Chapter 3

Algorithme de Deutsch et Jozsa

Dans ce chapitre nous exposons un algorithme quantique qui contient déjà la plupart des ingrédients d'une classe plus large. Cette classe, comme nous le verrons, traite le problème plus général de la recherche de symétries cachées. L'algorithme de Shor, permettant la factorisation en temps polynomial par rapport aux nombres de bits de l'entier, appartient à cette classe. Il est suspecté, mais pas démontré, que cette famille d'algorithmes quantiques permet une accélération exponentielle du temps de calcul.

3.1 Le problème de Deutsch-Jozsa

L'algorithme de Deutsch et Jozsa est probablement l'algorithme quantique le plus simple. Dans sa version initiale il fut inventé par David Deutsch dans un article fondateur[1] en 1985. L'algorithme fut ensuite amélioré par Deutsch et Jozsa (1992) et finalement par Cleve-Ekert-Macchiavello-Mosca (1998). Cette version finale fait clairement apparaître que l'algorithme est le prototype d'une classe plus vaste, étudiée dans les chapitre ultérieurs, basée sur la *transformée de Fourier quantique* et le *principe d'interférence des chemins quantiques*. De plus, il constitue une très bonne illustration d'un cas où l'on peut tirer parti du *parallélisme quantique*. Ces 3 notions seront discutées au fil de ce chapitre et des suivants.

Formulons d'abord le problème à résoudre.

$$f : \mathbb{F}_2^n \to \mathbb{F}_2, \qquad (x_1 \dots x_n) \mapsto f(x_1 \dots x_n) \in \{0, 1\}$$

une fonction booléenne dont on sait a priori qu'elle est *constante* ou *balancée*. La fonction est constante si l'image prend toujours la même valeur quelle que

[1] *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer* Proc. Roy. Soc of London A **400** pp 97-117 (1985)

Figure 3.1: Oracle classique retournant les valeurs de la fonction f.

soit l'arguement $(x_1 \dots x_n)$. Notez qu'il y a 2^n arguments possibles. *Balancée* signifie que pour une moitié des arguments (c.a.d 2^{n-1}) elle prend la valeur 0 et pour l'autre moitié (c.a.d 2^{n-1}) elle prend la valeur 1.

Le problème de Deutsch-Jozsa est un *problème de décision avec oracle.* Cela veut dire que l'on ne connaît pas la fonction f mais que l'on a à disposition un *Oracle* qui donne la réponse $f(x_1 \dots x_n)$ pour toute entrée $x_1 \dots x_n$ qui lui est soumise. Le problème est de décider si f est constante. Le nombre de questions nécessaires à l'oracle détermine la complexité temporelle de la résolution. Le but est de prendre la décision correcte en posant le moins de questions possibles.

Discutons d'abord la solution classique. Si on se limite à utiliser un algorithme déterministe, il existe des fonctions f pour lesquelles la complexité temporelle est de $2^{n-1}+1$, c'est-à-dire exponentielle par rapport à la taille des entrées. En effet supposons que f soit constante et prenne la valeur 0 si bien que l'oracle retourne toujours la réponse 0. Si l'on pose strictement moins de $2^{n-1}+1$ questions à l'oracle on n'a aucun moyen de savoir si la prochaine réponse sera aussi 0. Par contre si à la $2^{n-1}+1$ ième question la réponse est 0 alors on peut affirmer avec certitude que la fonction n'est pas balancée car si elle l'était cette réponse aurait été 1.

Notons que si la fonction est balancée le nombre minimum de questions à poser est deux et le maximum est $2^{n-1}+1$. Le point important est qu'il existe des situations défavorables qui nécessitent un nombre exponentiel de questions.

Si l'on utilise un algorithme aléatoire, la complexité du problème de Deutsch-Jozsa est bornée. Par conséquent pour ce problème il est plus raisonnable d'utiliser un algorithme aléatoire plutôt que déterministe. Précisons ce que nous entendons par algorithme aléatoire.

Nous sommes en présence d'un problème de décision : f est constante ou balancée. Soit $\epsilon < \frac{1}{2}$. L'algorithme aléatoire doit satisfaire

- si f est constante : l'algorithme répond "constante" avec probabilité $\geq 1-\epsilon$;
- si f est balancée : l'algorithme répond "constante" avec probabilité $\leq \epsilon$.

C'est-à-dire que pour toute fonction f probabilité d'erreur (mauvaise décision) est $\leq \epsilon < \frac{1}{2}$. En répétant l'expérience plusieurs fois et en appliquant

la règle de la majorité, on peut rendre cette probabilité d'erreur arbitrairement proche de 0. En effet si l'on répète l'expérience $2R$ fois la probabilité d'obtenir un nombre de réponses fausses plus grand que R est

$$\begin{aligned}\sum_{l=R+1}^{2R}\binom{2R}{l}\epsilon^l(1-\epsilon)^{2R-l} &\leq \sum_{l=R+1}^{2R} 2^{2R}\epsilon^l(1-\epsilon)^{2R-l}\\ &\leq 2^{2R}(1-\epsilon)^{2R}\big(\frac{\epsilon}{1-\epsilon}\big)^R\sum_{l=1}^{2R}\big(\frac{\epsilon}{1-\epsilon}\big)^l\\ &\leq (R+1)(2\sqrt{\epsilon(1-\epsilon)})^{2R}\end{aligned}$$

Ici on a utilisé $\binom{2R}{l}\leq 2^{2R}$, puis $\frac{\epsilon}{1-\epsilon}<1$ si $\epsilon<\frac{1}{2}$. La probabilité d'erreur peut être rendu aussi petite que l'on veut en augmentant R car pour $\epsilon<\frac{1}{2}$ on a $2\sqrt{\epsilon(1-\epsilon)}<1$. Notons que cette méthode d'amplification de la probabilité de succès basée sur la règle de la majorité est très générale et ne dépend pas du détail de l'algorithme. La seule condition est d'avoir une probabilité de succès $1-\epsilon>\frac{1}{2}$ lors de chaque expérience. La complexité totale est égale à $2R\times$ (complexite d′une experience).

Décrivons un algorithme aléatoire très simple. On soumet successivement à l'oracle K entrées aléatoires (tirées au hasard avec remise). Si pour toutes les K questions l'oracle donne les valeurs $(0,0,\ldots,0)$ ou $(1,1,\ldots,1)$ on déclare que f est "constante" (sinon on déclare "balancée"). Est ce que cet algorithme satisfait au critère ci-dessus ? Supposons que f soit constante: l'algorithme donnera la réponse "constante" avec probabilité 1 (donc $\geq 1-\epsilon$). Supposons que f soit balancée: la probabilité d'obtenir d'obtenir la suite $(\underbrace{0,0,\ldots,0}_{K\text{ fois}})$ est $\big(\frac{1}{2}\big)^K$ et celle d'obtenir la suite $(\underbrace{1,1,\ldots,1}_{K\text{ fois}})$ est $\big(\frac{1}{2}\big)^K$. Dans ce cas l'algorithme répondra "constante" avec probabilité[2] $\leq\frac{2}{2^K}<\epsilon<<\frac{1}{2}$ pour K est assez grand. La probabilité d'erreur peut être rendue arbitrairement proche de zéro en prenant K grand. Autrement si on veut se conformer à la définition ci-dessus on peut prendre $K=3$ et répéter l'expérience $2R$ fois. Le point important est que la complexité est bornée par $6R$. (ici on ne tient pas compte de la complexité sous-jacente de l'oracle[3]). En particulier, en prenant $R=O(n)$ on obtient une probabilité de succès de $1-O(\frac{1}{4^n})$ avec une complexité $O(n)$.

Nous allons maintenant voir qu'il existe un algorithme quantique qui permet de déterminer si f est balancée ou constante avec une et une seule utilisation de l'oracle et ceci quel que soit la fonction f. D'une certaine manière

[2]on utilise $P(A\cup B)\leq P(A)+P(B)$

[3]c'est pour cela qu'on appelle cela un oracle

cela est assez spectaculaire. Par rapport à un algorithme classique déterministe, le gain est exponentiel. Néanmoins par rapport à l'algorithme classique aléatoire du point de vue qualificatif il n'y a pas de gain car dans les deux cas la complexité est bornée. Pour l'algorithme quantique de DJ la probabilité de succès est exactement 1 (comme nous le verrons) alors que dans le cas classique aléatoire elle est seulement proche de 1, et on pourrait penser que l'on gagne quelque chose grâce à l'algorithme quantique. Néanmoins en pratique cet argument est illusoire car en pratique une machine quantique serait forcément légèrement imparfaite et ne fournirait pas une probabilité de succès strictment égale à 1.

Dans les paragraphes suivant nous donnons l'algorithme. Nous nous attellerons à son interprétation physique à la fin du chapitre.

3.2 Rappel sur les portes logiques

Pour chaque n on construit un circuit qui constitue l'algorithme. Les constituants du circuit sont la porte quantique de Hadamard et un *Oracle quantique*.

Nous rappelons que la porte de Hadamard n'est rien d'autre que la matrice unitaire

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

qui agit sur les états de la base computationelle (base canonique) $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ et $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ comme suit:

$$H \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$

$$H \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$

En notation de Dirac, les deux états canoniques d'un qubit sont $\begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$ et $\begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$, soit

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

et

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

Figure 3.2: Porte de Hadamard

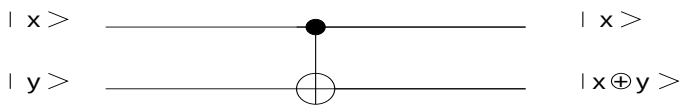

Figure 3.3: Porte Control-NOT

$$H|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) = |-\rangle$$

Le circuit correspondant possède un bit quantique d'entrée et un bit de sortie (figure 3.2: Notons que $H|+\rangle = H^2|0\rangle = |0>$ et $H|-\rangle = H^2|1\rangle = |1\rangle$. Rappelons que la porte de Hadamard peut être réalisée physiquement par la résonance magnétique nucléaire qui agit sur les spins des noyaux atomiques ou par des éléments optiques qui agissent sur les degrés de liberté des photons.

L'oracle quantique est une porte donnée (par la Nature par exemple; c'est-à-dire que cela pourrait être un système physique) qui effectue l'opération unitaire suivante :

$$U_f|x_1 \dots x_m, y\rangle = |x_1 \dots x_m, y \oplus f(x_1 \dots x_m)\rangle$$

Cet opérateur unitaire agit sur un Ket de Dirac à plusieurs qubits appartenant à l'espace

$$\underbrace{\mathbb{C}_2 \otimes \mathbb{C}_2 \dots \mathbb{C}_2}_{n \text{ fois}} \otimes \mathbb{C}_2$$

et donne un autre ket appartenant au même espace. C'est donc une matrice unitaire de dimensions $2^{n+1} \times 2^{n+1}$. Il s'agit en fait d'une généralisation du Control-NOT

$$CNOT|x, y\rangle = |x, y \oplus x\rangle$$

Pour ce dernier $n = 1$ et $f =$ identité (voir circuit sur figure 3.3). On peut si l'on veut, dire que l'Oracle agit comme une porte *multicontrôle* (c.a.d qu'il y a plusieurs qubits de contrôle). Son circuit quantique est représenté sur la figure 3.4 Vérifions que l'on a bien à faire a une matrice unitaire. Pour cela il suffit de montrer qu'elle préserve le produit scalaire. Prenons deux vecteurs

$$U_f|x_1 \dots x_m, y\rangle \text{ et } U_f|x'_1 \dots x'_m, y'\rangle$$

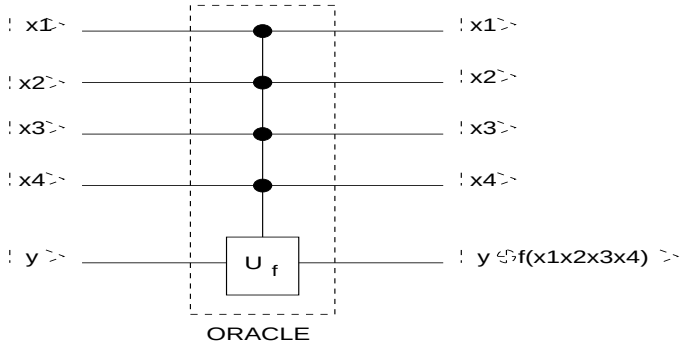

Figure 3.4: L'Oracle quantique retourne le résultat de f stocké dans les bits auxiliaires

et effectuons les produits scalaires :

$$\begin{aligned}\langle x'_1 \dots x'_m, y | U_f^\dagger U_f | x_1 \dots x_m, y\rangle &= \langle x'_1 \dots x'_m, y' \oplus f(x'_1 \dots x'_m) | x_1 \dots x_m, y \oplus f(x_1 \dots x_m)\rangle \\ &= \langle x'_1 | x_1\rangle \dots \langle x'_m | x_m\rangle \langle y' + f(x'_1 \dots x'_m) | y + f(x_1 \dots x_m)\rangle \\ &= \delta_{x_1 x'_1} \dots \delta_{x_m x'_m} \langle y' + f(x'_1 \dots x'_m) | y + f(x_1 \dots x_m)\rangle \\ &= \delta_{x_1 x'_1} \dots \delta_{x_m x'_m} \delta_{y' y}\end{aligned}$$

D'autre part

$$\begin{aligned}\langle x'_1 \dots x'_m, y' | x_1 \dots x_m, y\rangle &= \langle x'_1 | x_1\rangle \dots \langle x'_m | x_m\rangle \langle y' | y\rangle \\ &= \delta_{x'_1 x_1} \dots \delta_{x'_m x_m} \delta_{y' y}\end{aligned}$$

3.3 Algorithme quantique de Deutsch-Jozsa

Pour chaque n on construit le circuit de la figure 3.5 L'algorithme est initialisé dans l'état (instant t_0).

$$\underbrace{|0\rangle \otimes \dots \otimes |0\rangle}_{n \text{ fois}} \otimes |0\rangle = |\Psi_{in}\rangle$$

et se termine par une mesure dans la base computationnelle des n premiers qubits. Les projecteurs utilisés dans la mesure sont

$$P(b_1 \dots b_n) = |b_1 \dots b_n\rangle\langle b_1 \dots b_n|$$

Nous allons analyser l'évolution temporelle effectuée par un circuit aux instants t_{in}, t_2, t_3, t_{fin}. L'état final est donné par

$$|\Psi_{\text{fin}}\rangle = U(t_{\text{fin}}, t_{\text{in}})|\Psi_{in}\rangle = U(t_{\text{fin}}, t_4)U(t_4, t_3)U(t_3, t_2)U(t_2, t_{\text{in}})|\Psi_{in}\rangle$$

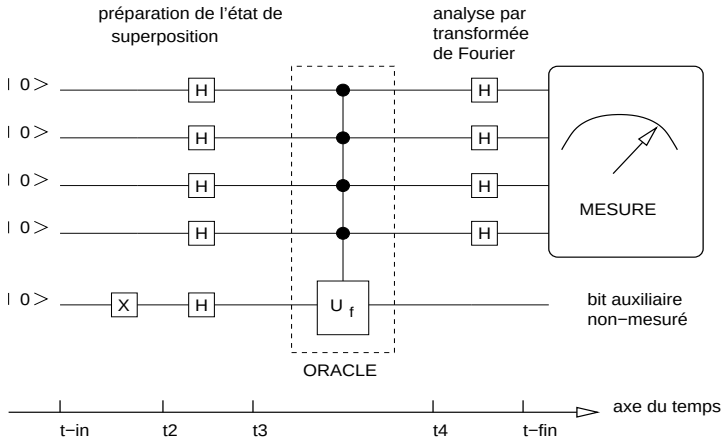

Figure 3.5: Circuit de l'algorithme de Deutsch-Jozsa

Les opérations d'évolution de chaque tranche sont

$$U(t_2, t_{\text{in}}) = (\underbrace{Id \otimes Id \otimes \cdots \otimes Id}_{n \text{ fois}}) \otimes X$$
$$U(t_3, t_2) = (\underbrace{H \otimes H \otimes \cdots \otimes H}_{n \text{ fois}}) \otimes H$$
$$U(t_4, t_3) = U_f$$
$$U(t_{\text{fin}}, t_2) = (\underbrace{H \otimes H \otimes \cdots \otimes H}_{n \text{ fois}}) \otimes Id$$

État à l'instant t_3.

$$\underbrace{(H \otimes H \otimes \cdots \otimes H)}_{n \text{ fois}} \otimes HX(\underbrace{|0\rangle \otimes \cdots \otimes |0\rangle}_{n \text{ fois}}) \otimes |0\rangle$$
$$= (\underbrace{H|0\rangle \otimes H|0\rangle \otimes \cdots \otimes H|0\rangle}_{n \text{ fois}}) \otimes HX|0\rangle$$
$$= \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \cdots \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \otimes H|1\rangle$$
$$= \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \cdots \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$
$$= \left(\frac{1}{2^{\frac{n}{2}}} \sum_{b_1 \ldots b_n} |b_1 \ldots b_n\rangle\right) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

A cet instant l'entrée est une *superposition cohérente de toutes les entrées classiques possibles*. Le dernier bit $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ est un bit auxiliaire qui va servir à stocker le résultat de l'oracle.

État à l'instant t_4.

$$\begin{aligned} U_f \frac{1}{2^{\frac{n}{2}}} \sum_{b_1 \dots b_n} |b_1 \dots b_n\rangle \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - |1\rangle \right) \\ = \frac{1}{2^{\frac{n}{2}}} \sum_{b_1 \dots b_n} \left(\frac{1}{\sqrt{2}} U_f|b_1 \dots b_n, 0\rangle - \frac{1}{\sqrt{2}} U_f|b_1 \dots b_n, 1\rangle \right) \\ = \frac{1}{2^{\frac{n}{2}}} \sum_{b_1 \dots b_n} \left(\frac{1}{\sqrt{2}}|b_1 \dots b_n, f(b_1 \dots b_n)\rangle - \frac{1}{\sqrt{2}}|b_1 \dots b_n, \overline{f(b_1 \dots b_n)}\rangle \right) \end{aligned}$$

Notons que si $f(b_1 \dots b_n) = 0$ le terme entre parenthèses vaut

$$|b_1 \dots b_n\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

et que si $f(b_1 \dots b_n) = 1$ le terme entre parenthèses vaut

$$|b_1 \dots b_n\rangle \otimes \frac{|1\rangle - |0\rangle}{\sqrt{2}}$$

On peut donc écrire létat à l'instant t_4 comme suit:

$$\left(\frac{1}{2^{\frac{n}{2}}} \sum_{b_1 \dots b_n} (-1)^{f(b_1 \dots b_n)} |b_1 \dots b_n\rangle \right) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Cet état est à nouveau une superposition cohérente ou l'oracle a déphasé chaque entrée classique de 0 à π suivant que[4] l'image de f est 0 ou 1.

État à l'instant t_{fin}. On applique finalement l'opérateur unitaire $\underbrace{H \otimes H \otimes \cdots \otimes H}_{n \text{ fois}} \otimes Id$, ce qui donne par linéarité :

$$|\Psi_{\text{fin}}\rangle = \left(2^{\frac{n}{2}} \sum_{b_1 \dots b_n} (-1)^{f(b_1 \dots b_n)} H|b_1\rangle \otimes \cdots \otimes H|b_n\rangle \right) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Notons que

$$H|b\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + (-1)^b |1\rangle \right) = \frac{1}{\sqrt{2}} \sum_c (-1)^{cb} |c\rangle$$

[4] Car $e^{\mathrm{i}0} = 1$ et $e^{\mathrm{i}\pi} = -1$.

si bien que

$$\begin{aligned} H|b_1\rangle \otimes \cdots \otimes H|b_n\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{b_1}|1\rangle\right) \otimes \cdots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{b_n}|1\rangle\right) \\ &= \frac{1}{\sqrt{2}} \sum_{c_1} (-1)^{c_1 b_1} |c_1\rangle \otimes \cdots \otimes \sum_{c_2} (-1)^{c_2 b_2} |c_2\rangle \\ &= \frac{1}{2^{\frac{n}{2}}} \sum_{c_1 \ldots c_n} (-1)^{\sum_{i=1}^{n} b_i c_i} |c_1 \ldots c_n\rangle \end{aligned}$$

Ainsi

$$|\Psi_{\text{fin}}\rangle = \sum_{c_1 \ldots c_n} \left\{ \frac{1}{2^n} \sum_{b_1 \ldots b_n} (-1)^{f(b_1 \ldots b_n)} (-1)^{\sum_{i=1}^{n} b_1 c_i} \right\} |c_1 \ldots c_n\rangle \otimes \frac{1}{\sqrt{2}} \left(|0\rangle - |1\rangle\right)$$

L'état final est à nouveau une superposition cohérente d'états classiques affectés d'amplitudes

$$\frac{1}{2^n} \sum_{b_1 \ldots b_n} (-1)^{f(b_1 \ldots b_n)} (-1)^{\sum_{i=1}^{n} b_i c_i}$$

Les amplitudes contiennent de l'information sur la fonction f. Si nous les connaissions toutes nous pourrions en fait déterminer cette fonction. Malheureusement la seule chose qui est à notre disposition est la totalité de l'état $|\Psi_{\text{fin}}\rangle$ (pris dans sa globalité) et *la seule chose que nous puissions faire, pour extraire de l'information, est une mesure.*

3.3.1 Dernière étape de l'algorithme: la mesure

Appliquons le postulat de la mesure. Si nous mesurons l'état des n premiers qubits dans la base computationnelle, l'état est projeté (ou réduit) sur *un* des états $|c_1 \ldots c_n\rangle$ avec probabilité

$$\begin{aligned} Prob(c_1 \ldots c_n) &= [\text{carré de l'amplitude devant } |c_1 \ldots c_n\rangle] \\ &= \frac{1}{2^{2n}} \left| \sum_{b_1 \ldots b_n} (-1)^{f(b_1 \ldots b_n)} (-1)^{\sum_{i=1}^{n} c_i b_i} \right|^2 \end{aligned}$$

La signification de cette assertion est la suivante : tant que la mesure est faite une fois sur un unique état $|\Psi_{\text{fin}}\rangle$ l'état final observé est un des états $|c_1 \ldots c_n\rangle$ et il n'y a aucun moyen de prédire lequel c'est (Einstein n'était pas d'accord avec le fait qu'il n'y a aucun moyen de prédire l'état observé $|c_1 \ldots c_n\rangle$ et il résuma son point de vue par la phrase "Dieu ne joue pas aux

dés"). Si l'on dispose d'un ensemble d'états $|\Psi_{\text{fin}}\rangle$, en répétant l'expérience est répétée plusieurs fois la fréquence des observations $|c_1 \dots c_n\rangle$ est donnée par $Prob(c_1 \dots c_n)$.

Calculons cette probabilité. Si f est constante on trouve

$$\begin{aligned} Prob(c_1 \dots c_n) &= \frac{1}{2^{2n}} \left| \sum_{b_1 \dots b_n} (-1)^{\sum_{i=1}^n c_i b_i} \right|^2 \\ &= \frac{1}{2^{2n}} \left| \sum_{b_1 \dots b_n} \prod_{i=1}^n (-1)^{c_i b_i} \right|^2 \\ &= \frac{1}{2^{2n}} \left| \prod_{i=1}^n \left((-1)^{c_i 0} + (-1)^{c_i 1} \right) \right|^2 \\ &= \begin{cases} 1 & \text{si } (c_1 \dots c_n) = (0 \dots 0) \\ 0 & \text{dans tous les autres cas} \end{cases} \end{aligned}$$

Donc si f est constante nous observerons certainement $(0 \dots 0)$ (c.a.d avec probabilité 1) en faisant une seule expérience ! Par contre si f est balancée on constate que

$$Prob(0 \dots 0) = \frac{1}{2^{2n}} \left| \sum_{b_1 \dots b_n} (-1)^{f(b_1 \dots b_n)} \right|^2 = 0$$

et on n'observera certainement pas $(0 \dots 0)$.

En conclusion : après le processus de mesure si $(0 \dots 0)$ est observé on peut conclure "f constante" et si autre chose est observé on peut conclure "f balancée". Remarquablement il suffit de faire l'expérience et d'utiliser l'oracle quantique qu'une seule fois !

Note sur la complexité de l'algorithme. En general nous devons distinguer la complexité du circuit et celle de l'algorithme proprement dit. Dans ce cours la complexité des circuits sera mesurée en terme de deux grandeurs, la "profondeur" et la "largeur". La taille du circuit est alors définie comme le produit $(profondeur) \times (largeur)$. Ici le circuit contient 3 tranches de temps intermediaires: on dit que ce circuit à une profondeur égale à 3 (ce qui est important ici c'est qu'elle est $O(1)$). Si le temps élémentaire requis pour effectuer une porte quantique (par RMN par ex) est de τ alors le temps de calcul du circuit est de 3τ. En effet les calculs effectués dans une tranche temporelle sont parallèles. C'est ce qu'on appelle le parallélisme quantique. La largeur du circuit est donnée par le nombre de bits d'entrée plus le nombre de bits auxiliaires, ici $n+1$, et représente le nombre de calculs que le circuit

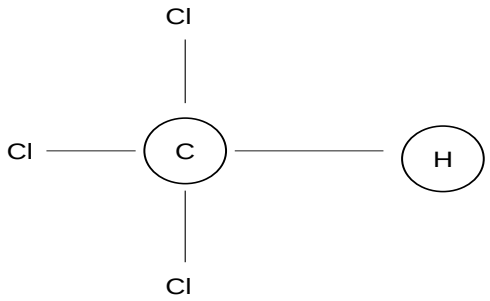

Figure 3.6: L'algorithme DJ a été réalisé par RMN sur les molécules de $CHCl_3$. On agit sur deux qubits associés aux spins nucléaires des atomes H et C entourés

quantique effectue en parallèle à chaque tranche temporelle. La taille du circuit de DJ est donc $3(n+1) = O(n)$. Finalement quelle est la complexité de l'algorithme lui même ? Puisqu'on peut résoudre le problème de DJ en posant une seule question à l'oracle, l'algorithme possède un temps de calcul $O(1)$ avec un circuit de taille $O(n)$.

3.4 Réalizations expérimentales

L'algorithme de DJ fut un des premiers algorithmes quantique a être réalisé experimentalement (2001) pour le cas $n = 1$ par la résonance magnétique nucléaire. Cette experience utilise un liquide de $CHCl_3$ (le chloroforme, fig. 3.6). Pour $n = 1$ il faut deux bits quantiques, un pour l'entrée et un bit auxiliare pour stocker le résultat de l'oracle. Ceux ci sont matérialisés par les spins nucléaires de l'atome d'hydrogène H et de carbone C. Les portes de Hadamard et l'Oracle peuvent être réalisés en manipulant ces deux spins nucléaires par des champs magnétiques de radiofréquence. Un des problèmes principaux est de préparer toutes les molécules du liquide dans l'état initial $|00\rangle$. En effet on ne peut pas se débarasser complètement des fluctuations thermiques qui induisent des transitions vers les autres états pour une fraction des molécules du liquide. Pour augmenter n il faut prendre de plus grosses molécules avec des atomes appropriés. Ceci pose un problème ("scalability problem") parceque plus la molecule est grosse plus les niveaux d'énergie sont nombreux et rapprochés (le spectre devient continu en quelque sorte) et il devient de plus en plus difficile de manipuler sélectivement les bits quantiques grâce à des transitions de radiofréquence. Plus récemment, l'agorithme de DJ a aussi été réalisé grâce aux technologies des pièges à ions et des cavités resonantes (cavity QED). Toutes ces experiences sont limitées à un faible

nombre de qubits ($<$ 10 qubits).