Revista Brasileira de Ensino de Física, vol. 44, e20210290 (2022)
www.scielo.br/rbef
DOI: https://doi.org/10.1590/1806-9126-RBEF-2021-0290

Artigos Gerais

Algoritmos Quânticos usando o Qiskit: Uma abordagem para o ensino de informação e computação quântica

Quantum Algorithmis using the Qiskit: An approach to teach quantum computation and information

Warley Marcos Santos Alves[1,2], Jean Carlos Coelho Felipe*[2]

[1]Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Física, Belo Horizonte, MG, Brasil.
[2]Universidade Federal dos Vales do Jequitinhonha e Mucuri, Instituto de Engenharia, Ciência e Tecnologia, Janaúba, MG, Brasil.

Recebido em 11 de agosto de 2021. Revisado em 03 de março de 2022. Aceito em 03 de março de 2022.

O presente estudo objetivou analisar os dados de simulações de dois algoritmos quânticos diferentes: Algoritmo de *Deutsch* e a Transformada de Fourier Quântica. Para tal abordagem foi utilizado o *framework Qiskit* na realização das simulações dos algoritmos anteriormente apresentados. Tais simulações se deram na máquina local (computador clássico) e nos chips quânticos da *International Business Machines* (IBM) via nuvem. Na máquina local, os resultados obtidos são os previstos pela teoria. Nos protótipos da IBM, no entanto, pequenas flutuações de erro puderam ser observadas. Todavia, os estados quânticos esperados foram obtidos com ótima probabilidade. Nossos resultados mostram que os computadores quânticos utilizados são robustos para os algoritmos estudados, sendo essa uma maneira bastante intuitiva para o ensino dos conceito de informação e computação quântica para o estudantes do ciclo profissional (técnico) da graduação.
Palavras-chave: Mecânica Quântica, Computação Quântica, *IBM Quantum Experience*, Algoritmo de *Deutsch*, Transformada de Fourier Quântica.

The present study aimed to analyze the simulation data of two different quantum algorithms: Deutsch Algorithm and the Quantum Fourier Transform. In order to execute this, has been used the Qiskit framework to implement the afore mentioned algotihms. Such simulations took place on local machine (classic computer) and on the quantum chips of International Business Machines (IBM) via cloud. On the local machine the results obtained are those predicted by the theory. In IBM prototypes, small error fluctuations have been observed. However, the expected states were obtained with high probability. Our results show that the quantum computers are robust to the studied algorithms, been this an intuitive way to teach the concepts of quantum information and computation to students in the professional (techinical) level of graduation.
Keywords: Quantum information, Quantum mechanics, Quantum computing, IBM Quantum Experience, Deutsch algorithm, Quantum Fourier Transform.

1. Introdução

Estudos em computação quântica tem se tornado recorrentes na atualidade, uma vez que a mesma oferece vantagens quanto a execução de algorítimos que não teriam como ser abordados em computadores clássicos [2]. Nesse sentido, a construção de protótipos de computadores quânticos tornaram-se uma realidade nos últimos anos, onde a necessidade atual é implementar o aumento do número qubits para um maior poder de processamento (do ponto de vista de rapidez) de algorítmos pré-estabelecidos [2].

Em 1982, o físico *Richard Feynman* postulou [3] que sistemas clássicos não seriam capazes de modelar de forma eficiente sistemas quânticos, e que estes só poderiam ser modelados atavés de outro sistema quântico [4]. Além disso, *Feynman* sugeriu que computadores, que operassem, tendo como base as leis da Mecânica Quântica, poderiam ser usados para simular sistemas quânticos (ao invés de computadores clássicos) [5]. Com base nisso, *Deutsch* foi o primeiro a pensar em um modelo de computador quântico universal e, em 1985, publicou um artigo [6] mostrando as vantagens que um computador quântico teria sobre um computador clássico. O importante passo dado por *Deutsch* foi desenvolvido nas décadas seguintes por muitos pesquisadores, culminando com o trabalho de *Peter Shor* [7], o qual mostrou que dois problemas muito importantes – o da fatoração em números primos [8] e o problema do logaritmo discreto [9] – seriam eficientemente resolvidos por um computador quântico. Isso atraiu o interesse da comunidade da área, pois já havia a ideia de que um computador clássico não poderia resolver esses problemas de forma eficiente.

* Endereço de correspondência: jean.cfelipe@ufvjm.edu.br

Desde então surgiram várias propostas, e algumas delas com resultados promissores [10]. O trabalho [7] impactou os esquemas criptográficos atuais [11], pois muitas das grandes empresas, bancos e órgãos governamentais utilizam um sistema de criptografia baseada na dificuldade de fatorar números primos muito grandes. Desse modo, qualquer pessoa com um computador quântico capaz de implementar o algoritmo de *Shor* a ponto de quebrar a criptografia de tais sistemas, representaria uma ameaça imediata para a segurança digital.

Com a revolução dos transistores, *Gordon Earl Moore*, observou e fez uma previsão, na qual o número de transistores em um *chip* era dobrado a cada 2 anos [12], sendo seu tamanho reduzido na mesma proporção. Mas com a diminuição no tamanho dos transistores, estes componentes chegariam a escala atômica, de maneira que o funcionamento dos mesmos dependeria das regras impostas pela Mecânica Quântica [13, 14].

Assim, o grande desafio na atualidade é a implementação de um *hardware* quântico [15–17]. Os avanços tem se tornado cada vez mais significativos e grandes empresas como *Google*, *IBM* e *Intel* começaram a divulgar seus progressos [18] e o roteiro para construção de *hardwares* quânticos. Assim, iniciou-se uma verdadeira corrida pela "supremacia quântica". Em 2019, a *Google* afirmou ter alcançado a "supremacia quântica" com seu computador quântico chamado de *Sycamore*. A tarefa que foi dada a essa máquina consisita em encontrar um padrão em uma série aleatória de números. O computador quântico da *Google* obteve uma resposta em 200 segundos. A *Google* estima que um supercomputador clássico de última geração levaria aproximadamente 10.000 anos para executar essa mesma tarefa [18]. No ano passado, a empresa construiu uma outra versão reduzida do *Sycamore* para simulações em Química Quântica. Com uma dúzia de *qubits*, a equipe consegui simular o mecanismo de isomerização do diazeno e a energia de ligação das cadeias de hidrogênio (H12) [19]. Apesar dessa simulação também poder ser realizada em um computador clássico, os avanços relacionados à Computação Quântica demonstram um grande passo na busca por novos fármacos e novos materiais.

Segundo *Jay Gambetta*, *Jerry Chow* e Zaira Nazario da *IBM*, a confecção de um computador quântico totalmente capaz e tolerante a falhas exigirá grandes investimentos na resolução de três desafios: circuitos quânticos eficientes com correção de erros, interconexões quânticas e aplicativos para o reconhecimento do *hardware* [20]. Apesar desses desafios, a *IBM* tem projetos para escalonar processadores com aumento gradual no número de *qubits* e, consequentemente, cada vez mais robustos. A meta da empresa é possuir no final de 2023 dispositivos com mais de 1.000 *q-bits* [20]. Em 2016, a *IBM* deu acesso a comunidade em geral a um protótipo de computador quântico de 5 *q-bits* [21]. Nos anos seguintes, outros protótipos foram liberados aos usuários externos. O acesso a esses dispositivos é realizado de forma remota utilizando o serviço de nuvem. Atualmente, a empresa possui computadores quânticos de até 65 *q-bits*, sendo alguns desses protótipos de acesso restrito. Essa inciativa da *IBM* permite a análise de algoritmos quânticos de modo relativamente simples. Desse modo, qualquer pessoa motivada pode contribuir para o desenvolvimento da área.

A *IBM QE* (*IBM Quantum Experience*) permite aos entusiastas acesso prático e rápido através de uma interface bastante intuitiva, permitindo que os mesmos executem projetos voltados à computação quântica [23]. Já o Qiskit (*Quantum Information Software Development Kit*) consiste de uma plataforma mais profissional na área, o qual permite aos usuários realizarem programação em alto nível, funcionando como uma plataforma de desenvolvimento de *softwares* quanto uma linguagem de Computação Quântica [23, 26]. Com um novo paradigma, a Computação Quântica fornece muitas possibilidades e abordagens para o processamento de informação. E esta é uma área que fornece muitos desafios para Físicos, cientista da computação e engenheiros, pois é uma área com muitas questões em aberto. Os frutos de investigações nesta área poderão resultar em dispositivos de processamento de informação muito superiores aos atuais, gerando grandes benefícios para a sociedade.

Diante do exposto, o presente trabalho visa abordar alguns aspectos sobre fundamentos teóricos que estão por trás da computação quântica para, na sequência, de maneira didática, apresentar os resultados da simulação do Algoritmo de *Deutsch* e a Transformada de *Fourier* Quântica utilizando os *chips* da *IBM* via *Qiskit*. Por fim, serão feitas análises dos resultados das simulações, tanto do ponto de vista do computador clássico (simulação via máquina local) quanto dos computadores quânticos (acesso remoto), analisando as diferenças e ganhos em seus resultados bem como a performance dos mesmos. Algumas dessas diferenças decorrem da arquitetura dos processadores, o que pode vir a permitir uma maior ou menor interação com o ambiente externo [1]. Acredita-se que essa plataforma seja de grande relevância para introduzir os conceitos de Mecânica Quântica e Informação Quântica aos estudantes da fase terminal dos de graduação em Ciências Exatas e Engenharias, despertando o interesse pelos avanços tecnológicos dessa área de pesquisa. Cabe salientar que este artigo baseia-se nos resultados obtidos no Trabalho de Conclusão de Curso, recentemente produzido, o qual versa sobre o conteúdo a ser descrito no corpo desse trabalho (para maiores detalhes, *vide* [36]).

2. Referencial Teórico

Nessa seção serão abordados conceitos importantes sobre o ferramental necessário para a execução e implementação de algoritmos quânticos.

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2021-0290

2.1. O *qubit* e suas propriedades

O *bit* clássico, matematicamente é uma variável aleatória que pode assumir dois valores [22], sendo a menor unidade de informação em uma teoria clássica de informação, correspondendo aos estados 0 ou 1. Pensando por exemplo em uma lâmpada, do ponto de vista clássico, os estados 0 e 1 podem representar a situação da lâmpada como "desligado" e "ligado", respectivamente [2]. Assim, em um computador clássico, sua arquitetura permite a manipulação e o armazenamento dos *bytes*, formado por um conjunto de 8 *bits* [23], que são uteis na representação de números, letras e diversos símbolos. No mundo quântico, a construção do conceito de unidade de informação é análogo e denominado de *bit quântico* ou *qubit* ou ainda *q-bit*, abreviadamente.

Assim como o *bit* clássico, o *qubit* também apresenta dois estados correspondentes

$$|0\rangle \longrightarrow 0, \quad |1\rangle \longrightarrow 1. \tag{1}$$

A diferença entre *bits* e *qubits* é que os *qubits* podem estar nos estados $|0\rangle$ ou $|1\rangle$ ao mesmo tempo, permitindo a existência de um conjunto infinito de estados. Ou seja eles podem formar combinações lineares, que implica na sobreposição dos estados $|0\rangle$ e $|1\rangle$, da forma

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2}$$

onde α e β são números complexos, cuja soma dos módulos quadrados deve ser igual a 1 (máximo que pode ser atingido após a soma de todas as probabilidades possíveis). Todavia, o estado de um *qubit* é um vetor em um espaço vetorial complexo de duas dimensões, na qual os estados $|0\rangle$ e $|1\rangle$ formam uma base ortonormal [2]. Quando mensurado, tem-se o estado $|0\rangle$ com probabilidade $|\alpha|^2$ e o estado $|1\rangle$ com probabilidade $|\beta|^2$, onde $|\alpha|^2 + |\beta|^2 = 1$. Assim, a vantagem do *q-bit* frente ao *bit* clássico está no princípio da superposição, o que permite a combinação linear de vetores que compõem a base computacional. Matricialmente, podemos representar os estados $|0\rangle$ e $|1\rangle$ da forma

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

e

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

respectivamente. A equação (2) pode ser reescrita em coordenadas esféricas. Isso é conveniente uma vez que a atuação de transformações unitárias sobre o *qubit* pode ser representada de forma geométrica. Tal representação é a visualização de uma rotação após a aplicação de um operador unitário. O estado de um *qubit* em coordenadas esféricas, de maneira geral, é dado pela seguinte equação [2]:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle, \tag{3}$$

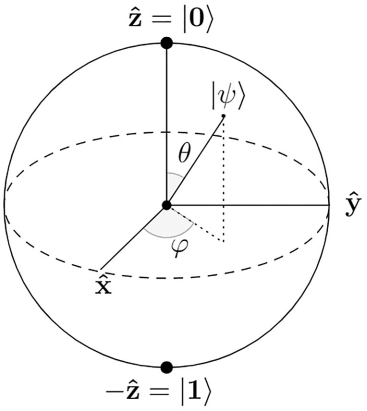

Figura 1: Representação de um *qubit* na esfera de Bloch.

com $\theta \in [0, \pi]$ e $\varphi \in [0, 2\pi]$. As quantidades θ e φ definem um ponto sobre a superfície de uma esfera de raio unitário. Essa esfera é chamada de esfera de *Bloch* e, através dessa esfera que são representados todos os estados de um *qubit*. A representação de um *qubit* na esfera de *Bloch* (descrito anteriormente) pode ser visualizado na Figura 1. A seguir, mostraremos o papel das portas quânticas e como elas atuam sobre os estados quânticos para fins de representação na esfera de *Bloch*.

2.2. Portas Quânticas

A computação quântica (similar ao modelo clássico de computação) pode ser realizada usando o modelo de circuitos tendo como base um sistema quântico. As operações lógicas são efetuadas por portas quânticas que, por sua vez, manipulam o estado dos *qubits*.

Um conjunto muito importante de portas quânticas são representadas pelas matrizes de *spin* de *Pauli*. As matrizes X, Y e Z passam a ser nomeadas de porta X, porta Y e porta Z, também denominadas de portas *Pauli*. A seguir será mostrado como essas portas atuam e alteram o estado de um *qubit*.

A porta X quando aplicada aos elementos da base computacional $|0\rangle, |1\rangle$ realiza a seguinte operação:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \Rightarrow \begin{cases} X|0\rangle = |1\rangle \\ X|1\rangle = |0\rangle \end{cases}. \tag{4}$$

Em razão deste resultado, a porta X também é chamada de porta NOT quântica. A porta NOT quântica, na esfera de *Bloch* gira o *qubit* de um ângulo π em torno do eixo x, como mostrado na Figura 2.

Já para a porta Z quando aplicada aos elementos da base computacional $|0\rangle, |1\rangle$, realiza a seguinte operação:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \Rightarrow \begin{cases} Z|0\rangle = +|0\rangle \\ Z|1\rangle = -|1\rangle \end{cases}. \tag{5}$$

Esta porta deixa a amplitude do *qubit* inalterada, ou seja, os estados $|0\rangle$ e $|1\rangle$ ainda serão os mesmos, mas a fase do *qubit* será deslocada por π radianos, como mostrado na Figura 3.

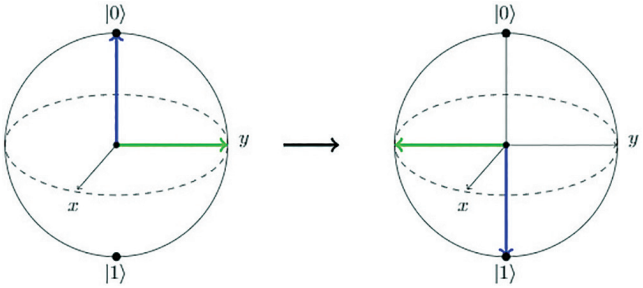

Figura 2: Estado do *qubit* antes e depois da porta Pauli-X. O vetor em azul indica o estado inicial (final) do sistema e o vetor em verde indica a fase inicial (final) associada a esse vetor, como pode ser visto nas representações das esferas de *Bloch* à esquerda (direita).

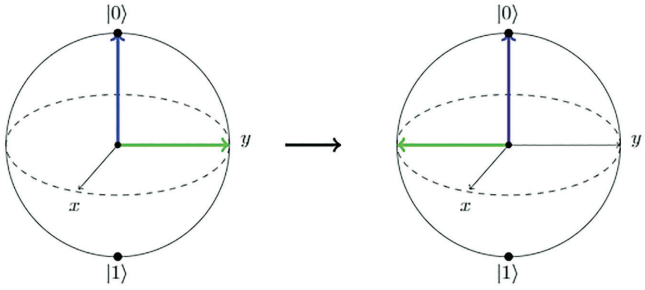

Figura 3: O estado do *qubit* antes e depois da porta Pauli-Z. O vetor em azul indica o estado inicial (final) do sistema e o vetor em verde indica a fase inicial (final) associada a esse vetor, como pode ser visto nas representações das esferas de *Bloch* à esquerda (direita).

Já para a porta Y quando aplicada aos elementos da base computacional $|0\rangle, |1\rangle$, perfaz a seguinte operação:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \Rightarrow \begin{cases} Y|0\rangle = +i|1\rangle \\ Y|1\rangle = -i|0\rangle \end{cases}. \tag{6}$$

Esta porta gira a amplitude do estado de π radianos, ao redor do eixo y, matendo a fase inalterada, como pode ser visto na Figura 4.

Uma porta muito importante, denominada de *Hadamard* é usada para forçar um *qubit* a se sobrepor. Quando aplicada aos elementos da base computacional $|0\rangle, |1\rangle$, tem-se que

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \Rightarrow$$
$$\begin{cases} H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2} = |+\rangle \\ H|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2} = |-\rangle \end{cases} \iff \begin{cases} H|+\rangle = |0\rangle \\ H|-\rangle = |1\rangle \end{cases}. \tag{7}$$

A operação gira o *qubit* π radianos em torno do eixo $x + z$ e pode ser visualizado na esfera de *Bloch* como uma rotação de $\frac{\pi}{2}$ radianos em torno do eixo y seguido de uma rotação de π radianos em torno do eixo x, como mostrado na Figura 5. Esta porta é utilizada

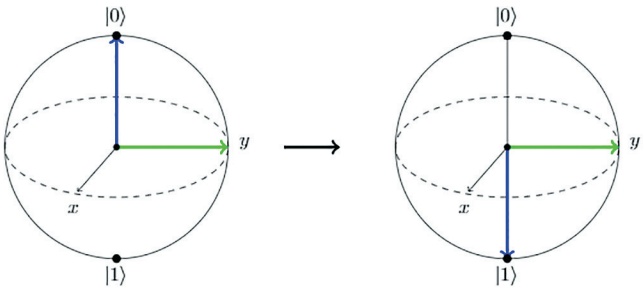

Figura 4: O estado do *qubit* antes e depois da porta Pauli-Y. O vetor em azul indica o estado inicial (final) do sistema e o vetor em verde indica a fase inicial (final) associada a esse vetor, como pode ser visto nas representações das esferas de *Bloch* à esquerda (direita).

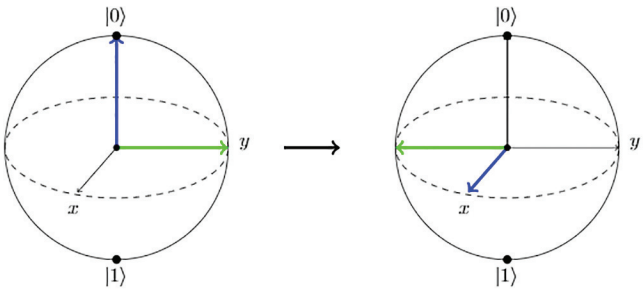

Figura 5: O estado do *qubit* antes e depois da porta *Hadamard*. O vetor em azul indica o estado inicial (final) do sistema e o vetor em verde indica a fase inicial (final) associada a esse vetor, como pode ser visto nas representações das esferas de *Bloch* à esquerda (direita).

para colocar o estado de um *qubit* em sobreposição com probabilidades iguais para os dois estados [30]. Tal porta é fundamental para implementação do Algoritmo de *Deutsch* e a Transformada de *Fourier* Quântica, por exemplo.

Um outro exemplo de porta lógica muito usado na computação quântica é a Porta CNOT, que atua sobre os estados de 2 *qubits* de entrada: o controle e o alvo. Portas controladas tem um *qubit* de controle e um *qubit* alvo, como é o caso da CNOT. Uma porta controlada atua conforme o estado do *qubit* de controle, sendo ativada apenas quando o *qubit* de controle estiver no estado $|1\rangle$. Os *qubits* de controle e alvo podem estar em superposição, como também, podem estar emaranhados [30]. A representação matricial da porta lógica quântica CNOT é a dada por:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{8}$$

A ação da CNOT pode ser representada na base computacional da seguinte maneira:

$$CNOT|a, b\rangle \rightarrow |a, a \oplus b\rangle, \tag{9}$$

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2021-0290

onde $a, b \in 0, 1$ e $\oplus$ é a operação denominada módulo 2. A operação módulo 2 é tal que:

$$0 \oplus 0 = 0,$$
$$0 \oplus 1 = 1,$$
$$1 \oplus 0 = 1,$$
$$1 \oplus 1 = 0.$$

Na Tabela 1 é mostrado explicitamente o que ocorre com o *qubit* de controle e com o *qubit* alvo quando a porta CNOT é aplicada.

Tabela 1: A porta CNOT opera em um registrador quântico de 2 *qubits*. Ela inverte o segundo *qubit* (o *qubit* alvo) se e somente se o primeiro *qubit* (o *qubit* de controle) for $|1\rangle$.

Antes		Depois	
Controle	Alvo	Controle	Alvo
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

Do ponto de vista geométrico, a representação da atuação da porta *CNOT* sobre um *qubit* pode ser descrita via esfera de *Bloch*, que pode ser visto nas Figuras 6 e 7, respectivamente.

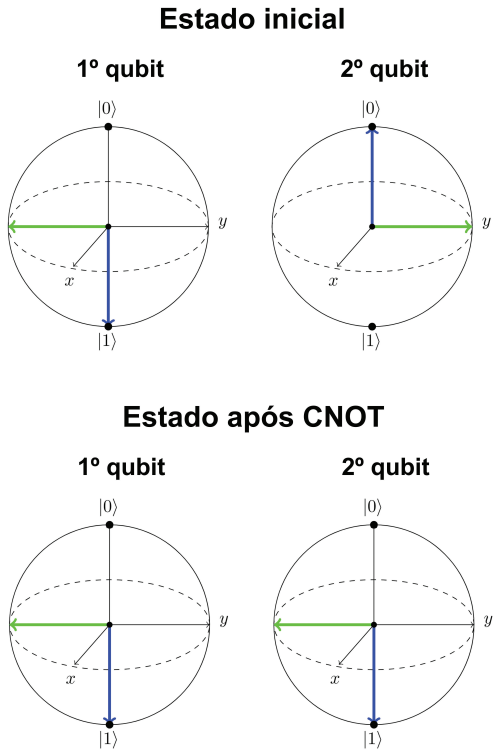

Figura 6: Estado inicial e final de dois *qubits*. O vetor em azul indica o estado inicial (final) do sistema e o vetor em verde indica a fase inicial (final) associada a esse vetor, como pode ser visto nas representações das esferas de *Bloch* à esquerda (direita).

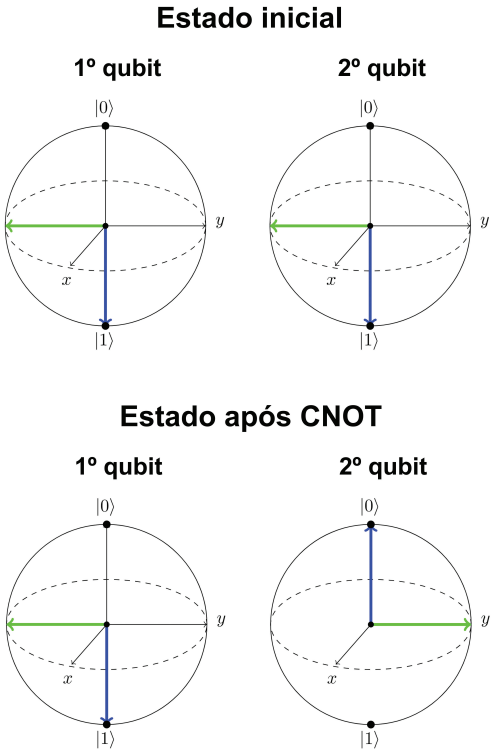

Figura 7: O estado do *qubit* antes e depois da porta CNOT. O vetor em azul indica o estado inicial (final) do sistema e o vetor em verde indica a fase inicial (final) associada a esse vetor, como pode ser visto nas representações das esferas de *Bloch* à esquerda (direita).

Na próxima seção vamos apresentar a plataforma da *IBM* bem como alguns detalhes sobre a maneira de acessá-la. Serão apresentados também alguns exemplos de simulações via *QisKit*, que é um Kit de Desenvolvimento de *Software* Quântico que permite a implementação dos algoritmos quânticos via nuvem, bem como emular os mesmos em um computador clássico localmente.

3. *IBM Quantum Experience*

A *IBM Quantum Experience* é uma plataforma online que fornece aos usuários (de maneira geral) o acesso a um conjunto de protótipos de processadores quânticos da *IBM* via nuvem, além de acesso a uma comunidade virtual para discussões, elaboração de tutoriais e uma documentação completa para auxiliar no desenvolvimento que vai de algoritmos quânticos mais simples até projetos mais complexos. Por se tratar de computação quântica via nuvem, os protótipos disponíveis podem ser utilizados remotamente. Alguns desses protótipos são de acesso livre, sendo necessário apenas criar uma conta na plataforma para ter acesso a todo o conteúdo da mesma.

A interação do usuário com os computadores quânticos dá-se através da construção de circuitos quânticos. Existem duas alternativas para projeção e

teste dos mesmos. A primeira e mais simples é através de uma interface de usuário online chamada *Quantum Composer*. Nela define-se a quantidade de *qubits* e *bits* clássicos a serem utilizados. As portas lógicas são apresentadas na forma de itens que podem ser arrastados com o *mouse* para serem adicionados ao circuito. O usuário também pode usar o *Quantum Composer* no modo de código de "montagem quântica" usando a linguagem de programação QASM [31]. A segunda alternativa é via um *framework*, conhecido como *QisKit*, sendo esta última a abordagem escolhida nesse trabalho. Falaremos mais sobre ele na próxima seção.

3.1. *QisKit*

O *QisKit* (*Quantum information software Kit*) é um *framework open-source*, que permite o desenvolvimento de *softwares* para computação quântica em nuvem. Ele fornece um conjunto de ferramentas para criar e manipular algoritmos quânticos e executá-los nos dispositivos da *IBM* bem como simulações na máquina do próprio usuário. O acesso pode ser diretamente na plataforma usando o *IBM Quantum Lab* ou instalando o pacote *Qiskit* (tendo o *Python* instalado). Ele segue o modelo de circuitos e pode ser usado para qualquer *hardware* quântico que siga esse modelo. Desta forma, qualquer pessoa pode usar esse *framework* para realizar simulações, tanto na sua máquina como também construir seus códigos e enviá-los para que sejam processados nos computadores quânticos da *IBM*.

Apesar de ser uma ferramenta nova para o desenvolvimento de algoritmos quânticos, as limitações vem sendo superadas com constantes atualizações. Entretanto, todo o potencial da plataforma só poderá ser explorado com o advento de novos computadores quânticos com boa robustez e com uma grande quantidade de *qubits*. A versão principal do *QisKit*[1] usa a linguagem de programação *Python*. Este é um ponto bastante positivo, pois *Python* é uma liguagem bastante acessível ao público[2].

Recentemente ocorreram algumas atualizações na plataforma[3] e bem como no *framework* QisKit[4], na qual ao iniciar o QisKit, o usuário possui encontra duas opções: iniciar o QisKit localmente (com *Python* instalado mais o pacote do QisKit), que no caso é uma forma mais privada ou começar com os *Jupyter Notebooks*, hospedados no *IBM Quantum Lab*[5].

3.2. Simulações via *QisKit* e interpretação dos resultados

No ambiente *QisKit* é possível realizar simulações, tanto do ponto de vista de uma máquina local (computador clássico) quanto via um *hardware* quântico real (*IBM*). Será apresentado agora o passo-a-passo para uma primeira simulação no *QisKit* através de um exemplo. Nele, vamos gerar o estado

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

e, na sequência, medir. Os passo-a-passo é o que segue:

- Comece importando o *QisKit*, como pode ser visto na Figura 8.

```
from qiskit import*
```

Figura 8: Passo-a-passo para utilização do *Qiskit*.

- Inicie dois *qubits* em $Qr = QuantumRegister$ e dois *bits* clássicos em $Cr = ClassicalRegister$, conforme a Figura 9;

```
nQ = 2 # <-- número de qubits
nC = 2 # <-- número de bits clássicos
Qr = QuantumRegister(nQ)
Cr = ClassicalRegister(nC)
```

Figura 9: Passo-a-passo para utilização do *Qiskit*.

- Em seguida, começamos a construir o circuito quântico composto pelos dois *bits* clássicos e pelos dois *bits* quânticos (*circuit = QuantumCircuit (Qr, Cr)*), onde o primeiro argumento é número de *bits* quânticos e o segundo argumento o número de *bits* clássicos, conforme apresentado na Figura 10;

```
circuit = QuantumCircuit(Qr , Cr )
```

Figura 10: Passo-a-passo para utilização do *Qiskit*.

- Na sequência, são adicionadas as portas, cujos comandos podem ser vistos na Figura 11. Do ponto de vista do circuito lógico, a implementação das portas pode ser vista na Figura 12;

```
# Porta Hadamard aplicada ao 1º qubit:
circuit.h(0)

# Porta CNOT: a ordem é controle, alvo:
circuit.cx(0 , 1)

# Medindo os dois qubits [0, 1],
# os resultados são armazenados
# em bits clássicos [0, 1] nesta ordem:
circuit.measure([ 0 , 1 ] , [ 0 , 1 ])

# Desenha o circuito:
circuit.draw(output='mpl')
```

Figura 11: Portas *Hadamard* e CNOT geradas via Qiskit.

[1] Para mais informações a respeito do *QisKit* e sua documentação acesse o link: https://qiskit.org/.
[2] Para uma leitura aprofundada sobre o QisKit, sua inicialização, comandos e implementação, *vide* [23].
[3] https://quantum-computing.ibm.com/
[4] https://qiskit.org/documentation/
[5] https://quantum-computing.ibm.com/lab/docs/iql/

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2021-0290

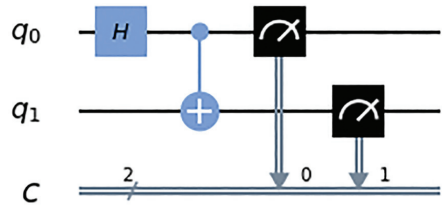

Figura 12: Portas geradas via *Qiskit*.

Aqui cabe um comentário importante sobre o circuito apresentado na Figura 12. Do ponto de vista clássico, um modelo computacional é composto por uma sequência de operações lógicas elementares realizadas por portas como NOT (negação) e AND (e) para solucionar funcões lógicas. Na computação quântica existe uma versão completamente análoga ao caso clássico. No modelo quântico, os *bits* dão lugar aos *qubits* e no lugar das portas clássicas surgem as portas quânticas. Basicamente os circuitos quânticos são compostos por portas lógicas e "fios". Os fios representam os transporte do *qubit* através do circuito, por exemplo, pode representar um fóton ou alguma outra partícula se movendo de um ponto a outro do espaço. A única regra é que os circuitos devem ser lidos da esquerda para a direita. No caso do circuito da Figura 12, inicialmente a porta H é aplicada ao primeiro qubit (caixa com a letra H na Figura); na sequência a porta CNOT é aplicada (haste com o símbolo + na Figura) e, por fim, os *bits* quânticos são medidos (representado no circuito pelo último desenho) e armazenados em *bits clássicos* $[0, 1]$ nessa ordem. Toda a explicação feita para esse circuito exemplo é válida também para os demais circuitos apresentados ao longo do texto (diferenciando apenas as portas a serem implementadas pelo mesmo) [2].

O próximo passo consiste em executar o código apresentado anteriormente na máquina local (computador clássico). Para tal, a simulação do circuito é realizada selecionando um *backend* como *qasm_simulator* (há outros, mas esse é de uso geral) do elemento *Aer* (elemento do *QisKit* que possui recursos para simulações quânticas por meio de computação de alto desempenho). O código para implementação pode ser visto na Figura 13. Na sequência, executa-se o circuito

```
simulator = Aer.get_backend('qasm_simulator')
```

Figura 13: Passo-a-passo para simulação via *Qiskit*.

no simulador e armazena-se o resultado na variável denominada *result*, como pode ser visto na Figura 14. Neste ponto estamos prontos para plotar o resultado

```
result = execute( circuit , backend = simulator).result( )
```

Figura 14: Passo-a-passo para simulação via *Qiskit*.

realizado via simulador. Para isso, primeiro, importa-se as ferramentas de visualização do *Qiskit*, cujo código pode ser visto na Figura 15.

```
from qiskit.visualization import plot_histogram
```

Figura 15: Passo-a-passo para simulação via *Qiskit*.

Por fim, é feita a plotagem dos resultados executados pelo simulador, através do código apresentado na Figura 16. o qual pode ser visto na Figura 17.

```
plot_histogram( result.get_counts(circuit) )
```

Figura 16: Passo-a-passo para simulação via *Qiskit*.

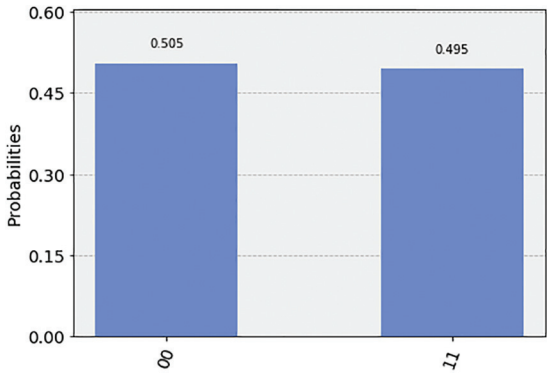

Figura 17: Resultado de uma simulação no *Qiskit* via máquina local.

Cabe ressaltar que esse resultado é feito sob a ótica do computador clássico.

Agora, será implementado o código em um *hardware* quântico real. Para isso, devem ser seguidos os seguintes passos:

- Registre-se na *IBM* usando um *token* que é obtido através da conta criada na *IMB Quantum Experience*. A implementação pode ser vista nas Figuras 18 e 19;

```
# Registrando-se na IBM Quantum Computing
from qiskit import IBMQ
#-------------------------------------------------------------
# Você pode obter o seu token em:
# https://quantum-computing.ibm.com/
QX_TOKEN = "Cole seu token aqui"
QX_URL = "https://quantumexperience.ng.bluemix.net/api"
#-------------------------------------------------------------
try :
    IBMQ.save_account(QX_TOKEN);
    print('Registado com sucesso!')
except :
    print('Algo deu errado.\nVocê inseriu o token correto?')
```

Figura 18: Passo-a-passo para simulação via *Qiskit*.

```
IBMQ.load_account()
```

Figura 19: Passo-a-passo para simulação via *Qiskit*.

- Verifica-se os *backends* disponíveis (no caso quais máquinas estão disponíveis bem como suas configurações e quantos usuários estão na fila), o que pode ser visto na Figura 20;

```
from qiskit.tools.monitor import  backend_overview

backend_overview()
```

Figura 20: Passo-a-passo para simulação via *Qiskit*.

- Escolhe-se um dos *backends* para execução do circuito em um dos protótipos da *IBM*. Após a execução dos comandos anteriores e, escolhendo uma máquina como *ibmqx*2 [42] por exemplo, deve-se seguir os passos presentes nas Figuras 21, 22 e 23;

```
provider=IBMQ.get_provider('ibm-q')
quantum_computer=provider.get_backend('ibmqx2')
```

Figura 21: Passo-a-passo para simulação via *Qiskit*.

```
execute_circuit=execute(circuit,backend=quantum_computer)
```

Figura 22: Passo-a-passo para simulação via *Qiskit*.

```
result=execute_circuit.result()
```

Figura 23: Passo-a-passo para simulação via *Qiskit*.

- Por fim, é feita a plotagem dos resultados obtidos. O código que implementa os resultados pode ser visto na Figura 24. Nesse exemplo, os resultados podem ser vistos na Figura 25.

```
plot_histogram(result.get_counts(circuit))
```

Figura 24: Passo-a-passo para simulação via *Qiskit*.

Analisando o resultado gerado pela simulação via *QisKit*, observa-se que o mesmo está de acordo com o esperado. No caso clássico, só podem ocorrer dois estados possíveis, com igual probabilidade para ambos ocorrerem.

Agora, analisando o resultado da simulação do mesmo código do ponto de vista de um *hardware* quântico real, nota-se que o resultado condiz com o esperado uma vez

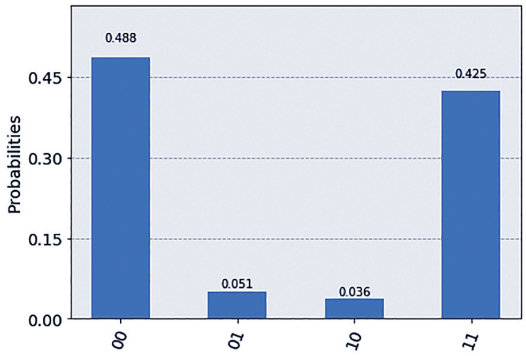

Figura 25: Resultado da mesma simulação do caso clássico, agora via um dos *hardwares* quânticos da *IBM*.

que, além dos estados obtidos para o caso clássico $|00\rangle$ e $|11\rangle$, observa-se também os estados intermediários $|01\rangle$ e $|10\rangle$, com menor probabilidade de ocorrência que os anteriores.

Um comentário importante a se fazer é que, comparando o último resultado com aquele obtido via computador clássico, pode-se observar que apesar do dispositivo quântico gerar o resultado esperado com alta probabilidade, o mesmo possui erros que resultam em estados não esperados teoricamente, mas com baixa probabilidade. Essas variações nos resultados estão associadas à estatística utilizada pelo *QisKit* para executar as linhas de comando. Há formas de melhorar o resultado, fazendo com que os resultados experimentais coincidam com o predito teoricamente, o que pode ser feito aumentando-se o número de repetições em cada circuito a ser executado [23]. Isso é importante até mesmo para corrigir o efeito natural de um circuito quântico que é a decoerência [1, 24, 25], que também pode afetar os resultados da simulação. Mesmo que esse efeito não seja detectado, um número alto de repetições nos cálculos aumenta de maneira considerável o resultado esperado em comparação com o previsto na teoria. Assim, à medida que se aumentam as repetições, melhora-se a estatística e, por consequência, os resultados experimental e teórico se aproximam consideravelmente. Tudo isso (claro!) sem levar em conta o efeito de decoerência o qual, por si só, já afeta as medidas em estados quânticos ao longo de sua evolução temporal.

4. Algoritmo de *Deutsch*

Em 1985, *Deutsch* apresentou o primeiro algoritmo quântico o qual mostrou o potencial da computação quântica. Esse algoritmo ficou conhecido como Algoritmo de *Deutsch* [6].

O algoritmo proposto por *Deutsch* resolve o seguinte o problema: dada a função lógica $f(x)$ de 1 *bit*, tal que $f : \{0,1\} \rightarrow \{0,1\}$, determine se $f(x)$ é constante ou balanceada. A função $f(x)$ é constante se $f(0) = f(1)$ e balanceada se $f(0) \neq f(1)$ [28]. Há quatro funções

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2021-0290

possíveis para o problema de *Deutsch*, como se pode ver na Tabela 2.

Tabela 2: As quatro funções possíveis do tipo $f : \{0,1\} \rightarrow \{0,1\}$.

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	0	0	1	1
1	0	1	0	1
$f_0(x), f_3(x) =$ constante				
$f_1(x), f_2(x) =$ balanceada				

Classicamente, para determinar se f é constante ou balanceada, é preciso computá-la duas vezes. Ou seja, calcular $f(0)$ e $f(1)$, compará-los e assim obter a resposta desejada.

Seguindo a ideia de *Deutsch*, imagine que se tenha uma caixa preta que computa uma função $f(x)$. Sendo a caixa preta um computador quântico, pode-se escolher a entrada como sendo uma sobreposição de $|0\rangle$ e $|1\rangle$. Comecemos pela apresentação do circuito apresentado na Figura 26, analisando o circuito da esquerda para a direita e observando as propriedades descritas de acordo com as operações a seguir. Uma observação importante é que, em computação quântica, a porta U_f apresenada na Figura 26 é chamada de "caixa preta" ou "oráculo", a qual consiste em uma operação unitária que computa a função $f(x)$. E com tal operação, o algoritmo segue os seguintes passos:

- O sistema é preparado no estado $|01\rangle$;
- Com duas portas *Hadamard* atuando sobre os *qubits* (uma atuando sobre o $|0\rangle$ e a outra atuando sobre o $|1\rangle$), o que leva a criação de um estado de superposição dos dois *qubits*;
- O oráculo atua sobre esse estado e computa $f(x)$. Esse cômputo é feito respeitando a operação

$$|x, y\rangle \rightarrow |x, y \oplus f(x)\rangle. \tag{10}$$

- Na sequência é aplicada uma porta *Hadamard* ao primeiro *qubit*;
- Por fim, é efetuada a medida sobre o primeiro *qubit*. Observa-se que, ao se realizar uma medição sobre o primeiro *qubit*, obtém-se uma propriedade global de $f(x)$, isto é, se o resultado obtido após a medida for 0, isso implica que a $f(x)$ e constante. Caso o resultado obtido for 1, a função $f(x)$ é balanceada.

Essa sequência de passos pode ser melhor visualizada na Figura 26, onde é representado o circuito quântico que implementa o Algoritmo de *Deutsch*.

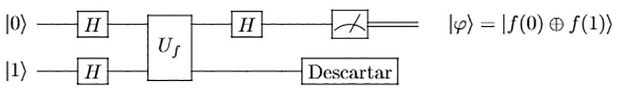

Figura 26: Circuito para o algoritmo de *Deutsch*.

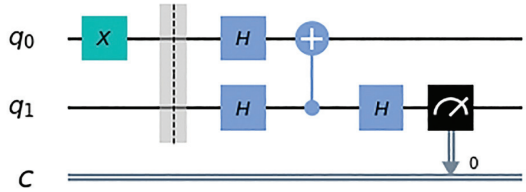

Figura 27: Circuito para o algoritmo de *Deutsch* com um registro no medidor clássico e com a porta $CX = CNOT$ no lugar do oráculo U_f.

Vamos analisar agora as simulações via máquina local e via os *chips* da *IBM*. Para isso, será utilizada uma versão simplificada do Algorítimo de *Deutsch* (para mais detalhes *vide* [27–29]), o qual procura extrair a informação deseja de f através de uma única medida.

Na Figura 27, a porta X aplicada em q_0 serve para levar o estado $|0\rangle$ ao estado $|1\rangle$: $X|0\rangle = |1\rangle$. Nesta simulação a porta $CX = CNOT$ será usada no lugar do oráculo U_f. Naturalmente agora o oráculo deixou de ser oráculo porque conhecemos a porta. Pode-se então calcular previamente o resultado esperado. Todavia, de acordo a Figura 27, uma medição do segundo *qubit* deve resultar em $|\varphi\rangle = |1\rangle$.

O código da função que gera o circuito da Figura 27 é mostrado na Figura 28. A simulação do mesmo segue os mesmos passos do exemplo apresentado para uma primeira simulação (para mais detalhes, *vide* Seção 3).

```
#---------------------------------------
# Função que monta o circuito de Deutsch
#---------------------------------------
def DEUTSCH_CNOT():
    from qiskit import QuantumRegister
    from qiskit import ClassicalRegister
    from qiskit import QuantumCircuit
#-----------------
    Q  = QuantumRegister(  2, name = 'q')
    C  = ClassicalRegister(1, name = 'c')
    QC = QuantumCircuit(Q, C, name = "DEUTSCH")
#-----------------
    QC.x(Q[0])
    QC.barrier()
    QC.h(Q)
    QC.cx(Q[1], Q[0])
    QC.h(Q[1])
    QC.measure(Q[1],C[0])
    print('DEUTSCH(): FUNCIONA!')
    return QC
```

Figura 28: Código para implementação do circuito representativo do algoritimo de *Deutsch*.

Tendo em vista qual será o resultado, temos 100% de probabilidade de obter $|1\rangle$ conforme pode ser observado na Figura 29, onde temos a simulação do circuito da Figura 27 na máquina local (computador clássico).

E nos computadores quânticos da *IBM*? Agora a manipulação de estados quânticos é real. Os resultados

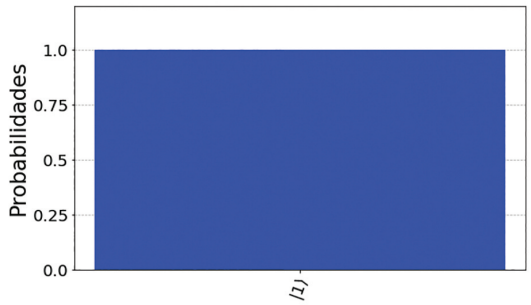

Figura 29: Simulação do circuito exibido na Figura 27: computador clássico.

na Figura 30 foram obtidos com os chips quânticos da IBM_ibmqx2, ibmq_ourense e ibmq_16_melbourne. O número de vezes que o circuito foi simulado em cada chip foi de 2^{13} repetições (*shots*), como também na máquina clássica (Figura 29). Pode-se observar na Figura 30, que o estado $|0\rangle$ também foi obtido com baixíssima probabilidade, diferente do resultado da Figura 29. A explicação para esse resultado deve-se ao fato de que sistemas quânticos são extremamente sensíveis e sua interação com o ambiente produz erros ao longo do processamento. Esses dispositivos são compostos por qubits supercondutores e são resfriados a baixíssima temperatura (da ordem de milikelvin) [37]. Para realizar a computação, o tempo de coerência dos *qubits* deve ser maior que o tempo do processo de computação (implementação das portas logicas). Processos físicos aleatórios e incontroláveis nos equipamentos de controle e medição de *qubits* ou no ambiente local ao redor do processador quântico são fontes de ruído que levam à descoerência e reduzem a fidelidade operacional dos qubits [38].

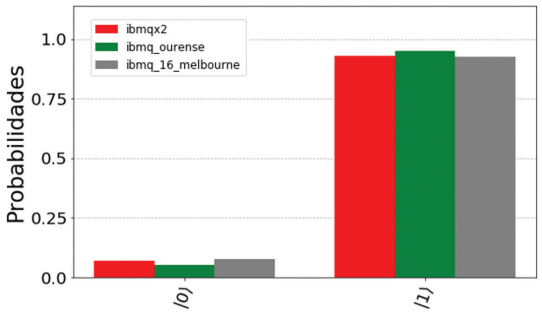

Figura 30: Experimento nos computadores quânticos da *IBM* do circuito exibido na Figura 27. As barras em vermelho, verde e cinza é a simulação remota nas máquinas quânticas da IBM ibmqx2 de 5 *qubits*, ibmq_ourense de 5 *qubits* e ibmq_16_melbourne de 15 *qubits*, respectivamente.

Na Figura 27 tem-se um medidor no registro clássico. Porém, se forem usados conforme a Figura 31, com a porta $CX = CNOT$ no lugar do oráculo U_f, o estado final do sistema será $(|10\rangle - |11\rangle)/\sqrt{2}$, com 50% de probabilidade para $|10\rangle$ e com 50% de probabilidade para $|11\rangle$.

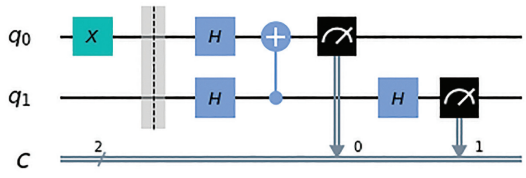

Figura 31: Circuito para o algoritmo de *Deutsch* com dois registros no medidor clássico e com a porta $CX = CNOT$ no lugar do oráculo U_f.

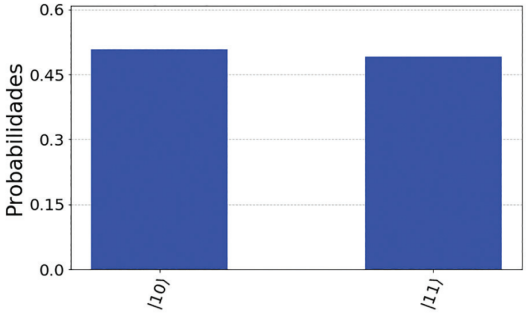

Figura 32: Simulação do circuito exibido na Figura 31: computador clássico.

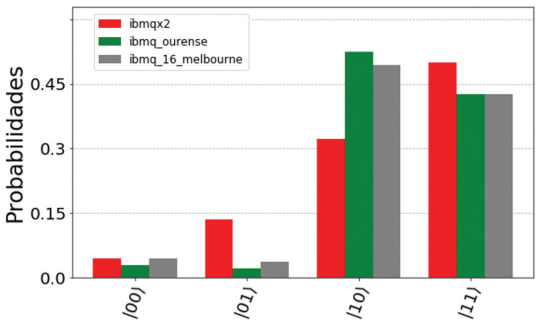

Figura 33: Experimento nos computadores quânticos da *IBM* do circuito exibido na Figura 31. As barras em vermelho, verde e cinza é a simulação remota nas máquinas quânticas da IBM ibmqx2 de 5 *qubits*, ibmq_ourense de 5 *qubits* e ibmq_16_melbourne de 15 *qubits*, respectivamente.

Na Figura 32 temos a simulação do circuito da Figura 31 na máquina local (computador clássico).

Os resultados na Figura 33 foram obtidos com os chips quânticos da IBM ibmqx2, ibmq ourense e ibmq16 melbourne, conforme feito no caso anterior. O número de vezes que o circuito foi simulado em cada chip foi de 2^{13} repetições (*shots*), bem como na máquina clássica (Figura 32).

Ao comparar os resultados da simulação artificial e em um dispositivo quântico real, pode-se observar no computador clássico que o resultado é equivalente a um sistema ideal. Nos *chips* da *IBM*, além dos resultados esperados teoricamente, há recorrência de outros estados de acordo com as combinações possíveis para os *q-bits* mensurados. Essas recorrências ocorrem com baixa probabilidade e são consideradas erros ao longo do

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2021-0290

processamento pois, como mencionado anteriormente, esses dispositivos são muito sensíveis. Apesar desses pequenos erros, os resultados das simulações nos *chips* da *IBM* condizem perfeitamente com a teoria.

5. Transformada de *Fourier* Quântica

Uma da formas mais eficientes de resolver problemas mais complexos (matematicamente falando) consiste em transforma-los em um problema mais simples cuja solução seja conhecida. E uma delas é a Transformada de *Fourier*, que na Computação Quântica é extremamente útil para computar determinadas tarefas de forma muito rápida [2].

A Transformada de *Fourier* discreta é uma transformada (DFT) que atua sobre conjuntos de dados discretos. Em sua notação usual, recebe como entrada um vetor de números complexos, $x_0, \ldots, x_{N-1}$, cujo comprimento N é um parâmetro fixo. A sua saída são os dados transformados, os quais gerarm um vetor também complexo $y_0, \ldots, y_{N-1}$, definido por

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \exp\left[\frac{2\pi i}{N} jk\right] x_j, \tag{11}$$

onde $k = 0, \ldots, N-1$. A Transformada de *Fourier* Quântica (QFT) atua da mesma maneira, porém em uma notação diferente. Em uma base ortonormal $|0\rangle, \ldots, |N-1\rangle$, sua transformação sobre um estado arbitrário é dada por:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \exp\left[\frac{2\pi i}{N} jk\right] |k\rangle. \tag{12}$$

Equivalentemente, sua ação sobre um estado arbitrário é:

$$\sum_{j=0}^{N-1} x_j |j\rangle \rightarrow \sum_{k=0}^{N-1} y_k |k\rangle, \tag{13}$$

em que cada amplitude y_k é a transformada de *Fourier* discreta das amplitudes x_j.

Seja o registro quântico de n *qubits*:

$$|j\rangle_n = |j_{n-1} j_{n-2} \cdots j_0\rangle, \tag{14}$$

onde tem-se que $|0\rangle_n = |00\cdots0\rangle$, $|N-1\rangle_n = |11\cdots1\rangle$ com $N = 2^n$. A QFT aplicada a $|j\rangle_n$ é:

$$\begin{aligned} \mathcal{F}|j\rangle_n &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left[\frac{2\pi i}{N} kj\right] |k\rangle_n \\ &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{kj} |k\rangle_n, \end{aligned} \tag{15}$$

onde $j = 0, \ldots, N-1$ e $\omega_N^{kj} = \exp\left[\frac{2\pi i}{N} kj\right]$.

A equação (15) pode ser reescrita de maneira a tornar mais fácil a construção do circuito quântico para a QFT. Entretanto, esse passo demanda uma extensa álgebra, a qual não faremos aqui[6].

Como exemplo, podemos observar na Figura 34 o algoritmo para a implementação da transformada de *Fourier* quântica para um sistema de dois *qubits*.

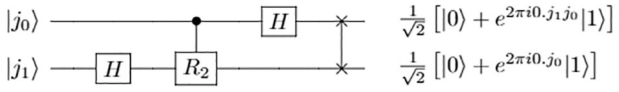

Figura 34: Circuito que calcula a QFT no registro de $n = 2$ *qubits* $|j\rangle_2 = |j_1 j_0\rangle$.

De maneira análoga ao cálculo e à construção da transformada quântica de *Fourier*, é possível também realizar operações através de sua transformada inversa. O procedimento é simples e a ideia é tomar a inversa das operações que contemplam a transformada direta, por assim dizer. Um exemplo de algorítmo para execução da transformada inversa pode ser visto na Figura 35, também no contexto de dois *qubits*[7]

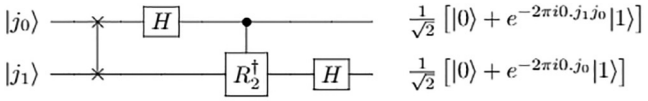

Figura 35: Circuito que calcula a QFT inversa no registro de $n = 2$ *qubits* $|j\rangle_2 = |j_1 j_0\rangle$.

A Transformada de *Fourier* Quântica é um dos ingredientes chave de muitos algoritmos quânticos importantes, como por exemplo o Algoritmo de *Shor* citado ao longo deste trabalho. Na literatura alguns trabalhos recentes que versam sobre a QFT são [32–35].

Agora serão apresentados os resultados decorrentes da simulação via *chips* da *IBM*. O código que implementa a QFT para 2 *qubits* é apresentada na Figura 36. Para implementar em um estado arbitrário bipartido, o mesmo deve ser preparado previamente e então a QFT deve ser aplicada. Já na Figura 37, são apresentado os circuitos simulados com os estados iniciais $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$ de entrada. Ao lado de cada circuito são apresentados os histogramas com os resultados das medidas. Independente do estado inicial de entrada, os estados de saída sobrepostos $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, teoricamente tem probabilidade de 0.25 cada. Para a simulação realizada nos três computadores quânticos, os resultados apresentaram oscilações nas amplitudes de probabilidade em relação a teoria, como também comparada a simulação no computador clássico. A barra azul do histograma é o resultado da simulação no computador clássico. Os resultados em vermelho, verde e cinza foram obtidos com os *chips* quânticos da IBM ibmqx2, ibmq_ourense e ibmq_16_melbourne, respectivamente. Nas Figuras 38 e 39 os histogramas seguem o mesmo padrão.

[6] Recomendamos ao leitor consultar a referência [2] para uma demonstração mais detalhada.
[7] Para maiores detalhes, *vide* [36].

```
#-----------------------------------------------------
# Função que monta o circuito da QFT de '2' qubits:
#-----------------------------------------------------
def QFT_2qubit():
    from qiskit import QuantumRegister
    from qiskit import ClassicalRegister
    from qiskit import QuantumCircuit
#-----------------
    Q  = QuantumRegister(  2, name = 'j')
    C  = ClassicalRegister(2, name = 'c')
    QC = QuantumCircuit(Q, C, name = "QFT_2qubit")
#-----------------
    QC.h(Q[1])
    Rk(QC, Q[0], Q[1], 2, 1.0)
    QC.barrier()
    QC.h(Q[0])
    QC.barrier()
    QC.swap( Q[0], Q[1] ) # <-- PORTA SWAP
    QC.barrier()
    QC.measure(Q, C)
    print('QFT_2qubit(): FUNCIONA!')
    return QC
#-----------------------------------------------------
# Função que define a porta Rk usando a porta cu1:
#-----------------------------------------------------
def Rk(QC, Q_CONTROL, Q_TARGET, k, a):
    import math as MT; PI = MT.pi
    QC.cu1( a*2.0*PI/2.0**k, Q_CONTROL, Q_TARGET)
    return None
```

Figura 36: Código para implementação do circuito representativo da Transformada de *Fourier* Quântica.

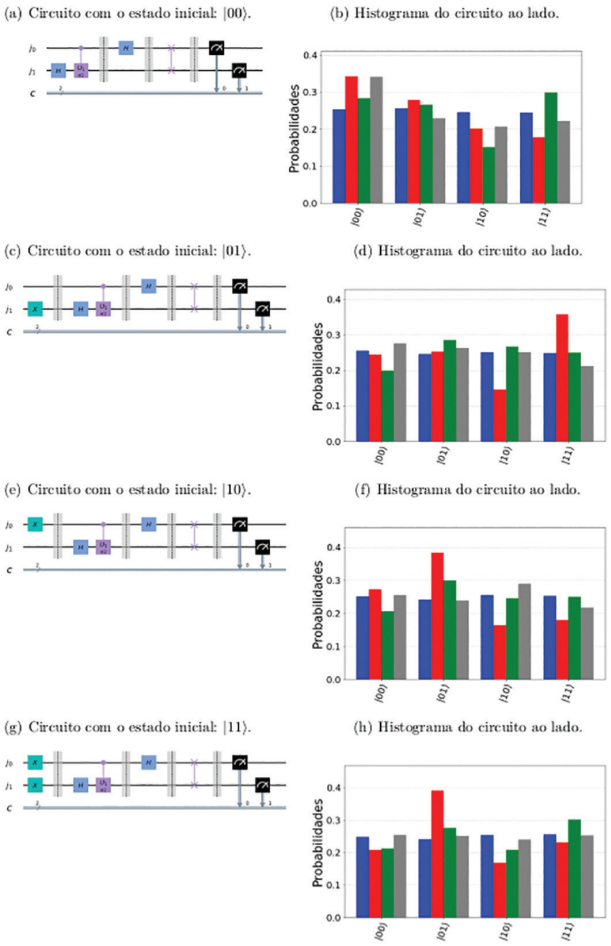

Figura 37: Circuitos e histogramas: As barras em azul representam a simulação artificial (computador clássico). Já as barras em vermelho, verde e cinza representam a simulação feita remotamente nas máquinas quânticas da *IBM* ibmqx2 de 5 *qubits*, ibmq_ourense de 5 *qubits* e ibmq_16_melbourne de 15 *qubits*, respectivamente.

A Figura 38 apresenta o circuito simulado com estado inicial $|000\rangle$ de entrada. Abaixo do circuito são apresentados os histogramas de duas simulações para o mesmo. Teoricamente os estados de saída tem probabilidade de 0.125 cada. A barra azul nos histogramas são os resultados da simulação realizada no computador clássico, os resultados são idênticos aos dados teóricos. As barras vermelhas, verdes e cinzas nos histogramas são os resultados da simulação realizada nos computadores quânticos. Outro detalhe a se observar nos histogramas (a) e (b) da Figura 38 é que a cada simulação para o mesmo sistema no computares quânticos, as probabilidades para cada estado variam a cada simulação efetuada em cada *chip* da IBM.

A Figura 39 apresenta a simulação da QFT inversa aplicada a QFT aplicada ao estado $|00\rangle$. Abaixo do circuito são apresentados os histogramas de duas simulações com os resultados das medidas. Teoricamente, como visto na seção anterior, o estado de saída deve ser $|00\rangle$ com probabilidade de 100% cada. A barra azul nos histogramas são os resultados da simulação realizada no computador clássico e os resultados são totalmente condizentes com a teoria. As barras vermelhas, verdes e cinzas nos histogramas são os resultados da simulação realizada nos computadores quânticos. Nos *chips* quânticos há recorrência de outros estados não previstos teoricamente, porém com pequena probabilidade. Também observar-se nos histogramas (a) e (b) da Figura 39 que a cada simulação para o mesmo sistema no computares quânticos, as probabilidades para cada estado variam com pequena probabilidade para os estados $|01\rangle, |10\rangle, |11\rangle$.

Pode-se observar que, assim como no caso do algoritmo de *Deutsch*, os *chips* quânticos apresentam uma determinada taxa de erro. Porém, os resultados obtidos via simulação estão próximos aos previstos teoricamente. Um detalhe importante, ao comparar os algoritmos, está na implementação das portas lógicas. Cada porta lógica tem um erro associado à sua implementação e o sucesso de sua realização depende da robustez do sistema perante ruídos e o tempo de coerência dos *qubits*. As operações controladas possuem um erro maior associado e quanto maior o número dessas operações, maiores serão os erros na implementação do algoritmo [39–41]. Evidentemente, a QFT possui mais erros em sua implementação, pois demanda muitas operações controladas, apenas a porta $SWAP$ é formada por três $CNOT$. Quanto maior a dimensão do sistema multi-*qubit*, o

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2021-0290

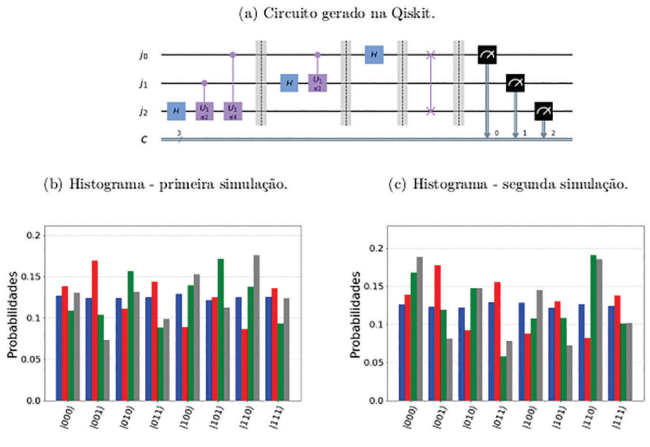

Figura 38: Circuitos e Histogramas de 3 *qubits* com estado inicial $|000\rangle$: As barras em azul representam a simulação artificial (computador clássico). As barras em vermelho, verde e cinza representam a simulação feita remotamente nas máquinas quânticas da *IBM* $ibmqx2$ de 5 *qubits*, ibmq_ourense de 5 *qubits* e ibmq_16_melbourne de 15 *qubits*, respectivamente.

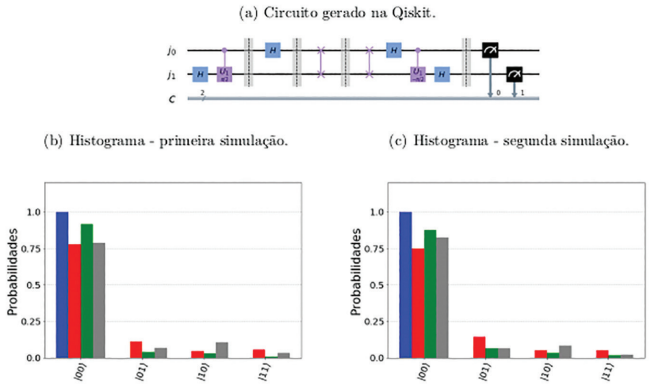

Figura 39: Circuito e histogramas da QFT inversa aplicada a QFT de 2 *qubits* com estado inicial $|00\rangle$: As barras em azul representam a simulação artificial (computador clássico). As barras em vermelho, verde e cinza representam a simulação realizada remotamente nas máquinas quânticas da *IBM* $ibmqx2$ de 5 *qubits*, ibmq_ourense de 5 *qubits* e ibmq_16_melbourne de 15 *qubits*, respectivamente.

acúmulo de erros se torna ainda maior. Isso por si só já é um grande desafio na construção desses sistemas.

6. Considerações Finais

Os resultados apresentados neste trabalho, embasado em [36], são de simulações realizadas em uma máquina clássica bem como nos respectivos *chips* quânticos da *IBM* $ibmqx2$ de 5 *qubits*, ibmq_ourense de 5 *qubits* e ibmq_16_melbourne de 15 *qubits* via nuvem. Na máquina clássica, os resultados obtidos foram de um sistema ideal, ou seja, sem erros experimentais. Nos protótipos da *IBM*, apesar das oscilações nas amplitudes de probabilidade e recorrências em estados não esperados de acordo a teoria, os resultados foram bem satisfatórios. Conforme discutido ao longo do trabalho, os erros gerados pelas maquinas quânticas, devem-se a interação do sistema quântico com o ambiente ou também do efeito de decoerência quântica. Para contornar esses erros é necessária a implementação de códigos de correção de erros e aumentar o número de *shots* (quantidade de repetições nas medições), melhorando as estatísticas dos resultados, aproximando os dados simulados dos dados previstos teoricamente.

Com relação a arquitetura dos processadores utilizados, a eficiência dos mesmos (em termos de cálculos) comparado aos cálculos feitos em uma máquina clássica, é medida comparando as energias entre dois níveis próximos (dois *qubits*) do circuito construído para a implementação dos algoritimos em estudo. Isso determina por qual porta o processo se dará no circuito. Assim, a depender do processador, ter-se-á um melhor desempenho e uma mitigação dos erros inerentes ao processo de implementação dos cálculos feito via os *chips* da *IBM*, o que reduz consideravelmente os erros e o efeito de decoerência, aproximando o resultado dos valores previstos em teoria, uma vez que cada um tem um tempo de relaxação, que o tempo em que o *qubits* se mantém estável até sofrer algum tipo de interferência do ambiente, sendo este tempo intrinsecamente relacionado ao desempenho dos processadores e aos resultados que venham a ser obtidos após os cálculos. Alguns dados experimentais podem ser encontrados com mais detalhes em [42] ($ibmqx2$), [43] (ibmq_16_melbourne) e [44] (ibmq_ourense). Cabe salientar que esses processadores vão sendo melhorados e aperfeiçoados com o passar do tempo, o que torna a implementação de algoritmos bem mais ágil, do ponto de vista de processamento.

Os algoritmos estudados tem papel fundamental, não só na computação quântica, mas em tecnologias quânticas que estão e serão desenvolvidas. Por exemplo, sem a transformada de *Fourier* quântica não há como implementar o algoritmo de *Shor*. Com a corrida de grandes empresas pela "supremacia quântica", a aplicação do algoritmo de *Shor* é apenas uma questão de tempo. Com essa aplicação, os sistemas de segurança digital atuais terão que buscar novas alternativas. Inclusive, já há comercialização de chaves quânticas para a proteção de informações ultra secretas.

Cabe ressaltar que esse trabalho é de grande valia para a introdução de estudantes de Ciências Exatas e Engenharia nesse tema de pesquisa promissora, a qual envolve e associa teoria e prática, através do desenvolvimento de tecnologias para a área, bem como para o ensino dos conceitos sobre informação e computação quântica. A disponibilização da IBM de uma plataforma e de recursos *open-source* para implementação de algoritmos quânticos é de suma importância para atrair os estudantes, uma vez que os estudantes podem implementar e simular resultados de modelos reais, visando a construção de novas tecnologias para instrumentação nessa área tão promissora que é a Informação Quântica.

Para os interessados, disponibilizamos os códigos das simulações no *GitHub*, que podem ser acessados através do link: https://github.com/WarleyMarcos/Paper_RBEF.

Agradecimentos

Os autores agradecem à Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM) pelo apoio. Os autores também agradecem à professora Neila Mara Gomes de Oliveira e ao professor Fabiano Alan Serafim Ferrari pelas frutíferas discussões.

Referências

[1] A.C. Santos, Rev. Bras. Ens. Fís. **39**, e1301 (2017).
[2] M.A. Nielsen e I.L. Chuang, *Computação Quântica e Informação Quântica* (Bookman, Porto Alegre, 2005).
[3] R.P. Feynman, Int. J. Theor. Phys. **21**, 467 (1982).
[4] A.A. Barbosa, *Um Simulador Simbólico de Circuitos Quânticos*. Dissertação de Mestrado, Universidade Federal de Campina Grande, Campina Grande (2007).
[5] A. Caldeira, Rev. Bras. Ens. Fís. **40**, e4211 (2018).
[6] D. Deutsch, Proc. R. Soc. Lond. A **400**, 97 (1985).
[7] P.W. Shor, Proc. A. Symp. Found. Comp. Sci. **35**, 124 (1994).
[8] M.K. Okurama, *Números primos e criptografia RSA*. Dissertação de Mestrado, Universidade de São Paulo, São Paulo (2014).
[9] M.M. Dullius, *O problema do logarítimo discreto*. Dissertação de Mestrado, Porto Alegre (2007).
[10] http://www.cbpf.br/~qbitrmn/didaticos/cqiq.pdf
[11] F.G.G.P.O. Siva, *O impacto da computação quântica na Criptografia moderna*. Dissertação de Mestrado, Universidade do Minho, Braga (2013).
[12] G. Moore, Electronics **38**, 144 (1965).
[13] F. Mattielo, G.G. Silva, R.G.G. Amorin e W.B. Silva, Cad. de Fís. da UFES **10**, 31 (2012).
[14] S.S. Feitosa, C.L. Nogueira, J.K. Vizzotto, Revista ComInG-Communications and Innovations Gazette **1**, 46 (2016).
[15] A. Ponnath, arXiv:cs/0602096 (2016).
[16] https://blogs.scientificamerican.com/observations/the-problem-with-quantum-computers/, acessado em 17/07/2021.
[17] L. Wang e C.A. Alexander, American Journal of Electrical and Electronic Engineering **8**, 43 (2020).
[18] F. Arute, K. Arya, R. Babbush, D. Bacon, J.C. Bardin, R. Barends, R. Biswas, S. Boixo, F.G.S.L. Brandao, D.A. Buell et al., Nature **574**, 505 (2019).
[19] Google AI quantum and Collaborators, Science **369**, 1084 (2020).
[20] https://www.ibm.com/blogs/research/2020/08/quantum-research-centers/, acessado em 17/07/2021.
[21] http://www-03.ibm.com/press/us/en/pressrelease/49661.wss
[22] M.T.O. Cunha, *Noções de Informação Quântica* (IMPA-SBM, 2007).
[23] G.F. Jesus, M.H.F. Silva, T.G.D. Netto, L.Q. Galvão, F.G.O. Souza e C. Cruz, Rev. Bras. Ens. Fis. **43**, e20210033 (2021).
[24] M. Schlosshauer, Physics Report **831**, 1 (2019).
[25] H.D. Zeh, Foundation of Physics **1**, 69 (1970).
[26] Ryan LaRose, Quantum **3**, 130 (2019).
[27] R. Cleve, A. Ekert, C. Macchiavello e M. Mosca, Proc. Math. Phys. Eng. Sci. **454**, 339 (1998).
[28] G.E.M. Cabral, A.F. Lima e B.L. Jr., Rev. Bras. Ens. Fís. **26**, 109 (2004).
[29] P.H. Grosman, D.G. Braga e J.A. Huguenin, Rev. Bras. Ens. Fís. **41**, e20180201 (2019).
[30] W.J.N. Silva, *Uma introdução à Computação Quântica*. Monografia, Universidade de São Paulo, São Paulo (2018).
[31] W. Cross, L.S. Bishop, J.A. Smolin e J.M. Gambetta, arXiv:1707.03429v2 (2017).
[32] A. Martin, L. Lamata, E. Solano e M. Sanz, Phys. Rev. Res. **2**, 013012 (2020).
[33] R. Azaka, K. Sakai e R. Yahagi, Quantum Inf. Process. **19**, 1 (2020).
[34] A. Zhang, X. Wang e S. Zhao, CCF Trans. High Perform. Comput. **2**, 221 (2020).
[35] Y. Nan, Y. Su e D. Maslov, Npj Quantum Inf. **6**, 1 (2020).
[36] W.M.S. Alves, Algoritmos Quânticos Usando a Plataforma IBM Quantum Experience. Monografia, Universidade Federal dos Vales do Jequitinhonha e Mucuri (2020).
[37] M. Naghiloo, arXiv:1904.09291 [quant-ph], (2019).
[38] P. Krantz, M. Kjaergaard, F. Yan, T.P. Orlando, S. Gustavsson e W.D. Oliver, Applied Phys. Rev. **6**, 021318 (2019).
[39] https://oxford.universitypressscholarship.com/view/10.1093/acprof:oso/9780199681181.001.0001/acprof-9780199681181
[40] M. Kjaergaard, M.E. Schwartz, J. Braumüller, P. Krantz, J.I.J. Wang, S. Gustavsson e W.D. Oliver, Annual Review of Condensed Matter Physics **11**, 369 (2020).
[41] S. Kwon, A. Tomonaga, G.L. Bhai, S.J. Devitt e J.S. Tsai, Journal of Applied Physics **129**, 041102 (2021).
[42] https://github.com/Qiskit/ibmq-device-information
[43] https://github.com/Qiskit/ibmq-device-information
[44] Y. Du, T. Huang, S. You, M. Hisieh e D. Tao, arxiv:2010.10217v2 (2020).