Note de synthèse sur l'algorithme de **Shor** et son implémentation avec Qiskit

BOUHAJA Mohammed
HAJJI-LAAMOURI Marwa

1 INTRODUCTION

Notre synthèse s'appuie sur l'article *Quantum Algorithm Implementations for Beginners*[1] visant à rendre la computation quantique accessible aux ingénieurs en simplifiant les concepts sous-jacents. Il commence par introduire le modèle mathématique, puis explore plusieurs algorithmes de base de l'algorithmique quantique, illustrant leur mise en œuvre théorique et pratique grâce à la librairie *Qiskit*. Cette dernière, une librairie Python, permet la simulation d'environnements quantiques ainsi que l'accès à des circuits quantiques réels via l'expérience IBM.

La factorisation des nombres entiers, un problème mathématique fondamental, représente également un défi de taille en cryptographie. Parmi les algorithmes traités dans l'article *Quantum Algorithm Implementations for Beginners*, l'algorithme de **Shor** se distingue. Ce dernier, un algorithme quantique, se révèle efficace pour la factorisation des entiers. Sa pertinence découle principalement de sa capacité à compromettre la sécurité des systèmes cryptographiques tels que RSA et ECC, reposant sur la difficulté exponentielle de la factorisation des grands nombres pour les algorithmes classiques.

Dans le domaine de l'informatique classique, la factorisation d'un nombre N en ses facteurs premiers requiert un temps exponentiel. Le *General Number Field Sieve* (GNFS) représente le meilleur algorithme classique connu pour cette tâche, avec une complexité de l'ordre de :

$$O\left(\exp\left((64/9)^{1/3} \cdot (\log N)^{1/3} \cdot (\log\log N)^{2/3}\right)\right).$$

En revanche, l'algorithme de **Shor** (un algorithme quantique conçu par Peter **Shor** en 1994) renverse cette perspective en offrant une solution en temps polynomial, évaluée à :

$$O\left(n^3 \log n\right),$$

où n représente le nombre de bits de N. Cette avancée quantique repose sur la recherche de périodes et exploite la capacité des ordinateurs quantiques à résoudre des problèmes très complexes pour leurs homologues classiques.

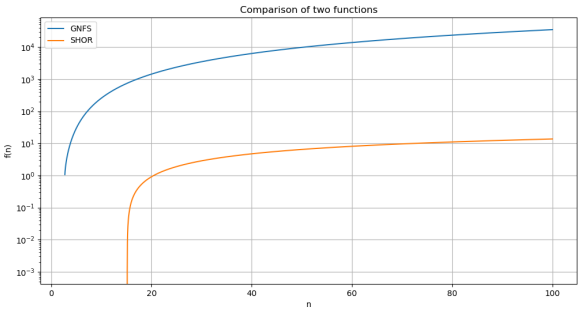


FIGURE 1 – Comparaison des complexités asymptotiques de GNFS et de l'algorithme de **Shor**.

Les premiers essais de l'algorithme de **Shor** sur des ordinateurs quantiques, notamment ceux d'IBM via Qiskit, ont déjà révélé son potentiel. Toutefois, la mise en œuvre à grande échelle de cet algorithme reste un défi, en raison des limitations actuelles des systèmes quantiques. Notons qu'à ce jour le plus grand nombre factorisé est 21.

Suivant une **démarche scientifique**, notre **problème** est la factorisation de nombres (*integer factorization problem*). Nous **supposons** que l'algorithme de **Shor** permet effectivement de résoudre ce problème en temps polynomial. L'utilisation de Qiskit permettra de vérifier la validité de notre hypothèse.

Dans un premier temps, nous aborderons la partie théorique en expliquant le fonctionnement de l'algorithme de **Shor**, en particulier la transformation de Fourier quantique et la période quantique. Ensuite, nous passerons à la partie pratique, où nous testerons l'implémentation de l'algorithme de **Shor** en utilisant Qiskit. Étant donné la difficulté d'appliquer l'algorithme de **Shor** sur de très grands nombres, cela nécessite des circuits extrêmement complexes. Par conséquent, nous nous intéresserons à de petits nombres, ce qui nous permettra de valider ou non notre hypothèse.

1. *Quantum Algorithm Implementations for Beginners : Lien vers l'article*

2 PARTIE THÉORIQUE : L'ALGORITHME DE SHOR

L'algorithme de **Shor** est un algorithme hybride qui combine une preuve classique et des notions quantiques. La partie **classique** commence par choisir un entier a et vérifier s'il est un facteur de N. Si ce n'est pas le cas, la partie **quantique** utilise la transformation de Fourier quantique pour trouver la période r de $f(x) = a^x \mod N$. De retour en classique, on vérifie si r peut être utilisé pour factoriser N. L'algorithme réduit la factorisation d'entiers à la recherche de périodes, transformant une estimation initiale en une solution plus précise.

2.1 Raisonnement derrière l'algorithme de Shor

On note N l'entier qu'on souhaite factoriser. Dans l'ensemble des étapes, nous allons effectuer des suppositions simplificatrices.

1) Pour un entier, qu'on suppose impair, $N = N_1 N_2$ donné, où $1 < N_1, N_2 < N$, on choisit un entier $k < N$ tel que $\text{pgcd}(k, N) = 1$.
2) Il est démontrable qu'il existe un exposant $p > 0$ tel que $k^p \equiv 1 \pmod N$.
3) En supposant que p est le plus petit tel exposant, si p est pair, alors, par la définition de l'opération modulo, N divise $k^p - 1 = (k^{p/2} - 1)(k^{p/2} + 1)$.
4) Puisque la différence entre $n_1 = k^{p/2} + 1$ et $n_2 = k^{p/2} - 1$ est de 2, n_1 et n_2 n'ont aucun facteur commun supérieur à 2.
5) De plus, les deux nombres sont non nuls en raison de la minimalité de p.
6) Puisque $N = N_1 N_2$ est supposé être impair, alors N_1 est un facteur de soit n_1 ou n_2 exclusivement.
7) Supposons que N_1 soit un facteur de n_1. Puisque N_1 est également un facteur de N, alors N_1 divise à la fois n_1 et N et on peut trouver N_1 en calculant $\text{pgcd}(n_1, N)$.
8) Ainsi, si l'on peut calculer un tel p, on peut trouver efficacement les facteurs de N car pgcd peut être calculé en temps polynomial (grâce à l'algorithme d'Euclide).

On peut conclure que le point central de l'algorithme réside dans le calcul de p. Dans la section suivante, nous examinerons en détail comment les propriétés quantiques entrent en jeu pour réduire considérablement la complexité de ce calcul par rapport à l'approche classique. Cette fois-ci, la pierre angulaire sera la Transformée de Fourier Quantique.

2.2 Calcul de la période p

Afin de calculer p, il nous faut considérer la période de la suite suivante :

$$A = (a_i)_i, \text{ où } a_i = k^i \mod N.$$

Si on suit la démarche classique, ce calcul n'est pas nécessairement économique en ressources. C'est ici qu'intervient la *QFT (Transformée de Fourier Quantique)* qui permet de trouver la période d'une entrée périodique en temps polynomial.

TRANSFORMÉE DE FOURIER QUANTIQUE

De manière générale, mathématiquement, la TFQ agit sur un état quantique représenté par un vecteur $|\psi\rangle$ de dimension M, où M est une puissance de deux. Si $|\psi\rangle$ est représenté en notation vectorielle comme $(a_0, a_1, \ldots, a_{M-1})$, la TFQ transforme cet état en un nouveau vecteur $|\phi\rangle$ tel que :

$$|\phi\rangle = \frac{1}{\sqrt{M}} \sum_{t=0}^{M-1} e^{2\pi i j t/M} |k\rangle,$$

où $|k\rangle$ représente la base de calcul quantique standard, i est l'unité imaginaire et $1/\sqrt{M}$ est un coefficient de normalisation. Cette formule décrit comment chaque coefficient a_j du vecteur d'entrée est transformé en une somme de termes, chacun pondéré par une exponentielle complexe en fonction de j et k.

La TFQ peut être interprétée comme une généralisation quantique de la transformée de Fourier discrète classique (TFD), où les calculs sont effectués sur des qubits au lieu de bits classiques. Contrairement à la TFD, qui traite des amplitudes réelles, la TFQ manipule des amplitudes complexes, permettant ainsi une représentation plus riche des données quantiques.

Pour rappel, dans le contexte de la factorisation entière, la TFQ est utilisée pour trouver la période de la suite $A = (a_i)_i$, comme mentionné plus haut.

Utilisation de la TFQ dans l'algorithme de Shor

Le circuit quantique pour la recherche de périodes est illustré ci-dessous :

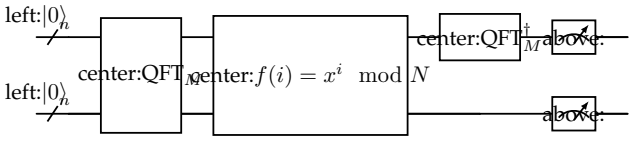


FIGURE 2 – Illustration du circuit de recherche de période.

Voici le déroulement de l'algorithme :

1. Initialisation

— Un registre A de $m = 2n$ qubits est initialisé dans une superposition égale de tous les états $|i\rangle$ (d'où l'utilisation des portes d'Hadamard) .
— Un registre B de n qubits est initialisé à $|0\rangle$.
— La première TFQ est appliquée sur le registre A pour créer la superposition :

$$\frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |i, 0\rangle .$$

2. Application de la fonction

Ensuite, une opération de modular exponentiation est appliquée pour calculer $f(i) = x^i \mod N$ sur le registre B, donnant :

$$\frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |i, f(i)\rangle .$$

3. Mesure

Le registre B est mesuré, ce qui projette l'état sur un état particulier $|s\rangle$, produisant :

$$\frac{1}{\sqrt{M/r}} \sum_{i=0, f(i)=s}^{M-1} |i, s\rangle .$$

4. TFQ

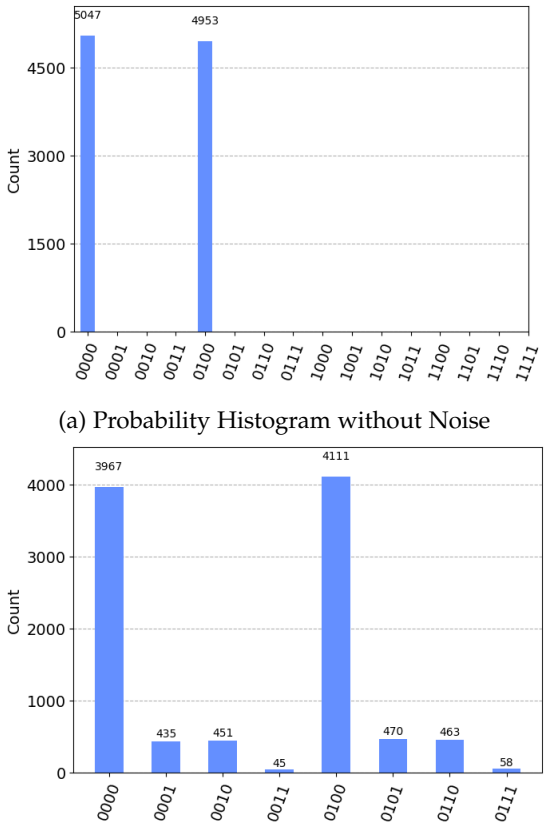
La TFQ est appliquée sur le registre A, qui transforme l'état en :

$$\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |k(M/r)\rangle .$$

Cela produit des pics de probabilité aux multiples de M/r.

5. Post-Processing

Finalement, une mesure est effectuée sur le registre A. Le résultat sera un multiple de M/r. Ce résultat est ensuite utilisé dans un calcul classique pour trouver le facteur premier souhaité.



(a) Probability Histogram without Noise

(b) Probability Histogram with Noise

FIGURE 4 – Les résulats des mesures avec et sans bruit

3 RÉSULTATS (IMPLÉMENTATION AVEC QISKIT)

3.1 Étude de l'Algorithme de Shor pour $N = 15$

Pour la première implémentation avec Qiskit, nous avons tenté de reproduire les résultats figurant dans l'article. Pour cela, nous avons reconstruit le circuit de **Shor** pour $N = 15$ et $x = 11$ avec 10000 exécutions, ou "shots".

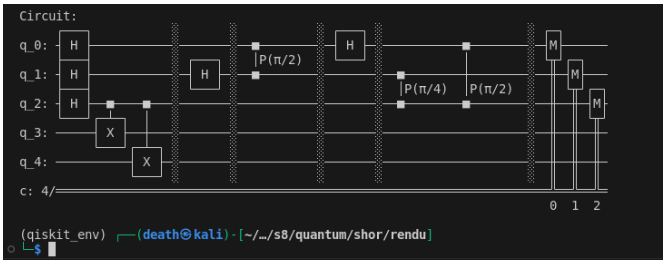


FIGURE 3 – Circuit quantique pour $N = 15$ et $x = 11$

La première figure (a) représente les résultats simulés sur un ordinateur quantique idéal. C'est pourquoi on observe que la probabilité d'obtenir un résultat différent de 0 ou 4 est nulle. En revanche, dans la figure (b), nous avons utilisé un modèle de bruit de la bibliothèque Qiskit, ce qui permet de simuler l'implémentation de l'algorithme sur un véritable ordinateur quantique.

Les deux valeurs ayant la plus grande probabilité, d'après les deux figures, sont 0 et 4. 0 est un cas trivial, mais 4 indique une période potentielle.

Pour vérifier que 4 est bien la période, nous pouvons procéder comme suit : $11^4 \equiv 1 \mod 15$.

Calculons $11^4 \mod 15$:

$11^2 \equiv 121 \mod 15 = 1, \quad \therefore 11^4 = (11^2)^2 \equiv 1^2 = 1 \mod 15$.

Ainsi, $r = 4$ est bien la période, ce qui confirme le résultat.

3.2 Choix de la Base x

Dans l'algorithme de **Shor**, la valeur de x est généralement choisie de manière aléatoire. Dans notre cas, la valeur $x = 11$ a été choisie à titre d'exemple théorique. L'objectif était de déterminer la période pour différentes valeurs de x. Ainsi, les circuits suivants ont été conçus pour tester le résultat pour différentes valeurs de x.

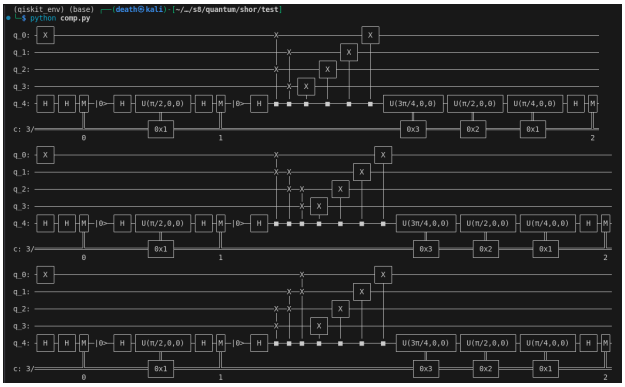


FIGURE 5 – Les trois circuits quantiques pour x=7,11 et 13

Nous observons que les résultats correspondent à nos attentes pour $x = 7$, $x = 13$ et $x = 11$. Cela démontre que, quelle que soit la valeur de x, l'algorithme indique que la période est toujours égale à 4 lorsque $N = 15$.

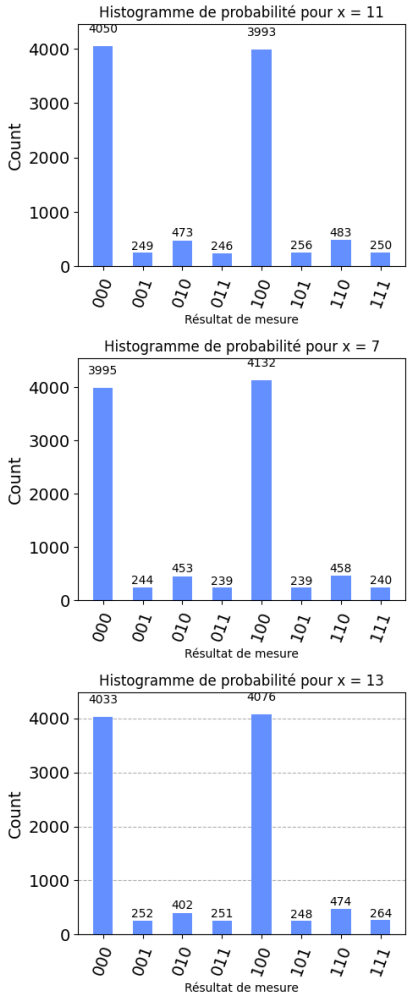


FIGURE 6 – Comparaison des résultat pour differents valeurs de x

3.3 Étude de l'Algorithme de Shor pour $N = 21$

Pour approfondir l'étude, nous avons pris le premier circuit utilisé pour $N = 15$ et $x = 11$ et tenté de l'adapter pour $N = 21$ et $x = 11$. En raison de sa simplicité, ce circuit semblait être un bon candidat pour cette extension. Pour cela, nous avons modifié le circuit en ajoutant un qubit supplémentaire, car 21 nécessite 6 qubits pour sa représentation binaire (puisque $2^5 = 32$ est le plus petit 2^k supérieur à 21).

Après avoir configuré le circuit avec ces paramètres, nous avons réalisé les exécutions sur le simulateur quantique de Qiskit, obtenant les résultats suivants. Ces résultats sont illustrés dans la figure 7.

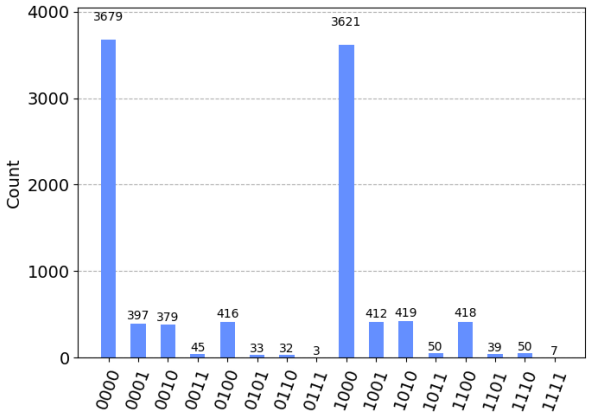


FIGURE 7 – Résultats des mesures pour $N = 21$ et $x = 11$.

Nous avons obtenus. Nous remarquons que la période trouvée est 8 ($0b1000$).

Pour vérifier si cette période est correcte, examinons les calculs suivants :

- $11^1 \equiv 11 \pmod{21}$
- $11^2 \equiv 121 \equiv 121 \mod 21 = 121 - 5 \cdot 21 = 16$
- $11^3 \equiv 11 \times 11^2 \equiv 11 \times 16 \equiv 176 \equiv 176 \mod 21 = 176 - 8 \cdot 21 = 8$
- $11^4 \equiv 11 \times 11^3 \equiv 11 \times 8 \equiv 88 \equiv 88 \mod 21 = 88 - 4 \cdot 21 = 4$
- $11^5 \equiv 11 \times 11^4 \equiv 11 \times 4 \equiv 44 \equiv 44 \mod 21 = 44 - 2 \cdot 21 = 2$
- $11^6 \equiv 11 \times 11^5 \equiv 11 \times 2 \equiv 22 \equiv 21 = 1 \mod 21$

Nous constatons que la période correcte est 6.

Bien que l'algorithme de **Shor** fonctionne bien pour $N = 15$, il rencontre des difficultés avec $N = 21$. Ces résultats soulignent l'importance de la gestion des erreurs et de la robustesse des circuits quantiques. La factorisation de nombres plus grands reste un défi pour l'algorithme de **Shor**, en particulier en présence de bruit et d'imprécisions.

Il est évident qu'un circuit simple ne peut pas toujours donner des résultats cohérents, comme ceux trouvés pour $N = 15$. Des circuits plus complexes et davantage de tests sont nécessaires pour obtenir des résultats corrects.

4 CONCLUSION

Dans ce document, nous avons étudié l'algorithme de **Shor**, son efficacité et son impact sur la cryptographie moderne, tout en abordant les défis liés à son implémentation dans des environnements quantiques réels. En adoptant une **démarche scientifique**, nous avons commencé par analyser la structure et le fonctionnement de l'algorithme, en mettant en avant deux éléments clés : la transformation de Fourier quantique et la période quantique.

Dans la partie théorique, nous avons détaillé le rôle de chaque étape de l'algorithme, expliquant comment il réduit le problème de la factorisation d'entiers à la recherche de périodes. Nous avons également discuté de la manière dont l'algorithme utilise la superposition quantique pour explorer de multiples solutions en parallèle, et comment la transformation de Fourier quantique est utilisée pour identifier la période de la fonction. Ensuite, dans la partie expérimentale, nous avons mis en œuvre l'algorithme en utilisant des outils tels que Qiskit, afin de tester son efficacité sur des cas particuliers.

En testant l'algorithme sur $N = 15$, nous avons pu démontrer son efficacité, tandis que sur $N = 21$, l'algorithme a rencontré des difficultés. Ces résultats suggèrent que l'algorithme nécessite une optimisation des paramètres et une amélioration de la gestion des erreurs dans les environnements quantiques actuels pour fonctionner correctement sur des nombres plus grands.

L'algorithme de **Shor** démontre le potentiel des algorithmes quantiques pour résoudre des problèmes difficiles pour les ordinateurs classiques. Cependant, la complexité des circuits quantiques et les limitations actuelles des ordinateurs quantiques indiquent qu'il reste des obstacles à surmonter avant que l'algorithme de **Shor** ne devienne une menace immédiate pour les systèmes cryptographiques existants.

Néanmoins, son développement souligne l'importance de poursuivre les recherches en informatique quantique, non seulement pour améliorer les algorithmes actuels, mais aussi pour renforcer la sécurité des systèmes cryptographiques face à de potentielles menaces futures. Dans l'ensemble, en suivant la **démarche scientifique**, nous avons formulé et testé une hypothèse de manière rigoureuse, ce qui a permis de valider notre approche et de mettre en lumière des aspects clés de l'algorithme de Shor.

RÉFÉRENCES

[1] Patrick J. Coles, Stephan J. Eidenbenz, Scott Pakin, Adetokunbo Adedoyin, John Ambrosiano, Petr M. Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo N. Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Andrey Y. Lokhov, Alexander Malyzhenkov, David Dennis Lee Mascarenas, Susan M. Mniszewski, Balu Nadiga, Dan O'Malley, Diane Oyen, Lakshman Prasad, Randy Roberts, Philip Romero, Nandaki**Shor**e Santhi, Nikolai Sinitsyn, Pieter Swart, Marc Vuffray, Jim Wendelberger, Boram Yoon, Richard J. Zamora, Wei Zhu : Quantum Algorithm Implementations for Beginners.

[2] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. Experimental realization of **Shor**'s quantum factoring algorithm using nuclear magnetic resonance. Nature, 414 :883–887, December 2001.

[3] IBM Quantum. `https://docs.quantum.ibm.com/`

[4] Enrique Martin-Lopez, Anthony Laing, Thomas Lawson, Roberto Alvarez, Xiao-Qi Zhou, and Jeremy L O'brien. Experimental realization of shor's quantum factoring algorithm using qubit recycling. Nature photonics, 6(11) :773–776, 2012.

[5] Michael A. Nielsen and Isaac L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, Cambridge, United Kingdom, 2016. 10th Anniversary Edition.

[6] Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh Vazirani. Algorithms. McGraw-Hill, Inc., New York, NY, USA, 2008.

[7] Peter W Shor. Algorithms for quantum computation : Discrete logarithms and factoring. In Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on, pages 124–134. IEEE, 1994.

[8] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 21(2) :120–126, February 1978.