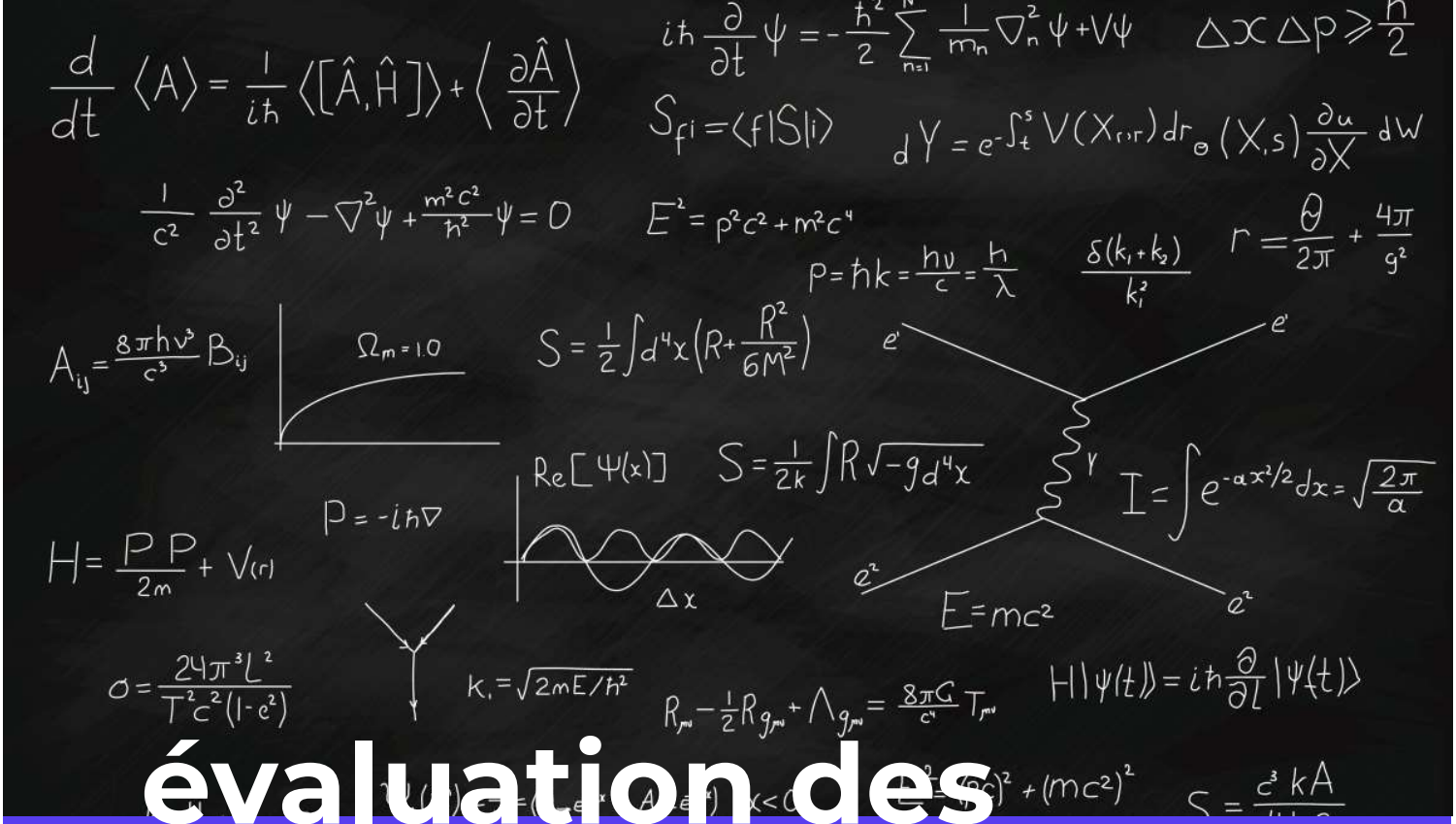


évaluation des technologies quantiques

application à résolution de problèmes 3-SAT et à la gestion de configuration.

F.BELLAICHE, PhD

JUILLET 2019

advisory & innovation
econocom

votre contact

Frédéric BELLAICHE

Directeur Technique – Cellule innovation
CTSA – Advisory & Innovation
mob.: +33 [0]6 07 85 37 23
e-mail: frederic.bellaiche@econocom.com

historique des versions

VERSION	DATE	PARAGRAPHE	ACTION	TYPE DE MISE A JOUR
000-A	12/06/2019	TOUT	C	Création du document
001-A	30/07/2019	TOUT	C	Finalisation du document

01

contexte et objectifs

1 contexte et objectifs

1.1 contexte

Un **problème de décision** est dit décidable s'il existe un algorithme qui, après un nombre fini d'étapes, peut répondre par oui ou par non à la question posée par le problème. S'il n'existe pas de tels algorithmes, le problème est dit indécidable.

Il existe plusieurs classes de problèmes de décision. Par exemple, la classe **P** est l'ensemble des problèmes de décisions dont la complexité est polynomiale (il peut être décidé par une machine de Turing déterministe). De même, la classe **NP** correspond à l'ensemble des problèmes dont la solution peut être vérifiée en temps polynomial (le problème peut être décidé par une machine de Turing non-déterministe).

Le **problème SAT** (boolean SATisfiability problem) est un problème de décision basé sur de la logique propositionnelle. Il consiste à décider si une formule en forme normale conjonctive d'ordre *n* est satisfiable, c'est-à-dire s'il existe une assignation de valeur de vérité (vrai ou faux) aux variables telle que toutes les clauses soient vraies. Un problème K-SAT est défini par une forme normale contenant K littéraux ($K \in \mathbb{N}^*$)

Les problèmes de type **3-SAT** sont centraux pour de nombreuses industries, et en particulier dans le secteur informatique. Ils correspondent typiquement à des problèmes de configuration ou d'assemblage à partir de composants standardisés. Ces problèmes peuvent s'exprimer de la manière suivante :

> Etant donné une configuration formée d'un ensemble de composants, de leurs dépendances et de conflits:
>
> - Déterminez si un nouveau composant peut être ajouté à la configuration
> - Ajoutez le composant tout en optimisant le dispositif (formalisé par une fonction linéaire)
> - Si le composant ne peut pas être ajouté, recherchez le moyen pour l'ajouter en supprimant autant que possible les composants en conflit de la configuration actuelle

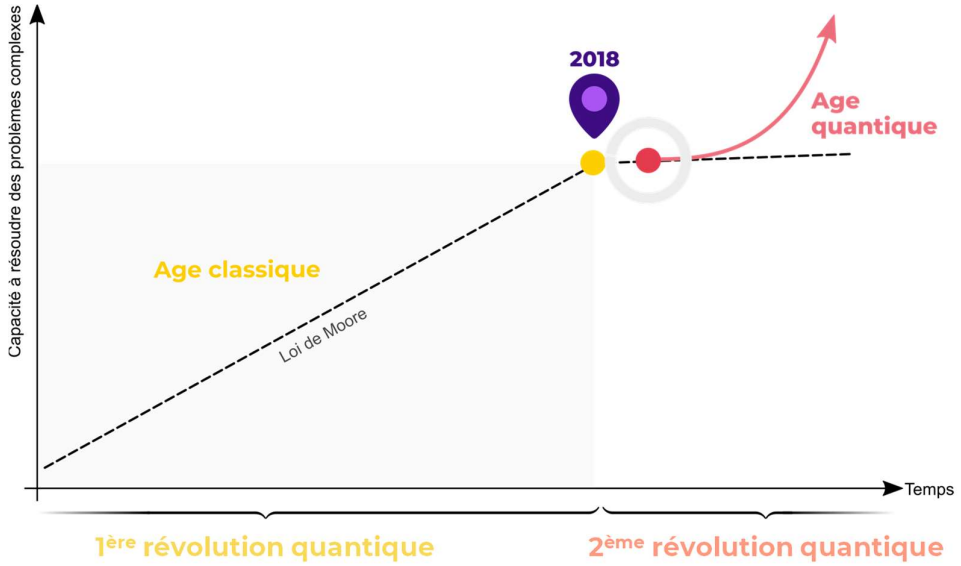
1.2 verrous technologiques et scientifiques

En 1971, **S. Cook** a établi que le problème **SAT** est **NP-complet**, c'est-à-dire qu'il appartient à la classe des problèmes les plus complexes à résoudre.

Cela signifie que même s'il existe des solutions classiques (i.e. non quantiques) aux problèmes SAT, celles-ci nécessitent nécessairement une quantité exponentielle de ressources informatiques, et donc implique un coût exponentiel.

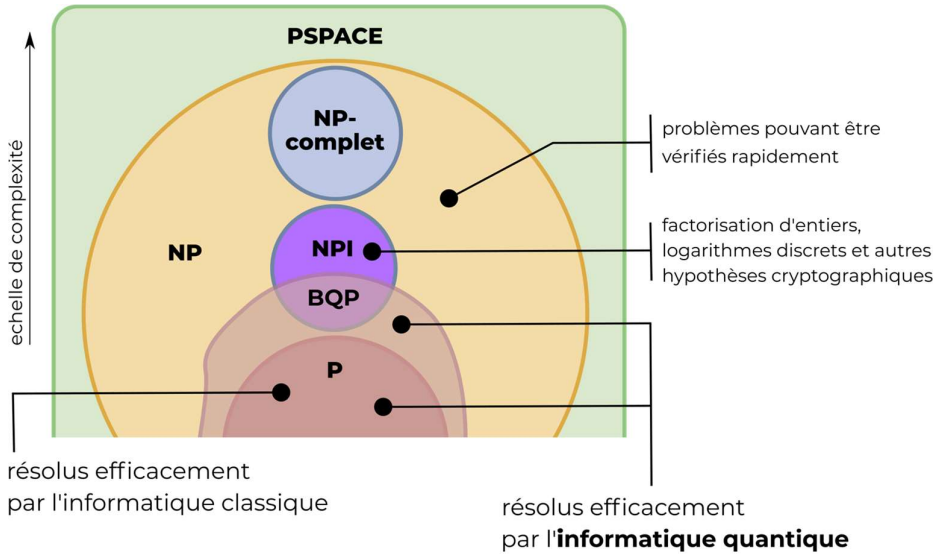
D'autre part, la **fin de la loi empirique de Moore** (la capacité de calcul double tous les 18 mois), estimée entre 2018 et 2020, constitut **une limite infranchissable à la capacité de résolutions des problèmes SAT** par des calculateurs conventionnels (non quantiques).

Les **calculateurs quantiques** sont une solution potentielle à ce problème. En exploitant les propriétés de **superposition** et d'**intrication** des états quantiques, il est possible de dépasser ces limitations, ouvrant le domaine dit de la « seconde révolution quantique » (la première ayant donné naissance aux technologies à base de semi-conducteurs, briques fondamentales de l'informatique actuelle) :



De manière plus formelle, la classe de complexité **BQP** (Bounded error Quantum Polynomial time), correspond aux problèmes de décision pouvant être résolus en temps polynomial par une machine de Turing quantique[1] avec une probabilité d'erreur d'au plus 1/3 dans tous les cas.

Celle-ci chevauche la classe de complexité **NP**, ce qui signifie qu'il existe des problèmes inaccessibles en pratiques aux ordinateurs classiques, mais pouvant être résolus par des algorithmes quantiques en temps polynomiaux :



1.3 objectifs de l'étude et des expérimentations

Comme le montre la figure précédente, les limites de la classe **BQP** ne sont pas encore complétement définis. L'extension de **BQP** au-delà des problèmes NP-intermédiaires (**NPI**) est **un sujet de recherche fondamental très actif**. Il n'est ainsi pas acquis que des algorithmes quantiques puissent résoudre tous les problèmes NP ou NP-complets.

[1] Une machine de Turing quantique définit de manière formelle le fonctionnement d'un calculateur quantique

Néanmoins, il est possible que certains **algorithmes quantiques** (et plus particulièrement l'algorithme de **Grover**) puisse résoudre de manière plus efficace **certains problèmes NP** ou **NP-complets**, comme la coloration de graphes, la détermination d'un chemin hamiltonien dans un graphe, certains cryptarithmes ou **certains problèmes SAT**.

1.4 déroulement de l'étude et de l'expérimentation

Dans ce contexte, le déroulement de l'étude et de l'expérimentation a été la suivante :

1. Etablir un état de l'art de l'informatique quantique
2. Etablir les domaines d'applications (sécurité, IA, data science, simulation, optimisation, …)
3. Etablir des critères de choix pour la sélection d'une plateforme quantique servant de base à l'expérimentation
4. Formuler un problème 3-SAT permettant l'évaluation d'un algorithme quantique (nous utiliserons l'algorithme de Grover)
5. Mettre en place un dispositif expérimental
6. Définir un oracle quantique pour le problème 3-SAT formulé
7. Implémenter l'oracle quantique et l'algorithme quantique sur simulateur
8. Implémenter l'oracle quantique et l'algorithme quantique sur calculateur quantique (accessible en SaaS)
9. A partir des résultats expérimentaux, évaluer les limitations dues au nombre de qubits utilisables par un processeur quantique actuel, aux bruits quantiques et aux temps de décohérences

02

technologies quantiques

2 technologies quantiques

Ce chapitre est une synthèse du document « Informatique Quantique ».

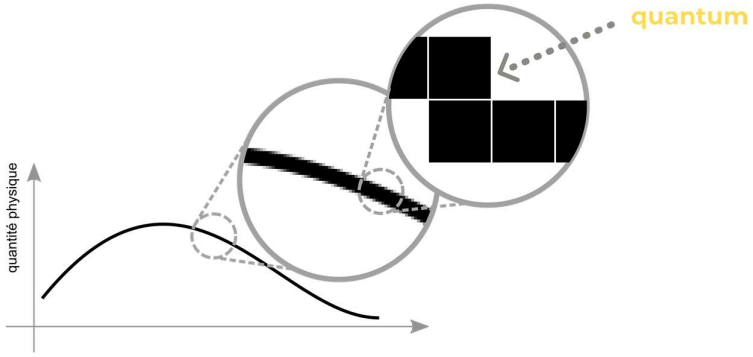
2.1 Notions quantiques

2.1.1 Quanta

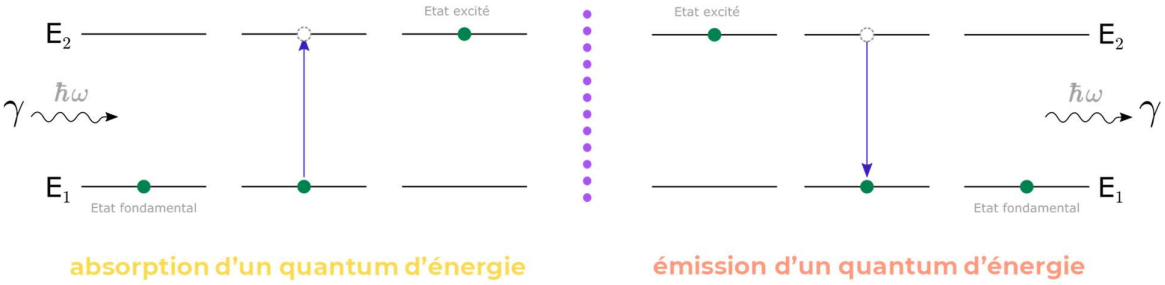
La théorie **quantique** est **l'un des deux pivots essentiels** à la compréhension de notre monde physique[1]. Née il y a 100 ans, elle est la source de bouleversements technologiques majeurs, du laser aux micro-ondes, en passant par les LED, l'imagerie médicale et, bien sûr, l'électronique basée sur **les semi-conducteurs, qui est le moteur de notre informatique contemporaine**.

Ces bouleversements ont été tellement disruptifs que l'on parle désormais de « **1ère révolution quantique** ». Ils reposent essentiellement sur deux propriétés quantiques de la Nature, dont la toute première est la notion de **quantum**.

De manière **très** imagée, un quantum est en quelque sorte un « pixel » élémentaire d'une quantité physique :



L'observation montre que **les grandeurs physiques** (énergie, matière, lumière, quantité de mouvement, champs,…) **ne sont pas de nature continue**. Les observables qui correspondent à ces grandeurs ne peuvent prendre qu'un **spectre de valeurs discrètes** :



Un quantum représente la plus petite mesure indivisible d'une quantité physique. Elles ne peuvent prendre que des **valeurs multiples entier d'un quantum**.

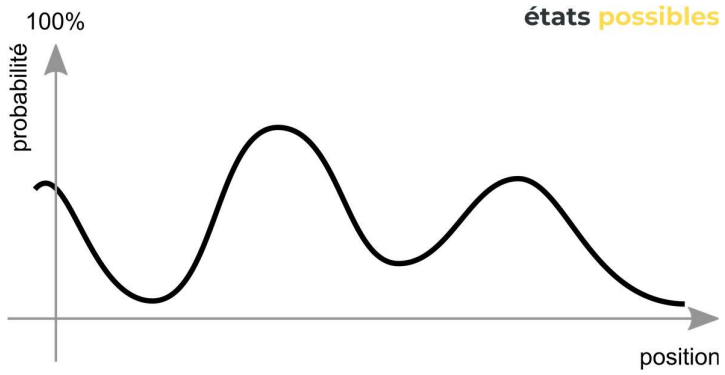
[1] Le second étant la relativité générale.

2.1.2 Probabilités

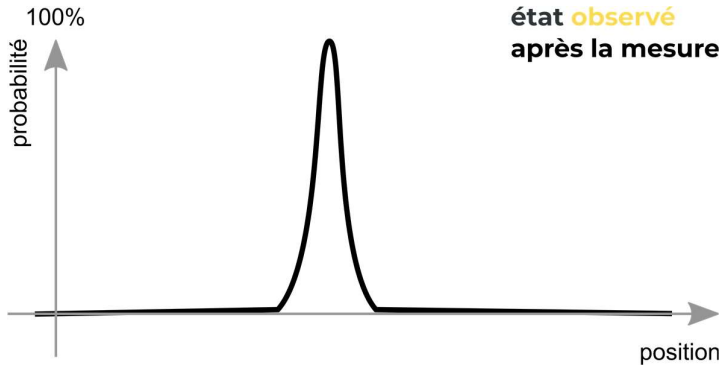
L'autre propriété à l'origine de la première révolution quantique est la notion de **probabilité** : alors que l'on pourrait imaginer qu'il soit possible de prédire avec exactitude le mouvement d'un objet, la **physique quantique constate (observe) qu'on ne peut pas prédire l'état final de l'objet, mais seulement les probabilités que tels ou tels états soient mesurés**.

De manière fondamentale, la Nature n'est pas **déterministe**, mais **probabiliste**. Les deux figures ci-dessus illustrent cette propriété contre-intuitive:

- Dans la première figure un système décrit par sa position spatiale, et les différentes probabilité de présence associée (telle position dans l'espace a telle probabilité d'être mesurée) :



- A l'issue de la mesure, on constate la position du système, parmi toutes celles possibles (et calculables). Cette « réduction » porte historiquement le nom d' « effondrement de la fonction d'onde » :



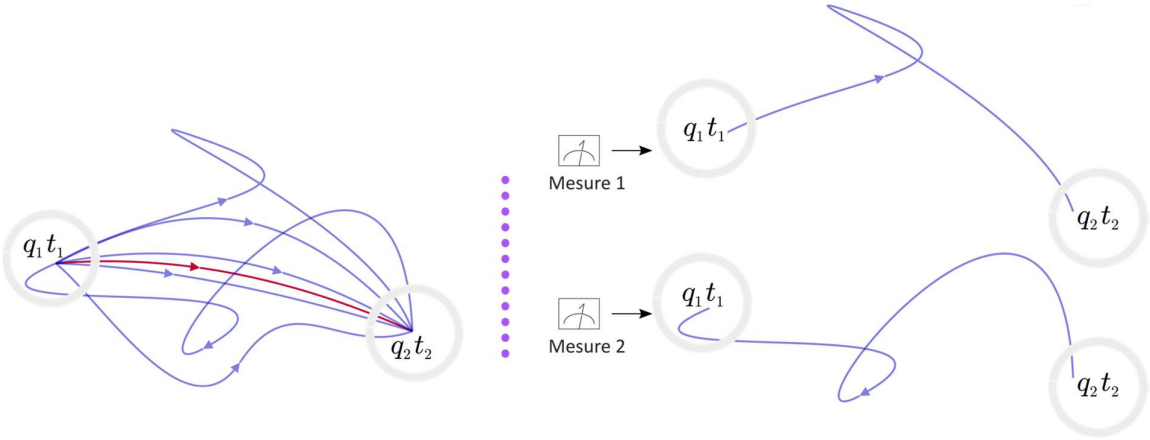
Au début des années 40, le physicien (et futur prix Nobel) Richard. P. Feynman, a introduit la notion d'**intégrale de chemin** pour décrire ce phénomène :

- L'état du système est décrit dans l'espace des phases (q_i, p_i) des coordonnées et impulsions généralisée à chaque instant ;
- Les chemins dans l'espace de phases les dynamiques possibles du système. A chacun de ces chemins est associé une amplitude de probabilité, c'est-à-dire ne fonction à valeurs complexes associée à la probabilité de trouver le système dans un état particulier. Le carré du module d'une amplitude de probabilité donne une probabilité d'observation ;

- Chaque chemin contribue à la probabilité d'observation proportionnellement à $e^{iS/\hbar}$ où S est l'action le long du chemin: en vertu du principe de moindre action, le chemin le plus probable est celui qui extrêmalise l'action.

La figure suivante illustre cette description (ici, dans le cas d'une seule dimension spatiale):

- Chaque chemin est figuré par une flèche
- Le chemin le plus probable (celui qui extrêmalise l'action) est figuré en rouge
- A l'issue de la mesure, le système se retrouve dans un des positions possibles dans l'espace des phases, avec une probabilité donnée, qui peut être calculée grâce au chemin associé
- Il est ainsi possible de prédire tous les états futurs possibles et leurs probabilités.
- Il est par nature **impossible** de prédire le résultat d'une mesure.



Si on recommence l'expérience, à chaque mesure, on obtiendra un des états possibles. Si on comptabilise ses résultats à chaque mesure, on constate qu'ils se distribuent selon une loi de probabilité, qu'il est possible de prédire entièrement à partir des lois de la mécanique quantique. Le « chemin en rouge » est alors le plus probable dans cette distribution de probabilité.

L'évolution dans le temps (c'est-à-dire la dynamique) de cette distribution est elle-même calculable par ces même lois (en l'occurrence, pour les phénomènes non-relativistes, cette évolution est la solution de l'équation de Schrödinger du système étudié)

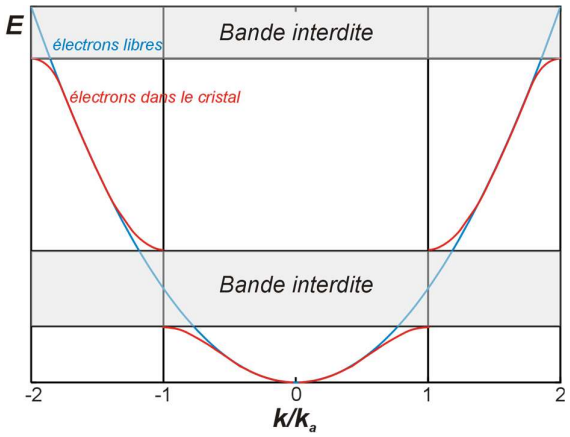
$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H|\psi(t)\rangle = \underbrace{-\frac{\hbar^2}{2m}\nabla^2|\psi(t)\rangle}_{\text{cinétique}} + \underbrace{V(r,t)|\psi(t)\rangle}_{\text{potentielle}}$$

2.1.3 Première révolution quantique

Ces deux propriétés (**quantum** et **probabilité**) sont à l'origine de la **première révolution quantique**.

Une illustration peut être donnée dans le cadre des semi-conducteurs. En physique quantique des solides, la théorie des bandes est une modélisation des valeurs d'énergie que peuvent prendre les électrons d'un solide à l'intérieur de celui-ci.

De façon générale, ces électrons n'ont la possibilité de prendre que des valeurs d'énergie comprises dans certains intervalles, lesquels sont séparés par des « bandes » d'énergie interdites. Cette modélisation conduit à parler de bandes d'énergie ou de structure de bandes :



(crédit : extrait de l'article Wikipédia sur les semi-conducteurs)

La dernière bande complètement remplie est appelée « bande de valence ». La bande d'énergie permise qui la suit est appelée « bande de conduction ». Elle peut être vide ou partiellement remplie. L'énergie qui sépare la bande de valence de la bande de conduction est appelée le « gap ».

Si le gap est grand, le solide ne contient aucun électron capable de participer à la conduction. On parle alors d'isolants.

Si la bande de conduction est partiellement occupée, alors un faible champ électrique peut faire passer un électron aux niveaux d'énergies supérieurs, sans dépenser beaucoup d'énergie, le solide est alors conducteur.

Entre ces deux cas se site le domaine des semi-conducteurs. Ce sont des matériaux qui ont les caractéristiques électriques d'un isolant, mais pour lequel la probabilité qu'un électron puisse contribuer à un courant électrique, quoique faible, est suffisamment importante. En d'autres termes, la conductivité électrique d'un semi-conducteur est intermédiaire entre celle des métaux et celle des isolants.

La conductivité électrique des semi-conducteurs peut être contrôlée par dopage, c'est-à-dire en introduisant une petite quantité d'impuretés dans le matériau afin de produire un excès d'électrons ou un déficit. Des semi-conducteurs dopés différemment peuvent être mis en contact afin de créer des jonctions, permettant de contrôler la direction et la quantité de courant qui traverse l'ensemble**. Cette propriété est à la base du fonctionnement des composants de l'électronique moderne : diodes, transistors, …** qui forment la base de nos micro-processeurs, RAM, et autre SSD.

La compréhension des isolants, des conducteurs et des semi-conducteurs est de nature quantique : l'ingénierie des semi-conducteurs est un pur produit de la compréhension des propriétés quantique des matériaux.

Il en est de même de la plupart des technologies contemporaines : **électronique et optoélectronique, laser, LED, IRM, microscope à effet tunnel, supraconductivité, superfluides, horloge atomique des GPS, fluorescence, micro-onde, cellule-photovoltaïques**, … La longueur de la liste justifie d'elle-même le titre de « **première révolution quantique** ».

Mais toutes les révolutions ont leurs limites. Dans le domaine de l'électronique, cette limite prend la forme symbolique de la fin de la loi de Moore, qui, techniquement, traduit la limitation de notre

capacité à graver de manière de plus en plus fine les microprocesseurs. Ironie du sort, **cette limite infranchissable** est de nature quantique.

Ironie dans l'ironie, **ce sont d'autres propriétés quantiques (superposition et intrication) qui sont à l'origine de la « seconde révolution quantique »**.

2.1.4 Superpositions

Dans le paragraphe 2.1.2, nous avons décrit sommairement la notion d'intégrale de chemin introduite par R.P. Feynman, où chaque chemin possible contribue à la probabilité d'observer un état.

Ainsi, en physique quantique, un état est représenté par la combinaison linéaire de tous les états finaux possibles. Autrement dit, **un état quantique est une superposition d'états de base**.

Pour illustrer le propos, considérons un système à deux états (que l'on nommera « $|flip\rangle$ ») de type « pile » ou « face » :

- Avant la mesure, l'état est inconnu, il peut être soit « face », soit « pile ».
- Après la mesure, l'état est connu, il devient « fac » ou « pile »
- Si l'état est connu avant la mesure (disons « face »), alors l'état final est connu (« face », en l'occurrence). La probabilité d'observer « face » est alors de $p = 1$ et la probabilité d'observer « pile » est de $p = 0$.
- Si l'état est inconnu avant la mesure, tout ce que l'on peut dire est qu'il existe une probabilité d'observer « face » ($p = 0.5$ si la pièce est parfaite) et une probabilité d'observer « pile » (qui est de $1 - p$). Les probabilités sont alors calculables, mais le résultat de la mesure ne l'est pas.

Pour **traduire mathématiquement** ce phénomène que l'on observe pour tout système microscopique à deux états, la mécanique quantique propose les éléments suivants:

- L'état $|flip\rangle$ est une combinaison linéaire des deux états de base « face » et « pile »
- Les « poids » α et β associés à ces états sont des nombres complexes : $\{\alpha, \beta\} \in \mathbb{C}$
- Lorsque l'on mesure l'état $|flip\rangle$ (« effondrement » de la fonction d'onde), la probabilité d'observer « face » ou « file » sont respectivement $|\alpha|^2$ ou $|\beta|^2$

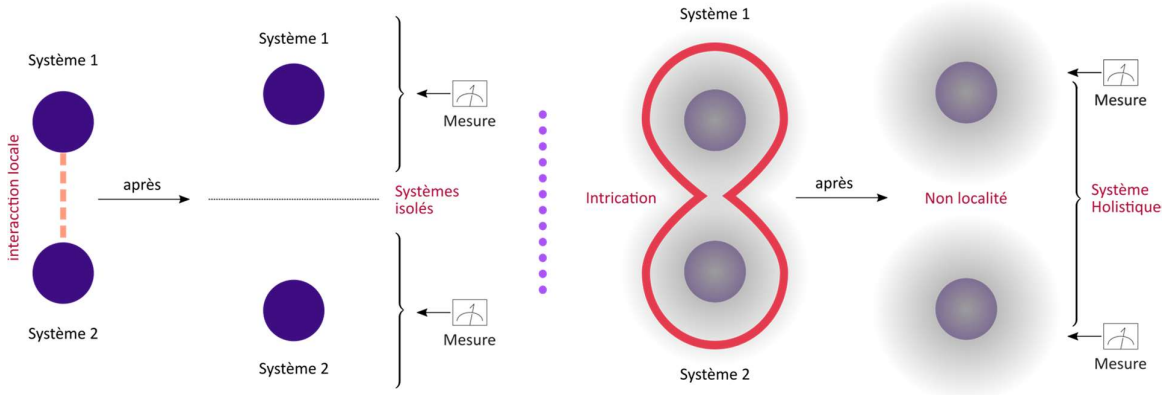
La notion s'étant naturellement à des systèmes à plusieurs états. Les coefficients complexes permettent de refléter les combinaisons constructives ou destructives que l'on observe lorsque les états se superposent (typiquement, les figures de diffraction observées dans l'expériences des fentes de Young).

2.1.5 Intrication

L'intrication est un phénomène quantique encore plus contre-intuitif. De fait, avant d'avoir été observé, il avait été proposé en 1935 dans une expérience de pensée sous la forme d'un paradoxe par A. Einstein, B. Podolsky et N. Rosen, lors de débat initiaux qui ont donné par la suite naissance à la mécanique quantique.

Des années plus tard, à la suite du théorème de J.S. Bell énoncé en 1964, puis des travaux du physicien français A. Aspect en 1981, ce qui était alors le paradoxe EPR est devenu une propriété fondamentale de la Nature.

L'intrication quantique concerne les systèmes en interaction. C'est un phénomène quantique dans lequel **deux états séparés forment en interagissant à un instant donné un état lié et présentent ensuite des états quantiques dépendants l'un de l'autre quelle que soit la distance qui les sépare**:



Les expériences EPR mettent en avant le fait que la physique quantique doit renoncer à la notion de localité : le système intriqué devient en quelque sorte un système **holistique**.

Notons :

- que la non-localité ne permet nullement la transmission d'un signal plus vite que la lumière (sans quoi d'ailleurs soit la causalité, soit la relativité serait violée)
- que l'intrication est un phénomène très fragile car elle est brisée par la moindre interaction (et en particulier la mesure)

2.1.6 Etats, opérateurs

Afin d'aller plus loin dans l'étude et l'expérimentation, il est nécessaire d'introduire des éléments de notation (définies par le physicien P.A.M. Dirac) et de formalisations mathématiques élémentaires.

La connaissance de l'**état quantique** est complétement contenue, à un instant t, par un **vecteur normalisable** (appelé « **ket** ») d'un espace de **Hilbert**[1] $\mathcal{H}$:

$$|\psi\rangle = \sum_i \alpha_i |k_i\rangle, \;\; \alpha_i \in \mathbb{C}, \;\; \sum_i |\alpha|^2 = 1, \;\; |k_i\rangle \in \mathcal{H}$$

A toute propriété **observable** (par exemple la position, l'énergie ou le spin), correspond un **opérateur Hermitien**[2] A agissant sur les vecteurs de l'espace de Hilbert $\mathcal{H}$.

La **mesure** d'une grandeur physique A correspond à l'une des **valeurs propres** de A. Les **vecteurs propres** associés correspondent aux **états quantiques** du système immédiatement **après la mesure** :

$$A|k_n\rangle = a_n|k_n\rangle$$

La **probabilité** de mesurer un état et la valeur moyenne associées à A sont :

$$\mathcal{P}_{|\phi\rangle} = |\langle\phi|\psi\rangle|^2 = \left|\sum_i \alpha_i \langle\phi|k_i\rangle\right|^2, \;\; \langle A\rangle_{|\phi\rangle} = \langle\phi|A|\phi\rangle$$

Un **opérateur** est, en mécanique quantique, une application linéaire d'un espace de Hilbert $\mathcal{H}$ sur lui-même. Quelques exemples :

- moment linéaire : $p = -i\hbar\nabla$
- hamiltonien : $H = \frac{p^2}{2m} + V(x)$
- moment angulaire : $L_z = -i\hbar\frac{\partial}{\partial\phi}$

Un **opérateur unitaire** est un opérateur linéaire U d'un espace de Hilbert $\mathcal{H}$ tel que $U.U^\dagger = U^\dagger.U = 1$

Les portes quantiques, qui seront introduites dans les prochains paragraphes, sont typiquement représentées par des matrices unitaires :

- matrices de Pauli : $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \;\; \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \;\; \sigma_z = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$

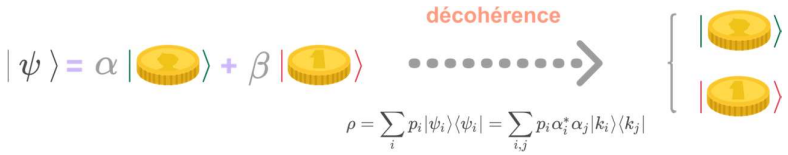
2.1.7 Décohérence

Un système ne peut pas être considéré comme complétement isolé. Il est en interaction avec un environnement possédant un grand nombre de degrés de liberté.

[1] Un espace de Hilbert est un espace vectoriel sur $\mathbb{C}$, complet (c'est-à-dire toute suite de Cauchy est convergente), pouvant être de dimension infinie, et doté d'une norme Hermitienne (c'est-à-dire sesquilinéaire à gauche et semi-linéaire à droite, positive, définie et doté de la propriété Hermitienne $\forall\, x, y \in \mathcal{H}\;\; \langle y|x\rangle = \overline{\langle x|y\rangle}$)

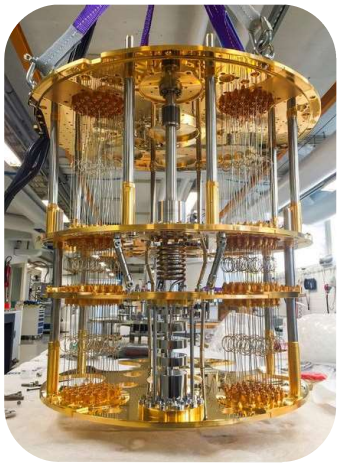
[2] Un opérateur Hermitien A est un endomorphisme autoadjoint, c'est-à-dire que $\forall\, x, y \in \mathcal{H}\;\; \langle x \mid A(y)\rangle = \langle A(x) \mid y\rangle$, i.e. $A = A^*$ (si A est borné). Ainsi, sur des espaces de dimension finie, les endomorphismes autoadjoints sont diagonalisable et ses valeurs propres sont réelles (et donc les valeurs physiques sont réelles)

Ces interactions provoquent la disparition rapide des états superposées. Ce phénomène s'appelle **décohérence** :



L'effondrement de la fonction d'onde n'est pas spécifiquement provoquée par un acte de mesure, mais peut avoir lieu spontanément, même en l'absence d'observation.

Les interactions et l'environnement ont des origines très diverses. **La décohérence est une des principales difficultés dans la construction d'un ordinateur quantique**. Ils nécessitent une grande **isolation** et l'introduction de mécanismes de **correction d'erreurs** :



2.2 Notions de performance et de complexité

Il est important de bien dissocier les notions de performance et de complexité.

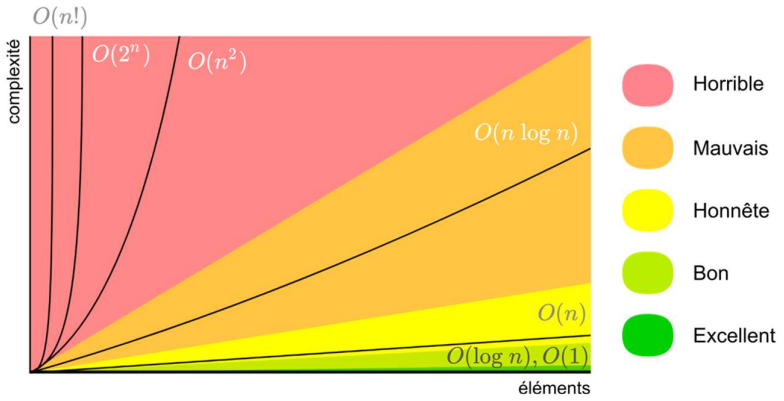
La **performance** peut se définir au travers du temps ou des ressources nécessaire à l'exécution d'un programme. Les performances dépendent de l'algorithme (code), mais aussi de la machine, du compilateur, etc.

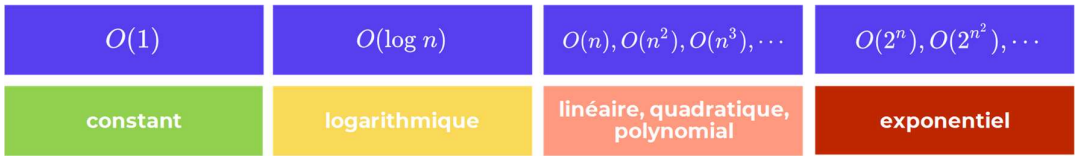
La **complexité** caractérise les besoins en ressources en fonction de la taille du problème : que se passe-t-il lorsque la taille du problème à résoudre devient de plus en plus grand ?

La complexité affecte la performance, mais pas l'inverse: un problème est complexe de manière inhérente.

Les notation « **big-O** » sont des **notations mathématiques** décrivant le **comportement limitant** d'une fonction lorsque l'argument croît. En informatique, les notations « big-O » sont utilisées pour **classer les algorithmes en fonction de la croissance de leur temps d'exécution ou de leurs besoins d'espace au fur et à mesure que la taille de l'entrée augmente.**

Exemple: une solution est dite « en temps polynomial » si son temps d'exécution est bornée (supérieurement) par une expression polynomiale (dans la taille de l'entrée)



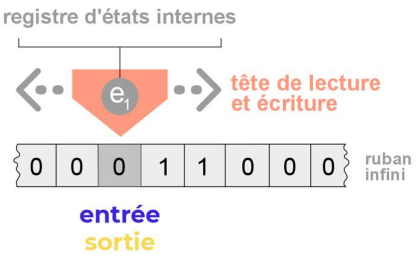


2.3 Machines de Turing

Les **machines de Turing** sont des **automates abstraits**, qui ont donné naissance aux premiers calculateurs universels programmables. **Tous les ordinateurs classique** (i.e. non quantique) sont **conçus selon ses principes**.

Une machine de Turing comporte les éléments suivants :

- Un **ruban infini** divisé en cases consécutives. Chaque case contient un symbole d'un alphabet fini donné. L'alphabet contient un symbole spécial appelé « symbole blanc », et un ou plusieurs autres symboles. Le ruban est supposé être de longueur infinie vers la gauche ou vers la droite, en d'autres termes la machine doit toujours avoir assez de longueur de ruban pour son exécution. On considère que les cases du ruban contiennent par défaut le « symbole blanc » ;
- Une **tête de lecture/écriture** qui peut lire et écrire les symboles sur le ruban, et se déplacer vers la gauche ou vers la droite du ruban ;
- Un **registre d'état** qui mémorise l'état courant de la machine de Turing. Le nombre d'états possibles est toujours fini, et il existe un état spécial appelé « état de départ » qui est l'état initial de la machine avant son exécution ;
- Une **table d'actions** qui indique à la machine quel symbole écrire sur le ruban, comment déplacer la tête de lecture (par exemple « $\leftarrow$ » pour une case vers la gauche, « $\rightarrow$ » pour une case vers la droite), et quel est le nouvel état, en fonction du symbole lu sur le ruban et de l'état courant de la machine. Si aucune action n'existe pour une combinaison donnée d'un symbole lu et d'un état courant, la machine s'arrête.



état	lit	écrit	déplace	suivant
e_1	⊗	⊗	←	e_2
e_2	0	0	←	e_2
	1	1	←	e_2
	⊗	⊗	→	e_3
e_3	0	1	→	FIN
	1	0	→	e_3
	⊗	1	←	FIN

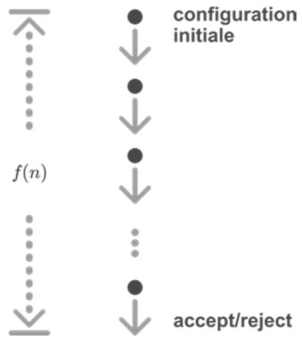
table d'actions
(ici, ajouter 1 en binaire)

Elles permettent de **formaliser** des notions importantes comme la **calculabilité**, la **décidabilité** ou la **complexité**.

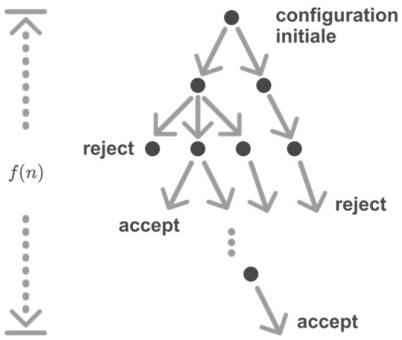
Formellement, une machine de Turing peut se définir de la manière suivante :

- Un ensemble fini d'états Q
- L'alphabet du ruban Γ
- Un état initial $\iota \in Q$
- Un symbole « blanc » $\square \in \Gamma$
- Un ensemble de symboles d'entrée $\Sigma \subseteq \Gamma \setminus \{\square\}$
- Une fonction de transition
  $\delta : Q \times \Sigma \to (Q \times \Gamma \times \{\leftarrow, \rightarrow\})$
- Un ensemble d'états finaux $F \subseteq Q$

Ce sont des **machines déterministes** (une seule transaction activable pour un état donné) :



Sur le modèle d'une machine de Turing déterministe, il est possible de construire des machines **non-déterministes**, c'est-à-dire pouvant avoir plusieurs transitions activables, pour un état donné :



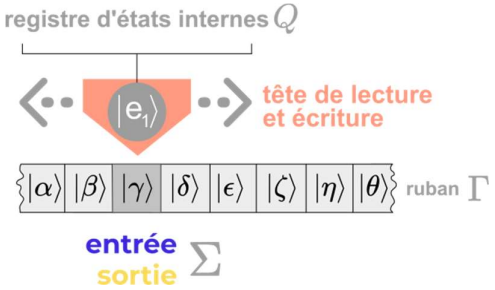
Formellement, une machine de Turing non-déterministes peut se définir de la manière suivante :

- Un ensemble fini d'états Q
- L'alphabet du ruban Γ
- Un état initial $\iota \in Q$
- Un symbole « blanc » $\square \in \Gamma$
- Un ensemble d'états acceptants $A \subseteq Q$
- Une fonction de transition binaire (fonctionnelle)
  $\delta : (Q \setminus A) \times \Gamma \to (Q \times \Gamma \times \{\leftarrow, \rightarrow\})$

Enfin, il est possible de définir formellement **une machine de Turing quantique** :

- L'ensemble fini d'états Q est remplacé par un espace de Hilbert
- L'ensemble des symboles Γ est également un espace de Hilbert
- L'état initial $\iota \in Q$ peut être un état quantique mixte ou pur
- Le symbole « blanc » $\square \in \Gamma$ est le vecteur nul
- Les entrées et sorties $\Sigma \subseteq \Gamma \setminus \{\square\}$ peuvent être quantiques ou non
- L'ensemble des états finaux est un sous espace $F \subseteq Q$
- La fonction de transition est formée de matrices unitaires (automorphismes de Q)
  $\delta : \Sigma \times Q \otimes \Gamma \to \Sigma \times Q \otimes \Gamma \times \{\leftarrow, \rightarrow\}$

De manière figurative, on peut alors représenter une machine de Turing quantique de la façon suivante :



La **thèse de Church-Turing** postule que toute fonction calculable peut s'exprimer à l'aide d'une machine de Turing. **Ce principe donna naissance aux ordinateurs classiques**.

La **thèse de Deutsch-(Church-Turing)** postule que tout système physique fini peut être simulé par une machine de Turing quantique. **Ce principe donna naissance aux ordinateurs quantiques**.

2.4 Classes de complexité

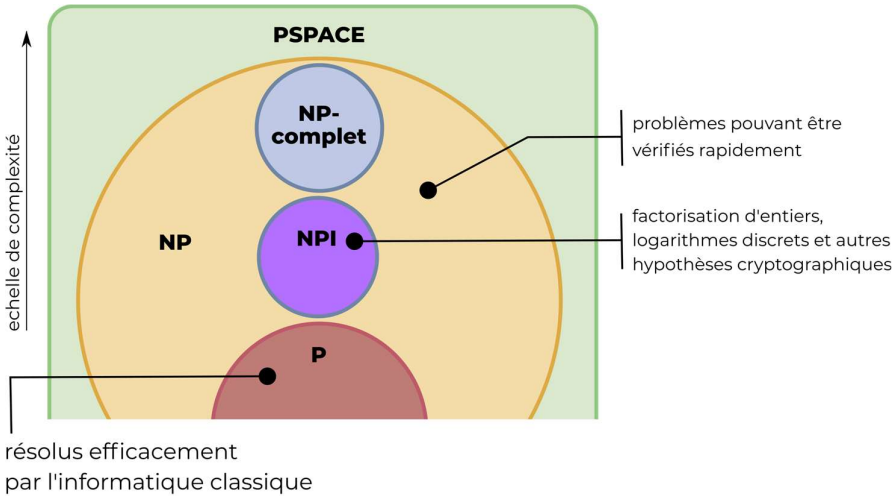
Une **classe de complexité** représente un ensemble de problèmes algorithmiques dont la résolution nécessite la même quantité d'une certaine ressource.

Elles sont en général définies au moyen de machines abstraites (**machines de Turing**) utilisant $O(f(n))$ ressources.

Il existe ainsi toute une hiérarchie de classes de complexité. On définit notamment :

- La classe **P** : problèmes qu'une machine de Turing peut résoudre en temps polynomial
- La classe **NP** : problèmes qu'une machine de Turing non déterministe peut résoudre en temps polynomial
- La classe **NP-complet** : problèmes NP les plus difficiles
- La classe **NPI** : problèmes NP intermédiaires

La figure ci-dessous illustre l'imbrication entre ces classes de complexité :

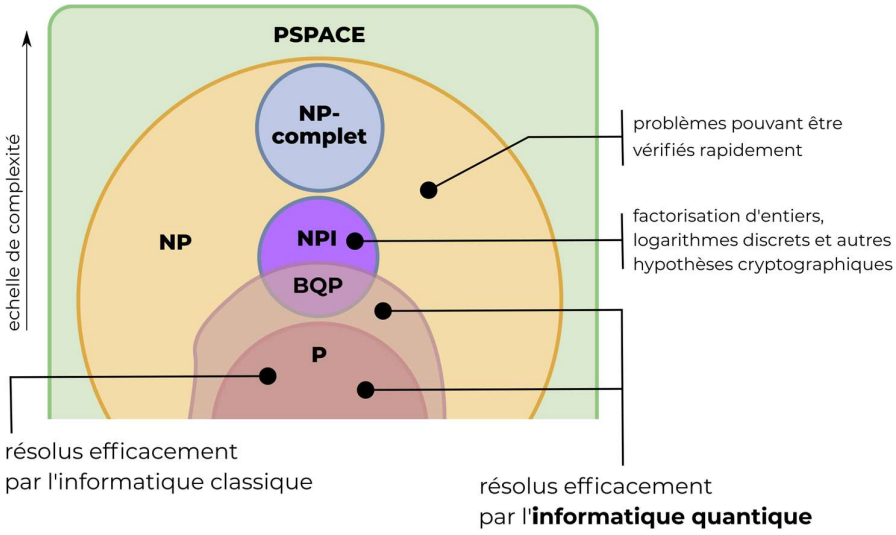


On peut également définir des classes de complexité à partir d'une **machine de Turing quantique**.

Par exemple :

- La classe **BQP** : problèmes pouvant être résolus en temps polynomial par une machine de Turing quantique, avec une probabilité d'erreur d'au plus 1/3 dans tous les cas

La frontières de ses classes de complexité n'est pas encore complètement identifiée, néanmoins, la figure suivante schématise les différentes imbrications :



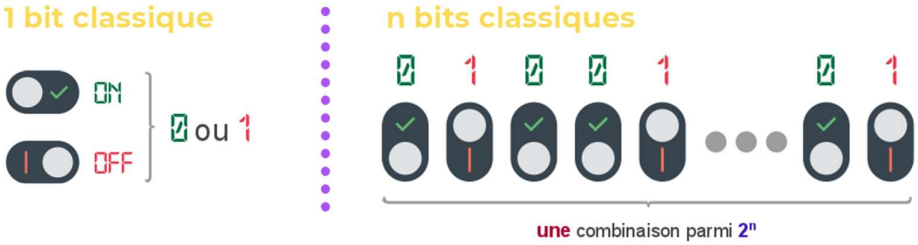
Il existe un chevauchement entre les classes **BQP** et **NP : les calculateurs quantiques sont capables de résoudre certains problèmes d'un complexité inaccessible au calculateurs classiques.**

2.5 Calculateurs quantiques

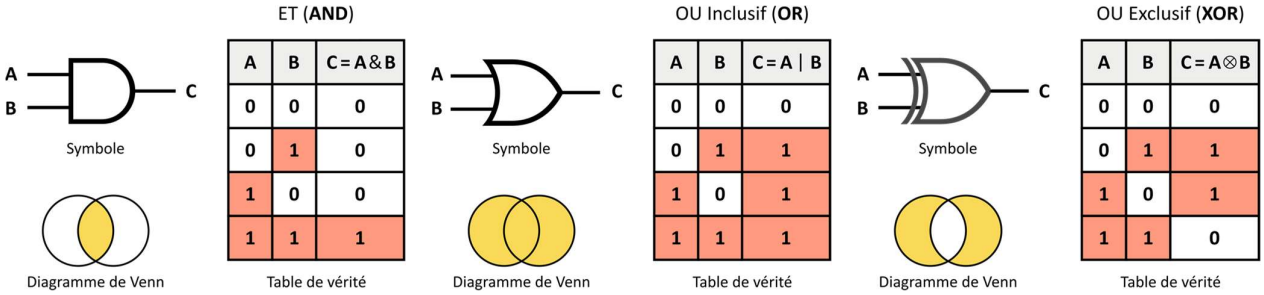
2.5.1 Principes des calculateurs classiques

De même que la conception des ordinateurs classiques s'est fondée sur la notion de machine de Turing classique, la conception des calculateurs quantiques se fonde sur la notion de machine de Turing quantique. Pour mieux comprendre ce parallèle, ce paragraphe rappelle les notions centrales de l'informatique classique :

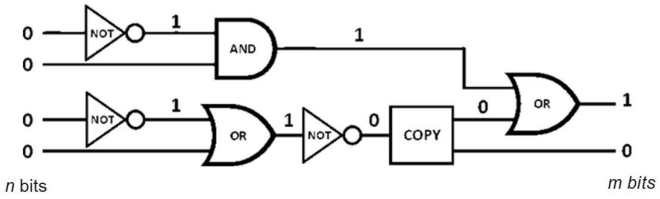
- **bit**: en théorie de l'information, un bit est la quantité minimale d'information transmise par un message, et constitue à ce titre l'unité de mesure de base de l'information. Un bit ne peut prendre que deux valeurs. En logique (algèbre de Boole), ces valeurs sont vrai et faux (oui et non). En arithmétique, ces valeurs sont 0 et 1.
- **ensembles de bits, octet:** un octet est un multiplet de 8 bits codant une information. Dans un système de codage s'appuyant sur le système binaire, un octet permet de représenter jusqu'à 2^8 nombres (1 nombre parmi 2^8). De manière général, n bits classiques permettent de représenter jusqu'à 2^n nombres (1 nombre parmi 2^n):



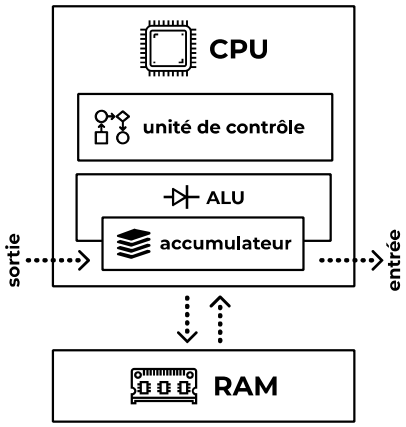
- **algèbre de Boole, opérations:** on peut agir sur des bits au moyen de fonctions (portes) logiques (conjonction, disjonction, négation, …) afin de modéliser des raisonnements logiques à l'aide d'opérateurs (assemblages de portes logiques) :



- **circuit:** un circuit Booléen est un graphe acyclique (i.e. sans cycles) dirigé (les liens ont une direction) avec n bits d'entrée et m bits de sortie:



- **architecture von Neumann:** on peut agir sur des bits au moyen de fonctions (portes) logiques (conjonction, disjonction, négation, …) afin de modéliser des raisonnements logiques à l'aide d'opérateurs (assemblages de portes logiques) :
  - **ALU** : unité arithmétique et logique, chargée des opérations
  - **CU** : unité de contrôle, chargée du séquençage des opérations
  - **RAM** : mémoire
  - **I/O** : dispositifs d'entrée-sortie

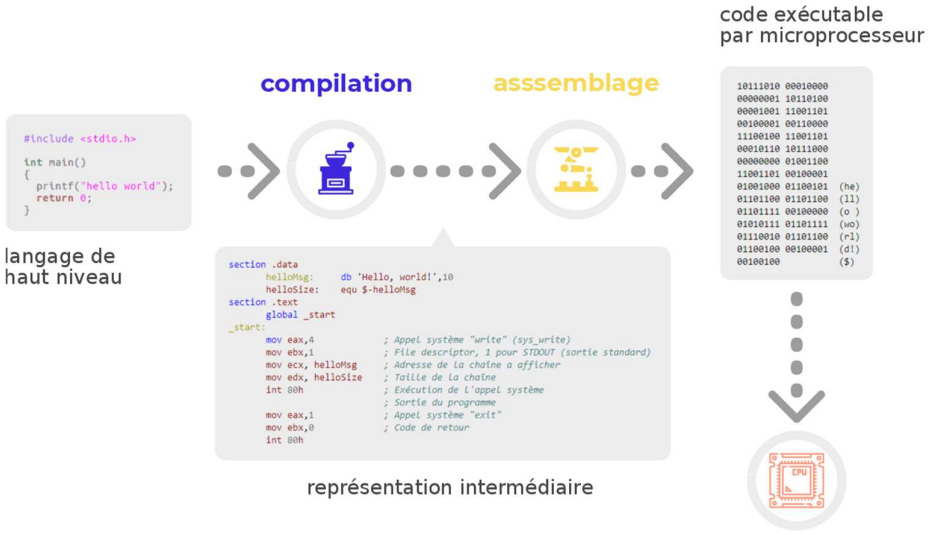


C'est sur la base de ses éléments que se fonde le cycle des langages de programmation classiques :

- Code source
- Analyse lexicale
- Analyse syntaxique
- Code intermédiaire

- Optimisation
- Génération
- Edition de liens
- Exécution

Ce cycle est représenté symboliquement par le diagramme suivant :



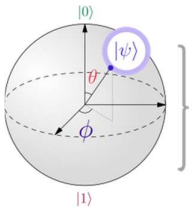
2.5.2 Bits quantiques

A l'image du bit classique, le **bit quantique** (ou qubit) représente la quantité minimale d'information.

Mais, en informatique quantique, il n'est pas représenté par une quantité binaire (vrai ou faux), mais par **un état quantique à deux niveaux**, c'est-à-dire, en utilisant les notations de Dirac (avec $i \in \{1,2\}$) :

$$|\psi\rangle = \sum_i \alpha_i |k_i\rangle, \;\; \alpha_i \in \mathbb{C}, \;\; \sum_i |\alpha|^2 = 1, \;\; |k_i\rangle \in \mathcal{H}$$

On peut donner une **représentation géométrique** d'un état pur d'un **système quantique à deux niveaux** grâce à la **sphère de Bloch** :



$$\begin{aligned} |\psi\rangle &= \alpha|0\rangle + \beta|1\rangle \\ &= e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right) \\ &0 \le \theta \le \pi, \;\; 0 \le \phi \le 2\pi \end{aligned}$$

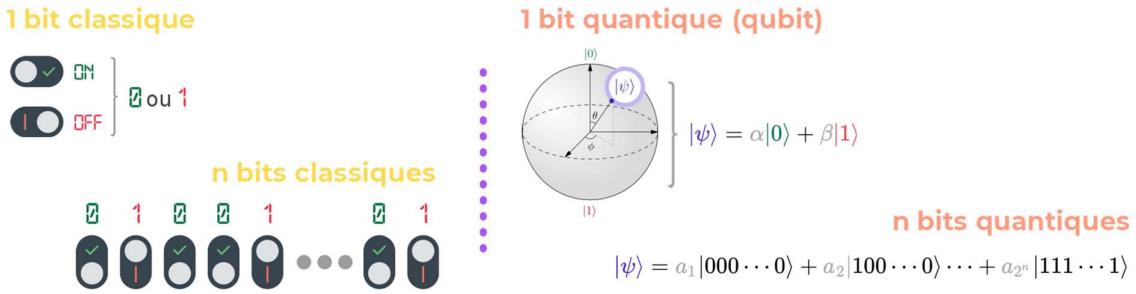
A l'image de l'octet, on peut rassembler entre eux plusieurs qubits. Un **registre quantique** est une combinaison de qubits, qui s'exprime sous la forme d'un **produit tensoriel** :

$$|\phi\rangle = \bigotimes_{i=1}^{n} |\alpha_i\rangle = |\alpha_1\rangle \otimes |\alpha_2\rangle \cdots |\alpha_n\rangle = |\alpha_1 \alpha_2 \cdots \alpha_n\rangle$$

Le tableau ci-dessous détaille ces notions pour un registre de 1 et 2 qubits :

	1 qubits	2 qubits
A espaces Hilbert	$\mathcal{H} = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$ ($|0\rangle$, $|1\rangle$)	$|0\rangle \otimes |0\rangle = |00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle = |1\rangle \otimes |1\rangle$ $\mathcal{H}_1 \otimes \mathcal{H}_2 = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}$
B états arbitraires	$|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$	$|\psi\rangle = \alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}$
C opérateurs	$U|\psi\rangle = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$	$U|\psi\rangle = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ u_{41} & u_{42} & u_{43} & u_{44} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}$

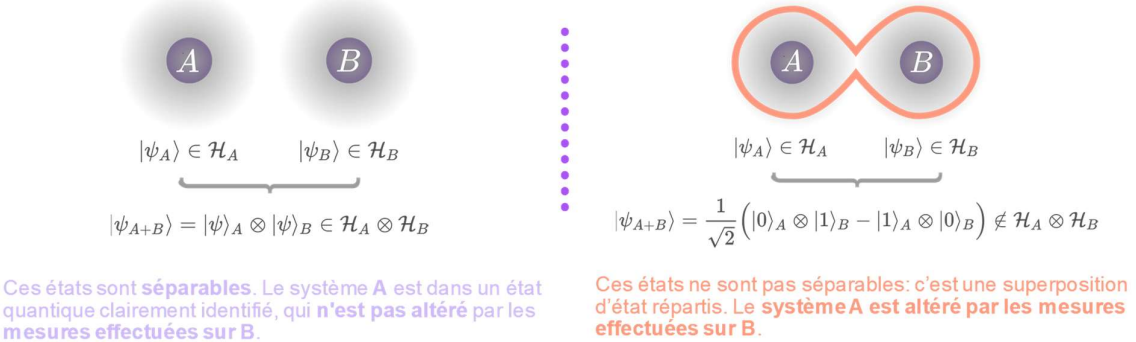
Notons une première conséquence immédiate : **là où *n* bits peuvent encoder n informations, n bits quantiques peuvent encoder 2^n informations :**



2.5.3 Etats intriqués

L'**intrication quantique** est un phénomène dans lequel deux particules forment un système lié et présentent des états quantiques dépendant l'un de l'autre quelle que soit la distance qui les sépare.

Formellement, deux états (ou plus) sont intriqués s'il n'est pas possible de l'écrire sous la forme d'un état séparable (élément du produit tensoriel des deux espaces de Hilbert) :



Ces états sont **séparables**. Le système **A** est dans un état quantique clairement identifié, qui **n'est pas altéré** par les **mesures effectuées sur B**.

Ces états ne sont pas séparables: c'est une superposition d'état répartis. Le **système A est altéré par les mesures effectuées sur B**.

Les **états de Bell** sont les **quatre états** à **2** particules les plus simples étant **maximalement** intriqués:

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B\right) \qquad |\Phi^+\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B\right)$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_A \otimes |1\rangle_B - |1\rangle_A \otimes |0\rangle_B\right) \qquad |\Phi^-\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_A \otimes |0\rangle_B - |1\rangle_A \otimes |1\rangle_B\right)$$

Un état de Greenberger–Horne–Zeilinger (**GHZ**) est un état **intriqué** à **au moins trois** particules tel que :

$$|\,\mathbf{GHZ}\,\rangle = \frac{|0\rangle^{\otimes n} + 1\rangle^{\otimes n}}{\sqrt{2}} \quad (n \geq 3)$$

Le plus simple des états GHZ est : $|\,\mathbf{GHZ}\,\rangle = \frac{|000\rangle + 111\rangle}{\sqrt{2}} \quad (n = 3)$

L'état **W** est l'autre classe d'états à 3 particules non séparables:

$$|W\rangle = \frac{1}{\sqrt{3}}\left(|100\rangle + |010\rangle + |001\rangle\right)$$

Il se généralise simplement : $|W\rangle = \frac{1}{\sqrt{n}}\left(|100\cdots0\rangle + |010\cdots0\rangle + \cdots + |00\cdots01\rangle\right)$

2.5.4 Portes quantiques

Une porte quantique (ou porte logique quantique) est un circuit quantique élémentaire opérant sur un petit nombre de qubits :

- Les portes quantiques sont réversibles (il est néanmoins possible, avec la porte de Toffoli, d'implémenter une fonction booléenne, au prix de devoir ajouter des bits auxiliaires)
- Elles sont représentées par des matrices unitaires $2^n \times 2^n$ (où n est le nombre de qubits sur lesquels la porte agit)
- Par exemple, une porte agissant sur 1 qubit est représentée par une matrice 2×2 :

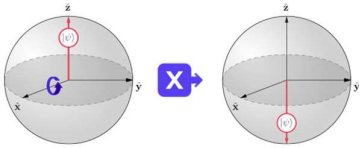
$$U|\psi\rangle = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$$

Les portes quantiques de **Pauli** agissent sur 1 qubit :

1 Pauli-X (bit-flip)

Equivalent d'une porte **NOT.** Rotation de π autour de l'axe X.

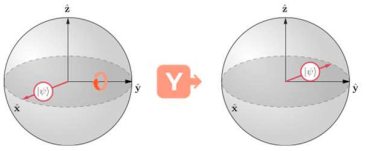
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$



2 Pauli-Y

Rotation de π autour de l'axe Y de la sphère de Bloch.

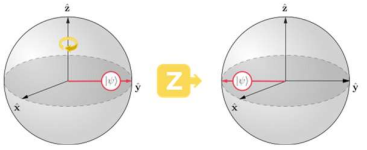
$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$



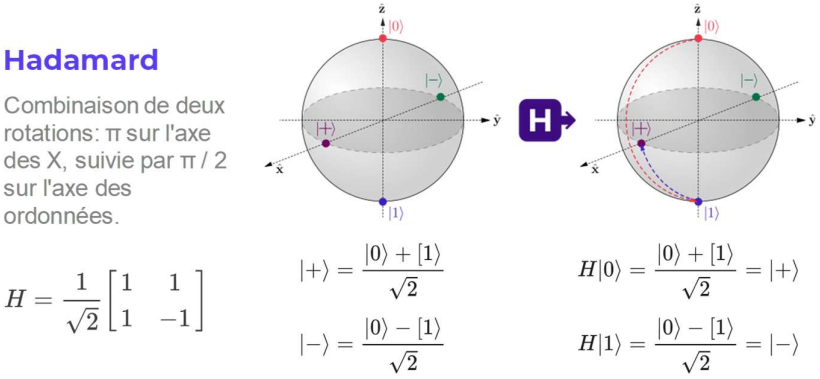
3 Pauli-Z (phase-flip)

Rotation de π autour de l'axe Z (changement de phase).

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



De même, la porte de **Hadamard** peut être définie par :



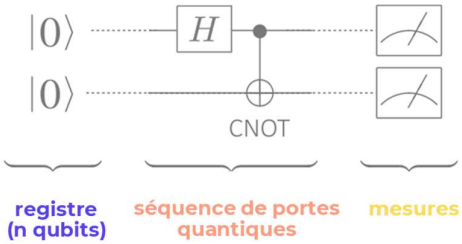
Le document « Informatique quantique » (.PPTX) détaille également les portes agissants sur plusieurs qubits :

- Portes contrôlés[1] **CNOT**, **CZ**, **CY** et **CU**
- Porte **SWAP**
- Porte **CSWAP**
- Porte de **Toffoli**
- Notion de portes universelles

2.5.5 Circuits quantiques

Un **circuit quantique** est un modèle dans lequel un algorithme quantique est exprimé par une séquence de portes quantiques opérants sur un registre de n bits quantiques :

- Un **registre** de n qubits est fourni en entrée
- Les **portes** quantiques sont les unités élémentaires du circuit quantique
- Des **mesures** sont effectuées en sortie



[1] Les portes contrôlées agissent sur 2 qubits (ou plus) comme un contrôle pour certaines opérations. Par exemple, la porte CNOT agit sur 2 qubits et n'effectue l'opération NOT sur le second qubit (target) que lorsque le premier qubit (control) est $|1\rangle$:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2.6 Correction d'erreurs et NISQ

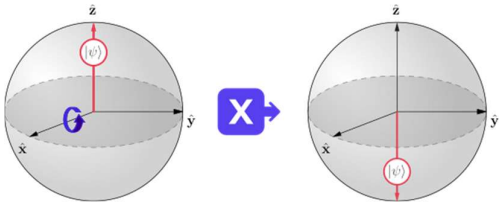
Un système quantique ne peut pas être considéré comme isolé, mais en interaction avec un environnement possédant un grand nombre de degrés de liberté. Ces interactions provoquent la disparition rapide des états superposés.

La décohérence est une des principales difficultés pour la construction d'un ordinateur quantique. Les opérations (portes), peuvent également introduire du bruit.

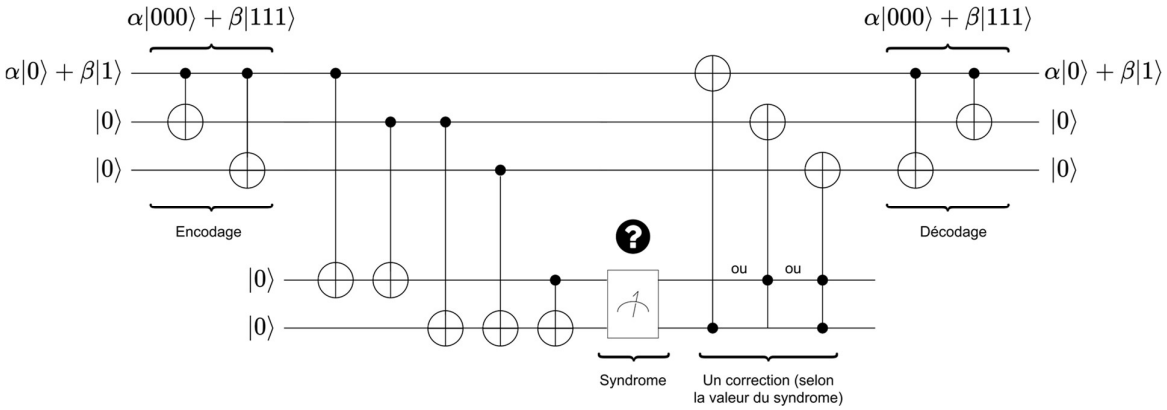
La durée de vie d'une superposition quantique est inversement proportionnelle à sa taille. Les ordinateurs quantiques nécessitent des mécanismes de correction d'erreur.

Pour cela, ces mécanismes doivent protéger les états contre :

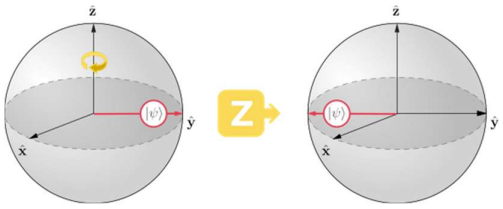
- Les **bit-flips** :



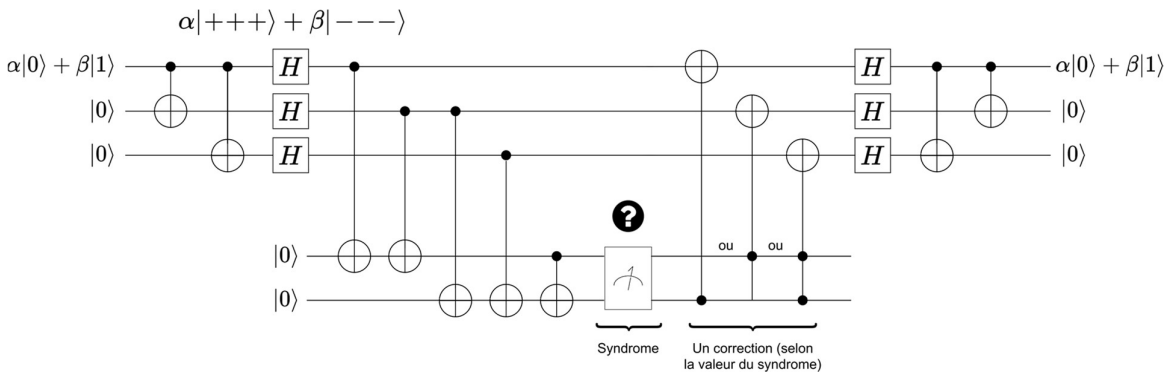
Le **code correcteur** pour un bit-flip peut être représenté par le circuit quantique suivant:



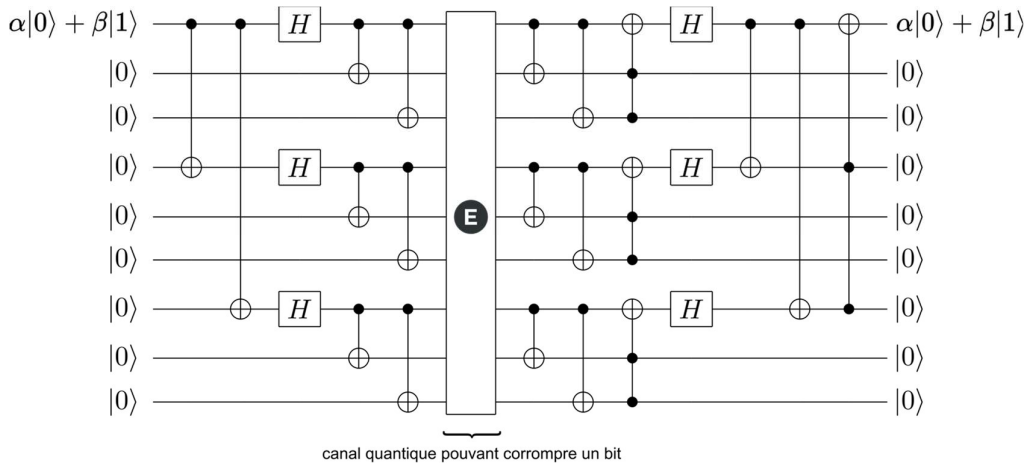
- Les **phase-flips** :



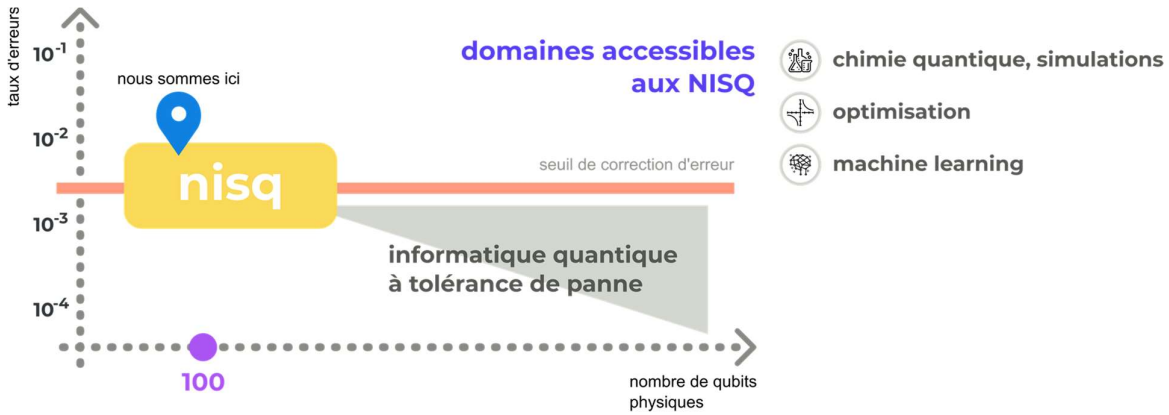
Le **code correcteur** pour un phase-flip peut être représenté par le circuit quantique suivant:



Le circuit quantique suivant correspond au code de correction de P. Shor sur 9 bits, qui combine les deux principes précédents (protection contre les deux types d'erreur) :



Le physicien John Preskill a inventé en 2017 le terme NISQ (« noisy intermediate-scale quantum computers »), désignant les types d'ordinateurs quantiques qui seront disponibles dans les prochaines années :



Le nombre de qubits n'est en aucun cas la seule mesure permettant de déterminer la puissance d'un ordinateur quantique. La qualité des qubits, la capacité à mesurer les erreurs et apporter des corrections est également un critère fondamental.

2.7 Algorithmes quantiques

Il existe un certain nombre d'algorithmes quantiques fondamentaux, qui servent de base ensuite à différents algorithmes appliqués.

Les algorithmes quantiques principaux sont :

- Agorithme de Deutsch-Josza
- Algorithme de Bernstein-Vazirani
- Algorithme de Simon
- Amplification d'amplitude et Algorithme de Grover
- Transformation quantique de Fourier
- Algorithme de Shor
- Téléportation quantique
- Codage superdense
- Marche aléatoire quantique
- Algorithme HHL
- …

Le schéma ci-dessous présentent ces principaux algorithmes, qui sont détaillés dans le document de référence « Informatique Quantique » :

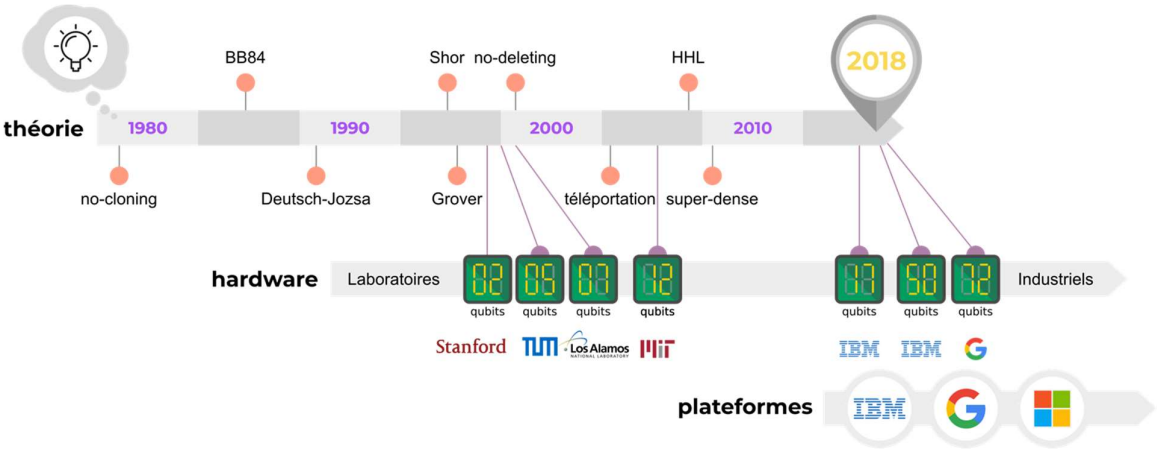
Sur la base de ces algorithmes fondamentaux, on peut citer les algorithmes appliqués :

- BB84, E90, … (distribution quantique de cles)
- Quantum SVM Kernel (IA, data science, optimisation)
- Quantum K-NN (IA, data science, optimisation)
- Quantum k-means clustering (IA, data science, optimisation)

- Machines de Boltzmann quantiques (simulations, optimisation)
- Réseaux bayésiens quantiques (IA, data science, optimisation)
- Modèles markoviens quantiques (IA, data science, simulation)
- Variational quantum eigensolver (optimisation, simulation)
- Etc.

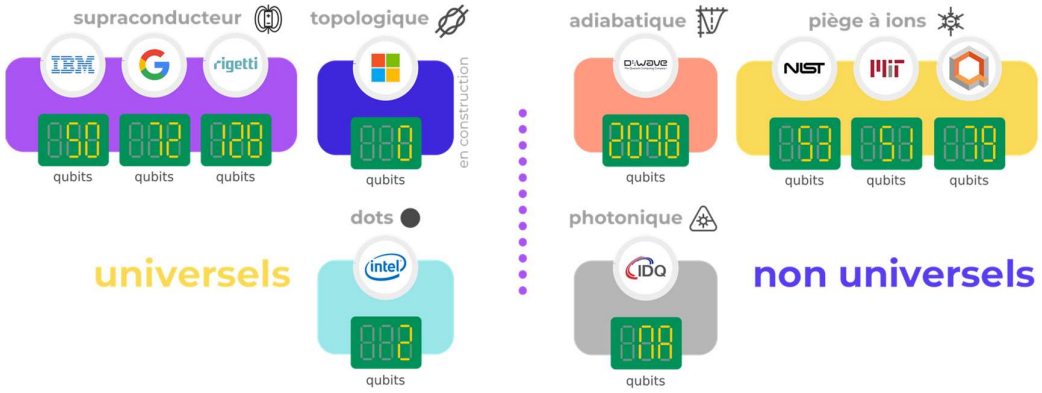
2.8 Plateformes quantiques

Si les bases théoriques de l'informatique quantique datent du début des années 1980, on constate depuis environ 2 ans une très forte accélération, portée par la disponibilité de plateformes quantiques en mode SaaS :



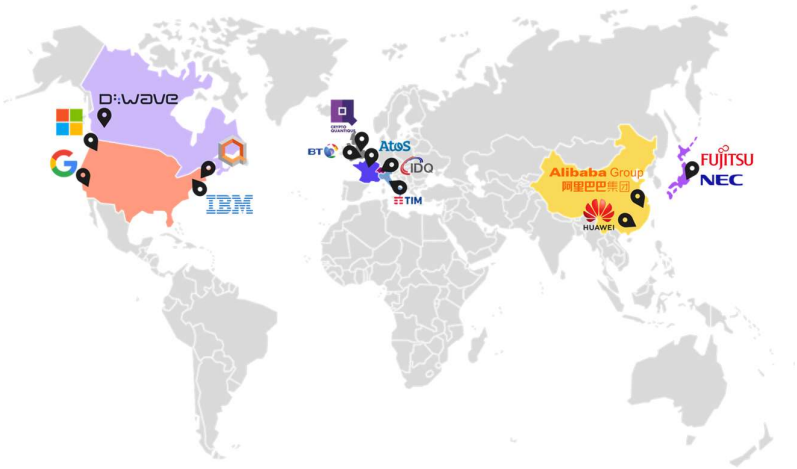
Il existe à l'heure actuelle différentes technologies permettant de concevoir un ordinateurs quantique. Les principales sont les suivantes :

- Ordinateurs quantiques à ions piégés
- Ordinateurs quantiques adiabatiques (dits également « à recuit quantique »)
- Ordinateurs quantiques optiques
- Ordinateurs quantiques à supraconduction
- Ordinateurs quantiques topologiques



Pour l'évaluation, nous avons identifié les constructeurs suivants[1] :

- **IBM**
- **D-Wave**
- **Rigetti**
- **Google**
- **Microsoft**
- **Atos**
- **IonQ**
- **Huawei**
- **Alibaba**



Sur la base de ces technologies, les plateformes quantiques se présentent généralement sous la forme de ce type de kit :

[1] Nous avons éliminé de la liste les solutions purement logicielles ainsi que les solutions purement matérielles et celles dont les composantes logicielles ne sont pas accessibles en SaaS

2.9 Critères d'évaluation de la plateforme d'expérimentation

Afin d'établir des critères de choix pour la détermination de la plateforme d'expérimentation, nous avons d'abord appliqué les critères de DiVincenzo. Ceux-ci ont été établies par David DiVincenzo pour caractériser les exigences minimales d'un calculateur quantique :

- Un système pouvant passer à l'échelle, avec des qubits bien identifiés et pouvant interagir entre eux
- Il doit être possible d'initialiser l'état initial du registre
- Il doit être doté d'un ensemble universel de portes (transformation unitaires)[1]
- Le temps de décohérence doit être bien plus long que le temps nécessaire aux opérations
- L'état du qubit doit pouvoir être mesuré

A ces critères fondamentaux, nous avons ajouté des critères spécifiques à notre expérimentation :

- La possibilité d'accéder en SaaS à la plateforme
- La disponibilité d'une plateforme matérielle accessible en SaaS
- La fourniture d'un Kit de développement sous licence Open Source
- La disponibilité au sein du Kit d'un simulateur (soit en SaaS, soit *on premise*)
- La disponibilité au sein du Kit d'un dispositif permettant d'évaluer et d'étudier le bruit quantique de la plateforme, ainsi que la fourniture d'éléments permettant la correction d'erreurs (bit-flips et phase-flips)

2.10 Plateforme quantique retenue pour l'expérimentation

A l'heure actuelle, seule la plateforme IBM Q Expérience satisfait, en plus des critères de DiVincenzo, à l'ensemble des critères définis au paragraphe précédent :

- ☑ La possibilité d'accéder en SaaS à la plateforme
- ☑ La disponibilité d'une plateforme matérielle accessible en SaaS
- ☑ La fourniture d'un Kit de développement sous licence Open Source
- ☑ La disponibilité au sein du Kit d'un simulateur (soit en SaaS, soit *on premise*)
- ☑ La disponibilité au sein du Kit d'un dispositif permettant d'évaluer et d'étudier le bruit quantique de la plateforme, ainsi que la fourniture d'éléments permettant la correction d'erreur (bit-flips et phase-flips)

[1] De fait, nous avons éliminé de la liste les ordinateurs quantiques adiabatiques (dont D-Wave), bien qu'ils satisfassent à la plupart des critères. En effet, pour notre expérimentation, nous nous concentrons sur les calculateurs à portes quantiques.

plateforme	Critères 1 (DiVincenzo)		Critères 2	
Microsoft QDK	✗	Pas encore de plateforme matérielle	✓	
D-Wave	✗	Pas de porte quantique (adiabatique)	✗	(pas encore de plateforme matérielle)
IBM Q Experience	✓		✓	
ATOS	✓		✗	Pas de caractérisation du bruit. Pas Open Source.
Rigetti	✓		✗	Pas accessible en SaaS
IonQ	✓		✗	Pas accessible en SaaS
Google	✓		✗	Pas de caractérisation du bruit
Huawei	✓		✗	SaaS pas encore accessible en France
Alibaba	✓		✗	SaaS pas encore accessible en France

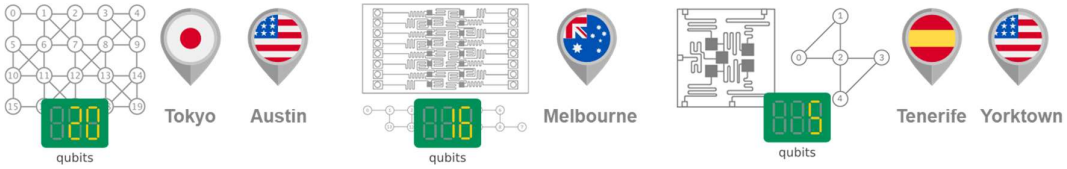
Note : les évaluations ont eu lieu en Janvier 2019. Depuis, les plateformes Rigetti et IonQ ont été ouvertes en SaaS et satisfont aux critères.

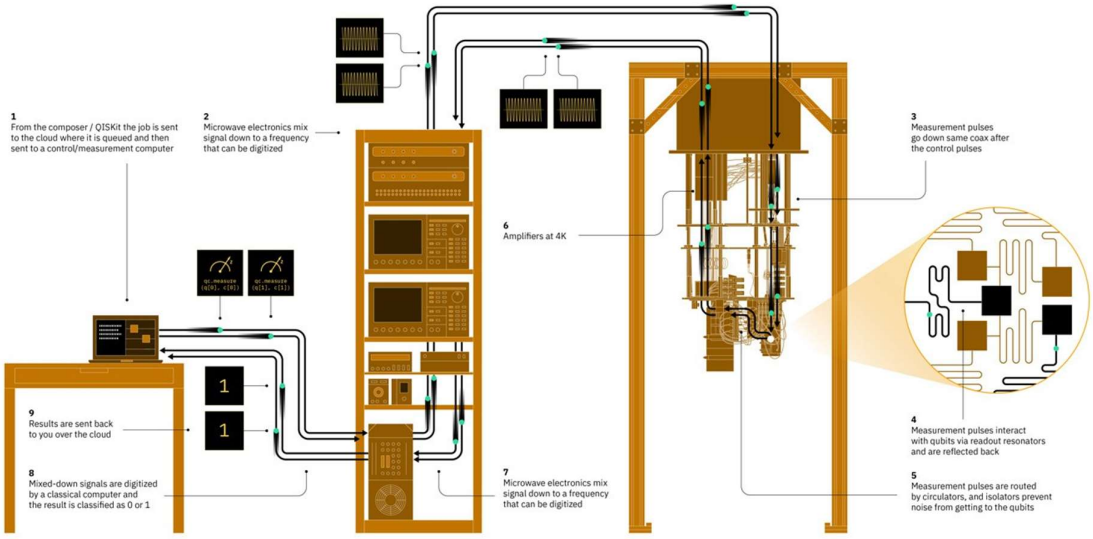
Nous avons retenu la plateforme **IBM Q Expérience** comme plateforme quantique pour notre expérimentation.

IBM Q Experience est une plateforme matérielle et logicielle, satisfaisant aux critères de DiVincenzo, qui permet aux utilisateurs d'accéder à un ensemble de **processeurs quantiques** via le **cloud**.

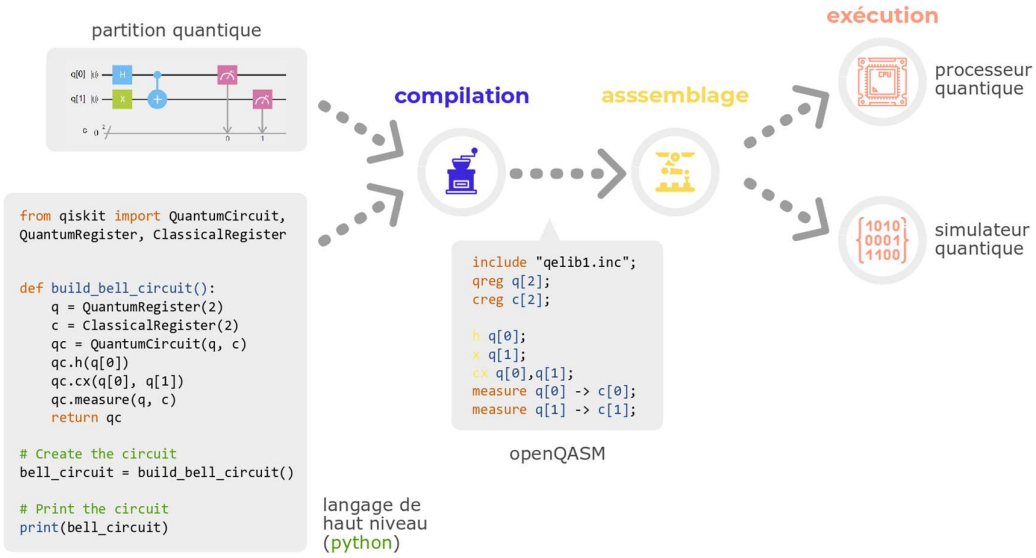
Les processeurs quantiques sont basés sur des transmons **supraconducteurs**. La solution est dotée de 2 équipements dédiés aux clients - IBM Q 20 (Tokyo, Austin) – et de 3 équipements publics : IBM Q 16 (Melbourne), IBM Q 5 (Tenerife et Yorktown).

L'ensemble est complété par un simulateur sur 32 qubits. Le tout est accessible en SaaS.





Le schéma ci-dessous illustre une vue d'ensemble de la plateforme logicielle :



openQASM (Quantum ASseMbly language) est au cœur de cette plateforme. C'est une représentation intermédiaire pour **instructions quantiques** :

- QASM est agnostique vis-à-vis du matériel
- Il a une syntaxe empruntant au langage C et à l'assembleur
- Il utilise des registres classiques et quantiques, offre un ensemble de portes quantiques basiques (U, CX, Z, ..) qu'il est possible d'étendre, permet d'initialiser des qubits, d'effectuer des mesures et de faire des opérations numérique et conditionnelles.
- La compilation s'effectue offline sur un ordinateur classique
- La génération de circuit s'effectue en ligne sur ordinateur classique, en interaction avec le processeur quantique
- L'exécution s'effectue sur processeur quantique

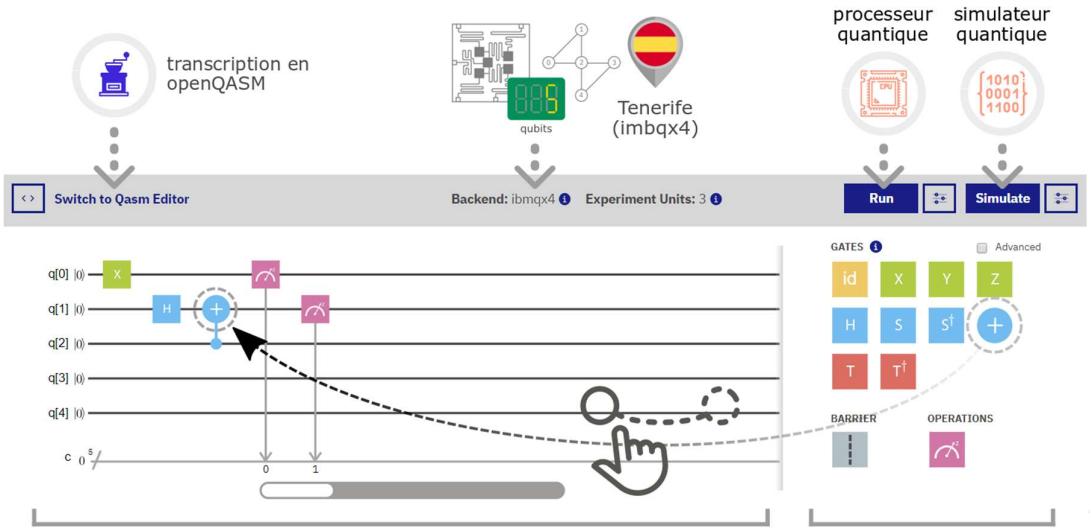
- Le post-processing (utilisation des résultats) s'effectue sur ordinateur classique

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

u3(-1.23096,0,0) q[0];
u3(pi/4,0,0) q[1];
cx q[0],q[2];
z q[2];
h q[2];
cx q[1],q[2];
z q[2];
u3(pi/4,0,0) q[1];
h q[2];
cx q[1],q[2];

measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

Exemple de code QASM pour la création d'un état $|W\rangle$

Au-dessus d'openQASM existe plusieurs couches d'abstraction et d'outillages.

Le premier outil est le « quantum composer ». Il s'agit d'un qui permet d'écrire une « partition » quantique :



Cet outil interactif permet de concevoir des circuits quantiques à partir d'un ensemble de portes quantique. Cette composition est transcrite en QASM et peut être exécutée soit sur simulateur quantique, soit sur processeur quantique.

En plus du « quantum composer », la plateforme IBM fournit le QISKit (Quantum Information Science Kit). Il s'agit d'un framework Open Source pour la programmation quantique, composé de quatre grands modules :

Terra

Terra permet la composition de programmes quantiques au niveau des circuits

Aqua

Aqua permets de construire des algorithmes quantiques pour des domaines appliqués

Aer

Aer groupe des ensembles de simulateurs pour l'étude d'algorithmes et d'applications quantiques en environnement NISQ

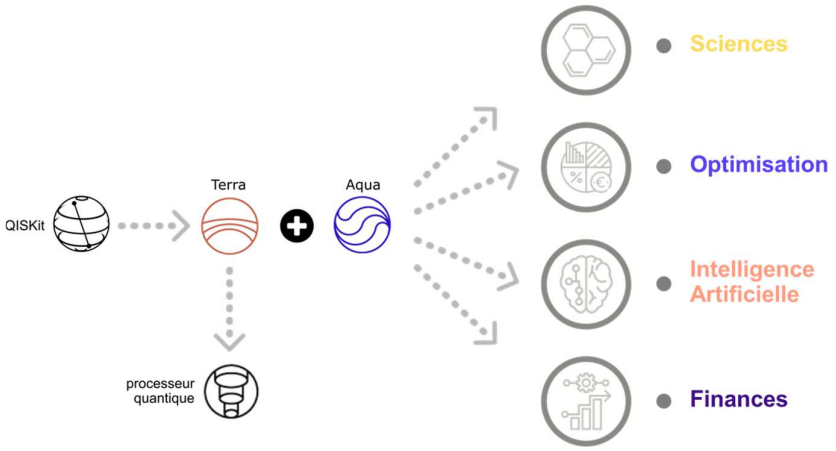
Ignis

Ignis permet de caractériser le bruit quantique, améliorer l'efficacité des portes, mesurer les temps de décohérence, …

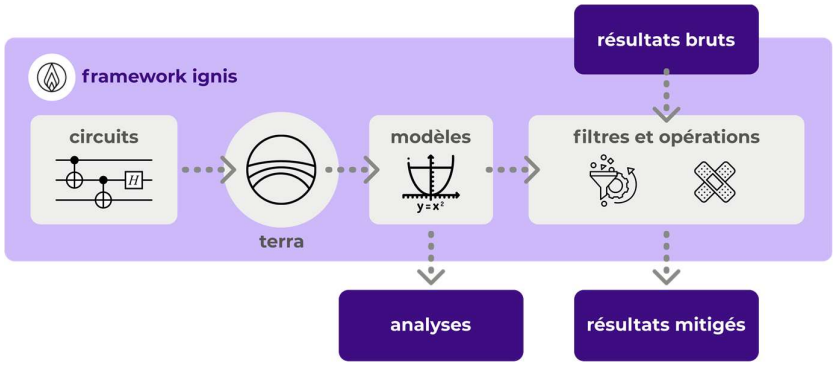
```python
from qiskit import QuantumCircuit,
QuantumRegister, ClassicalRegister


def build_bell_circuit():
    q = QuantumRegister(2)
    c = ClassicalRegister(2)
    qc = QuantumCircuit(q, c)
    qc.h(q[0])
    qc.cx(q[0], q[1])
    qc.measure(q, c)
    return qc

# Create the circuit
bell_circuit = build_bell_circuit()

# Print the circuit
print(bell_circuit)
```

Les quatre composantes seront utilisées lors des expérimentations :

- **Terra** pour les circuits quantiques et leurs exécution
- **Aer** pour la simulation
- **Aqua** pour l'écriture des algorithmes (en python) :



- **Ignis** pour la caractérisation du bruit :



03

problème 3-SAT

3 problème 3-SAT

3.1 problème SAT et complexité

3.1.1 introduction

En informatique et en mathématique, le problème **SAT** (boolean SATisfiability problem) - ou problème de satisfiabilité booléenne - est le **problème de décision**, qui, **étant donné une formule de logique propositionnelle, détermine s'il existe une assignation des variables propositionnelles qui rend la formule vraie**.

Ce problème est fondamental en théorie de la complexité. En effet, SAT est le premier problème qui s'est avéré NP-complet (théorème de Cook – Levin, 1971).

Cela signifie que tous les problèmes de la classe de complexité NP, qui inclut un large éventail de problèmes de décision et d'optimisation, sont tout aussi difficiles à résoudre que SAT.

Aucun algorithme connu ne résout efficacement chaque problème SAT, et il est généralement admis qu'un tel algorithme n'existe pas; pourtant cette croyance n'a pas été prouvée mathématiquement, et résoudre la question de savoir si SAT a un algorithme polynomial est équivalent au problème P versus NP, qui est un problème majeur toujours ouvert.

3.1.2 notations et définitions

Les **formules** de la **logique propositionnelle** sont construites à partir de **variables propositionnelles** et des **connecteurs booléens** conjonction « **et** » (∧), disjonction « **ou** » (∨) et négation « **non** » (¬).

Une formule est dite « **satisfaisable** » s'il existe une assignation des variables propositionnelles qui rend la formule logiquement vraie.

Exemples classiques :

- La formule $(p \wedge q) \vee \neg p$ est satisfaisable car si p prend la valeur faux, la formule est évaluée à vrai ;
- La formule $(p \wedge \neg p)$ n'est pas satisfaisable car aucune valeur de p ne peut rendre la formule vraie.

Un **littéral** ℓ est une variable propositionnelle v_j ou la négation d'une variable propositionnelle $\neg v_j$ (littéral négatif).

Une **clause** est un prédicat formé uniquement de la disjonction de littéraux :

$$\bigvee_{i=1}^{n} \ell_i = (\ell_1 \vee \ell_2 \vee \cdots \vee \ell_n)$$

où les ℓ_i sont des littéraux.

Une **formule** du **calcul propositionnel** est en **forme normale conjonctive** si elle s'écrit comme la conjonction de clauses.

Un **problème de décision** est dit **décidable** s'il existe un algorithme qui, après un nombre fini d'étapes, peut répondre par oui ou par non à la question posé par le problème. S'il n'existe pas de tels algorithmes, le problème est dit **indécidable**.

P est l'ensemble des problèmes de décision dont la complexité polynômiale. Il peut être décidé par une machine de Turing déterministe.

NP est l'ensemble des problèmes de décision tels que, si une solution possible est donnée, il est possible de vérifier cette solution en temps polynômial. Il peut être décidé par une machine de Turing non-déterministe.

Un problème est dit **NP-Complet** si :

- Il est dans NP
- Il au moins aussi difficile que tout problème NP
- Tous les problèmes de la classe NP se ramènent à celui-ci via une réduction polynômiale

Le problème **SAT** est un problème de décision. Il consiste à décider si une formule en forme normale conjonctive d'ordre *n* est satisfiable, c'est-à-dire s'il existe une assignation de valeurs de vérité aux variables telle que toutes les clauses soient vraies.

3.2 problème 3-SAT et applications

Il s'agit de la restriction du problème SAT aux formules qui sont des formes normales conjonctives avec au plus 3 littéraux.

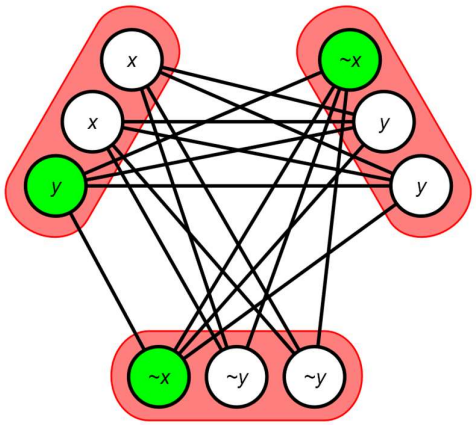
Par exemple, l'expression suivante :

$$(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6)$$

est une expression booléenne 3-SAT, composée de deux clauses, chacune ayant 3 littéraux. Le problème est ici de déterminer s'il existe des valeurs de $x_1, x_2, \cdots, x_6$ pour laquelle l'expression est vraie.

La figure suivante illustre le cas pour l'instance 3-SAT :

$$(x \vee x \vee y) \wedge (\neg x \vee \neg y \vee \neg y) \wedge (\neg x \vee y \vee y)$$



où les vertex de couleur verte correspondent à l'assignement satisfaisant $x =$ faux, $y =$ vrai.

Il fait partie des 21 problèmes NP-Complets de Karp (introduits en 1972 par Richart Karp), qui sont des problèmes difficiles de combinatoire et de théorie des graphes qui sont réductibles

entre eux. Dans la liste ci-dessous, les 21 problèmes sont organisés en indentations de façon à indiquer la direction de la réduction servant à prouver leur NP-complétude :

- **SAT** :
    - Problème de la clique et problème de l'ensemble indépendant)
    - Empaquetage d'ensemble
    - Problème de couverture par sommets (vertex)
        - Problème de couverture par ensembles
        - Feedback arc set
        - Feedback vertex set
        - Graphe hamiltonien dirigé
        - Graphe hamiltonien non dirigé
- Optimisation linéaire en nombres entiers
- **3-SAT** :
    - Coloration de graphe
        - **Partition en cliques**
        - **Couverture exacte**
        - Appariement à 3 dimensions
        - **Arbre de Steiner**
        - Ensemble intersectant
        - **Problème du sac à dos**
            - **Séquençage de tâches**
            - **Problème de partition**
            - Problème de la coupe maximum

Comme on peut ainsi le constater, 3-SAT est en liaison directe avec de nombreux problèmes industriels (tout particulièrement les sous-ensembles du problèmes du sac à dos[1]). Dans le domaine IT, de tels problèmes incluent la vérification formelle d'équivalence, la vérification de modèles, la vérification formelle de microprocesseurs ou de composants électronique, la génération automatique de patterns de test, les problèmes de planification, ou, de manière plus générale, **les problèmes de gestion de configuration comme l'assemblage à partir de composants standards** :

> Etant donné une configuration formée d'un ensemble de composants, de leurs dépendances et de conflits:

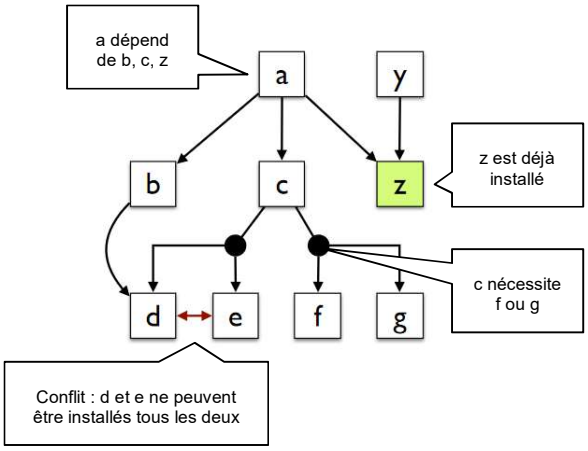
[1] Le problème du sac à dos est un problème d'optimisation combinatoire.

Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou partie d'un ensemble donné d'objets ayant chacun un poids et une valeur. Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum.

Ce problème NP-complet est à la base du premier algorithme de chiffrement asymétrique présenté en 1976 par Martin Hellman, Ralph Merkle et Whitfield Diffie (avant même RSA, proposé l'année suivante). Il est également utilisé pour modéliser des systèmes d'aide à la gestion financière, au chargement d'avion ou de bateau ou à la découpe de matériaux.

- Déterminez si un nouveau composant peut être ajouté à la configuration
- Ajoutez le composant tout en optimisant le dispositif (formalisée par une fonction linéaire)
- Si le composant ne peut pas être ajouté, recherchez le moyen pour l'ajouter en supprimant autant que possible les composants en conflit de la configuration actuelle

Pour expliciter le problème, imaginons l'ensemble de composants et de dépendances suivant :



Pour **décider** si un **composant peut être installé** dans cette **configuration**, il est nécessaire et suffisant de **résoudre** le problème **3-SAT** suivant :

$$(\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg a \vee z) \wedge$$
$$(\neg b \vee d) \wedge$$
$$(\neg c \vee d \vee e) \wedge (\neg c \vee f \vee g) \wedge$$
$$(\neg d \vee \neg e) \wedge$$
$$a \wedge z$$

3.3 algorithmes classiques

Comme 3-SAT est un problème NP-complet (et souvent utilisé comme point de départ pour démontrer que d'autres problèmes sont NP-difficiles) seuls les algorithmes ayant une complexité exponentielle dans le pire des cas sont connus.

Malgré cela, des algorithmes pour SAT ont été développés au cours des années 2000 et ont contribué à la capacité à résoudre automatiquement des problèmes comportant des dizaines de milliers de variables et des millions de contraintes (c'est à dire de clauses).

La plus évidente des méthodes pour résoudre un problème SAT ou 3-SAT est de **parcourir la table de vérité du problème**, mais la **complexité** est alors **exponentielle** par rapport au **nombre de variables**.

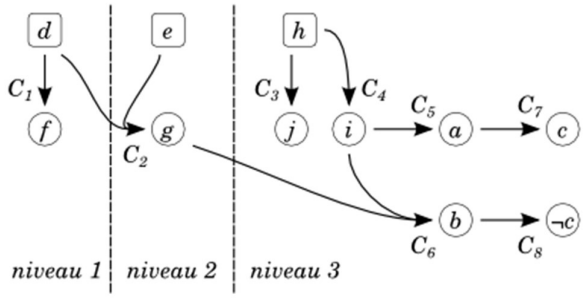
Il existe deux catégories d'algorithmes **classiques** pour résoudre ces problèmes :

- **apprentissage de clauses par conflits** (CDCL / Conflict-Driven Clause Learning): le principe des solveurs de type CDCL est d'utiliser un mécanisme pour apprendre de

nouvelles clauses en cours de recherche, puis d'utiliser au maximum les informations apprises dans l'espoir d'améliorer le temps de recherche.

Le principe est le suivant : lorsqu'un conflit apparaît lors de la recherche, c'est-à-dire lorsqu'une assignation partielle est démontrée non cohérente avec l'ensemble des clauses, on peut isoler un sous-ensemble de ces assignations et un sous-ensemble de ces clauses, qui sont responsables du conflit (les assignations ne sont pas cohérentes avec les clauses).

L'apprentissage de clause permet d'éviter de refaire plusieurs fois les mêmes erreurs dans l'arbre de recherche. De plus, la clause apprise sera en contradiction avec l'état de l'arbre de recherche. De ce fait, il sera nécessaire de réaliser un retour en arrière pour s'assurer qu'au moins un de ces littéraux soit vrai.



(illustration issue de Wikipédia)

Il existe des implémentations CDCL, qui sont en général dérivés de celui de l'algorithme de Davis–Putnam–Logemann–Loveland (DPLL). On peut citer les implémentation **Chaff**, **zChaff**, **SAT GRASP**, **Minisat** ou **ManySAT**.

- **recherche stochastique locale** : ces méthodes sont basées sur l'amélioration itérative d'une affectation des variables jusqu'à ce que toutes les contraintes soient satisfaites.

  En particulier, les algorithmes de recherche locaux modifient généralement la valeur d'une variable dans une affectation à chaque étape. La nouvelle affectation est proche de la précédente dans l'espace d'affectation, d'où le nom recherche locale.

  Tous les algorithmes de recherche stochastique locale utilisent une fonction qui évalue la qualité de l'affectation, par exemple le nombre de contraintes violées par l'affectation. Ce montant s'appelle le coût de la mission. Le but de la recherche locale est de trouver une affectation de coût minimal, ce qui est une solution, le cas échéant.

  Il existe plusieurs approches et plusieurs implémentations : la méthode « Hill Climbing », avec **GSAT** (Greedy SAT), ou par marche aléatoire, avec **WalkSAT**.

3.4 expérimentation quantique

Notre objectif est d'**évaluer expérimentalement** les technologies **quantiques** appliquées à la résolution de **problèmes 3-SAT**, comme alternative aux méthodes classiques (résolution brute, CDCL ou recherche locale).

Pour cela, nous devons lever les verrous suivants :

- Identifier l'algorithme quantique ;
- Implémenter un oracle quantique adapté à la problématique 3-SAT ;
- Implémenter l'algorithme et l'oracle quantique sur simulateur quantique ;

- Implémenter et/ou adapter l'algorithme et l'oracle quantique sur calculateur quantique en fonction des résultats et des contraintes réelles ;
- Evaluer et mesurer l'impact du faible nombre de qubits disponibles et du bruit quantique.

04

dispositif expérimental

4 dispositif expérimental

4.1 méthodologie expérimentale

Nous avons suivit la méthodologie expérimentale suivante :

Phases expérimentales	Dates

• Etude des plateformes quantique
  ○ Etablissement de critères d'évaluation
  ○ Choix de la plateforme sur la base des critères
	Janvier-Avril 2019

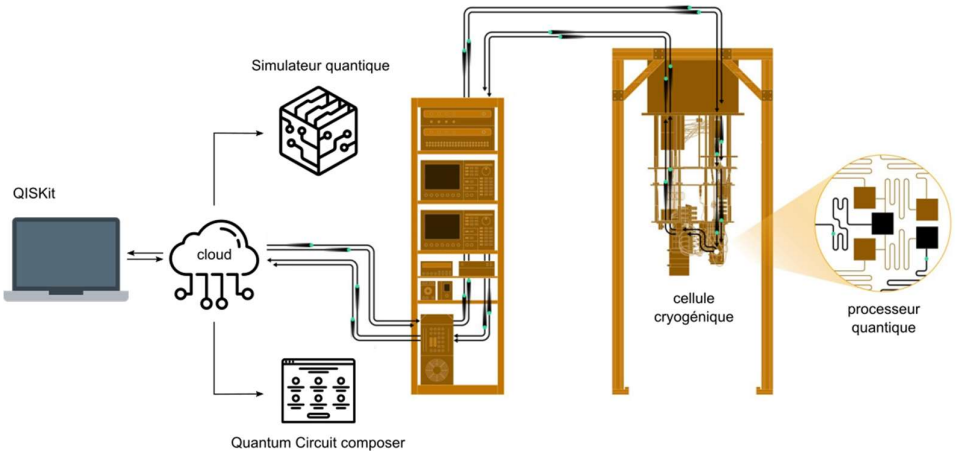
• Formalisation du problème 3-SAT	Mai 2019
• Mise en œuvre de la plateforme quantique d'expérimentation	Mai 2019

• Algorithme quantique
  ○ Identification de l'algorithme
  ○ Conception de l'oracle quantique
  ○ Etude de faisabilité de l'oracle sur « quantum composer »
  ○ Implémentation de l'algorithme quantique
  ○ Optimisation (réduction du nombre de qubits auxiliaires)
  ○ Tests sur simulateurs quantiques de l'oracle et de l'algorithme
  ○ Adaptation de l'oracle et de l'algorithme sur calculateur quantique
  ○ Instrumentation pour mesure du bruit
	Mai-Juin 2019

• Résultats expérimentaux	Juillet 2019

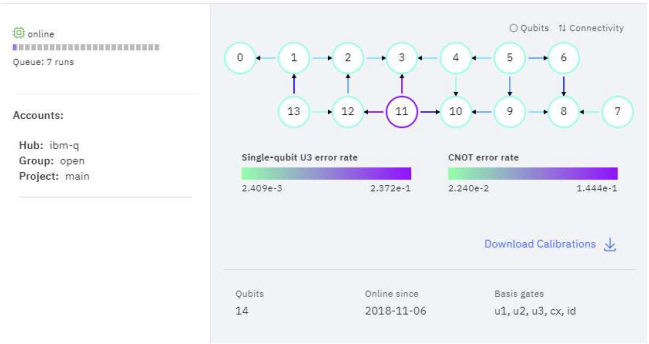
4.2 dispositif quantique

Le schéma ci-dessous illustre le dispositif expérimental mis en œuvre :

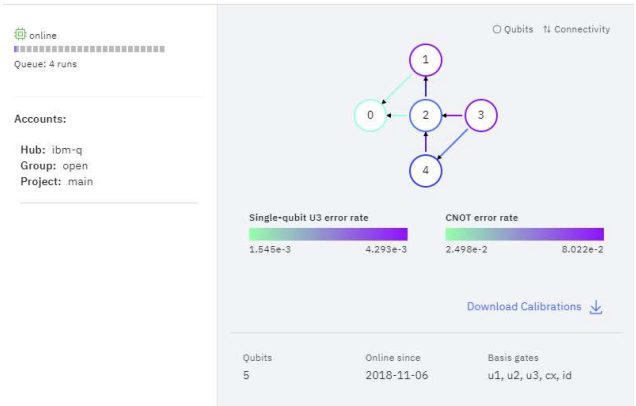


De manière plus détaillée, le dispositif est composé des éléments suivants :

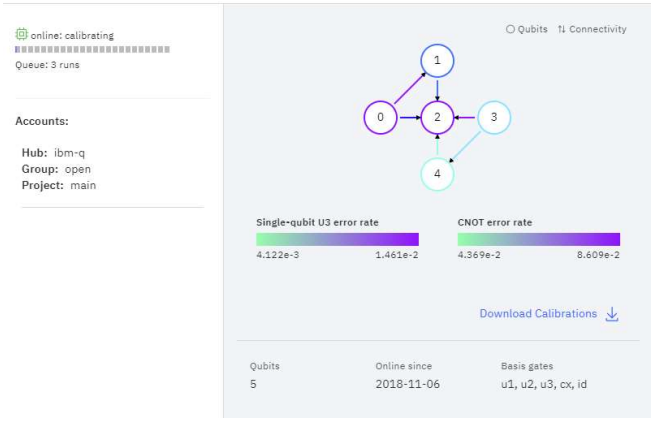
- Plateforme SaaS « **IBM Q Expérience** »:
    - IBM Q 16 Melbourne (processeur quantique **14 qubits**)



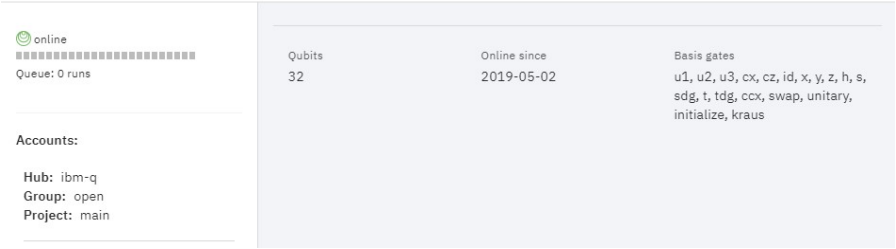
    - IBM Q 5 Tenerif (processeur quantique **5 qubits**)



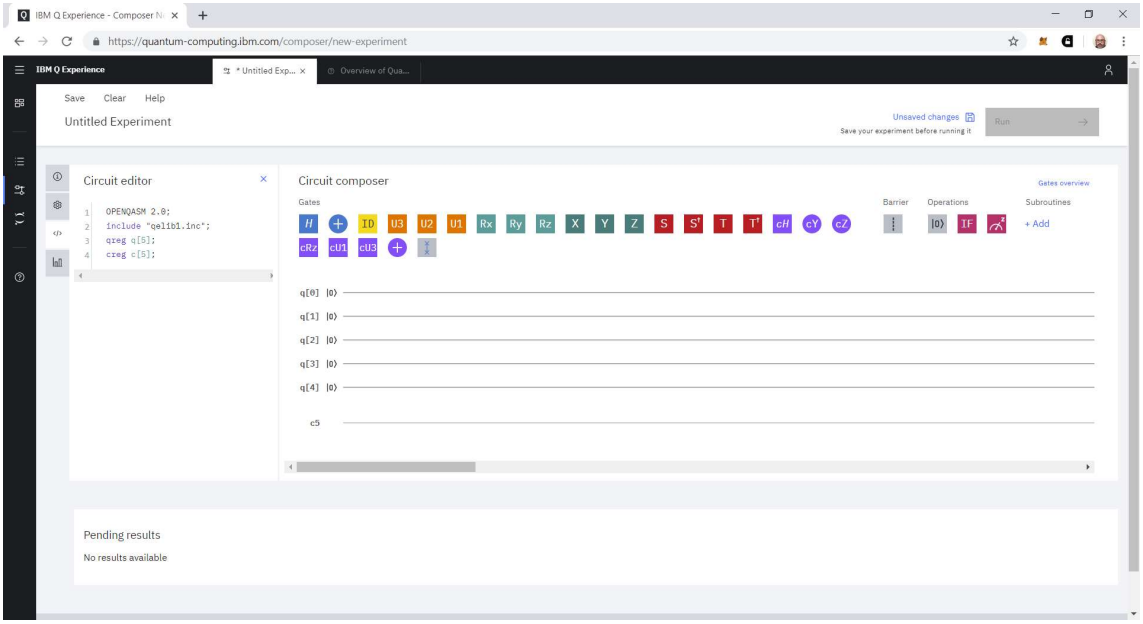
- IBM Q 5 Yorktown (processeur quantique **5 qubits**)



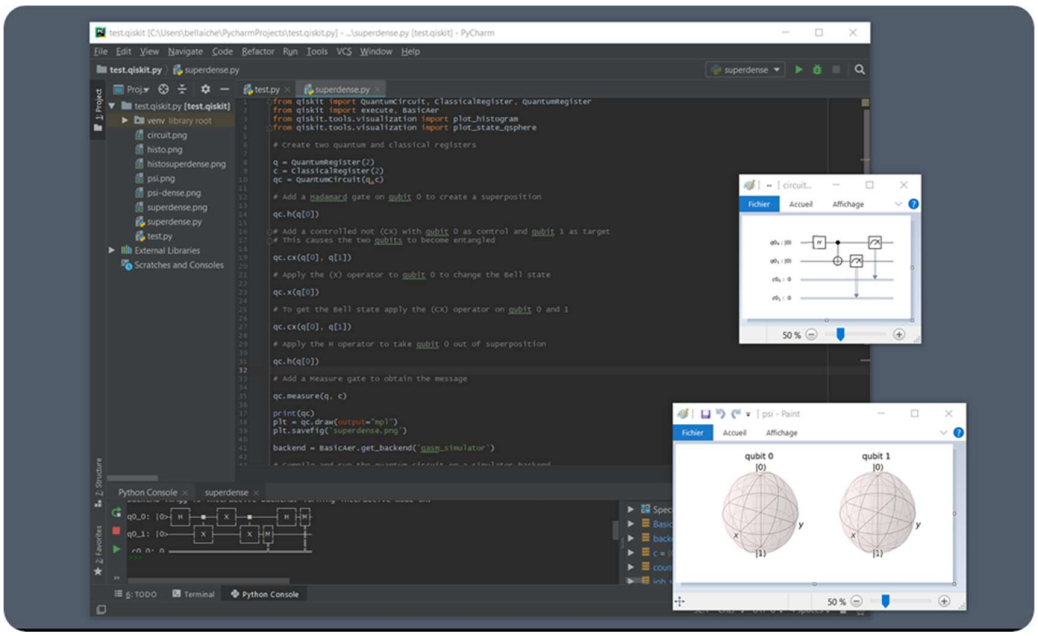
- QASM Simulator (**32 qubits**, v 0.1.547)



- Environnement en ligne « **quantum-computing.ibm.com** », et en particulier du « **quantum circuit composer** » :

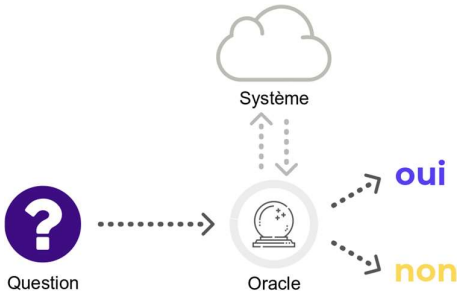


- QISKit en environnement local
  - Environnement Windows 10 64bits
  - **python** 3.7.2
  - **QISKit** 0.10.4
  - QISKit **Terra** 0.8.2
  - QISKit **Aear** 0.2.1
  - QISKit **Aqua** 0.5.1
  - QISKit **Ignis** 0.1.1
  - JetBrains PyCharm Community Edition 2018.3.6



4.3 identification de l'algorithme quantique

Un **oracle** est un dispositif fonctionnant comme une boite noire, en interaction avec le système, et capable de répondre uniquement par oui ou par non à une question posée :



On peut le considérer comme une machine abstraite capable de résoudre en temps constant O(1) un problème posé sur le système.

Les oracles sont fréquemment utilisés dans les expériences de pensée, les problèmes de calculabilité, de décision ou les problèmes cryptographiques.

Ils sont également souvent au cœur de nombreux algorithmes quantiques. On peut d'ailleurs classifier ces algorithmes selon qu'ils soient de nature oraculaire ou non :

Les algorithmes quantiques oraculaires se prêtent les mieux aux problèmes SAT. Intuitivement, on comprend bien qu'il s'agit de minimiser le nombre d'appels à un oracle quantique, répondant par « oui » ou « non » au problème de décision.

Le tableau ci-dessous dresse une liste de quelques-uns de ces algorithmes, ainsi que leurs facteurs d'accélération théorique :

Algorithme quantique oraculaire	Accélération
Deutsch-Josza	Exponentielle
Simon	Super polynomiale
Bernstein-Vazirani	Polynomiale (en direct), Super polynomiale (en récursif)
Grover	Quadratique
Sous-groupes Abéliens cachés (HSP)	Super polynomiale
Décalages cachés (Hidden Shifts)	Super polynomiale

Le principe de base utilisé par ces algorithmes est de commencer avec une superposition uniforme d'états de base, puis d'appliquer des opérations faisant en sorte que les états de base interagissent les uns avec les autres, de sorte que le module des coefficients pour certains états de base (souhaitables) augmente, ce qui implique que les autres coefficients diminution. Effectuer une mesure révélera alors avec une probabilité élevée la solution au problème en question.

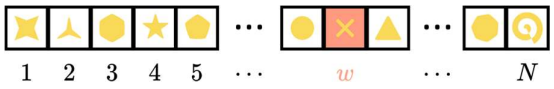
Les algorithmes basés sur l'amplification d'amplitude, comme **algorithme de Grover** se prête particulièrement bien à notre sujet d'expérimentation. [AAM2005] constate pour certains **problèmes 3-SAT** une **accélération quadratique** par rapport au algorithmes naïfs classique

(parcours d'une table de vérité par exemple), mais également par rapport aux algorithmes optimisés.

L'algorithme quantique de Grover a donc été sélectionné pour l'expérimentation. Celui-ci est présenté dans le paragraphe suivant.

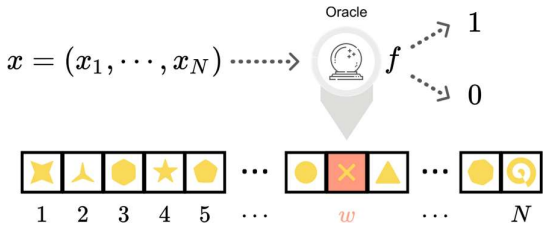
4.4 algorithme de Grover

L'algorithme de **Grover** est un algorithme de **recherche**, permettant de rechercher un ou plusieurs éléments qui répondent à un critère donné parmi N éléments non classés :



L'exemple typique d'utilisation est la recherche dans un annuaire téléphonique du nom qui correspond à un numéro de téléphone donné.

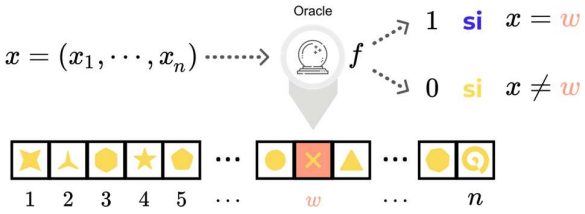
L'algorithme de Grover met en œuvre un oracle qui indique si un ensemble vérifie ou non un critère de recherche :



Effectuons une analyse classique du problème : pour trouver l'item w dans la liste en interrogeant l'oracle, il faut, au minimum, N/2 questions, et, dans le pire des cas, N questions.

Le solution classique est donc en $O(N)$. L'algorithme de **Grover** est un algorithme quantique qui permet de trouver une réponse en $O(\sqrt{N})$, soit une **accélération quadratique** par rapport à un algorithme naïf classique.

Nous allons considérer une fonction $(x_1, \cdots, x_n) \mapsto f(x_1, \cdots, x_n) \in \{0,1\}$



Nous ne connaissons pas explicitement la fonction f, mais nous pouvons interroger à notre gré l'oracle, qui donne la réponse pour toute entrée qui lui est soumise.

On peut exprimer l'oracle sous la forme d'un opérateur unitaire U_w, défini par :

$$\begin{cases} U_w|x\rangle|y\rangle = |x\rangle|\neg y\rangle & \text{si } x = w, \text{ c'est-à-dire si } f(x) = 1 \\ U_w|x\rangle|y\rangle = |x\rangle|y\rangle & \text{si } x \neq w, \text{ c'est-à-dire si } f(x) = 0 \end{cases}$$

Il peut s'écrire simplement écrit, sous la forme :

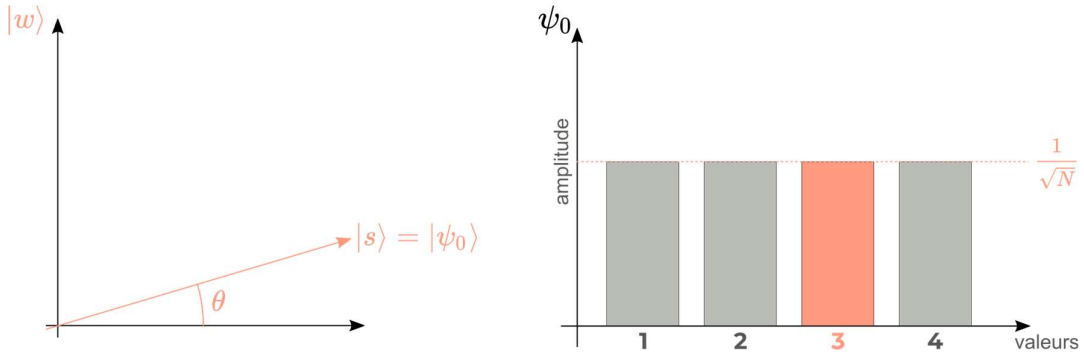
$$U_w|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$
$$U_w|x\rangle = (-1)^{f(x)}|x\rangle$$

Le principe est d'introduire en entrée une superposition d'état, sous la forme d'un état **équi-superposé** (créé en appliquant des **portes** de **Hadamard**) :

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

Si, à ce stade, nous effectuons une mesure dans la base $\{|x\rangle\}$, cette superposition s'effondrerait dans un des état de base, avec une probabilité de $1/N$.

La chance de trouver la bonne valeur de *w* est alors de $1/N$ (correspondant à une amplitude de probabilité de $1/\sqrt{N}$), et il faudrait donc ainsi N essais en moyenne :



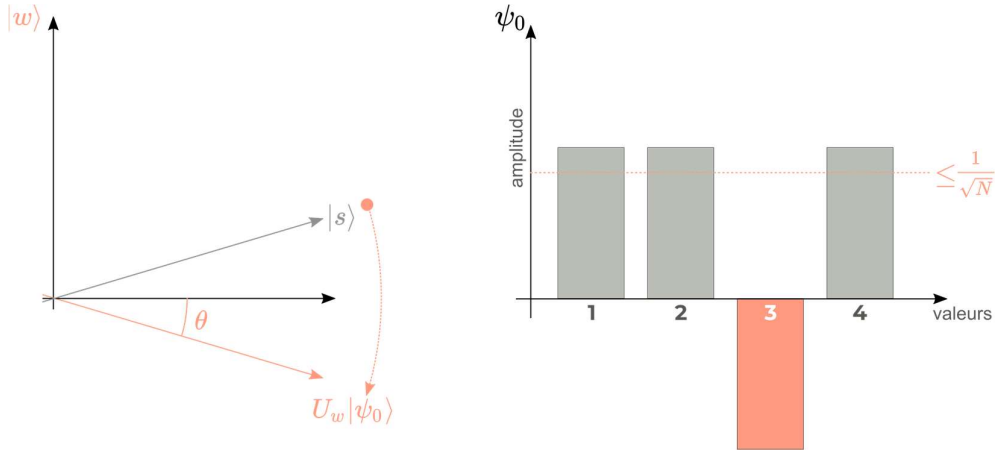
Appliquons maintenant l'**oracle** à notre état :

$$|\psi_0\rangle = |s\rangle \rightarrow U_w|\psi_0\rangle = (-1)^{f(x)}|\psi_0\rangle$$

Notons qu'il correspond à une **transformation miroir** par rapport à *w*, car :

$$U_w|w\rangle = -|w\rangle$$

L'amplitude correspondant à *w* est donc **négative**, et il s'ensuit que la **moyenne des amplitudes est diminuée** et se retrouve donc inférieure à $1/\sqrt{N}$:



Introduisant maintenant l'**opérateur de Grover** :

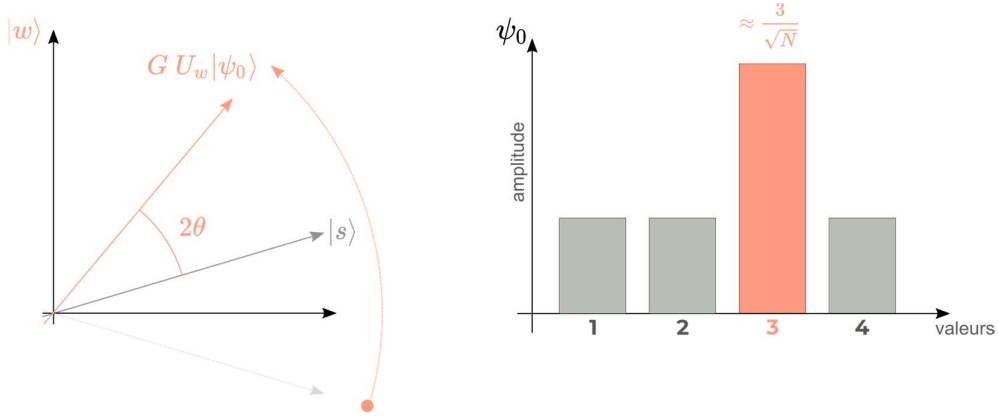
$$G_s = 2|s\rangle\langle s| - \mathrm{I}$$

En appliquant l'opérateur G à un état, chaque amplitude est transformée:

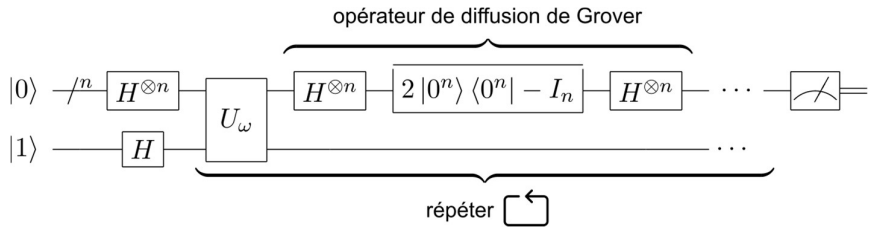
$$G|\psi\rangle = \sum_k \big(2\langle c\rangle - c_k\big)|k\rangle$$

L'opérateur G applique un **effet miroir sur les amplitudes autour de la moyenne des amplitudes**.

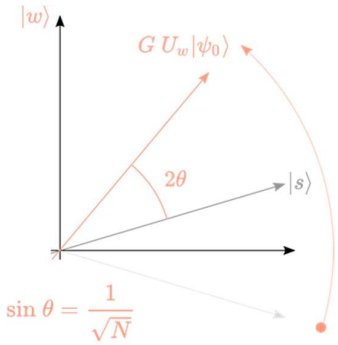
Cette nouvelle rotation rapproche l'état courant de la solution $|w\rangle$. L'amplitude correspondant à w est boostée (environ 3 fois sa valeur originelle) et les autres sont diminuées :



L'algorithme de Grover continue, en répétant les deux opérations (et en appliquant des portes de Hadamard en amont et en aval) :



A chaque itération, l'état de rapproche de la solution $|w\rangle$:



Il est alors facile d'établir géométriquement que l'algorithme prend fin à l'itération n telle que :

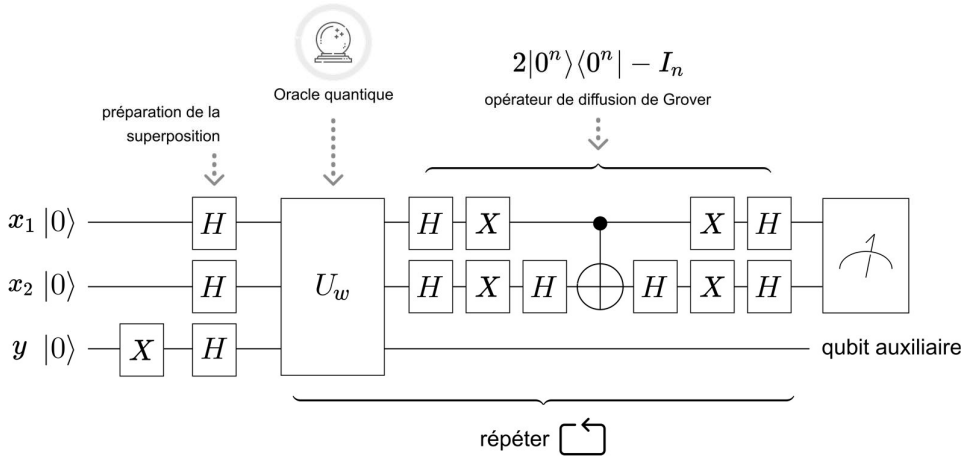
$$\sin^2\Big((2n+1)\theta\Big) = 1$$

il suffit de $\sim \sqrt{N}$ itérations pour converger

La solution quantique est donc en $O(\sqrt{N})$, soit une **accélération quadratique** par rapport à la solution classique en $O(N)$.

Le **circuit quantique** correspondant à l'algorithme de Grover est alors le suivant :

$$U_w|x_1 \cdots x_n, y\rangle = |x_1 \cdots x_n, y \oplus f(x_1 \cdots x_n)\rangle$$



4.5 oracles quantiques et problèmes 3-SAT

L'algorithme de Grover nécessite l'utilisation d'un oracle. Dans le cas du problème 3-SAT, il s'agit d'**implémenter un oracle répondant par « vrai » ou « faux » selon que la formule de logique propositionnelle soit « vraie » ou « fausse »**.

Le premier verrou à lever est donc de montrer comment transformer une formule de logique propositionnelle en termes quantiques.

Il existe en effet **plusieurs difficultés**, qui sont conséquence des **théorèmes « no-go »** (« no-cloning » et « no-deleting »).

Supposons en effet qu'il existe une opération U qui permettent le **clonage** d'un état quantique, c'est-à-dire qui permette la **copie** de l'état $|a\rangle_A$ d'un système ***A*** sur un système ***B*** (qui est initialement dans l'état t $|b\rangle_B$):

$$\forall |a\rangle_A,\ U|a\rangle_A \otimes |b\rangle_B = |a\rangle_A \otimes |a\rangle_B$$

Donc, quelque que soient les états $|a\rangle$ et $|b\rangle$, on a :

$$\begin{cases} U|a\rangle \otimes |b\rangle = |a\rangle \otimes |a\rangle \\ U|\alpha\rangle \otimes |b\rangle = |\alpha\rangle \otimes |\alpha\rangle \end{cases}$$

L'opérateur U étant unitaire, on a :

$$\begin{aligned} \left(\langle b| \otimes \langle a|U^\dagger\right)U|\alpha\rangle \otimes |b\rangle &= \left(\langle b| \otimes \langle a|U^\dagger\right)|\alpha\rangle \otimes |\alpha\rangle \\ \langle b| \otimes \langle a|\alpha\rangle \otimes |b\rangle &= \langle a| \otimes \langle a|\alpha\rangle \otimes |\alpha\rangle \\ \langle b|b\rangle \otimes \langle a|\alpha\rangle &= \langle a|\alpha\rangle^2 \\ \langle a|\alpha\rangle &= \langle a|\alpha\rangle^2 \end{aligned}$$

Ce qui n'est possible que si $|a\rangle$ et $|b\rangle$ sont égaux ou sont orthogonaux, en contradiction avec l'hypothèse et démontrant par l'absurde que **l'opération de clonage *U* n'existe pas de manière générale**.

Ce théorème, dit de « non-clonage », a été démontré pour la première fois par W. Wooters, W. Zuek et D. Dieks: **« il est impossible de copier à l'identique un état quantique arbitraire** ».

A.K. Pati et S.L. Braunstein ont également démontré de manière similaire le second théorème no-go : « étant données deux états quantiques identiques, il est impossible de supprimer une des copies ».

Si ces **théorèmes** ont des conséquences importantes, notamment en cryptographie (avec, par exemple, l'impossibilité de cloner un clef quantique), **ils rendent complexes les transcriptions quantiques d'opérations booléennes**.

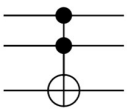
En informatique classique, n'importe quel bit de mémoire peut être activé ou désactivé à volonté. Ce n'est pas le cas en informatique quantique où toutes les opérations doivent être réversibles: l'activation ou la désactivation d'un bit entraînerait la perte des informations relatives à la valeur initiale de ce bit.

Pour cette raison, dans un algorithme quantique, il n'y a aucun moyen de placer des bits de manière déterministe dans un état prescrit spécifique à moins d'avoir accès aux bits dont l'état initial est connu à l'avance. De tels bits, dont les valeurs sont connues a priori, sont appelés bits auxiliaires (« **ancilla bits »** ou bits ancillaires). Ils interviennent en quelque sorte comme « catalyseurs » et sont jetés en fin de circuit.

Par exemple, un seul bit auxiliaire est nécessaire (et suffisant) pour créer un ensemble de portes universels (i.e. à partir de laquelle n'importe quelle logique booléenne peut être construite). Utiliser plus d'un bit auxiliaire n'est pas nécessaire mais peut simplifier le circuit.

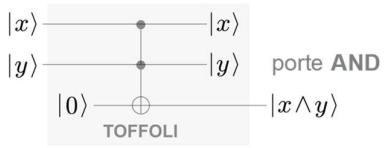
La porte de Toffoli en est un exemple. C'est une porte à 3 bits en entrée et 3 bits en sortie. Elle agit comme une porte NOT à double contrôle: si les deux premiers bits sont dans l'état $|1\rangle$ on applique une Pauli-X (NOT) sur le troisième bit, sinon rien :

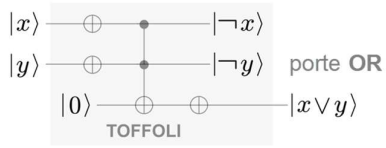
$$\text{TOFFOLI} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Toffoli est une porte universelle: n'importe quelle logique booléenne peut être construite de manière réversible à partir d'elle et de bits auxiliaires.

Les portes **NOT**, **CNOT** et **HADAMARD** forment ainsi un ensemble de **portes quantique universelles**. Par exemple :





Sur ces principes, construisons un oracle quantique pour problème 3-SAT. Imaginons la clause suivante :

$$y = (q_1 \vee \neg q_2 \vee q_3)$$

Transformons maintenant cette clause de manière à l'exprimer uniquement en termes correspondant à des portes quantiques :

$$y = q_1 \oplus \neg q_2 \oplus \neg q_3 \oplus (q_1 \wedge \neg q_2 \wedge q_3)$$

Ce qui permet d'exprimer la clause uniquement en termes de portes :

- Pauli-X (**NOT**)
- **CNOT**
- Toffoli (**CCNOT**)

Il est donc possible à partir de ce type de formulation de définir une fonction boîte noire vérifiant les solutions de la clause, ce que nous cherchions à montrer. **Le dispositif expérimental implémente ainsi sous cette forme l'oracle quantique**.

05

expérimentations et résultats

5 expérimentations et résultats

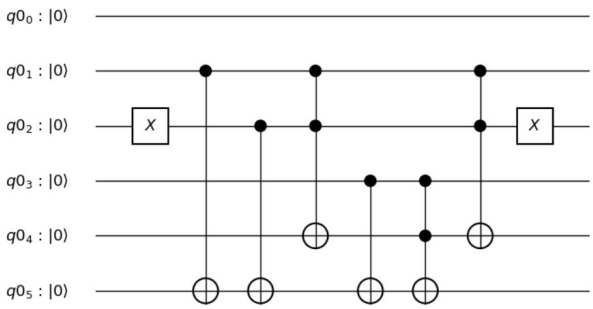
5.1 implémentation sur simulateur quantique

5.1.1 oracle quantique

Dans le précédent chapitre, nous avons démontré qu'il était possible d'exprimer un clause 3-SAT sous une forme d'opérateurs quantiques :

$$y = q_1 \oplus \neg q_2 \oplus \neg q_3 \oplus (q_1 \wedge \neg q_2 \wedge q_3)$$

Le circuit quantique correspondant est le suivant (avec la présence d'un qubit ancillaire $|q_0\rangle$ qui sera nécessaire pour la suite de l'expérimentation et l'implémentation de l'algorithme de Grover) :



Pour l'implémentation de l'oracle quantique :

- Nous écrivons sous une forme pouvant être paramétrisée l'expression 3-SAT, sous forme normale conjonctive ;
- Nous n'avons considéré que des problèmes Exactly-1 3-SAT[1], c'est-à-dire pour lesquels il existe exactement un littéral vrai (et donc 2 littéraux faux) ;
- Nous avons implémenté un algorithme « boîte noire » qui construit de manière dynamique un circuit quantique à partir de la paramétrisation de la clause Exactly-1 3-SAT.

Nous avons opéré par étapes :

- **Etapes 1** : évaluation de la méthode en implémentant de manière manuelle un circuit quantique pour l'oracle grâce au « Quantum Circuit Composer » (exécution en SaaS sur simulateur quantique) ;
- **Etapes 2** : implémentation du même circuit via QISKit et exécution en local sur simulateur quantique ;
- **Etapes 3** : implémentation de l'oracle via QISKit et exécution sur simulateur quantique.

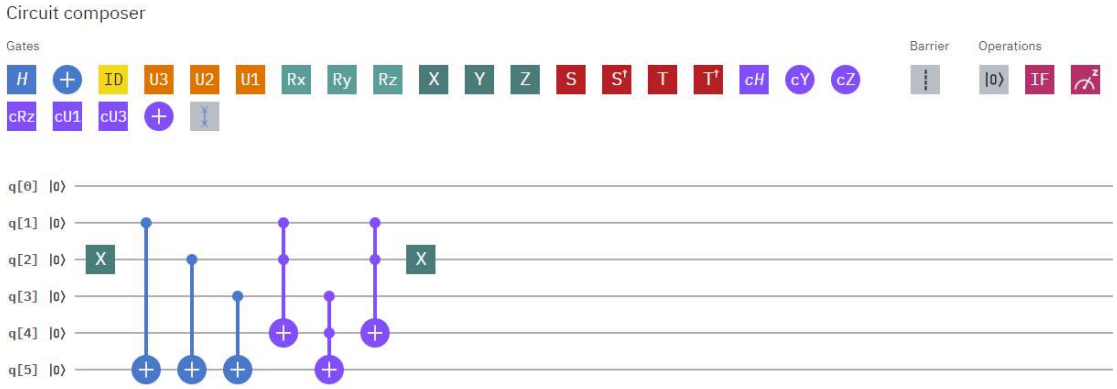
[1] Qui sont des problèmes NP-complets. L'hypothèse permet de limiter le nombre de qubits nécessaires à l'expérimentation.

5.1.1.1 étape 1 : évaluation sur compositeur quantique

Nous avons évalué la clause 3-SAT type étudiée au chapitre précédente, exprimée sous forme de portes quantiques :

$$y = q_1 \oplus \neg q_2 \oplus \neg q_3 \oplus (q_1 \wedge \neg q_2 \wedge q_3)$$

Le copie d'écran ci-après montre l'expression de cette clause via le « Circuit Composer »:



La copie d'écran ci-dessous montre la traduction du circuit quantique via le « Circuit editor » :

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[6];
5  creg c[1];
6
7  x q[2];
8  cx q[1],q[5];
9  cx q[2],q[5];
10 cx q[3],q[5];
11 ccx q[1],q[2],q[4];
12 ccx q[3],q[4],q[5];
13 ccx q[1],q[2],q[4];
14 x q[2];
```

Une grande partie de la recherche sur l'informatique quantique consiste à **déterminer comment exécuter des circuits quantiques sur des dispositifs réels**.

Dans ces dispositifs, les erreurs expérimentales et la décohérence introduisent des erreurs lors du calcul. Ainsi, pour obtenir une mise en œuvre robuste, il est essentiel de **réduire le nombre de portes** et le **temps de fonctionnement global du circuit quantique**.

Les **transpilers** introduisent une concept de gestionnaire de passes pour permettre aux utilisateurs d'**optimiser** et de **trouver le meilleur circuit quantique** pour un algorithme donné.

Ainsi, l'exécution du circuit au travers de la plateforme quantique IBM Q Expérience, permet d'exécuter le transpiler, **valider** le circuit quantique et, le cas échéant, l'**optimiser** en vue d'une exécution sur processeur quantique.

5.1.1.2 étape 2 : implémentation avec QISKit

Après cette première étape de validation, nous avons implémenté le circuit avec **QISKit**. La **clause 3-SAT** étudiée précédemment peut s'écrire de la manière suivante:

```
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister


q = QuantumRegister(6)
c = ClassicalRegister(6)
qc = QuantumCircuit(q)
qc.x(q[2])
qc.cx(q[1], q[5])
qc.cx(q[2], q[5])
qc.cx(q[3], q[5])
qc.ccx(q[1], q[2], q[4])
qc.ccx(q[3], q[4], q[5])
qc.ccx(q[1], q[2], q[4])
qc.x(q[2])

```

Ce code **QISKit** peut s'exécuter en local sur **simulateur**, sur **simulateur SaaS** et sur **processeur quantique**.

5.1.1.3 étape 3 : algorithme « boîte noire » avec QISKit

Pour implémenter l'oracle, il est nécessaire au préalable de **paramétriser la clause 3-SAT** à étudier.

Dans cet objectif, nous représentons les littéraux sous la forme d'entiers, positifs ou négatifs, afin d'indiquer une variable booléenne ou sa négation :

```
# clause 3-SAT à satisfaire, en forme
# normale conjonctive. Nous représentons les
# littéraux avec des entiers, positifs ou
# négatif, pour indiquer une variable booléenne
# ou sa négation.

exactly_1_3_sat_formula = [[1, 2, -3], [-1, -2, -3], [-1, 2, 3]]
```

A partir de cette paramétrisation et en reprenant les principes du paragraphe 4.5, il est possible d'exprimer l'oracle U_w.

Nous avons défini les conventions suivantes :

- **`f_in`** est le registre d'entrée de l'oracle U_w
- **`f_out`** est le registre de sortie de l'oracle U_w
- **`n`** est la taille du registre **`f_in`** (**`f_out`** étant un registre de 1 qubit)

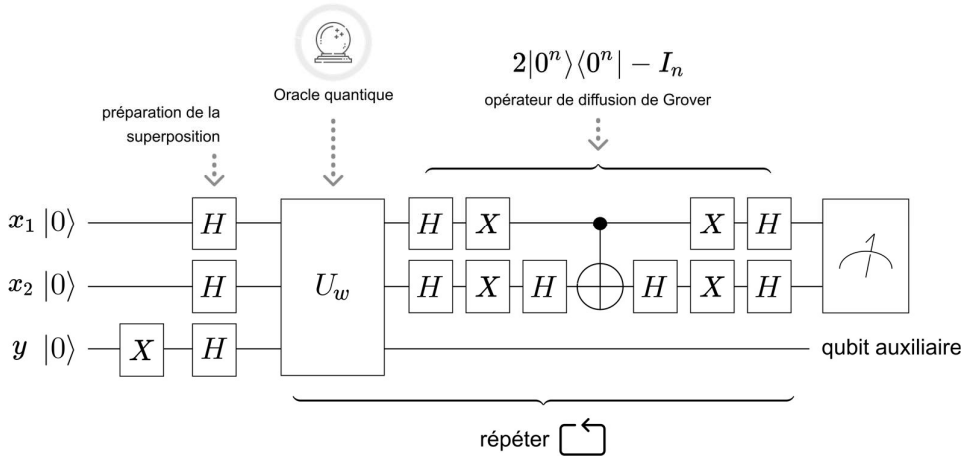
Le code ci-dessous présente sont implémentation avec QISKit[1] :

```
def oracle_u_w(circuit, f_in, f_out, aux, exactly_1_3_sat_formula):
    """ oracle
    """
    num_clauses = len(exactly_1_3_sat_formula)

    if num_clauses > 3:
        raise ValueError('Au plus 3 clauses')
    for(k, clause) in enumerate(exactly_1_3_sat_formula):
        for literal in clause:
            if literal > 0:
                circuit.cx(f_in[literal-1], aux[k])
            else:
                circuit.x(f_in[-literal-1])
                circuit.cx(f_in[-literal-1], aux[k])

        circuit.ccx(f_in[0], f_in[1], aux[num_clauses])
        circuit.ccx(f_in[2], aux[num_clauses], aux[k])

        circuit.ccx(f_in[0], f_in[1], aux[num_clauses])

        for literal in clause:
            if literal < 0:
                circuit.x(f_in[-literal-1])

    if num_clauses == 1:
        circuit.cx(aux[0], f_out[0])
    elif num_clauses == 2:
        circuit.ccx(aux[0], aux[1], f_out[0])
    elif num_clauses == 3:
        circuit.ccx(aux[0], aux[1], aux[num_clauses])
        circuit.ccx(aux[2], aux[num_clauses], f_out[0])
        circuit.ccx(aux[0], aux[1], aux[num_clauses])

    for (k, clause) in enumerate(exactly_1_3_sat_formula):
        for literal in clause:
            if literal > 0:
                circuit.cx(f_in[literal-1], aux[k])
            else:
                circuit.x(f_in[-literal-1])
                circuit.cx(f_in[-literal-1], aux[k])
        circuit.ccx(f_in[0], f_in[1], aux[num_clauses])
        circuit.ccx(f_in[2], aux[num_clauses], aux[k])
        circuit.ccx(f_in[0], f_in[1], aux[num_clauses])

        for literal in clause:
            if literal < 0:
                circuit.x(f_in[-literal-1])
```

5.1.2 algorithme quantique

La figure suivante présente l'algorithme quantique que nous avons sélectionné pour l'expérimentation :

[1] L'implémentation sur base sur [NAN2018], en particulier, l'utilisation et l'implémentation des portes CCNOT



Nous avons vu au chapitre précédent comment implémenter l'oracle quantique. Il reste donc pour finaliser l'algorithme :

- **Préparer** la superposition
- Implémenter l'**opérateur de diffusion** de Grover (amplification d'amplitude)
- **Itérer** (appel de l'Oracle et application de l'opérateur de Grover)
- Effectuer une **mesure**

Comme précédemment, nous avons opéré par étapes successives:

- **Etapes 1** : évaluation manuelle du circuit quantique à l'aide du « Quantum Circuit Composer » (exécution en SaaS sur simulateur quantique)
- **Etapes 2** : implémentation de l'ensemble du circuit quantique via QISKit et exécution sur simulateur quantique (en local ou en SaaS)

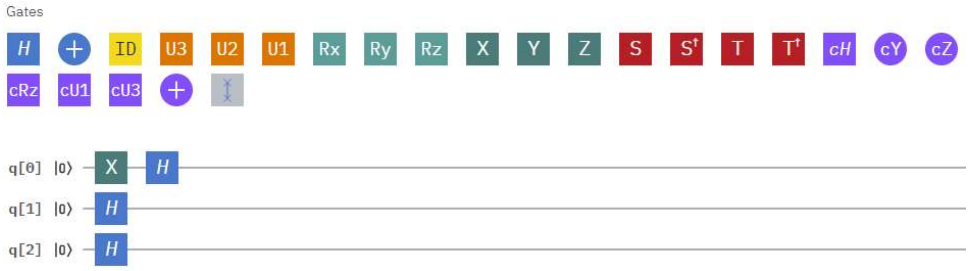
5.1.2.1 étape 1 : évaluation sur compositeur quantique

Pour créer une **superposition uniforme**, nous allons appliquer des **portes de Hadamard** à l'ensemble des qubits en entrée du circuit quantique, ces états étant initialisés au préalable à $|0\rangle_n = |0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle$.

En effet :

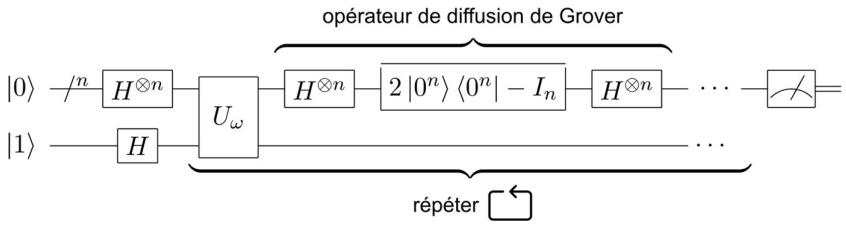
$$H^{\otimes n}|0\rangle_n = (H|0\rangle)^n = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{j\in\{0,1\}^n} |j\rangle$$

Le circuit quantique pour créer la superposition uniforme est alors très simple :

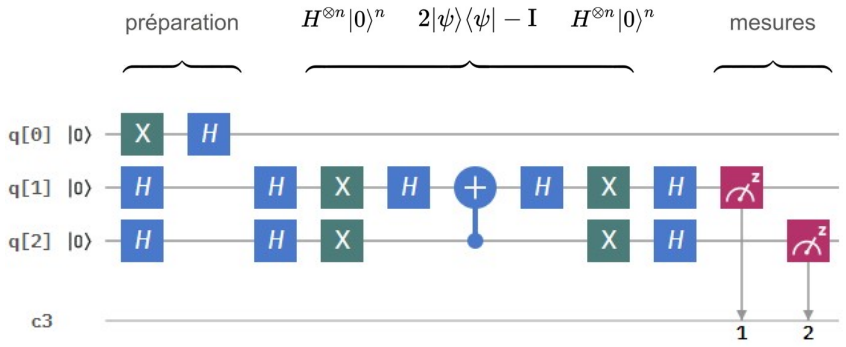


L'algorithme lui-même (inversion autour de la moyenne, voir chapitre précédent) est plus complexe.

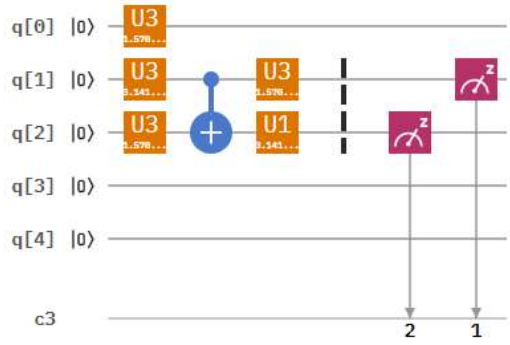
Son circuit quantique de principe est le suivant :



Celui-ci se traduit de la manière suivante sur compositeur quantique (la partie préparation étant celle décrite précédemment) :



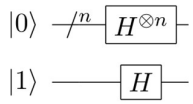
Une fois transpilé (sur ibmqx2), le circuit devient (hors oracle quantique, qui n'est pas représenté dans la figure précédente et avec une seule itération de l'opérateur de Grover):



5.1.2.2 étape 2 : implémentation avec QISKit

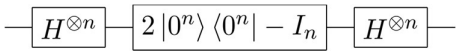
Après l'étape de validation précédente, nous avons implémenté le circuit avec **QISKit**.

La première partie est la préparation de l'état d'entrée. Dans cette optique, le code ci-dessous permet de construire l'état initial du circuit, en fonction de la taille du registre d'entrée **n** (l'état des qubits étant $|00\cdots0\rangle$) :




```python
def input_state(circuit, f_in, f_out, n):
    """ Etat initial"""
    for j in range(n):
        circuit.h(f_in[j])

    circuit.x(f_out)
    circuit.h(f_out)
```

La deuxième partie est l'implémentation de l'algorithme de Grover lui-même, et en particulier de l'inversion autour des moyennes (opérateur de diffusion):



L'opérateur de diffusion de Grover s'écrit :

$$G|\psi\rangle = \sum_k \big(\langle c\rangle - c_k\big)|k\rangle$$

où

$$\langle c\rangle = \sum_{j\in\{0,1\}^n} \frac{\alpha_j}{2^n}$$

est la moyenne. L'opération correspond ainsi à une transformation de la forme suivante :

$$\sum_{j\in\{0,1\}^n} \frac{\alpha_j}{2^n}|j\rangle_n \mapsto \sum_{j\in\{0,1\}^n} \frac{\alpha_j}{2^n}\left[2\left(\sum_{k\in\{0,1\}^n}\frac{\alpha_k}{2^n}\right)-\alpha_j\right]|j\rangle_n$$

Cette expression peut s'écrire de manière matricielle :

$$\begin{pmatrix} \frac{2}{2^n} & \frac{2}{2^n} & \cdots & \frac{2}{2^n} \\ \frac{2}{2^n} & \frac{2}{2^n} & \cdots & \frac{2}{2^n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{2^n} & \frac{2}{2^n} & \cdots & \frac{2}{2^n} \end{pmatrix} - I^{\otimes n}$$

En introduisant des portes de Hadamard, et en notant que $H^{\otimes n}H^{\otimes n} = I^{\otimes n}$, il est possible de réécrire cette matrice sous la forme simple:

$$-H^{\otimes n}\underbrace{\mathrm{diag}(\,1,1,\cdots,1\,)}_{2^n}H^{\otimes n} = -H^{\otimes n}DH^{\otimes n}$$

Notons $C^{n-1}Z$ la porte quantique (dite « porte Z multi-contrôlée ») qui applique une porte Z au qubit n si les qubits $1,\cdots,n-1$ sont $|1\rangle$. L'opérateur D peut alors s'écrire :

$$D = X^{\otimes n}(C^{n-1}Z)X^{\otimes n}$$

A. Barenco et al. [BAR2018] ont montré qu'il était possible de construire $C^{n-1}Z$ en implémentant un $C^{n-2}\mathrm{NOT}$ et une porte Z contrôlée.

La porte contrôlée s'implémente elle-même avec une porte **CCNOT** (Toffoli) et deux portes de **Hadamard**.

Le code QISKit suivant présente l'implémentation suivant ces principes d'une porte C^nZ :

```
def n_controlled_z(circuit, controls, target):
    """  n  x CZ """

    if len(controls) > 2:
        raise ValueError("Non implementé")
    elif len(controls) == 1:
        circuit.h(target)
        circuit.cx(controls[0], target)
        circuit.h(target)
    elif len(controls) == 2:
        circuit.h(target)
        circuit.ccx(controls[0], controls[1], target)
        circuit.h(target)
```

A partir de cette implémentation, la fonction d'inversion autour de la moyenne peut alors s'écrire de manière simple :

```
def inversion_about_average(circuit, f_in, n):
    """  Inversion des moyennes"""

    #  Hadamards

    for j in range(n):
        circuit.h(f_in[j])

    #  Matrice D

    for j in range(n):
        circuit.x(f_in[j])
    n_controlled_z(circuit, [f_in[j] for j in range(n-1)], f_in[n-1])
    for j in range(n):
        circuit.x(f_in[j])

    #  Hadamards

    for j in range(n):
        circuit.h(f_in[j])
```

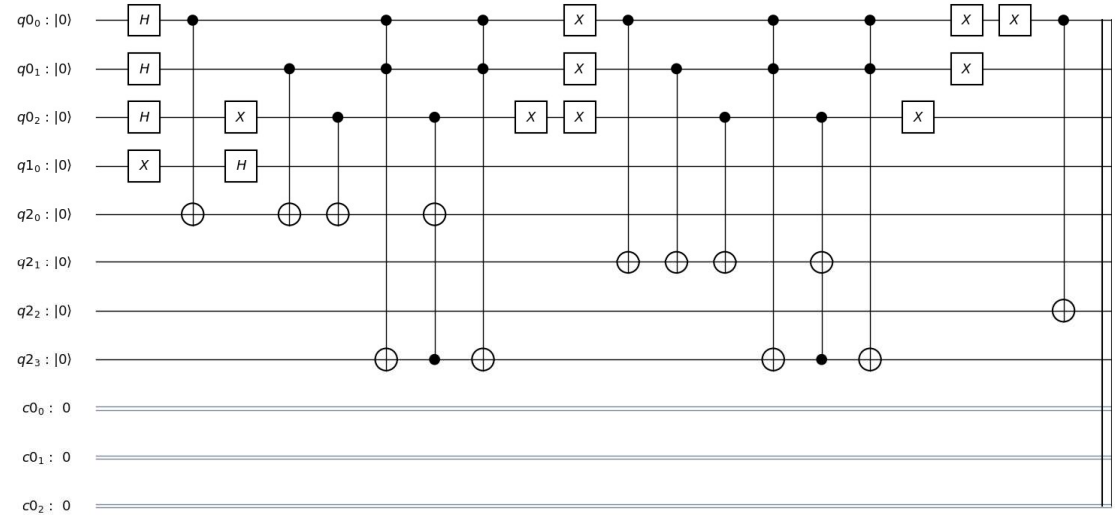
Le code ci-dessous présente l'assemblage des différentes opérations :

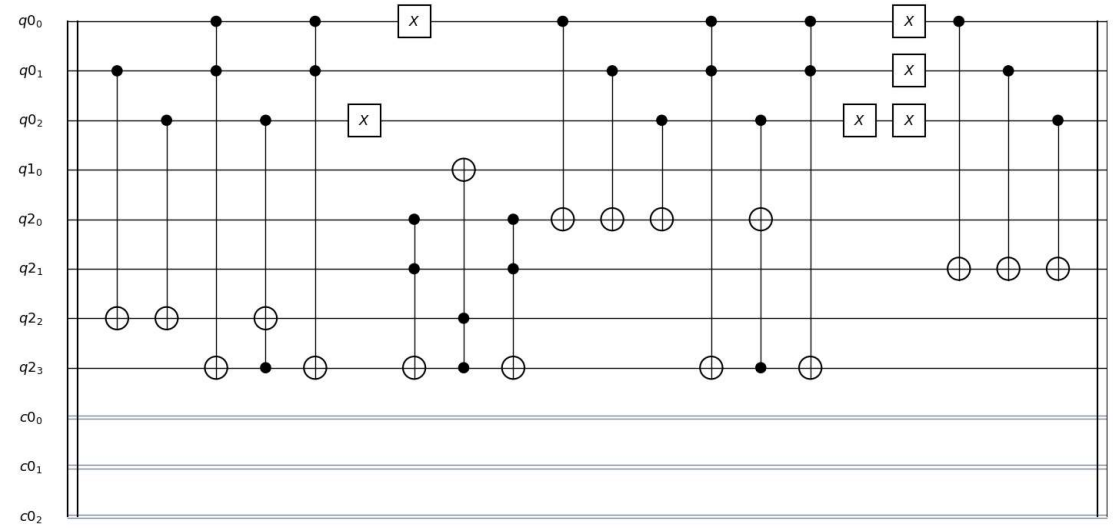
- Formalisation de la clause 3-SAT
- Définition de la longueur du registre quantique d'entrée
- Création des registres quantiques et classiques
- Création du circuit quantique
- Préparation de l'état d'entrée
- Algorithme de Grover sur 2 itérations: appels de l'oracle quantique et applications de l'opérateur de diffusion de Grover

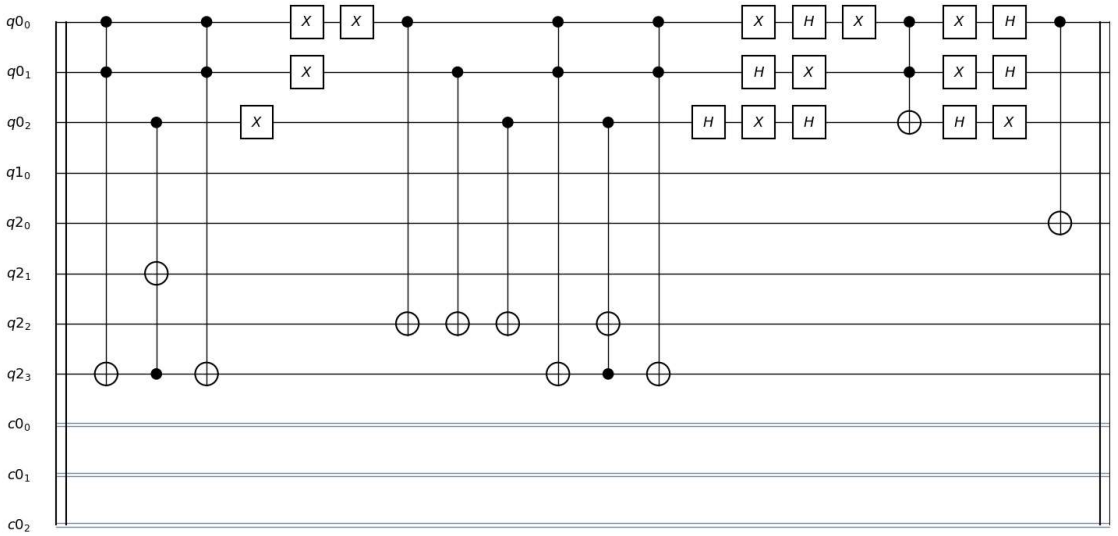
- Mesures du registre de sortie
- Création d'un backend d'exécution sur simulateur
- Exécution du job
- Visualisation des résultats

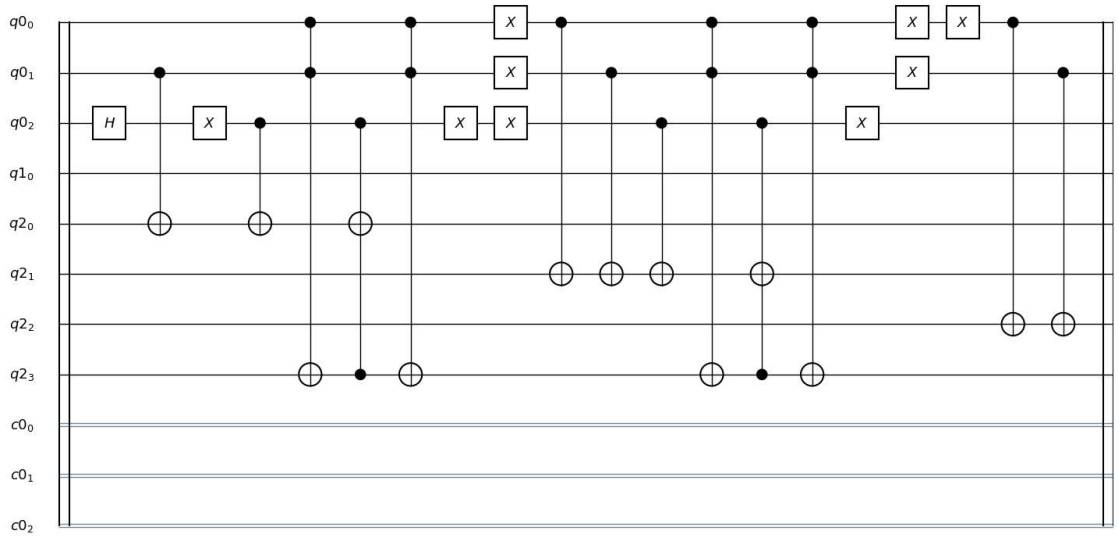
```
exactly_1_3_sat_formula = [[1, 2, -3], [-1, -2, -3], [-1, 2, 3]]

n = 3

f_in = QuantumRegister(n)
f_out = QuantumRegister(1)
aux = QuantumRegister(len(exactly_1_3_sat_formula) + 1)
ans = ClassicalRegister(n)
qc = QuantumCircuit(f_in, f_out, aux, ans, name="grover")

input_state(qc, f_in, f_out, n)

#  2 iterations

oracle_u_w(qc, f_in, f_out, aux, exactly_1_3_sat_formula)
inversion_about_average(qc, f_in, n)
oracle_u_w(qc, f_in, f_out, aux, exactly_1_3_sat_formula)
inversion_about_average(qc, f_in, n)

#  Mesure du registre de sortie

for j in range(n):
    qc.measure(f_in[j], ans[j])

#  Création d'une instance du simulateur quantique
#  et exécution

backend = Aer.get_backend('qasm_simulator')

print(qc)
plt = qc.draw(output="mpl")
plt.savefig('grover.png')

job = execute(qc, backend)

result = job.result()

counts = result.get_counts("grover")
plt = visualization.plot_histogram(counts)
plt.savefig('histo.png')
```

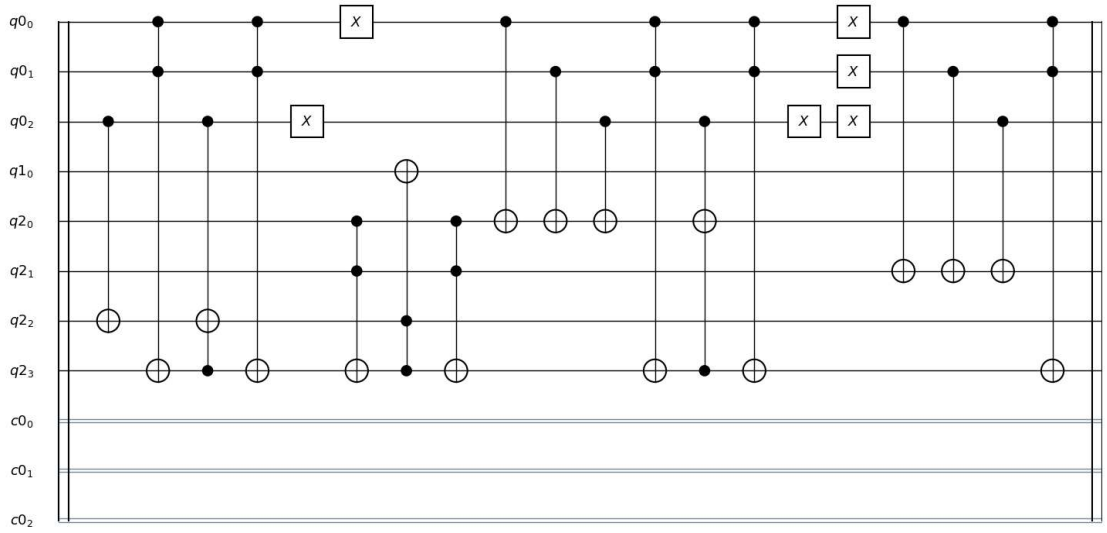
Les diagrammes suivants représentent l'ensemble des éléments du circuit quantique généré par le code QSKit. Ils correspondent à deux itérations de l'opérateur de diffusion de Grover:

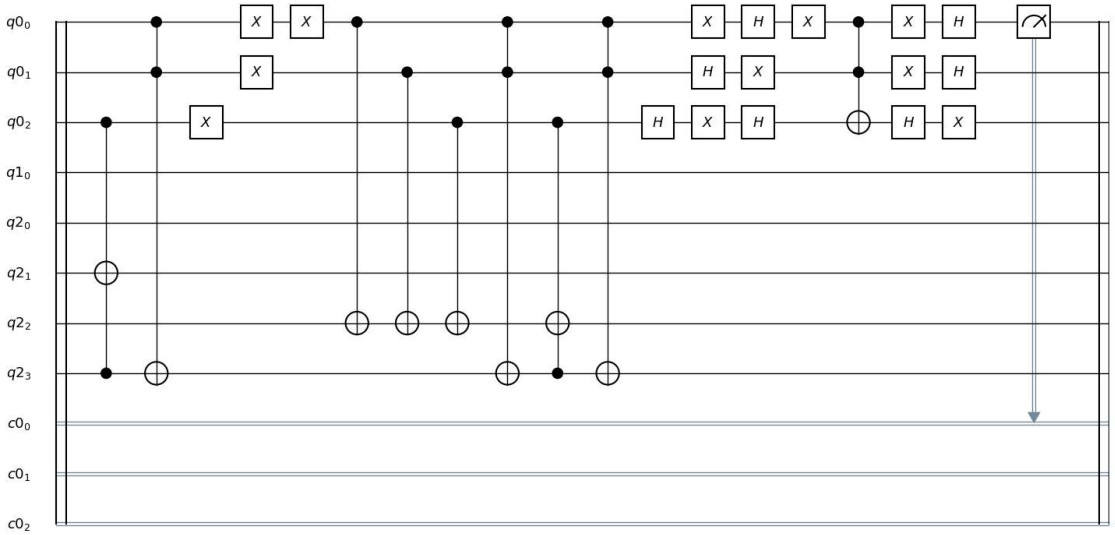

q0₀ : |0⟩
q0₁ : |0⟩
q0₂ : |0⟩
q1₀ : |0⟩
q2₀ : |0⟩
q2₁ : |0⟩
q2₂ : |0⟩
q2₃ : |0⟩
c0₀ : 0
c0₁ : 0
c0₂ : 0


q0₀
q0₁
q0₂
q1₀
q2₀
q2₁
q2₂
q2₃
c0₀
c0₁
c0₂


q0₀
q0₁
q0₂
q1₀
q2₀
q2₁
q2₂
q2₃
c0₀
c0₁
c0₂


q0₀
q0₁
q0₂
q1₀
q2₀
q2₁
q2₂
q2₃
c0₀
c0₁
c0₂
H
X

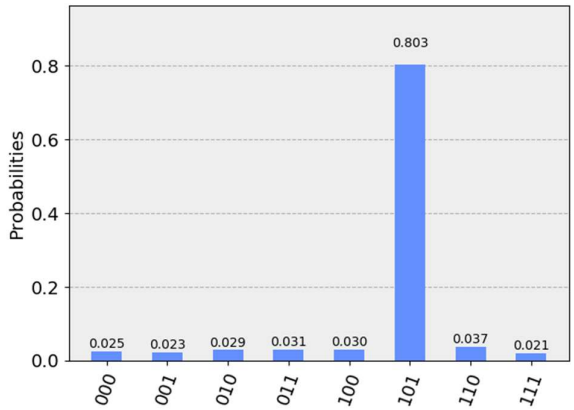

q0₀
q0₁
q0₂
q1₀
q2₀
q2₁
q2₂
q2₃
c0₀
c0₁
c0₂
X


q0₀
q0₁
q0₂
q1₀
q2₀
q2₁
q2₂
q2₃
c0₀
c0₁
c0₂
X
H

5.2 résultats expérimentaux

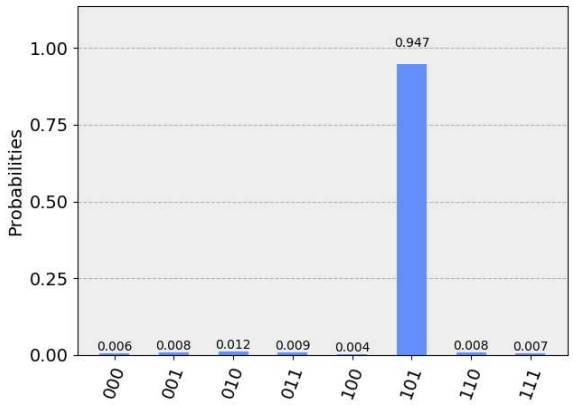
5.2.1 expérimentations sur simulateur quantique

Le figure ci-dessous présente les résultats de l'**expérimentation sur simulateur quantique**, pour **une itération** et 1024 tirs (probabilité de **80.3%** d'obtenir le bon résultat):



Résultats avec une itération

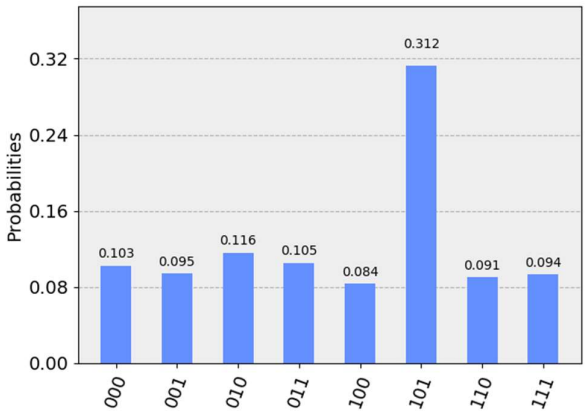
Le figure ci-dessous présente les résultats de l'expérimentation sur **simulateur quantique**, avec **deux itérations** (c'est à dire le **nombre optimal d'itérations** de l'algorithme de Grover pour ce nombre de qubits en entrée) et 1024 tirs :



Résultats avec deux itérations

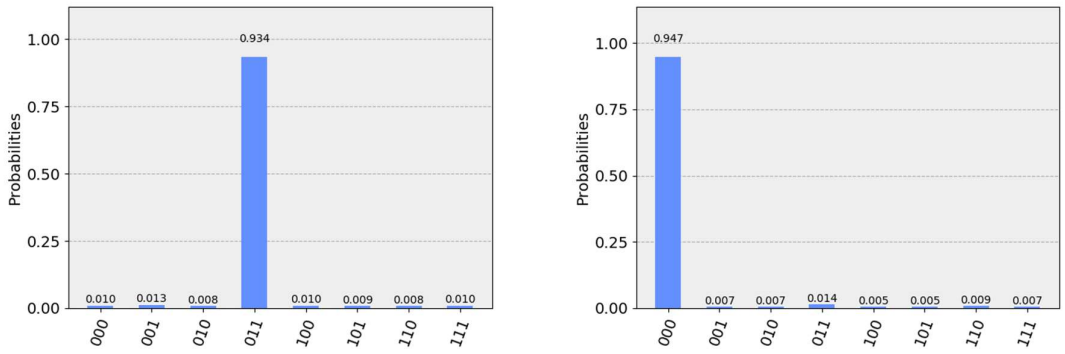
Nous constatons clairement que l'algorithme débouche sur la réponse correct ($|101\rangle$) au problème 3-SAT formalisé, avec un probabilité de trouver le bon résultat de 94.7%.

Notons qu'il est important de s'**arrêter** au nombre d'itération optimal. Ici, au-delà des 2 itérations (qui est le nombre optimal), les résultats se dégradent, avec une probabilité de trouver le bon résultat **chutant à 31.2%**:



Résultats au-delà de deux itérations

Notons également que l'algorithme débouche sur des résultats corrects indépendamment de la formalisation du problème 3-SAT (avec des probabilités supérieures à 93%) :



Résultats sur deux itérations pour différentes formulations (changement de paramètres) du problèmes 3-SAT

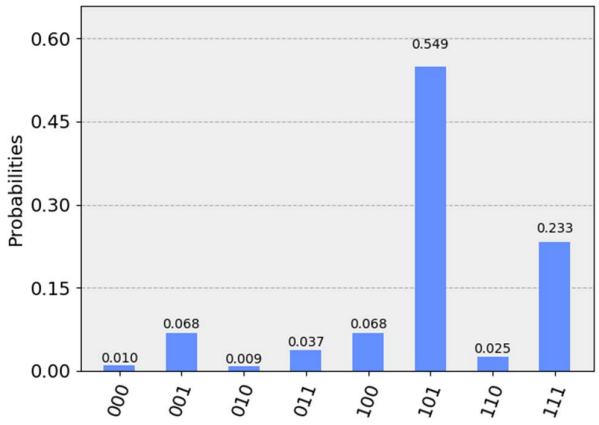
5.2.2 expérimentations avec modèle de bruit, calibration et mitigation via Ignis

L'objectif de cette étape est d'étudier des résultats plus réalistes en ajoutant au dispositif une simulation de bruit quantique.

Dans l'expérience ci-dessous, nous introduisons un modèle de bruit et effectuons une nouvelle simulation :

```
noise_model = noise.NoiseModel()
for qi in range(5):
    read_err = noise.errors.readout_error.ReadoutError([[0.75, 0.25], [0.1, 0.9]])
    noise_model.add_readout_error(read_err, [qi])

job = execute(qc, backend, shots=1024, noise_model=noise_model)
```

Le graphe ci-après présentent les résultats de la mesure, avec deux itérations et un modèle de bruit (**la probabilité d'obtenir le bon résultat chutant de 94.7% à 54,9%**):



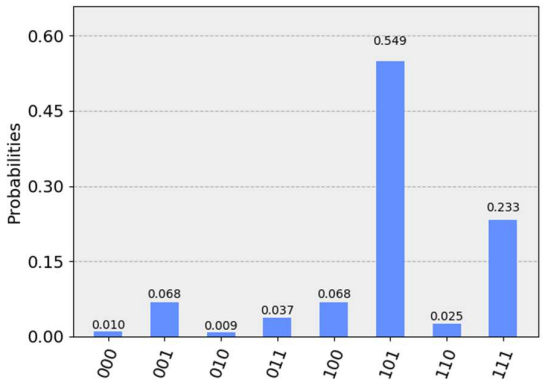
Résultats avec deux itérations un modèle de bruit

Dans l'expérience suivante (voir le code QISKit ci-dessous) :

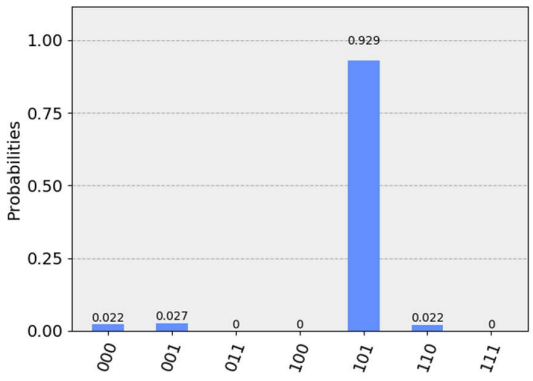
- Nous introduisons le même **modèle de bruit**
- Nous instrumentons en effectuant une **calibration** du circuit
- Nous **exécutons** l'algorithme sur deux itération et le même modèle de bruit
- Nous effectuons une **mesure** du registre de sortie
- Nous utilisons le module **Ignis** de la plateforme quantique IBM Q Experience pour mitiger les résultats
- Nous affichons les résultats mitigés pour comparaison avec les résultats bruts

```
# Modèle de bruit

noise_model = noise.NoiseModel()
for qi in range(5):
    read_err = noise.errors.readout_error.ReadoutError([[0.75, 0.25], [0.1, 0.9]])
    noise_model.add_readout_error(read_err, [qi])

f_in = QuantumRegister(n)
f_out = QuantumRegister(1)
aux = QuantumRegister(len(exactly_1_3_sat_formula) + 1)
ans = ClassicalRegister(n)

meas_cals, state_labels = complete_meas_cal(qubit_list=[0, 1, 2], qr=f_in)

# Calibration du circuit

backend = Aer.get_backend('qasm_simulator')
job = execute(meas_cals, backend=backend, shots=1000, noise_model=noise_model)
cal_results = job.result()
meas_fitter = CompleteMeasFitter(cal_results, state_labels)

# Circuit de Grover

qc = QuantumCircuit(f_in, f_out, aux, ans, name="grover")

input_state(qc, f_in, f_out, n)

#  2 iterations

oracle_u_w(qc, f_in, f_out, aux, exactly_1_3_sat_formula)
inversion_about_average(qc, f_in, n)
oracle_u_w(qc, f_in, f_out, aux, exactly_1_3_sat_formula)
inversion_about_average(qc, f_in, n)

#  Mesure du registre de sortie

for j in range(n):
    qc.measure(f_in[j], ans[j])

#  Création d'une instance du simulateur quantique
#  et exécution

backend = Aer.get_backend('qasm_simulator')

print(qc)
plt = qc.draw(output="mpl")
plt.savefig('grover.png')


job = execute(qc, backend, shots=1024, noise_model=noise_model)

result = job.result()

# Résultats sans mitigation

counts = result.get_counts("grover")
plt = visualization.plot_histogram(counts)
plt.savefig('histo_noise.png')

# Résultats avec mitigation

meas_filter = meas_fitter.filter
mitigated_counts = meas_filter.apply(counts)
plt = visualization.plot_histogram(mitigated_counts)
plt.savefig('histo_noise-mitigated.png')
```

Les graphes ci-dessous montrent les résultats **avec** et **sans mitigation** du bruit simulé. Nous constatons une amélioration très sensible du résultat, le **probabilité d'obtenir le bon résultat passant de 55% à 93%** :



Résultats avec deux itérations et bruit simulé



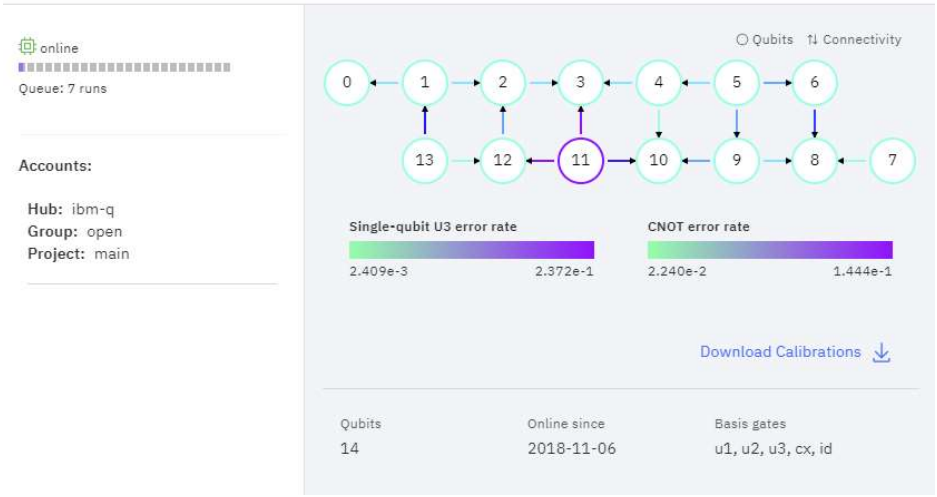
Résultats avec deux itérations et mitigation du bruit

5.2.3 expérimentations sur calculateur quantique

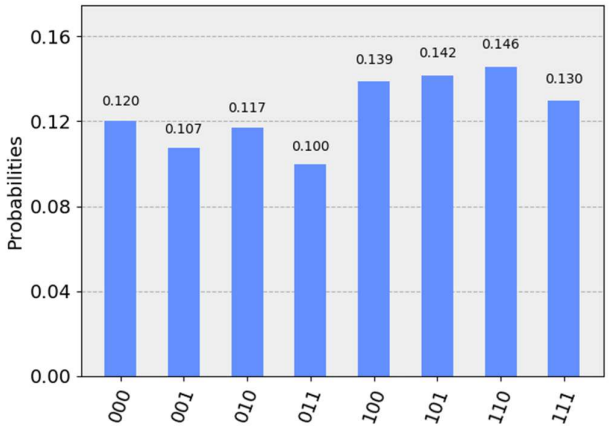
L'expérimentation QISKit ci-dessous permet la connexion avec les services SaaS de la plateforme quantique et la sélection d'un backend :

```
#  Création d'une instance en SaaS (processeur quantique)
#  et exécution

IBMQ.save_account("[redacted]",
                  'https://api.quantum-computing.ibm.com/api/Hubs/ibm-q/Groups/open/Projects/main')

IBMQ.load_accounts()


ibmq_backends = IBMQ.backends()

print("Remote backends: ", ibmq_backends)

backend = IBMQ.backends(name="ibmq_16_melbourne")[0]
print("Running on : ", backend)

job = execute(qc, backend, shots=1024)

result = job.result()
```

Nous avons sélectionné le backend **« IBM Q 16 Melbourne** » (processeur quantique avec 14 qubits, les dispositifs IBM Q5 n'ayant pas assez de qubits pour l'expérimentation) :



Le figure suivante montre le résultat de l'expérience, sans calibration du processeur quantique ni application de filtre pour réduire le bruit :



Ces résultats ne sont pas bons, mais s'expliquent facilement : les dispositifs quantiques accessibles actuellement ont une **connectivité limitée**, c'est-à-dire qu'ils permettent uniquement l'application de portes à deux qubits entre certaines paires de qubits (typiquement, des **qubits adjacents**).

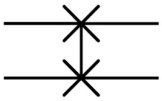
Le code de l'expérimentation suppose une connectivité totale (compatible avec une exécution sur un simulateur), ce qui simplifie grandement le code, mais n'est pas réaliste pour une exécution sur processeur quantique actuel.

Pour prendre en compte cette connectivité limitée, il est nécessaire d'adapter le circuit quantique en introduisant des portes **SWAP**.

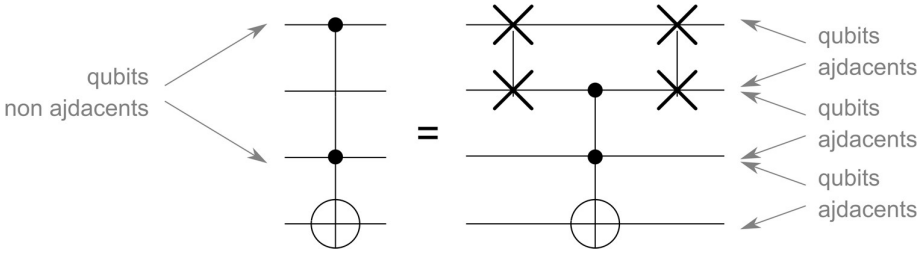
Une porte **SWAP** invertit deux qubits. Elle correspond à la transformation $|00\rangle \rightarrow |00\rangle$, $|01\rangle \rightarrow |10\rangle$, $|10\rangle \rightarrow |01\rangle$, $|11\rangle \rightarrow |11\rangle$. Dans la base $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, elle est donc représentée par la matrice:

$$\mathrm{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Sa représentation symbolique dans un circuit quantique est la suivante :



Dans notre circuit nous utilisons des portes de **Toffoli** sur des qubits non adjacents, qui sont la source du problème rencontré. Pour contourner cette difficulté, nous pouvons introduire deux portes **SWAP** supplémentaires en amont et en aval d'une porte de **Toffoli** appliquée à des qubits adjacents :



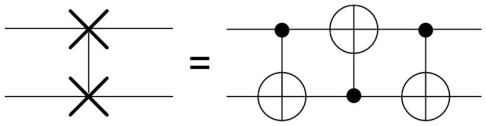
Grace à cette astuce, nous pouvons construire l'équivalent d'une porte **CCNOT** agissant sur des **qubits non adjacents**, en n'utilisant uniquement des **portes agissants sur des qubits adjacents**.

Le prix à payer est l'ajout de portes supplémentaire, et donc de bruit quantique. Le gain est de pouvoir expérimenter notre algorithme sur processeur quantique.

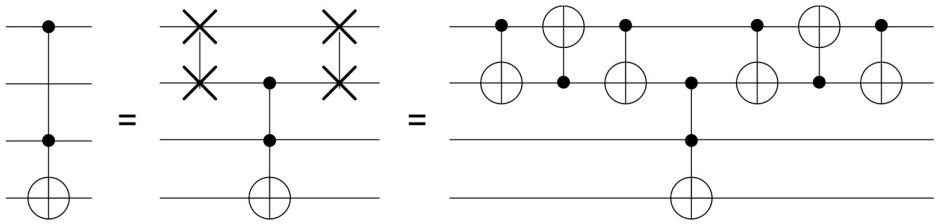
Notons qu'une porte **SWAP** peut s'écrire au moyen de portes **CNOT** :

$$\mathrm{SWAP}_{12} = \mathrm{CNOT}_{12}\ \mathrm{CNOT}_{21}\ \mathrm{CNOT}_{12}$$

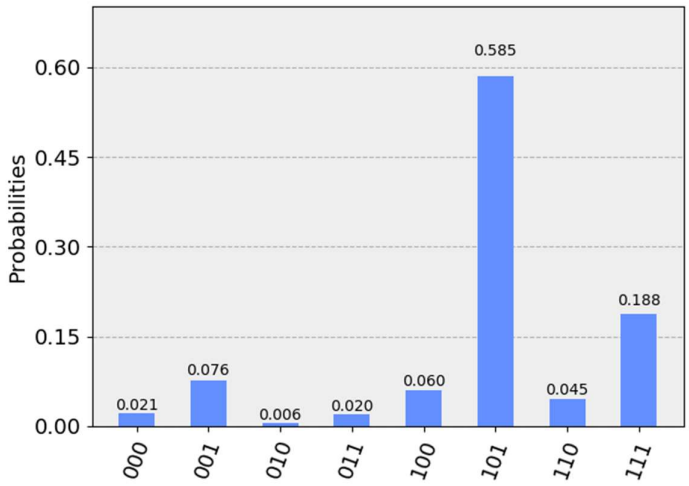
ou, de manière équivalente, en termes de circuit quantique :



Grâce à ces deux astuces, il est possible d'adapter l'expérimentation sur simulateur à un contexte d'expérimentation sur processeur quantique :



Une fois cette adaptation appliquée, le graphique suivant présente les résultats de l'expérimentation, en intégrant calibration et filtrage du bruit (run de 1024 tirs):



La probabilité trouver la bonne réponse est d'environ **59%**. Ces résultats sont **positifs**, bien que **moins bons que ceux de l'expérimentation sur simulateur quantique**. Ils s'**expliquent** néanmoins **bien** :

- Nous utilisons des **portes quantiques réelles**, donc entachées d'**erreurs** ;
- Nous n'appliquons **pas d'algorithme de correction d'erreur**[1] (mais uniquement une mitigation via Ignis à partir d'une mesure de calibration) ;
- En adaptant l'algorithme pour tenir compte des limitations de connectivité**, nous avons introduit un nombre supplémentaire important de portes** (au pire, multipliant par 6 le nombre de portes quantique utilisées) et donc augmentant le bruit, impactant la probabilité de trouver la bonne réponse.

5.3 synthèse

En général, résoudre les problèmes NP-difficiles en temps polynomial avec des ordinateurs quantiques n'est pas considéré comme possible: la plupart des chercheurs pensent que la classe de complexité BQP n'est pas égal à NP [Bennett et al., 1997].

Ce n'est pas surprenant, car montrer que BQP = NP (ou montrer qu'ils sont différents), résoudrait l'une des questions ouvertes les plus importantes en informatique, qui n'ont pas été résolues malgré les efforts des plus brillants chercheurs.

Même si nous ne pouvons pas résoudre tous les problèmes difficiles en temps polynomial sur ordinateur quantique, nous avons pu montrer et expérimenter qu'il existe en pratique des algorithmes quantiques plus rapides que n'importe quel algorithme classique connu.

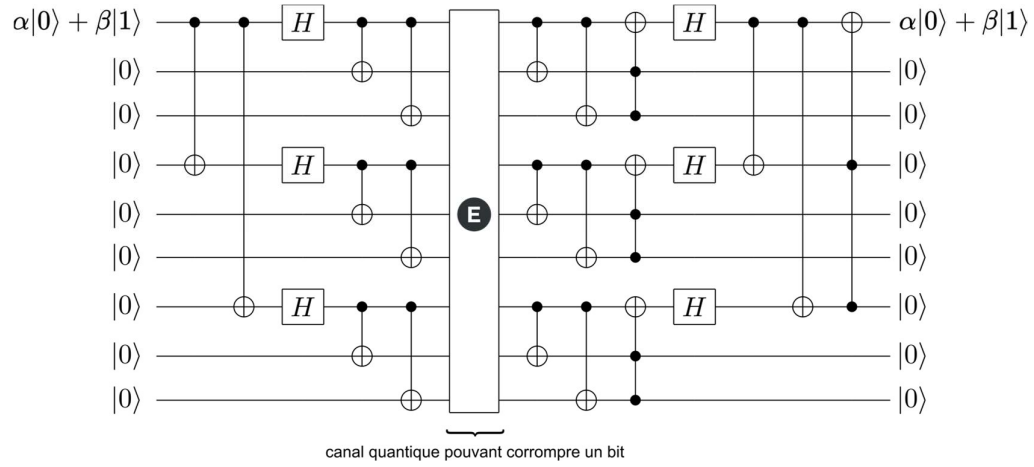
En effet, tout le circuit quantique de l'inversion autour de la moyenne s'exprime en $O(n)$ portes, conférant une accélération quadratique à notre algorithme quantique par rapport à un algorithme classique.

Si cela constitue un résultat fort, nous avons également mis en évidence la difficulté d'implémentation en pratique et les limitations des plateformes quantiques actuellement disponibles.

En effet :

[1] Ajouter un code correcteur d'erreur à l'algorithme est possible, mais nécessite d'augmenter de manière importante le nombre de qubits nécessaire.

- les dispositifs quantiques accessibles actuellement ont une connectivité limitée, ce qui contraint en pratique à adapter les circuits quantiques et introduire une nombre de portes supplémentaires important ;
- les portes supplémentaires ajoutent naturellement du bruit et diminuent d'autant la probabilité de succès ;
- d'autre part, nous ne disposons en libre accès que de 14 qubits au maximum pour exprimer notre algorithme. Or, si nous devions introduire un code correcteur d'erreurs (comme le circuit ci-dessous à 9 bits pour protéger un qubit d'un bit-flip et/ou d'un phase-flip), le nombre de qubits nécessaires serait largement plus grand (une centaine de qubits serait nécessaires) ;



- Enfin, il serait nécessaire pour aller plus loin d'étendre notre algorithme à un nombre arbitraire de clauses et de variables du problème 3-SAT à résoudre. Ceci d'une part demanderait des qubits supplémentaires, mais surtout nécessiterait plus d'itérations de l'algorithme (ici, deux seulement sont nécessaires) ;

  La longueur du circuit deviendrait alors incompatible avec le temps de décohérence des dispositifs actuellement disponibles.

Ceci étant dit, **l'expérimentation démontre clairement le faisabilité**.

Harmut Neven (Directeur du Quantum Artificial Intelligence Laboratory de Google) a quantifié au Google Quantum Spring Symposium de mai 2019 la croissance des calculateurs quantiques.

Le « **loi de Neven** » remplacerait en quelque sorte la « loi de Moore » : la croissance de la capacité de calcul des ordinateurs quantique suivrait ainsi **une loi doublement exponentielle** par rapport à l'informatique classique : $2^{2^1}, 2^{2^2}, 2^{2^3}, 2^{2^4}, \ldots$

Formulé de manière simple, la loi dit que si les processeurs quantiques se développent à un rythme exponentiel et qu'ils sont exponentiellement plus rapides que les processeurs classiques, les systèmes quantiques se développent à un rythme doublement exponentiel par rapport à leurs équivalents classiques

En traçant sur une échelle log(log()) le nombre d'opérations requises par un ordinateur classique en fonction du temps, la règle prédit ainsi une ligne droite. En extrapolant cette ligne, il existe alors un point que les ordinateurs classiques ne peuvent pas atteindre, même avec un algorithme optimisé. Ce point est appelée ***suprématie quantique***.

A ce rythme de développement, les conditions nécessaires seront donc très rapidement atteintes. IBM, par exemple, espère mettre à disposition un processeur quantique à 100 qubits l'année prochaine. Il en est de même pour Google, Rigetti ou IonQ.

Il sera alors extrêmement intéressant de poursuivre cette expérimentation sur ces nouveaux équipements, avec des résultats encore plus probants, ouvrant la porte à une nouvelle informatique qui redéfinira les pourtours de notre industrie.

06

bibliographie

6 bibliographie

Informatique quantique, théorie de l'information quantique, plateformes quantiques, algorithmes quantiques :

- [KOK2008] P. Kok et al, Linear optical quantum computing
- [CRO2017] A.W.Cross et al, Open Quantum Assembly Language
- [LER2016] Y. Leroyer et al, Introduction à l'informatique quantique
- [LOC2015] M. Loceff et al, A course in quantum computing, volume 1
- [KAY2007] Ph. Kaye et al, An introduction to quantum computing
- [TAL2018] A. Talha, Q# : a quantum programming language by Microsoft
- [COL2018] P.J. Coles et al, Quantum Algorithm Implementation, IBM
- [SEV2015] S. Gharibian, lecture on Deutsch's Algorithm
- [NIL2010] M.A. Nielsen, I.L. Chang, Quantum Computation and Quantum Information
- [LAN2017] M. Van der Lans, Quantum algorithms and their implementation on quantum computer simulators
- [MAT1996] K. Mattle et al, Dense coding in experimental quantum communication, Phys. Rev.Lett
- [HHL2009] A.W. Harrow, A. Hassidim et S. Lloyd, Quantum algorithm for linear systems of equations
- [NAN2018] G. Nannicini, An introduction to quantum computing, without the physics
- [GRA2014] B.J. Gram-Hansen, An insight into quantum walks
- [PRE2018] J. Preskill, Quantum Computing in the NISQ era and beyond
- [DER2018] D. Dervovic et al, quantum linear systems algorithms
- [WRI2015] J. Wright, Lecture on quantum information theory and Holevo's bound
- [VOL2008] S. Vogelsberger, Décohérence et intrication quantique
- [AIM2007] E. Aïmeur et al, Quantum clustering algorithms
- [IEV2016] I. Oshurko, Quantum Information and Computation
- |GRE2000] H.S. Green, Information Theory and Quantum Physics
- [ANS2018] E. R. Anschuetz et al, Variational Quantum Factoring
- [KOC2019] D. Koch et al, Introduction to Coding Quantum Algorithms
- [MOL2017] N. Moll et al, Quantum optimization using variational algorithms on near-term quantum devices
- [GUO2019] Y.Guo et al, Advances in quantum dense coding
- [CLE2015] J.R. McLean et al, The theory of variational hybrid quantum-classical algorithms
- [PER2013] A. Peruzzo et al, A variational eigenvalue solver on a quantum processor
- [JAK2009] D. Jaksch, Quantum Communication
- [BEN1993] C.H. Bennet et al, Teleporting an unkown quantum state via dual classical and Einstein-Podolsky-Rosen channels, Phys. Rev. Lett.

- [BRA2017] S. Bravyu et al, Quantum advantage with shallow circuits
- [KNI1996] E. Knill, R. Laflamme, A theory of quantum error-correcting codes
- [GOT2009] D. Gottesman, An introduction to quantum error correction and fault-tolerant quantum computation
- [DEV2013] S.J. Devitt et al, Quantum error correction
- [BAR1995] A.Brenco et al, Elementary gates for quantum computation, Phys. Ver. A

Applications (cryptographie, IA et machine learning, data science et simulations)

- [TAN2013] XTan, Introduction to quantum cryptographie
- [ETS2015] ETSI White Paper No 8, Quantum safe cryptography and security
- [GAG2017] T. Gagliardoni, Quantum security of cryptographic primitives
- [MAK2007] V. Makarov, Quantum cryptography and quantum cryptanalysis
- [BER2016] F. Bergami, Lattice-based cryptography
- [DEN2017] J-C. Deneuville, Contributions à la cryptographie post-quantique
- [AAR2019] S. Aaronson, G. N. Rothblum, Gentel measurement of quantum states and differential privacy
- [GID2019] C. Gidney, M. Ekera, How to factor 2048bits RSA integers in 8 hours using 20 million noisy qubits

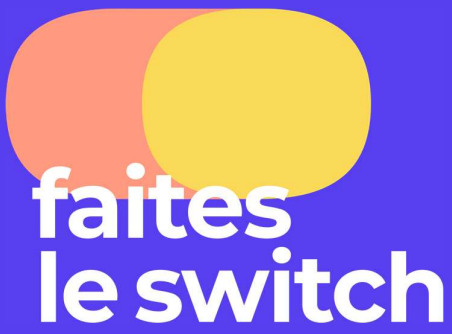
- [PER2018] A. Perdomo-Ortiz et al, Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers
- [SCH2014] M. Schuld et al, An introduction to quantum machine learning
- [WIE2014] N. Wiebe et al, Quantum nearest-neighbor algorithms for machine learning
- [WEI2018] W. Hu, comparison of two quantum nearest neighbor classifiers on IBM's quantum simulator
- [HAV2018] V. Havlicek et al, Supervised learning with quantum enhanced feature spaces
- [ANG2003] D. Anguita et al, Quantum optimization for training support vector machines
- [MOR2016] C. Moreira, A. Wichert, Quantum-like Bayesian networks for modeling decision making
- [AAR2015] S. Aaronson, Quantum Machine Learning algorithms: read the fine print, Nature
- [WIT2014] P. Wittek, Quantum Machine Learning

- [AMI2016] M.H. Amin et al, Quantum Boltzmann machine
- [LOW2014] G.H. Low et al, Quantum inference on Bayesian networks, Phys. Rev
- [SRI2017] S. Srinivasan et al, Learning Hidden Quantum Markov Models
- [MON2017] A. Montanaro, Quantum speedup for Monte Carlo methodes

- [BEN2018] M. Benedetti et al, Quantum-assisted Helmholtz machines
- [LLO2013] S. Lloyd et al, Quantum algorithms for supervised and unsupervised machine learning
- [RIS2017] D. Risté et al, Demonstration of quantum advantage in machine learning
- [ZHA2018] Z. Zhoa et al, Bayesian deep learning on quantum computer
- [REB2014] P. Rebentrost, M. Mohseni, S. Lloyd, Quantum support vector machine for big data classification
- [BER2018] D.W. Berry et al, Quantum algorithms for Hamiltonian simulation
- [KOP2018] D. Kopezyk, Quantum machine learning for data scientists

Complexité et problèmes SAT

- [AAR2008] S. Aaronson, Lecture on quantum complexity theory
- [DEU1985] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer
- [AAM2005] A. Ambainis, Quantum search algorithms
- [GIO2009] V. Giovannetti, S. Lloyd, L. Maccone, Quantum random access memory
- [SHO2000] P. Shor, AT&T Labs, Quantum Information Theory: results and open problems
- [SRI2018] K. Srinivasan et al, Efficient quantum algorithm for solving travelling salesman problem
- [PRE1998] J. Perskill, Quantum information and computation
- [WIL2016] M.M. Wilde, From classical to quantum Shannon theory
- [AAR2009] S. Aaronson, lecture on Church-Turing thesis
- [CHE2006] S-T. Cheng, M-H Tao, Quantum cooperative search algorithm for 3-SAT
- [AAR2005] S. Aaronson, NP-Complete problems and physical reality
- [SEL1996] B. Selman et al, Generating hard satisfiability problems
- [MAR2008] J. Marques-Silva, Practical applications of boolean satisfiability
- [YAN2008] W. L. Yang et al, Solution to satisfiability problem by a complete Grover search with trapped ions
- [CBA2016] C. Barron-Romero, Classical and quantum algorithms for the Boolean Satisfiability problem
- [PMA2012] P. Maandag, Solving 3-SAT
- [VDA2002] W. Van Dam et al, Quantum Algorithms for some Hidden Shift problems



econocom