FEDERAL UNIVERSITY OF RIO GRANDE DO SUL
INSTITUTE OF PHYSICS
BACHELOR OF SCIENCE IN ENGINEERING PHYSICS

Natália Capra Ferrazzo

Grover's Search Algorithm:

Comparative study of the performance of quantum hardware made available by IBM, Microsoft and Amazon.

Porto Alegre
2022

Natália Capra Ferrazzo

Grover's Search Algorithm:

Comparative study of the performance of quantum hardware made available by IBM, Microsoft and Amazon.

Diplomation work in Engineering Physics at the Institute of Physics, Federal University of Rio Grande do Sul, as a requirement for obtaining the degree of Bachelor in Engineering Physics.

Advisor: Prof. Dr. Sandra Denise Prado
Co-advisor: Prof. Dr. Wilson Ricardo Matos Rabelo

Porto Alegre
2022

ACKNOWLEDGEMENT

I think it is fair to start my acknowledgements with the person responsible for helping me in my first steps in the area of exact sciences: my grandmother, Zaida. It was she who taught me how to do my first mathematical calculations. She used to prepare addition and subtraction calculations in a notebook that was next to the telephone in her living room. And I loved to spend my afternoons solving them. As soon as I was done, I would run into her lap asking for more, and she promptly complied with my request. Without a doubt, this was the first activity to awaken my interest in the area, the memories of which I will carry with me forever. Thank you, Grandma. I love you.

At this same time there was also another person essential to my education: my late grandfather, Duílio. He was the one who used to take my cousin Vinícius and I to and from school every day. I remember to this day that Vini and I would always argue about which way to go back home. It was up to my grandfather to solve this impasse. Thank you, Grandpa. Rest in peace. I will always love you.

After a while, when I was a little older, my afternoons were spent at the computer school of my father, Vilceu. It was there that my second passion arose: computing. It was also there where my interest in exact sciences and logic intensified. To keep me entertained while working, my father was creative: he would always come up with a different logic test or math problem for me to solve. My favorite was Einstein's Riddle - which I later learned is of uncertain authorship. However, it matters little whether the name is actually attributed to the correct person or not: discovering that the German was the owner of the fish consumed me a good deal of time; but more than that, it introduced me to fundamental concepts of logic – even though, at that time, I didn't even understand what that meant and was just curious to solve the challenge. Thank you, Dad. I love you.

As a teenager it was her turn – my mother, Claudete – to contribute to this journey. The role of a mother in the life of a child is so sublime that thanking her with words becomes an almost impossible task. Without her care and dedication, which began even before I was born, all this would be impossible. But I won't lie: our relationship was not always the best – especially during adolescence. However, she always did everything she could for my future and well-being, even if it meant renouncing her own well-being. Or forcing me to take English classes – something that today, with a more mature outlook, I am very grateful for. She was also the one who stood by me in the most difficult moment of my graduation, the last semester.

If it wasn't for her helping me at such a difficult time, when so much was happening, I don't know if I would have been able to finish it. Thank you, mom. I love you.

Speaking of difficult times, the one who is always there to listen to me and take me in when I need it is my sister, Vanessa. We are so connected that sometimes it feels like we are only one, she understands me like no one else in the world is able to. Thank you, Vani. I love you.

During my graduation I could count on exceptional teachers and mentors to guide me along the way. It is worth mentioning here the one who introduced me to my great passion, and subject of this work, quantum computing. It was in the Fundamentals of Quantum Mechanics discipline that Professor Pedro Grande made me see the beauty and strangeness of this unique area that is just taking its first steps. Thank you, prof. Pedro.

To define the subject and guide my research in the development of this project, I could count on my co-advisor, Professor Wilson Rabelo, who was always very helpful and accessible during this work. Thank you, Prof. Wilson.

I also thank the professors who accepted the invitation to participate in the committee and who oriented me in the process of creating this work. I thank them for their commitment and valuable notes. Thank you, prof. David. Thank you, Prof. Leonardo.

Finally, of course, I thank the one who is undoubtedly the greatest gift of this journey, whose honor to know I only got at the end of the graduation: my advisor, Professor Sandra Prado. Besides being an exceptional mentor and an extremely dedicated teacher, she is a person with a huge heart, whose contact I wish to maintain beyond graduation. Thank you, Prof. Sandra.

ABSTRACT

This work aims to detail the implementation process of Grover's search algorithm, focusing on the encoding of classical data into quantum data. It details the quantum fundamentals that govern the algorithm's operation. In addition, the platforms available for free for the implementation of quantum algorithms from IBM, Microsoft and Amazon are presented. Finally, a performance comparison is made between their quantum hardware. Through this, it is expected to provide the academy with a complete and updated study that is accessible not only to students from areas of Physics, but also to students of Engineering, Computer Science and related areas, aiming to promote research in different fields that can get benefit from quantum computing. In addition, it will provide Brazilian academia with a first step towards the development of benchmark methodologies in quantum hardware.

Palavras-chave: Grover's Search Algorithm; Quantum Computing; Encoding; Benchmark.

TABLE OF CONTENTS

1 INTRODUCTION

As Feynman predicted 40 years ago (R.P. FEYNMAN, 1982), several technologies based on quantum mechanics are currently used. From electronic components, integrated circuits in semiconductor chips, laser, electron microscopes, LED, superconductors to magnetic resonance imaging (MRI) tomography, all are based on the properties of quantum particle systems and their control. Concrete examples of quantum phenomena present in technologies are the *tunnelling effect* in transistors, the *coherence of photons* in lasers, the *quantum leaps* of electrons in atomic clocks. Physicists and engineers have become accustomed to these strange quantum phenomena. They form the basis of the first quantum revolution (JAEGER, 2018).

According to L. Jaeger (JAEGER, 2018, p. 21),

> Previous quantum technologies were essentially based on the behavior of many-particle quantum systems. The next generation of quantum technologies has its foundation in *manipulation of the states of single quantum particles*.

Corroborating Jaeger's sight, the emerging generation of quantum technologies has been developed based on directed preparation, control, manipulation and selection of *individual quantum particle* states and their interactions with each other. Because of this, one of the most peculiar phenomena of the quantum world is the star of this scenario: the *entanglement*. Through it, it is described as quantum particles can be in states in which they behave as if they were "connected" to each other, even if physically distant.

Beyond the “physical” world, one of the greatest benefits of understanding quantum mechanics is precisely that – understanding it; to explore its properties and apply it to problems previously approached in a classical way. Shor's algorithm (SHOR, 1994), Grover's algorithm (GROVER, 1996) and Deutsch-Jozsa algorithm (DEUTSCH; JOZSA, 1992) are good examples of this. By applying quantum concepts and quantum properties to algorithms, a new way of thinking and approaching problems emerges, often bringing exceptional computational performance gains and opening the possibility for a new era in computing.

1.1 CONTEXTUALIZATION

In recent years, many research centers in quantum technology have emerged in the world. High-tech companies are aware of the possibilities and envision the new applications raised by quantum technology: IBM, Amazon, Google and Microsoft, for example, are recognizing the huge revenue and development potential and are investing heavily in research on how to explore entangled quantum states and superposition, through partnerships with universities (CDOTRENDS, 2022), launching new processors with more *qubits*[1] (IBM RESEARCH BLOG, 2021) and promoting international competitions and trainings (IBM RESEARCH BLOG, 2022).

In May 2016, European scientists signed *Quantum Manisfesto,* an initiative to promote coordination between academia and industry to research and develop quantum technologies in Europe ("Quantum Manifesto A New Era of Technology", 2016). The goal was to draw the attention of politicians in the region to the fact that Europe risks being left behind in the research and development of quantum technologies: China currently dominates the field of quantum communication and U.S. companies lead the development of quantum hardware. The objective of the manifesto was achieved and soon the European Union commission decided to promote a research project in quantum technologies of one billion euros per year for the next ten years focusing on four areas: communication, computing, sensors and simulations, with the ultimate goal of developing a European quantum computer (JAEGER, 2018).

Just a few months later, in August 2016, China launched the world's first quantum satellite, developed to establish secure quantum communications transmitting quantum keys from space to Earth (YAN, 2016). Since then, the country has directed its efforts to improve security in the transmission of information. On April 6, 2022, an article was published in which chinese scientists broke the record for direct communication distance with quantum security (QSDC), exceeding 100 km of distance between the issuing source and the receiver (ZHANG et al., 2022).

---

[1] The *qubit* is the elementary unit of quantum computing, analogous to bit of classical computing. Through it, basic computational operations are performed, as will be discussed later.

In the US, research has focused on hardware enhancement. One of the new discoveries in the area is to use qubits made of silicon. In a recent study (MILLS et al., 2021), an unprecedented level of *fidelity*[2] was achieved using a logical port with two of these semiconductor qubits. Surpassing 99%, this is the highest fidelity achieved so far for a two-qubit gate on a semiconductor, and on par with the best results achieved by competing technologies such as superconducting qubits. Due to the great promise, silicon qubits have recently been manufactured on an industrial scale for the first time (ZWERVER et al., 2022). This demonstration promises to accelerate the use of silicon technology as a viable alternative to other quantum computing technologies.

A way of evaluating investment in a given sector is through the number of patent applications. In **Figure 1** one can see that in 2018 China had almost twice as many patent applications as the United States for quantum technology in general, a category that includes communication and encryption devices. The United States, however, leads the world in requirements related to the most valued segment of the field – quantum computers – thanks to the heavy investment of companies such as IBM, Amazon, Microsoft, etc., (WHALEN, 2019), as illustrated in **Figure 2**.

[2] The *fidelity* is a measure of the ability of a *qubit* to perform operations without errors.

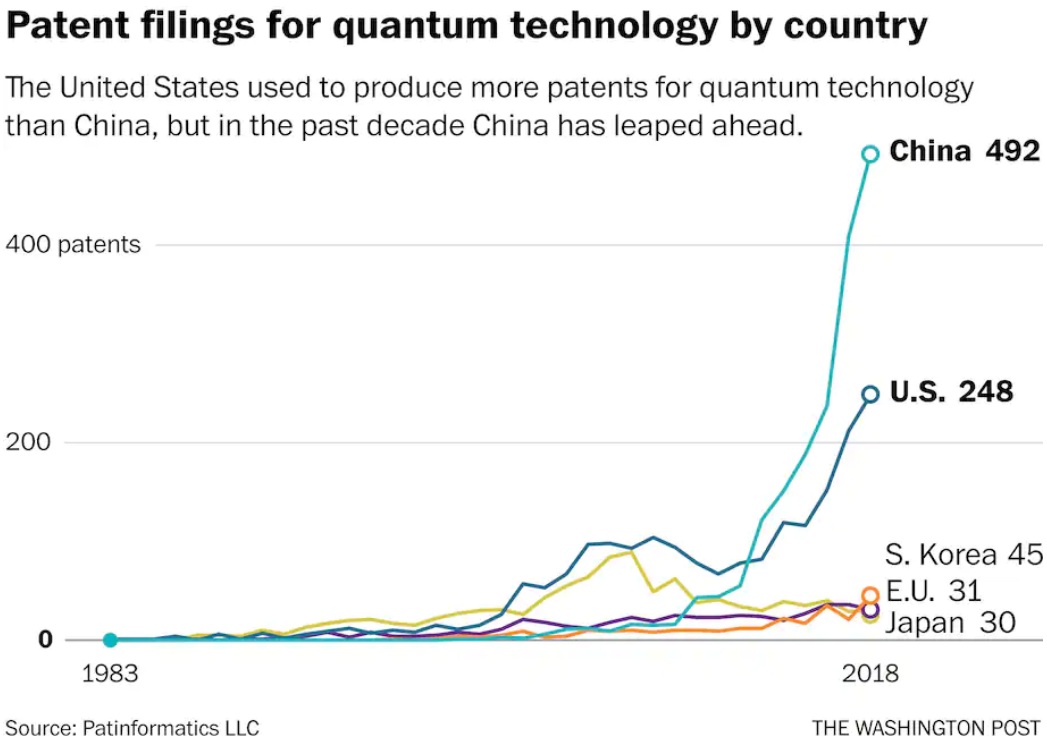

Figure 1. Patent Applications for "Quantum Technology" by country.
Source: (PATINFORMATICS LLC, 2017).

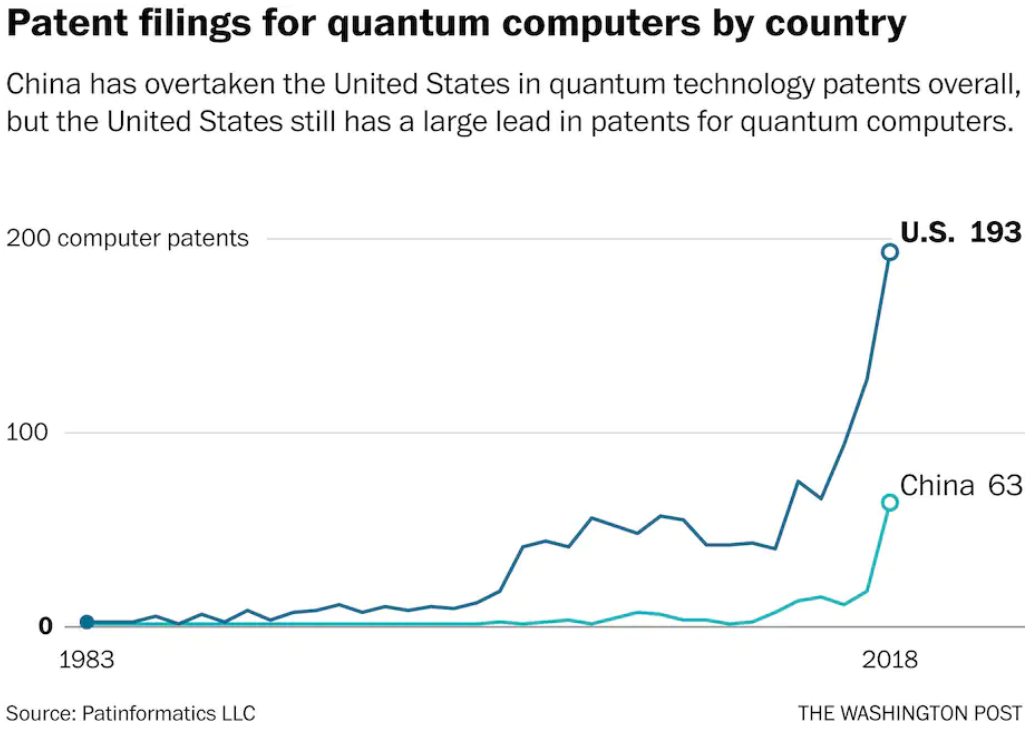

Figure 2. Patent application for "*Quantum Computer*" by country.
Source: (PATINFORMATICS LLC, 2017).

Despite the prominence of these countries, investment in the sector is not a one-off. Several countries are adopting initiatives to encourage these technologies, as can be seen in the survey carried out by QuRECA (QURECA, 2022). It lists the public investments made by each country, which total almost US$ 30 billion, as illustrated in **Figure 3**.

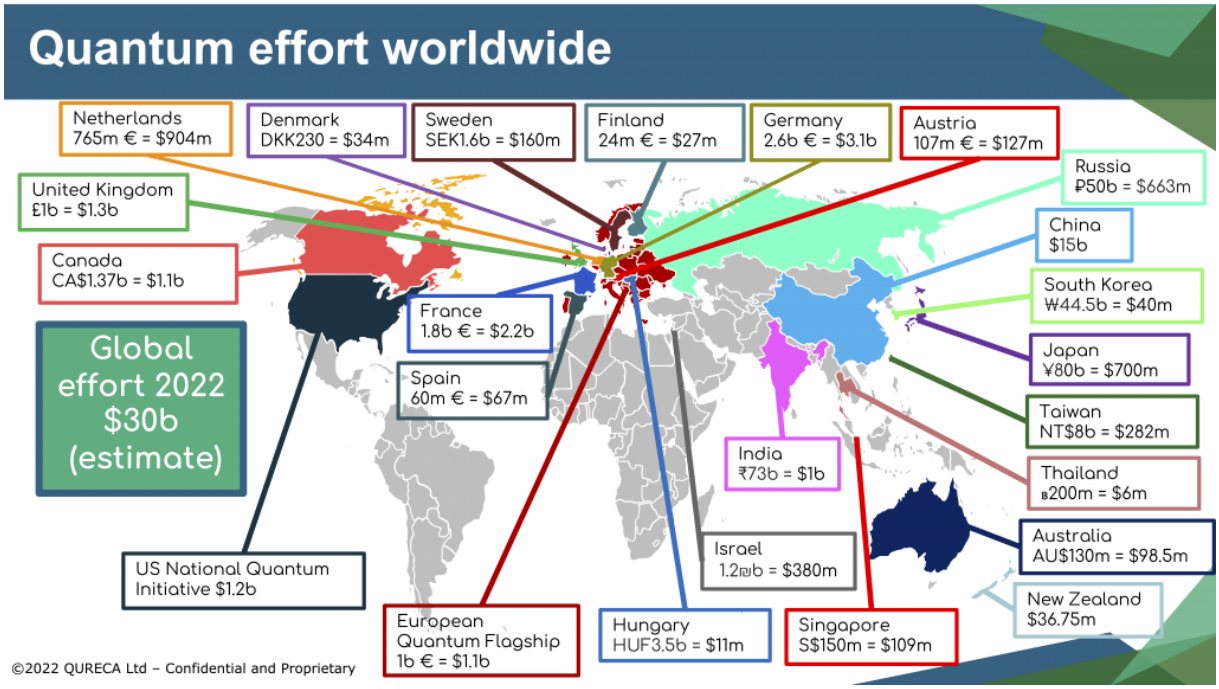

Figure 3. Overview of public funding in quantum technologies.
Source: (QURECA, 2022)

As is evident from the infographic, in Brazil there is still no effort by federal public authorities to adhere to this trend. Despite this, some universities in the country have been establishing partnerships and developing initiatives to try to cover this lack.

1.2 TECHNICAL PROBLEM

Many of the problems that computers solve are types of search problems. A web search using a search engine (such as www.google.com) is nothing more than a program that creates a database from websites and allows the user to search on it. A database can be interpreted as a program that receives an address as input and returns the data contained in that address. A phone book is an example of a database: each entry in the book contains a name and a number. For example, one can ask the database to give the data on the 410th address and it will return the 410th name and number in the book.

To perform a database search in classical computing, it is necessary to check on average $N/2$ until you find the requested address, where N is the number of entries in the database. That is, the time complexity of classical search algorithms is $O(N)$. With quantum computing, this complexity undergoes a quadratic reduction: a quantum hardware, through Grover's search algorithm, performs a search in a database at $O(\sqrt{N})$.

To implement Grover's algorithm, it is necessary to encode database information on a quantum basis. Although it is possible to find works on implementations of Grover's algorithm in Portuguese (MICROSOFT, 2022; PRADO; DILLENBURG, 2014; QISKIT, 2021a) there is a lag in the study of information encoding strategies – a necessary step to transform a classical base into a quantum base. For this reason, this study will focus mainly on this part.

Although quantum computing has great potential to change the way various problems are approached, the great challenge of today's quantum computers, without a doubt, is related to *stability*. The quantum hardwares currently available are noisy – they have a high *quantum noise* rate, which generates a loss of performance in the system. In addition, there is *decoherence*, which causes qubits to lose their superposition and entanglement properties, limiting the complexity of quantum calculations. This generates measurement errors compared to the expected theoretical measurement.

1.3 OBJECTIVES

1.3.1 Primary Objective

The primary objective of this work is to provide a comparative study between the different quantum hardware currently available, analyzing their *performance*, *noise level* and *processing time*, in addition to providing a detailed and reliable research of the implementation of Grover's search algorithm for the Brazilian academic community, thereby seeking to promote the study of quantum algorithms in this environment and contributing to the dissemination of knowledge on the subject.

1.3.2 Secondary Objectives

Secondary objectives of this research are:

a) To present a detailed study of the implementation of Grover's search algorithm, accessible to the public from any area of the Exact Sciences, covering the theoretical foundation, operating mechanisms and logical gates necessary for its implementation.
b) To describe the different information encoding strategies in n-qubit systems that describe a state.
c) To implement Grover's algorithm in the quantum computing environments available from IBM, Microsoft and Amazon.
d) To analyze the performance of quantum hardware provided by these companies, comparing their performance in the implementation of the search algorithm.
e) To detail the entire process so that this study is a reliable basis for future Brazilian academic research in the fields of Exact Sciences.

2 THEORETICAL CONCEPTS

Quantum mechanics is the foundation for understanding quantum computing and quantum information. To understand its postulates and implications, some familiarity with linear algebra concepts is necessary.

2.1 LINEAR ALGEBRA AND THE FUNDAMENTALS OF QUANTUM COMPUTING

Linear algebra is the study of vector spaces and linear operations on those spaces. A good understanding of quantum mechanics is based on a solid understanding of elementary linear algebra (NIELSEN; CHUANG, 2010). Since quantum mechanics here is the motivation for the study of linear algebra, it is interesting to make use of *Dirac's Notation* (DIRAC, 1939). The most common notations, together with their descriptions, are indicated in **Table 2.1**.

The object of study of linear algebra *is vector spaces*. Among them, the most interested in this work is $\mathbb{C}^n$, space containing all n-vectors of complex numbers, *(z_1, …, z_n)*.

Table 2.1 Main notations used in quantum mechanics, known as Dirac's Notations.

Notation	Description
z	Complex number formed by $z \equiv x + iy$ where $x, y \in \mathbb{R}$ and $i \equiv \sqrt{-1}$.
z^*	Conjugated complex of complex number *z*. $(x + iy)^* = x - iy$
$\|\psi\rangle$	Vector *ket.*
$\langle\psi\|$	Vector *bra.* It is the dual vector to $\|\psi\rangle$.
$\langle\varphi\|\psi\rangle$	Internal product between vectors $\|\varphi\rangle$ and $\|\psi\rangle$.
$\|\varphi\rangle\otimes\|\psi\rangle$	Tensorial product of $\|\varphi\rangle$ e $\|\psi\rangle$.
$\|\varphi\rangle\ \|\psi\rangle$	Abbreviated notation for the tensorial product of $\|\varphi\rangle$ e $\|\psi\rangle$.
A^*	Conjugated complex matrix of matrix *A.*
A^T	Matrix transposed from matrix *A.*
$A^\dagger$	Hermitian conjugate or Adjunct of matrix *A.* Where $A^\dagger = (A^T)^*$.

2.1.1 Basis e Qubit

A *basis* of a vector space is a set of vectors $|v_1\rangle, \dots, |v_n\rangle$ with which any vector $|v\rangle$ of this space can be written as a linear combination of the vectors in the set.

$$|v\rangle = \sum_i a_i |v_i\rangle. \tag{2.1}$$

For vector space $\mathbb{C}^2$, required to describe a qubit, a base is given by the set:

$$|v_1\rangle \equiv \begin{bmatrix}1\\0\end{bmatrix};$$

$$|v_2\rangle \equiv \begin{bmatrix}0\\1\end{bmatrix},$$

so that any vector of $\mathbb{C}^2$ space can be written from linear combinations of $|v_1\rangle e\ |v_2\rangle$, through the expression:

$$|v\rangle = a_1|v_1\rangle + a_2|v_2\rangle. \tag{2.2}$$

It is important to highlight that, in general, a vector space has numerous different bases. However, for the object of study of this work, the aforementioned basis is the most usual. This base is so important in quantum computing that it receives special labels, which allude to classical computing:

$$|0\rangle \equiv \begin{bmatrix}1\\0\end{bmatrix};$$

$$|1\rangle \equiv \begin{bmatrix}0\\1\end{bmatrix}.$$

These states are the *computational basis* of quantum computing and form an orthonormal basis for this vector space.

Unlike classical computing, where *bits* only assume values 0 or 1, *qubits* can be in *linear combinations* of these states, called *superposition states* and expressed by the equation:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{2.3}$$

Thus, a quantum vector space, here a *qubit*, is represented by $|\psi\rangle$, where are α e β complex numbers that represent the *amplitude associated* with each state, $|0\rangle$ and $|1\rangle$.

It is possible to note that, in fact, a *qubit* may be in a *continuous* spectrum of states between $|0\rangle$ e $|1\rangle$. Furthermore, given the postulates of quantum theory, it is impossible to know the state of a qubit before measuring it and, after measuring it, the state *collapses* to $|0\rangle$ or $|1\rangle$.

Two important aspects are related to amplitude. For this, α and β must be normalized:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.4}$$

One of them is the *magnitude,* given by $|\alpha|^2$ and $|\beta|^2$, and which represents, respectively, the probability that, after the measurement, the state obtained is $|0\rangle$ and $|1\rangle$.

The other is the *relative phase,* determined from the imaginary parts of α and β, which represents the degree to which different computational pathways interfere constructively or destructively.

In this way, when a qubit in the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

is measured, it will collapse to the state $|0\rangle$ in 50% of the times and to $|1\rangle$ in the other times.

Despite its strangeness, qubits are real, and their existence and behavior have already been extensively validated by experiments and various physical systems (NIELSEN; CHUANG, 2010, cap. 7). The most usual way nowadays to obtain a qubit is from different polarizations in photons. Other ways are, for example, by aligning the nuclear spin in uniform magnetic fields and shining light on an atom in a way that changes its state – for example, from the *ground state* to a *excited state,* which can be defined as $|0\rangle$ e $|1\rangle$. The interesting thing is that, when light is incident for an adequate time, the electron can be in a state of superposition or "midway" between $|0\rangle$ and $|1\rangle$, which generates the probabilistic characteristic of qubit states (NIELSEN; CHUANG, 2010).

2.1.1.1 The Bloch sphere

Using the fact that $|\alpha|^2 + |\beta|^2 = 1$, it is possible to rewrite the Equation (2.3) to represent a qubit through the polar form of complex numbers:

$$|\psi\rangle = cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\, sin\left(\frac{\theta}{2}\right)|1\rangle. \tag{2.5}$$

where θ and φ are real numbers. The term $e^{i\gamma}$ that appears in the transformation can be ignored due to the fact that it has no observable effects (NIELSEN; CHUANG, 2010, cap. 1, p. 15). This allows its visualization in a three-dimensional reference system, called *Bloch Sphere*, as illustrated in **Figure 4**.

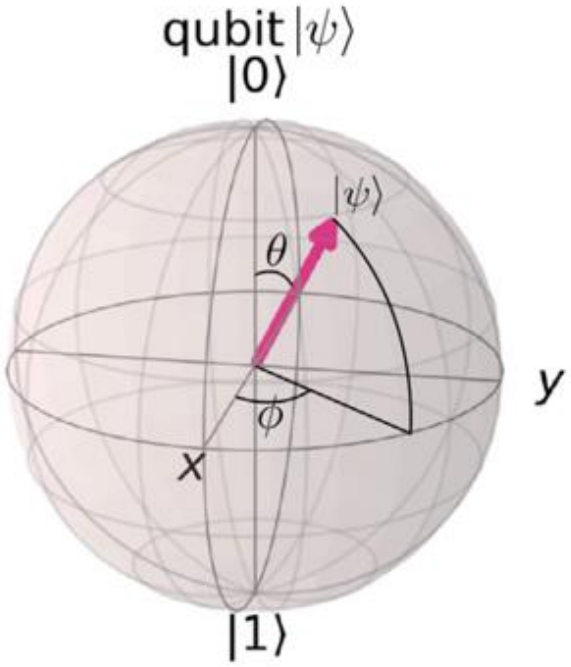

Figure 4. Representation of a generic state in the Bloch sphere.

2.1.2 Linear Operators and Quantum Gates

A *linear operator* is a function that applies a linear transformation of a vector space in itself, transforming each of its vectors linearly (MILLER, 2008), according to:

$$A\left(\sum_i a_i\,|v_i\rangle\right) = \sum_i a_i\,A\,(|v_i\rangle). \tag{2.6}$$

The most convenient way to work with operators is through their matrix representation. Not every linear operator, however, can be a quantum gate: due to normalization, only *unitary* operators can be applied to qubits, as these are *reversible*. They operate as follows:

$$U|\psi\rangle = U[\alpha|0\rangle + \beta|1\rangle] = \alpha U|0\rangle + \beta U|1\rangle. \tag{2.7}$$

There are four matrices that are quantum operators – and, therefore, quantum gates – extremely useful: the *Pauli Matrices*. They are 2×2 matrices and are given special notations according to their applicability, illustrated below along with their form in Dirac's notation:

[1] $I \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1|;$

[2] $\sigma_x \equiv X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|;$

[3] $\sigma_y \equiv Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i|0\rangle\langle 1| + i|1\rangle\langle 0|;$

[4] $\sigma_z \equiv Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|.$

Their representations in quantum mechanics are usually given by Greek letters with indices (σ_x, σ_y, σ_z). For quantum circuits, Arabic letters are preferred (*X*, *Y* e *Z)*.

Operation **[1]** corresponds to *Identity*. It is useful in matrix equation calculations. However, its practical result is null: when applying the matrix to a vector/state, the result is the vector/state itself.

Operation **[2]**, *Pauli-X*, on the other hand, plays a key role in the functioning of quantum computing: it corresponds to the classical logic gate NOT, inverting the state of the qubit from $|0\rangle$ to $|1\rangle$ and vice versa. Graphically, it is possible to interpret the performance of this operator as a rotation of π radians around the x axis, as shown in **Figure 5**.

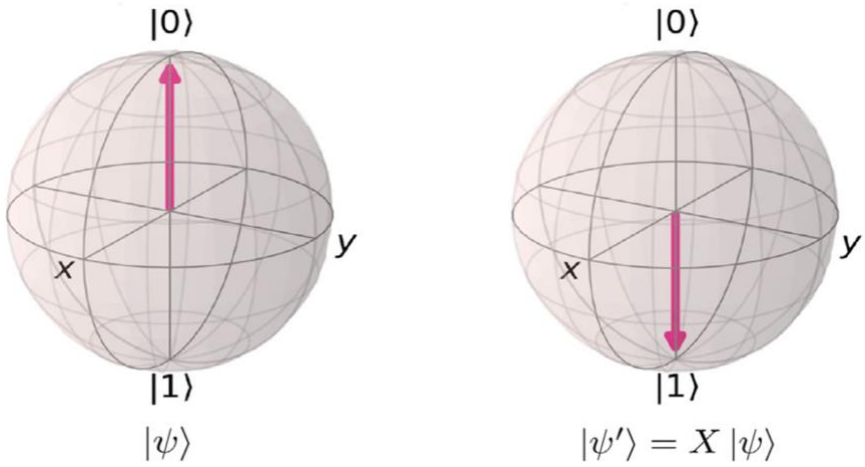

Figure 5. X-gate applied to a qubit in state $|0\rangle$: $X|0\rangle = |1\rangle$.

The matrix **[3]**, *Pauli-Y*, acts by changing both the *state* and the *phase* of the qubit. In a similar way to the previous operator, it works by rotating the state of the qubit in π radians around the *y-axis*.

The last one, *Pauli-Z*[3], leaves the state $|0\rangle$ unchanged, changing only the phase of $|1\rangle$. Like the others, it also rotates the state of the qubit around the *z-axis* by π radians.

[3] It is interesting to note that the elements $\{|0\rangle,|1\rangle\}$ are *eigenstates* of this operator. For this reason, *measurements* of quantum circuits are usually called "*z-measurement*".

Another useful quantum gate is *Hadamard*. It creates the *superposition* state and, consequently, allows *entanglement*. Its unitary operator is:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|).$$

When applying it, for example, to a *qubit* in the state $|\psi\rangle = |0\rangle$, one gets:

$$|\psi'\rangle = \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|)(|0\rangle);$$

$$|\psi'\rangle = \frac{1}{\sqrt{2}}(|0\rangle\langle 0|0\rangle + |0\rangle\langle 1|0\rangle + |1\rangle\langle 0|0\rangle - |1\rangle\langle 1|0\rangle);$$

$$|\psi'\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix}.$$

That is, after applying the quantum operator *H*, the qubit will be in a *superposition* state, expressed by the equation below and which can be seen in **Figure 6**.

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

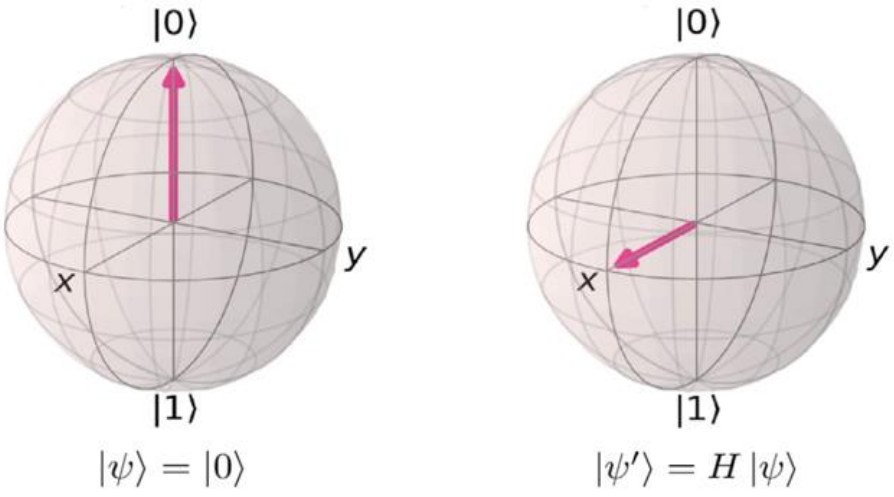

Figure 6. Illustration of Hadamard gate applied to a *qubit* initially in state $|0\rangle$: H$|0\rangle$.

As this state, along with its analogue, are eigenstates of σ_x, they receive special labels:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle); \qquad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

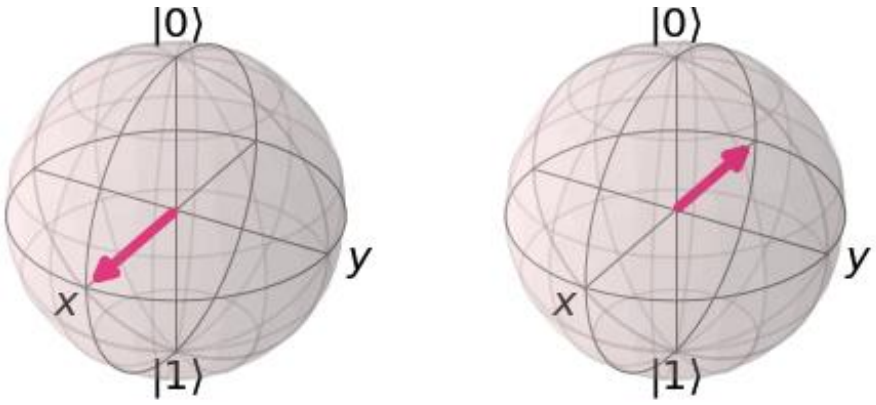

Figure 7. Illustration of states $|+\rangle$ and $|-\rangle$.

Like the eigenstates of σ_x and σ_z, the eigenstates of σ_y also receive specific notations:

$$|+i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle); \qquad |-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle).$$

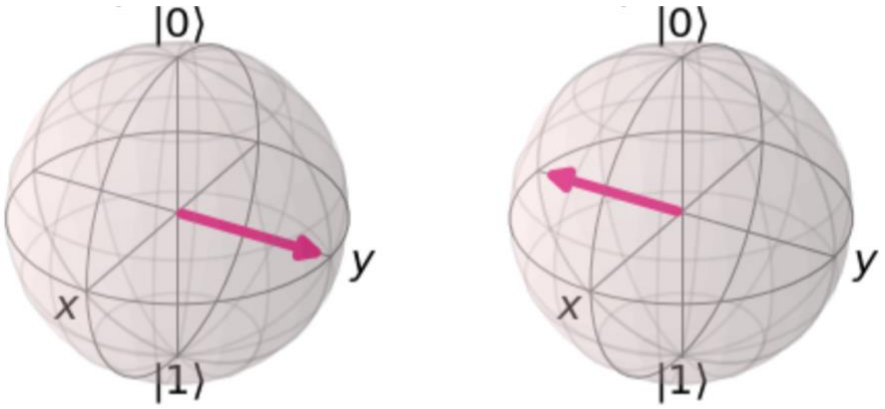

Figure 8. Illustration of states $|+i\rangle$ and $|-i\rangle$.

2.1.3 Multiple *Qubits* and the Bell States

Although a single qubit already has exceptional behavior, when dealing with more than one qubit new possibilities arise. The most remarkable of all, with no doubt, is the *entanglement*. For this, it is necessary to discuss the concept of *quantum register*.

A quantum register of size *n* comprises a quantum system with *n* qubits, where each qubit q_i with $i \in \{0, \dots, n-1\}$ is represented by a unitary vector of *Hilbert space*[4] $\mathcal{H}_i$ with $i \in \{0, \dots, n-1\}$. With this, the resulting quantum register is represented by an *n*-dimensional unitary vector of Hilbert space, computed through the tensor product of the primary vectors:

$$\mathcal{H} = \mathcal{H}_{n-1} \otimes \mathcal{H}_{n-2} \otimes \cdots \otimes \mathcal{H}_0. \tag{2.8}$$

Therefore, just as n bits have 2^n possible states, the *computational basis* for the Hilbert space of n qubits will have dimension 2^n.

In general, the state of a quantum system is expressed by the equation:

$$|\psi\rangle = \sum_{i=1}^{2^n} \alpha_i |i\rangle. \tag{2.9}$$

Thus, given a circuit with 2 qubits, the state of the system is represented by:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle.$$

For example, if a qubit q_A is in the state $|1\rangle_A$ and a qubit q_B is in the state $|0\rangle_B$, the system state is:

$$|\psi\rangle = |1\rangle_A \otimes |0\rangle_B = |10\rangle_{AB} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

[4] The *Hilbert Space* is one variation of complex space that has special properties addressed by Griffiths et. al, (GRIFFITHS; SCHROETER, 2001, p. 118-121).

To obtain an entanglement state, in addition to the Hadamard gate, the CNOT gate (also called CX) is required. The CNOT (Controlled NOT) gate operates on two qubits: a *control* qubit and a *target* qubit. This gate applies the logical NOT operation to the *target qubit* only if the *control qubit* has the value 1. The unitary operator for this gate is as follows:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10|.$$

When applying this operator to a circuit in the state $|00\rangle$, for example, one can notice that nothing happens. But when applying it to the state $|10\rangle$, the state $|11\rangle$ is returned.

However, the most notable result of this operator is achieved when applying it to systems with qubits in superposition states, as for example, the one obtained previously, when applying the Hadamard gate in state $|0\rangle$: $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

If this qubit is in a two-qubit system, where the other was left in the initial state $|0\rangle$, the system state is described by:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle;$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |0\rangle);$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle).$$

Then, it is possible to calculate the result of the CNOT operator applied to the system:

$$|\psi'\rangle = CNOT|\psi\rangle;$$

$$|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10|)(|00\rangle + |10\rangle);$$

$$|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle\langle 00|00\rangle + |01\rangle\langle 01|00\rangle + |10\rangle\langle 11|00\rangle + |11\rangle\langle 10|00\rangle)$$
$$+ \frac{1}{\sqrt{2}}(|00\rangle\langle 00|10\rangle + |01\rangle\langle 01|10\rangle + |10\rangle\langle 11|10\rangle + |11\rangle\langle 10|10\rangle);$$

$$|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

This state, together with its three variations, form an orthonormal basis and are known as the *Bell States*[5]. Because they represent the maximum form of entanglement between two qubits, they receive special notations:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle);$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle);$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle);$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$

[5] Sometimes also called *EPR states*, in honor of Einstein, Podolsky and Rosen – who, along with Bell, were the first ones to notice the peculiar properties of these states.

2.1.4 Relative Phase and Fase Kickback

It's not just X gate that has a controlled version – it's possible to embed the control into any gate. Given a generic operator U, whose matrix is:

$$U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix},$$

the *Controlled-U* operator will be:

$$CU = \begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}.$$

A peculiar behavior occurs in the *Controlled-Z* gate – and Grover's algorithm uses precisely this mechanism. To understand it, it is useful to know that the Z gate is derived from the *Phase* operator, substituting $\lambda = \pi$ in the matrix below:

$$Phase = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix}.$$

Replacing $\lambda = {}^{\pi}/_{4}$ gives T gate, which is equivalent to a rotation of ${}^{\pi}/_{4}$ around z-axis. That is, when applying T gate to a qubit in state $|1\rangle$, a *phase* of ${}^{\pi}/_{4}$ is added to this qubit:

$$T|1\rangle = e^{i\pi/4}|1\rangle.$$

This is a *global phase* and is not observable. But when controlling this operation using another qubit in a superposition state, for example in the state $|+\rangle$, the phase is no longer *global*, but *relative*, which changes the *relative phase* of the control qubit:

$$|+1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |1\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |11\rangle);$$

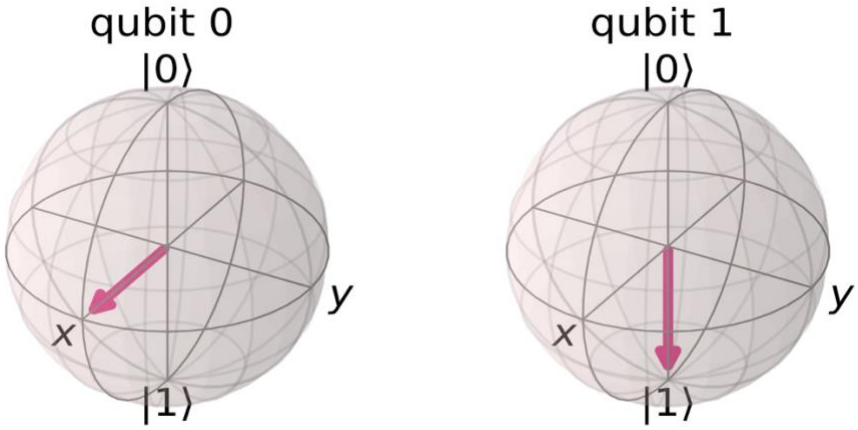

Figure 9. Illustration of the system in the state $|1 +\rangle$.

$$CT|1 +\rangle = \frac{1}{\sqrt{2}}(|01\rangle + e^{i\pi/4}|11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle) \otimes |1\rangle;$$

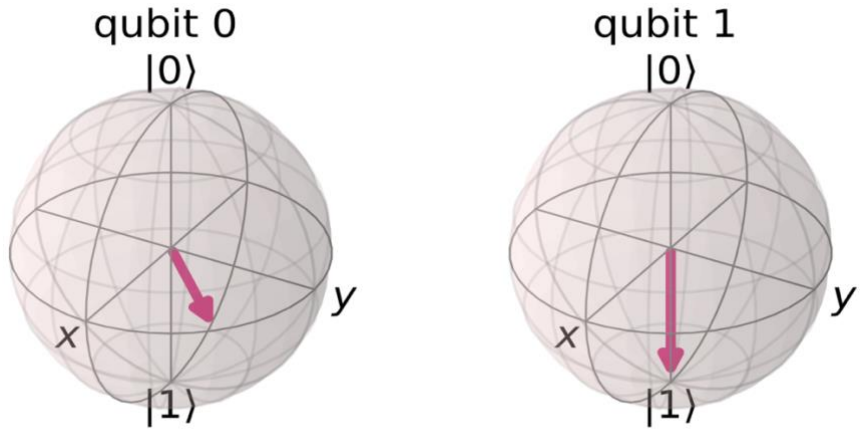

Figure 10. Illustration of the system after applying CT: $CT|1 +\rangle$.

In other words: applying this controlled gate produces the effect of rotating the control qubit around the z-axis of the Bloch sphere, leaving the target qubit unchanged! This effect is called *Phase Kickback* and is critical to Grover's algorithm.

2.2 GROVER'S SEARCH ALGORITHM

One of the most famous quantum algorithms, Grover's algorithm, was proposed by Lov Grover in 1996 (GROVER, 1996). It belongs to the class of quantum search algorithms, which aims to solve the following problem: Given a database of N elements and no prior information about the structure of its information, one wishes to find an element of this database that satisfies a certain condition, a "marked" element. For example, suppose a list of N boxes. Among these boxes, there is a box ω with a unique property that one wants to find: while all other boxes are gray, this is purple, as illustrated in **Figure 11**.

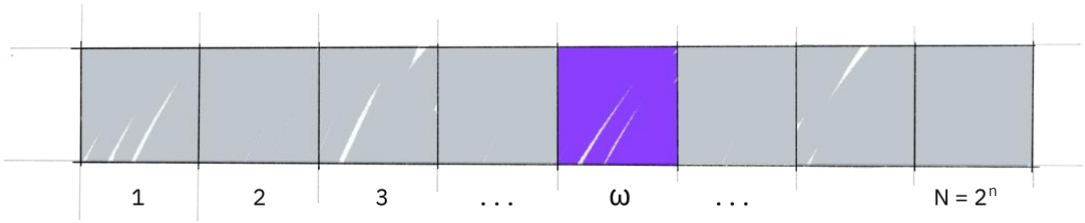

Figure 11. Illustration of N boxes with box ω marked.
Fonte: (QISKIT, 2021b)

This problem is called *Unstructured Search*. To find the purple box – the marked item – using classic computing, it is necessary to check on average $N/2$ boxes and, at worst, all of them – which results in a temporal complexity of $O(N)$. On the other hand, quantum search algorithm requires only approximately $\sqrt{N}$ operations to solve it, through Grover's *amplitude amplification* method. A quadratic acceleration, which represents substantial time savings to find marked items in long lists. Additionally, the algorithm does not use the internal structure of the list, which makes it generic.

2.2.1 Information Encoding

Before discussing the algorithm, it is necessary to perform the encoding of information in a system of *n*-qubits described by a state described by expression (2.9). There are several strategies available for this task that can be divided into 4 types, according to (SCHULD; PETRUCCIONE, 2018a): base encoding, amplitude encoding, qsample encoding and dynamic encoding. **Table 2.2** summarizes these four encoding types.

Table 2.2 Summary of different types of data encoding.

Classic data	Properties	Quantum states
Basis encoding		
$(b_1, \ldots, b_N),\ b_i \in \{0,1\}$	b encodes $x \in \mathbb{R}^N$ in binary	$\|x\rangle = \|b_i, \ldots, b_{N-1}\rangle$
Amplitude encoding		
$x \in \mathbb{R}^{2^n}$	$\sum_{i=1}^{2^n} \|x_i\|^2 = 1$	$\|\psi_x\rangle = \sum_{i=1}^{2^n} x_i \|i\rangle$
$A \in \mathbb{R}^{2^n \times 2^m}$	$\sum_{i=1}^{2^n} \sum_{j=1}^{2^m} \|a_{ij}\|^2 = 1$	$\|\psi_A\rangle = \sum_{i=1}^{2^n} \sum_{j=1}^{2^m} a_{ij} \|i\rangle\|j\rangle$
$A \in \mathbb{R}^{2^n \times 2^n}$	$\sum_{i=1}^{2^n} a_{ii} = 1, a_{ij} = a_{ji}^*$	$\rho_A = \sum_{ij} a_{ij} \|i\rangle\langle j\|$
Qsample encoding		
$p(x), x \in \{0,1\}^{\otimes n}$	$\sum_x p(x) = 1$	$\sum_x \sqrt{p(x)} \|x\rangle$
Dynamic encoding		
$A \in \mathbb{R}^{2^n \times 2^n}$	Unitary matrix A	U_A com $U_A = A$
$A \in \mathbb{R}^{2^n \times 2^n}$	Hermitian A	H_A com $H_A = A$
$A \in \mathbb{R}^{2^n \times 2^n}$	-	$H_{\tilde{A}}$ com $\tilde{A} = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$

2.2.1.1 Basis Encoding

Basis encoding is the simplest and most usual of them all. It associates a classical binary computational basis with the quantum state of *n*-qubits. In this way, each bit is directly replaced by a qubit. That is, basis encoding uses a binary representation to represent real numbers, as is the case in classical computing.

With this type of encoding, the absolute value squared from the amplitudes of the basis states represents the probability of measuring that state. Thus, for implementations that use basis encoding, the goal of the quantum algorithm is to increase the amplitude of the basis state that corresponds to the encoded solution.

There are several ways to represent a real number in binary form, which can be queried in (DA CUNHA, 2017). One of them is the *fraction binary representation*, where each actual number of the range [0.1) is represented by a sequence of τ-bits such that:

$$r = \sum_{k=1}^{\tau} b_k \frac{1}{2^k}, \tag{2.10}$$

where τ represents the accuracy and b the coefficient of the encoding. So, for example, to encode a vector $x = (0.1, -0.4, -1.0)$ in binary representation, with precision $\tau = 4$ and the first bit of the sequence representing the sign, one has:

$$0.1 \rightarrow 0\ 0001;$$

$$-0.4 \rightarrow 1\ 0110;$$

$$-1.0 \rightarrow 1\ 1111.$$

Thus, in quantum computation the x vector would be represented by the state:

$$b = |00001\ 10110\ 11111\rangle.$$

2.2.1.2 Amplitude Encoding

Amplitude encoding, less common in quantum computing, associates classical information with quantum amplitudes. There are several ways to perform this encoding. It is possible, for example, to represent a normalized classical vector $x \in \mathbb{C}^{2^n}$, $\sum_k |x_k|^2 = 1$, by the amplitudes of a quantum state $|\psi\rangle$:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_{2^n} \end{bmatrix} \leftrightarrow |\psi_x\rangle = \sum_{i=1}^{2^n} x_i |i\rangle. \tag{2.11}$$

Analogously, a matrix $A \in \mathbb{C}^{2^n \times 2^m}$, with entries a_{ij} that satisfy the normalization $\sum_{ij} a_{ij} = 1$, can be encoded by:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1j} \\ \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ij} \end{bmatrix} \leftrightarrow |\psi_A\rangle = \sum_{i=1}^{2^m} \sum_{j=1}^{2^n} a_{ij} |i\rangle|j\rangle. \tag{2.12}$$

For Hermitian matrices $A \in \mathbb{C}^{2^n \times 2^n}$, with trace[6] $tr(A) = 1$, there is another option: it is possible to associate their elements with elements of a density matrix ρ_A such that $a_{ij} \leftrightarrow \rho_{ij}$.

A restriction of this method is that only *normalized* classical vectors can be encoded. This implies that by encoding a vector in this way, the quantum states will represent the data in one less dimension – that is, one less degree of freedom. A classical two-dimensional vector (x_1, x_2), for example, can only be associated with an amplitude vector (α_1, α_2) of a qubit that satisfies $|\alpha_1|^2 + |\alpha_2|^2 = 1$, which represents a unit circle (a one-dimensional shape in a two-dimensional space), as illustrated in **Figure 12**.

[6] The trace is defined as the sum of the diagonal elements of a square matrix. Thus, a matrix A of size $n \times n$ will have the trace defined by: $tr(A) = a_{11} + a_{22} + \ldots + a_{nn}$.

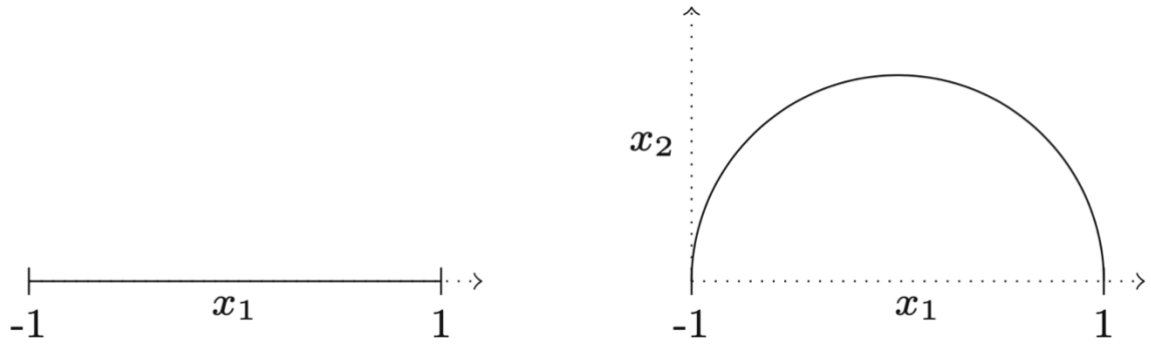

Figure 12. Data points in the one-dimensional range [−1, 1], on the left, can be projected onto normalized vectors by adding a constant value in a second dimension x_2 and renormalizing the resultant vector.
Source: (SCHULD; PETRUCCIONE, 2018b)

A method to get around the loss of degrees of freedom caused by this encoding is to increase the space of the classical vector in a dimension $x_{N+1} = 1$ and normalize the resulting vector. The N–dimensional space will then be embedded in an N+1–dimensional space in which the data are normalized without loss of information (SCHULD; PETRUCCIONE, 2018b).

Thus, to represent the same vector as before $x = (0.1, -0.4, -1.0)$, in amplitude encoding, one first need to normalize it and then pad it with zeros to the appropriate dimension:

$$x' = (0.085, -0.342, -0.855, 0.000),$$

and then represent it by a quantum state of 2 qubits:

$$0.085|00\rangle - 0.342|01\rangle - 0.855|10\rangle + 0.000|11\rangle.$$

It is interesting to note that this same state also encodes the matrix A:

$$A = \begin{pmatrix} 0.085 & -0.342 \\ -0.855 & 0.000 \end{pmatrix}$$

for the case where the first qubit represents the index for the row and the second qubit represents the index for the column.

2.2.1.3 Qsample Encoding

Qsample encoding associates a real amplitude vector $\left(\sqrt{p_1}, \ldots, \sqrt{p_N}\right)^T$ with a classical discrete probability distribution $(p_1, \ldots, p_N)$. This type of encoding can be interpreted as a hybrid case of basis and amplitude encoding, since the information of interest is represented by amplitudes, but the N elements are encoded in qubits:

$$|\psi\rangle = \sum_{i=1}^{2^n} \sqrt{p_i}|i\rangle. \tag{2.13}$$

2.2.1.4 Dynamic Encoding

For some applications it may be useful to encode matrices in dynamic form, for example, in unitary operators. One way is to associate a Hamiltonian H with a square matrix A. If A is not a Hermitian matrix, the following transformation can be used:

$$\tilde{\mathrm{A}} = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}. \tag{2.14}$$

In this way, the eigenvalues of A can be processed in a quantum routine, for example, to multiply A or A^{-1} by an amplitude encoded vector (SCHULD; PETRUCCIONE, 2018b).

2.2.2 The Algorithm

Grover's algorithm consists of three main steps: *state preparation*, *oracle* and *diffusion*, as illustrated in **Figure 13**.

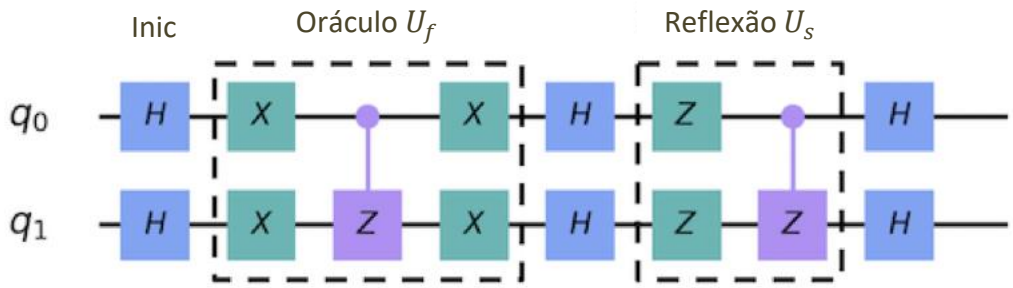

Figure 13. Illustration of Grover's Algorithm circuit for 2 qubits.
Source: (YE, 2020)

The *state preparation* is where the search space is created, which are all possible response cases. In the previously mentioned list example, the search space would be all items in that list. The *oracle* is what marks the correct answer, and the *diffusion* operator amplifies that answer so that it can stand out and be measured at the end of the algorithm. The correct answer may cover more than one item, in which case they are all marked and expanded using this algorithm.

This procedure is called *amplitude amplification* and is the way a quantum computer significantly increases the probability of measuring the marked response. When increasing/amplifying the amplitude of the marked item, there is a decrease in the amplitude of the other items, so the end-state measurement will return the correct item with high probability.

Grover's algorithm has an interesting geometric interpretation, as it produces two reflections that generate a rotation in the two-dimensional plane. The only two states that are necessary to consider are the winner $|\omega\rangle$ and the uniform superposition $|s\rangle$. These two vectors span a two-dimensional plane in the $\mathbb{C}^N$ vector space, as will be shown below.

2.2.2.1 State Preparation

The amplitude amplification procedure begins with the preparation of the state. The uniform superposition $|s\rangle$, can be constructed by applying the Hadamard gate on all qubits, initialized at $|0\rangle$:

$$|s\rangle = H^{\otimes n}|0\rangle^{n}.$$

With this, the state of the system $|s\rangle$ can be expressed by:

$$|s\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle. \tag{2.15}$$

If a measurement were performed on the standard basis $|x\rangle$ this superposition would collapse into any of the states of the basis with the same probability of $\frac{1}{N} = \frac{1}{2^n}$.

Figure 14 shows, on the left, the two-dimensional span generated by the perpendicular vectors $|w\rangle$ and $|s'\rangle$ and which allows expressing the initial state $|s\rangle$ and, on the right, a bar graph of the state amplitudes $|s\rangle$.

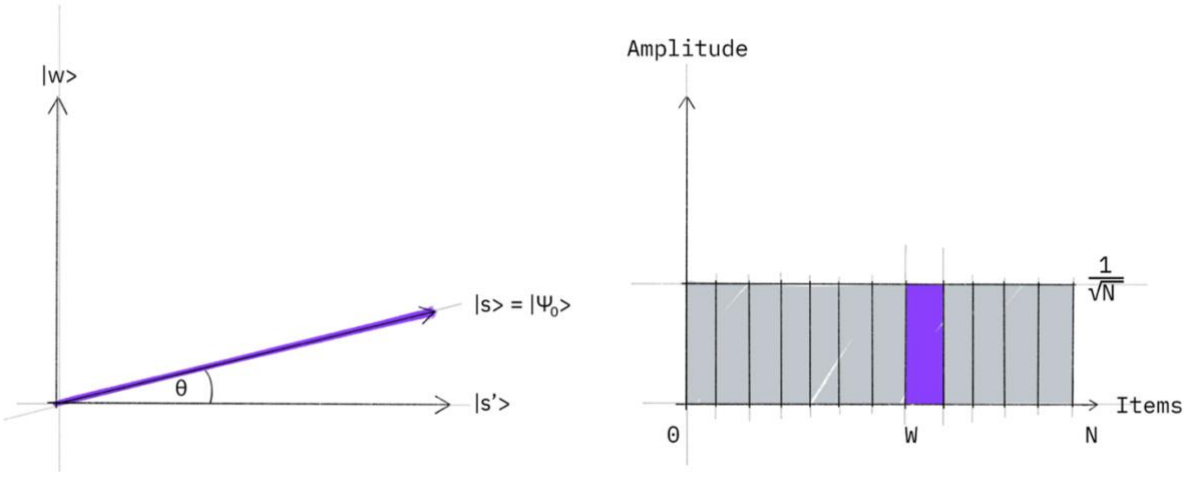

Figure 14. Illustration of the two-dimensional span on the left and the amplitudes bar chart on the right: $|s\rangle$.
Source: (QISKIT, 2021b)

2.2.2.2 Oracle

The next step is to apply the reflection oracle U_f to the state $|s\rangle$. Geometrically, this corresponds to a reflection of the state $|s\rangle$ on $|s'\rangle$. This transformation means that the state amplitude becomes negative, which implies that the average amplitude – indicated by the dashed line in the right image of **Figure 15** – has been reduced.

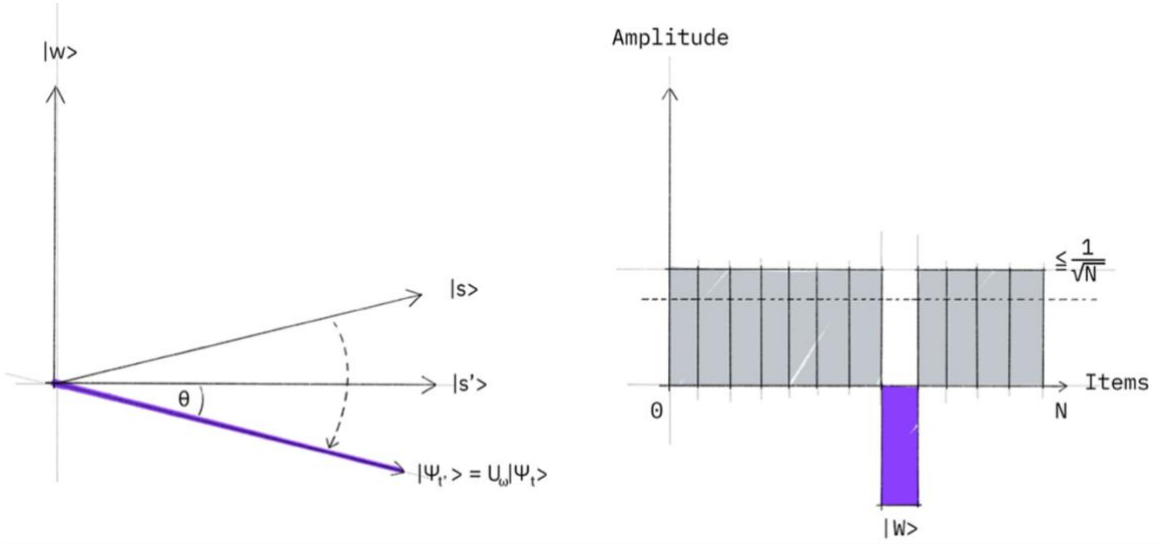

Figure 15. Illustration of the two-dimensional span on the left and amplitudes bar chart on the right: $U_f|s\rangle$.
Source: (QISKIT, 2021b)

Oracles add a negative phase to solution states. This oracle will be a diagonal matrix, where the entry that corresponds to the marked item will have a negative phase. That is, for any state $|x\rangle$ in the computational basis:

$$U_\omega|x\rangle = \begin{cases} |x\rangle & se\ x \neq \omega; \\ -|x\rangle & se\ x = \omega. \end{cases} \tag{2.16}$$

Thus, the oracle operator is the CZ gate that can be represented by an $N \times N$ matrix, whose main diagonal elements are all equal to 1, except for the last element, which has a negative sign, to mark the desired state:

$$U_f = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -1 \end{pmatrix}. \tag{2.17}$$

2.2.2.3 Diffuser

In this step, an additional reflection U_s is applied to the state $|s\rangle$, where:

$$U_s = 2|s\rangle\langle s| - \mathbb{1}. \tag{2.18}$$

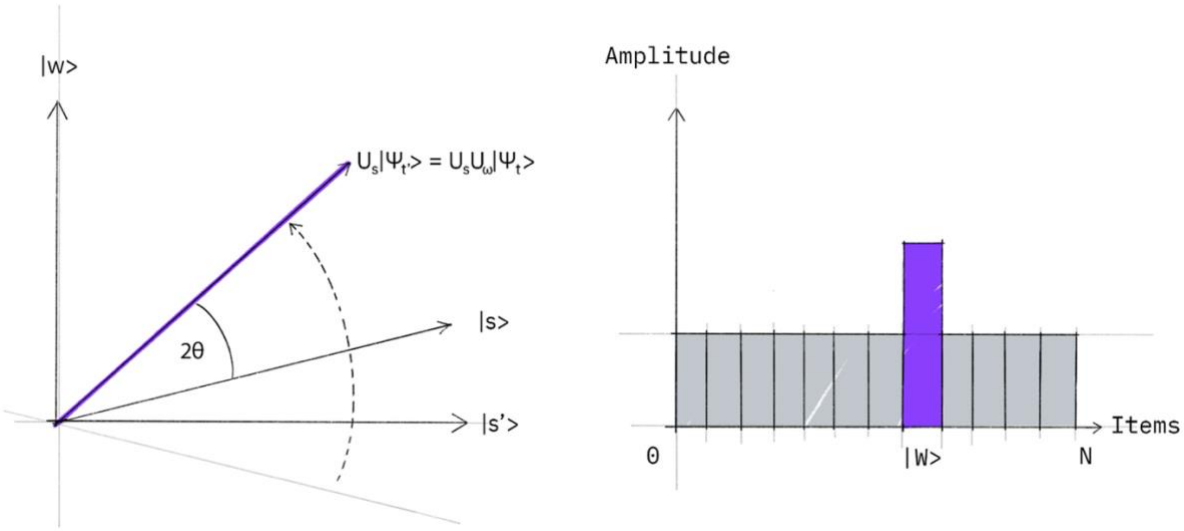

Figure 16. Illustration of the two-dimensional span on the left and amplitude bar chart on the right: $U_s U_f|s\rangle$.
Source: (QISKIT, 2021b)

This transformation maps the state $|s\rangle$ to $U_s U_f|s\rangle$ and completes the transformation.

2.2.2.4 Iterations

Two reflections result in a rotation. In other words, the application of $U_s U_f$ rotates the initial state $|s\rangle$ closer to the winning state $|\omega\rangle$. This procedure must be repeated t times and the state $|\psi_t\rangle$ will be obtained in such a way that:

$$|\psi_t\rangle = \left(U_s U_f\right)^t |s\rangle. \tag{2.19}$$

According to (SCHULD; PETRUCCIONE, 2018a), the ideal number of iterations required is expressed by the equation:

$$t = \frac{\pi}{4}\sqrt{\frac{N}{m}}. \tag{2.20}$$

Where $N = 2^n$ is the size of the search space (with $n = n^{\underline{o}}$ of qubits) and m is the number of answers to be searched. **Table 2.3** shows the ideal number of iterations for a search space built from 2, 3, 4 and 5 qubits and with $m = 1$, that is, a single winning state. Rounding cases are adjusted to the smallest integer.

Table 2.3 Calculation of the ideal number of iterations for 2, 3, 4 and 5 *qubits* with a single winning state.

Number of qubits	**Number of iterations *t***
2	$t = \frac{\pi}{4}\sqrt{2^2} = 1{,}57 \approx 1$
3	$t = \frac{\pi}{4}\sqrt{2^3} = 2{,}22 \approx 2$
4	$t = \frac{\pi}{4}\sqrt{2^4} = 3{,}14 \approx 3$
5	$t = \frac{\pi}{4}\sqrt{2^5} = 4{,}44 \approx 4$

2.3 PERFORMANCE ANALYSIS

Benchmarks for conventional computers are standardized methods that test and evaluate computer *hardware*, *software*, and *systems*. The results of these tests are expressed through metrics that measure system resources and behaviors, such as speed and accuracy. For quantum computers, new benchmarks are needed to address these same metrics while also taking into account differences in underlying technologies and computational models (IEEE QUANTUM, 2019).

System benchmarks measure the fundamental characteristics of a quantum computer. These measurements allow assessments of machine performance without considering specific use cases and provide a clear record of progress. *Application benchmarks*, on the other hand, provide a more comprehensive view of a quantum computer's performance in specific tasks. They can help end users and investors assess the performance of an entire system and measure performance based on real-world use cases, providing significant value beyond just measuring quantum advantage (LANGIONE et al., 2022).

Currently, some benchmark tests for quantum hardware are already starting to be developed to test the quality of qubits (PIRES, 2021). There are even companies specializing in this market[7]. According to (LANGIONE et al., 2022):

> Effective use of benchmarks can be an important source of competitive advantage for investors and end users

Also according to the authors:

> Performance benchmarking – assessing the absolute and relative capabilities of different platforms or systems – has proved useful in assessing other deep technologies. We believe performance benchmarking can accelerate progress in quantum computing. The key is designing benchmarks that are useful (they tell users what they need to know), scalable (they can expand and adapt to evolving technologies), and comprehensive (they cover all the relevant attributes)—tricky business for such a radical and complex technology. That said, a number of

[7] Which is the case of QuantumBenchmark®: https://quantumbenchmark.com/. Access: 2 jul. 2022.

> organizations are already making headway, and we will likely see more benchmarking resources surface as the technology moves closer to market.

That is, good quantum hardware is not made just of the amount of qubits *that* compose it. It is also important that there is *stability* in the system, for example. With this in mind, David P. DiVicenzo published what became known as the DiVicenzo Criteria (DIVINCENZO, 2000), where the requirements for the physical implementation of a quality quantum computer are listed:

1. Scalability: it should be a scalable physical system with its well-characterized parts, usually *qubits*.
2. Initialization: the ability to initialize the state of the qubits to a simple fiducial state.
3. Control: the ability to control the state of the computer through the use of universal elementary logic ports.
4. Stability: long relevant decoherence times and the the ability to suppress this decoherence through error correction and the application of fault-tolerant computing.
5. Measurement: the ability to measure system state on a convenient basis.

According to (GEORGOPOULOS; EMARY; ZULIANI, 2021), within the Noisy Intermediate-Scale Quantum (NISQ)[8] era (PRESKILL, 2018), benchmarking the capabilities and performance of quantum computers when running quantum algorithms is of paramount importance, especially for assess its *scalability*:

> An intuitive approach to benchmarking quantum computers is establishing a set of quantum programs and measuring the performance of a quantum computer When

[8] Corresponds to the current era of quantum computing, with computers of 50-100 qubits which are limited by noise in quantum gates.

> executing each one. Such a work gains more merit as bigger quantum computers are built. (...)
> The different competing quantum technologies pose a major challenge. The technologies have different topologies and thus have unique strengths and weaknesses. For example, the connectivity of an ion-trap computer provided a large advantage on some benchmarks over a superconducting quantum computer (LINKE et al., 2017). (...)
> However, entirely new issues may be introduced when scaling up and it is difficult to say whether performances measured today are good indicators of future performance. For example, IonQ's computer (WRIGHT et al., 2019) has all 11 qubits fully-connected. This configuration is possible at this scale, but this might not be true for a system with hundred or a thousand qubits.

In **Figure 17** it is possible to see how the construction of the 11-qubit IonQ processor, mentioned by the author, is. And **Figure 18** illustrates the architecture of some of IBM's processors, for comparison.

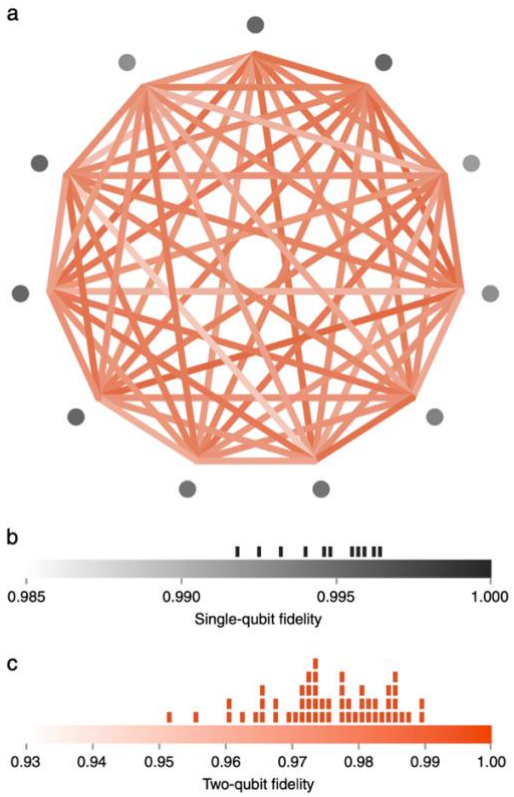

Figure 17. Illustration of IonQ processor construction architecture.
Source: (WRIGHT et al., 2019)

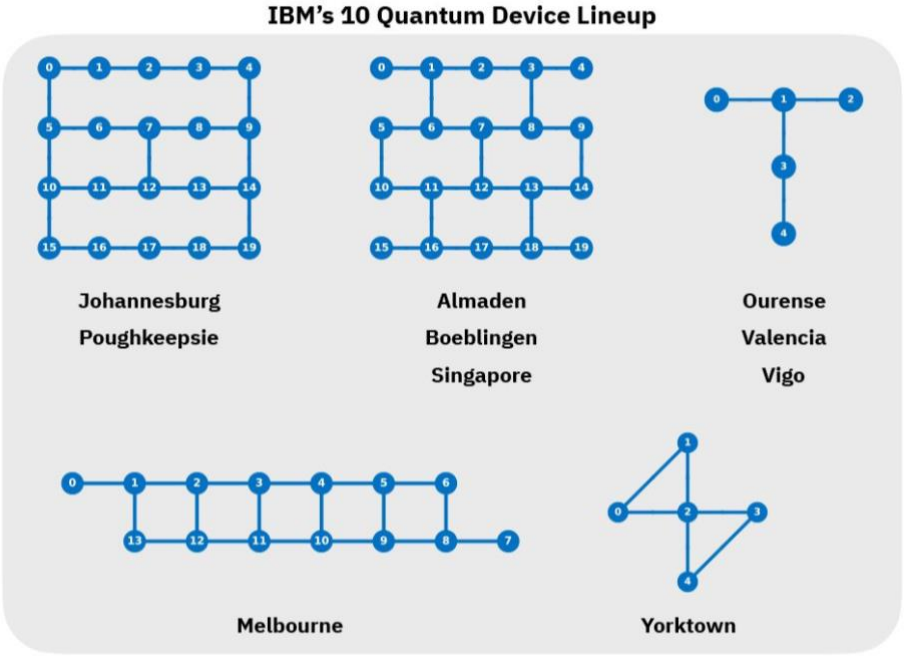

Figure 18. Illustration of the construction architecture of some of IBM's processors.
Source: (IBM RESEARCH, 2019)

2.3.1 Algorithms for Benchmarking

Choosing quantum algorithms for benchmarking is crucial. These algorithms should be chosen according to three main characteristics that are interesting for benchmarking:

1. Scalability : the algorithm must be able to increase complexity (and decrease) so that it can be run on increasingly larger quantum systems.
2. Predictability: the algorithm must produce a result that is easily predictable. An important addition to predictability is susceptibility to noise: the algorithm must provide a result whose distortion under the effects of noise is easily distinguishable from optimal evolution.
3. Quantum advantage: The algorithm must provide computational acceleration over its classical contrapart or, in other words, represent a possibly relevant real-world application.

According to (GEORGOPOULOS; EMARY; ZULIANI, 2021), *quantum search* (or QS) represents an ideal algorithm for benchmarking quantum computers because:

1. The algorithm can scale to search for an item in a larger database by simply adding qubits to the relevant quantum register.
2. The result is easily predictable, as it is simply the item sought, as well as highly susceptible to noise (e.g., a wrong result may appear due to noise).
3. In addition, QS can accelerate an unstructured search problem in a quadratic way, making it a very attractive application for quantum computers.
4. And yet, Grover's algorithm can serve as a subroutine to achieve quadratic runtime improvements to a variety of other algorithms through amplitude amplification (SCHULD; PETRUCCIONE, 2021).

Also according to the authors, there are four characteristics of a quantum circuit that are of interest for benchmarking, which will be explored in this study:

i. The number of gates in the circuit,
ii. The number of active qubits, or qubits that are used by the quantum circuit (also called workspace),
iii. The depth of the circuit, i.e., the longest path between the beginning of the circuit and a measuring gate,
iv. The runtime of the circuit in the quantum processor.

3 METHODOLOGY

For the implementation of the proposed algorithm, five different quantum hardware will be used, made available by IBM, Microsoft and Amazon and the results obtained will be compared at the end of the study. Results will also be generated using Qiskit Aer, a high-performance simulator for studying quantum computing algorithms and applications (QISKIT, 2018).

Although each company has its own languages/libraries for the development of quantum algorithms, all accept the implementation through Qiskit libraries – an open source SDK[9] to work with quantum computers at pulse level, circuits and application modules in Python language, and which will be used in this project.

3.1 PLATFORMS AND HARDWARES

3.1.1 IBM Quantum®

The IBM Quantum[10] platform provides several tools for implementing quantum algorithms, in addition to accessing the company's own quantum hardware. To access it, one can just create a free account, IBMid. Within the platform, there is the option of using the IBM Quantum Composer® to implement quantum circuits graphically or the IBM Quantum Lab®, which provides access to *Jupyter Notebooks* for implementing circuits through algorithms.

The company currently offers free access to eight of its processors, six of them with 5 qubits each (*ibmq_lima*, *ibmq_belem*, *ibmq_quito*, *ibmq_manila*) and two with 7 qubits (*ibm_nairobi*, *ibm_oslo*), in addition to five different simulators. The use of these processors is completely free, with no quota or maximum usage time. The only restriction is regarding the

[9] The acronym SDK means *Software Development Kit*.
[10] Available at: https://quantum-computing.ibm.com/. Access: 24 jun. 2022.

number of *shots*[11] per task, with 20,000 being the maximum allowed. There are processors with a greater number of qubits, but with paid access. All company's processors are built using superconducting qubit technology.

The hardware used in this study was *ibmq_belem*, due to the shorter waiting time in the queue on the algorithm execution date. **Figure 19** shows its build architecture.

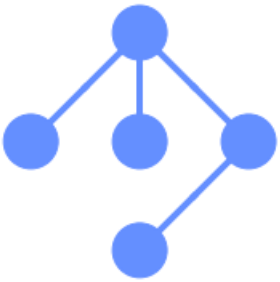

Figure 19. Illustration of IBM's processor construction architecture, *ibmq_belem*.

3.1.1.1 IBM Quantum Composer®

One can easily build quantum circuits graphically with the IBM Quantum Composer. It allows users to dynamically see some useful properties of the circuit, such as measurement result probabilities, state vectors, phase, and *Q-sphere* (a generalization of the Bloch sphere for systems with 2 or more qubits). In addition, it provides the circuit implementation in Python code.

3.1.1.2 IBM Quantum Lab®

The IBM Quantum Lab provides a collection of tutorials on Jupyter Notebooks created by the Qiskit® team. In addition, you can create and run your own notebook using Qiskit libraries. Due to the greater versatility of this environment, this option will be used for the implementation of the algorithm in the company's quantum hardware.

---

[11] A *shot* is a single execution of an algorithm on a quantum processor.

3.1.2 Microsoft Azure Quantum®

Microsoft's cloud quantum circuit development platform is Azure Quantum®. It is possible to use the quantum hardware made available by the company – currently available processors from IonQ, Quantinuum and Rigetti – through Jupyter Notebooks. The company allows the use of Qiskit libraries in Python, although it has developed the Q# language – created by the company specifically to work with quantum algorithms.

Although there are costs to run circuits on processors made available by Microsoft, when creating an account on the Azure platform, the company provides a credit of $500 for each of the quantum processors.

3.1.2.1 IonQ

The IonQ Harmony is a trapped ion quantum computer and is dynamically configurable to use up to 11 qubits. All qubits are fully connected as illustrated in **Figure 17**, which means that it is possible to run a two-qubit gate between any pair (AZURE QUANTUM, 2022).

Also available is IonQ Aria, the company's latest generation of trapped ion quantum computer. With a system also dynamically configurable, but with 23 qubits. However, due to a bug in the integration of the Azure platform to the IonQ system, until the end of this study, there was no success in running the circuit on this processor.

3.1.2.2 Quantinuum

Quantinuum provides access to high-fidelity trapped ion systems and fully connected qubits. The company's H1 system model generation of quantum computers includes two target computers: H1-1 and H1-2. Both quantum computers are fundamentally the same design and both meet a nominal set of technical requirements (QUANTINUUM, 2022). While H1-1 has 20 qubits, H1-2 has 12 qubits (AZURE QUANTUM, 2022b). On the execution date of the circuits in this study, only H1-2 was available for use.

As one can see in the illustration in **Figure 20**, its construction architecture follows the linear model.

Figure 20. Illustration of Quantinuum's H1 processor build architecture.

3.1.2.3 Rigetti

Although Microsoft makes this processor available in its range of quantum computers, its integration into the platform is recent. On the execution date of this project it was not in operation.

3.1.3 Amazon Braket®

Inspired by the nomenclature of Dirac's notation, this service platform is the one that gives access to the largest number of quantum processors from different companies: hardwares from D-Wave, IonQ, OQC (Oxford Quantum Circuits), Rigetti and Xanadu are available. However, there is no modality of free access to computers for the general public, and the *Pay As You Go* mode is the company's choice of offer. **Table 3.1** shows the values practiced by Amazon for each available processor, at the time of this study.

Although it is the company with the widest range of quantum hardware options from different companies, the service is not yet fully integrated with the Qiskit SDK, requiring the implementation of the algorithm through the *Amazon Braket Python SDK*® (AMAZON, 2022a) to use the most processors. However, in June 2022 the company announced the integration of part of the hardwares to Qiskit, namely: Rigetti, OQC and IonQ (AMAZON, 2022b). Due to

the project's algorithm being using Qiskit in all other implementations, it was preferred to follow the same SDK.

Furthermore, as the IonQ processor is available from Microsoft Azure® – which offers credit for its use – this hardware was not used through Amazon platform.

Table 3.1 Values practiced by Amazon for each hardware offered.

Hardware Provider	QPU family	Per-task price	Per-shot price
D-Wave	2000Q	$0.30000	$0.00019
D-Wave	Advantage	$0.30000	$0.00019
IonQ	IonQ device	$0.30000	$0.01000
OQC	Lucy	$0.30000	$0.00035
Rigetti	Aspen-11	$0.30000	$0.00035
Rigetti	Aspen-M	$0.30000	$0.00035
Xanadu	Borealis	$0.30000	$0.0002

Source: (AMAZON BRAKET PRICING, 2022)

3.1.3.1 Rigetti

Rigetti Aspen-M-2 is an 80-qubit processor based on scalable multi-chip technology and features enhanced readout capabilities that contribute to better overall circuit fidelities independent of depth and width. The Aspen chip topology is octagonal with 3-fold (2-fold for edges) connectivity and features both *CPHASE* and *XY* entangling gates that allow developers

to optimize programs for performance and minimize circuit depth (RIGETTI, 2022), as illustrated in **Figure 21**.

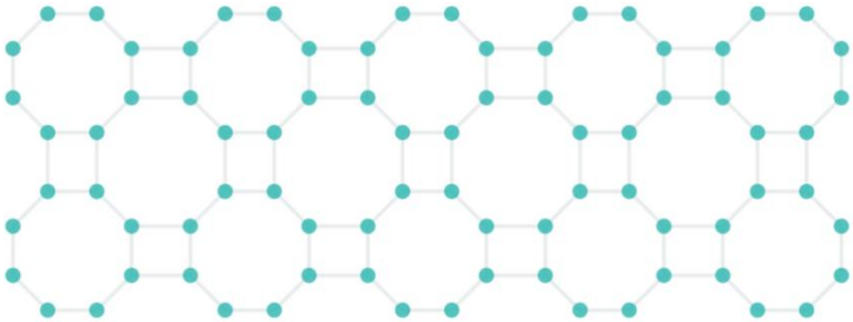

Figure 21. Illustration of Rigetti's Aspen-M processor build architecture.
Source: (RIGETTI, 2022)

3.1.3.2 OQC

Oxford Quantum Circuits (OQC) is a leader in quantum computing in the UK and Europe. Its latest system, Lucy, is a superconducting 8-qubit quantum processor named after Lucy Mensing, a German physicist who pioneered quantum mechanics (AMAZON WEB SERVICES, 2022).

The main technological innovation of the company's processors, *Coaxmon*[12], is a type of qubit that has a three-dimensional architecture and brings key components off the chip for greater simplicity, flexibility, ease of engineering and – crucially – scalability (OXFORD QUANTUM CIRCUITS, 2022).

[12] A short video illustrating this technology can be viewed at: https://oxfordquantumcircuits.com/wp-content/uploads/2021/07/OQC-arch-vid-1.mp4. Access: 25 set. 2022.

3.2 ALGORITHM

Due to the current limitations of quantum hardware, the largest number of qubits available for free, common to all tested processors, is five units. Therefore, circuits with 2, 3, 4 and 5 qubits were executed. In all cases the winning state was unique and corresponding to all bits equal to 1 – in order to facilitate the graphic visualization – represented, respectively, by: $|11\rangle, |111\rangle, |1111\rangle, |11111\rangle$, using the basis encoding. Two types of circuits were built: with a single application of the search routine and with the ideal number of iterations t, obtained by equation (2.20). In addition, three groups of tasks[13] were performed: with 500 shots, with 1000 shots and with 2000 shots.

3.2.1 Algorithm Success Probability

A formula to determine the probability of success of Grover's algorithm from an initial superposition state is provided by (BOYER et al., 1996). According to the authors, the *Algorithm Success Probability* (ASP) of measuring a single target state ω (and the other states s') from the state $|\psi_m\rangle$, after any number of iterations $t = m + 1$ is given by:

$$|\psi_m\rangle = |\psi(\omega_m, s'_m)\rangle,$$

where:

$$ASP(\omega_{m+1}) = \left(\frac{N-2}{N}\omega_m + \frac{2(N-1)}{N}s'_m\right)^2; \quad (3.1)$$

[13] A *task* is a sequence of shots repeated based on the same circuit design.

$$ASP(s'_{m+1}) = \left(\frac{N-2}{N} s'_m - \frac{2}{N}\omega_m\right)^2, \quad (3.2)$$

where $N = 2^n$, with $n = n^{\underline{o}}$ of qubits, and $\omega_0 = s'_0 = \frac{1}{\sqrt{N}}$.

Table 3.2 presents the theoretical measurement and amplitude probabilities of the winning state of Grover's search algorithm for each number of qubits executed. As one can see, for 2 qubits, one iteration is enough to reach 100% probability of measuring the chosen state. As for 3 qubits, after 2 iterations there is a 94.53% probability; for 4 qubits, it takes 3 iterations to reach 96.13%; finally, for 5 qubits, a 99.92% probability of measuring the chosen state is reached after 4 iterations.

Table 3.2 Theoretical ASP values for each number of qubits and iterations, calculated from (3.1) and (3.2).

2 qubits		3 qubits		4 qubits		5 qubits	
n	2	n	3	n	4	n	5
N	4	N	8	N	16	N	32
Amplitude inicial		**Amplitude inicial**		**Amplitude inicial**		**Amplitude inicial**	
ω_0	0,5000	ω_0	0,3536	ω_0	0,2500	ω_0	0,1768
s'_0	0,5000	s'_0	0,3536	s'_0	0,2500	s'_0	0,1768
Probabilidade após 1ª iteração		**Probabilidade após 1ª iteração**		**Probabilidade após 1ª iteração**		**Probabilidade após 1ª iteração**	
m	0	m	0	m	0	m	0
$ASP(\omega_1)$	1,0000	$ASP(\omega_1)$	0,7813	$ASP(\omega_1)$	0,4727	$ASP(\omega_1)$	0,2583
$ASP(s'_1)$	0,0000	$ASP(s'_1)$	0,0313	$ASP(s'_1)$	0,0352	$ASP(s'_1)$	0,0239
Amplitude após 1ª iteração		**Amplitude após 1ª iteração**		**Amplitude após 1ª iteração**		**Amplitude após 1ª iteração**	
ω_1	1,0000	ω_1	0,8839	ω_1	0,6875	ω_1	0,5082
s'_1	0,0000	s'_1	0,1768	s'_1	0,1875	s'_1	0,1547
Probabilidade após 2ª iteração		**Probabilidade após 2ª iteração**		**Probabilidade após 2ª iteração**		**Probabilidade após 2ª iteração**	
m	1	m	1	m	1	m	1
$ASP(\omega_2)$	0,2500	$ASP(\omega_2)$	0,9453	$ASP(\omega_2)$	0,9084	$ASP(\omega_2)$	0,6024
$ASP(s'_2)$	0,2500	$ASP(s'_2)$	0,0078	$ASP(s'_2)$	0,0061	$ASP(s'_2)$	0,0128
Amplitude Após 2ª Iteração		**Amplitude Após 2ª Iteração**		**Amplitude Após 2ª Iteração**		**Amplitude Após 2ª Iteração**	
ω_2	0,5000	ω_2	0,9723	ω_2	0,9531	ω_2	0,7762
s'_2	0,5000	s'_2	0,0884	s'_2	0,0781	s'_2	0,1132
Probabilidade após 3ª iteração		**Probabilidade após 3ª iteração**		**Probabilidade após 3ª iteração**		**Probabilidade após 3ª iteração**	
m	2	m	2	m	2	m	2
$ASP(\omega_3)$	1,0000	$ASP(\omega_3)$	0,7812	$ASP(\omega_3)$	0,9613	$ASP(\omega_3)$	0,8969
$ASP(s'_3)$	0,0000	$ASP(s'_3)$	0,0313	$ASP(s'_3)$	0,0026	$ASP(s'_3)$	0,0033
Amplitude após 3ª iteração		**Amplitude após 3ª iteração**		**Amplitude após 3ª iteração**		**Amplitude após 3ª iteração**	
ω_3	1,0000	ω_3	0,8839	ω_3	0,9805	ω_3	0,9471
s'_3	0,0000	s'_3	0,1768	s'_3	0,0508	s'_3	0,0577
Probabilidade após 4ª iteração		**Probabilidade após 4ª iteração**		**Probabilidade após 4ª iteração**		**Probabilidade após 4ª iteração**	
m	3	m	3	m	3	m	3
$ASP(\omega_4)$	0,2500	$ASP(\omega_4)$	0,9453	$ASP(\omega_4)$	0,9084	$ASP(\omega_4)$	0,9992
$ASP(s'_4)$	0,2500	$ASP(s'_4)$	0,0078	$ASP(s'_4)$	0,0061	$ASP(s'_4)$	0,0000
Amplitude após 4ª iteração		**Amplitude após 4ª iteração**		**Amplitude após 4ª iteração**		**Amplitude após 4ª iteração**	
ω_4	0,5000	ω_4	0,9723	ω_4	0,9531	ω_4	0,9996
s'_4	0,5000	s'_4	0,0884	s'_4	0,0781	s'_4	0,0051

3.2.2 Circuit Preparation

The circuits were prepared according to the figures below (**Figure 22** to **Figure 28**). Oracle operator was implemented according to (2.17) and diffuser follows the formula (2.18).

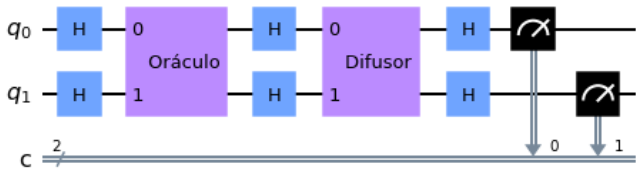

Figure 22. Circuit illustration for 2 qubits and one iteration – the optimal number of iterations.

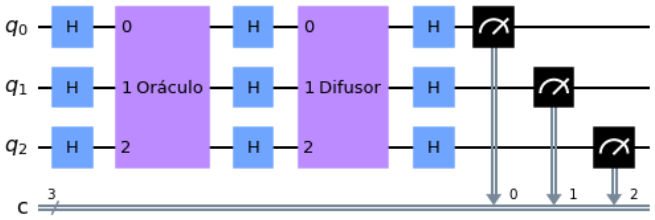

Figure 23. Circuit illustration for 3 qubits and a single iteration.

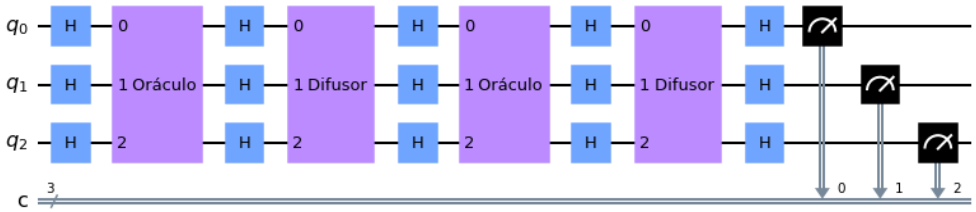

Figure 24. Circuit illustration for 3 qubits and the ideal number of iterations.

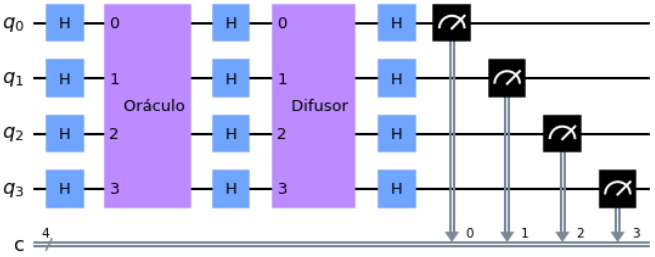

Figure 25. Circuit illustration for 4 qubits and a single iteration.

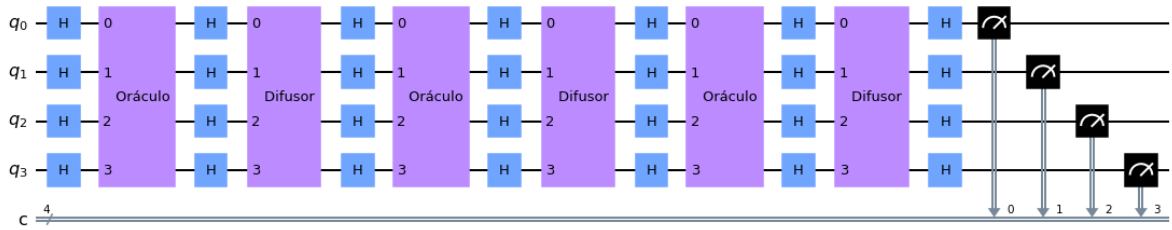

Figure 26. Circuit illustration for 4 qubits and the ideal number of iterations.

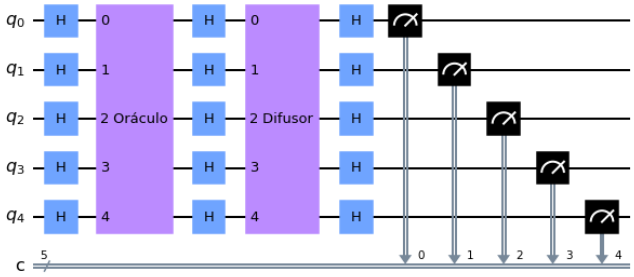

Figure 27. Circuit illustration for 5 qubits and a single iteration.

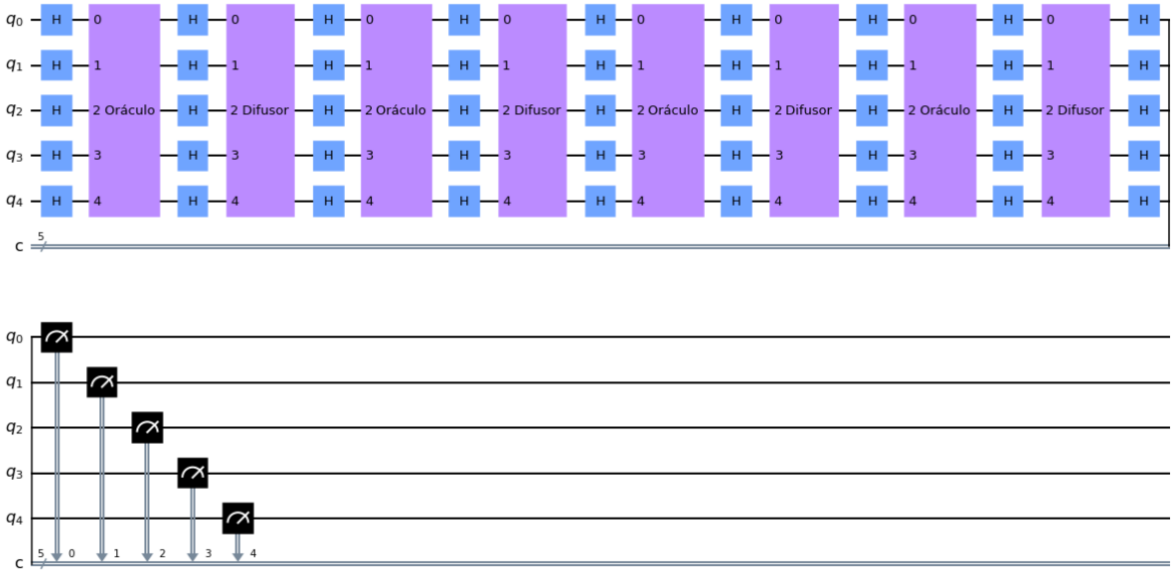

Figure 28. Circuit illustration for 5 qubits and the ideal number of iterations.

3.2.3 Circuit Characteristics

It is important to note that when running an algorithm on quantum hardware, the increase in the number of qubits causes an increase in the number of gates needed for the implementation of the circuit and in its *depth* – which directly influences the noise contained in the final response.

3.2.3.1 Depth

The depth of a circuit is a metric that calculates the longest path between data input and output (GUNZI, 2020). Each gate counts as a unit. An important point to consider is that if one qubit depends on another, one of them has to wait for the other to be computed, so the gates applied to the first qubit count towards its dependent. Thus, the depth of the example circuit in **Figure 29** is 6.

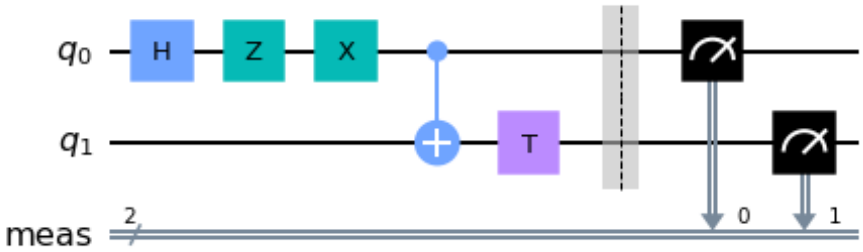

Figure 29. Illustration of a 6-depth circuit.

3.2.4 Transpilation

Transpilation is the process of rewriting a given input circuit to match the topology of a specific quantum device and optimizing the circuit for running in today's noisy quantum systems. Most circuits must go through a series of transformations that make them compatible with a given target device and optimize them to reduce the effects of noise on the results.

Rewriting quantum circuits to match hardware constraints and optimize performance may be far from trivial (QISKIT 0.38.0, 2022). However, Qiskit has pre-built transpilation methods available, which were used in this project.

After transpilation, the number of gates grows considerably, depending on the native gates of each processor, the circuit number of qubits and the hardware architecture.

Table 3.3 presents this information for each implemented circuit.

Figure 30 illustrates the circuit with 2 qubits, transpiled for execution on the IBM hardware and **Figure 31** illustrates the same circuit transpiled for execution on the IonQ processor. While the IBM processor requires 19 gates and a circuit depth equal to 12, the IonQ processor, due to its fully interconnected qubits, the transpiled circuit has 10 gates and a depth equal to 6.

In **Figure 32** one can see the effect of adding a single qubit to the search space: the number of gates required for implementation in IBM hardware grows exponentially for each qubit added to the algorithm's search space. This is not just the case with the IBM processor, as one can see in the graph in **Figure 33**[14] – however, the company's hardware is the one with the highest growth rate of gates per number of qubits. In **Figure 34** it is possible to see that the depth follows the same trend. In addition, it can be noted that the IonQ and Quantinuum processors need far fewer gates for their implementations. This is due to the architecture of building these hardwares, as discussed earlier.

[14] For ease of identification, all graphics in this project will have the following color scheme: purple for IBM processor, blue tones for processors used through Microsoft Azure and orange tones for those used through Amazon.

Table 3.3 Main characteristics of the implemented circuits.

Nº of qubits	**Nº of iterations**	***Hardware***	**Nº of gates**	**Depth**
2	1	Belem (IBM)	19	12
		Harmony (IonQ)	10	6
		H1-2 (Quantinuum)	10	6
		Aspen-M2 (Rigetti)	10	6
		Lucy (OQC)	17	10
3	1	Belem (IBM)	153	109
		Harmony (IonQ)	48	32
		H1-2 (Quantinuum)	47	32
		Aspen-M2 (Rigetti)	82	57
		Lucy (OQC)	111	74
3	2	Belem (IBM)	316	210
		Harmony (IonQ)	82	56
		H1-2 (Quantinuum)	88	61
		Aspen-M2 (Rigetti)	196	140
		Lucy (OQC)	208	141
4	1	Belem (IBM)	725	503
		Harmony (IonQ)	83	57
		H1-2 (Quantinuum)	82	55
		Aspen-M2 (Rigetti)	295	210
		Lucy (OQC)	431	296
4	3	Belem (IBM)	2157	1576
		Harmony (IonQ)	233	165
		H1-2 (Quantinuum)	230	159
		Aspen-M2 (Rigetti)	1252	929
		Lucy (OQC)	868	624
5	1	Belem (IBM)	2962	2090
		Harmony (IonQ)	154	114
		H1-2 (Quantinuum)	151	114
		Aspen-M2 (Rigetti)	1258	892
		Lucy (OQC)	1754	1214
5	4	Belem (IBM)	11977	8470
		Harmony (IonQ)	586	450

		H1-2 (Quantinuum)	574	447
		Aspen-M2 (Rigetti)	4996	3559
		Lucy (OQC)	5028	3567

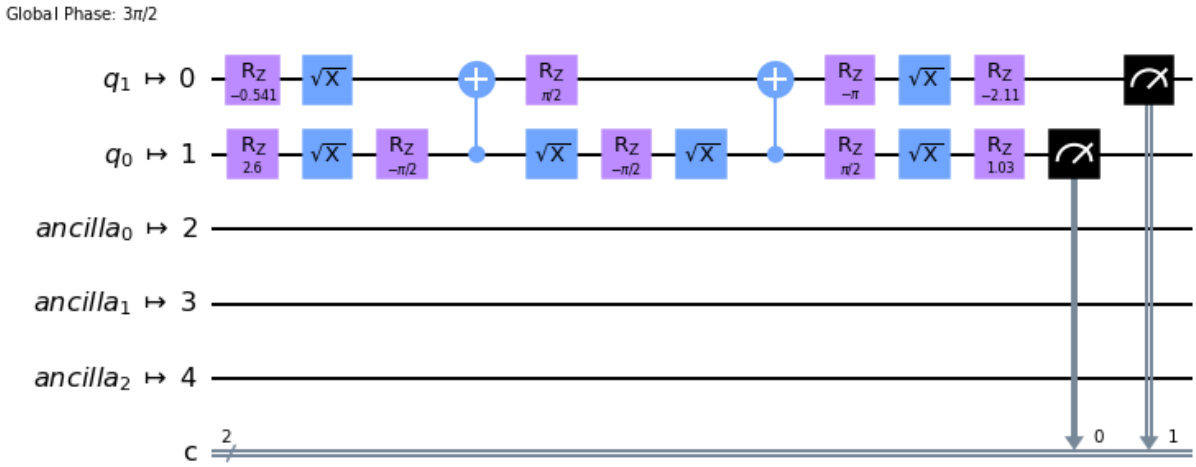

Figure 30. Illustration of the 2-qubit circuit transpiled to run on IBM's Belem hardware[15].

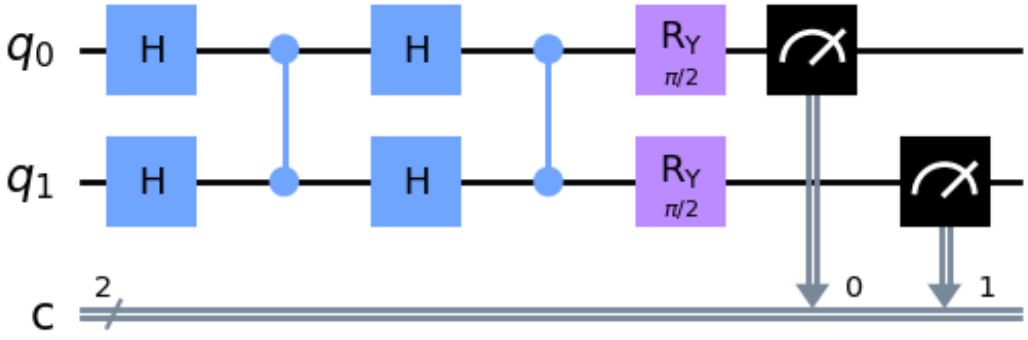

Figure 31. Illustration of the 2-qubit circuit transpiled to run on IonQ's Harmony hardware.

[15] Because this hardware is not dynamically configurable, all your quibits are "called" to tasks. Those not used in the circuit are allocated as *ancilla*. However, one can see that they are not being used in the circuit – there are no gates applied to them.

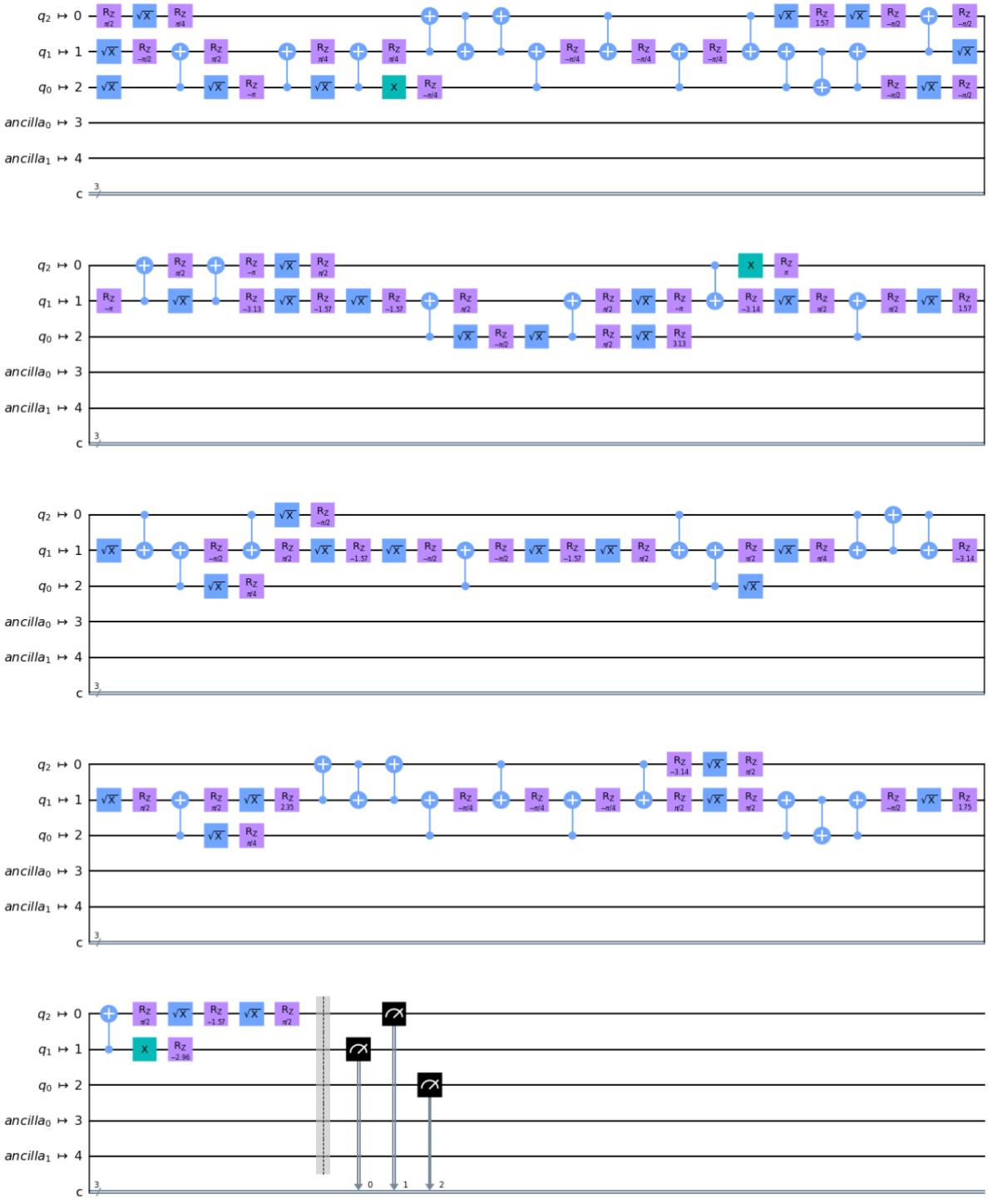

Figure 32. Illustration of the 3-qubit circuit with *t* ideal for running on IBM's Belem hardware.

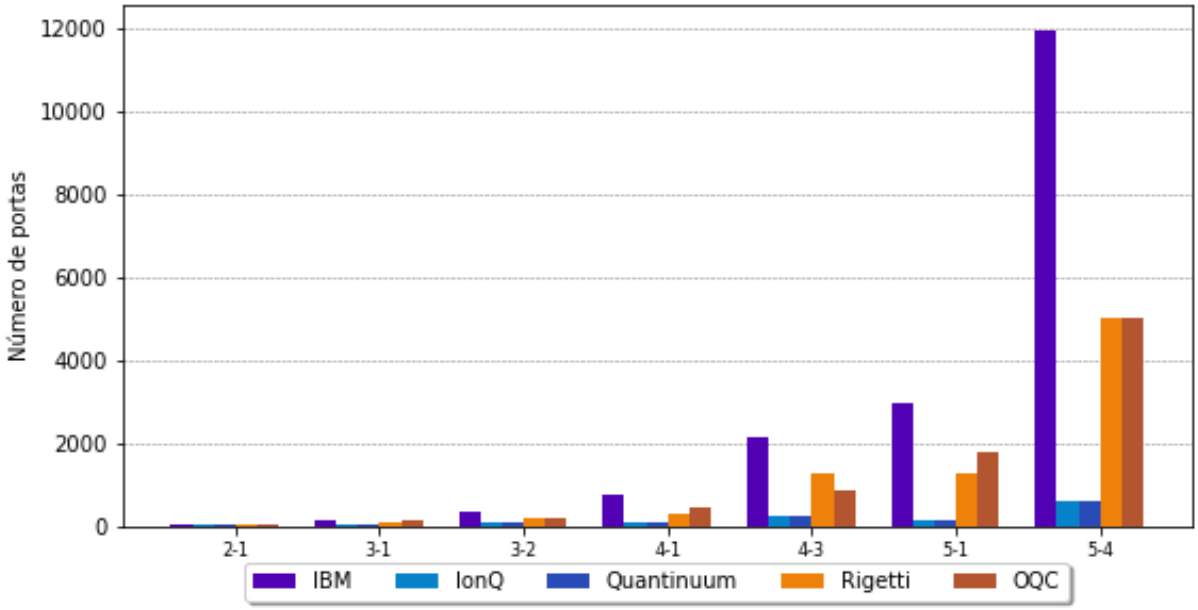

Figure 33. Comparative chart of the number of gates required for implementation on each processor.

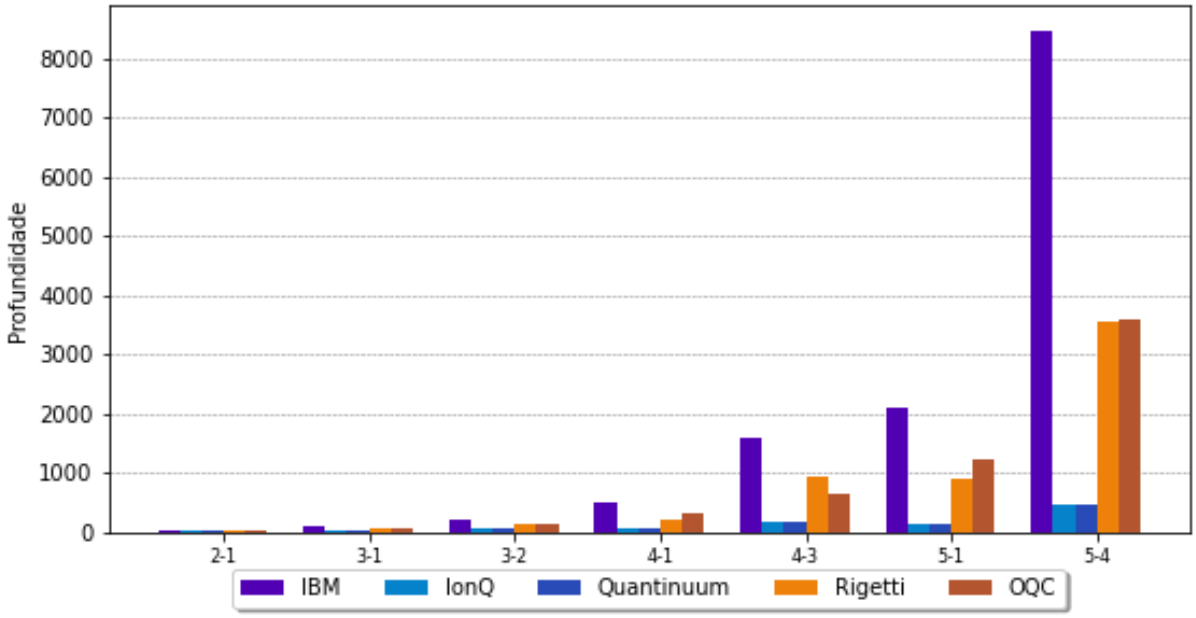

Figure 34. Comparative graph of the depth of the quantum circuit in each processor.

4 RESULTS

The results were separated according to the number of shots executed.

4.1 RESULTS OBTAINED WITH 500 SHOTS

For 500 shots there was success in running the algorithm for hardware from IBM, IonQ and Quantinuum. Due to the high cost of the Quantinuum processor, it was not possible to run the circuit with 4 qubits – for this configuration, the company's emulator was used, which uses a realistic physical model and a noise model of the H1-2 (AZURE QUANTUM, 2022b).

In **Table 4.1** are the performance results for the tested processors, where *Theoretical ASP* is the value calculated analytically, according to **Table 3.2**, *Effective ASP* is the value acquired through the execution of the circuit in the hardware, *Execution time* is the time required to execute the algorithm in the quantum processor and *Total Time* is the time elapsed from sending the task to receiving the results (*Execution time* + *Queuing Time*).

Table 4.1 Processor performance information for 500 *shots*

Nº of qubits	**Nº of iterations**	**Theoretical ASP (%)**	***Hardware***	**Effective ASP (%)**	**Execution time**	**Total Time**
2	*1*	100	Aer Simulator	100	-	-
			Belem (IBM)	89.8	00:00:02	03:31:00
			Harmony (IonQ)	96.0	00:00:04	00:15:45
			H1-2 (Quantinuum)	98.2	00:01:27	11:07:00
3	*1*	78.13	Aer Simulator	78.0	-	-
			Belem (IBM)	23.2	00:00:03	03:31:00
			Harmony (IonQ)	53.6	00:00:05	00:17:06
			H1-2 (Quantinuum)	74.2	00:01:48	11:23:16
3	*2*	94.53	Aer Simulator	94.2	-	-
			Belem (IBM)	23.2	00:00:03	03:31:00

			Harmony (IonQ)	44.2	00:00:06	00:18:57
			H1-2 (Quantinuum)	73.8	00:02:27	11:07:42
4	*1*	47,27	Aer Simulator	48.6	-	-
			Belem (IBM)	4.8	00:00:03	03:31:00
			Harmony (IonQ)	14.4	00:00:07	00:19:20
			H1-2 sim (Quantinuum)	41.2	-	-
4	*3*	96,13	Aer Simulator	97.4	-	-
			Belem (IBM)	5.2	00:00:05	23:19:56
			Harmony (IonQ)	7.0	00:00:14	00:23:45
			H1-2 sim (Quantinuum)	69.4	-	-
5	*1*	25,83	Aer Simulator	23.8	-	-
			Belem (IBM)	1.6	00:00:04	23:19:56
			Harmony (IonQ)	3.2	00:00:11	00:24:49
			H1-2 (Quantinuum)	20	00:33:53	20:09:25
5	*4*	99,92	Aer Simulator	100	-	-
			Belem (IBM)	1.2	00:00:08	02:11:56
			Harmony (IonQ)	2.6	00:00:32	00:26:43
			H1-2 (Quantinuum)	17	00:37:01	19:29:36

In **Figure 35** is the comparison graph of the value of ASP for each processor in each of the implemented circuits. **Figure 36** shows the execution times that each processor took to execute each of the circuits. The total cost to implement on each hardware is described in **Table 4.2**.

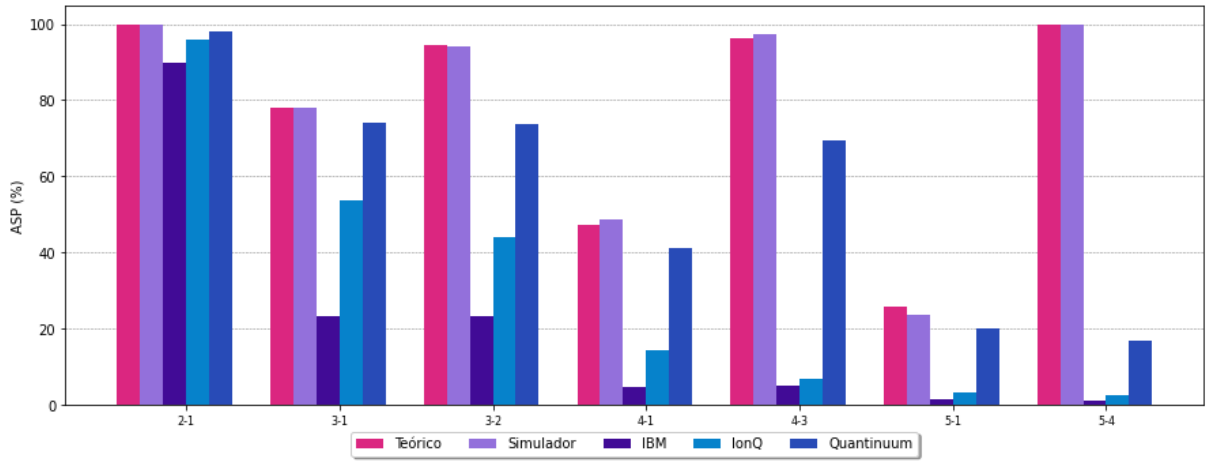

Figure 35. ASP graph obtained on each processor for each of the circuits.

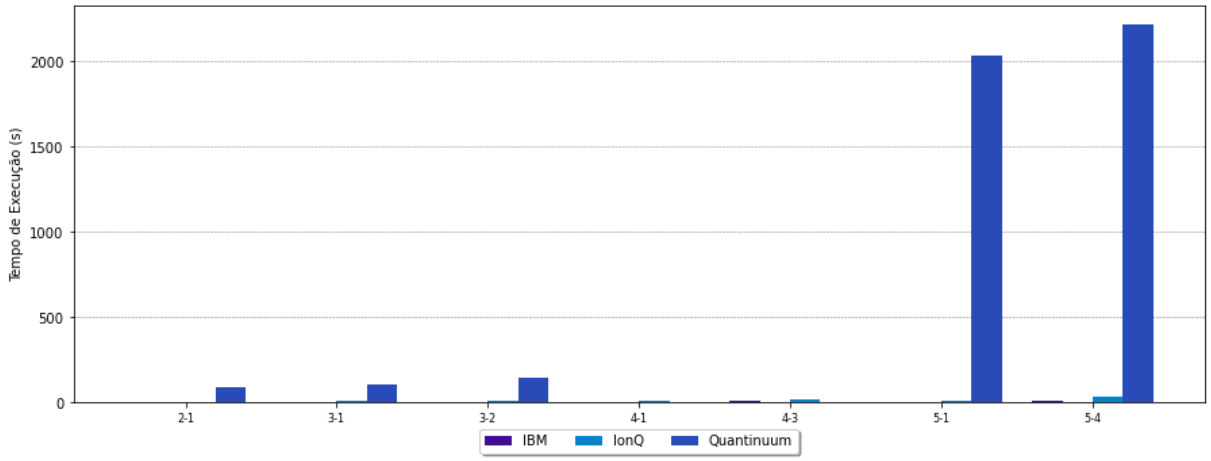

Figure 36. Execution time required by each processor for each of the circuits.

Table 4.2 Total cost for running the circuits.

Hardware	Cost
Belem (IBM)	U$ 0.00
Harmony (IonQ)	U$ 83.92
H1-2 (Quantinuum)	U$ 600.20

4.1.1 Histograms

Following are the histograms generated for each of the circuits implemented with 500 shots.

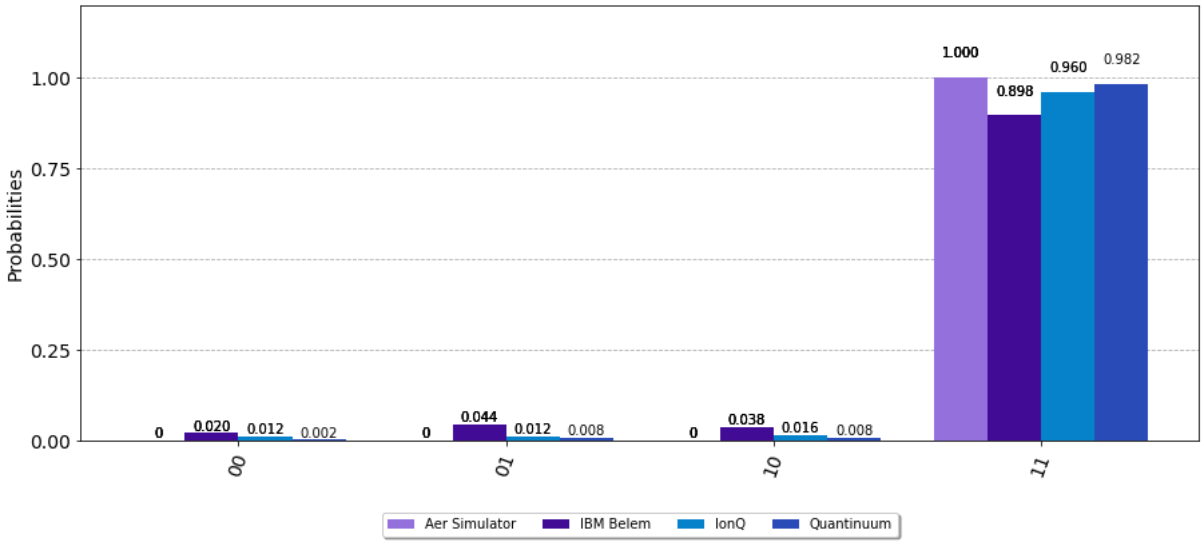

Figure 37. Histogram with results for 2 qubits.

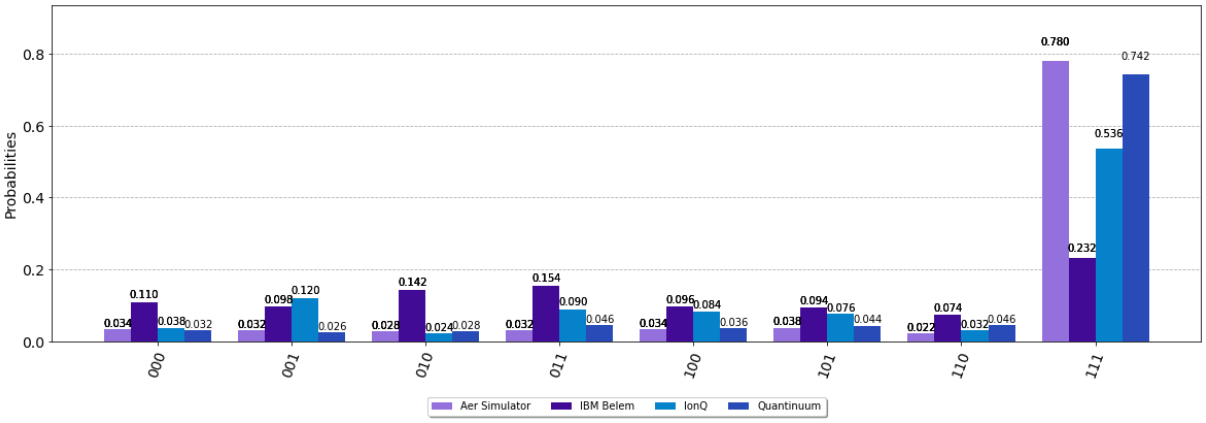

Figure 38. Histogram with results for 3 qubits and 1 iteration.

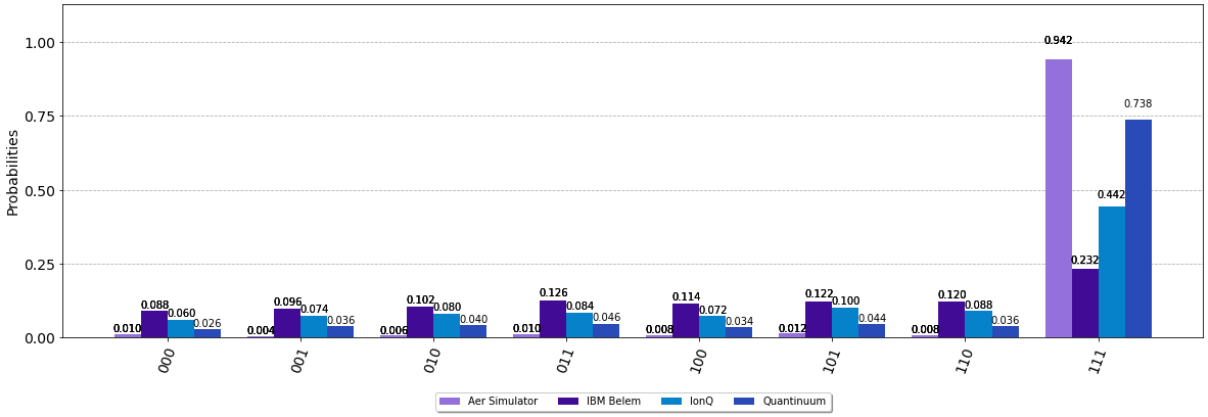

Figure 39. Histogram with results for 3 qubits and number of optimal iterations.

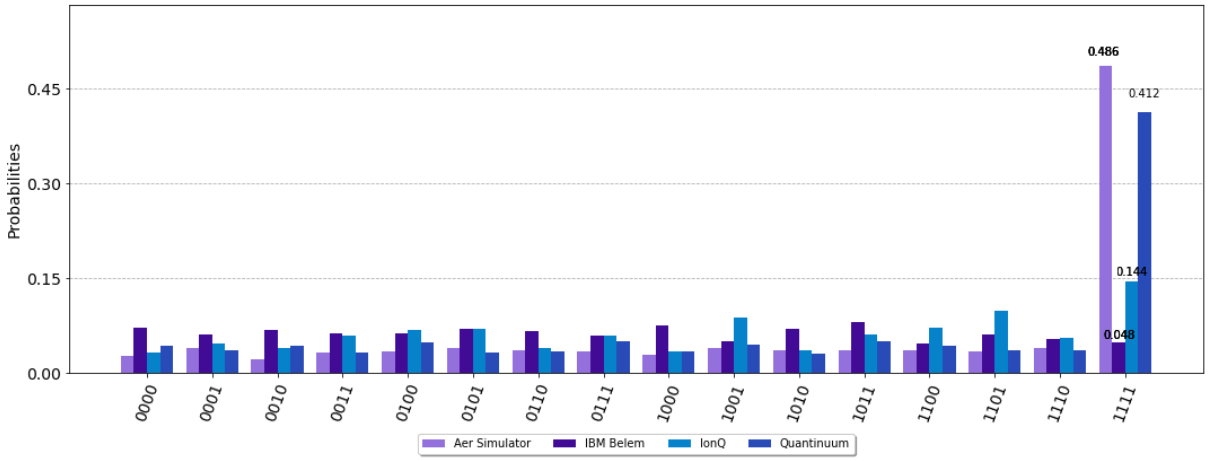

Figure 40. Histogram with results for 4 qubits and 1 iteration.

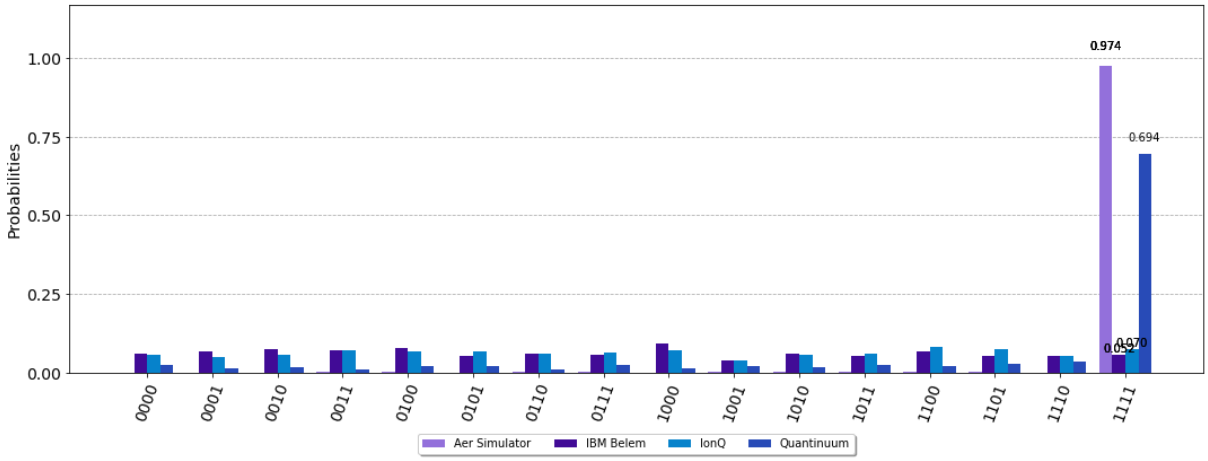

Figure 41. Histogram with results for 4 qubits and number of optimal iterations.

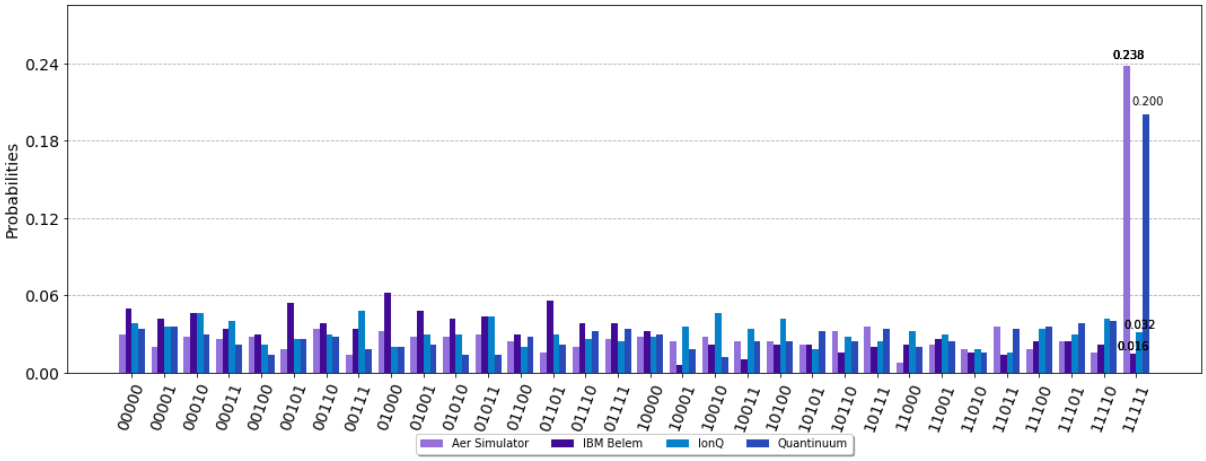

Figure 42. Histogram with results for 5 qubits and 1 iteration.

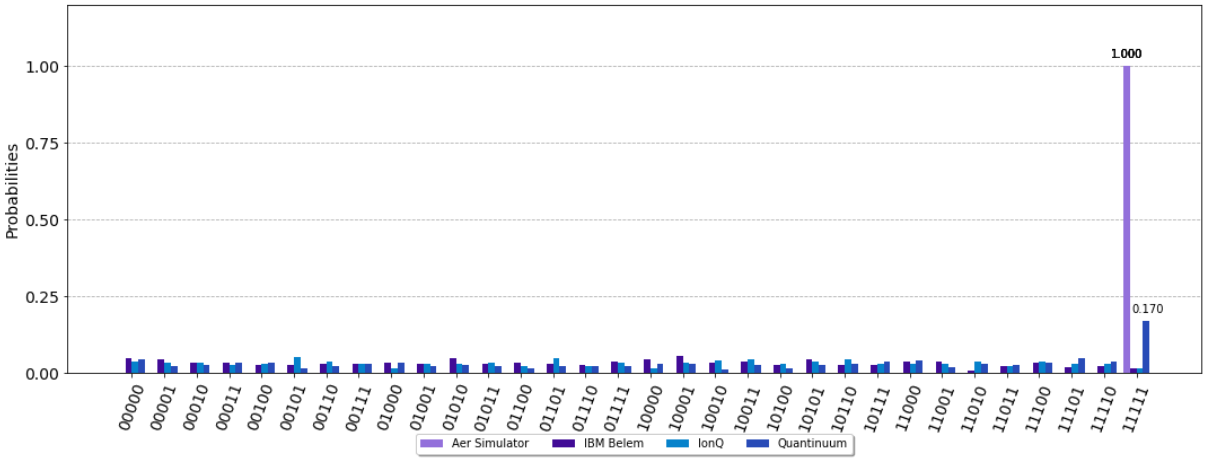

Figure 43. Histogram with results for 5 qubits and number of optimal iterations.

4.2 RESULTS OBTAINED WITH 1000 SHOTS

For the implementation with 1000 repetitions of the circuit, the processors from IBM, IonQ, Rigetti and OQC were used.

Amazon does not provide access to execution time of tasks sent to the quantum hardware available. For this reason, this metric could not be analyzed in this scenario.

Furthermore, Rigetti's processor was unsuccessful in running the circuit for 5 qubits with the optimal number of iterations – a “circuit too long” message returned in this scenario. The same occurred for OQC's Lucy processor, for 4 and 5 qubits, in both configurations (one iteration and ideal number of iterations). For this reason, in **Table 4.3** the *Effective ASP* corresponding to these processors is set to zero in these cases.

Table 4.3 Processor performance information for 1000 *shots*.

Nº of qubits	**Nº of iterations**	***Theoretical* ASP (%)**	***Hardware***	**Effective ASP (%)**
2	*1*	100	Aer Simulator	100
			Belem (IBM)	86.6
			Harmony (IonQ)	94.2
			Aspen M-2 (Rigetti)	84.6
			Lucy (OQC)	36.6
3	*1*	78.13	Aer Simulator	78.5
			Belem (IBM)	18.4
			Harmony (IonQ)	10.4
			Aspen M-2 (Rigetti)	11.9
			Lucy (OQC)	15.6
3	*2*	94.53	Aer Simulator	94.7
			Belem (IBM)	13.5
			Harmony (IonQ)	11.9
			Aspen M-2 (Rigetti)	23.1
			Lucy (OQC)	10.6
4	*1*	47,27	Aer Simulator	46.1
			Belem (IBM)	3.6
			Harmony (IonQ)	7.3
			Aspen M-2 (Rigetti)	12.1
			Lucy (OQC)	0.0
4	*3*	96,13	Aer Simulator	96.0
			Belem (IBM)	4.1
			Harmony (IonQ)	7.3

			Aspen M-2 (Rigetti)	16.7
			Lucy (OQC)	0.0
5	*1*	25,83	Aer Simulator	26.4
			Belem (IBM)	1.0
			Harmony (IonQ)	3.7
			Aspen M-2 (Rigetti)	12.7
			Lucy (OQC)	0.0
5	*4*	99,92	Aer Simulator	100
			Belem (IBM)	0.7
			Harmony (IonQ)	3.8
			Aspen M-2 (Rigetti)	0.0
			Lucy (OQC)	0.0

Figure 44 shows the comparative graph of the ASP values obtained in each processor and, in **Table 4.4**, is the total cost of each hardware to execute the circuits.

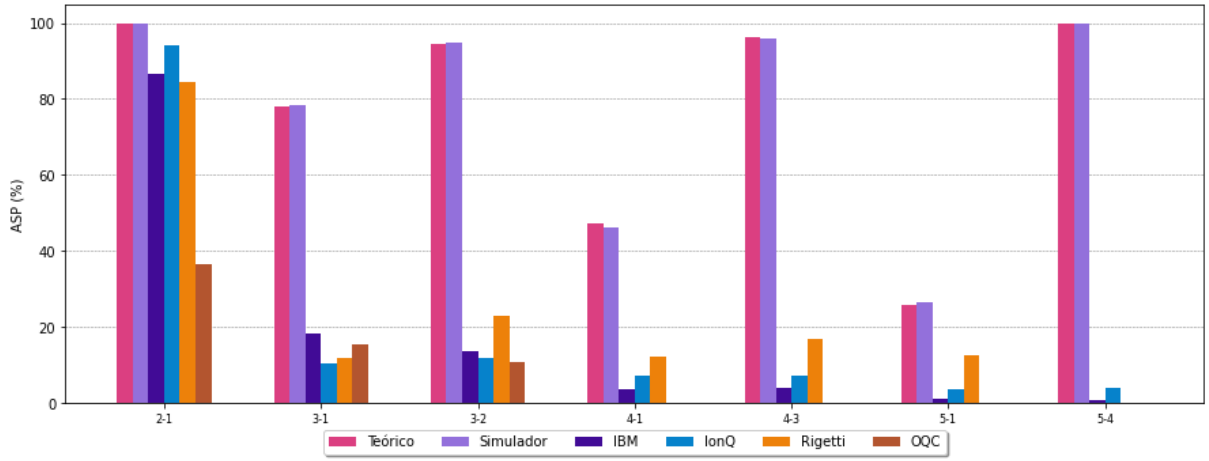

Figure 44. ASP graph obtained on each processor for each of the circuits.

Table 4.4 Total cost for running the circuits.

Hardware	**Cost**
Belem (IBM)	U$ 0.00
Harmony (IonQ)	U$ 197.80
Aspen M-2 (Rigetti)	U$ 3.25
Lucy (OQC)	U$ 4.55

4.2.1 Histograms

Following are the histograms generated for each of the circuits implemented with 1000 shots.

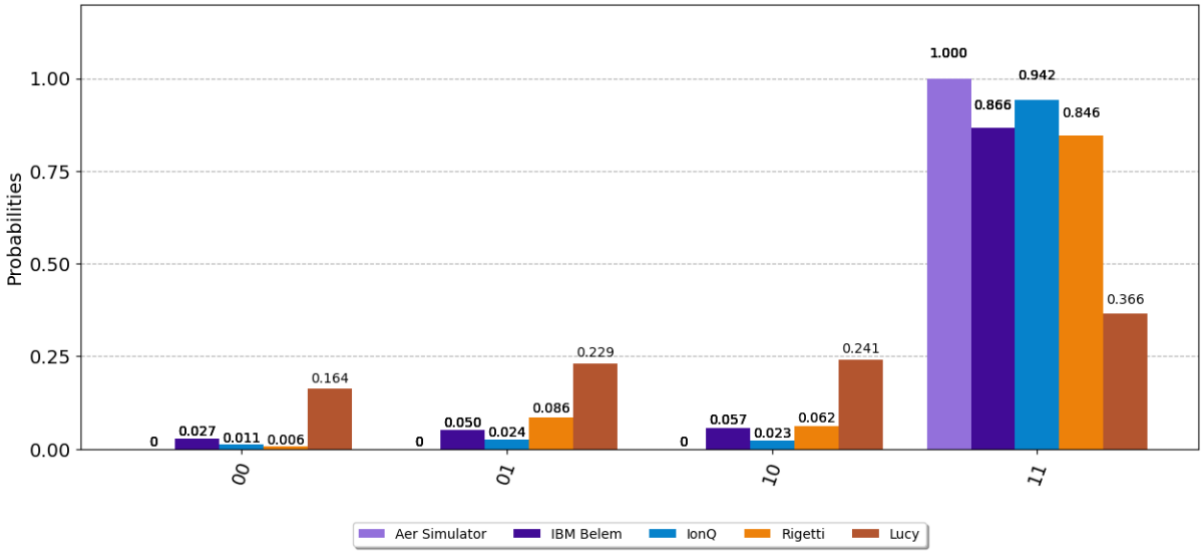

Figure 45. Histogram with results for 2 qubits.

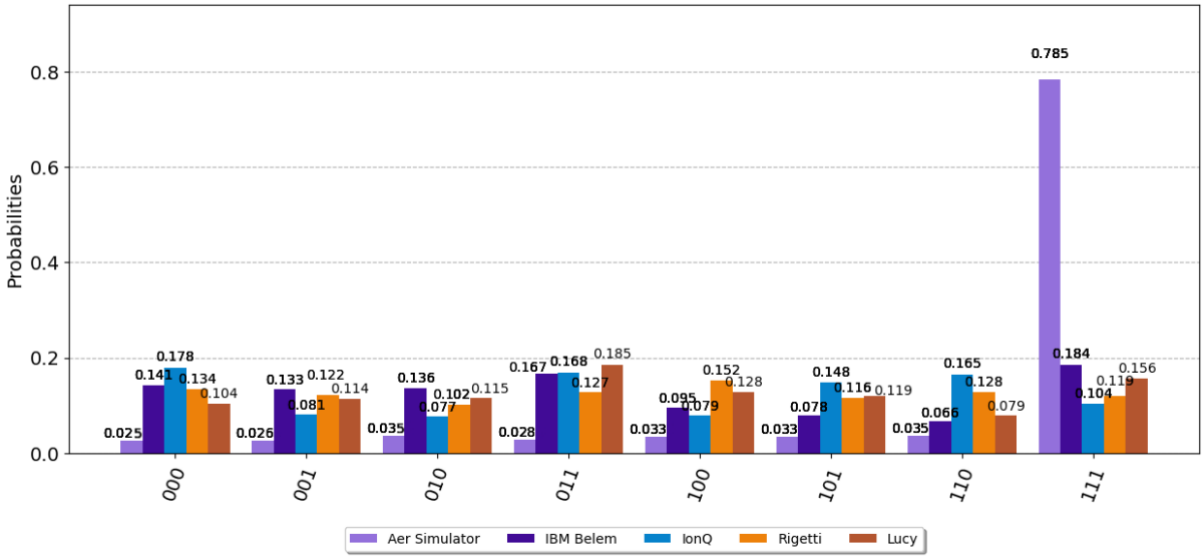

Figure 46. Histogram with results for 3 qubits and 1 iteration.

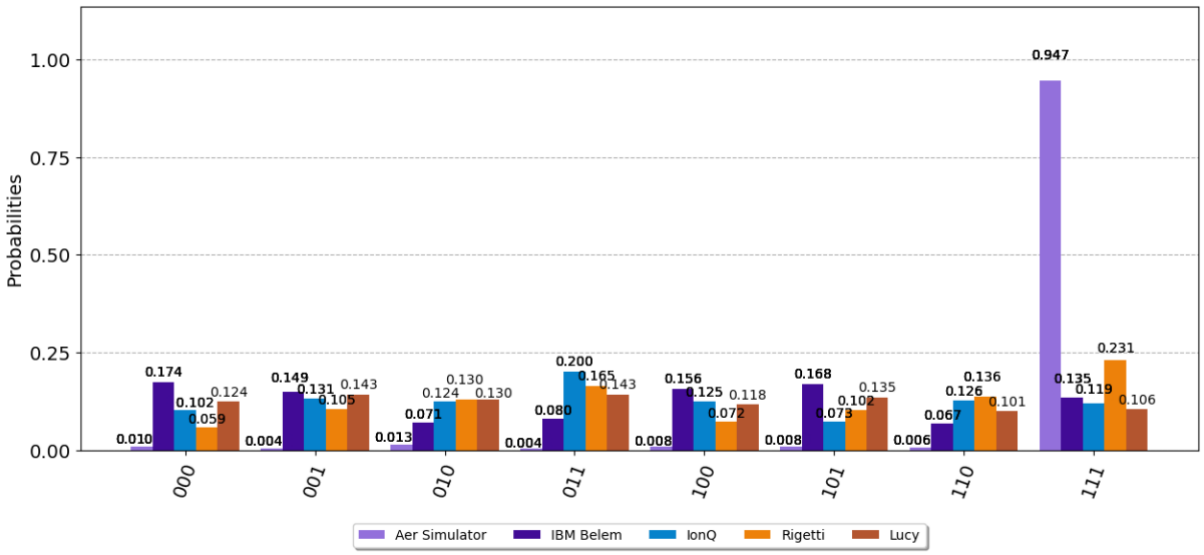

Figure 47. Histogram with results for 3 qubits and number of optimal iterations.

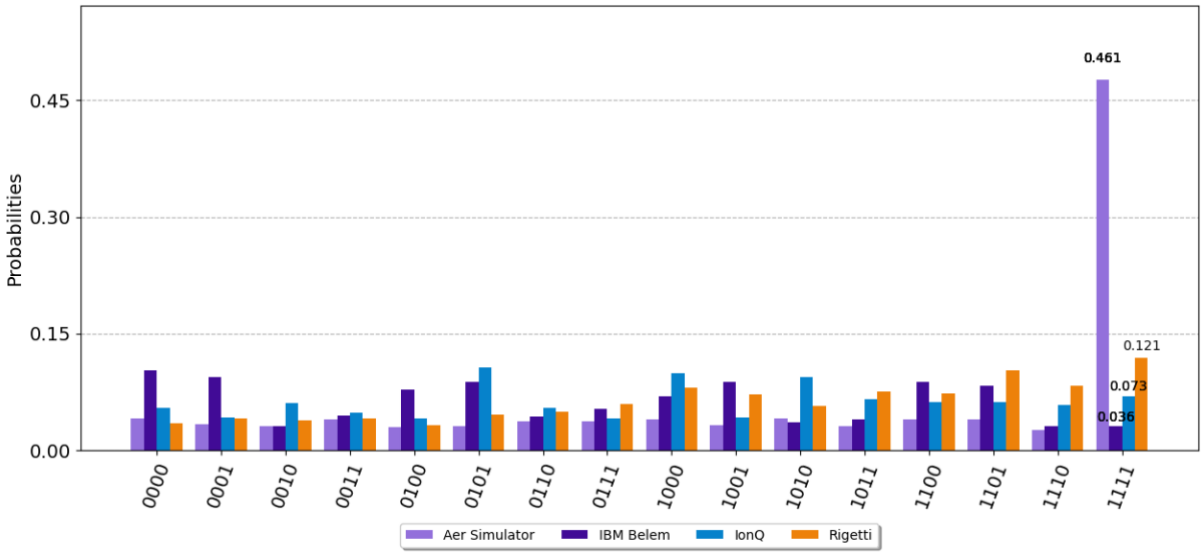

Figure 48. Histogram with results for 4 qubits and 1 iteration.

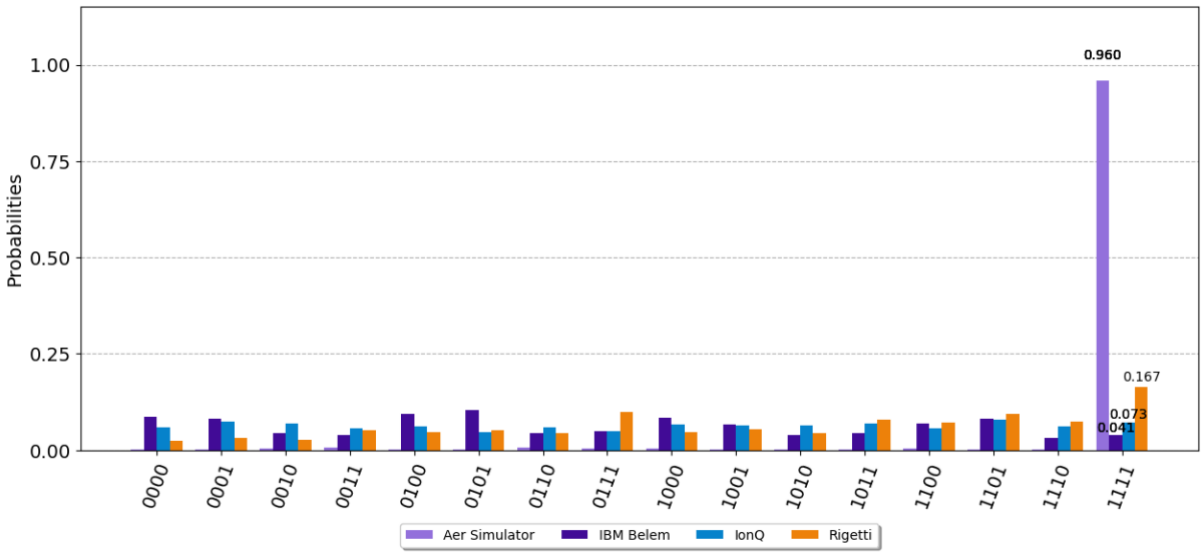

Figure 49. Histogram with results for 4 qubits and number of optimal iterations.

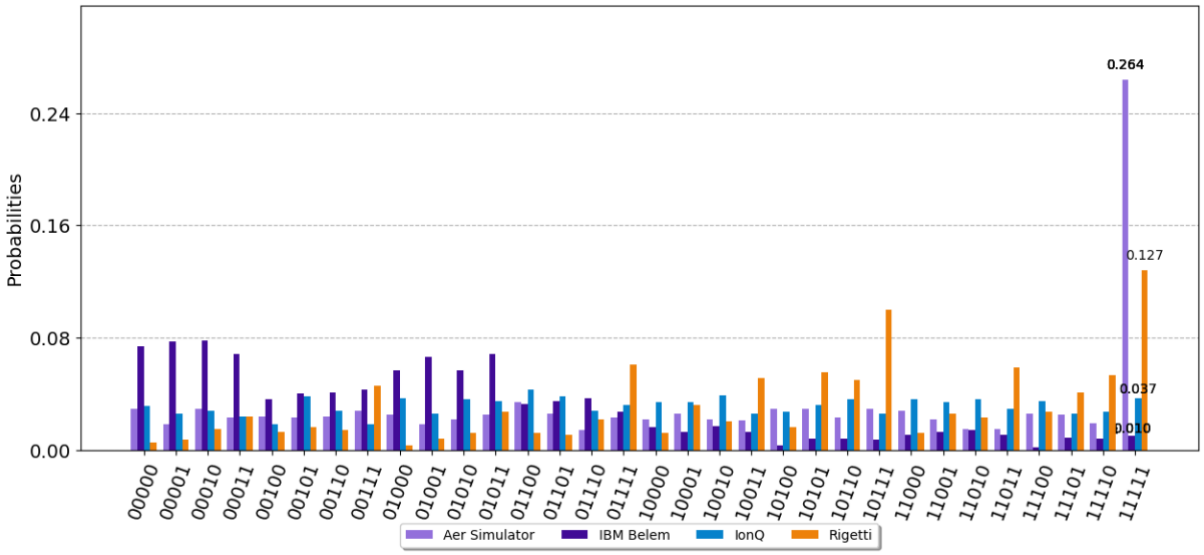

Figure 50. Histogram with results for 5 qubits and 1 iteration.

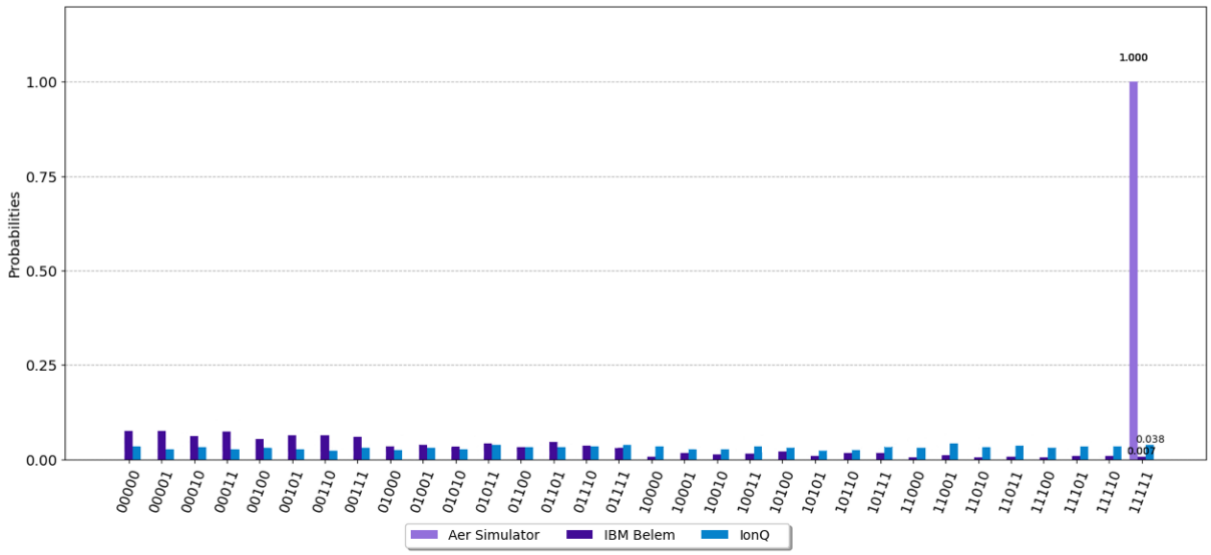

Figure 51. Histogram with results for 5 qubits and number of optimal iterations.

4.3 RESULTS OBTAINED WITH 2000 SHOTS

For the implementation with 2000 repetitions of the circuit, only IBM and IonQ processors were successfully used, whose results are described in **Table 4.5**.

Table 4.5 Processor performance information for 2000 shots.

Nº of qubits	**Nº of iterations**	**Theoretical ASP (%)**	***Hardware***	**Effective ASP (%)**	**Execution time**	**Total Time**
2	1	100	Aer Simulator	100	-	-
			Belem (IBM)	89.50	00:00:02	01:53:58
			Harmony (IonQ)	96.90	00:00:13	00:19:26
3	1	78.13	Aer Simulator	78.00	-	-
			Belem (IBM)	20.80	00:00:03	01:46:47
			Harmony (IonQ)	49.55	00:00:17	00:03:08
3	2	94.53	Aer Simulator	95.10	-	-
			Belem (IBM)	17.90	00:00:03	00:47:14
			Harmony (IonQ)	26.10	00:00:23	00:02:59
4	1	47,27	Aer Simulator	47.9	-	-
			Belem (IBM)	5.70	00:00:04	00:47:31
			Harmony (IonQ)	22.60	00:00:28	00:05:28
4	3	96,13	Aer Simulator	96.50	-	-
			Belem (IBM)	4.70	00:00:05	00:47:47
			Harmony (IonQ)	6.45	00:00:54	00:05:31
5	1	25,83	Aer Simulator	24.90	-	-
			Belem (IBM)	1.10	00:00:05	00:47:24
			Harmony (IonQ)	2.95	00:00:41	00:03:36
5	4	99,92	Aer Simulator	100	-	-
			Belem (IBM)	0.80	00:00:12	00:48:12
			Harmony (IonQ)	4.00	00:02:05	00:39:09

Figure 52 shows the comparative graph of ASP values obtained in each processor and **Figure 53** shows the time used by each processor to execute each circuit. **Table 4.6** shows the total cost of each hardware to run the circuits.

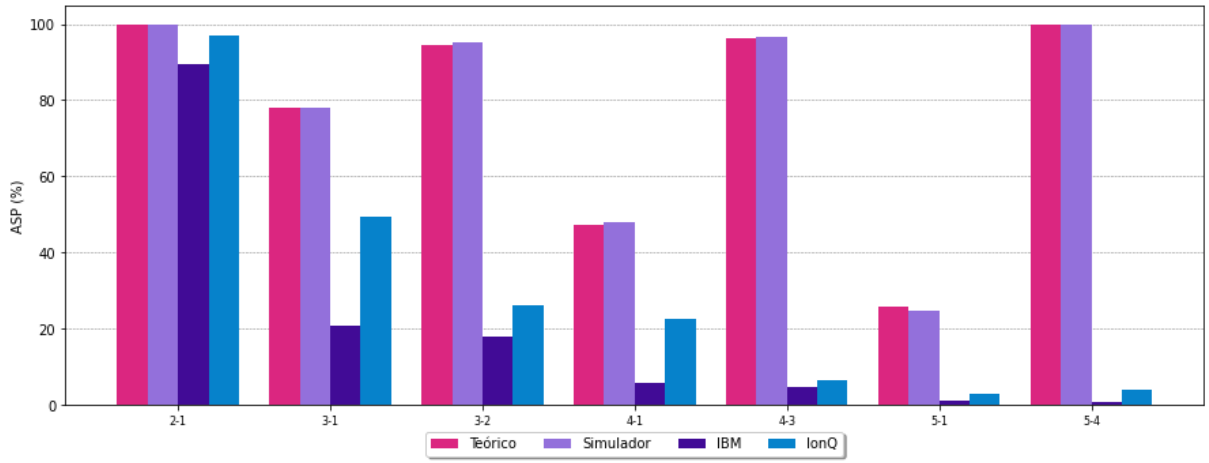

Figure 52. ASP graph obtained in each processor for each of the circuits.

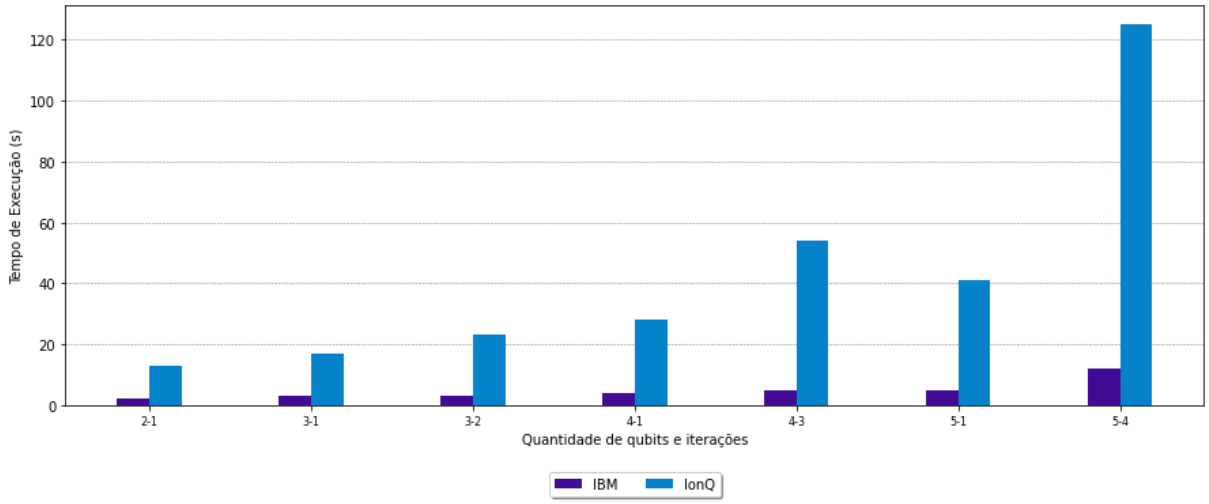

Figure 53. Execution time required by each processor for each of the circuits.

Table 4.6 Total cost for running the circuits.

Hardware	Cost
Belem (IBM)	U$ 0.00
Harmony (IonQ)	U$ 332.58

4.3.1 Histograms

Following are the histograms generated for each of the circuits implemented with 2000 shots.

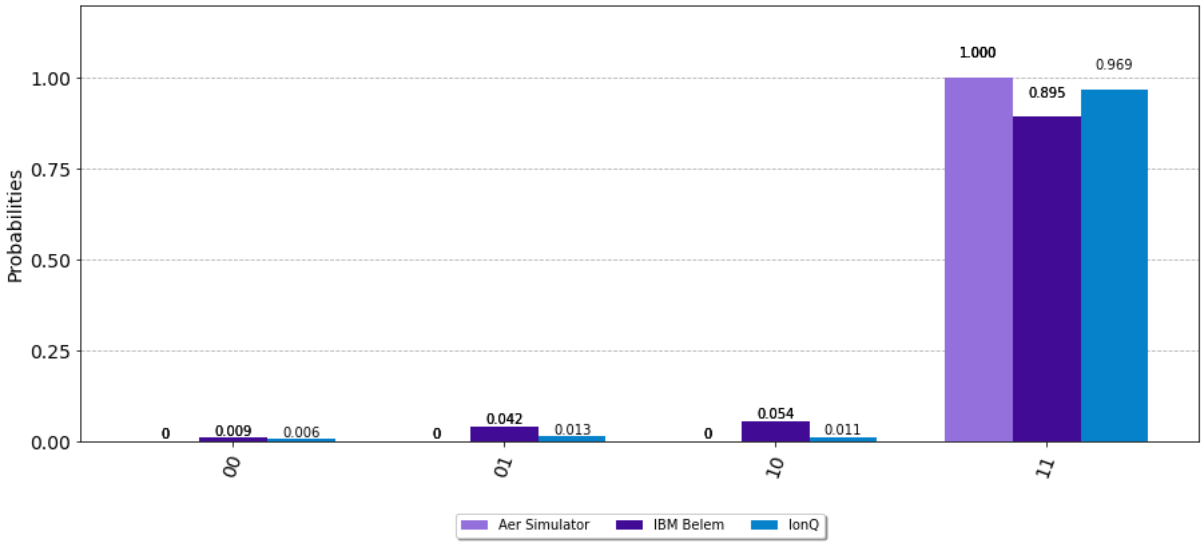

Figure 54. Histogram with results for 2 qubits.

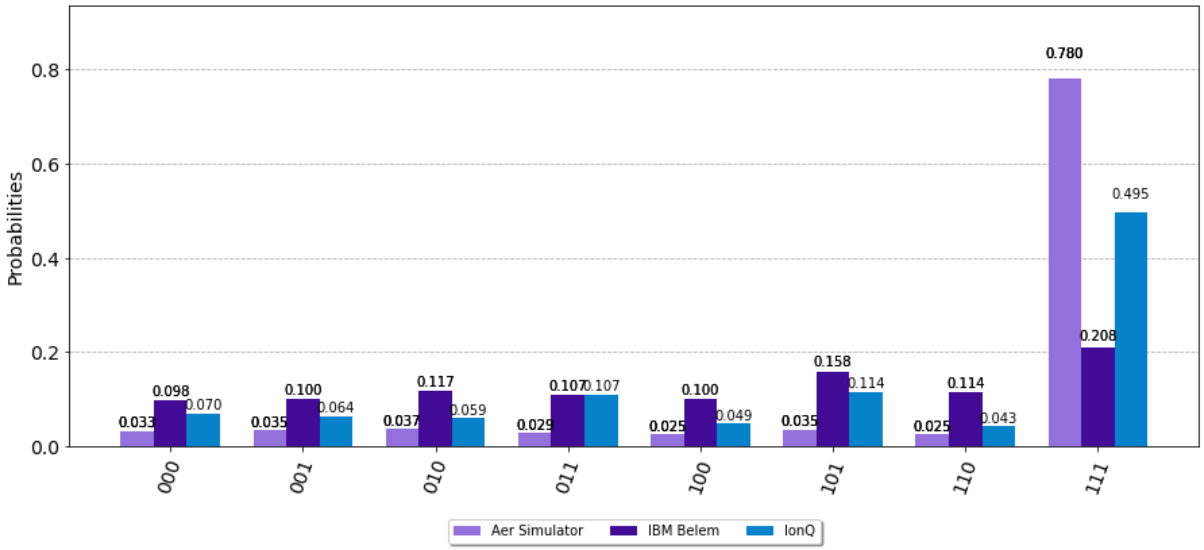

Figure 55. Histogram with results for 3 qubits and 1 iteration.

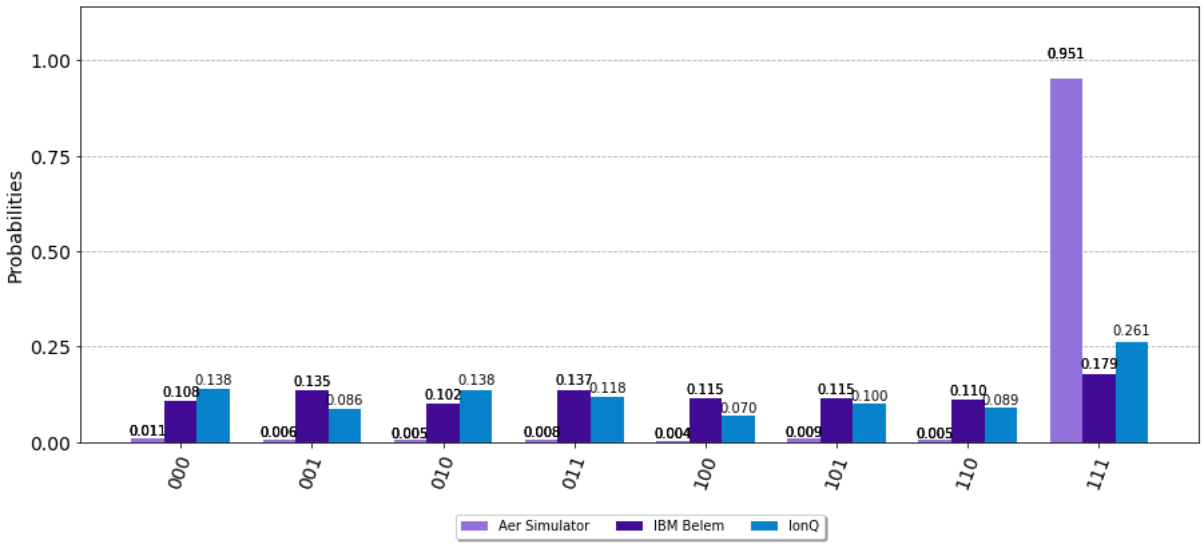

Figure 56. Histogram with results for 3 qubits and number of optimal iterations

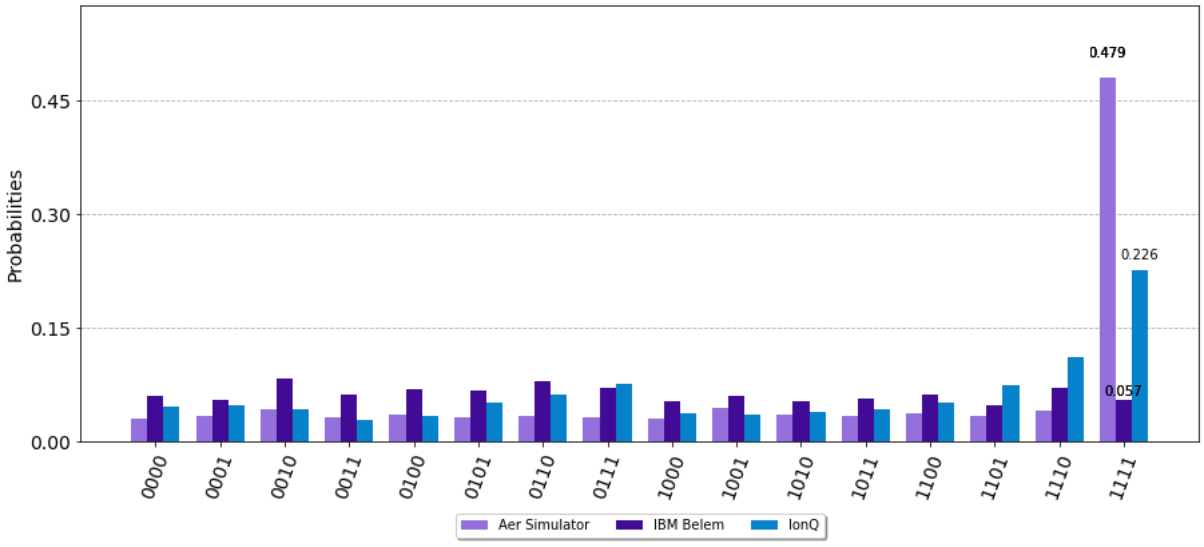

Figure 57. Histogram with results for 4 qubits and 1 iteration.

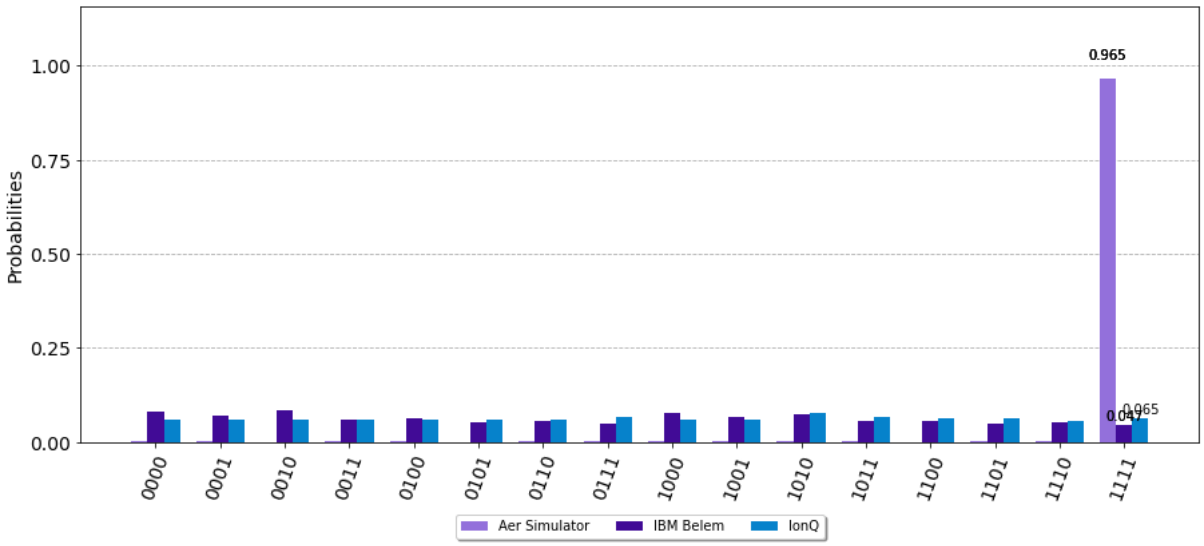

Figure 58. Histogram with results for 4 qubits and number of optimal iterations.

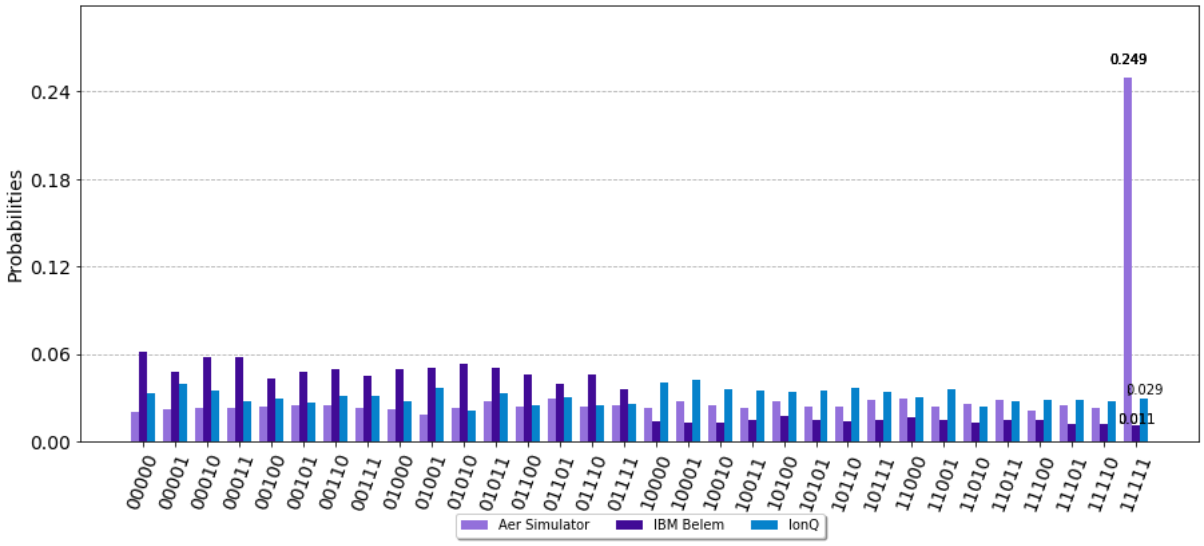

Figure 59. Histogram with results for 5 qubits and 1 iteration.

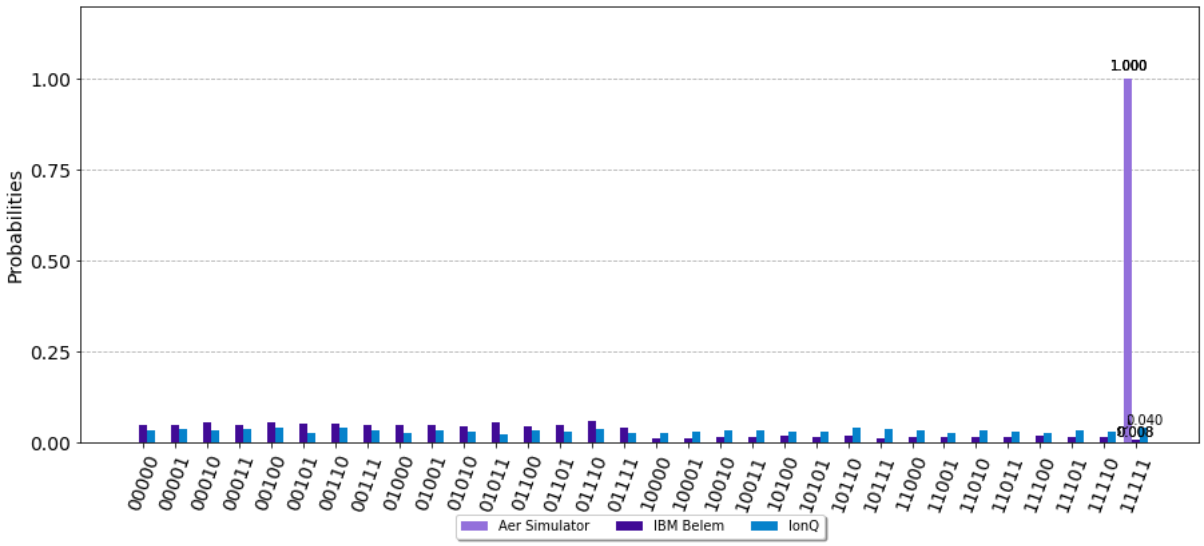

Figure 60. Histogram with results for 5 qubits and number of optimal iterations.

5 CONCLUSION

The results are similar to those found in (ADEDOYIN et al., 2022; BLOMKVIST KARLSSON; STRÖMBERG, 2018; FIGGATT et al., 2017; HU, 2018; JOSHI; GUPTA, 2021; MANDVIWALLA; OHSHIRO; JI, 2018). However, in this project more hardware was tested.

Amazon's platform has an integration error with Qiskit, which makes the circuit always run 1000 times, regardless of the number of shots stipulated by the algorithm. Thus, the hardware Lucy, from OQC, and Aspen M-2, from Rigetti, could only be tested in this condition.

OQC's Coaxmon technology, *a priori*, did not bring great performance benefits to the hardware, on the contrary: even having a number of gates and depth similar to Rigetti's processor – which indicates an equivalent construction topology – its ASP values were significantly lowest, and the worst obtained among all the hardware tested in this study.

Rigetti's Aspen M-2 stands out for being one of the most affordable processors, as shown in **Table 4.4**, and with a good performance obtained for the ASP values. In addition, this hardware, along with IonQ's, are the only ones available on more than one platform. Unfortunately, due to the limited integration of Amazon's platform with Qiskit, it was not possible to obtain its processing times. On Microsoft's Azure platform, it was inactive at the time of this study.

The Harmony processor, from IonQ, has the best cost-benefit ratio according to the results obtained in this study. In addition, the company stands out for its coverage of all the most popular cloud providers, libraries and tools (IONQ, 2022).

As is evident from the graph in **Figure 35**, the hardware with the best ASP performance was Quantinuum. However, this is also the processor with the highest cost, as shown in **Table 4.2**, which is why it was possible to run only 500 shots on this processor. In addition, it takes the longest to run the circuits, reaching more than 30 minutes to run the 5-qubit circuit, not counting the queuing time.

Although IBM hardware hasn't performed the best, the company stands out for the affordability of its processors. It's the only one where one can run 20,000 shots for free. To give an idea, the price of IonQ to run a 5-qubit circuit with the ideal number of iterations (4 iterations) would be $1,733.40. At Quantinuum, this value would be U$11,613.00, for example. Whereas, on the IBM processor, this task would be free. Another strong point is the speed of the processor, which took just a few seconds to execute each task, being the fastest processor among

those tested, even having to apply many more operations than the others due to the high number of gates it needed for the implementation of the algorithm.

In addition, the IBM platform has the richest user experience. The system is fully integrated – with histograms drawn directly on the platform, for example – and without having to deal with several tokens. Just choose the provider and run the algorithm. The rest have a slightly more difficult use and often fail to perform the task – which is very frustrating because the queued task time is usually high on these platforms (with an average around 20h). In IBM's this does not usually occur, and the average queue time is significantly shorter (around 3h).

Although an attempt has been made to include Google hardware in this study, the company currently does not make it available to the public, releasing access only to its simulators. One can use the company's SDK, Cirq, to run circuits on third-party hardware. However, this study was limited to using only Qiskit in order to avoid algorithm performance problems.

Finally, it is worth mentioning that, at least for the general public, it is still more advantageous to perform searches in databases using classical algorithms. It is also evident from the graphs that, due to the low number of qubits currently available and the noise level present in the hardware, it is faster and more accurate to use quantum simulators than effectively quantum processors.

Still, it must be remembered that in less than 50 years classical computing has made a huge leap, from processors with a few thousand *transistors*[16] in 1970 to tens of billions of them today (OWD, 2017). The quantum computers available today are just the first to be built, and if the trend continues, processors with a much larger number of qubits and less noise will soon be available, making it possible to use algorithms like Grover's for much faster searches in databases than current classical algorithms.

[16] *Transistors* are the physical devices that implement the classical bits, 0 and 1.

5.1 FUTURE DIRECTIONS

A processor that has drawn attention for its precursor technology is Xanadu's Borealis, which has qubits based on quantum photonics and uses quantum light sources that emit compressed light pulses for compatibility with continuous variable quantum computing, a paradigm that uses continuous quantum states, known as *qumodes*. The device implements a specific protocol, known as *Gaussian Boson Sampling* (GBS) (XANADU, 2022). It is available through the Amazon platform and requires the *Amazon Braket Python* SDK to be implemented, for this reason it was not explored in this study. However, it is a good option for future studies.

As Rigetti's hardware is also available on Azure, it is interesting to use it also on this platform in order to know its processing time – information not available from Amazon – and compare costs.

During the execution of this project, a sponsorship of U$1,000.00 was approved by Microsoft to be used in Quantinuum's processor tasks, through the Azure platform, for research development. With this, the idea is to expand some metric measured in this project and carry out a new study, to be published in the scientific community soon.

6 SCHEDULE

The elaboration and execution of the project followed the schedule presented in **Table 6.1**.

Table 6.1 Project Schedule

Period	Step	Status
01/25/2022 – 02/15/2022	Study about development of engineering course conclusion works	Concluded
02/16/2022 – 04/25/2022	Literature review and development of theoretical foundation	Concluded
04/26/2022 – 05/17/2022	Study about global and national panorama of quantum computing	Concluded
05/18/2022 – 06/12/2022	Academic recess	-
06/13/2022 – 07/03/2022	Development of remaining project topics and study of implementation platforms	Concluded
07/04/2022 – 07/15/2022	Final adjustments and presentation of the TDEF I to the committee	Concluded
07/16/2022 – 08/10/2022	Literature review on techniques to measure performance of quantum hardware	Concluded
08/16/2022 – 08/25/2022	Literature review on quantum base encoding techniques	Concluded
08/26/2022 – 09/20/2022	Algorithm implementation and results analysis	Concluded
09/21/2022 – 10/01/2022	Final adjustments and presentation of the TDEF II to the committee	Concluded

REFERENCES

ADEDOYIN, A. et al. Quantum Algorithm Implementations for Beginners. 2022.

AMAZON. **Amazon Braket Python SDK**. Disponível em: <https://amazon-braket-sdk-python.readthedocs.io/en/latest/>. Acesso em: 15 set. 2022a.

AMAZON. **Introducing the Qiskit provider for Amazon Braket**. Disponível em: <https://aws.amazon.com/pt/blogs/quantum-computing/introducing-the-qiskit-provider-for-amazon-braket/>. Acesso em: 15 set. 2022b.

AMAZON BRAKET PRICING. **Quantum Computer and Simulator**. Disponível em: <https://aws.amazon.com/pt/braket/pricing/>. Acesso em: 15 set. 2022.

AMAZON WEB SERVICES. **OQC**. Disponível em: <https://aws.amazon.com/pt/braket/quantum-computers/oqc/?nc1=h_ls>. Acesso em: 16 set. 2022.

AZURE QUANTUM. **Provedor de computação quântica do IonQ para o Azure Quantum**. Disponível em: <https://docs.microsoft.com/pt-br/azure/quantum/provider-ionq#ionq-harmony-quantum-computer>. Acesso em: 13 set. 2022a.

AZURE QUANTUM. **Provedor de Quantinuum**. Disponível em: <https://docs.microsoft.com/pt-br/azure/quantum/provider-quantinuum>. Acesso em: 14 set. 2022b.

BLOMKVIST KARLSSON, V.; STRÖMBERG, P. **4-qubit Grover's algorithm implemented for the ibmqx5 architectureDEGREE PROJECT COMPUTER SCIENCE**. [s.l: s.n.].

BOYER, M. et al. Tight bounds on quantum searching. 1996.

CDOTRENDS. **Google Partners With Education Institutions to Upskill Australians**. Disponível em: <https://www.cdotrends.com/story/16410/google-partners-education-institutions-upskill-australians>. Acesso em: 4 maio. 2022.

DA CUNHA, C. R. **History & Binary Representation**. [s.l: s.n.].

DEUTSCH, D.; JOZSA, R. Rapid solution of problems by quantum computation. **Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences**, v. 439, n. 1907, p. 553–558, 8 dez. 1992.

DIRAC, P. A. M. A new notation for quantum mechanics. **Mathematical Proceedings of the Cambridge Philosophical Society**, v. 35, n. 3, p. 416–418, 24 jul. 1939.

DIVINCENZO, D. P. The Physical Implementation of Quantum Computation. 2000.

FIGGATT, C. et al. **Complete 3-Qubit Grover Search on a Programmable Quantum Computer**. [s.l: s.n.].

GEORGOPOULOS, K.; EMARY, C.; ZULIANI, P. Quantum Computer Benchmarking via Quantum Algorithms. 17 dez. 2021.

GRIFFITHS, D. J.; SCHROETER, D. F. **Introduction to Quantum Mechanics**. [s.l: s.n.].

GROVER, L. K. A fast quantum mechanical algorithm for database search. 1996.

GUNZI, A. **How to calculate the depth of a quantum circuit in Qiskit?** Disponível em: <https://medium.com/arnaldo-gunzi-quantum/how-to-calculate-the-depth-of-a-quantum-circuit-in-qiskit-868505abc104>. Acesso em: 9 set. 2022.

HU, W. Empirical Analysis of Decision Making of an AI Agent on IBM's 5Q Quantum Computer. **Natural Science**, v. 10, n. 01, p. 45–58, 2018.

IBM RESEARCH. **Quantum computation center opens**. Disponível em: <https://www.ibm.com/blogs/research/2019/09/quantum-computation-center/>. Acesso em: 29 ago. 2022.

IBM RESEARCH BLOG. **IBM Quantum breaks the 100-qubit processor barrier**. Disponível em: <https://research.ibm.com/blog/127-qubit-quantum-processor-eagle#pageStart>. Acesso em: 23 abr. 2022.

IBM RESEARCH BLOG. **IBM Quantum Spring Challenge**. Disponível em: <https://research.ibm.com/blog/quantum-spring-challenge-2022>. Acesso em: 4 maio. 2022.

IEEE QUANTUM. **Summary of the IEEE Workshop on Benchmarking Quantum Computational Devices and Systems**. Disponível em: <https://quantum.ieee.org/education/quantum-supremacy-and-quantum-computer-performance>. Acesso em: 1 jul. 2022.

IONQ. **Trapped Ion Quantum Computing**. Disponível em: <https://ionq.com/>. Acesso em: 29 set. 2022.

JAEGER, L. **The Second Quantum Revolution**. [s.l.] Springer, 2018. v. 1

JOSHI, S.; GUPTA, D. Grover's Algorithm in a 4-Qubit Search Space. **Journal of Quantum Computing**, v. 3, n. 4, p. 137–150, 2021.

LANGIONE, M. et al. **The Race to Quantum Advantage Depends on Benchmarking**. [s.l: s.n.]. Disponível em: <https://www.bcg.com/publications/2022/value-of-quantum-computing-benchmarks>. Acesso em: 1 jul. 2022.

LINKE, N. M. et al. Experimental comparison of two quantum computing architectures. **Proceedings of the National Academy of Sciences of the United States of America**, v. 114, n. 13, p. 3305–3310, 28 mar. 2017.

MANDVIWALLA, A.; OHSHIRO, K.; JI, B. **Implementing Grover's Algorithm on the IBM Quantum Computers**. [s.l: s.n.].

MICROSOFT. **Implementar o algoritmo de pesquisa do Grover no Q#**. Disponível em: <https://docs.microsoft.com/pt-br/azure/quantum/tutorial-qdk-grovers-search?tabs=tabid-visualstudio>. Acesso em: 22 jun. 2022.

MILLER, D. A. B. **Quantum Mechanics for Scientists and Engineers**. [s.l.] Cambridge, 2008.

MILLS, A. R. et al. Two-qubit silicon quantum processor with operation fidelity exceeding 99%. 23 nov. 2021.

OWD. **Moore's Law: The number of transistors per microprocessor**. Disponível em: <https://ourworldindata.org/grapher/transistors-per-microprocessor>. Acesso em: 16 out. 2022.

NIELSEN, M. A.; CHUANG, I. L. **Quantum computation and quantum information**. [s.l.] Cambridge University Press, 2010.

OXFORD QUANTUM CIRCUITS. **Technology**. Disponível em: <https://oxfordquantumcircuits.com/technology>. Acesso em: 16 set. 2022.

PATINFORMATICS LLC. Practical Quantum Computing: A Patent Landscape Report. 2017.

PIRES, F. **Quantum Computing Benchmarks Begin to Take Shape**. Disponível em: <https://www.tomshardware.com/news/quantum-computing-benchmarks-begin-to-take-shape>. Acesso em: 1 jul. 2022.

PRADO, S.; DILLENBURG, R. O Algoritmo de Grover. 2014.

PRESKILL, J. Quantum computing in the NISQ era and beyond. **Quantum**, v. 2, 6 ago. 2018.

QISKIT. **Introducing Qiskit Aer: A high performance simulator framework for quantum circuits**. Disponível em: <https://medium.com/qiskit/qiskit-aer-d09d0fac7759>. Acesso em: 24 set. 2022.

QISKIT. **Algoritmo de Grover e Amplificação De Amplitude**. Disponível em: <https://qiskit.org/documentation/locale/pt_BR/tutorials/algorithms/06_grover.html>. Acesso em: 22 jun. 2022a.

QISKIT. **Grover's Algorithm**. Disponível em: <https://qiskit.org/textbook/ch-algorithms/grover.html>. Acesso em: 22 jun. 2022b.

QISKIT 0.38.0. **Transpiler (qiskit.transpiler)**. Disponível em: <https://qiskit.org/documentation/apidoc/transpiler.html>. Acesso em: 16 set. 2022.

QUANTINUUM. **Products | H1**. Disponível em: <https://www.quantinuum.com/products/h1>. Acesso em: 15 set. 2022.

Quantum Manifesto A New Era of Technology. 2016.

QURECA. **Overview on quantum initiatives worldwide**. Disponível em: <https://qureca.com/overview-on-quantum-initiatives-worldwide-update-2022/>. Acesso em: 30 jun. 2022.

RIGETTI. **Rigetti QCS**. Disponível em: <https://qcs.rigetti.com/qpus>. Acesso em: 15 set. 2022.

R.P. FEYNMAN. **Simulating physics with computers**. [s.l.] Int. J. Theor. Phys., 1982.

SCHULD, M.; PETRUCCIONE, F. Supervised Learning with Quantum Computers. Em: [s.l: s.n.]. p. 108–114.

SCHULD, M.; PETRUCCIONE, F. **Supervised Learning with Quantum Computers**. [s.l: s.n.].

SCHULD, M.; PETRUCCIONE, F. Search and Amplitude Amplification. Em: **Machine Learning with Quantum Computers**. Second Edition ed. [s.l: s.n.]. p. 256–269.

SHOR, P. W. **Algorithms for quantum computation: discrete logarithms and factoring**. Proceedings 35th Annual Symposium on Foundations of Computer Science. **Anais**...1994.

WHALEN, J. **Chinese scientists are at the forefront of the quantum revolution**. Disponível em: <https://www.washingtonpost.com/business/2019/08/18/quantum-revolution-is-coming-chinese-scientists-are-forefront/>. Acesso em: 4 maio. 2022.

WRIGHT, K. et al. Benchmarking an 11-qubit quantum computer. **Nature Communications**, v. 10, n. 1, 1 dez. 2019.

XANADU. **Borealis**. Disponível em: <https://www.xanadu.ai/products/borealis/>. Acesso em: 29 set. 2022.

YAN, S. **China launches satellite aimed at "hack-proof" communications**. Disponível em: <https://money.cnn.com/2016/08/16/technology/china-quantum-satellite/index.html>. Acesso em: 30 abr. 2022.

YE, A. **Grover's Algorithm — Quantum Computing**. Disponível em: <https://medium.com/swlh/grovers-algorithm-quantum-computing-1171e826bcfb>. Acesso em: 3 ago. 2022.

ZHANG, H. et al. Realization of quantum secure direct communication over 100 km fiber with time-bin and phase quantum states. **Light: Science & Applications**, v. 11, n. 1, p. 83, 6 dez. 2022.

ZWERVER, A. M. J. et al. Qubits made by advanced semiconductor manufacturing. **Nature Electronics**, v. 5, n. 3, p. 184–190, mar. 2022.