Empirical Investigation of Quantum Computing on Solving Complex Problems

Shahid Hussain[1](✉), Yuba Neupane[1], Wen-Li Wang[1], Naseem Ibrahim[1], Saif Ur Rehman Khan[2], and Asif Kareem[3]

[1] Department of Computer Science and Software Engineering, School of Engineering, Penn State Behrend, Erie, USA
{shussain,wxw18,nii1}@psu.edu
[2] Department of Computing, Shifa Tameer-E-Millat University (STMU), Islamabad 45550, Pakistan
saif_rehman.ssc@stmu.edu.pk
[3] Advocate Health, Downers Grove, IL, USA

Abstract. Context: The rules of Quantum Mechanics have been exploited through Quantum Computing (QC) to solve specific problems and process information in expeditious ways as compared to Conventional Computing (CC) such as factoring integers. **Problem:** With the alluring computation capability of QC, it is still important to assess the implications and limitations of QC in solving a variety of computationally demanding problems. **Method:** In this regard, an empirical study was conducted to assess the efficacy of QC in terms of solving certain complex problems by keeping a tradeoff between the execution time and problem size. An analysis was performed based on the widely used Shor's algorithms and the efficacy of QC as compared to CC was reported. **Results:** The outcomes show that QC has the potential to exponentially speed up the identification of a solution to certain polynomial problems that are intractable for CC. However, further research is needed to fully understand the potential and limitations of QC for Non-Polynomial (NP) complete problems.

Keywords: Quantum Computing · Conventional Computing · Shor's Algorithm · Complexity

1 Introduction

Quantum Computing (QC) is a revolutionary technology that exploits the rules of Quantum Mechanics to speedily solve specific problems and process information in ways that are not possible with classic Conventional Computing (CC). In CC, computer scientists are interested in solving optimization and decision problems within polynomial and/or non-polynomial time limit. This is very challenging for NP-complete problems for the problem size can grow exponentially. In contrast to CC for having one state to be on or off at a time, QC can deal with multiple states at the same time, contributing to its high-speed computation performance. Hence, computer science and software engineering communities have endeavored to leverage the discriminative power of QC to

P. Kruchten and P. Gregory (Eds.): XP 2022/2023 Workshops, LNBIP 489, pp. 222–230, 2024.
https://doi.org/10.1007/978-3-031-48550-3_22

find solutions for difficult optimization and decision problems by possibly maintaining a polynomial time even after a significant growth in problem size. One of such problems is to factor numbers into primes and determine the answer in a reasonable time, especially when the number of digits grows exponentially [1]. Factoring integers is a complex problem that has been widely studied in number theory and cryptography. However, the conventional computers have struggled to factor large integers within a polynomial time until the introduction of Shor's algorithm [2]. The algorithm is a QC approach developed by American mathematician Peter Shor in 1994 to solve this hard problem. Shor's algorithm is also recognized as a strong tool in the cryptographic and code-cracking domain to crack cryptographic algorithms that are frequently used for online communication and trade. This is because many encryption algorithms, including the RSA, are developed based on the challenge of factoring huge integers. The Shor's method can factor large integers significantly faster and more quickly than traditional algorithms, thus defeating the RSA encryption and many other encryption schemes [3].

QC intrigues the community to perform complex tasks by exploiting the principles of quantum mechanics that can account for multiple states simultaneously to gain great performance. Such ability is deemed impossible or impractical to CC with only one state at a time. Today, computer scientists are interested in solving complex problems as quickly as possible using QC even with an exponential growth in the problem size. In [4], Devitt et al. have investigated the practical implementation of Shor's algorithm. In this study, an empirical study is conducted to investigate the impact, ability, and limitation of QC in terms of solving complex problems in polynomial time. A comparative analysis is performed between QC and CC in terms of solving complex problems in polynomial time. Moreover, the experiment explores the effects and benefits of QC in solving less complicated polynomial problems.

2 Related Works

This section gives a summary of the related work, implications, and limitations. In [5], Ugwuishiwu et. al. Investigated the mechanisms of quantum cryptography and compared quantum and classical encryption schemes. In their study, the authors gave a brief overview of quantum computation and explained Shor's algorithm. Moreover, they demonstrated the power of QC in terms of how encryption could be accomplished by utilizing the properties of quantum particles and provided examples of the complexities of Shor's algorithm [5].

Quantum computers and Shor's algorithm can pose a threat to today's cryptographic systems. With the prevalence of data breaches, researchers are increasingly interested in finding ways to safeguard data security using quantum cryptography [4].

In addition to the benefits of QC, Aaronson [6] also identified certain limitations. The investigation showed that quantum computers could be extremely efficient at some specific tasks, where the computation abilities outperformed current computers moderately for most problems. This realization indicates a potential to lead to the discovery of new fundamental physical principles. The concept of a "magic computer" capable of quickly solving NP- complete problems, could change the world, as it could be used to find patterns in large datasets such as stock market data or brain activity recordings

[6]. Nevertheless, the author also claimed that quantum computers are not an all-purpose device. They are best suited for specific types of computations, such as those that involve large amounts of data and require high-speed processing. These are known as quantum advantage tasks, e.g., quantum simulation and quantum cryptography [5, 6].

In [7], Nene and Upadhyay scrutinized a well-known and widely used encryption-based RSA algorithm and investigated the difficulty of factoring large integers within polynomial time. The authors described the capability of Shor's quantum algorithm in terms of breaking RSA encryption in a reasonable time. The authors further presented a systematic approach to factoring integers using Shor's quantum algorithm on a classical computer through simulation. The results were verified theoretically and concluded a need for further empirical investigations [8–10].

Thomas et al. [12] tried to leverage and scale Shor's algorithm in their study on Ion-trap quantum computers (a particular kind of quantum computer). The authors applied the algorithm with the use and manipulation of seven qubits and four "cache qubits". They factored the number 15 through extended arithmetic operations and modular multipliers. With a degree of confidence of more than 99%, the algorithm was able to provide the proper factors. This is a crucial milestone towards the creation of a scalable quantum computer and the actual use of quantum algorithms [13, 14].

3 Proposed Method

This study empirically assesses the efficacy of QC in solving complex problems and reason the tradeoff between execution time and problem size. The layout of the proposed method is shown in Fig. 1, and the constituent components are described as follows.

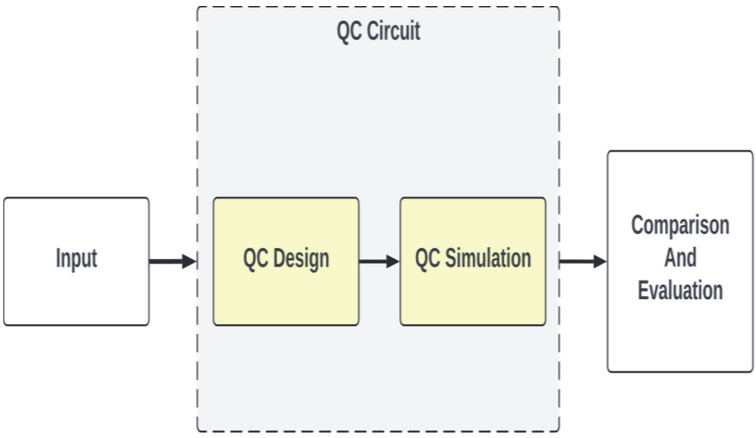

Fig. 1. Workflow of the proposed method

3.1 Input

This component represents the input for a chosen algorithm. Take the Shor's algorithm for example. The input is an integer value, whose prime factor is expected to be found.

3.2 QC-Circuit

The QC-Circuit in the proposed method is functional in two steps as follows.

3.2.1 Quantum Circuit Design

The layout of the QC-circuit design is shown in Fig. 2. The exponent register is first placed into an equal superposition of states using Hadamard gates. This is a vital step in making sure the adopted quantum algorithm can fully benefit from quantum physics' features. The final qubit in the target register is then subjected to the application of a phase kickback gate. This gate is used to help make sure the algorithm can run smoothly and produce the most accurate result possible.

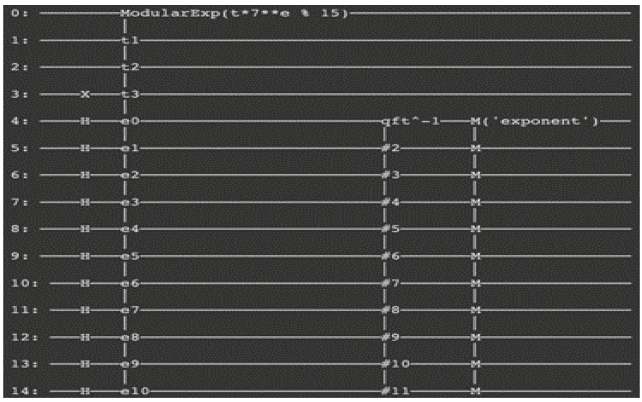

Fig. 2. Quantum Circuit design for input value of 15 in Shor's Algorithm

Afterwards, a modular exponential gate is used to perform the series of controlled unitary units required for phase estimation. This gate, which is an essential part of the process, enables us to precisely determine the phase of the quantum state that is dealt with. Once finished, the next step will apply the inverse quantum Fourier transform to the exponent register. This phase is crucial because it enables us to get pertinent data about the qubits' current state from the register and ensures that the final output is as precise as feasible.

To obtain the outcome, the exponent register will be measured and the data from the register's qubits in this last phase of the procedure can be retrieved. The output of the algorithm is the result of this measurement, and it may be utilized to carry out different computing operations.

3.2.2 Quantum Circuit Design

For simulation of the proposed QC-design, the capabilities of Google's Cirq was leveraged. Using Google's Cirq, the Quantum Virtual Machine enables us to operate and test quantum circuits on simulated hardware that replicates the limitations and noise behavior of real quantum devices. Before placing our quantum algorithms to use on the actual

quantum hardware, the simulator of the virtual machine provides an economical way to test and debug the algorithms.

3.3 Comparison and Evaluation

Google's open-source quantum computing framework, Cirq, provides an accessible platform for researchers and developers to experiment with Shor's algorithm and test it with different integer inputs. With the help of Cirq, we utilized the already implemented Shor's algorithm to run on a quantum simulator by Google, to test the performance of the algorithm in factoring integers with different numbers of digits.

For comparison, the general number field sieve (GNFS) algorithm [15] in number theory was also applied to integers with different numbers of digits because this traditional computing approach can factor integers within a much more reasonable amount of time than the brute-force method documented on Cirq with demo code. However, it's still not as efficient as the quantum algorithm. Our evaluation is based on the size of the input integer and the time required to locate the prime factors. These two are considered as the major criteria for assessing the QC performance in solving the integer factoring problems in polynomial time. The size of the input integer will be determined by the number of digits, which represents how lengthy the number is [9]. The time it takes to locate the prime factors will be measured in seconds and used to reflect the computational efficiency of the algorithm.

The success of comparing the performance of QC with that of the traditional computing approach through the input size and time as two key metrics will enable us to study the influence of further issues on QC performance and identify other potential barriers to its wider adoption involving more problems. The statistics can then provide straightforward and objective results to assess QC's performance.

4 Experimental Procedure

The following steps present our experimental procedure of the proposed study to run the Shor's algorithm.

Step-1: Initializing an input and output qubit register in the quantum computer is the first step, with n and $n0$ qubits, respectively, assigned to the state $|\psi 0\rangle = |0\ldots 0\rangle n|0\ldots 0\rangle n0$.

Step-2: The next step is to prepare a superposition by applying q Hadamard gates, where $q = 2n$.

Step-3: The output register of state $|\psi 2\rangle$ is measured. The result of that measurement is then discarded and put into the input register of state $|\psi 3\rangle$.

Step-4: The result y of the input register of state $|\psi 4\rangle$ is measured. This value helps determine the continued fraction representation of y/q. Finally, each convergent result is tested in order and reduced to their lowest terms.

5 Result and Discussion

The result of the proposed method is displayed in Table 1 and Fig. 3. In the table, it shows the measured values of QC and CC in terms of keeping a tradeoff between problem size and polynomial time. Figure 3 on the other hand gives a visual representation of the same results for easy comparison between QC and CC using a bar chart.

The classical computer can take an exponential amount of time to calculate the factors as the number of digits in the integers becomes larger. In contrast, the quantum computer spends about the same amount of time to factor the integers, regardless of the number of digits. The time gap between the quantum and conventional computers becomes more and more significant as the problem size increases. For example, according to the results of Table 1 for the case of 231273, the quantum computer takes less than one second, but the conventional computer takes more than five seconds.

Table 1. Time analysis for factoring using Quantum and Classical computing approaches

Tests			Quantum Computer		Classical Computer	
	Input	Result	Time(s)	Time(ms)	Time(s)	Time(ms)
1	4	[2,2]	0.000999212	0.999212	0.00097537	0.97537
2	6	[2,3]	0.001001596	1.001596	0.001053333	1.053333
3	8	[2,4]	0.000996828	0.996828	0.000994921	0.994921
4	9	[3,3]	0.001024961	1.024961	0.001020432	1.020432
5	10	[5,2]	0.001003265	1.003265	0.000999928	0.999928
6	12	[2,6]	0.000989676	0.989676	0.000981808	0.981808
7	14	[2,7]	0.001004457	1.004457	0.001044273	1.044273
8	15	[3,5]	0.001005457	1.005457	0.001003027	1.003027
9	314	[2,157]	0.001009226	1.009226	0.001000166	1.000166
10	529	[23,23]	0.001008749	1.008749	0.001025677	1.025677
11	1011	[3,337]	0.001010418	1.010418	0.002021385	2.021385
12	23127	[3,7709]	0.000991106	0.991106	0.002010656	2.010656
13	231273	[21,11013]	0.000995636	0.995636	0.005043123	5.043123

Finding 1:

With the increase in problem size, the efficacy of QC over the CC becomes more evident.

The results of Fig. 3 conclude our first finding that quantum computer can be exponentially quicker than the classical computer in solving the proposed algorithm (i.e., Shor's algorithm), especially when the problem size increases. However, the advantage of QC over CC diminishes when the problem size is rather small, adhered to [11].

Finding 2:

As compared to CC, the efficacy of QC could be significantly improved when job isaccomplished with a large input size.

It is crucial to remember that the performance of a quantum computer is dependent on a variety of parameters, including the number of qubits. Factoring big numbers is simply one of the many polynomial problems that quantum computers can perform faster

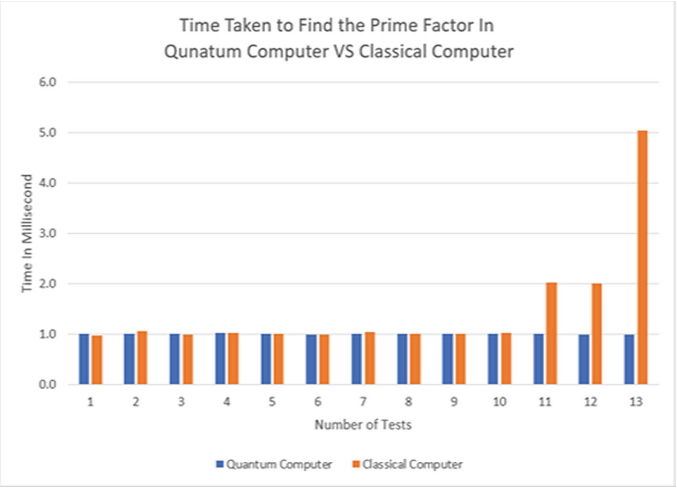

Fig. 3. Comparative analysis of QC and CC for Shor's algorithm

than classical computers. One of the significances is that quantum computers can take a rather constant time, as shown in Fig. 3, to find solutions for the increasing problem sizes. This on the other hand indicates our second finding that QC can process a large input size of data remarkably faster than CC as well. Nonetheless, the two findings do not imply that quantum computers are superior to classical computers in all activities.

6 Implications to Research Community

The aim of the proposed study is to empirically investigate the implications of QC to solve complex problems in terms of polynomial time and its advantage over CC. Through the experimental results of the proposed study, we have realized the following consequences for the research community.

- Quantum computing has the ability to dramatically accelerate the process of factoring big numbers in polynomial time.
- Cryptography and code cracking could be possibly done in polynomial time.
- Rapid resolution for polynomial problems can also have significant effects in other disciplines, including machine learning, optimization, and simulation.
- Researchers might employ quantum computers to more accurately and efficiently model physical systems and tackle challenging optimization issues.
- To secure sensitive information, the improvement to many existing security methods could rely on QC.
- The speed with which complex or hard problems may be solved also suggests that other issues, like as the Traveling Salesman Problem or other kinds of search tasks, which are not known to be time-consuming on classical computers, can be solved using quantum computers.

In short, the speed of quantum computers in rapid problem-solving has the potential to revolutionize numerous industries and offer new opportunities for research and technological development.

7 Conclusion

The experimental results have indicated the efficacy of Quantum Computing (QC) over Classical Computing (CC) to solve complex problems such as optimization and decision problems in terms of polynomial time. This proposed study was conducted to investigate the QC claim regarding its speed of solving complex problems and efficacy over CC. The well-known and widely used Shor's algorithm was exploited, and the capabilities of Google Cirq were leveraged to design and simulate the QC circuit for the algorithm. This empirical study successfully measures the performance of QC and CC and performs a comparative analysis. The conclusions of this work are; 1) With a small problem size, the performance of QC shows no superiority to CC for Shor's algorithm, 2) QC starts to outperform CC when the problem size or input size grows large, 3) Cryptography and code cracking application could rely on QC for its ability to solve complex problems more quickly, and 4) Research community can rely on QC to solve NP-complete problems in a much greater performance, such as Knapsack, Hamiltonian path and Travelling salesman problems.

References

1. Rietsche, R., et al.: Quantum computing. Electron. Mark. 1–12 (2022). https://doi.org/10.1007/s12525-022-00570-y
2. Voorhoede, D.: Superposition and entanglement. Quantum Inspire. https://www.quantum-inspire.com/kbase/superposition-and- entanglement/
3. Knill, E.: Quantum computing. Nature **463**(7280), 441–443 (2010)
4. Devitt, S.J., Fowler, A.G., Hollenberg, L.C.: Investigating the practical implementation of Shor's algorithm. In: SPIE Proceedings (2005)
5. Ugwuishiwu, C.H., Orji, U.E., Ugwu, C.I., Asogwa, C.N.: An overview of quantum cryptography and Shor's algorithm. Int. J. Adv. Trends Comput. Sci. Eng. **9**(5), 7487–7495 (2020)
6. Aaronson, S.: The limits of quantum. Sci. Am. **298**(3), 62–69 (2008). JSTOR. http://www.jstor.org/stable/26000518
7. Nene, M., Upadhyay, G.: Shor's algorithm for quantum factoring. In: Choudhary, R., Mandal, J., Auluck, N., Nagarajaram, H. (eds.) Advanced Computing and Communication Technologies, pp. 325–331. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-1023-1_33
8. Google: Shor's algorithm: Cirq. Google Quantum AI. https://quantumai.google/cirq/experiments/shor
9. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)
10. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. Quantum Phys. 20–22 (1996)
11. Artur, E., Jozsa, R.: Shor's quantum algorithmfor factorizing numbers. Rev. Mod. Phys. **68**, 733–753 (1996)

12. Thomas, M., et al.: Realization of a scalable Shor algorithm. Science **351**, 1068–1070 (2016)
13. Eleanor, R., Polak, W.: An introduction to quantum computing for non-physicists. ACM Comput. Surv. **32**, 300–335 (2000)
14. Li, W., et al.: An image classification algorithm based on hybrid quantum classical convolutional neural network. J. Quantum Eng. (2022)
15. Crandall, R., Pomerance, C.: Prime Numbers: A Computational Perspective, 2nd edn. Springer, New York (2001). Section 6.2: Number field sieve, pp. 278–301. https://doi.org/10.1007/978-1-4684-9316-0