Rapport de séminaire : L'algorithme quantique

RAETH Léandre, SANNA Thomas
L3 Sciences et Technologie P. Informatique

14 mars 2025

Table des matières

1 Introduction

1.1 Présentation du sujet

1.2 Exemple introductif

2 Informatique classique vs. Informatique quantique

2.1 Bits classiques

2.2 Qubits quantiques

3 Propriétés fondamentales de l'informatique quantique

3.1 Superposition

3.2 Intrication

3.3 Interférence

4 Génération de Nombre aléatoire

4.1 Problématique

Lors de la génération d'un nombre aléatoire en informatique traditionnelle, par exemple, en Python avec le module 'random()', le nombre généré est en réalité dit pseudo-aléatoire car l'algorithme de Mersenne Twister repose sur une graine pour initialiser le générateur de nombres aléatoires. Cette graine permet de prédire le prochain nombre pseudo-aléatoire, ce qui constitue un problème important pour de nombreux projets confidentiels, y compris la cryptographie.

Pour les projets impliquant une confidentialité sévère, comme la cryptographie, l'un des plus grands problèmes avec la génération de nombres pseudo-aléatoires est la capacité à produire des nombres élevés qui sont extrêmement difficiles à prédire. Par exemple, lors de la génération du sel du mot de passe afin de le hacher, un générateur de nombres aléatoires est utilisé. Si ce générateur est prévisible, un hacker compétent trouverait facilement le mot de passe en un rien de temps.

C'est pourquoi il est important de s'assurer que la graine choisie permettra un certain degré d'imprévisibilité.

En revanche, en informatique quantique, la génération de nombres aléatoires peut être véritablement aléatoire grâce aux propriétés quantiques expliquées il y a peu de temps. En effet, en utilisant un ordinateur quantique, on peut mesurer l'état d'un qubit en superposition pour obtenir un résultat aléatoire non-déterministe !

4.2 Fonctionnement

Pour comprendre comment fonctionne la génération de nombres aléatoires en informatique quantique, il faut se pencher sur les propriétés des qubits. Un qubit, comme on l'a vu plus tôt, peut être dans un état de superposition, ce qui signifie qu'il peut représenter simultanément les états 0 et 1. Lorsqu'on mesure un qubit en superposition, le résultat de la mesure est complètement aléatoire entre 0 et 1.

— **Préparation du qubit** : On commence par préparer un qubit dans un état de superposition. Cela peut être réalisé en appliquant une porte Hadamard (H) à un qubit initialement dans l'état $|0\rangle$. La porte Hadamard transforme l'état $|0\rangle$ en une superposition égale des états $|0\rangle$ et $|1\rangle$:

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

La porte d'Hadamard est extrêmement utilisée en informatique quantique Ces superpositions ne sont pas des simulations de probabilités, mais des états réels qui peuvent être mesurés.

— **Mesure du qubit** : Une fois le qubit en superposition, on procède à sa mesure. La mesure d'un qubit en superposition donne un résultat aléatoire, soit 0 soit 1, avec une probabilité de 50% pour chaque état. Ce

processus est intrinsèquement aléatoire et ne peut pas être prédit, même si l'on connaît l'état initial du qubit.
— **Génération de séquences aléatoires** : En répétant ce processus de préparation et de mesure de qubits en superposition `n` fois, on peut générer des séquences de bits aléatoires.

Ainsi, la génération de nombres aléatoires en informatique quantique repose sur les principes fondamentaux de la mécanique quantique, offrant une source de véritable aléa, contrairement aux méthodes pseudo-aléatoires utilisées en informatique classique. Nous pourrons trouver le code de l'algorithme de génération de nombres aléatoires en informatique quantique dans la section suivante.

4.3 Comparaison des méthodes

Informatique classique	**Informatique quantique**
Pseudo-aléatoire	Non déterministe
Prédictible	Imprévisible
Basé sur des algorithmes	Basé sur des propriétés quantiques
Temps de calcul rapide	Temps de calcul plus lent

TABLE 1 – Comparaison de la génération de nombres aléatoires en informatique classique et quantique

5 Expérimentation : coder un algorithme quantique

5.1 Introduction

Il existe deux manières de coder un algorithme quantique : en utilisant un simulateur quantique ou un véritable ordinateur quantique. Pour coder sur un vrai ordinateur quantique, il est possible d'utiliser des services cloud comme IBM Quantum Experience, qui permettent d'accéder à des ordinateurs quantiques en ligne. IBMQP 2025

Cependant, pour des raisons de simplicité, nous allons utiliser un simulateur quantique, qui permet de simuler un ordinateur quantique sur un ordinateur classique. Pour cela, nous allons utiliser le langage de programmation Qiskit.

En effet, Qiskit est un framework open-source développé par IBM en 2017 pour la programmation d'algorithme quantique en Python QISKIT 2025. Qiskit permettra d'utiliser cette fameuse porte Hadamard pour préparer un qubit dans un état de superposition.

5.1.1 Les schémas de circuits quantiques

Un circuit quantique est une représentation graphique d'un algorithme quantique. Il est composé de qubits, de portes quantiques et de mesures. Les portes

quantiques sont des opérations unitaires qui agissent sur un ou plusieurs qubits. Les mesures permettent de lire l'état d'un qubit.

Porte Hadamard : La porte de Hadamard, on l'a vu plus tôt, est une porte quantique qui permet de mettre un qubit dans un état de superposition. Elle est représentée par la lettre H dans un schéma de circuit quantique. Elle prend un qubit initialement dans l'état $|0\rangle$ (ou $|1\rangle$, peu importe, cela n'affectera pas la porte) et le transforme, à la sortie, en une superposition équilibrée des états $|0\rangle$ et $|1\rangle$.

FIGURE 1 – Schéma de circuit quantique pour la porte Hadamard (WIKI 2025)

Mesure : La mesure d'un qubit permet de lire son état. Elle est représentée par un symbole de balance dans un schéma de circuit quantique. La mesure d'un qubit en superposition donne un résultat aléatoire entre 0 et 1, avec une probabilité de 50% pour chaque état. Elle prend dont un qubit en entrée et renvoie un bit classique (0 ou 1) en sortie.

Information : *Les deux lignes horizontales représentent un bit classique, tandis qu'une seule ligne représente un qubit.*

FIGURE 2 – Schéma de circuit quantique pour la mesure d'un qubit (WIKI 2025)

Circuit d'un algorithme de nombre aléatoire : Ce circuit quantique décris l'algorithme de génération de nombres aléatoires en informatique quantique.

On travaille ici avec qu'un seul qubit `q` initialisé à l'état $|0\rangle$ ou $|1\rangle$ (conventionnellement, on choisit $|0\rangle$). On applique ensuite une porte Hadamard à ce qubit pour le mettre dans un état de superposition. Enfin, on mesure le qubit pour obtenir un résultat aléatoire entre 0 et 1.

Cette mesure est sortie dans la partie `c` du circuit (`c` signifie *classique*). Cette partie renvoie un bit classique, qui est le résultat de la mesure du qubit. En effet, le `1` à droite de `c` signifie le nombre de bits classiques en sortie. Le `0` à côté de la sortie de la mesure, signifie que le bit mesuré est à l'index numéro 0.

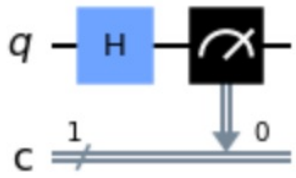

FIGURE 3 – Portes logiques pour la génération d'un nombre aléatoire entre 0 et 1 (SAP 2023)

Circuit avec 8 qubits : Le problème avec le circuit précédent est qu'il ne génère qu'un seul bit aléatoire, ce qui veut dire que le nombre aléatoire généré est soit 0 soit 1. Pour générer un nombre aléatoire plus grand, il suffit de répéter le processus de préparation et de mesure de qubits en superposition plusieurs fois, ou alors d'utiliser plusieurs qubits en superposition en même temps.

Ici, on obtient un circuit quantique pour la génération d'un nombre aléatoire avec 8 qubits. En une seule exécution, ce circuit génère un nombre aléatoire entre 0 et 255 (car $2^8 = 256$).

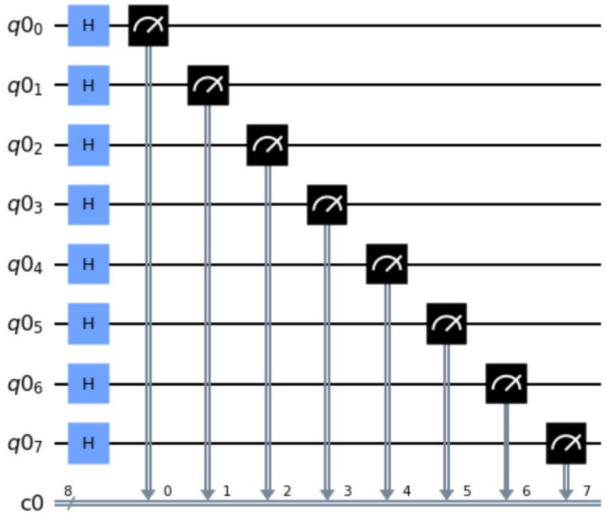

FIGURE 4 – Circuit quantique pour la génération d'un nombre aléatoire avec 8 qubits (SAP 2023)

5.2 Code de l'algorithme

Dans cette partie, il est question de coder l'algorithme de génération de nombres aléatoires avec 8 qubits en informatique quantique en utilisant Qiskit.

5.2.1 Prérequis

Il est étonnament assez simple de coder un algorithme quantique en Python avec Qiskit. En voici les prérequis :

— Installer Python : `https://www.python.org/downloads/`
— Installer Qiskit : `pip install qiskit`
— Installer Qiskit Aer : `pip install qiskit-aer` (Aer 2025)

Il est aussi préférable de coder sur Jupyter Notebook, qui permet d'exécuter du code Python en temps réel.

5.2.2 Initialisation du circuit

Le circuit quantique est initialisé avec 8 qubits en entrée et 8 bits classiques en sortie. On applique ensuite une porte Hadamard à chaque qubit pour les mettre dans un état de superposition, comme expliqué précédemment. On mesure ensuite chaque qubit pour obtenir un résultat aléatoire entre 0 et 1, stocké dans leurs bits classiques respectifs. (Par exemple, le résultat du qubit 0 est stocké dans le bit classique 0, et ainsi de suite.)

```
from qiskit import QuantumRegister, ClassicalRegister,
  QuantumCircuit
from qiskit_aer import Aer

q = QuantumRegister(8, 'q') # initialisation d'un registre
  quantique de 8 qubits
c = ClassicalRegister(8, 'c') # initialisation d'un registre
  classique de 8 bits
circuit = QuantumCircuit(q, c) # initialisation d'un circuit
  quantique avec les registres q et c

circuit.h(q[0]) # porte de Hadamard sur le premier qubit
circuit.h(q[1]) # porte de Hadamard sur le deuxieme qubit
circuit.h(q[2]) # ...
circuit.h(q[3])
circuit.h(q[4])
circuit.h(q[5])
circuit.h(q[6])
circuit.h(q[7])

circuit.measure(q, c) # mesure de tous les qubits dans les bits
  classiques correspondants
```

Listing 1 – Initialisation du circuit avec 8 qubits

5.2.3 Exécution du circuit

Pour exécuter le circuit, on utilise un simulateur quantique. Ici, on utilise le simulateur Aer de Qiskit, qui permet de simuler un ordinateur quantique sur un ordinateur classique.

À la fin de l'execution, on obtient en retour un dictionnaire contenant les résultats de la simulation, c'est-à-dire les nombres aléatoires générés.

```
simulateur = Aer.get_backend('aer_simulator') # initialisation du simulateur Aer avec le backend aer_simulator
circuitCompile = transpile(circuit, simulateur) # compilation du circuit pour le simulateur
result = simulateur.run(circuitCompile, shots=1).result() # shot=1 veut dire qu'on execute le circuit une seule fois

counts = result.get_counts(circuit)
print("\nResultats de la mesure :", counts)
```

Listing 2 – Exécution du circuit

Note additionnelle *: On peut voir dans certains codes sur internet que le simulateur s'appelle* `qasm_simulator` *au lieu de* `aer_simulator`*. En effet,* `qasm_simulator` *est une version nientôt obsolète du simulateur Aer. On conseille aujourd'hui d'utiliser* `aer_simulator` *depuis peu. (*EgrettaThula *2022)*

En retour de cette exécution, on obtient un dictionnaire qui comporte le nombre binaire généré :

```
Resultats de la mesure : {'00001010': 1}
```

Listing 3 – Résultat de la mesure : {'00001010': 1}

Ce dictionnaire contient le nombre binaire 8 bits généré, avec sa fréquence d'apparition. Il n'y a qu'une seule clé dans ce dictionnaire, qui est le nombre binaire généré, et la valeur associée est le nombre de fois que ce nombre a été généré (une seule fois car le circuit a été exécuté une fois, CF Listing 2 `"shots=1"`).

Ici, le nombre généré est 00001010, qui correspond à 10 en décimal. On peut donc dire que le nombre aléatoire généré est 10.

6 Limites et perspectives

6.1 Défis actuels

6.2 Futur de l'informatique quantique

7 Conclusion et Q&A

7.1 Récapitulatif des points clés

7.2 Session de questions-réponses

8 Bibliographie

Références

Aer (2025). *Getting started - Qiskit Aer 0.16.1 — qiskit.github.io*. https://qiskit.github.io/qiskit-aer/getting_started.html. [Accedé le 13-03-2025].

EgrettaThula (2022). *What are the differences between Qiskit's AerSimulator, QasmSimulator and StatevectorSimulator ? — quantumcomputing.stackexchange.com*. https://quantumcomputing.stackexchange.com/questions/24072/what-are-the-differences-between-qiskits-aersimulator-qasmsimulator-and-statev. [Accedé le 14-03-2025].

IBMQP (2025). *IBM Quantum Platform - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/IBM_Quantum_Platform. [Accedé le 13-03-2025].

Qiskit, IBM (2025). *Qiskit - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Qiskit. [Accedé le 13-03-2025].

SAP (2023). *True randomness with Quantum (Quantum Random number generator) — community.sap.com*. https://community.sap.com/t5/additional-blogs-by-sap/true-randomness-with-quantum-quantum-random-number-generator/ba-p/13549945. [Accedé le 13-03-2025].

Wiki (2025). *Porte quantique — Wikipédia — fr.wikipedia.org*. https://fr.wikipedia.org/wiki/Porte_quantique. [Accedé le 13-03-2025].