : Quantum lower bound for the collision problem. *In STOC '02 : Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 635–642, New York, NY, USA, 2002. ACM Press.

[Aar04] Scott Aaronson : Lower bounds for local search by quantum arguments. *In Proc. STOC 2004*, pages 465–474. ACM, 2004.

[AK06] V. Arvind et Piyush P. Kurur : Graph isomorphism is in SPP. *Inf. Comput.*, 204(5):835–852, 2006.

[Amb99] Andris Ambainis : A better lower bound for quantum algorithms searching an ordered list. *In FOCS '99 : Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 352, Washington, DC, USA, 1999. IEEE Computer Society.

[Amb02] Andris Ambainis : Quantum lower bounds by quantum arguments. *J. Comput. Syst. Sci.*, 64(4):750–767, 2002.

[Amb03] Andris Ambainis : Polynomial degree vs. quantum query complexity. *In FOCS '03 : Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 230, Washington, DC, USA, 2003. IEEE Computer Society.

[Amb05] A. Ambainis : Polynomial degree and lower bounds in quantum complexity : Collision and element distinctness with small range. *Theory of Computing*, 1(3):37–46, 2005.

[AS04] Scott Aaronson et Yaoyun Shi : Quantum lower bounds for the collision and the element distinctness problems. *J. ACM*, 51(4):595–605, 2004.

[BBC+98] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca et Ronald de Wolf : Quantum lower bounds by polynomials. *In FOCS '98 : Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 352, Washington, DC, USA, 1998. IEEE Computer Society.

[BD99] Harry Buhrman et Wim Van Dam : Quantum bounded query complexity. *In COCO '99 : Proceedings of the Fourteenth An-*

nual IEEE Conference on Computational Complexity, page 149, Washington, DC, USA, 1999. IEEE Computer Society.

[Bea97] Robert Beals : Quantum computation of Fourier transforms over symmetric groups. *In STOC '97 : Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 48–53, New York, NY, USA, 1997. ACM Press.

[BH97] Gilles Brassard et Peter Høyer : An exact quantum polynomial-time algorithm for Simon's problem. *In Israel Symposium on Theory of Computing Systems*, pages 12–23, 1997.

[Bol04] Béla Bollobás : *Extremal Graph Theory*. Dover Publications, Incorporated, 2004.

[BSS03] Howard Barnum, Michael Saks et Mario Szegedy : Quantum query complexity and semi-definite programming. *complexity*, 00:179, 2003.

[BYKS01] Ziv Bar-Yossef, Ravi Kumar et D. Sivakumar : Sampling Algorithms : lower bounds and applications. *In Proc. STOC 2001*, pages 266–275. ACM, 2001.

[Cam99] Peter J. Cameron : *Permutation groups*, volume 45 de *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 1999.

[Deu85] David Deutsch : Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Ser. A*, A400:97–117, 1985.

[dGR02] Mart de Graaf et Ronald de Wolf : On quantum versions of the Yao principle. *In STACS '02 : Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 347–358, London, UK, 2002. Springer-Verlag.

[EHK04] Mark Ettinger, Peter Høyer et Emanuel Knill : The quantum query complexity of the hidden subgroup problem is polynomial. *Inf. Process. Lett.*, 91(1):43–48, 2004.

[EZ64] H. Echlich et K. Zeller : Schwankung von polynomen zwischen gitterpunkten. *Mathematische Zeitschrift*, 86:41–44, 1964.

[Fey82] Richard Feynman : Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6&7):467–488, 1982.

[Gro96] Lov K. Grover : A fast quantum mechanical algorithm for database search. *In STOC '96 : Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, New York, NY, USA, 1996. ACM Press.

[Hal02] Lisa Ruth Hales : *The quantum Fourier transform and extensions of the abelian hidden subgroup problem*. Thèse de doctorat,

University of California at Berkeley, 2002. Chair-Umesh V. Vazirani.

[Hel23] Eduard Helly : Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jahresbericht Deutsch. Math. Verein.*, 32:175–176, 1923.

[HMR+06] Sean Hallgren, Cristopher Moore, Martin Rötteler, Alexander Russell et Pranab Sen : Limitations of quantum coset states for graph isomorphism. *In STOC '06 : Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 604–617, New York, NY, USA, 2006. ACM Press.

[Hø97] Peter Høyer : Conjugated operators in quantum algorithms. Rapport technique, University of Southern Denmark, 1997.

[Joz98] Richard Jozsa : Quantum algorithms and the Fourier transform. *Proc Roy Soc Lond A*, pages 323–337, 1998.

[KNP] Pascal Koiran, Vincent Nesme et Natacha Portier : A quantum lower bound for the query complexity of Simon's problem. `http://www.arxiv.org/pdf/quant-ph/0501060.`

[KNP05a] P. Koiran, V. Nesme et N. Portier. : A quantum lower bound for the query complexity of Simon's problem. *In Proc. ICALP 2005*, volume 3580 de *Lecture Notes in Computer Science*, pages 1287–1298. Springer, 2005.

[KNP05b] Pascal Koiran, Vincent Nesme et Natacha Portier : The quantum query complexity of the abelian hidden subgroup problem. Rapport technique RR2005-17, LIP, `http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2006/RR2006-27.ps.gz`, 2005.

[KNP06] Pascal Koiran, Vincent Nesme et Natacha Portier : On the probabilistic query complexity of transitively symmetric problems. Rapport technique, LIP, 2006.

[KS04] Hans Kurzweil et Bernd Stellmacher : *The Theory of Finite Groups, An Introduction.* Universitext. Springer, 2004.

[KST93] Johannes Köbler, Uwe Schöning et Jacobo Torán : *The graph isomorphism problem : its structural complexity.* Birkhauser Verlag, Basel, Switzerland, Switzerland, 1993.

[Kup03] Greg Kuperberg : A subexponential-time quantum algorithm for the dihedral hidden subgroup problem, 2003.

[Kut03] Samuel Kutin : Quantum lower bound for the collision problem. *quant-ph/0304162*, 2003.

[Kut05] Samuel Kutin : Quantum lower bound for the collision problem with small range. *Theory of Computing*, 1:29–36, 2005.

[Lan65] Serge Lang : *Algebra*. Addison-Wesley, 1965.

[LM04] S. Laplante et F. Magniez : Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. *In Proc. 19th IEEE Annual Conference on Computational Complexity (CCC'04)*. IEEE, 2004.

[Lom04] Chris Lomont : The hidden subgroup problem - review and open problems, 2004.

[NC00] Michael A. Nielsen et Isaac L. Chuang : *Quantum computation and quantum information*. Cambridge University Press, New York, NY, USA, 2000.

[Nis91] Noam Nisan : CREW PRAMs and decision trees. *SIAM J. Comput.*, 20(6):999–1007, 1991.

[NS94] Noam Nisan et Mario Szegedy : On the degree of boolean functions as real polynomials. *Comput. Complex.*, 4(4):301–313, 1994.

[NY04] Harumichi Nishimura et Tomoyuki Yamakami : An algorithmic argument for nonadaptive query complexity lower bounds on advised quantum computation (extended abstract). *In* Jirí Fiala, Václav Koubek et Jan Kratochvíl, éditeurs : *MFCS*, volume 3153 de *Lecture Notes in Computer Science*, pages 827–838. Springer, 2004.

[Pat92] Ramamohan Paturi : On the degree of polynomials that approximate symmetric boolean functions (preliminary version). *In STOC '92 : Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 468–474, New York, NY, USA, 1992. ACM Press.

[Phi03] Pierre Philipps : Bornes inférieures en calcul quantique : Méthode par adversaire vs. méthode des polynômes. Rapport de stage de DEA, effectué au LRI sous la direction de Frédéric Magniez, 2003.

[RC66] T. J. Rivlin et E. W. Cheney : A comparison of uniform approximations on an interval and a finite subset thereof. *SIAM Journal on Numerical Analysis*, 3(2):311–320, june 1966.

[Ros73] A. L. Rosenberg : On the time required to check properties of graphs : A problem. *SIGACT News*, pages 15–16, 1973.

[RP98] Eleanor G. Rieffel et Wolfgang Polak : An introduction to quantum computing for non-physicists, 1998.

[Sho97] Peter W. Shor : Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[Sim94] Daniel R. Simon : On the power of quantum computation. *In Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 116–123, 1994.

[Sim97] Daniel R. Simon : On the power of quantum computation. *SIAM J. Comput.*, 26(5):1474–1483, 1997.

[ŠS06] Robert Špalek et Mario Szegedy : All quantum adversary methods are equivalent. *Theory of Computing*, 2(1):1–18, 2006.

[Vaz06] Vijay V. Vazirani : *Algorithmes d'approximation*. IRIS. Springer, 2006. Traduction de Nicolas Schabanel.

[Wik] Wiki : Complexity zoo. `http://qwiki.caltech.edu/wiki/Complexity_Zoo`.

[Yao77] Andrew Chi-Chih Yao : Probabilistic computations : Toward a unified measure of complexity (extended abstract). *In FOCS*, pages 222–227, 1977.