Quantum computation: a brief introduction

João Lucas Martinello de Oliveira

[a] Faculty of Engineering and Computer Science, Universidade Regional Integrada do Alto do Uruguai e das Missões (URI), Rio Grande do Sul, Brazil, joaololiveira@aluno.santoangelo.uri.br.

Abstract. This article seeks to explain the basic concepts of Quantum Computation. While classical computers rely on binary information transmitted through digital electrical signals and is processed through boolean logic, quantum computers represent information as qubits, which can exist in a linear combination of states, and follows the rules of quantum mechanics. They can be physically implemented as polarized photons, superconducting circuits and more. The article explores quantum logic gates, the challenge of quantum decoherence and how quantum error correction tries to mitigate it. Furthermore, computational complexity theory is introduced as a way of understanding the limitations of classical and quantum computers and their differences. Even though the type of computable problems are the same for quantum and classical, quantum computers have some advantages in efficiency for certain types of problems, such as simulation of quantum systems, since the computations themselves are made in quantum systems, factorization of prime integers and unstructured search.

Keywords. Quantum computing, quantum logic, quantum information, computational complexity theory.

1. Introduction

Quantum computing is a relatively new field that is at the direct intersection between computer science and quantum mechanics. By leveraging properties present at the quantum scale, we can represent information and make computations in quantum systems. But why use quantum computers to do computations instead of a regular computer? What kind of computations can a quantum computer do that a classical Turing machine can't? Richard Feynman [1] observed that quantum systems cannot be efficiently simulated by classical computers, due the intrinsic properties of the systems. Furthermore, Peter W. Shor [2] developed algorithms for quantum computers to factorize prime numbers and discrete logarithms in polynomial time, problems considered hard on a classical computer.

The term "quantum computing" is becoming increasingly popular as it approaches mainstream media. Many believe quantum computing will completely shift the computing paradigm and is here to replace classical computers, without having any deeper knowledge of its functioning. As approaching a (relatively) new field, one can quickly lose sight of scientific ground. For this reason, the proposed article seeks to explain the foundations of quantum computing in a simple manner, from basic concepts of quantum physics, how information is represented in classical and quantum computers, to how they fit in the computational complexity theory.

2. Research Methods

A bibliographical review was conducted as the primary research method. This process involved a systematic identification and selection of relevant literature related to the topic in different knowledge databases.

3. Quantum mechanics concepts

To understand how a quantum computer works, it is first necessary to understand some concepts in quantum physics.

3.1 Wave function and measurement

Quantum particles exhibit both particle and wave properties, depending on the experimental circumstance.

In classical physics, the act of measurement doesn't affect an object. All the information of a classical object can be known with precision without necessarily disturbing it. In quantum physics, this is not the case. The act of measurement of a quantum particle plays a fundamental role [3]. Before measurement, a quantum system can be described as a wave function, which is a complex-valued probability amplitude. This amplitude is a representation of the probability distribution of the system's properties, like position and momentum.

After measurement, the wave function "collapses" to a random outcome according to the probability distribution and returns to a regular classical description.

4. Classical Computation

4.1 Information and processing

Classical computing relies on binary information. Bits can be either in a 0 or 1 state and, as you increase the number of bits in a single string, you increase the number of possible representations by 2^n, where n is the length of the binary word. Physically, computers use electrical signals to represent bits. A high voltage level represents 1 and a low voltage level represents 0, for example.

To make computations, logic gates are utilized. They perform on *Boolean* logic, that is, binary inputs to produce a single binary output. Some gates take two bits as input and produce a single bit output: the AND gate outputs a 1 only when both inputs are 1. The OR gate outputs a 1 only when at least one of the inputs is 1. The NOT gate is a single input gate that inverts the value entered. There are multiple logic gates, and together they are used to create circuits that can perform more complex computations.

It is important to highlight that classical computing is mostly described and operated within the mathematical framework of discrete math, as it deals with discrete, binary data.

5. Quantum computation

5.1 Qubits

In quantum computation, the elementary unit is a quantum bit, or *qubit.* The state of a qubit can be represented as a vector in a two-dimensional complex vector space:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Where $|\psi\rangle$ is the state vector (holds all the information of the system) of the qubit; $|0\rangle$ and $|1\rangle$ are the basis state, corresponding to the classical binary states 0 and 1. They can also be represented as matrices:

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

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

α and β are complex numbers that describes the probability amplitude of the qubit being in the $|0\rangle$ or $|1\rangle$ states, respectively. It's important to know that α and β must satisfy the condition

$$|\alpha|^2 + |\beta|^2 = 1$$

since the probabilities must sum to one, as it guarantees the measurement to be in either state [4].

This means that, different from a classical bit, which can only be at one state, qubits can exist as a *linear combination of states,* which is also called a *superposition* of states [5].

There is another way of visualizing single qubits, and that is in the form of a three-dimensional sphere, often called *a Bloch sphere.* It provides a way to visualize all the states that a qubit can possibly be in.

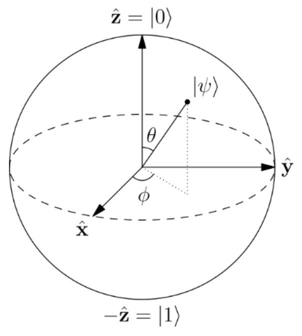

Fig 1. Bloch sphere representation of a qubit.

Any state of a qubit in this representation can be written as:

$$|\psi\rangle = cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}sin\left(\frac{\theta}{2}\right)|1\rangle$$

Where the angle θ corresponds to latitude and angle ϕ corresponds to longitude.

There are a few different ways of physically implementing a qubit, such as using polarization of a photon; ion traps; superconducting circuits and more [6].

5.2 Quantum logic gates

Analogue to classic computing, quantum computers also use logic gates to perform computations. However, different types of gates are utilized. The following are some examples of quantum gates that operates on a single qubit: the Pauli-X gate, or just X gate, is the quantum equivalent of a classical NOT gate:

$$X{:}\ |0\rangle \rightarrow |1\rangle \text{ or } |1\rangle \rightarrow |0\rangle$$

In the form of a matrix:

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

The Hadamard gate, represented by *H,* puts a qubit into a superposition state, where it has an even probability of collapsing in either state:

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

One example of a two-qubit quantum gate is the controlled-NOT gate, or CNOT. The first qubit is usually called the control bit and the second, target bit: If the control qubit is in state $|1\rangle$, the gate will perform a Pauli-X gate on the target qubit. In matrix representation

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

So, for example, let the control qubit = $|1\rangle$ and the target qubit = $|0\rangle$. You can represent the state of the two qubits as a single vector by taking the tensor product between each qubit:

$$\begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} = |10\rangle$$

Calculating the matrix multiplication

$$\text{CNOT} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}0\\0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} = |11\rangle$$

5.3 Interference

Due to the wave-like behavior of quantum particles, qubits can exhibit interference patterns, as seen in double-slit experiments. And it turns out that interference plays an important role in developing quantum algorithms [7]. The objective is to arrange all the interference patterns in a way that only the computations of our interest in a given problem remains, and all the rest to cancel out [8]. Different paths leading towards a wrong answer are arranged in a way that *destructively* interfere and cancel eachother out, and paths towards the right answer will *constructively* interfere and add up [9]. One example is Grover's algorithm [10] for unstructured search, which uses amplitude amplification to find a single marked element.

5.4 Decoherence and quantum error correction

Quantum computers need an extremely appropriate environment, since maintaining the quantum properties of the quantum system is essential to keep its coherence, that is, maintaining the quantum state over time without disturbance by any external factors. If the quantum system is not perfectly isolated from the external environment, the properties of the environment can interfere with those of the system, compromising the computations [11]. There are also other factors that can cause a disturbance in the quantum system, such as incorrectly applied gates.

One way to mitigate the problems of decoherence of quantum states over arbitrarily long periods of time is using techniques of *Quantum Error Correction,* or QEC. At its most basic level, QEC uses the idea of redundant encoding [12]. Like classical error correcting codes, redundant information is added in a way that you will be able to identify which information transmitted was wrong, and correct it. For example, Shor [13] developed a 9-qubit error correcting code that is able to correct a logical qubit from one bit-flip, one phase-shift or both, where a single qubit is encoded into nine qubits.

6. Complexity theory

The difficulty of solving computational problems is a fundamental study in the field of computer science, more specifically in computational complexity theory. Difficulty is formalized in terms of the number of resources required by different models of computers to solve problems. Constraints of resources (time and space) and computation models (deterministic, nondeterministic, and probabilistic) are used to make classifications. It is concerned with the fundamental capabilities and limitations of computers [14].

The theory is focused on determining the different types of problems based on how efficiently they can be solved. Easy problems can be classified as those in which you have a perfect method to arrive at a solution. Problems with only a modest method of finding solutions are considered more difficult. If there is no known method to efficiently solve a problem, it is considered hard [15].

The two most known classes are P and NP. The *Polynomial-Time* class, P, is the class of all decision problems (problems where there is a *n*-bit string input and the outputs is either yes or no) that are solvable in polynomial-time by a deterministic Turing machine, that is, a classical computer. Problems in this class can be easily solvable and verified, if given an answer, within polynomial time.

NP is the set of decision problems that are solvable in polynomial time with a Nondeterministic Turing machine and verifiable in polynomial time by a deterministic Turing machine. This means that, though solving the problem is considered hard or even impossible for a classical machine, depending on the problem, verifying the answer in it is possible and easy [15]. There is also a subset of NP problems that are called NP-complete, which is the set of problems that, if you had an algorithm that can solve in polynomial-time any of the problems, then there exists a polynomial-time algorithm for all the NP problems [16]. NP-complete problems are considered as hard as the hardest problems in NP.

The question of whether P = NP is not only the most important unsolved problem in computer science, but one of the deepest questions that human beings have ever asked, according to Scott Aaronson [16]. It is mostly believed that P ≠ NP, but there is still no formal proof.

In quantum computation, there are different types of complexity classes. One of the most important classes is BQP, which stands for *Bounded-error quantum polynomial time,* as it represents the set of decision problems that can be efficiently solved by a quantum computer [17], with an error probability of at most 1/3. BQP is the quantum analogue for BPP (*bounded-error probabilistic polynomial time),* regarded as the complexity class of efficiently solvable problems by a Turing machine with an error probability of at most 1/3 [18].

The exact relationship between all complexity classes is still unknown. There is evidence that BPP ≠ BQP, that is, whether quantum computing is more powerful than classical computing, such as Shor's [2] algorithm for factoring and discrete logarithms, which is in the NP class. Furthermore, quantum computers are not known nor believed to be able to solve NP-complete problems efficiently [18].

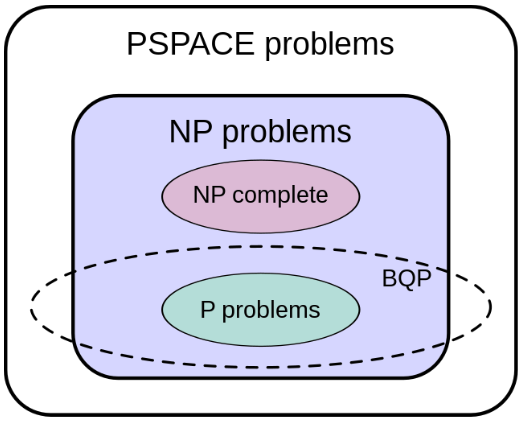

Fig 2. Suspected relationship of BQP with other classes, where PSPACE is the class of problems solvable by a deterministic Turing machine using polynomial amount of memory.

It is important to know that, even though the precise relationship between each complexity class is still unknown, the type of computable problems is the same for both quantum and classical computers, meaning that if a problem is incomputable for a classic computer, is also incomputable for a quantum one.

7. Final considerations

The development of the present article made it possible to obtain a brief knowledge in the mechanisms of quantum computers. Classical and quantum computing offer distinct approaches to solving problems, and one is not to replace the other, rather complement each other.

Quantum computers are still in the early stages, being the first experimental implementation made in 1998 [19], but there are already some use cases that are more efficient than their classical counterparts. Moreover, as the field evolve, many real-world applications will start to be feasible, such as efficiently simulation of quantum systems: useful for quantum chemistry, high-energy physics, cosmology, for example; cybersecurity: since most of modern encryption is based on RSA cryptography, which relies on the fact that factoring large prime numbers is hard (but not for a quantum computer), and so on [20] [21].

8. References

[1] R.P. Feynman, Simulating physics with computers. *Int. J. Theor. Phys*. 1982; 21 (6): 467–488.

[2] P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput*. 1997: 26 (5) 1484– 1509.

[3] A. C. Philips. *Introduction to Quantum Mechanics*. Manchester Physics Series. John Wiley & Sons; 2013.

[4] Azure Quantum documentation. The qubit in quantum computing. Available at: https://learn.microsoft.com/en-us/azure/quantum/concepts-the-qubit. Access date: Sept, 07th 2023.

[5] Nielsen MA, Chuang IL. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge: Cambridge University Press; 2010.

[6] Ladd, T., Jelezko, F., Laflamme, R. et al. Quantum computers. *Nature*; 2010; 464: 45–53.

[7] McMahon, David. *Quantum computing explained*. John Wiley & Sons; 2007.

[8] Aharonov, Dorit. Quantum Computation. *Annual Reviews of Computation Physics VI*; 1999. pp 259-346

[9] Mandel, Jacob R. "Quantum Computing: Resolving Myths, From Physics to Metaphysics."; 2021.

[10] Grover, Lov. K. A fast quantum mechanical algorithm for database search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*; 1996.

[11] DiVincenzo, David P. Quantum Computation. *Science*; 1995. vol. 270, no. 5234 : 255–61.

[12] Devitt, Simon, J. et al. Quantum error correction for beginners. *Rep. Prog. Phys. 76 076001;* 2013.

[13] Shor, P. W. (1995). Scheme for reducing decoherence in quantum computer memory. *Physical Review A*; 1995. 52, R2493

[14] Sipser, Michael*. Introduction to the Theory of Computations*. Course Technology Cengage Learning; 2013.

[15] Aaronson, Scott. Lecture 6: P, NP, and Friends. PHYS771: *Quantum Computing Since Democritus.* University of Waterloo; lecture given Fall 2006

[16] Watrous, John. Quantum Computational Complexity. *Encyclopedia of Complexity and Systems Science;* 2018. Available at: https://doi.org/10.48550/arXiv.0804.3401.

[17] Bernstein, Ethan, Varizani, Umesh. Quantum complexity theory. *Proceedings of the Twenty-fifth annual ACM symposium on Theory of computing;* 1993. 11-20.

[18] Aaronson, Scott. "Why philosophers should care about computational complexity*." Computability:*

Turing, Gödel, Church, and Beyond; 2013. 261.

[19] Chuang, Isaac L., Gershenfeld, Neil, Kubinec, Mark. Experimental Implementation of Fast Quantum Searching. *Phys. Rev. Lett.;* 1998. 80, 3408.

[20] Georgescu, Iulia M., Sahel Ashhab, and Franco Nori. "Quantum simulation." *Reviews of Modern Physics*; 2014. 86.1 153. Available at: https://doi.org/10.48550/arXiv.1308.6253

[21] Bova, F., Goldfarb, A. & Melko, R.G. Commercial applications of quantum computing. *EPJ Quantum Technol;* 2021. 8, 2.