Introduction à l'informatique quantique

Philippe DENIEL (philippe.deniel@cea.fr)

CEA

2024

Plan

Quelques citations

Niels Bohr - Si la mécanique quantique ne vous a pas encore profondément choqué, alors vous ne l'avez pas encore comprise. Tout ce que nous appelons réel est fait de choses qui ne peuvent pas être considérées comme étant réelles.

Niels Bohr - Si une idée ne semble pas bizarre, il n'y a rien à espérer d'elle.

Heinz Pagels - Dieu a utilisé de merveilleuses mathématiques pour créer le monde.

Richard Feynmann - Je pense pouvoir dire sans trop me tromper que personne ne comprend la mécanique quantique.

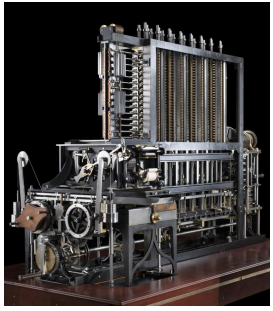
Richard Feynmann, 1982 - Nature isn't classical, dammit, and if you want to make a simulation of nature, you better make it quantum mechanical

Albert Einstein - If you can't explain it simply, you don't understand it well enough

Qu'est-ce qu'un ordinateur ?

D'après le Dictionnaire Larousse,

> *Machine automatique de traitement de l'information, obéissant à des programmes formés par des suites d'opérations arithmétiques et logiques.*



Plusieurs déclinaisons ont été faites autour de ce concept :

- la *machine à différences* de Lord Babbage, basée sur des composants mécaniques,
- les ordinateurs basés sur l'électronique, qui s'appuient sur la théorie physique de l'électromagnétisme et donc sur les équations de Maxwell

L'ordinateur quantique,

- une évolution logique du concept, mais s'appuyant sur les phénomènes décrits par la physique quantique.

L'informatique quantique n'est pas si récente

Le *Quentum Computing* est né dans les années 80

- Une première conférence sur le QC au MIT en 1981
- Beaucoup d'études théoriques sur le QC
  - Algorithme de Shor, 1994
  - Algorithme de Grover, 1996

L'informatique quantique a de solides bases théoriques

- dans le domaine de la physique (physique quantique, physique statistique)
- dans le domaine des mathématiques (algèbre linéaire, algèbre hermitienne)

En revanche les implémentations physiques "réelles" des QPUs sont très récents.

Les liens entre informatique quantique et physique quantique

Le *Quantum Computing* est un nouveau paradigme informatique basé sur les phénomènes à la physique quantique :

1. la superposition,
   - qui permet d'induire une forme de parallélisme (à relativiser...)
2. l'intrication,
   - qui permet de coupler des systèmes simples pour de bâtir des systèmes plus complexes
3. les interférences,
   - qui permettent de mesurer les états quantiques et d'obtenir des résultats de calcul.

La première révolution quantique

La physique quantique est déjà très présente dans notre monde

- imagerie médicale : IRM (imagerie par résonance magnétique,
- composants électroniques : transistors à effet tunnel, LED,
- lasers,
- écrans LCD,
- panneaux solaires photovoltaïques : interaction photon/matière.

Le QC et la deuxième révolution quantique

L'informatique quantique fait partie de la deuxième révolution quantique

Les **QPU** commencent à apparaître sur le marché.

On prévoit plusieurs phases dans le QC

- NISQ : **N**oisy **I**ntermediate **S**cale **Q**uantum
- FTQC : **F**ault **T**olerant **Q**uantum **Q**uantum
- LSQ : **L**arge **S**cale **Q**uantum

On est actuellement dans la période NISQ

8/117

Ce que l'informatique quantique n'est pas.

Le QC ne va pas remplacer le HPC

- Les QPUs appartiennent à un nouveau paradigme du HPC, comme les GPUs.
- le QC est une nouvelle arme dans l'arsenal du HPC

Attention à la "hype" autour du QC, entretenue par certains constructeurs (dont IBM et Google)

- Des annonces "publicitaires", parfois éloignées de la réalité scientifique
- Des notions "grand public" aux contours souvent très flous...
- le QC ce n'est pas "faire 2^n calculs en même temps, le parallélisme exponentiel existe mais il est loin d'être systématique
- le QC est efficace pour caractériser rapidement une propriété globale d'un problème
  - périodicité d'une fonction entière (Shor)
  - parcours de graphe (marches quantiques)
  - Recherches d'optima de forme quadratiques (QUBO)

9/117

Plan

L'équation de Schrödinger

La physique quantique décrit des phénomènes qui se produisent à l'échelle des atomes.

Elle est décrite par l'équation de Schrödinger

$$i\hbar\frac{\partial}{\partial t}\Psi(x,t) = -\frac{\hbar^2}{2m}\nabla^2\Psi(x,t) + V(x,t)\Psi(x,t).$$

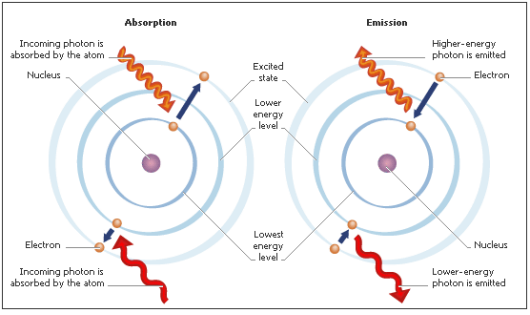
On note que

- c'est une équation à valeurs dans $\mathbb{C}$, on remarque en particulier le i des imaginaires purs dans la partie gauche.
- on reconnaît l'opérateur ∇ de la géométrie différentielle, utilisé dans les équations de Maxwell (électromagnétisme) et dans l'équation de Navier-Stokes (hydrodynamique)
- $\hbar = \frac{h}{2\pi}$ où h est la constante de Planck ($h = 6,62607015.10^{-34} J.s$)
- les actions sur des objets quantiques sont des opérateurs unitaires dans un espace de Hilbert

Conséquences de l'équation de Schrödinger

Plusieurs choses découlent directement de l'équation de Schrödinger

- Les phénomènes quantiques adoptent des valeurs discrètes ou *quanta*
- Les objets de la physique quantique ne passent pas continument d'un état à un autre
- Les objets quantiques peuvent être dans plusieurs états en même temps (superposition)
- les objets quantiques sont à la fois des ondes et des particules (dualité ondes-particules)



La superposition

Les objets quantiques sont des compositions linéaires d'états de base qui forment une base orthonormée, par exemple

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle + \gamma\,|2\rangle + \delta\,|3\rangle$$

Les coefficients complexes α, β, γ et δ sont des *probabilités complexes*

- Ils respectent la condition de normalisation $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$
- le carré du module de chaque coefficient représente la probabilité de mesurer l'état concerné (ainsi on a une probabilité $|\alpha|^2$ de voir l'état $|0\rangle$ quand on observe $|\psi\rangle$

Observation et effondrement quantique

En physique quantique, observe est une opération destructive, elle altère l'état que l'on regarde.

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle + \delta|3\rangle$$

Observer $|\Psi\rangle$ va provoquer **l'effondrement** de l'état $|\Psi\rangle$ sur l'un des quatre états de base possibles

- On va voir uniquement l'un de ses états, et pas les autres
- une fois que $|\Psi\rangle$ est effondré, il ne reviendra jamais dans l'état précédent

Dans l'informatique quantique, on construit une "expérience de physique quantique instrumentée", on peut refaire l'expérience qui construit $|\Psi\rangle$ autant que nécessaire

- en faisant plusieurs "tirs", on connaît probabilité de chaque état de base
- **ATTENTION :** on connaît les carrés des modules des coefficients complexes (on mesure $|\alpha|^2$ et pas α)
- on ne voit pas toute l'information qui est "portée" par l'état $|\Psi\rangle$
- en particulier les "facteurs de phase" (de la forme $e^{i\theta}$) sont invisibles car unitaires.

Intrication

L'intrication est un concept fondamental et profondément contre-intuitif.

Einstein n'y croyait pas, cela a donné lieu au célèbre paradoxe EPR (du nom de Einstein-Podolsky-Rosen).

L'expérience de Alain Aspect en 1982, qui lui a valu le Prix Nobel de Physique en 2022, prouve que l'intrication est réelle (et que Albert Einstein avait tort sur ce point).

Un état intriqué regroupe plusieurs particules qui doivent être considérées comme un seul objet quantique

- si deux particules A et B sont intriquées, tout ce qui agit avec A agit aussi sur B, car cela affecte le système $A \otimes B$
- on ne peut pas isoler les particules A et B, elles doivent être considérées ensemble et plus indépendamment
- l'intrication n'est pas une interaction, elle n'est pas du tout affectée par la distance
- en particulier mesurer A provoque son effondrement quantique et provoque aussi l'effondrement de B

Les bases de l'informatique quantique

L'informatique quantique se base sur les trois principes de la physique quantique

- encoder de l'information dans des états quantiques grâce à la **superposition d'états** dans des *qubits*
- construire des systèmes complexes de plusieurs qubits grâce à l'**intrication**
- mesurer et obtenir des résultats en interférant avec le système obtenu, en **observant** l'état final ce qui provoque un effondrement de cet état.

De solides fondations mathématiques

La physique quantique s'appuie sur un socle mathématique très solide :

- les qubits sont des éléments d'un $\mathbb{C}$-espace vectoriel de dimension 2, qui est un espace de Hilbert
- un système à n qubits évolue dans un espace de Hilbert de dimension 2^n

On sait traduire mathématiquement

- l'action des opérateurs quantiques sur les qubits
- l'action des qubits les uns sur les autres
- la mesure d'un ou plusieurs qubits
- l'intrication s'exprime en particulier de manière formelle

L'informatique quantique implique souvent beaucoup d'algèbre linéaire dans des espaces de Hilbert.

Idées clefs sur cette partie

Fondamentalement, un programme quantique c'est

- une expérience de physique quantique instrumentée
- l'exploitation intelligente d'un phénomène quantique pour accélérer une phase de calcul complexe
  - Cette phase est autant que possible un problème en temps exponentiel ou un problème de type NP
  - il faudra toujours du HPC pour faire marcher efficacement le QC

Souvent un programme quantique est bon pour déterminer les caractéristiques d'une propriété globale qu'on connaît a priori

- j'ai une fonction f, périodique, quelle est la période de f ? (algorithme de Shor)
- j'ai une fonction f nulle presque partout, quels sont les x tels que $f(x) \neq 0$? (algorithme de Grover)
- recherche de minima et de maxima de formes quadratiques exprimées comme des matrices (QUBO)
- recherche de propriétés d'un graphe (MIS, MVC, Clique, MaxCut,...)

Plan

19/117

Les qubits

Un qubit est un système quantique qui possède deux états de base $|0\rangle$ et $|1\rangle$, c'est l'analogue quantique d'un bit.

Un qubit est un état superposé, écrit sous la forme

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \alpha \in \mathbb{C}, \beta \in \mathbb{C},\ \text{avec}\ |\alpha|^2 + |\beta|^2 = 1$$

Par exemple $|\Psi\rangle = 0.6\,|0\rangle + 0.8\,|1\rangle$ (et on a $0.6^2 + 0.8^2 = 0.36 + 0.64 = 1$)

Zoologie des qubits physiques

Les qubits peuvent être de différentes natures

- spin de particule (encodé sur $|UP\rangle$ et $|DOWN\rangle$]
- des états d'excitation d'un électron (atomes froids, ions piégés, cavités NV)
- le sens du courant dans une boucle supraconductrice (qubits supraconducteurs)
- la polarisation d'un photon (photonique quantique)
- l'emplacement d'une particule dans un système (qubit topologique)

Nota Bene : On pourrait également construire une informatique quantique sur des *qu-trits* à trois états de bases, voire des *4-qudits* ou des 5-qudits. L'implémentation physique st plus simple avec des qubits, mais les mathématiques sous-jacentes n'imposent aucune limitation.

Notation de Dirac

Les qubits de base, qui forment la base canonique de $\mathbb{C}^2$ sont écrits selon la notation de Dirac.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Donc si $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$$|\Psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

22 / 117

Les portes à 1 qubit

À cause de l'équation de Schrödinger, tout opérateur agissant sur les qubits est un opérateur unitaire

- est une matrice $\mathbb{C}^{2\times2}$, isomorphe à une matrice réelle de dimension 4×4, dans $\mathbb{R}^4$
- est un opérateur unitaire U, dont l'adjoint est l'inverse $U \times U^\dagger = \mathbb{I}$
- est une rotation dans $\mathbb{C}^2$ qui peut être vue comme une rotation dans $\mathbb{R}^4$

On dispose des opérateurs X, Y et Z

- duaux des opérateurs de Pauli de la physique quantique
- duaux des éléments i, j et k du corps des quaternions $\mathbb{H}$

Toute porte peut s'écrire sous la forme

$$U = a\mathbb{I} + bX + cY + dZ, a \in \mathbb{R}, b \in \mathbb{R}, c \in \mathbb{R}, d \in \mathbb{R}$$

23/117

Superposition et porte H

La porte H, ou *porte de Hadamard* permet d'introduire la superposition d'états.

On a

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \end{aligned} \tag{1}$$

H est une matrice 2×2

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

24/117

Porte X et inversion de qubits

La porte X est analogue du NOT booléen

$$X|0\rangle = |1\rangle \text{ et } X|0\rangle = |1\rangle$$

Par conséquent

$$\text{Si } |\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \text{ alors } X|\Psi\rangle = \beta|0\rangle + \alpha|1\rangle$$

X est une matrice 2×2

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Il est important de voir X comme une rotation et pas comme une inversion booléenne, on peut trouver une racine carrée de X et faire X en deux fois, ainsi

$$\sqrt{X} = \frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix} \text{ est telle que } \sqrt{X} \times \sqrt{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = X$$

Porte Z et modification de phase

La porte Z vient inverser la phase, donc changer le signe, de la composante selon $|1\rangle$

$$\text{Si } |\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \text{ alors } Z|\Psi\rangle = \alpha|0\rangle - \beta|1\rangle$$

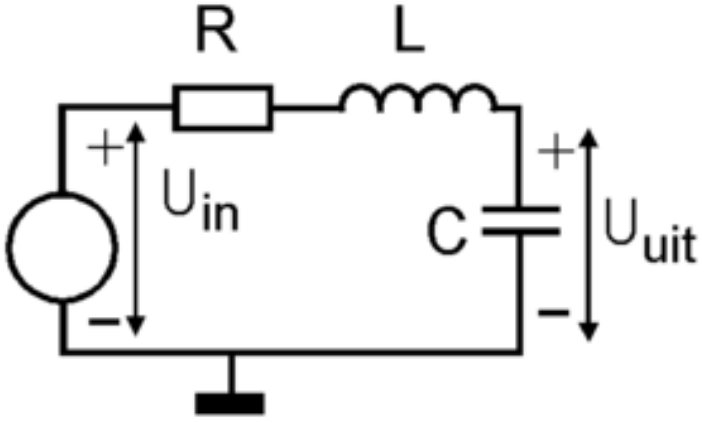
Z est une matrice 2×2

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

La porte Y se construit à partir des portes X et Z

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \text{ et } Y = iXZ$$

Ceci est un circuit (électrique)

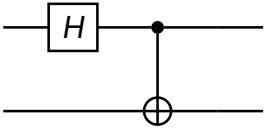


Ceci n'est pas une pipe

28/117

Ceci n'est pas un circuit, ceci est une matrice

le circuit suivant construit une *paire EPR*



On verra que ce circuit correspond à la matrice

$$\frac{1}{2}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix}$$

29/117

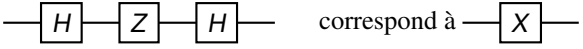
Mon premier circuit quantique

La programmation à porte construit des matrices complexes comme compositions de composants plus simples.

Revient à appliquer H, puis Z, puis H, ce circuit décrit l'action du produit matriciel $H \times Z \times H$

$$H \times Z \times H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \times \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} \times \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$
$$= \frac{1}{2}\begin{pmatrix}0 & 2\\ 2 & 0\end{pmatrix} = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} = X$$

Ce circuit est donc équivalent à une seule porte X



IMPORTANT : Les circuits quantiques ne sont qu'une manière de représenter une matrice potentiellement grande, ils n'ont rien à voir avec des circuits électroniques.

30/117

Intrication et porte CNOT 1/2

La porte CNOT, ou CX, introduit la notion d'intrication, elle introduit une notion de "if-then-else" qui permet de construire des schémas de programmation conditionnelle.

CNOT agit sur un état à 2 qubits, si le premier est $|0\rangle$, elle ne touche pas au second, mais s'il vaut $|1\rangle$ elle l'inverse en appliquant une porte X

$$CX\,|00\rangle = |00\rangle\,, CX\,|01\rangle = |01\rangle\,, CX\,|10\rangle = |11\rangle\,, CX\,|11\rangle = |10\rangle$$

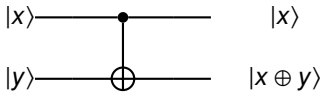
CX est une matrice 4×4

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

31/117

Intrication et porte CNOT 2/2

D'une manière générique, si x et y sont des booléens $(x, y) \in \{0, 1\}^2$, alors le premier qubit $|x\rangle$ est inchangé et $|y\rangle$ devient $|x \oplus y\rangle$ ($\oplus$ est le XOR booléen).



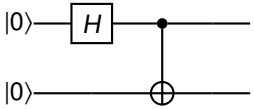
CX est une rotation de $\mathbb{C}^4$, isomorphe à une rotation dans $\mathbb{R}^8$, on peut la faire "à moitié" et calculer sa racine carrée

$$\sqrt{CX} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1+i & 1-i \\ 0 & 0 & 1-i & 1+i \end{pmatrix}$$

32 / 117

Construction de la paire EPR

La paire EPR $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ est un état intriqué simple à construire via ce circuit



Ce circuit produit bien la paire EPR en partant de $|00\rangle$, en effet

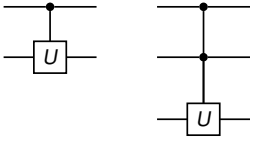
- la porte H produit l'état $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$
- on applique CX on obtient $\frac{1}{\sqrt{2}}(CX|00\rangle + CX|10\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

33/117

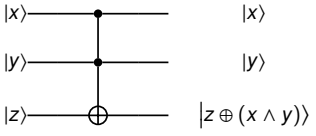
Portes contrôlées

La porte CNOT est très importante, on peut l'utiliser pour construire des portes contrôlées

- étant donné U , on l'applique sur le second qubit si le premier est dans l'état $|1\rangle$
- on peut construire des portes à double, triple, quadruple.. contrôles
- le fait qu'il existe des racines de U est critique dans la construction de ces opérateurs à contrôle multiples



En particulier la porte CCNOT, ou porte de Toffoli est très classiquement utilisée



Idées clefs sur cette partie

La programmation à porte utilise des opérateurs unitaires simples comme briques de base pour construire des opérateurs plus complexes sur n qubits

- la **porte H de hadamard** crée un état superposé à 1 qubit
- la **porte X** ou porte NOT ressemble à un NOT booléen (mais c'est une rotation)
- la **porte CNOT** utilise l'intrication quantique et est analogue à "if-then-else"

Faire un programme quantique à porte c'est

- ramener mon problème à l'exécution d'un opérateur unitaire
- construire cette opéérateur sous la forme d'un circuit
- l'exécuter en un temps plus court que le HPC, sur des QPUs.

Et surtout **toutes les actions sont des matrices unitaires**

Plan

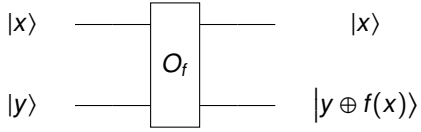
36 / 117

Les oracles

Soit f une fonction binaire : $f : \{0,1\} \to \{0,1\}$.

On intègre f dans un circuit quantique comme une "boite noire" nommé **oracle**.
L'oracle doit être linéaire et inversible

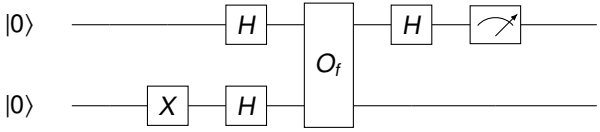
- les n premiers qubits portent l'input et ne sont pas modifiés
- les autres qubit, ou **acillae**, sont XOR-é avec le résultat de f



Le schéma est le même pour une fonction $f : \{0,1\}^m \to \{0,1\}^n$, $m, n \in \mathbb{N}^*$
On aura un oracle O_f sur $m+n$ qubits : m qubits de données et n ancillae pour les résultats

Algorithme de Deustch-Josza

Soit f une fonction binaire : $f : \{0,1\} \to \{0,1\}$, qui est soit constante, soit uniforme.
En informatique classique, il faut faire deux évaluations de f, le QC n'invoque O_f qu'une fois.



L'état final de ce circuit est

$$|\Psi_{final}\rangle \frac{|0\rangle + (-1)^{f(0)+f(1)}|1\rangle}{\sqrt{2}} \otimes |-\rangle$$

- Si f est constante, $f(0) = f(1)$ alors $|\phi_2\rangle = \frac{(|0\rangle+|1\rangle)(|0\rangle-|1\rangle)}{2} = |+\rangle|-\rangle$
- Si f est uniforme, $f(0) \neq f(1)$ alors $|\phi_2\rangle = \frac{(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)}{2} = |-\rangle|-\rangle$

En mesurant le premier qubit, on voit $|+\rangle$ ou $|1\rangle$ selon la nature de f.

Soit un chapeau contenant deux boules, soit deux noires, soit une noire et une blanche, Deutsch-Josza permet de décider en un coup. Il est généralisable à une fonction binaire sur n bits.

38/117

Algorithme de Bernstein-Vazirini

Étant donné une chaine s de n bits, on peut construire une fonction f basé sur le produit pointé :

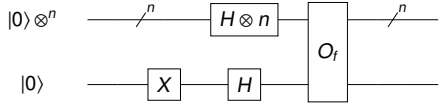
$$f : \{0,1\}^n \to \{0,1\}$$
$$x \mapsto s \cdot x = s_1 x_1 \oplus s_2 x_2 \oplus \cdots \oplus s_n x_n = s_1 x_1 + \cdots + s_n x_n \text{ modulo } 2$$

On connaît f, comment déterminer s ?

Avec une approche classique, il faut calculer les 2^n valeurs et résoudre un système linéaire (simple) pour trouver s.

39/117

Circuit de Bernstein-Vazirini



va conduire à l'état

$$|\psi\rangle = \frac{1}{2^n} \sum_{y=0}^{2^{n-1}} \left[\sum_{x=0}^{2^n-1} (-1)^{f(x)+x\cdot y} \right] |y\rangle$$

Chaque état $|y\rangle$ aura une amplitude complexe $\alpha_y = \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)+x\cdot y}$
Quand $y = s$, $\alpha_s = f(x) + x.s = x.s + x.s = 2x.s$ donc $\alpha_s = 2^n/2^n = 1$
Comme la somme des carrés des modules des α_y vaut 1, les autres coefficients sont nuls, donc $|\psi\rangle = |s\rangle$
On peut connaître s en **une seule** invocation de l'oracle.

Algorithme Shor - Idée de base

Soit N un entier (très grand) qui est le produit de 2 facteurs, quels sont ces facteurs ?

Soit a et r deux entiers tels que $a^r \equiv 1[N]$ donc $a^r - 1 \equiv 0[N]$

Si r est paire, alors $r = 2p$ et donc $a^{2P} - 1 = (a^p - 1)(a^p + 1) \equiv 0[N]$

Par conséquent, il existe un entier k tel que $(a^p - 1)(a^p + 1) = kN$

Il suffit de trouver le PGCD de N, $(a^p - 1)$, $(a^p + 1)$ pour identifier un facteur de N

Cette opération est simple à réaliser (algorithme d'Euclide).

Algorithme de Shor - Recherche de période

Connaissant N, pour tout entier a inférieur à $\sqrt{N}$ on construit la fonction

$$f_a : k \mapsto a^k \text{ modulo } N$$

Si il y a un entier r tel que $f_a(r) = a^k = 1$ alors $f_a(p+r) = a^p a^r = 1$ et donc f_a est r-périodique.

La partie quantique de l'algorithme de Shor consiste à

- prendre une valeur a
- identifier si f_a a une période r, si r est paire, on a gagné.

Cette partie s'appuie sur

- un cricuit quantique inspiré de l'algorithme de Simon, qui trouve la périodicité d'une fonction booléeene
- sur des transformées de Fourier discrètes (QFT) qui sont des implémentations quantique des transformées de Fourier discrète (DFT)

Rappel - Discrete Fourier Transform (DFT) - Définition

La transformée de Fourier "classique", ou "continue", associe à une fonction f la fonction

$$\hat{f} : \mathbb{R} \to \mathbb{C}, \hat{f}(x) = \int_{-\infty}^{+\infty} f(x) e^{-ixy}\, dy$$

parfois reformulée, avec une fréequence ν exprimée en hertz

$$\hat{f}(\nu) = \int_{-\infty}^{+\infty} f(t) e^{-2i\pi\nu t}\, dt$$

Si $(s_n)_{0 \leq n < N}$ est une suite de N valeurs, on définira sa transformée de Fourier discrète, $(\hat{s}_n)_{0 \leq n < N}$ comme la suite de valeurs

$$0 \leq n < N, \hat{s}_n = \sum_{k=0}^{N-1} s_k . e^{-2i\pi \frac{kn}{N}}$$

43/117

Rappel - Discrete Fourier Transform (DFT) - Matrice de Vandermonde

On peut exprimer la DFT sous forme matricielle, c'est la matrice de Vandermonde.

$$QF_n = \frac{1}{\sqrt{N}} W_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdot & 1 \\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{N-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)(N-1)} \end{pmatrix}$$

Exemple : Dans le cas où il y a 3 qubits, on aura la matrice 8×8 suivante

$$QF_8 = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^1 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{pmatrix} \text{ avec } \omega = e^{\frac{2i\pi}{8}}$$

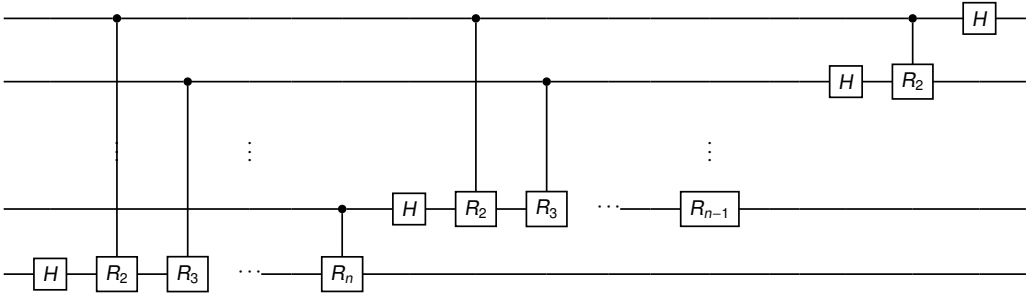
La matrice de Vandermonde est unitaire.

Circuit QFT

Le circuit suivant implémente la matrice de Vandermonde

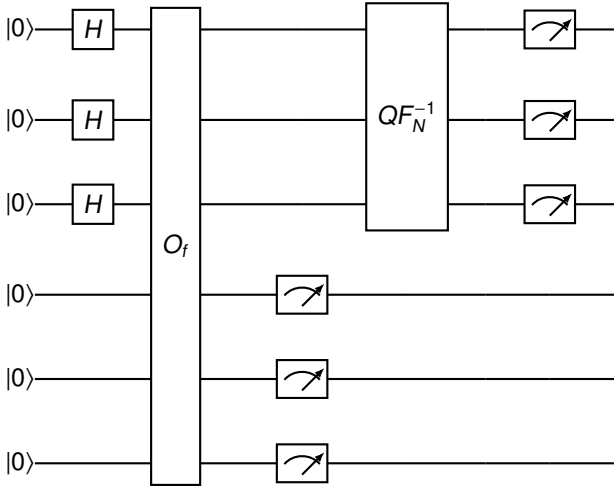
Elle s'appuie sur la formulation mathématique

$$QF_n \times |k\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} e^{-2i\pi \frac{kj}{2^n}} |j\rangle = \frac{1}{\sqrt{2^n}} \bigotimes_{j=0}^{n} (|0\rangle + e^{2i\pi \frac{k}{2^j}} |1\rangle)$$



Circuit implémentant l'algorithme de Shor

La partie quantique de l'algorithme de Shor est la suivante



Où O_f est un oracle qui implémente l'élévation à la puissance d'un entier a.

Plan

Encore des maths

La base de l'informatique quantique analogique est la théorie de la complexité

Dans les grandes lignes :

- certains problèmes "compliqués" peuvent se traduire sous la forme d'une expérience de physique quantique
- on modifie le problème que l'on veut résoudre pour le ramener à cette expérience
- on fait une "expérience instrumentée" dans un QPU analogique
- on en déduit la solution du problème initial

Parmi les implémentations analogiques, on trouve dans le paysage actuel :

- les QPU D-Wave basés sur des boucles supraconductrices
- les QPU Pasqal basés sur des atomes froids manipulés par des lasers
- les machines quantiques photoniques comme ce que propose Qandela

P et NP

En théorie de la complexité informatique, on identifie différentes classes, parmi elles

- les problèmes **de classe P** peuvent être résolu dans un temps qui varie de manière polynomiale avec la taille du problème. Les problèmes P peuvent être résolus avec du HPC
- les problèmes **de classe NP** sont *non-deterministic polynomial*, le temps de résolution est "pire" que polynomiale, mais si on me propose une solution, vérifier qu'elle résoud effectivement le problème prend un temps polynomial.

ATTENTION : NP ne **signifie pas** non-polynomial (c'est un très mauvais acronyme).

Exemples de P et de NP

Les problèmes polynomiales sont ceux du HPC classique, par exemple

- trouver le PGCD ou le PPCM de deux nombres très grands,
- inverser une matrice,
- savoir si un nombre est premier

Exemple de problème NP : factorisation d'un produit de deux nombres premiers

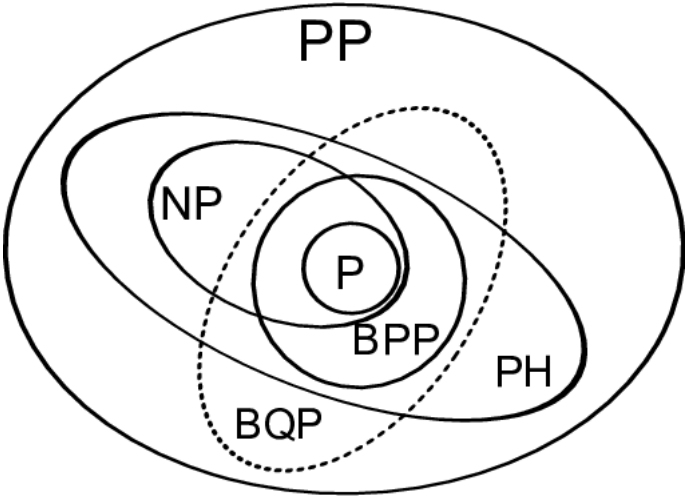
- Soit le nombre 62062883, quelles sont (de tête) ses facteurs
- C'est compliqué à trouver... ce n'est pas polynomial selon le nombre de chiffres

J'affirme que $62062883 = 7877 \times 7879$,

- On m'a exposé une solution potentielle
- vérifier qu'elle est correcte est simple... on fait la multiplication (c'est polynomial)

La factorisation est typiquement un problème NP.

N et NP ne sont pas les seules classes de complexité



Les problèmes NP-Complets - théorème de Cook

Certains problèmes sont dit **NP-difficiles** si tous les problèmes NP peuvent se ramener à résoudre ce problème en particulier.

Un problème NP qui est également NP-difficile est dit **NP-complet**.

Le théorème de Cook démontre l'existence de problèmes NP-Complets.

Il existe de nombreux problèmes NP-complets (QUBO, Voyageur de commerce, SAT, MIS, ...) on en recensait plus de 21 en 1972.

Les problèmes de Karp 1/2

En 1972, dans la foulée du théorème de Cook en 1971, le mathématicien Karp rescence 21 problèmes qui sont tous de nature **NPC**, on peut passer de l'un à l'autre au prix d'une transformation polynomiale.

- SATISFIABILITY : le problème SAT pour les formules en forme normale conjonctive
- CLIQUE : le problème de détection de clique dans un graphe
- SET PACKING : Set packing (empaquetage d'ensemble)
- VERTEX COVER : le problème de couverture par sommets, le problème MIS qui lui est dual
- SET COVERING : le problème de couverture par ensembles
- FEEDBACK ARC SET : feedback arc set
- FEEDBACK NODE SET : feedback vertex set
- DIRECTED HAMILTONIAN CIRCUIT
- UNDIRECTED HAMILTONIAN CIRCUIT
- INTEGER PROGRAMMING : voir optimisation linéaire en nombres entiers
- 3-SAT : problème SAT dont les clauses ont 3 arguments au plus
- CHROMATIC NUMBER : coloration de graphe

53/117

Les problèmes de Karp 2/2

- CLIQUE COVER : partition en cliques
- EXACT COVER : couverture exacte
- MATCHING à 3 dimensions : appariement à 3 dimensions
- STEINER TREE : arbre de Steiner
- HITTING SET : ensemble intersectant
- KNAPSACK : problème du sac à dos
- JOB SEQUENCING : séquençage de tâches
- PARTITION : problème de partition
- MAX-CUT : problème de la coupe maximum
- QUBO : optimisation quadratique
- TSP : problème du voyageur de commerce

54/117

Simulated Annealing

Le QUBO peut être résolu, par des méthodes HPC classiques, à l'aide de l'algorithme de *Simulated Annealing* ou "recuit simulé" créé en 1983.

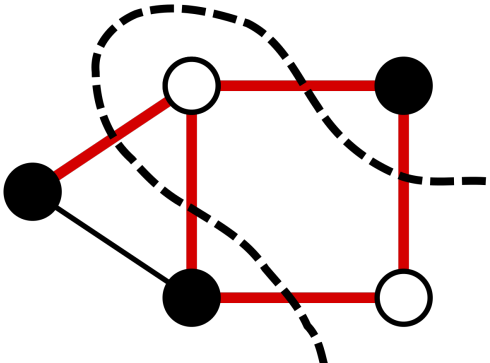
Cet algorithme se base sur une idée empruntée à la métallurgie. Dans ce domaine, il est utile d'amener le métal à son niveau d'énergie le plus bas, où il devient ductile et est plus facile à travailler. Les forgerons réalisent cela en alternant des cycles comprenant des refroidissements lents et des étapes de réchauffage, ou *recuits*[1].

L'algorithme de recuit simulé suit une approche dite "métaheuristique" qui dérive de l'algorithme de Metropolis-Hastings , issu de la modélisation des phénomènes thermodynamiques. De nombreuses implémentations existent, en particulier dans l'outil Mathematica. Il a souvent été utilisé pour résoudre des problèmes de graphes, en particulier ceux relatifs à la topologie de grands réseaux informatiques.

1. par opposition, un refroidissement rapide, ou *trempe* va laisser le métat dans un état d'énergie élevé où il est dur mais peut casser, pour forger des couteaux par exemple

Le problème Max-Cut 1/2

Étant donné un graphe, formé de sommets reliés par des arêtes, il s'agit de réaliser une coupe, donc de séparer le graphe en deux sous-ensembles complémentaires, tels que cette coupe au moins autant d'arêtes que n'importe quelle autre coupe possible.
Le problème Max-Cut est souvent associé à une notion de pondération des arêtes entre les sommets. On peut ainsi définir un "poids de la coupe" qui correspondra à la somme des poids des arêtes coupées. On dira qu'une arête est coupé quand les deux sommets, à chacune de ses extrémités, ne sont pas dans le même ensemble réalisé par la coupe.



Le problème Max-Cut 2/2

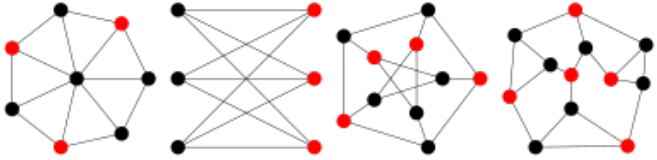
Remarque : d'une manière analogue, on peut définir un Min-Cut, une coupe mininum telle que le poids soit la plus faible valeur possible.
On peut associer deux types de problème à une coupe maximum :

- problème de *décision* : étant donné un graphe G et un entier k, existe-t'il une coupe de G dont le poids est au moins égal à k ;
- problème d'*optimisation* : étant donné un graphe G, quelle est la coupe maximum, celle qui maximise le poids ;

On peut démontrer que le problème Max-Cut se résout en un temps polynomial quand le graphe est planaire, il se ramène alors à l'identification des arêtes du graphe qui n'ont pas de sommets en commun. Un graphe est planaire s'il admet une représentation sagittale dans un plan sans que les arêtes se croisent.
Quand les graphes deviennent plus complexes, le problème de décision est NP-complet mais le problème d'optimisation est NP-dur.

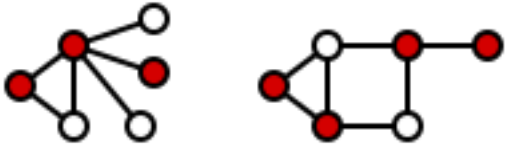
Le problème MIS

Étant donné un graphe G on peut définir un *ensemble indépendant* (ou *independant set*) par un sous-ensemble S de sommets de G tel qu'il n'existe aucune arête qui relit deux éléments de S.
Il existe souvent plusieurs solutions à ce problème.
Les MIS sont des *sous-ensembles dominants*. On dit qu'un sous-ensemble D d'un graphe G est dominant si chaque sommet de G est soit un élément de D, soit dispose d'un voisin dans D (donc il existe une arête qui le relie à un élément de D) s'il n'est pas dans D. Les MIS sont les plus grands sous-ensembles dominants.



Le problème MVC

Le problème MVC, acronyme de *Minimum Vertex Cover*, ou *problème de couverture par sommets*, est un problème dual du problème MIS.
Une couverture par sommets, aussi appelée *transversal* d'un graphe G est un ensemble C de sommets tel que chaque arête de $G = (V, E)$ est incidente à au moins un sommet de C, C'est à dire un sous-ensemble de sommets $S \subseteq V$ tel que pour chaque arête (u, v) de G on a $u \in S$ ou $v \in S$. On dit que l'ensemble C couvre les arêtes de G.



59/117

Le problème SAT 1/2

Le problème SAT ou le *problème de satisfaisabilité booléenne* est un problème de décision. Étant donné des variables booléennes (qui prennent les valeurs *vrai* ou *faux*) et une proposition, c'est-à-dire une formule qui combine ces variables avec des opérateurs booléens, on cherche à savoir s'il existe une combinaison de valeurs des variables qui rende vraie cette proposition .
Par exemple, la proposition $(p \wedge q) \vee \neg p$ est vraie pour toutes valeurs de q si p a la valeur *faux*, de même la proposition $(p \wedge \neg p)$ ne peut être satisfaite par aucune valeur de p.
Le problème SAT est l'archétype même des problèmes NP-complets. On peut identifier différentes formes simplifiées de SAT. Avant d'aller plus loin, on doit définir ce qu'est une *Forme Normale Conjonctive*, ou CNF [2]. Une conjonction est une opération "AND", une forme normale conjonctive est une équation booléenne qui est une évaluation d'une succession de clauses qui ne contiennent pas de AND.

2. en anglais l'acronyme devient *Conjunctive Normal Form*

Le problème SAT 2/2

Une CNF est donc de la forme $(clause1) \wedge (clause2) \wedge \cdots \wedge (clauneN)$. Chaque clause est elle-même de la forme $a \vee \cdots b$ avec éventuellement des négations $\neg v$. Une CNF s'écrit donc

$$\bigwedge_{i=1}^{p}(\bigvee_{j=1}^{q} l_{ij}) \text{ avec } l_{ij} = a_{ij} \text{ ou } l_{ij} = \neg a_{ij}$$

Les simplifications les plus courantes du problème SAT sont :

- le problème **CNF-SAT** correspond au cas où la proposition est une CNF ;
- le problème **3-SAT** est une restriction de CNF-SAT où chaque clause comporte au plus 3 variables ;
- le problème **2-SAT** est une restriction de CNF-SAT où chaque clause comporte au plus 2 variables.

Le problème 2-SAT est de complexité P, mais 3-SAT est de difficulté NP.
On peut montrer que le problème SAT se ramène toujours au problème 3-SAT.
Le problème SAT est la base de la démonstration du théorème de Cook.

Le problème QUBO 1/2

L'acronyme *QUBO* signifie *Quadratic Unconstrained Binary Optimisation*. Il permet de résoudre des problèmes d'optimisation qui se ramènenent, dans les grandes lignes, à la recherche d'optima d'une forme quadratique.
Etant donnée un entier $n \in \mathbb{N}$, étant donné $\mathbb{B}^n = \{0; 1\}n$, l'ensemble des vecteurs de taille n formés de 0 et de 1, étant donné une forme quadratique f_Q représentée par une matrice $Q \in n\mathbb{R}^{n \times n}$, quel est la valeur $x^* \in \mathbb{B}^n$ qui minimise $f_Q(x) = x^T.Q.x$
Le problème QUBO possède différentes propriétés :

- multiplier Q par un facteur α ne change pas l'optimum ;
- inverse le signe de Q (mettre un moins devant) revient à chercher un maximum plutôt qu'un minimum ;
- si Q est une matrice diagonale, le problème est trivial également, le bit de rang i sera 0 si Q_{ii} est positif et 1 sinon ;

62 / 117

Le problème QUBO 2/2

D'une manière générale, on ajoute parfois au terme quadratique un terme linéaire, l'énoncé devient alors

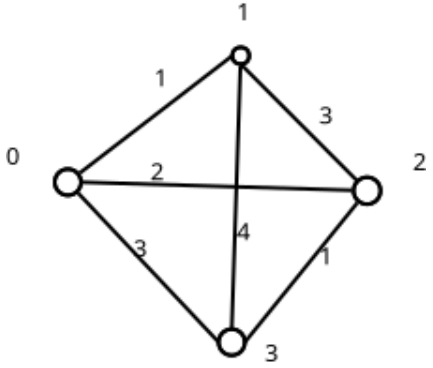
$$Q \in \mathbb{R}^{n \times n}, c \in \mathbb{R}^n, \text{ trouver la valeur } x^* \text{ qui minime } f(x) = x^T.Q.x + c^T.x$$

Le problème QUBO trouve des applications dans de nombreux domaines, car il est bien adapté à la recherche d'un optimum. Il intéresse ainsi les domaines de la finance, de l'économie, mais aussi la logistique et l'intelligence artificielle.
Le problème QUBO est de nature NPC

63/117

Résoudre MaxCut avec QUBO 1/3

Considérons le graphe suivant, il dispose de 4 noeuds, numérotés de 0 à 4, les arêtes portent différents poids allant de 1 à 4.
Ainsi, l'arête entre les noeuds 1 et 4 porte le poids 4 et celle entre 2 et 3 le poids 1.



Résoudre MaxCut avec QUBO 2/3

On peut traduire ce graphe par une "matrice de connectivité" W dont les coefficients w_{ij} sont

- 0 si $i = j$
- le poids de l'arête entre i et j si $i \neq j$

Dans notre exemple, on va construire la matrice suivante

$$W = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 4 \\ 2 & 3 & 0 & 1 \\ 3 & 4 & 1 & 0 \end{pmatrix}$$

Dans le cadre du problème MaxCut, on cherche une coupe maximale, donc un sous-ensemble E des nœuds du graphe qui maximise la coupe.
Connaissant E, on peut associer au noeud de rang i la valeur 1 s'il est dans E et 0 sinon. Cela nous permet de construire un vecteur x de valeur binaire. Inversement, un vecteur binaire $x \in \mathbb{B}^4$ décrit parfaitement une coupe.

65/117

Résoudre MaxCut avec QUBO 3/3

Considérons à présent la fonction de coût suivante qui représentée par un vecteur binaire x

$$x \in \mathbb{B}^4, C(x) = \sum_{i,j} W_{ij} x_i (1 - x_j)$$

- x_i est non nul si le noeud i est dans E
- $1 - x_j$ est non nul si le noeud i **n'est pas** dans E
- le terme $x_i(1 - x_j)$, qui est associé à l'arête entre i et j, sera non nul si cette arête est dans la coupe, donc si elle relie deux points dans E et hors de F
- on fait donc la somme des poids W_{ij} des arêtes dans la coupe, les autres termes sont nuls.

$$c \in \mathbb{R}^4, c_i = \sum_j W_{ij} \text{ et } Q = -W, \forall i, j Q_{ij} = -W_{ij}$$

$$C(x) = \sum_{i,j} W_{ij} x_i (1 - x_j) = C(x) = -\sum_{i,j} W_{ij} x_i . x_j + \sum_i c_i . x_i = x^T . Q . x + c^T . x$$

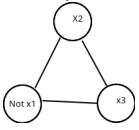
ce qui est la formulation classique du QUBO

Le problème 3-SAT est un problème de type MIS 1/2

Le problème 3-SAT, peut être résolu à l'aide d'un QUBO. La procédure consiste à traduire la formulation de 3-SAT en un problème de graphe de type MIS qui peut être lui-même transformé en QUBO.
Considérons une CNF qui comprend n variables et m clauses.

- pour chaque clause, on construit un petit graphe à 3 sommets, chaque sommet étant une variable ou la négation d'une variable ;
- chacun de ses "sous-graphes"[3], est connecté en raccordant les nœuds qui représentent une variable et sa négation.

Par exemple, la clause $\neg x_1 \vee x_2 \vee x_3$ sera représentée par le graphe de la figure suivante



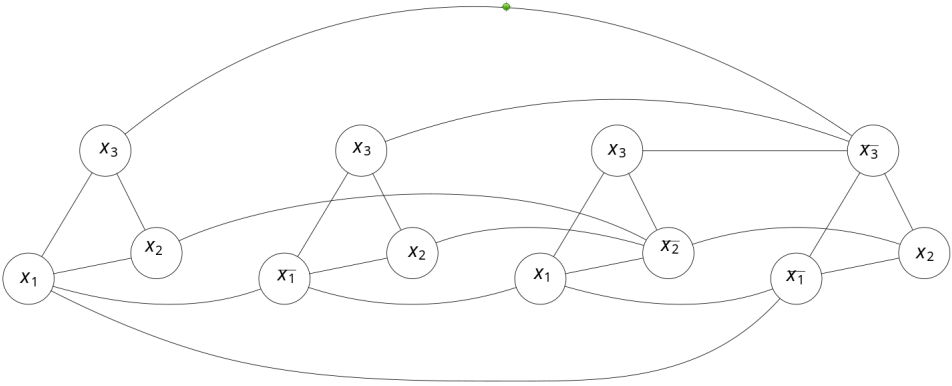
3. un tel sous-graphe est désigné sous le nom de *clique*

67/117

Le problème 3-SAT est un problème de type MIS 2/2

On interconnecte ensuite les sous-graphes en reliant les variables et leurs négations.
Considérons la CNF suivante

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$$



Problème 3-SAT via QUBO/MIS

On peut démontrer que la résolution du problème MIS sur ce graphe permet de trouver une solution maximale de la clause correspondante. Si la taille de ce MIS est égale au nombre de clauses dans la CNF, alors celle-ci peut être satisfaite.

On rappelle que SAT veut savoir si une clause peut oui ou non être satisfait, mais ne cherche pas forcément une combinaison de variables booléennes qui satisfasse. SAT cherche juste à savoir si une telle solution existe.

Si la taille du MIS trouvée est inférieure au nombre de clauses, alors la CNF ne peut pas être satisfaite.

69/117

Les problèmes NP sont partout

Les problèmes NP sont très communs en théorie des graphes et dans le domaine de l'optimisation.
Ils sont souvent simples à décrire mais complexe à résoudre

- Le problème du voyageur de commerce (TSP) est important en logistique
- la détection de "cliques" dans des graphes intéressent les réseaux sociaux
- de nombreux problèmes d'optimisation (coûts quadratiques) se ramènent à un QUBO
- MAxCut est associé à des algorithmes de prise de décisiion

Si le théorème de Cook garantit que la conversion "NP vers NPC" est P, trouver comment traduire est très compliqué. Une bibliogrpahie importante existe autour de QUBO qui peut être approché depuis 1984 avec du HPC.

70/117

Retour sur le problème MIS

Le problème que peut traiter la machine est PASQAL est le *Maximum Independant Set*

- considérons un graphe $(G; E)$
  - G est un ensemble de points, ou sommets (ou *vertices*) dans un espace à n dimensions
  - E est un ensemble de segments, ou arêtes (ou *edges*) qui relient deux points de G
- on cherche le, ou les, plus grand(s) ensemble de sommets dont les membres ne sont pas connectés entre eux.

Dans le cas des machines PASQAL, on considère des graphes **unitaires**

- les sommets dont la distance est inférieur à un rayon **R** donné sont connectés
- deux sommets connectés sont forcément plus proche que **R**

Le problème MIS est dual du problème MCS (*Minimum Convering set*

- identifier l'ensemble le plus petit possible de sommets où poser des "caméras" pour voir toutes les arêtes du graphe
- on démontre facilement que les solutions de MIS et MVS sont complémentaire

Machine PASQAL : Les bases physiques

Pour résoudre MIS, la machine PASQAL met en œuvre différents mécanismes

- des atomes de rubidium monovalents émis par une ampoule dans une chambre à vides, ralentis et positionnés par des lasers
- le niveau de valence le plus bas et le niveau le plus haut encode les états $|0\rangle$ et $|1\rangle$
- les atomes sont intriqués via le phénomène de blocage de Rydberg (*Rydberg blocade*)
- on sait éjecter les atomes dans l'état $|1\rangle$ (mais on sait parfaitement où ils étaient)
- on sait voir les autres par flurorescence, on sait donc mesurer les états de chaque atome

Par ailleurs

- ne sont intricables que des atomes dont la distance est inférieur au rayon de Rydberg
- deux atomes intriqués sont dans l'état $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$

L'API Pulser n'est pas une API de programmation

L'API de Pasqal est une API de "control command"

- On dit où sont les atomes
- on définit les rayon du graphe unitaire, ce qui définit les arêtes
- on définit la forme de l'onde laser qui va exciter les atomes
- on fait beaucoup de tirs et on en tire des conclusions

On ne fait pas de "programmes", on définit une expérience qui représente le problème qui nous intéresse

Exemple de code - Initialisation

On initialise l'environnement de programmation comme suit

```
import numpy as np
from pulser import Pulse, Sequence, Register
from pulser_simulation import Simulation
from pulser.devices import MockDevice
from pulser.waveforms import RampWaveform, ConstantWaveform

import matplotlib.pyplot as plt
import qutip
```

Exemple de code - description du graphe

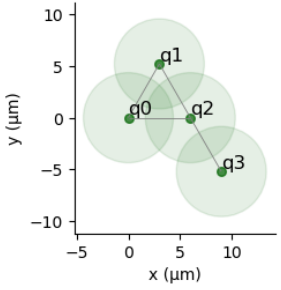
```
# Define a dictionary where each key is the name of the qubit,
# and each value is the qubit's position (in um)

qubit_positions = {
    'q0': (0, 0),
    'q1': (3, 5.2),
    'q2': (6, 0),
    'q3': (9, -5.2)
}
```

Ce dictionnaire est ensuite transformé en registre.

```
# Arrangements of qubits on the machine are called a register
# Define a register in Pulser by passing the qubit dictionary
reg = Register(qubit_positions)
reg.draw()
```

Rayon du graphe unitaire

```
# Define the maximum Rabi frequency via a blockade radius
blockade_radius = 8.7
Omega_max = MockDevice.rabi_from_blockade(blockade_radius)

# Visualize the edges induced by the chosen blockade radius
reg.draw(blockade_radius=blockade_radius)
```



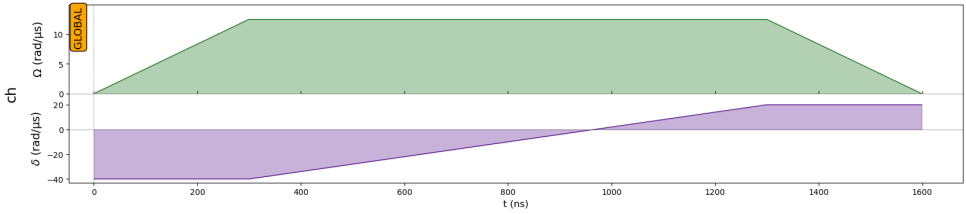
Définir l'impulsion laser

```
# A Sequence is the object that contains all
# the info about the quantum evolution
seq = Sequence(reg, MockDevice)

# Now we want to fill the channel with pulses
# First we need to define the waveforms for the pulses
# First ramp
omega_wf_1 = RampWaveform(300, 0, Omega_max)

#arguments are duration (ns), detuning (rad/us)
delta_wf_1 = ConstantWaveform(300, -40)

first_pulse = Pulse(omega_wf_1, delta_wf_1, 0)
seq.add(first_pulse, 'ch')
seq.draw()
```


77/117

Forme de l'impulsion

L'impulsion laser aura ainsi la forme suivante



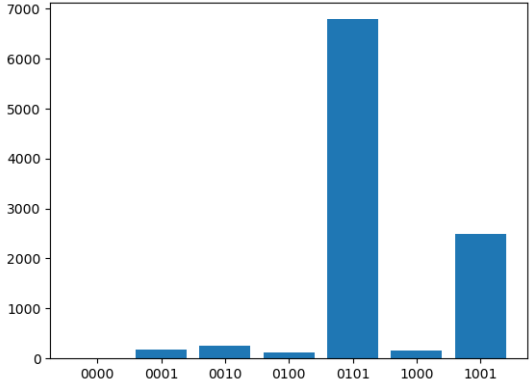
Pulser - fin du code

```
# The sequence is ready now for simulation

sim = Simulation(seq)
results = sim.run()

# The result can be sampled
samples = results.sample_final_state(10000)

# And the sampling can be visualized
plt.bar(samples.keys(), samples.values())
```


79/117

Résultat

On voit le résultat suivant



On voit deux solutions : $|0101\rangle$ et $|1001\rangle$. ce sont les solutions du problème MIS.

Besoin de middleware analogique

Les machines analogiques sont moins sensibles au bruit que les machines à portes, mais elles offrent une interface de très bas niveau

- il faut du middleware pour écrire un problème utilisateur en MIS
- il faut du middleware pour convertir le MIS en expérience quantique

Le gros challenge (pour les futurs informaticiens quantiques) sera de réaliser cette intégration du HPC et du QC, ce qui passe par l'écriture de middleware : **tout est à faire ou presque ! ! ! !**

Plan

82 / 117

Le HPC et le QC sont des alliés naturels

Le HPC et le QC sont complémentaires du point de vue du système

- on ne tournera (a priori) jamais d'OS sur un ordinateur quantique
- il faut du HPC standard pour faire une bootstrap du QC

Le HPC et le QC sont complémentaires sur le plan numérique

- le QC est très efficace pour résoudre des problèmes NP
- certains problèmes complexes deviennent simple en QC (ex : les DFT vues comme des QFT)
- certaines opérations très simples sont difficiles sur QC (additions et produits...)

Mélanger le HPC et le QC est naturel et on peut ainsi bénéficier du meilleur des deux mondes.

83/117

QAOA : un exemple concret

Le *Quantum Approximate Optimization Algorithm* ou **QAOA** est très populaur. Ils donnt de bons résultats avec peu de portes et peu de qubits, même sur une machine bruitée (NISQ)

QUBO peut être résolu par du quantique analogique : D-Wave réalise une implémentation matérielle du QUBO

QAOA sur des portes utilise des itérations optimisées sur des circuits peu profonds pour émuler l'évolution adiabatique exploitée par D-Wave.

84/117

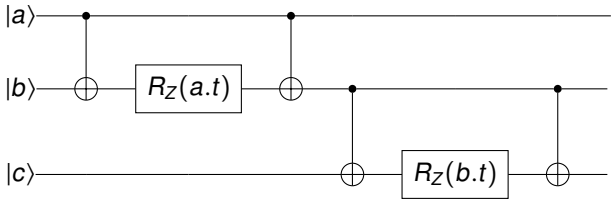
QAOA "à portes" en avance rapide

D'un point de vue très global, on peut résumer QAOA ainsi

- discrétiser l'évolution adiabatique en minuscules étapes
- pour chaque étape,
  1. créeer un petit circuit paramétré par les vecteurs β et γ
  2. on obtient un résultat ρ qu'on injecte avec (β, γ) dans un opetimiseur HPC (COBYLA est souvent utiliisé par d'autres solutions existes)
  3. on obtient les paramètres (β', γ'), on configure le circuit et on recommence au point #1.
- itérer jusqu'à ce que l'optimiseur décide que le résultat est assez précis

Design des circuits QAOA

Ces circuits exploitent peu de qubits et peu de portes.
On utilise des rotations de Pauli standard et des portes CNOT.



QAOA sur hardware Pasqal

Les planches suivantes montrent comment réaliser un QUBO, via QAOA puis en adiabatique pur, sur Pasqal. Elles sont inspirés du tutoriel disponible ici :
`https://pulser.readthedocs.io/en/latest/tutorials/qubo.html`

QUBO est défini par une matrice réelle et symétrique Q dont la taille est $N \times N$. si $z = (z_1, z_2, \cdots, z_N) \in \{0,1\}^N$, on associe Q avec $f(z) = z^T.Q.z$.

On cherche)à minimiser f quand z varie dans $\{0,1\}^N$

La matrice Q dans nos exemple

Dans tout ce qui suit, la matrice Q est définie ainsi en syntaxe NumPy

```
Q = np.array(
    [
        [-10.0, 19.7365809, 19.7365809, 5.42015853, 5.42015853],
        [19.7365809, -10.0, 20.67626392, 0.17675796, 0.85604541],
        [19.7365809, 20.67626392, -10.0, 0.85604541, 0.17675796],
        [5.42015853, 0.17675796, 0.85604541, -10.0, 0.32306662],
        [5.42015853, 0.85604541, 0.17675796, 0.32306662, -10.0],
    ]
)
```


88/117

QUBO en mode paresseux : la version HPC

Connaissant Q on exploite la fonction suivante

```
bitstrings = [np.binary_repr(i, len(Q)) for i in range(2 ** len(Q))]
costs = []
# this takes exponential time with the dimension of the QUBO
for b in bitstrings:
    z = np.array(list(b), dtype=int)
    cost = z.T @ Q @ z
    costs.append(cost)
zipped = zip(bitstrings, costs)
sort_zipped = sorted(zipped, key=lambda x: x[1])
print(sort_zipped[:3])
```

Cela résoud QUBO ! ! Mais le coût est en $O(2^N)$
On trouve que 01011 et 00111 sont les solutions optimales

Exploitation de l'hamiltonien de Rydberg

L'interaction de Rydberg est la base du hardare Pasqal, son hamiltonien est

$$H = \sum_{i=1}^{N} \frac{\hbar\Omega}{2}\sigma_i^x - \sum_{i=1}^{N} \frac{\hbar\delta}{2}\sigma_i^z + \sum_{j<i} \frac{C_6}{|r_i - r_j|^6} n_i n_j$$

Via Pulser, on peut positionner les atomes où l'on veut, on paramètre ceux-ci en fonction de Q. En particulier les paramètres non-diagonaux de Q sont relatifs à l'interaction $U = \frac{C_6}{|r_i - r_j|^6}$ entre deux atomes.

On peut donc traduire QUBO en un *registyre d'atomes*. On peut cherche l'état de base de cet hamiltonien et en déduire un minimum de $f(z) = z^T.Q.z$

90/117

Traduction de la matrice en registre d'atomes 1/2

```
def evaluate_mapping(new_coords, *args):
    """Cost function to minimize. Ideally, the pairwise
    distances are conserved"""
    Q, shape = args
    new_coords = np.reshape(new_coords, shape)
    new_Q = squareform(
        DigitalAnalogDevice.interaction_coeff / pdist(new_coords) ** 6
    )
    return np.linalg.norm(new_Q - Q)
```

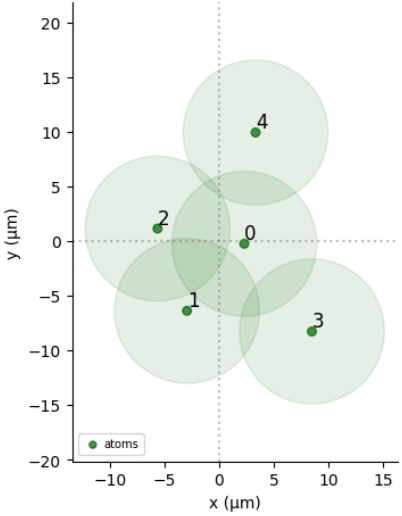
Traduction de la matrice en registre d'atomes 2/2

```
shape = (len(Q), 2)
costs = []
np.random.seed(0)
x0 = np.random.random(shape).flatten()
res = minimize(
    evaluate_mapping,
    x0,
    args=(Q, shape),
    method="Nelder-Mead",
    tol=1e-6,
    options={"maxiter": 200000, "maxfev": None},
)
coords = np.reshape(res.x, (len(Q), 2))
```


92 / 117

Allure de la grille d'atomes

Ces fonctions vont positionner les atomes ainsi pour notre matrice Q



93/117

Sequence Pulser paramétréee

```
LAYERS = 2

# Parametrized sequence
seq = Sequence(reg, DigitalAnalogDevice)
seq.declare_channel("ch0", "rydberg_global")

t_list = seq.declare_variable("t_list", size=LAYERS)
s_list = seq.declare_variable("s_list", size=LAYERS)

for t, s in zip(t_list, s_list):
    pulse_1 = Pulse.ConstantPulse(1000 * t, 1.0, 0.0, 0)
    pulse_2 = Pulse.ConstantPulse(1000 * s, 0.0, 1.0, 0)

    seq.add(pulse_1, "ch0")
    seq.add(pulse_2, "ch0")

seq.measure("ground-rydberg")
```

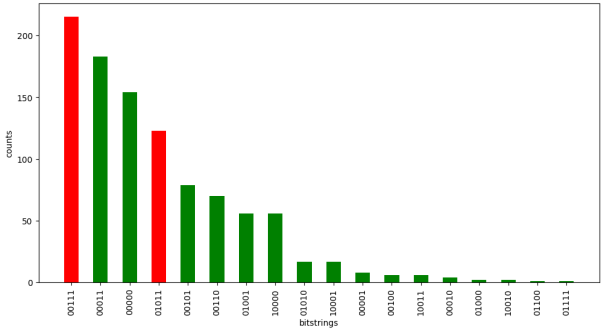
Itérations de la séquence

```
def quantum_loop(parameters):
    params = np.array(parameters)
    t_params, s_params = np.reshape(params.astype(int), (2, LAYERS))
    assigned_seq = seq.build(t_list=t_params, s_list=s_params)
    simul = QutipEmulator.from_sequence(assigned_seq, sampling_rate=0.01)
    results = simul.run()
    count_dict = results.sample_final_state()  # sample from the state vector
    return count_dict

np.random.seed(123)  # ensures reproducibility of the tutorial
guess = {
    "t": np.random.uniform(8, 10, LAYERS),
    "s": np.random.uniform(1, 3, LAYERS),
}
example_dict = quantum_loop(np.r_[guess["t"], guess["s"]])
```


95/117

Résultats après plusieurs itérations

Après un run de `quantum_loop`, On voit ceci (solutions cherchées en rouge)



On sait que 01011 et 00111 sont les solutions, elles ne ressortent pas de façon flagrante.

96/117

Passage par un optuiliseur

On construit une fonction de coût `get_cost`, qui calcule $z^T.Q.z$, elle est polynomiale.
On l'embarque dans `quantum_loop`

```
def get_cost(counter, Q):
    cost = sum(counter[key] * get_cost_colouring(key, Q) for key in counter)
    return cost / sum(counter.values())  # Divide by total samples

def func(param, *args):
    Q = args[0]
    C = quantum_loop(param)
    cost = get_cost(C, Q)
    return cost
```

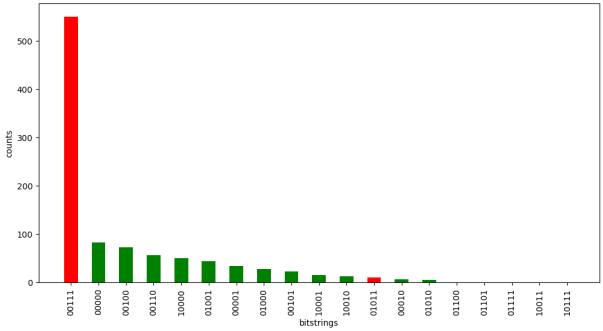

97/117

Invocation d'un optimiseur HPC

```
scores = []
params = []
(...)
    try:
        res = minimize(
            func,
            args=Q,
            x0=np.r_[guess["t"], guess["s"]],
            method="Nelder-Mead",
            tol=1e-5,
            options={"maxiter": 10},
        )
        scores.append(res.fun)
        params.append(res.x)
(...)
```


98/117

Résultats après plusieurs itérations de l'optimiseur



Une solution ressort clairement, mais la second est toujours inaccessible.

Thérorème adiabatique

En 1928, les physiciens Max Born et Vladimir Fock, énonce le théorème adiabatique :

> *Un système physique est maintenu dans son état propre instantané si une perturbation donnée agit sur lui suffisamment lentement et s'il y a un intervalle significatif entre la valeur propre et le reste du spectre de l'hamiltonien.*

En d'autres termes, si l'on sait placer un système quantique dans un état d'énergie de base connu, et qu'on le fait évoluer assez lentement et sans lui apporter d'énergie (c'est une évolution adiabatique) alors le système final, qui sera décrit par un nouvel hamiltonien, sera, lui aussi, dans un état d'énergie de base.

Calcul quantique adiabatique

Toute l'informatique quantique analogique s'appuie sur ce principe :

- on encode le problème à résoudre sous la forme d'un hamiltonien ;
- on part d'un système quantique dans un état de base connu avec un hamiltonien de référence connu ;
- on fait évoluer cet hamiltonien depuis celui de référence vers celui qui encode le problème, en respectant les contraintes du théorème adiabatique ;
- on dispose à la fin de l'état de base de l'hamiltonien de destination, celui qui encode le problème

Cette approche est très efficace, elle permet de trouver rapidement et de façon purement analogique, des minima de fonctions très complexes, dont la recherche avec des ordinateurs classiques relèvent de la nature *NP hard*.

Evolution adiabatique d'un hamiltonien de Rydberg

On exploite l'hamiltonien de Rydberg vu plus haut

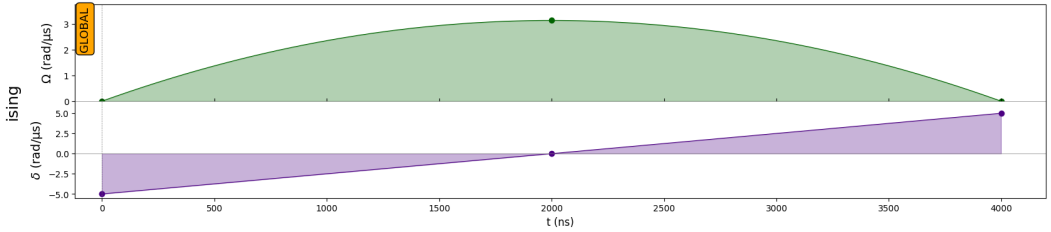
- on part de H_I où $\Omega = 1 rad/\mu s$ et $\delta = 0 rad/\mu s$
- on finit sur H_Q ou $\Omega = 0 rad/\mu s$ et $\delta = 1 rad/\mu s$
- on fabrique une séquence Pulser avec des formes d'ondes paramétrées qui réalise l'évolution adiabatique.

Code Pulser correspond

```
# We choose a median value between the min and the max
Omega = np.median(Q[Q > 0].flatten())
delta_0 = -5  # just has to be negative
delta_f = -delta_0  # just has to be positive
T = 4000  # time in ns, we choose a time long enough to ensure the propagation of

adiabatic_pulse = Pulse(
    InterpolatedWaveform(T, [1e-9, Omega, 1e-9]),
    InterpolatedWaveform(T, [delta_0, 0, delta_f]),
    0,
)
seq = Sequence(reg, DigitalAnalogDevice)
seq.declare_channel("ising", "rydberg_global")
seq.add(adiabatic_pulse, "ising")
seq.draw()
```

Evolution adiabatique

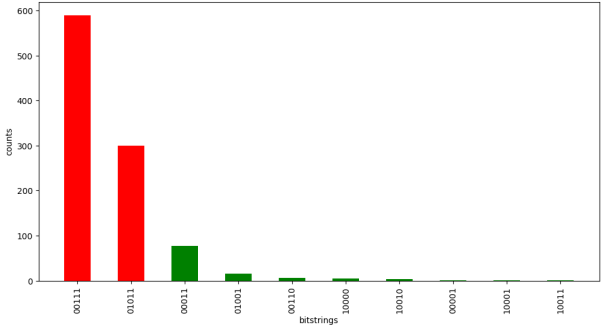
Les formes d'onde Ω et δ sont les suivantes



104/117

Résultats finaux

On obtient assez rapidement le résultat suivant



La solution arrive rapidement, cependant garantir que l'évolution de (Ω, δ) reste adiabatique peut être très complexe et requiert de connaître finement les phénomènes sous-jacents.

105/117

Plan

106/117

Crypto post quantique - contexte

La cryptographie post-quantique se propose de fournir des éléments de cryptographie qui soient résistants au QC

- l'algorithme de Shor met à mal la fiabilité de RSA
- la cryptographie post-quantique s'appuie sur des phénomènes quantiques, au niveau physique
- de vraies expériences de communication chiffrées avec du quantique ont été réalisées

107/117

Crypto post quantique - Implémentation

En fait les choses sont très simples, bien plus que de la cryptographie standard

- On utilise des "qubits volants" qui peuvent se déplacer (des photos)
- On construit un état intriqué et superposé dont Alice et Bob possèdent chacun une moitié
- le premier qui observe fait effondrer l'ensemble de l'état intriqué et superposé
- Alice et Bon vont mesurer la même chose chacun de leur côté
- ils se servent de cette mesure pour construire une clef commune

De là, on repart sur de la cryptographie standard, on peut faire des choses aussi simples que coder avec un XOR.

Si un intrus est présent, et veut se placer en MIM, on peut démontrer que la probabilité de le détecter est exponentielle avec la taille de la clef.

108/117

Corrections des erreurs 1/2

Les systèmes quantiques, souffrent de la présence de l'environnement

- Ils vont interagir avec l'environnement, donc s'intriquer avec lui
- cela revient à avoir une "boite noire" connectée au système et dont on ne connaît rien
- on peut démontrer que les erreurs sont toujours modélisables comme des circuits de portes classiques

109/117

Corrections des erreurs 2/2

On peut démontrer mathématiquement que

- les erreurs qui sont les plus intéressantes à corriger sont de deux natures
  - bit flip : présence d'une porte X qui va inverser les $|0\rangle$ et $|1\rangle$
  - phase flip : présence d'une porte Z qui va inverser la phase entre $|0\rangle$ et $|1\rangle$

110/117

Stratégies de correction des erreurs

Les approches classiques de l'informatique traditionnelle sont mises en œuvre :

- On construit des "qubit logiques" en agrégeant des qubits physiques
- On définit des stratégies qui permettent de détecter une erreur et parfois de la corriger
- les portes quantiques doivent être capables de gérer les qubits logiques, cela peut devenir potentiellement très complexe
- la stratégie de correction introduit plus de portes... donc plus de sources d'erreurs
- il faut trouver le bon compromis

Ces aspects disposent de bases mathématiques très solides (groupes normalisateurs et groupes stabilisateurs).

Plan

Le paradoxe de la modernité ancienne

L'informatique quantique effectue ses tous premiers pas, même si des bases solides existent

- la théorie physique est centenaire
- la théorie de l'informatique quantique a plus de trente ans
- en revanche les "vrais" ordinateurs quantique ont moins de dix ans

Il faut garder en tête que

- L'informatique quantique est donc au croisement de technologies et de techniques très récentes, mais avec des bases théoriques anciennes.
- La théorie plante un cadre, mais elle n'est sûrement pas exhaustive vis à vis des évolutions à venir.

113/117

Foisonnement technologique

L'informatique quantique s'axe autour de multiples axes technologiques

- différents paradigmes : portes quantiques, annealers, machines analogique, photonique quantique
- différents "qubits" sur des bases technologiques différentes : photons, boucles supraconductrices, qubits de chat, ions piégés, atomes neutres, qubits topologiques...
- différentes contraintes : cryogénie, gestion de laser, gestion de l'environnement magnétique de la salle machine

Dans ce contexte

- quelles technologies passeront à l'échelle pour offrir des milliers de qubits ?
- la question de l'énergie est très dimensionnante, encore plus que dans le HPC

Aujourd'hui, il est très difficile de dire quelles technologies ou paradigmes vont survivre dans un avenir au delà de 10 ans.

Controverses

Le sujet de l'informatique quantique a longtemps été ignoré avant l'algorithme de Shor qui lui a offert un violent coup de projecteur. Il regorge de controverses scientifiques

- le QC fournira-il un vrai "avantage quantique" ou bien le HPC sera-t'il toujours devant ?
- les algorithmes quantiques ne sont-ils pas que des "jouets pour mathématiciens" ? L'algorithme de Grover est en particulier très critiqué
- la logique "à portes" est-elle viable à long terme ?
- les systèmes analogiques ne sont-ils pas limités par nature ?
- le QC va-t'il finir par remplacer purement et simplement le HPC ?
- le QC n'est-il pas en train de changer le HPC ? (la logique "quantum inspired")

115/117

Un vaste terreau d'innovations

L'informatique quantique émerge, et elle progresse rapidement, et un grand nombre de choses restent à faire et à découvrir

- au niveau matériel (avec beaucoup de physique quantique)
- au niveau algorithmique (avec beaucoup de mathématiques)
- au niveau "système" car l'intégration HPC/QC c'est aussi adapter les environnements HPC existants
  - avec des impacts dans les environnements utilisateurs et de programmation
  - avec des impacts sur les techniques d'ordonnancement
  - avec de nouveaux services offerts aux utilisateurs
- au niveau du middleware : rien n'existe... ou presque, donc tout est à faire !

Quelque soit le visage qu'adopte le QC dans quelques années, tous ces points devront être abordés et résolus

116/117

La conclusion de la conclusion...

Le QC est un véritable "Far West" au sens du HPC : une nouvelle terre où tout est à bâtir et à découvrir.

Les domaines techniques et scientifiques touchés sont si larges que toutes les compétences étudiées peuvent être mises à contribution

Y investir du temps et de l'énergie n'est clairement pas une perte de temps ! ! ! !