UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-graduação em Física

Warley Marcos Santos Alves

RECONSTRUÇÃO PTICOGRÁFICA DE ESTADOS DE MÚLTIPLOS QUBITS EM UM COMPUTADOR QUÂNTICO

Belo Horizonte
2023

Warley Marcos Santos Alves

RECONSTRUÇÃO PTICOGRÁFICA DE ESTADOS DE MÚLTIPLOS QUBITS EM UM COMPUTADOR QUÂNTICO

Dissertação apresentada ao Programa de Pós-Graduação em Física do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para obtenção do título de Mestre em Ciências.

Orientador: Leonardo Teixeira Neves

Belo Horizonte

2023

Dados Internacionais de Catalogação na Publicação (CIP)

A474r Alves, Warley Marcos Santos.
Reconstrução pticográfica de estados de múltiplos qubits em um computador quântico / Warley Marcos Santos Alves. – 2023.
140 f. : il.

Orientador: Leonardo Teixeira Neves.
Dissertação (mestrado) – Universidade Federal de Minas Gerais, Departamento de Física.
Bibliografia: f. 103-110.

1. Tomografia quântica. 2. Computação quântica. 3. Qubits. I. Título. II. Neves, Leonardo Teixeira. III. Universidade Federal de Minas Gerais, Departamento de Física.

CDU – 530.145:004 (043)

Ficha catalográfica elaborada por Romário Martins Ribeiro – CRB6 3595
Biblioteca Professor Manoel Lopes de Siqueira – Departamento de Física - UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM FÍSICA

FOLHA DE APROVAÇÃO

A presente dissertação, intitulada **"Reconstrução pticográfica de estados de múltiplos qubits em um computador quântico"**, de autoria de **WARLEY MARCOS SANTOS ALVES**, submetida à Comissão Examinadora, abaixo-assinada, foi aprovada para obtenção do grau de **MESTRE EM FÍSICA** em primeiro de dezembro de 2023.

Belo Horizonte, 01 de dezembro de 2023.

Prof. Leonardo Teixeira Neves
Orientador do aluno
Departamento de Física/UFMG

Prof. Sebastião José Nascimento de Pádua
Departamento de Física/UFMG

Prof. Alexandre Ferreira Ramos
Escola de Artes, Ciências e Humanidades/USP

Documento assinado eletronicamente por **Leonardo Teixeira Neves**, **Professor do Magistério Superior**, em 04/12/2023, às 13:17, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Alexandre Ferreira Ramos**, **Usuário Externo**, em 04/12/2023, às 15:10, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Sebastião Jose Nascimento de Padua**, **Presidente de comissão**, em 16/12/2023, às 23:11, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Agradecimentos

Primeiramente, expresso minha profunda gratidão ao meu orientador, Professor Dr. Leonardo Teixeira Neves, por todo o apoio, disponibilidade, paciência e valiosos conselhos ao longo deste período. Sua excelência profissional não apenas orientou, mas também inspirou meu percurso acadêmico.

Aos pilares da minha vida, meus pais, devo um reconhecimento especial. Eles estiveram ao meu lado, oferecendo apoio incondicional e amor inabalável, mesmo nos momentos mais desafiadores. Sou profundamente grato por moldarem a pessoa que sou hoje.

À minha querida irmã, agradeço pelo apoio constante e pela amizade verdadeira que sempre compartilhamos.

À minha madrinha, Marizete, e seu esposo, Antônio, expresso meu agradecimento pelo apoio, atenção e carinho ao longo de toda a jornada acadêmica. Ao meu primo e grande amigo, Antônio Gustavo, agradeço pelos momentos memoráveis que compartilhamos.

Aos meus avós, agradeço pelo apoio de sempre, atenção e carinho ao longo de toda a minha vida.

À minha querida namorada, agradeço pelo seu apoio, amor constante e carinho que tornam os meus dias mais alegres.

Ao Rogerio Ruivo e ao Carlos Speglich, sou grato pelo apoio e pelas oportunidades proporcionadas, que me permitiram mergulhar ainda mais no fascinante mundo da computação quântica.

Aos professores do Departamento de Física do ICEX, com os quais tive aulas e que proporcionaram uma valiosa experiência de aprendizado, mesmo diante dos desafios do ensino virtual durante a pandemia. Também gostaria de estender meus agradecimentos aos colegas de classe, cuja colaboração e apoio tornaram essa jornada de aprendizado ainda mais enriquecedora.

Registro aqui meu agradecimento ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo financiamento da minha pesquisa durante os anos dedicados ao mestrado em Física.

Por fim, agradeço à International Business Machines (IBM) por conceder-me uma conta de pesquisador, possibilitando a realização de experimentos nos computadores quânticos disponibilizados, os quais desempenharam um papel fundamental neste trabalho.

Resumo

O desenvolvimento de tecnologias quânticas de segunda geração (sensores quânticos, computadores quânticos e internet quântica) vem sendo realizado por grandes empresas e tem recebido investimentos estatais da ordem de bilhões de dólares em vários países. Dentre essas tecnologias, o computador quântico ganhou um destaque maior na mídia por sua promessa de resolver problemas de forma muito mais rápida e eficiente que computadores clássicos, o que poderia produzir inúmeros avanços científicos com impactos na sociedade e na indústria. Nos últimos anos, empresas como IBM, Google, Microsoft, entre outras, apresentaram grandes progressos na construção dos primeiros protótipos de hardware e software quânticos. No entanto, esses protótipos são dispositivos que operam com poucos qubits e são bastante afetados por ruído, o que ainda os impedem de realizar computação realmente útil. Em particular, a IBM tem disponibilizado o acesso remoto a algumas de suas máquinas para o público geral, além de acesso especial a pesquisadores selecionados no seu programa de pesquisadores. Usando esse recurso, neste trabalho nós estudamos um método para reconstrução de estados puros de um sistema de múltiplos qubits em um computador quântico. No contexto da computação quântica, esta é uma ferramenta muito útil, por exemplo, para determinar estados de entrada ou de saída de um dado algoritmo, ou para a caracterização das operações lógicas que constituem o algoritmo. O método que estudamos é conhecido como *pticografia quântica*, onde um estado puro é submetido a um conjunto de projeções com algum grau de interseção entre si, seguidas de uma medição projetiva em uma única base ortonormal. Os resultados gerados no processo servem como entrada para um algoritmo iterativo que estima o estado. O nosso estudo se dividiu entre simulações e experimentos. As simulações foram realizadas em cenários ideais sem ruído e cenários mais realísticos com ruído, onde o método foi demonstrado com êxito para uma variedade de estados de até 6 qubits. Nos experimentos, obtivemos bons resultados para a reconstrução de estados de até 4 qubits; os resultados para um número maior foram fortemente afetados pelo ruído nos dispositivos. As vantagens e limitações do método pticográfico são discutidas em comparação com outros métodos de reconstrução de estados. Além de demonstrar a pticografia quântica em um computador quântico nós também obtivemos alguns avanços para o método. O primeiro foi uma modificação no algoritmo iterativo de reconstrução, que melhora a sua convergência e, consequentemente, a estimativa do estado. O segundo foi a utilização de bases de medição que são menos afetadas por ruído em computador quântico. Este segundo avanço ainda é parte de um estudo preliminar mas aponta caminhos promissores para investigações futuras.

Palavras-chave: Pticografia quântica, Tomografia de estados quânticos puros, Computação quântica, Estados multi-qubits.

Abstract

The development of second-generation quantum technologies (quantum sensors, quantum computers and quantum internet) has been carried out by large companies and has received state investments in the order of billions of dollars in several countries. Among these technologies, the quantum computer gained greater prominence in the media due to its promise to solve problems much faster and more efficiently than classical computers, which could produce numerous scientific advances with impacts on society and industry. In recent years, companies such as IBM, Google, Microsoft, among others, have made great progress in building the first prototypes of quantum hardware and software. However, these prototypes are devices that operate with few qubits and are heavily affected by noise, which still prevents them from performing truly useful computation. In particular, IBM has provided remote access to some of its machines available to the general public, as well as special access to selected researchers in its research program. Using this resource, in this work we study a method for reconstructing pure states of a multi-qubit system in a quantum computer. In the context of quantum computing, this is a very useful tool, for example, to determine input or output states of a given algorithm, or to characterize the logical operations that constitute the algorithm. The method we study is known as *quantum ptychography*, where a pure state is subjected to a set of projections with some degree of intersection between them, followed by a projective measurement on a single orthonormal basis. The results generated in the process serve as input to an iterative algorithm that estimates the state. Our study was divided between simulations and experiments. The simulations were carried out in ideal scenarios without noise and more realistic scenarios with noise, where the method was successfully demonstrated for a variety of states of up to 6 qubits. In experiments, we obtained good results for reconstructing states of up to 4 qubits; the results for a larger number were strongly affected by noise in the devices. The advantages and limitations of the ptychographic method are discussed in comparison with other state reconstruction methods. In addition to demonstrating quantum ptychography on a quantum computer, we also achieved some advances in the method. The first was a modification to the iterative reconstruction algorithm, which improves its convergence and, consequently, the state estimation. The second was the use of measurement bases that are less affected by noise in a quantum computer. This second advance is still part of a preliminary study but points to promising avenues for future investigations.

Keywords: Quantum ptychography, Pure quantum state tomography, Quantum computing, Multi-qubit states.

Sumário

1 Introdução

No nosso dia a dia, os computadores se tornaram indispensáveis para realizar tarefas cotidianas de forma fácil e rápida. Eles nos possibilitam desvendar novos horizontes em campos como a medicina, clima, logística, dentre outros. Tudo isso aconteceu graças ao advento do transistor, um componente eletrônico cujos dois estados, "carregado" ou "descarregado", servem para codificar um *bit*, a unidade fundamental de informação. Quanto maior o número de transistores em um dispositivo, maior o seu poder de processamento.

Em 1965, Gordon Moore postulou que o número de transistores em um chip dobraria a cada dois anos [1], o que ficou conhecido como lei de Moore. De fato, a evolução dos chips tem corroborado a lei de Moore, pois à medida que fomos reduzindo o tamanho dos transistores conseguimos colocar um número maior deles em um único chip, consequentemente obtendo maior capacidade de processamento. Atualmente, temos dispositivos com bilhões de transistores em um único chip. No entanto, essa redução chega a um limite onde começam a surgir efeitos quânticos, como o tunelamento, e então a física quântica deve ser levada em consideração. E agora, estaremos fadados a construir chips cada vez maiores para acomodar um número maior de transistores? Além da questão do tamanho do chip, existem vários problemas de interesse prático que levariam um tempo inviável (milhares ou milhões de anos) de processamento em um computador clássico ou que são de impossível resolução. Essas limitações da computação clássica, abriram um caminho natural para um novo paradigma, a computação quântica [2]. Agora, ao invés de transistores, temos sistemas quânticos, que dão origem aos denominados *qubits*. Ao trabalharmos com qubits e explorarmos as suas propriedades quânticas sem análogo na física clássica, podemos, em princípio, contornar as limitações dos computadores clássicos e produzir grandes avanços científicos e tecnológicos. Por exemplo, o computador quântico tornaria possível a simulação de estruturas moleculares complexas, que possibilitaria a descoberta de novos materiais ou novos medicamentos. Além disso, devido à sua capacidade de realizar alguns cálculos exponencialmente mais rápido do que qualquer computador clássico, ele teria forte impacto sobre a segurança cibernética,[1] um tema de interesse bastante atual.

Atualmente, além dos computadores quânticos, também temos pesquisa e desenvolvimento de sensores quânticos buscando medidas mais precisas, redes de comunicação e criptografia quântica, que buscam uma comunicação mais segura. Estas são as chamadas tecnologias quânticas de segunda geração.[2] Diversas iniciativas, públicas e privadas, ao

[1] Atualmente, a segurança da criptografia é baseada na complexidade do problema matemático que teria que ser resolvido para contorná-la, por exemplo, a fatoração de números primos na criptografia RSA. Em um computador clássico, este seria um problema intratável.

[2] As chamadas tecnologias quânticas de primeira geração incluem, por exemplo, o laser, a ressonância magnética nuclear e dispositivos semicondutores.

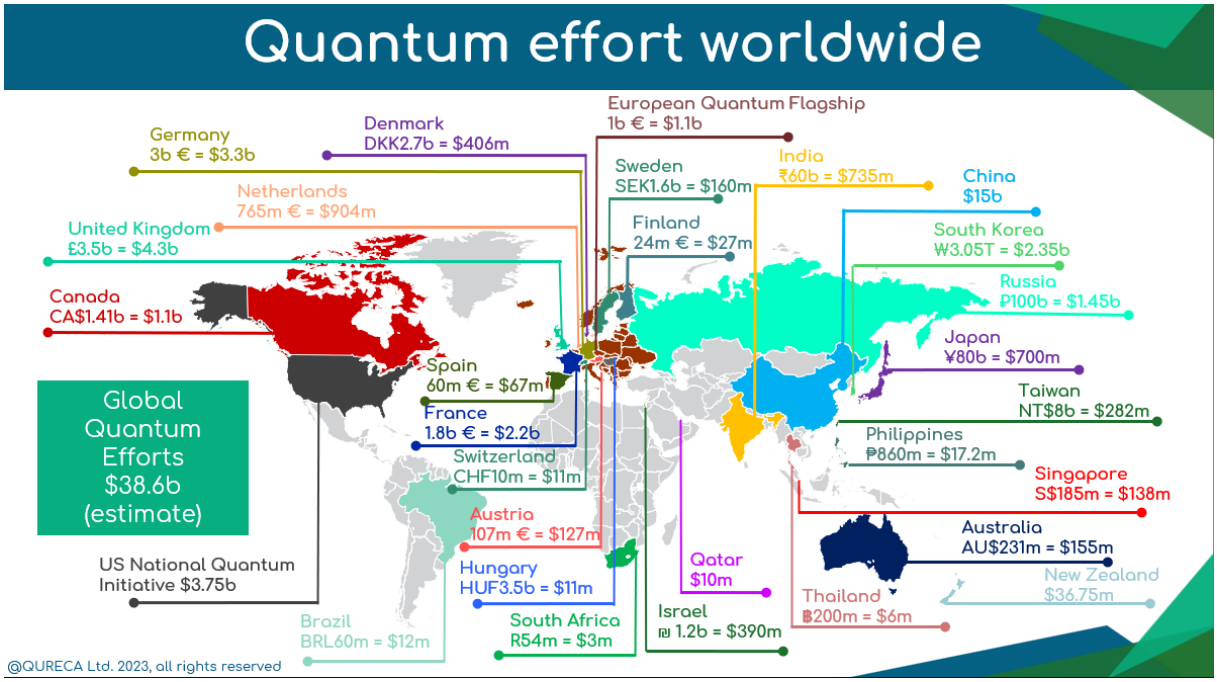


Figura 1 – Visão geral das iniciativas quânticas ao redor do mundo em 2023. Figura retirada de: https://qureca.com/overview-of-quantum-initiatives-worldwide-2023/.

redor do mundo estão buscando desenvolver tais tecnologias: a figura 1 destaca os investimentos por país em 2023, que podem chegar a alguns bilhões de dólares em alguns deles. Dentre essas tecnologias quânticas, o computador quântico tem ganhado destaque maior nos últimos anos, não só pelos potenciais avanços que ele pode trazer com impactos na sociedade e indústria, mas por conta de alguns trabalhos que tiveram bastante repercussão ao serem divulgados [3, 4]. Nesses trabalhos, demonstrou-se experimentalmente a chamada "vantagem quântica", através de uma tarefa específica que o computador quântico realizava exponencialmente mais rápido que qualquer algoritmo clássico.

O destaque para os computadores quânticos se deve também ao grande investimento que grandes empresas de tecnologia como Google, Microsoft, Intel, IBM, entre outras, estão fazendo para o seu desenvolvimento e até mesmo sua popularização. Nos últimos anos, grandes progressos na construção dos primeiros protótipos de hardware e software quânticos foram alcançados. No entanto, esses protótipos são dispositivos que operam com poucos qubits e atingem níveis de ruído que levam à perda das propriedades quânticas [2, 5]. Por esta razão, é dito que atualmente vivemos a chamada era NISQ da computação quântica, definida no quadro abaixo:

A era NISQ

O estágio atual da computação quântica é conhecido como a era da *computação quântica ruidosa de escala intermediária*. Esta é a tradução da expressão em inglês *noisy intermediate scale quantum (NISQ) computing*, criada em 2018 pelo físico John Preskill para descrever os dispositivos quânticos atuais. Os chamados dispositivos NISQ estão sujeitos a altas taxas de erro e possuem um número limitado de qubits, portanto, ainda são incapazes de realizar computação de uma forma geral.

A figura 2, mostra uma pesquisadora da IBM montando um computador quântico da empresa (vamos descrever este computador com mais detalhes no capítulo 3). Note que todo o aparato representa um dispositivo quântico de poucos qubits. Por exemplo, o lado esquerdo da figura mostra um chip de 7 qubits que fica na ponta do "abajur" dourado. Devido ao alcance limitado para um público maior à dispositivos desse tipo, gigantes da tecnologia e *startups* vêm criando e disponibilizando serviço em nuvem para que empresas, acadêmicos, dentre outros, possam acessar máquinas de maneira remota. Exemplos de serviços em nuvem para computação quântica são os disponibilizados pela Amazon,[3] Microsoft[4] e IBM.[5] Em geral, tais serviços são pagos, porém a IBM disponibiliza gratuitamente algumas de suas máquinas. Essas máquinas de livre acesso, apesar de serem de poucos qubits são excelentes para iniciar os estudos (tanto para acadêmicos quanto para entusiastas), bem como realizar pesquisa de ponta.

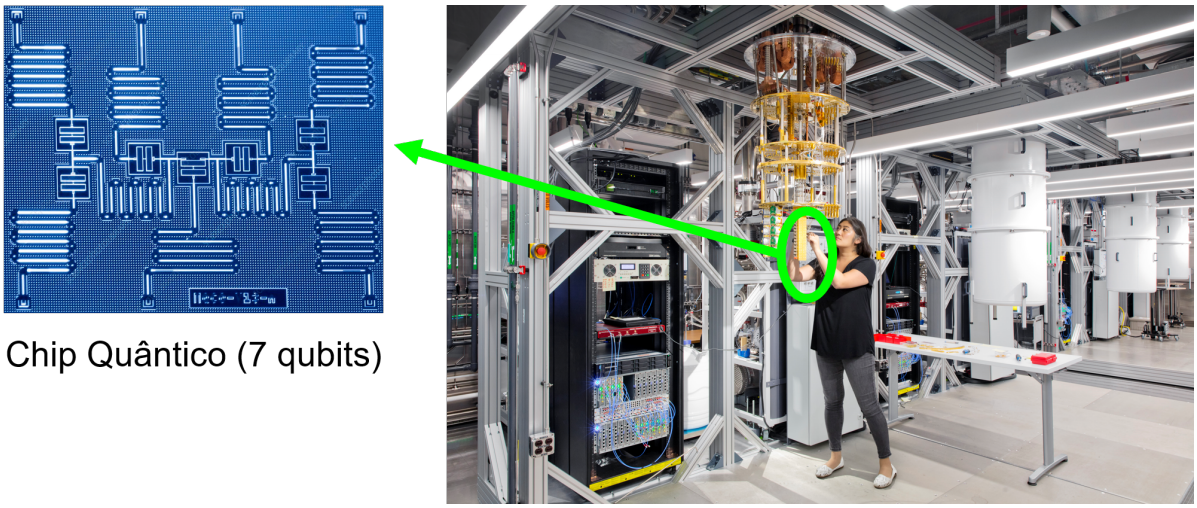


Figura 2 – Pesquisadora trabalhando em um computador quântico supercondutor da IBM. A figura ao lado ilustra um chip de 7 qubits que fica na ponta do aparato. Figura adaptada de: https://newsroom.ibm.com/media-quantum-innovation?keywords=quantum&l=100.

[3] Amazon Braket: https://aws.amazon.com/pt/braket/
[4] Azure Quantum: https://azure.microsoft.com/pt-br/products/quantum/#overview
[5] IBM Quantum: https://quantum-computing.ibm.com/

Usando este recurso da IBM, neste trabalho nós estudamos um método para reconstrução de estados puros de um sistema de múltiplos qubits em um computador quântico. No contexto da computação quântica, a reconstrução de estados é uma ferramenta muito útil. Ela nos permite, por exemplo, determinar os estados de entrada ou de saída de um dado algoritmo, ou até mesmo caracterizar as operações lógicas que constituem o algoritmo. O método que estudamos é conhecido como *pticografia quântica* [6], onde um estado puro é submetido a um conjunto de projeções com algum grau de interseção entre si, seguidas de uma medição projetiva em uma única base ortonormal. Os dados gerados no processo servem como entrada para um algoritmo iterativo que estimará o estado. O nosso estudo se dividiu entre simulações e experimentos. As simulações foram realizadas em cenários ideais sem ruído e cenários mais realísticos com ruído; os experimentos foram realizados remotamente nos dispositivos da IBM. Os resultados obtidos mostram que a pticografia quântica pode ser uma ótima ferramenta para reconstrução de estados em um computador quântico, especialmente quando estes dispositivos forem menos ruidosos.

Para apresentar os resultados desse estudo, dividimos a dissertação da seguinte forma: no capítulo 2, apresentamos os elementos básicos do modelo de circuitos da computação quântica, que é o modelo adotado nos computadores da IBM. No capítulo 3, descrevemos as características e limitações dos dispositivos da IBM que permitirão ao usuário construir os circuitos quânticos para um dado algoritmo; além disso, discutimos brevemente a implementação física desses dispositivos baseada em qubits supercondutores. No capítulo 4, descrevemos de forma geral o processo de reconstrução de estados quânticos, e apresentamos dois métodos que serão posteriormente comparados ao nosso. O capítulo 5 é dedicado a uma revisão do método pticográfico para reconstrução de estados puros. Nos capítulos 6 e 7, apresentamos os resultados obtidos através das simulações e experimentos, respectivamente. Por fim, no capítulo 8, consolidamos os achados deste estudo e apresentamos nossas conclusões.

2 Qubits, portas lógicas e circuitos quânticos

Os simuladores e computadores quânticos da IBM, onde realizamos as simulações e os experimentos deste trabalho, utilizam o modelo de circuitos da computação quântica. Neste modelo, um algoritmo é implementado através de transformações unitárias e medições sobre um dado estado quântico de entrada. Supondo que o leitor esteja familiarizado com estes conceitos da mecânica quântica, neste capítulo, nós vamos introduzir os elementos básicos do modelo quântico de circuitos. Os não familiarizados podem consultar [2, 7, 8].

2.1 Bit quântico

Nos computadores clássicos a unidade básica de informação é o bit, que é uma variável que pode assumir o valor 0 ou 1. Por exemplo, em um transistor, a ausência de corrente representa o estado lógico 0, enquanto a presença de corrente representa o estado lógico 1 [9]. Já nos computadores quânticos, ao invés do bit, temos o *quantum bit* (bit quântico) ou qubit, que representa um sistema quântico de dois níveis [2]. Dentre as possibilidades para esse tipo de sistema, temos os qubits supercondutores [10–12], íons aprisionados [13, 14], sistemas fotônicos [15, 16], dentre outras. O "zero" como bit quântico é representado pelo vetor $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ e o "um" por $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. O mais interessante do bit quântico é o fato de $|0\rangle$ e $|1\rangle$ poderem ser escritos como uma combinação linear da forma

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.1}$$

onde α e β são números complexos complexos que satisfazem $|\alpha|^2 + |\beta|^2 = 1$. Essa combinação linear significa que os estados $|0\rangle$ e $|1\rangle$ podem podem estar em superposição, algo impossível para bits clássicos [2]. A superposição é uma das propriedades que estabelecem as vantagens computacionais de um computador quântico sobre o clássico.[1]

A equação (2.1) pode ser reescrita em coordenadas esféricas [2]

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle, \tag{2.2}$$

onde $\theta \in [0, \pi]$ e $\phi \in [0, 2\pi]$. O estado do qubit pode ser representado em uma esfera de raio unitário denominada esfera de Bloch [2, 19]. Esta esfera fornece uma maneira visual

[1] O primeiro a mostrar isto foi o físico David Deutsch que propôs o seguinte problema, conhecido como algoritmo de Deutsch [17, 18]: dada uma função de um bit, $f : \{0, 1\} \to \{0, 1\}$, determine se ela é constante ($f(0) = f(1)$) ou balanceada ($f(0) \neq f(1)$). Enquanto um algoritmo clássico demanda o cálculo de $f(0)$ e $f(1)$ (duas rodadas), mostrou-se que o algoritmo quântico, aproveitando-se da superposição, pode avaliar $f(0)$ e $f(1)$ simultaneamente, determinando uma propriedade global da função em uma única rodada.

na qual o estado é um vetor de norma 1 que vai do centro à superfície, como mostrado na figura 3.

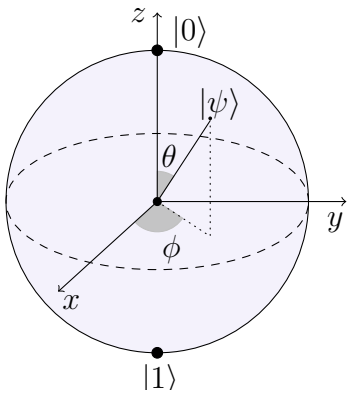


Figura 3 – Esfera de Bloch: representação do estado $|\psi\rangle$ de um qubit dado pela equação (2.2). Os estados $|0\rangle$ e $|1\rangle$ nos polos da esfera formam a base computacional.

2.2 Sistemas de múltiplos qubits e emaranhamento quântico

Na seção anterior foi visto como o estado de um qubit pode ser descrito, mas e para mais qubits? No caso de um qubit, os estados $\{|0\rangle, |1\rangle\}$ formam uma base ortonormal, chamada de base computacional. Para um sistema de n qubits ($n \geq 2$), a base computacional é formada pelo produto tensorial da base computacional de cada subsistema, ou seja, $\{|0\rangle, |1\rangle\}^{\otimes n} = \{|000...0\rangle, |000...1\rangle, ..., |111...1\rangle\}$. Esta é a representação binária da base computacional que pode também ser escrita de forma mais compacta usando a representação decimal de um número binário,[2] isto é, $\{|0\rangle, |1\rangle, ..., |2^n - 1\rangle\}$. Desta forma, um estado de n qubits pode ser escrito como

$$\begin{aligned} |\psi\rangle &= \alpha_0|000...0\rangle + \alpha_1|000...1\rangle + \cdots + \alpha_{2^n-1}|111...1\rangle \\ &= \sum_{j=0}^{2^n-1} \alpha_j|j\rangle, \end{aligned} \tag{2.3}$$

onde $\sum_i |\alpha_i|^2 = 1$.

Uma consequência do princípio da superposição é que estados de sistemas quânticos compostos, como são os multi-qubits, podem ser emaranhados, caracterizando uma forte correlação entre os subsistemas. O emaranhamento é um fenômeno quântico e os estados emaranhados podem ser definidos da seguinte forma: dado um sistema de n partes (rotuladas por $0, 1, \ldots n-1$), o seu estado $|\psi\rangle$ será emaranhado se não for possível escrevê-lo como o produto dos estados de cada parte, ou seja, se

$$|\psi\rangle \neq |\psi_0\rangle \otimes |\psi_1\rangle \otimes \ldots \otimes |\psi_{n-1}\rangle; \tag{2.4}$$

[2] Um número binário de n bits $[b_{n-1}b_{n-2}\ldots b_1b_0]_2$ ($b_j = 0, 1$) corresponde a um número inteiro N na base 10 dado por $N = \sum_{j=1}^{n} b_{n-j}2^{n-j}$, onde é definido no intervalo $N = 0, \ldots, 2^n - 1$.

em caso de igualdade o estado é dito separável.

Para dois qubits, alguns exemplos de estados separáveis são os estados da base computacional $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, $|\psi\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes |0\rangle$, etc. Para estados emaranhados de dois qubits temos como exemplo os estados de Bell (em homenagem a John Stewart Bell por suas importantes descobertas sobre emaranhamento [20]):

$$|\psi_{\pm}\rangle = \frac{|00\rangle \pm |11\rangle}{\sqrt{2}}, \tag{2.5}$$

$$|\phi_{\pm}\rangle = \frac{|01\rangle \pm |10\rangle}{\sqrt{2}}. \tag{2.6}$$

Os estados de Bell estão presentes em vários protocolos de comunicação quântica como codificação superdensa [21], teleporte [22] e troca de emaranhamento [23].

Estados emaranhados de 3 qubits muito conhecidos na literatura são o estado $|GHZ\rangle$ [24] (em alusão aos pesquisadores Greenberger, Horne e Zeilinger que o estudaram pela primeira vez) e o estado[3] $|W\rangle$ [25, 27], cujas expressões são

$$|GHZ\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}, \tag{2.7}$$

$$|W\rangle = \frac{1}{\sqrt{3}} \left(|100\rangle + |010\rangle + |001\rangle\right). \tag{2.8}$$

A generalização para n qubits é

$$|GHZ\rangle = \frac{|0\rangle^{\otimes n} + |1\rangle^{\otimes n}}{\sqrt{2}}, \tag{2.9}$$

$$|W\rangle = \frac{1}{\sqrt{n}} \left(|100\ldots0\rangle + |010\ldots0\rangle + \cdots + |000\ldots1\rangle\right). \tag{2.10}$$

Os estados GHZ e W pertencem a duas classes de estados emaranhados, uma vez que que um não leva a outro por operações locais e comunicação clássica [25]. Assim como os estados de Bell, ambos possuem aplicações em computação e comunicação quântica [28–30].

2.3 Portas lógicas quânticas

Agora, a questão que deve ser respondida é como realizar operações sobre um ou mais qubits? Uma das formas de fazer isso é aplicando um conjunto de operações unitárias definidas em relação à base computacional denominadas portas lógicas quânticas.

2.3.1 Portas de um qubit

As portas lógicas quânticas que operam sobre um único qubit são matrizes unitárias 2×2. Primeiramente, vamos apresentar as portas de Pauli, oriundas das conhecidas matrizes

[3] Existem informações conflitantes sobre a origem do nome. Algumas fontes atribuem a Wolfgang Dür, um dos três autores de [25]; outras, a William K. Wootters, um dos três autores de [26], onde o estado $|W\rangle$ aparece pela primeira vez.

de Pauli e que formam um conjunto de operações importantes para a computação quântica. Comecemos pela porta **X** que é dada por

$$\mathbf{X} = |1\rangle\langle 0| + |0\rangle\langle 1| = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{2.11}$$

cuja atuação na base computacional será

$$\mathbf{X}|0\rangle = |1\rangle, \quad \mathbf{X}|1\rangle = |0\rangle. \tag{2.12}$$

Pelo fato de inverter o estado nesta base, a porta **X** é também conhecida como porta **NOT** quântica (em analogia à porta clássica **NOT** que inverte um bit[4]). A próxima porta de Pauli é a porta **Z** e é dada por

$$\mathbf{Z} = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.13}$$

e sua atuação na base computacional produz

$$\mathbf{Z}|0\rangle = |0\rangle, \quad \mathbf{Z}|1\rangle = -|1\rangle, \tag{2.14}$$

ou seja, ela introduz uma fase de π no estado $|1\rangle$. A terceira e última porta de Pauli é a porta **Y** dada por

$$\mathbf{Y} = i|1\rangle\langle 0| - i|0\rangle\langle 1| = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \tag{2.15}$$

que realiza tanto a inversão do estado na base computacional quanto adiciona fase:

$$\mathbf{Y}|0\rangle = -i|1\rangle, \quad \mathbf{Y}|1\rangle = i|0\rangle. \tag{2.16}$$

As portas de Pauli dão origem às portas de rotação definidas como

$$\mathbf{R}_x(\theta) \equiv e^{-i\theta\mathbf{X}/2} = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \tag{2.17}$$

$$\mathbf{R}_y(\theta) \equiv e^{-i\theta\mathbf{Y}/2} = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \tag{2.18}$$

$$\mathbf{R}_z(\theta) \equiv e^{-i\theta\mathbf{Z}/2} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{2.19}$$

O efeito de uma dada porta $\mathbf{R}_j(\theta)$ é rotacionar o estado de um ângulo θ em torno do eixo j ($j = x, y, z$) da esfera Bloch.

[4] Esta analogia só é válida em relação à base computacional. Uma porta **NOT** quântica universal deveria atuar sobre um estado de entrada qualquer e transformá-lo em um estado ortogonal. Suponha que ela exista e seja denotada por $\mathbf{U}_{\text{NOT}}$. Neste caso, $\mathbf{U}_{\text{NOT}}|0\rangle = |1\rangle$ e $\mathbf{U}_{\text{NOT}}|1\rangle = |0\rangle$. Para os estados ortogonais $|\pm\rangle \equiv \frac{|0\rangle \pm |1\rangle}{\sqrt{2}}$ teríamos $\mathbf{U}_{\text{NOT}}|\pm\rangle = \pm|\pm\rangle \neq |\mp\rangle$. Portanto, a linearidade da mecânica quântica proíbe a existência de $\mathbf{U}_{\text{NOT}}$.

A porta Hadamard, denotada por $\mathbf{H}$, é dada por

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.20}$$

Sua atuação na base computacional produz uma superposição uniforme dos seus elementos, isto é

$$\begin{aligned} \mathbf{H}|0\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \equiv |+\rangle, \\ \mathbf{H}|1\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}} \equiv |-\rangle. \end{aligned} \tag{2.21}$$

Por esta atuação, a porta $\mathbf{H}$ não possui análogo clássico. Observe que os estados $\{|+\rangle, |-\rangle\}$ (que são autoestados de $\mathbf{X}$) formam outra base ortonormal no espaço de um qubit, e a atuação de $\mathbf{H}$ sobre eles desfaz a superposição:

$$\begin{aligned} \mathbf{H}|+\rangle &= |0\rangle, \\ \mathbf{H}|-\rangle &= |1\rangle. \end{aligned} \tag{2.22}$$

As equações (2.21) e (2.22) mostram que a porta Hadamard transforma uma base na outra.

Duas outras portas interessantes de um qubit são as portas $\mathbf{S}$ e $\mathbf{T}$ cujas matrizes são dadas, respectivamente, por

$$\mathbf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \tag{2.23}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}. \tag{2.24}$$

Diferentemente das portas de Pauli e Hadamard, $\mathbf{S}$ e $\mathbf{T}$ não são Hermitianas, portanto, suas inversas são $\mathbf{S}^\dagger$ e $\mathbf{T}^\dagger$, respectivamente. Estas duas portas, assim como a $\mathbf{Z}$, se derivam de uma porta mais geral, chamada de porta de fase, dada por

$$\mathbf{P}(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix}; \tag{2.25}$$

seu efeito é introduzir uma fase $e^{i\lambda}$ ao estado $|1\rangle$ da base computacional. Facilmente podemos ver que $\mathbf{P}(\lambda = \pi) = \mathbf{Z}$, $\mathbf{P}(\lambda = \pi/2) = \mathbf{S}$ e $\mathbf{P}(\lambda = \pi/4) = \mathbf{T}$.

Finalizando esta discussão, sabe-se que uma operação unitária arbitrária sobre o estado de um qubit pode ser decomposta em uma sequência de três rotações como [2] (capítulo 4, página 176 do livro)

$$\mathbf{U}(\theta, \phi, \lambda) = e^{i(\phi+\lambda)/2} \mathbf{R}_z(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\lambda) = \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\phi+\lambda)}\cos\frac{\theta}{2} \end{bmatrix}, \tag{2.26}$$

onde $\theta \in [0, \pi]$, ϕ e $\lambda \in [0, 2\pi]$. Portanto, desta operação se derivam todas as portas de um qubit, por exemplo, $\mathbf{X} = \mathbf{U}(\pi, 0, \pi)$, $\mathbf{Y} = \mathbf{U}(\pi, \pi/2, \pi/2)$, $\mathbf{Z} = \mathbf{U}(0, \pi, 0)$, $\mathbf{H} = \mathbf{U}(\pi/2, 0, \pi)$ e $\mathbf{P} = \mathbf{U}(0, 0, \lambda)$.

2.3.2 Portas de dois qubits

No modelo de circuitos da computação quântica, uma maneira simples de se implementar portas de dois (ou mais) qubits é através das chamadas portas controladas. Nestas portas, um dos qubits é o controle e o seu estado determinará se uma operação unitária (porta de um qubit) será aplicada ou não ao outro qubit que é o alvo. Por exemplo, tomando o primeiro qubit como controle e o segundo como alvo, considere uma porta $\mathbf{U}$ controlada, $\mathbf{U}^{\text{ctrl}}$, cuja matriz é dada por

$$\mathbf{U}^{\text{ctrl}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{00} & U_{01} \\ 0 & 0 & U_{10} & U_{11} \end{bmatrix}, \tag{2.27}$$

onde U_{ij} são os elementos de uma porta de um qubit arbitrária, $\mathbf{U}$. A ação de $\mathbf{U}^{\text{ctrl}}$ na base computacional resulta em

$$\mathbf{U}^{\text{ctrl}}(|0\rangle_c|0\rangle_a) = |00\rangle, \tag{2.28a}$$
$$\mathbf{U}^{\text{ctrl}}(|0\rangle_c|1\rangle_a) = |01\rangle, \tag{2.28b}$$
$$\mathbf{U}^{\text{ctrl}}(|1\rangle_c|0\rangle_a) = |1\rangle(U_{00}|0\rangle + U_{10}|1\rangle), \tag{2.28c}$$
$$\mathbf{U}^{\text{ctrl}}(|1\rangle_c|1\rangle_a) = |1\rangle(U_{01}|0\rangle + U_{11}|1\rangle). \tag{2.28d}$$

Neste caso, se o estado do qubit controle for $|1\rangle$, a operação $\mathbf{U}$ atua sobre o alvo, caso contrário, a operação não é aplicada.

Para $\mathbf{U} = \mathbf{X}$ (equação (2.11)), temos a porta **NOT** controlada ou **CNOT**. Sua forma matricial é

$$\mathbf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.29}$$

e sua ação na base computacional produz

$$\mathbf{CNOT}(|0\rangle_c|0\rangle_a) = |00\rangle, \tag{2.30a}$$
$$\mathbf{CNOT}(|0\rangle_c|1\rangle_a) = |01\rangle, \tag{2.30b}$$
$$\mathbf{CNOT}(|1\rangle_c|0\rangle_a) = |11\rangle, \tag{2.30c}$$
$$\mathbf{CNOT}(|1\rangle_c|1\rangle_a) = |10\rangle. \tag{2.30d}$$

Como se observa, ela inverte o estado do qubit alvo quando o estado do controle é $|1\rangle$.

Para $\mathbf{U} = \mathbf{P}(\lambda)$ (equação (2.25)), temos a porta de fase controlada $\mathbf{P}^{\text{ctrl}}(\lambda)$ dada por

$$\mathbf{P}^{\text{ctrl}}(\lambda) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\lambda} \end{bmatrix}, \tag{2.31}$$

e que atua na base computacional como

$$\mathbf{P}^{\text{ctrl}}(|0\rangle_c|0\rangle_a) = |00\rangle, \tag{2.32a}$$
$$\mathbf{P}^{\text{ctrl}}(|0\rangle_c|1\rangle_a) = |01\rangle, \tag{2.32b}$$
$$\mathbf{P}^{\text{ctrl}}(|1\rangle_c|0\rangle_a) = |10\rangle, \tag{2.32c}$$
$$\mathbf{P}^{\text{ctrl}}(|1\rangle_c|1\rangle_a) = e^{i\lambda}|11\rangle. \tag{2.32d}$$

Em particular, quando $\lambda = \pi$ temos a conhecida porta $\mathbf{Z}$ controlada; quando $\lambda = 2\pi/2^k$ $(k = 0, \ldots, n)$, temos as portas controladas $\mathbf{P}_k^{\text{ctrl}}$ que são essenciais para a implementação da transformada quântica de Fourier, como será visto no capítulo 5. Como pode ser visto das equações (2.32), a porta de fase controlada, ao contrário da **CNOT**, é simétrica em relação aos papéis dos qubits controle e alvo.

Por fim, outra porta de dois qubits de interesse aos nossos propósitos é a **SWAP**, que executa uma operação de permutação dos estados de entrada, ou seja

$$\mathbf{SWAP}(|\psi\rangle|\phi\rangle) = |\phi\rangle|\psi\rangle, \tag{2.33}$$

onde $|\psi\rangle$ e $|\phi\rangle$ são estados arbitrários de um qubit. Para isto, é fácil verificar que basta uma operação que execute as seguintes permutações na base computacional: $|01\rangle \longrightarrow |10\rangle$ e $|10\rangle \longrightarrow |01\rangle$. Assim, a forma matricial da porta **SWAP** será dada por

$$\mathbf{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.34}$$

Esta porta também é parte da transformada quântica de Fourier, como veremos no capítulo 5, e executa um papel importante ao permitir a interação entre qubits de um computador quântico que não estejam diretamente conectados, como será visto no capítulo 7.

2.3.3 Portas quânticas universais

Além das portas de um e dois qubits, existem também portas quânticas de três qubits como a **CNOT** e a **SWAP** controladas [2]. Porém, elas não serão discutidas aqui por não serem necessárias no nosso estudo. Além disso, as portas de um qubit **H**, **S** e

T juntamente com a porta de dois qubits **CNOT**, definidas anteriormente, formam um conjunto universal para computação quântica. Isto significa que dada uma operação unitária arbitrária sobre um número qualquer de qubits, é possível construir uma combinação de portas deste conjunto que aproxima esta unitária com precisão arbitrária. A demonstração de universalidade deste conjunto pode ser vista em [2] (capítulo 4, seção 4.5).

2.4 Circuitos quânticos

Uma maneira de representar algoritmos em um computador quântico é através de diagramas chamados circuitos quânticos. Nesses circuitos, os qubits são representados por linhas horizontais simples e as portas quânticas de um qubit por retângulos sobre as linhas com o símbolo correspondente da operação unitária que implementa. As portas controladas de dois qubits são representadas por uma linha vertical conectando a linha do qubit controle à porta sobre o qubit alvo. Nesses circuitos, a medição é representada por uma caixa com um ícone de medidor sobre a linha do qubit que será medido. Uma linha dupla saindo da caixa de medição representa o bit de informação clássica gerado no processo. O circuito é "lido" da esquerda para a direita, indicando a evolução temporal do sistema.

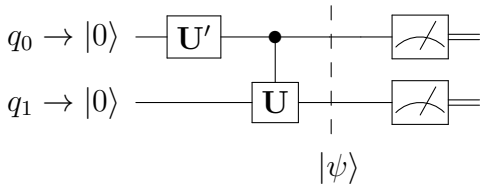


Figura 4 – Circuito genérico para dois qubits.

A figura 4 ilustra um circuito genérico para dois qubits, q_0 e q_1, contendo todos os elementos descritos acima. Neste circuito, consideramos o estado inicial dos dois qubits como $|00\rangle$. Após a implementação das portas, o estado é transformado em

$$|\psi\rangle = \mathbf{U}_{0,1}^{\text{ctrl}}(\mathbf{U}' \otimes \mathbf{I})|00\rangle, \tag{2.35}$$

onde $\mathbf{I}$ representa a operação identidade sobre q_1 e os subíndices de $\mathbf{U}_{0,1}^{\text{ctrl}}$ indicam o qubit controle (q_0) e alvo (q_1) desta operação.

2.4.1 Portas de um e dois qubits

As figuras 5, 6 e 7 ilustram a representação em circuitos quânticos das portas de um e dois qubits apresentadas na seção anterior. Na figura 6, o símbolo $\oplus$ que caracteriza a **CNOT** denota adição de módulo dois, pois $\mathbf{CNOT}|i\rangle_c|j\rangle_a = |i\rangle_c|j \oplus i\rangle_a$ $(i, j = 0, 1)$. Para a porta de fase controlada, $\mathbf{P}^{\text{ctrl}}(\lambda)$, a representação à direita simboliza sua simetria em relação aos papéis dos qubits controle e alvo. Na figura 7, observa-se que a porta **SWAP**

pode ser decomposta em três **CNOT**s com papéis alternados dos qubits controle e alvo, o que é fácil de ser verificado.

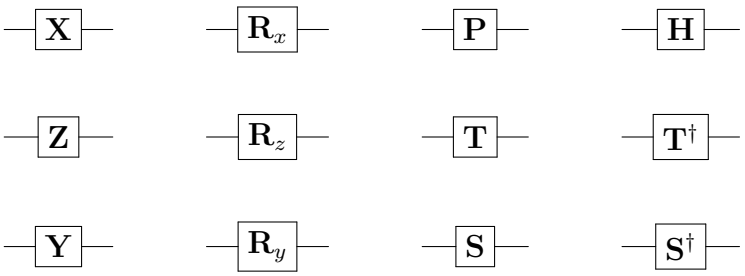


Figura 5 – Portas lógicas de um qubit.

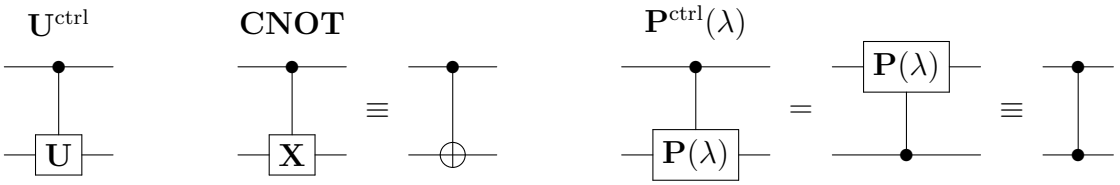


Figura 6 – Portas controladas de dois qubits.

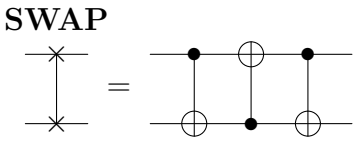


Figura 7 – Porta **SWAP**: esta operação equivale a uma sequência de três **CNOT**s.

Decomposição ABC

Um resultado importante sobre portas de dois qubits é que uma operação controlada arbitrária, $\mathbf{U}^{\text{ctrl}}$, pode ser implementada através de uma combinação de duas portas **CNOT** e até quatro portas de um qubit, com a configuração mostrada na figura 8. A validade desta construção está no fato que uma matriz unitária arbitrária sobre um qubit pode ser decomposta como $\mathbf{U} = e^{i\lambda}\mathbf{AXBXC}$ (decomposição ABC), onde **A**, **B** e **C** são unitárias de um qubit que satisfazem $\mathbf{ABC} = \mathbf{I}$ [2]. É fácil ver que se o qubit controle (q_0) estiver no estado $|1\rangle$ a operação $e^{i\lambda}\mathbf{AXBXC} = \mathbf{U}$ é aplicada ao qubit alvo (q_1). Caso contrário, a operação sobre o alvo é $\mathbf{ABC} = \mathbf{I}$, que não altera seu estado.

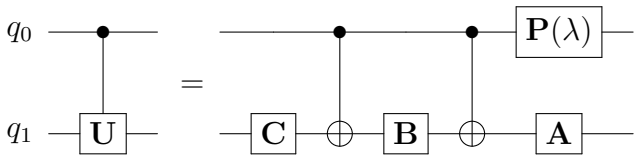


Figura 8 – Decomposição ABC de uma porta $\mathbf{U}^{\text{ctrl}}$ arbitrária; $\mathbf{ABC} = \mathbf{I}$.

Computadores quânticos como os da IBM possuem apenas um conjunto restrito de portas quânticas disponíveis que podem ser implementadas fisicamente. Para se implementar uma porta controlada fora deste conjunto, pode ser necessário usar este tipo de decomposição. O mesmo pode ocorrer para portas de um qubit que não estejam disponíveis no conjunto. Vamos ver essas questões com mais detalhes no próximo capítulo.

2.4.2 Circuitos de preparação de estados

Vamos mostrar agora circuitos para preparação de estados multi-qubits, em particular para os estados emaranhados discutidos na seção 2.2. Estes são alguns dos estados que serão caracterizados pelo método pticográfico (capítulos 6 e 7). Em todos os casos, consideramos o estado de entrada como $|0\rangle^{\otimes n}$. Com a aplicação das portas apresentadas na seção 2.3 nesta entrada, é fácil obter o estado de saída.

A figura 9 mostra os circuitos para a preparação dos estados de Bell e a figura 10 os circuitos para a preparação dos estados GHZ e W de três qubits.

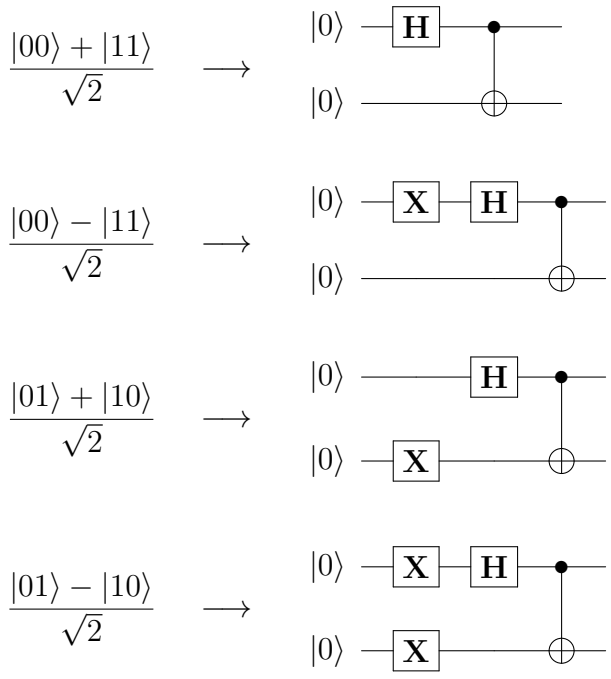


Figura 9 – Circuitos de preparação dos estados de Bell.

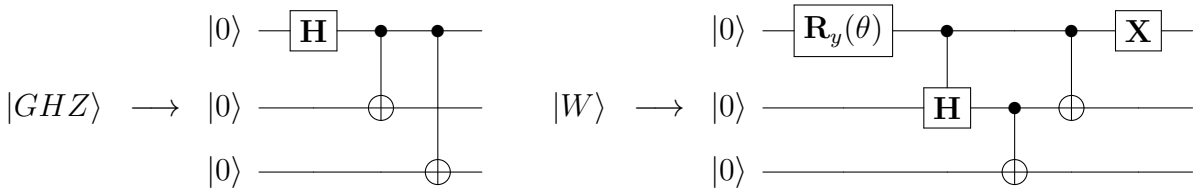


Figura 10 – Circuitos de preparação dos estados GHZ e W de 3 qubits. O ângulo de rotação da porta $\mathbf{R}_y$ é $\theta = 2\arccos(1/\sqrt{3})$.

2.4.3 Medições

A medição é um elemento fundamental da computação quântica. Ela pode ser realizada em etapas intermediárias de um circuito quântico[5] e, principalmente, ao final do circuito para se extrair a informação processada no algoritmo, a qual será armazenada em bits clássicos.

Medições sobre um qubit

Em um computador quântico, a medição implementada sobre o qubit é uma medição projetiva, geralmente, na base computacional $\{|0\rangle, |1\rangle\}$. Neste caso, o observável associado é dado pelo operador de Pauli $\mathbf{Z} = |0\rangle\langle 0| - |1\rangle\langle 1|$ e associamos o bit 0 (1) gerado à projeção em $|0\rangle$ ($|1\rangle$). Portanto, dado um estado $|\psi\rangle$ arbitrário de um qubit, sua medição no circuito produzirá o resultado "0" com probabilidade $|\langle 0|\psi\rangle|^2$ ou "1" com probabilidade $|\langle 1|\psi\rangle|^2$. Caso o qubit não seja destruído pela medição, seu estado será projetado em $|0\rangle$ ou $|1\rangle$, respectivamente.[6] A figura 11(a) ilustra este processo.

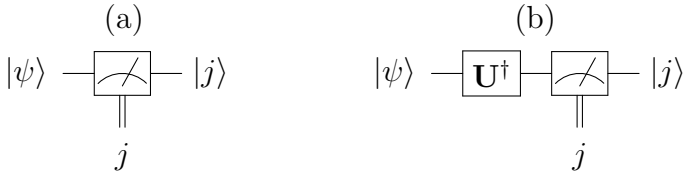


Figura 11 – Medição projetiva: (a) base computacional e (b) base arbitrária; $j = 0, 1$.

É possível também realizar medições em outras bases da seguinte forma: seja $\{|u_0\rangle = \mathbf{U}|0\rangle, |u_1\rangle = \mathbf{U}|1\rangle\}$ uma base ortonormal no espaço de um qubit gerada por uma dada operação unitária $\mathbf{U}$ sobre a base computacional. As probabilidades de projeção de um estado $|\psi\rangle$ nesta base serão $p_j = |\langle u_j|\psi\rangle|^2 = |\langle j|\mathbf{U}^\dagger|\psi\rangle|^2$ $(j = 0, 1)$. Portanto, medir o estado $|\psi\rangle$ nesta nova base é equivalente a medir o estado $\mathbf{U}^\dagger|\psi\rangle$ na base computacional. O circuito quântico correspondente para a medição nesta base é ilustrado na figura 11(b). Observe que o estado do qubit após a medição será projetado em um dos estados da base computacional.

Na reconstrução pticográfica de estados multi-qubits nós vamos executar medições intermediárias sobre um dos qubits na base computacional e também nas bases dos autoestados de $\mathbf{X}$ e $\mathbf{Y}$. É fácil verificar que no caso de $\mathbf{X}$ teremos $\mathbf{U} = \mathbf{H}$ e no caso de $\mathbf{Y}$, $\mathbf{U} = \mathbf{HS}$. Os circuitos para estas medições projetivas são mostrados na figura 12.

[5] Medições intermediárias permitem implementar, em conjunto com as portas unitárias, transformações não-unitárias sobre o sistema. Sobre transformações não unitárias o leitor pode consultar a seção 8.3 do capítulo 8 de [2].

[6] Nos dispositivos da IBM que vamos utilizar, o qubit não é destruído no processo de medição, o que permite a implementação de medições intermediárias em um circuito [31]. Isto é importante para o protocolo pticográfico de reconstrução de estados multi-qubits estudado neste trabalho, como veremos no capítulo 5.

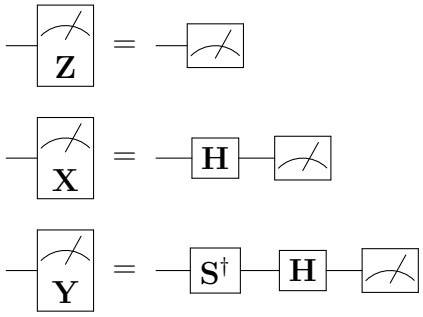


Figura 12 – Medições projetivas de Pauli.

Medições sobre n qubits

As medições em circuitos com n qubits procedem de maneira similar à descrição acima. O circuito para a medição na base computacional deste sistema é mostrado na figura 13(a). Para medições em outras bases, generalizando o caso de um qubit da figura 11(b), temos o circuito mostrado na figura 13(b). Neste circuito, $\mathbf{U}$ é uma operação unitária atuando no espaço de n qubits e gera a base de medição desejada a partir da base computacional. Em sistemas de múltiplos qubits podemos encontrar dois tipos de bases de medição: as fatoráveis e as não-fatoráveis. Bases fatoráveis são aquelas que podem ser decompostas como produto tensorial de n bases de um qubit, como é o caso da base computacional $\{|0\rangle, |1\rangle\}^{\otimes n}$. Elas são caracterizadas por uma unitária que pode ser fatorada como $\mathbf{U} = \mathbf{U}_0 \otimes \mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_{n-1}$, onde $\mathbf{U}_j$ é uma operação unitária de um qubit atuando no espaço do j-ésimo qubit. Caso $\mathbf{U}$ não possa ser decomposta desta maneira, a base será não-fatorável. Por exemplo, considerando um sistema de dois qubits, a operação unitária $\mathbf{CNOT}_{0,1}(\mathbf{H}_0 \otimes \mathbf{I}_1)$ não pode ser decomposta como $\mathbf{U}_0 \otimes \mathbf{U}_1$, portanto, é não-fatorável. A atuação desta operação sobre a base computacional gera uma base ortonormal de estados emaranhados[7] que são os estados de Bell definidos nas equações (2.5) e (2.6). O circuito que executa a medição projetiva nesta base será caracterizado pela unitária $[\mathbf{CNOT}_{0,1}(\mathbf{H}_0 \otimes \mathbf{I}_1)]^\dagger$ e terá a forma mostrada na figura 14.

Portas quânticas controladas por medições

Em um circuito quântico é possível também executar operações sobre um ou mais qubits a partir do resultado clássico de uma medição. São chamadas de portas lógicas quânticas com comunicação clássica. Um exemplo disto, e que também ressalta a importância da medição na base de Bell discutida acima, é o circuito de teleporte quântico mostrado na figura 15. Na primeira parte do circuito (antes da linha tracejada), o estado

[7] Uma base não-fatorável pode conter tanto estados emaranhados (como o exemplo da base de Bell) quanto estados separáveis (por exemplo, a base gerada pela transformada quântica de Fourier, que será discutida no capítulo 5). Por outro lado, bases fatoráveis não contêm estados emaranhados, pois as operações unitárias que geram essas bases são locais e não podem criar emaranhamento [2].

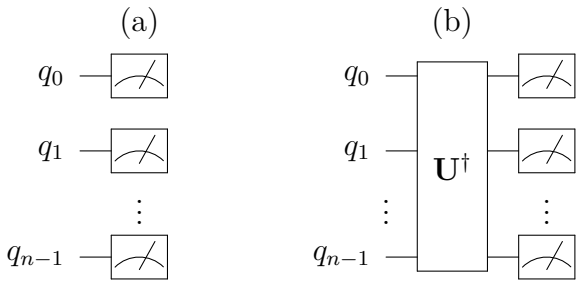


Figura 13 – Medições projetivas para um circuito de n qubits. (a) Medição na base computacional. (b) Medição em uma base arbitrária: se $\mathbf{U} = \mathbf{U}_0 \otimes \mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_{n-1}$, a base é fatorável; caso contrário, a base é não-fatorável.

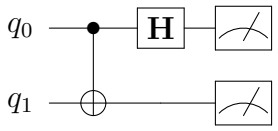


Figura 14 – Medição projetiva na base de Bell.

de Bell $|\psi_+\rangle$ é preparado entre os qubits q_1 e q_2. Na segunda parte, realiza-se uma medição na base de Bell entre q_0 e q_1. Os dois bits de informação $i, j = 0, 1$ gerados no processo (representados pela linha dupla) condicionam a atuação da operação $\mathbf{Z}^j\mathbf{X}^i$ sobre q_2. Ao final, o estado $|\psi\rangle$ do qubit q_0 é transmitido (ou, teleportado) para o qubit q_2.

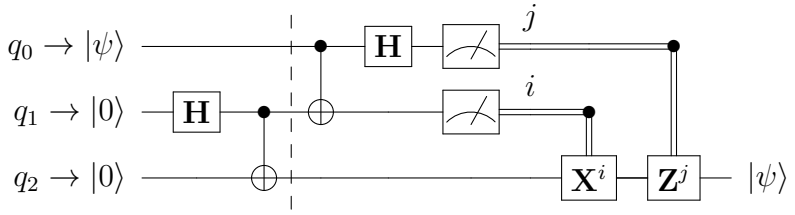


Figura 15 – Circuito do teleporte quântico.

2.5 Propriedades de circuitos quânticos

Existem várias propriedades que podem quantificar o "tamanho" de um circuito quântico e sua capacidade de operar em dispositivos ruidosos. O número de qubits é uma delas e o seu entendimento é imediato. Outras duas propriedades importantes neste contexto são o número de fatores tensoriais e a profundidade do circuito, descritas a seguir.

2.5.1 Número de fatores tensoriais

Seja $\mathbf{U}_{\text{circ}}$ uma operação unitária global implementada por um circuito de n qubits. Se esta operação pode ser decomposta como $\mathbf{U}_{\text{circ}} = \mathbf{U}_1 \otimes \mathbf{U}_2 \otimes \cdots \otimes \mathbf{U}_T$, onde $\{\mathbf{U}_j\}_{j=1}^T$ são unitárias atuando em subespaços do sistema, temos que $1 \leq T \leq n$ é o número de fatores

tensoriais de $\mathbf{U}_{\text{circ}}$. Para $T = n$, o circuito terá apenas portas de um qubit, indicando um circuito mais simples pois não apresenta interação entre os qubits. Para $T = 1$, haverá pelo menos uma porta de dois qubits atuando em cada um dos qubits e isto é um indício de maior complexidade do circuito. Vamos ilustrar esta propriedade utilizando um circuito de 3 qubits. Na figura 16(a) temos que $\mathbf{U}_{\text{circ}} = \mathbf{U}_1 \otimes \mathbf{U}_2 \otimes \mathbf{U}_3 = \mathbf{H} \otimes \mathbf{X} \otimes \mathbf{H}$, portanto, $T = 3$. Em 16(b) temos $T = 2$ pois $\mathbf{U}_{\text{circ}} = \mathbf{U}_1 \otimes \mathbf{U}_2$, onde $\mathbf{U}_1 = \mathbf{CNOT}(\mathbf{H} \otimes \mathbf{X})$ (caixa tracejada) e $\mathbf{U}_2 = \mathbf{SH}$ (caixa sólida). Em 16(c) a operação $\mathbf{U}_{\text{circ}}$ não pode ser fatorada, portanto, $T = 1$.

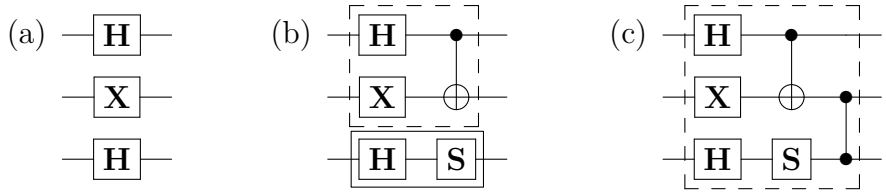


Figura 16 – Circuitos de 3 qubits com (a) $T = 3$, (b) $T = 2$ e (c) $T = 1$.

2.5.2 Profundidade de um circuito

Um circuito quântico pode ser dividido em "camadas", as quais contêm um conjunto de portas que podem ser executadas em paralelo. O número de camadas necessárias para completar sua operação é conhecido como profundidade do circuito, P. Uma vez que as portas são implementadas em um certo intervalo de tempo no dispositivo quântico, esta propriedade está associada ao tempo total que o dispositivo levará para executar o circuito.

O circuito de 4 qubits mostrado na figura 17 possui quatro camadas, ou seja, $P = 4$. A primeira camada é composta pelas portas **H** sobre os qubits q_0 e q_1. A segunda, pelas duas **CNOT**s sobre os qubits (q_0, q_2) e (q_1, q_3), onde os primeiros são os controles. A terceira camada é composta pela **CNOT** sobre (q_2, q_0) e as portas **X** sobre q_1 e q_3. A quarta e última camada é dada pelas medições sobre todos os qubits.

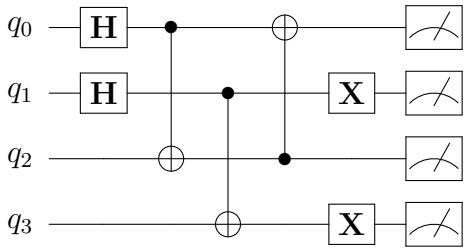


Figura 17 – Circuito de 4 qubits com profundidade $P = 4$.

Uma maneira gráfica de entender e determinar a profundidade de um circuito é através de uma analogia com o jogo Tetris.[8] Considere o circuito de 12 qubits com diversas

[8] Esta analogia foi proposta na documentação da ferramenta Qiskit. A descrição e a animação de onde

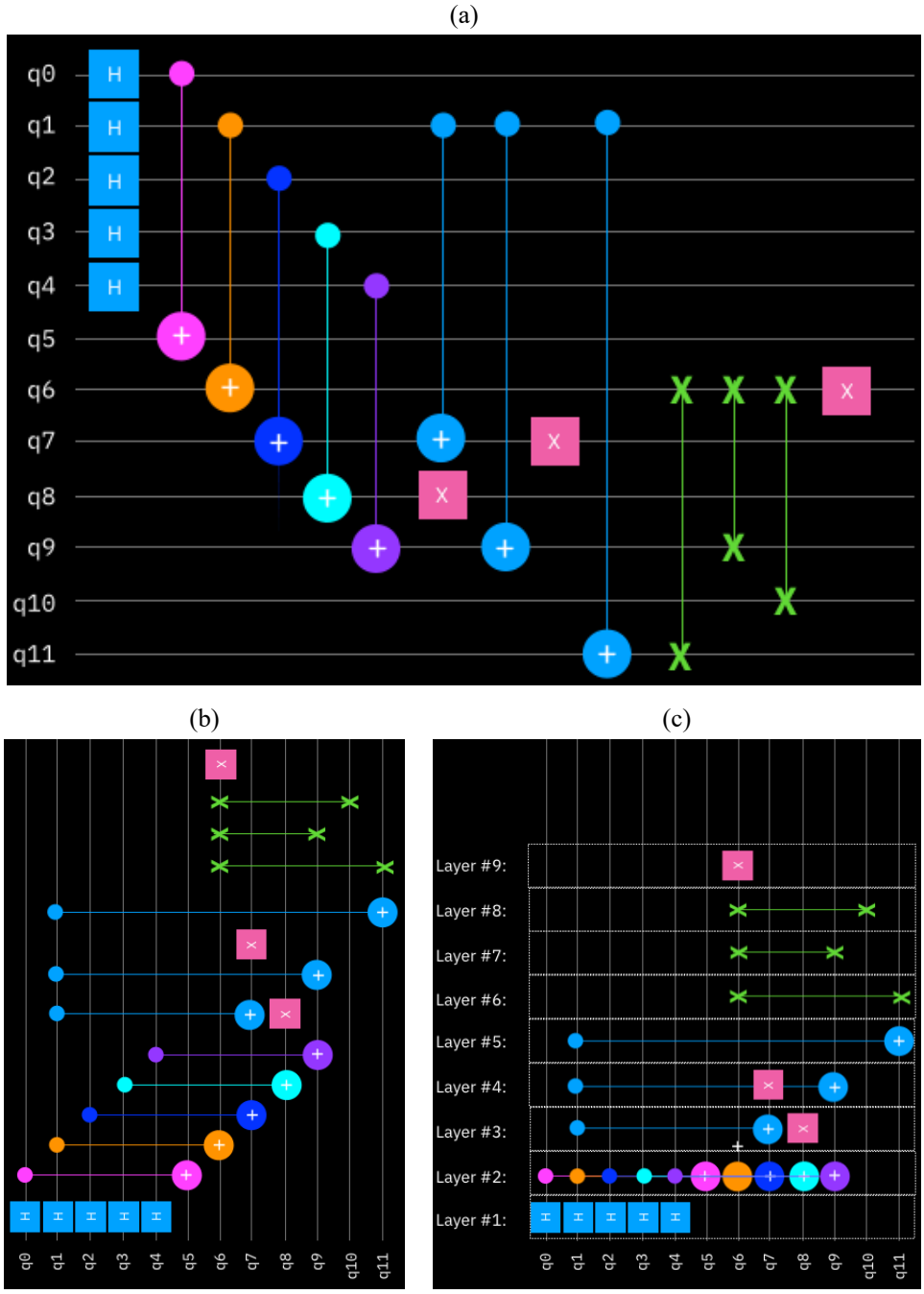


Figura 18 – Como computar a profundidade de um circuito "jogando" Tetris (veja a descrição no texto). Os quadros desta figura foram retirados de um gif animado disponível em https://qiskit.org/documentation/apidoc/circuit.html

portas de um e dois qubits mostrado na figura 18(a). Imagine que este circuito seja rodado de 90° no sentido anti-horário, como mostrado em 18(b), e as portas comecem a cair o mais longe possível, assim como as peças (*tetraminós*) do jogo. Ao final, as portas que podem ser executadas em paralelo se agruparão em camadas como mostrado na figura 18(c). O número de camadas é a profundidade do circuito, neste caso, $P = 9$.

retiramos os quadros da figura 18 podem ser vistas em https://qiskit.org/documentation/apidoc/circuit.html.

3 Os computadores quânticos da IBM

No capítulo anterior, nós introduzimos os elementos básicos para a realização de algoritmos quânticos no modelo de circuitos da computação quântica. Neste capítulo, vamos descrever como os algoritmos são implementados a partir de um circuito abstrato e como é possível realizar experimentos de forma remota nos computadores quânticos da IBM. Finalizamos com uma breve discussão sobre a implementação física desses computadores baseados em qubits supercondutores.

3.1 Implementando algoritmos em um computador quântico real

Os circuitos quânticos para implementação de algoritmos em um computador quântico real da IBM podem ser construídos e simulados através de uma interface gráfica chamada *Quantum composer*[1] ou de um kit de desenvolvimento de código aberto chamado *Qiskit*, sobre o qual falaremos ao final desta seção. Ao executar os circuitos construídos nas máquinas disponíveis atualmente, o usuário deve conhecer aspectos operacionais básicos que serão úteis para entender suas limitações, interpretar os resultados obtidos e otimizar os algoritmos. Vamos discutir alguns desses aspectos nesta seção.

3.1.1 Portas-base

Na construção de circutos quânticos, seja através do *Quantum Composer* ou do Qiskit, o usuário dispõe de todas as portas de um e dois qubits apresentadas no capítulo anterior e até mesmo de portas de mais qubits. Porém, apenas algumas dessas portas, chamadas de *portas-base*, podem ser implementadas fisicamente nos computadores da IBM. Na maioria dos computadores atuais, incluindo aqueles que usamos neste trabalho, as portas-base são $\{\mathbf{I}, \mathbf{X}, \sqrt{\mathbf{X}}, \mathbf{R}_z(\theta), \mathbf{CNOT}\}$, mostradas na figura 19. Com exceção de $\sqrt{\mathbf{X}}$, a representação matricial das demais portas deste conjunto foram dadas no capítulo 2. Usando as equações (2.11) e (2.17), temos

$$\sqrt{\mathbf{X}} = \frac{1}{2}\begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix} = e^{i\pi/4}\mathbf{R}_x(\pi/2). \tag{3.1}$$

A menos de uma fase global, a porta $\sqrt{\mathbf{X}}$ corresponde a uma rotação de $\pi/2$ em torno do eixo x da esfera de Bloch e ao ser aplicada duas vezes em sequência produz a porta $\mathbf{X}$.

Este conjunto restrito de portas-base é universal, ou seja, ele permite, em princípio, a implementação de operações unitárias arbitrárias sobre estados de n qubits nos computa-

[1] Para acessar a interface, o usuário deve criar uma conta na *IBM Quantum Platform* https://quantum-computing.ibm.com/.

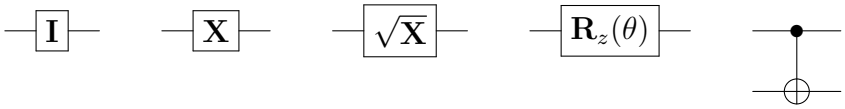


Figura 19 – Portas-base dos computadores quânticos da IBM.

dores da IBM. Como descrevemos na seção 2.3.3, o conjunto de portas $\{\mathbf{H}, \mathbf{S}, \mathbf{T}, \mathbf{CNOT}\}$ é universal. Entre as portas-base da IBM já temos a **CNOT**; a menos de uma fase global, as portas de um qubit $\mathbf{H}, \mathbf{S}$ e $\mathbf{T}$ do conjunto universal podem ser implementadas pelas portas do conjunto base como

$$\mathbf{H} = e^{i\pi/4}\mathbf{R}_z(\pi/2)\sqrt{\mathbf{X}}\mathbf{R}_z(\pi/2), \tag{3.2}$$

$$\mathbf{S} = e^{i\pi/4}\mathbf{R}_z(\pi/2), \tag{3.3}$$

$$\mathbf{T} = e^{i\pi/8}\mathbf{R}_z(\pi/4), \tag{3.4}$$

o que pode ser verificado usando as equações (2.19), (2.20), (2.23), (2.24) e (3.1). A figura 20 registra a relação entre estas portas nos computadores quânticos da IBM.

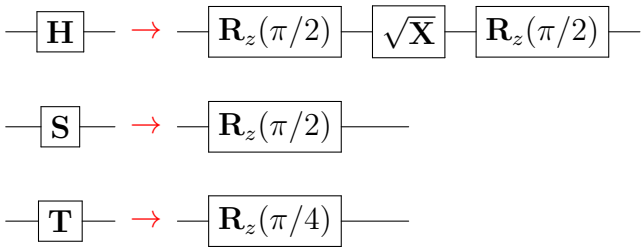


Figura 20 – Implementação das portas **H**, **S** e **T** em termos das portas-base nos computadores quânticos da IBM.

Em geral, o conjunto restrito de portas-base implicará que os circuitos implementados fisicamente no computador quântico tenham mais portas e, portanto, maior profundidade (veja seção 2.5.2). Isto ocorre porque muitas portas de interesse do usuário deverão ser implementadas por combinações de duas ou mais portas deste conjunto; a decomposição de **H** mostrada na figura 20 é um exemplo disso. Como consequência do aumento do número de portas e do tempo de execução do circuito, a performance do algoritmo será deteriorada pelo aumento do ruído, como vamos discutir mais adiante.

Para ilustrar o problema de forma mais concreta, vamos considerar operações que serão importantes para o método de reconstrução de estados que vamos implementar:

- **Medições projetivas de Pauli**. Os circuitos originais para estas medições foram mostradas na figura 12 e, em termos das portas-base, eles terão a forma mostrada na figura 21. Observa-se que para as medições associadas a **Z** e **Y** o número de portas não se altera; no caso de **Y** temos que $\mathbf{HS}^\dagger \to \mathbf{R}_z(\pi/2)\sqrt{\mathbf{X}}\mathbf{R}_z(\pi/2)\mathbf{R}_z(-\pi/2) = \mathbf{R}_z(\pi/2)\sqrt{\mathbf{X}}$. Para a medição associada a **X**, o número de portas aumenta de um

para três, e a profundidade vai de $P = 2$ no circuito original para $P = 4$ no circuito implementado fisicamente.

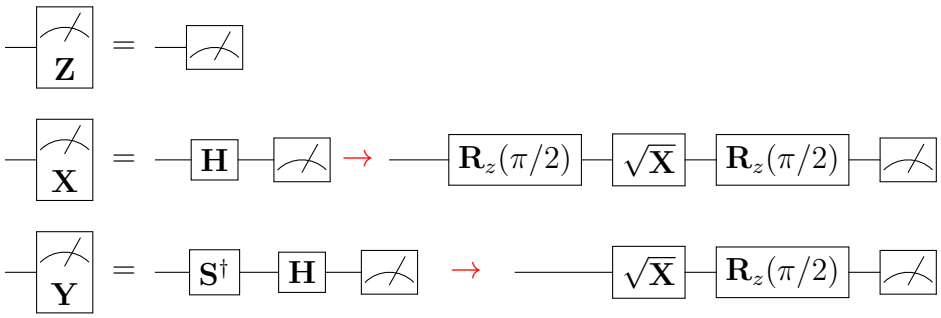


Figura 21 – Medições projetivas de Pauli em termos das portas-base da IBM.

- **Portas de fase controladas**. Como veremos no capítulo 5, a transformada quântica de Fourier em um circuito de n qubits é implementada através de portas **H**, $\mathbf{P}^{\text{ctrl}}(\lambda)$ e **SWAP**. A decomposição de **H** foi mostrada acima, e no capítulo 2 vimos que a **SWAP** é decomposta em uma sequência de três **CNOT**s (figura 7). Para as portas de fase controladas, utiliza-se a decomposição ABC descrita na seção 2.4.1 (veja figura 8). Em termos das portas-base, a decomposição ABC de $\mathbf{P}^{\text{ctrl}}(\lambda)$ (a menos de uma fase global de $e^{-i\lambda/4}$) é mostrada na figura 22. Portanto, esta porta é implementada nos computadores da IBM através de três portas de um qubit e duas **CNOT**s, em um circuito com profundidade $P = 5$.

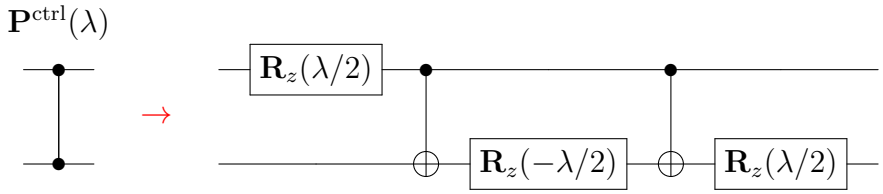


Figura 22 – Decomposição da porta de fase controlada em termos das portas-base da IBM.

Além das operações importantes para o nosso método de reconstrução de estados, a restrição do conjunto de portas-base também terá efeito na preparação de estados. Por exemplo, na preparação do estado emaranhado W de três qubits, dado pela equação (2.8), uma das portas utilizadas é a Hadamard controlada. A implementação desta operação em termos das portas-base requer uma **CNOT** e cinco portas de um qubit, em um circuito com $P = 6$, como mostrado na figura 23.

3.1.2 Conectividade

Em um computador quântico, dois qubits são ditos "conectados" quando é possível aplicar portas de dois qubits entre eles de forma direta. Esta conexão é unidirecional, quando

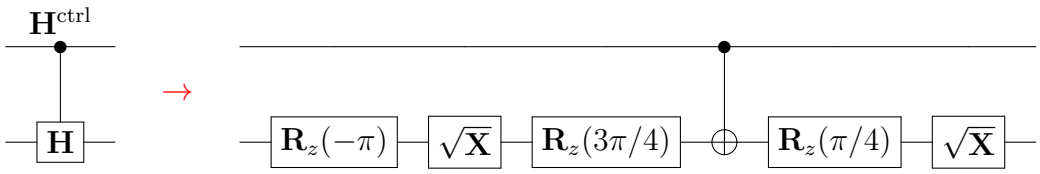


Figura 23 – Decomposição da porta Hadamard controlada em termos das portas-base da IBM.

apenas um deles pode ser o qubit controle em uma porta controlada, ou bidirecional, quando ambos podem ser controle, como ilustrado nas figuras 24(a) e 24(b), respectivamente. Os diagramas acima dos circuitos da figura 24 são conhecidos como mapas de conectividade ou mapas de acoplamento de um computador quântico. Nestes diagramas, os qubits são representados por círculos e as conexões entre eles por setas (caso unidirecional) ou linhas (caso bidirecional) ligando os círculos.

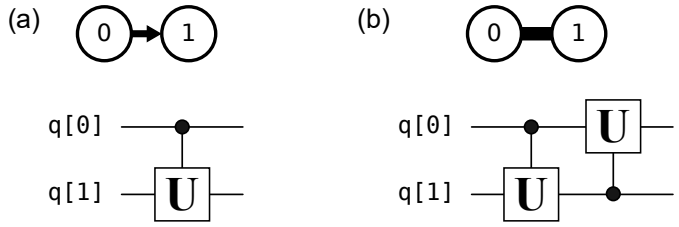


Figura 24 – Conectividade de dois qubits: (a) unidirecional e (b) bidirecional.

Nos primeiros modelos de computadores quânticos da IBM, as conexões entre os qubits eram apenas unidirecionais; atualmente, todas as máquinas em operação possuem apenas conexões bidirecionais. Assim, quando dois qubits, i e j, estão conectados, é possível aplicar $\mathbf{CNOT}_{i,j}$ e $\mathbf{CNOT}_{j,i}$ entre eles. Para ilustrar a questão da conectividade nas máquinas da IBM, vamos considerar um computador de cinco qubits. Antes de prosseguir, é importante fazer a seguinte distinção entre os qubits representados em um mapa de acoplamento e em um circuito quântico:

- **Qubits "físicos"**: os qubits em um mapa de acoplamento são a representação física dos qubits usados na computação.
- **Qubits "virtuais"**: os qubits em um circuito quântico são a representação virtual dos qubits físicos e devemos ter um mapeamento um-para-um entre eles. Portanto, um qubit q_j em um circuito não necessariamente corresponderá ao qubit j do mapa.

Para simplificar a discussão a seguir, vamos considerar o mapeamento trivial $q_j \to j$ entre os qubits do circuito e do mapa. O mapa de acoplamento no topo da figura 25(a) representa um computador *completamente conectado*, pois todos os qubits estão ligados entre si por uma linha. Ao todo, há conexão entre 10 pares distintos de qubits e portanto,

20 **CNOT**s permitidas entre eles, como mostra o circuito abaixo do mapa. Por outro lado, os computadores de cinco qubits da IBM em operação têm os mapas de acoplamento mostrados no topo das figuras 25(b) e 25(c). Em ambos, observa-se que somente quatro pares de qubits são conectados, e as 8 **CNOT**s permitidas em cada caso são mostradas nos circuitos abaixo dos mapas.

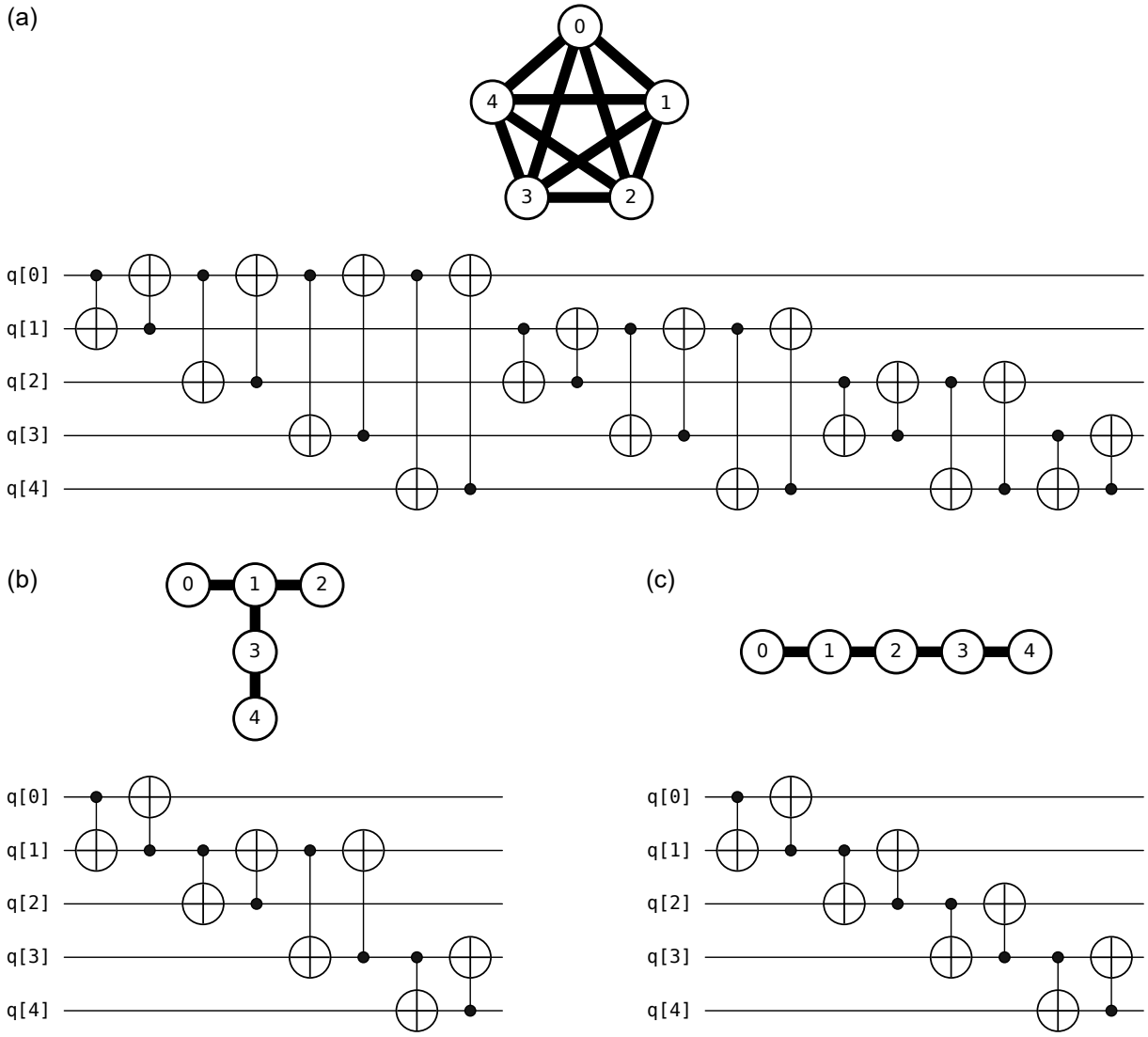


Figura 25 – Mapas de acoplamento e circuitos de **CNOT**s permitidas em um computador quântico de cinco qubits, considerando o mapeamento trivial $q_j \to j$ entre qubits virtuais e físicos. (a) Computador completamente conectado. (b) e (c) computadores da IBM em operação atualmente.

Em geral, um computador quântico completamente conectado de n qubits requer a conexão entre todos os $n(n-1)/2$ diferentes pares. A figura 26 mostra os mapas de acoplamento de algumas das máquinas da IBM em operação com $n = 7, 27, 65$ e 127 qubits.[2] Em todos eles se observa uma conectividade bem abaixo da completa. Neste

[2] As cores dos círculos e linhas representam a magnitude de parâmetros físicos associados aos qubits e às **CNOT**s, respectivamente, tais como tempos de decoerência dos qubits, erros na execução das portas,

trabalho, nós utilizamos computadores de 7 qubits com mapas equivalentes ao da figura 26(a), os quais possuem apenas 6 pares de qubits conectados entre os 21 possíveis.

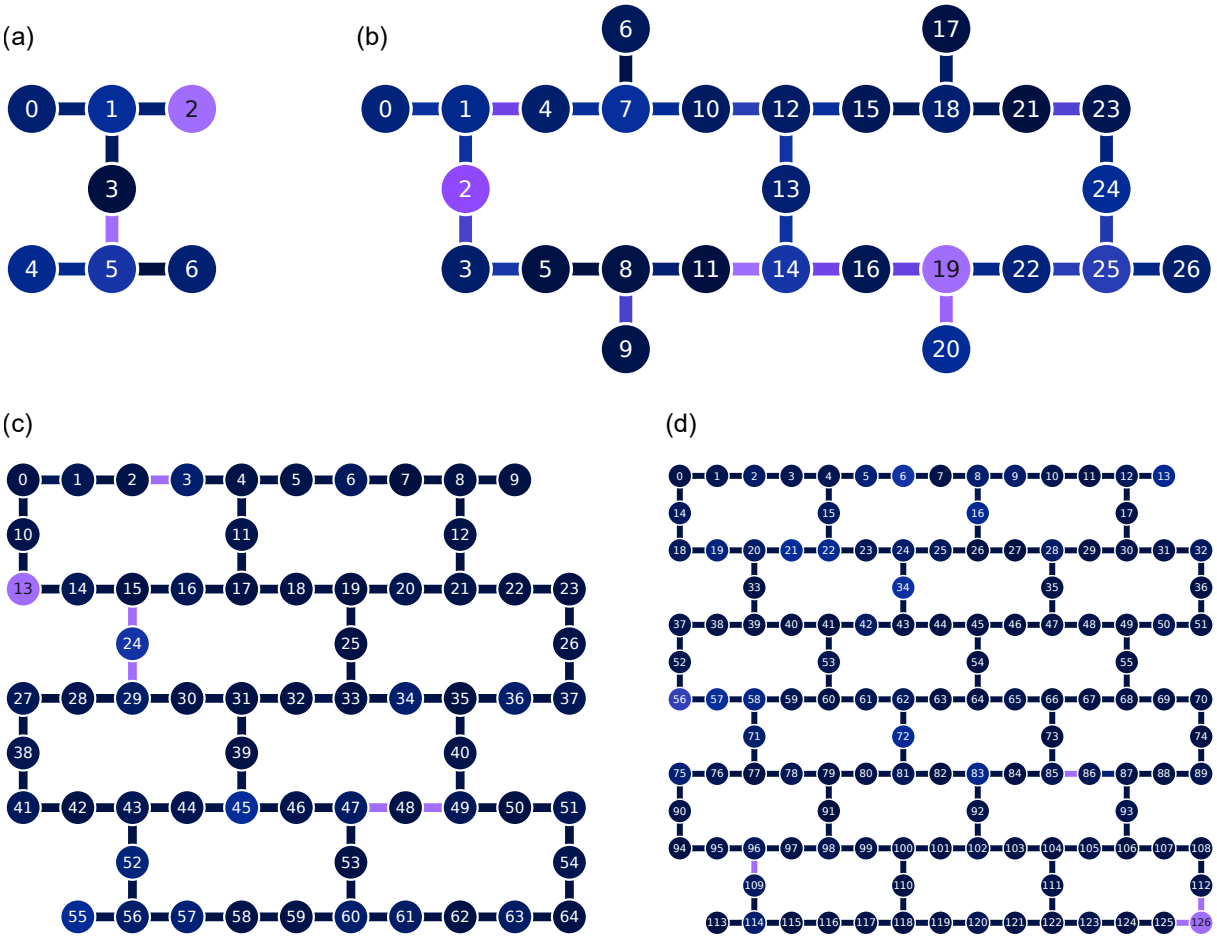


Figura 26 – Mapas de acoplamento de alguns computadores quânticos da IBM em operação atualmente com (a) $n = 7$, (b) $n = 27$, (c) $n = 65$ e (d) $n = 127$ qubits.

Como implementar uma **CNOT** entre dois qubits não-conectados

Na construção de circuitos quânticos, além de dispor de todas as portas de um, dois ou mais qubits, o usuário também dispõe de conectividade completa, ou seja, as portas de dois qubits podem ser aplicadas em qualquer par de qubits no circuito. Porém, vimos que nem todos os pares de qubits estão diretamente conectados nos computadores quânticos da IBM. Neste caso, para implementar uma **CNOT** em um par não-conectado, é necessário aplicar uma ou mais portas **SWAP** para mover os estados dos qubits de interesse até que eles estejam em posições adjacentes no mapa de acoplamento.

Transpilação de circuitos quânticos

O conjunto restrito de portas-base e a necessidade de usar portas **SWAP**(= 3 **CNOT**s) para suprir a conectividade limitada são fatores que aumentarão o número de

etc. O usuário pode verificar cada um desses parâmetros em tempo real ao executar um circuito.

portas e a profundidade de um circuito quântico a ser executado no computador. Como consequência, haverá aumento de ruído, o que pode inviabilizar a computação. Para reduzir estes efeitos existe um recurso importante: a transpilação[3] de circuitos quânticos, descrita resumidamente no quadro abaixo (para informações mais detalhadas, consulte [32]).

Transpilação de circuitos quânticos

Transpilação é o processo de reescrever um dado circuito de entrada em um circuito compatível com o conjunto de portas-base e a conectividade do computador quântico. É possível ainda otimizar este circuito no processo para reduzir efeitos de ruído. O transpilador realiza as seguintes tarefas:

- ***Layout:*** mapeia os qubits virtuais do circuito de entrada nos qubits físicos do mapa de acoplamento do computador.
- **Roteamento:** introduz portas **SWAP** no circuito para torná-lo compatível com a conectividade do computador.
- **Tradução:** decompõe as portas quânticas do circuito de entrada em termos das portas-base do dispositivo.
- **Otimização:** busca reduzir a profundidade do circuito através da combinação ou eliminação de portas quânticas (especialmente das **SWAP**s).

Os três pontos iniciais já são suficientes para executar um circuito de entrada no dispositivo quântico, ou seja, é possível transpilar o circuito sem otimizá-lo.[4] Neste caso, o transpilador faz o mínimo necessário: inicia com um *layout* trivial ($q_j \to j$), adiciona as **SWAP**s necessárias e converte todas as portas para as portas-base. Como exemplo, vamos considerar a preparação do estado GHZ de quatro qubits (equação (2.9)) em um computador quântico com o mapa de acoplamento mostrado na figura 27(a). O circuito de entrada para a preparação é mostrado na figura 27(b). O circuito transpilado sem otimização é mostrado na figura 27(c); com o *layout* trivial, é necessário aplicar uma **SWAP** para "conectar" o qubit 0 aos qubits 2 e 3. Finalmente, a figura 27(d) mostra o circuito transpilado com otimização; com o *layout* mostrado à esquerda do circuito foi possível eliminar a porta **SWAP**.

O objetivo primordial na etapa de otimização é reduzir o ruído no circuito de saída

[3] Em computação, transpilação é a tradução do código-fonte de um programa escrito em uma linguagem de programação em um código-fonte equivalente na mesma ou em outra linguagem de programação. Um transpilador é conhecido também como compilador *source-to-source* ou transcompilador.

[4] Ao aplicar a função `transpile` na ferramenta Qiskit, o usuário define o nível desejado de otimização do circuito. Este nível vai de 0 a 3, onde o nível 0 representa nenhuma otimização e os níveis de 1 a 3 realizam otimizações progressivamente mais robustas através de algoritmos progressivamente mais sofisticados e custosos computacionalmente [32].

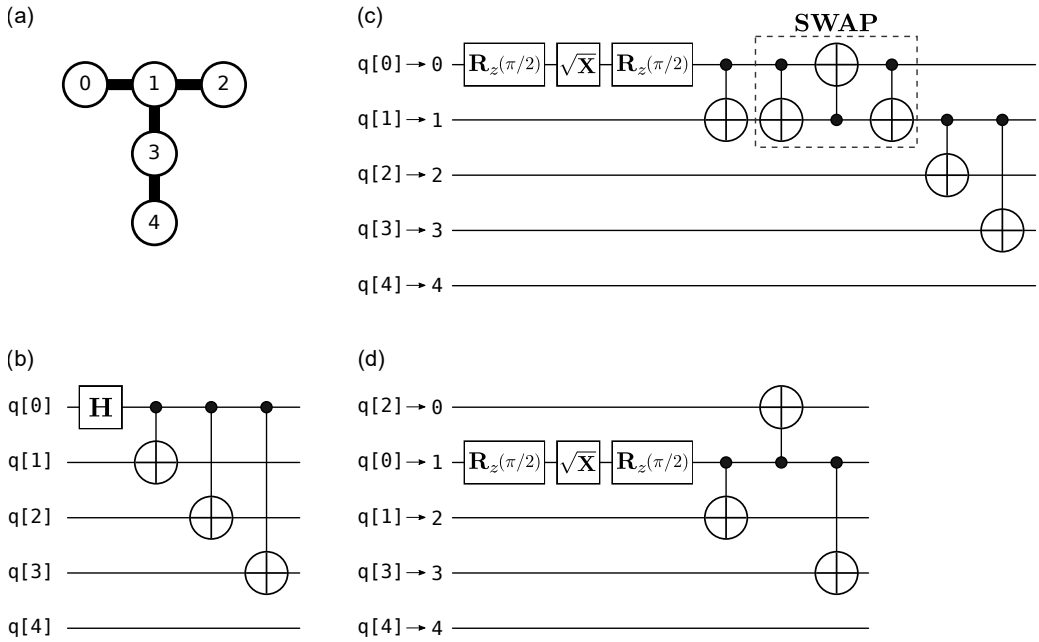


Figura 27 – (a) Mapa de acoplamento de um computador de 5 qubits. (b) Circuito de entrada para preparação do estado GHZ de 4 qubits neste computador. (c) Circuito transpilado sem otimização. (b) Circuito transpilado com otimização.

o máximo possível. Isto requer a minimização do número de portas,[5] especialmente de **SWAP**s. Esta operação utiliza três portas **CNOT** que é a mais ruidosa do conjunto-base do dispositivo. No exemplo acima, esta foi uma tarefa fácil devido à simplicidade do problema. Em circuitos mais complexos, a tarefa de encontrar o *layout* que minimize o número de **SWAP**s é muito mais difícil e custosa computacionalmente. O transpilador contorna esta dificuldade utilizando um método estocástico para determinar um bom *layout*, mas que não necessariamente será o melhor. Com este método, um circuito de entrada transpilado repetidamente vai gerar diferentes circuitos de saída. Para determinar qual o melhor circuito possível, uma opção é transpilar diversas vezes a entrada, computando a profundidade e o número total de portas de um e dois qubits para cada saída. A partir das distribuições obtidas desses parâmetros, se escolhe o circuito que apresenta os valores mínimos. Nós adotamos esta abordagem na otimização dos circuitos de reconstrução de estados, como será discutido no capítulo 7.

3.1.3 Ruído

O ruído é uma característica marcante dos computadores quânticos atuais, constituindo um obstáculo para a construção de dispositivos de maior escala e a realização

[5] Um do passos da otimização é tentar combinar as portas do circuito de entrada em grupos menores e também eliminar sequências de portas sem efeito no algoritmo. Exemplos: a sequência $\mathbf{R}_z(\pi/2)\mathbf{R}_z(\pi/4)$ é equivalente a $\mathbf{R}_z(3\pi/4)$; a sequência $\mathbf{CNOT}_{i,j}\mathbf{CNOT}_{i,j} = \mathbf{I}$ e pode ser eliminada; portas dadas por matrizes diagonais imediatamente antes de uma medição são eliminadas, pois não alteram os resultados da medição na base computacional.

de computação realmente útil. Por esta razão, esses dispositivos são chamados de NISQs, conforme definido no capítulo 1. Nesta subseção vamos apenas descrever as principais fontes de ruído e seus efeitos, sem a preocupação de explicar a origem física de cada um ou como são caracterizados, que está além dos propósitos deste trabalho.

Podemos dividir o ruído no dispositivo quântico em três fontes [33]: infidelidade de portas, decoerência e erros de medição. Vamos descrever cada uma dessas fontes e mostrar, como exemplo, os dados de calibração de um processador quântico da IBM de 7 qubits, cujo mapa de acoplamento é mostrado na figura 26(a). Os gráficos que serão apresentados estão disponíveis na plataforma da IBM e não diferem de forma significativa na comparação entre os diferentes processadores atuais.

Infidelidade de portas

A infidelidade de uma porta se refere à imprecisão da porta que é implementada fisicamente em relação à porta especificada no circuito. O erro induzido pela aplicação desta porta imprecisa pode ser caracterizado [33]: a figura 28 mostra a medida das taxas de erro da porta **X** sobre cada qubit (gráfico superior) e da **CNOT** sobre cada par de qubits conectado (gráfico inferior); a linha tracejada em cada gráfico representa a mediana dos resultados: $\sim 2{,}6 \times 10^{-4}$ e $\sim 8{,}3 \times 10^{-3}$, respectivamente. As demais portas de um qubit geram gráficos similares ao de **X**. É nítido que o erro induzido pela **CNOT** é maior que o das portas de um qubit e por isso é importante minimizar o número de **SWAP**s no algoritmo.

Decoerência

A coerência quântica representa a capacidade de superposição de estados quânticos, que é o grande diferencial da computação quântica sobre a clássica. Porém, quando qubits físicos interagem com o ambiente, há uma perda gradual de coerência ao longo do tempo. Este efeito de *decoerência* introduz ruído na execução de um algoritmo à medida que ele avança no tempo e, a longo prazo, levará um computador quântico a se comportar como um objeto clássico. Existem duas escalas temporais, T_1 e T_2, que caracterizam a decoerência, definidas como:

- Tempo de relaxação (T_1): é uma medida de quão rapidamente o estado $|1\rangle$ se transforma espontaneamente no estado $|0\rangle$.
- Tempo de defasagem (T_2): é uma medida de quão rapidamente os termos de coerência[6] de uma superposição entre $|0\rangle$ e $|1\rangle$ são atenuados.

[6] Termos fora da diagonal do operador densidade do estado (veja capítulo 4). Quando esses termos são nulos, o estado é incoerente na base em que está escrito [7].

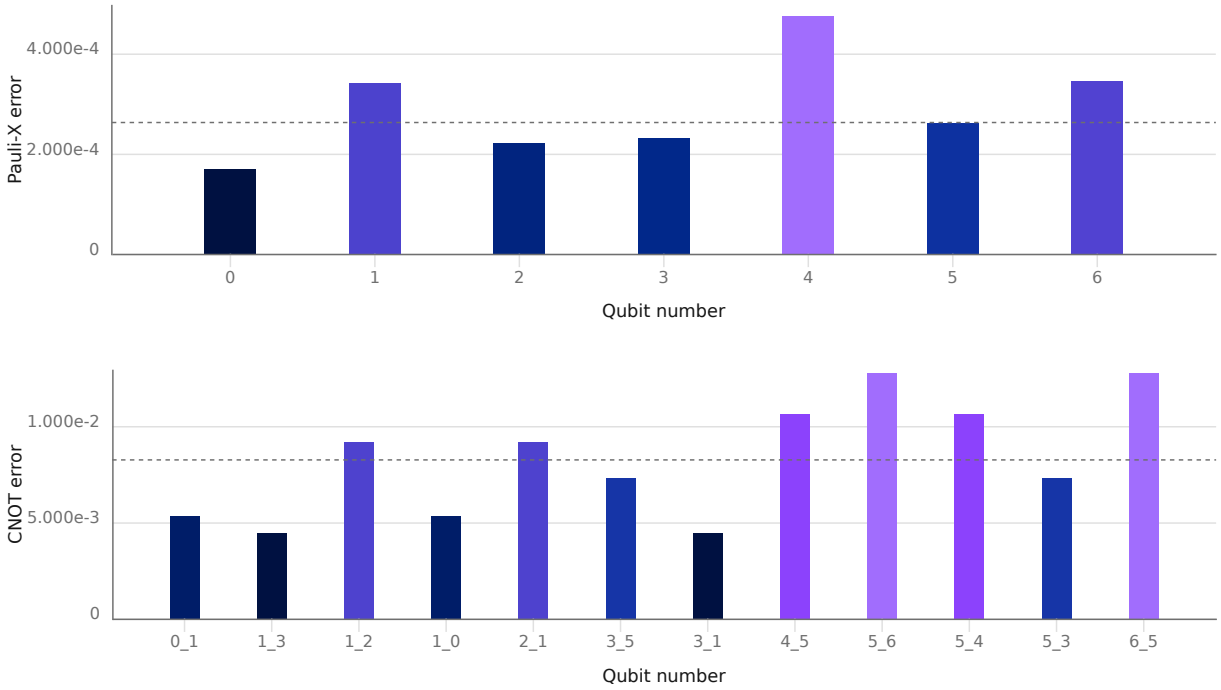


Figura 28 – Taxa de erros da porta **X** (gráfico superior) e da **CNOT** (gráfico inferior) em um computador quântico de 7 qubits com o mapa de acoplamento mostrado na figura 26(a). A linha tracejada representa a mediana dos resultados.

Os efeitos de decoerência podem ser minimizados ou até suprimidos, desde que o tempo de execução de um algoritmo, t_{circ}, satisfaça as condições $t_{\text{circ}} \ll T_1$ e $t_{\text{circ}} \ll T_2$. Portanto, as quantidades T_1 e T_2 limitam a profundidade dos circuitos que podem ser implementados de forma eficaz em um computador quântico.

A decoerência afeta os qubits a taxas diferentes como se vê na figura 29, onde T_1 e T_2 apresentam medianas em torno de 160 e 100 μs, respectivamente. Comparativamente, é interessante mostrar também os tempos de execução de elementos do circuito. A figura 30 mostra o tempo de execução de cada **CNOT** (gráfico superior) e das medições (gráfico inferior), com medianas em torno de 500 e 700 ns, respectivamente.[7]

Erros de medição

A imprecisão do processo de medição em um computador quântico é caracterizada pela probabilidade de obtermos um resultado incorreto ao medir o qubit. A figura 31 mostra uma distribuição típica de probabilidades de erro de leitura, com uma mediana $\sim 2{,}5 \times 10^{-2}$. Este valor mostra que a medição é um processo mais sensível a erros que a aplicação das portas quânticas.

[7] O tempo para executar medições é o parâmetro com discrepâncias mais significativas entre os processadores da IBM. Existem processadores cujos tempos chegam a 6000 ns.

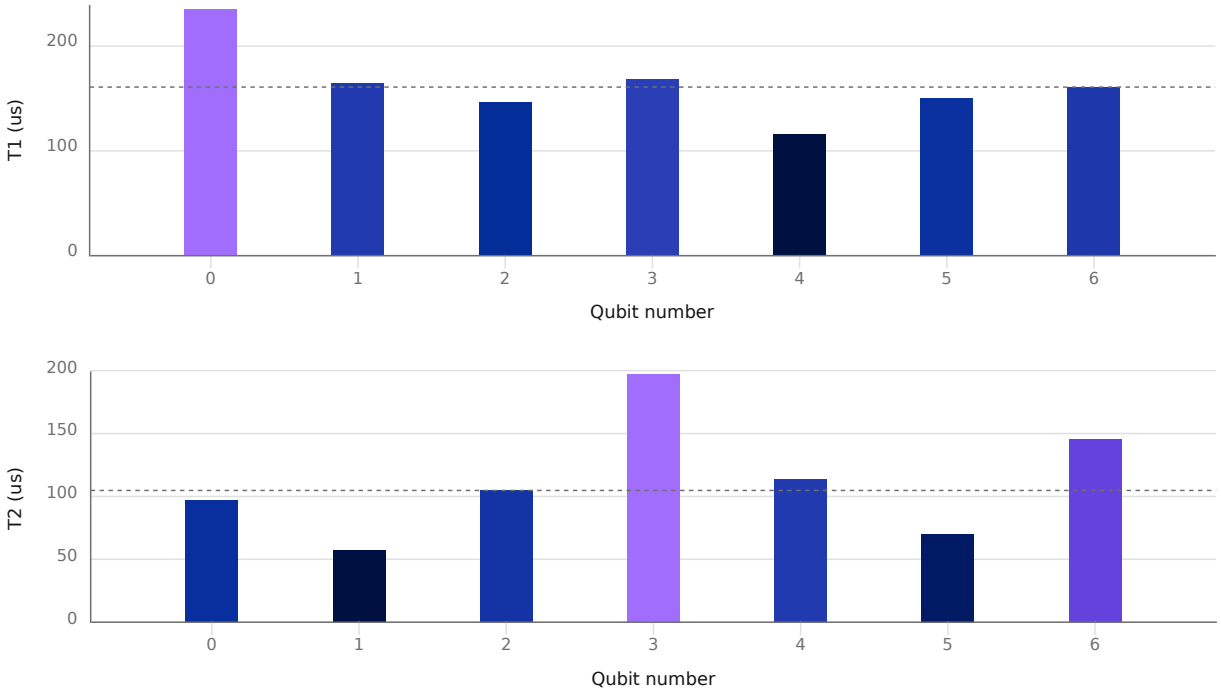


Figura 29 – Escalas temporais de decoerência: tempo de relaxação, T_1, (gráfico superior) e tempo de defasagem, T_2, (gráfico inferior) em um computador quântico de 7 qubits com o mapa de acoplamento mostrado na figura 26(a). A linha tracejada representa a mediana dos resultados.

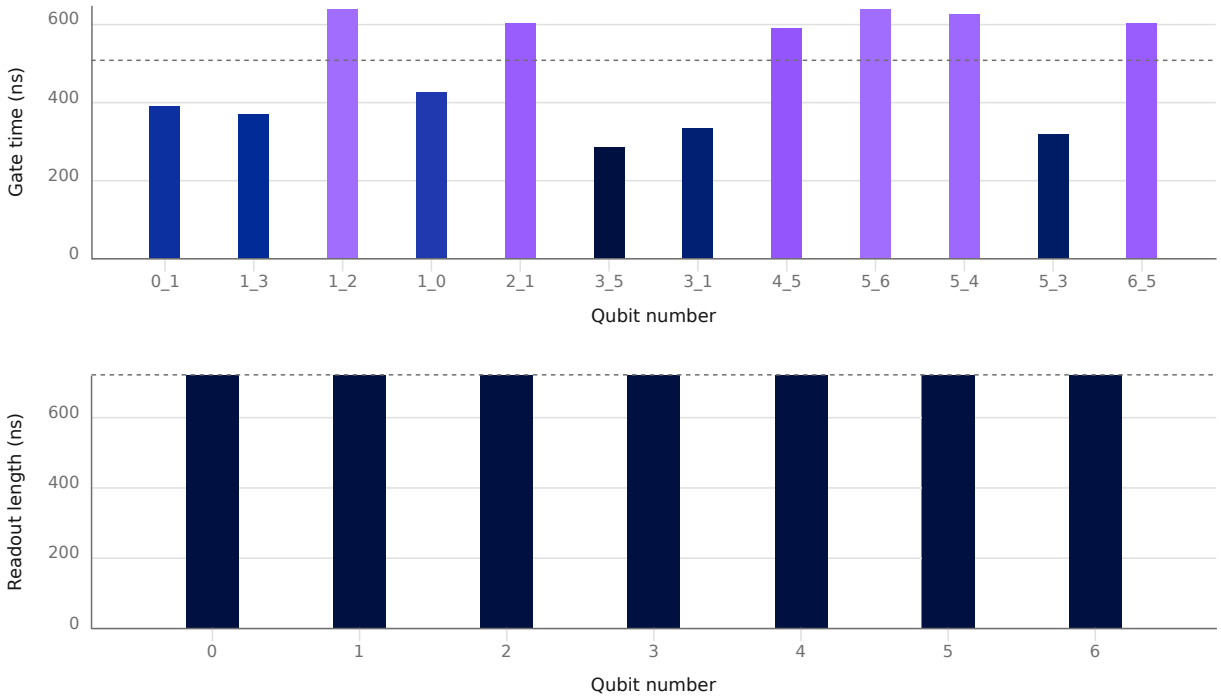


Figura 30 – Tempos de execução das portas **CNOT** (gráfico superior) e das medições (gráfico inferior) em um computador quântico de 7 qubits com o mapa de acoplamento mostrado na figura 26(a). A linha tracejada representa a mediana dos resultados.

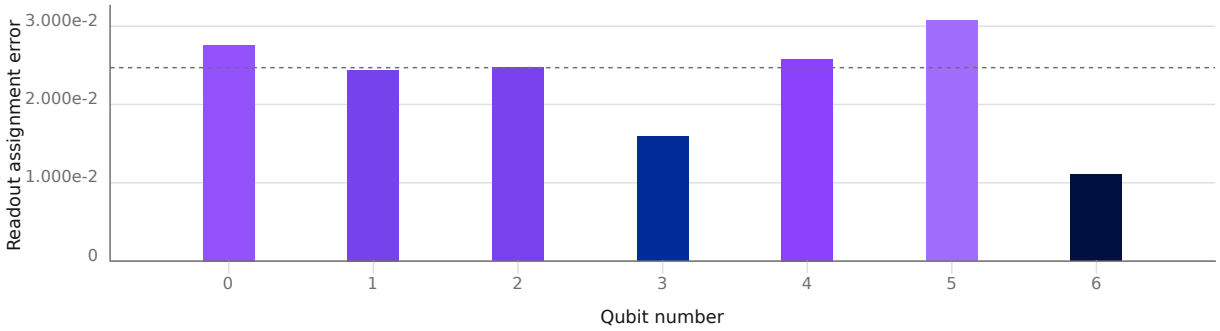


Figura 31 – Taxa de erros no processo de medição em um computador quântico de 7 qubits com o mapa de acoplamento mostrado na figura 26(a). A linha tracejada representa a mediana dos resultados.

3.1.4 Computação quântica na nuvem usando o Qiskit

Após apresentar alguns dos aspectos operacionais básicos dos computadores quânticos da IBM, vamos finalizar esta seção descrevendo, em linhas gerais, como um usuário pode realizar desde simulações computacionais até experimentos em processadores quânticos reais através do serviço de nuvem.

Para realizar computação quântica na nuvem, a IBM criou o Qiskit (*Quantum Information Software Development Kit*), um kit de desenvolvimento de *software* de código aberto baseado na linguagem de programação Python. Com esta ferramenta, o circuito do algoritmo quântico a ser implementado é construído através de um código em Python. Este código, que representa o circuito de entrada, é enviado à nuvem e então transpilado. Na nuvem os trabalhos são organizados em uma fila com uma estrutura de prioridades.[8] Ao chegar a vez do usuário, o algoritmo é executado e os resultados ficam armazenados na nuvem; eles podem ser enviados ao Qiskit do usuário, se assim for programado. A figura 32 ilustra este processo.

Existem inúmeros tutoriais (incluindo vídeos na internet) sobre o Qiskit. Entre esses, há um vasto material disponível gratuitamente no próprio sítio do Qiskit;[9] para iniciantes, a referência [34] fornece uma visão geral, bem como os primeiros passos para começar a programar.

Experimentos e simulações no Qiskit

Além de possibilitar a realização de experimentos via nuvem, o Qiskit é também uma ferramenta para simulação computacional. Com ele, é possível emular um computador quântico no computador (clássico) do usuário, sem necessidade de acesso à internet. Assim,

[8] Por padrão, a ordem na qual os trabalhos são executados é determinada por um algoritmo que tenta estabelecer uma forma justa de organização. Veja detalhes em *Fair-share queuing:* https://docs.quantum-computing.ibm.com/run/queue.

[9] https://qiskit.org/.

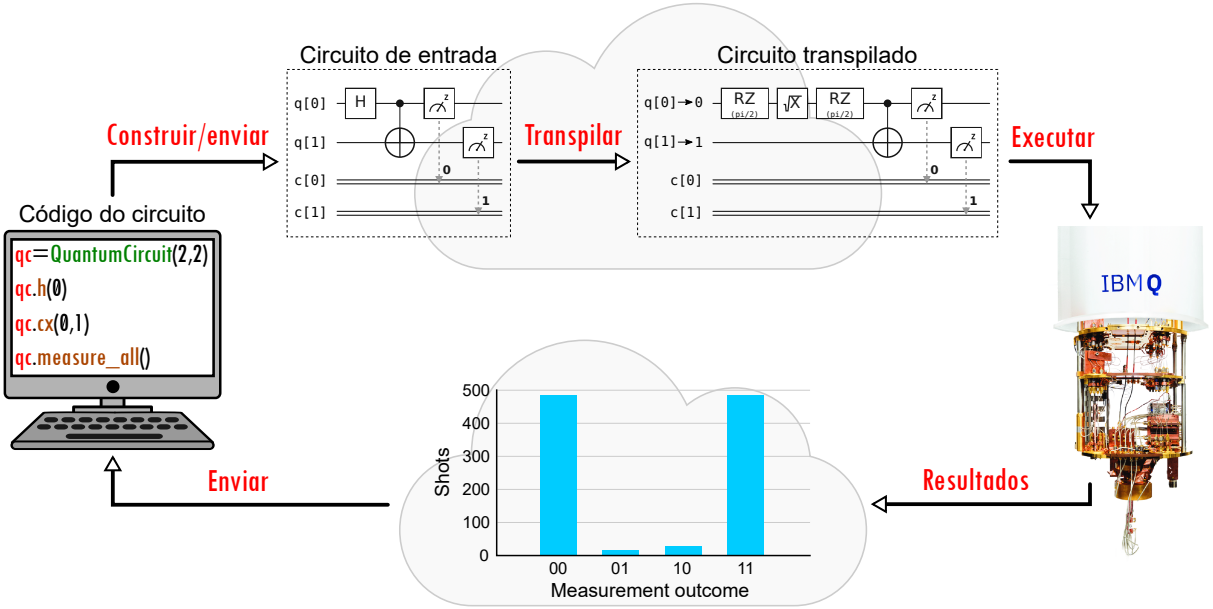


Figura 32 – Computação quântica na nuvem: implementando algoritmos em um computador quântico da IBM usando o Qiskit.

podemos simular algoritmos quânticos simples emulando cenários ideais sem ruído ou adicionando ruído.[10] Vamos voltar a esse ponto com mais detalhes no capítulo 6.

Ao rodar um dado circuito de n qubits em um computador quântico ou simulá-lo em um computador clássico, ele será executado repetidas vezes, produzindo contagens para as 2^n sequências de n bits clássicos. O número de repetições (chamadas de *shots* no Qiskit) é definido no código; na simulação esse número não tem limite, mas um valor muito alto implicará em tempo de execução maior, o que pode se tornar impraticável para circuitos mais complexos. Nas máquinas de livre acesso, o número máximo atualmente é 2×10^4 repetições. Em máquinas de acesso restrito, este número pode ser bem maior. Algumas dessas máquinas restritas são disponibilizadas pelo programa de pesquisadores da IBM, mediante a aceitação de uma solicitação. Para este trabalho, nossa solicitação foi aceita e pudemos utilizar alguns dos processadores quânticos de 7 qubits com até 10^5 *shots*. Em processadores mais avançadas de até 127 qubits, o acesso se dá através de planos pagos e o número de repetições dependerá do plano e do processador acessado.

3.2 Computadores quânticos supercondutores

Pela longa experiência em pesquisa com supercondutores a IBM optou por desenvolver seus hardwares quânticos tendo como base esses materiais. Nesta seção vamos quebrar um pouco da abstração por trás do modelo de circuitos da computação quântica e descrever de forma sucinta o que são os elementos físicos que compõem esses dispositivos.

[10] Tutoriais para simulações no Qiskit podem ser encontrados em https://qiskit.org/ecosystem/aer/tutorials/index.html.

3.2.1 Supercondutividade

Os materiais supercondutores exibem uma característica notável chamada supercondutividade, na qual a resistência elétrica vai a zero quando resfriados abaixo de uma temperatura crítica específica. Essa descoberta foi feita em 1911 por Heike Kamerlingh Onnes, que observou que a resistência de mercúrio praticamente desapareceu a temperaturas extremamente baixas, próximas do zero absoluto [35].

A chave para a supercondutividade reside na formação de pares de elétrons mediada por fônons; a composição dos 2 elétrons mais 1 fônon é chamada de par de Cooper, em homenagem ao físico Leon Cooper que descreveu pela primeira vez a ligação de um par de elétrons a baixas temperaturas. A formação desses pares gera deformações na rede cristalina do material, permitindo que a corrente elétrica flua sem dissipação de energia [36]. Essa propriedade torna os supercondutores extremamente eficientes em transportar eletricidade e abre portas para uma ampla gama de aplicações tecnológicas, como é o caso dos computadores quânticos da IBM.

3.2.2 O qubit supercondutor

A supercondutividade é um fenômeno quântico e podemos usar materiais com essa propriedade para construir qubits. Para entender como isto é possível vamos primeiramente considerar um circuito elétrico composto por um indutor e um capacitor (circuito LC), como ilustrado na figura 33(a). Formalmente, este circuito é análogo a um oscilador harmônico simples, com sua energia total constante oscilando (com frequência angular ω_r) entre a energia armazenada no capacitor e no indutor. No regime clássico, a energia total do sistema pode ter qualquer valor. Vamos supor agora um pequeno circuito LC construído com elementos supercondutores e mantido a baixas temperaturas (da ordem de poucos milikelvins). Neste cenário, o circuito será análogo a um oscilador harmônico quântico com níveis discretos de energia igualmente espaçados por $\hbar\omega_r$, como mostra a figura 33(b). Como os níveis são igualmente espaçados, não é possível controlar as transições entre somente dois deles, ou seja, não é possível definir o espaço de um qubit. Para contornar este problema, é necessário introduzir uma anarmonicidade no sistema, o que é feito trocando o indutor por uma junção Josephson no circuito, como mostra a figura 33(c). A junção Josephson atua como um indutor não-linear[11] gerando assim um espaçamento não-uniforme entre os níveis de energia do sistema, como mostra a figura 33(d). Portanto, agora é possível confinar a dinâmica de transições somente entre dois níveis do sistema e assim definir o espaço de um qubit; normalmente se utiliza o estado fundamental e o primeiro estado excitado ($|0\rangle$ e $|1\rangle$, respectivamente na figura 33(d)). O circuito construído

[11] Uma junção Josephson é composta por dois supercondutores separados por uma fina barreira de material isolante. Devido ao efeito quântico de tunelamento, os pares de Cooper podem passar através da junção, gerando uma corrente. Este fenômeno foi previsto por Brian David Josephson [37].

dessa forma constitui um átomo artificial devido à equivalência entre seus níveis de energia e os de um átomo.

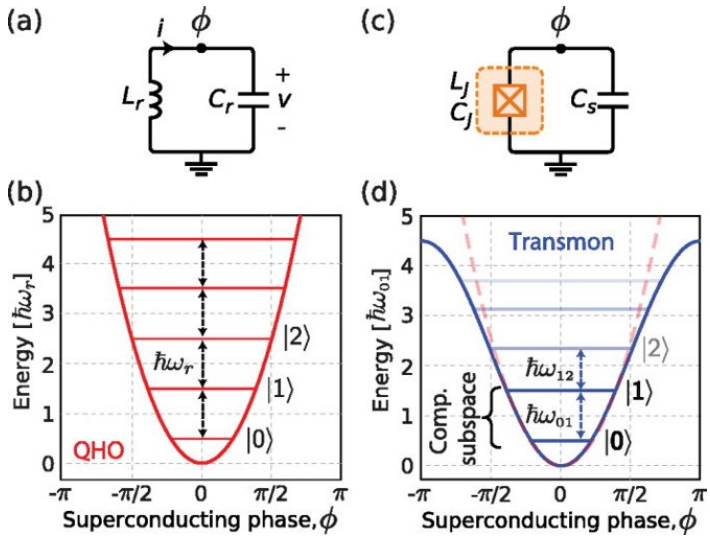


Figura 33 – Circuitos supercondutores e o regime transmon. (a) Circuito LC supercondutor do oscilador harmônico quântico (QHO). (b) Energia potencial do QHO com níveis equidistantes. (c) Circuito do transmon. (d) Energia potencial do transmon com níveis não uniformes. Figura retirada de [38].

Para se controlar os qubits de átomos artificiais, adiciona-se um capacitor ao circuito (como mostra a figura 34(a)), o qual permite a entrada de sinais externos (fótons) que serão usados para implementação de portas lógicas e medições. Porém, a inserção deste canal capacitivo provoca mudanças na separação dos níveis do átomo artificial em função da carga no capacitor, o que pode desconfigurar a anarmonicidade. Este problema é contornado aumentando-se a capacitância do capacitor no átomo artificial, que torna os seus níveis de energia menos sensíveis à carga do canal capacitivo e ao mesmo tempo preserva a anarmonicidade. O regime em que esse efeito ocorre é chamado de regime *transmon*, que também dá nome ao qubit supercondutor transmon e é o tipo de qubit usado nos computadores quânticos da IBM. Uma descrição detalhada dos vários tipos de qubits supercondutores, incluindo o transmon, pode ser vista em [10].

3.2.3 Operações quânticas

A implementação de portas quânticas sobre um qubit transmon é realizada através da aplicação de pulsos de micro-ondas, como ilustrado na figura 34(b). A frequência desse pulso é ajustada de acordo com a frequência de transição entre os níveis de energia do qubit. Além da frequência, outras características importantes para definir uma determinada porta quântica são a amplitude, a fase, a duração e o formato do pulso. Com pulsos cuidadosamente projetados, é possível implementar as portas-base de um qubit descritas na seção anterior.

Na implementação das portas de um qubit basta endereçar o pulso para o qubit de interesse. Mas, para implementar portas de dois qubits, como a **CNOT**, primeiro é necessário estabelecer uma conexão entre os qubits transmon. Esta conexão é realizada por meio de um acoplamento capacitivo, como ilustrado na figura 34(c), que permite que os qubits interajam entre si [12], tornando possível a execução de operações quânticas que produzam, por exemplo, estados emaranhados. Tais portas são realizadas por meio de um pulso de micro-ondas de "ressonância cruzada" (conhecido em inglês como *cross-resonance gate*). O pulso é aplicado em um dos dois qubits, mas com uma frequência que corresponde à frequência natural do outro qubit. Isto gera um efeito de excitação no segundo qubit, porém com uma taxa que depende crucialmente do estado do primeiro qubit [39, 40]. Portanto, o pulso provoca uma rotação controlada no qubit alvo, que é determinada pelo estado do qubit de controle.

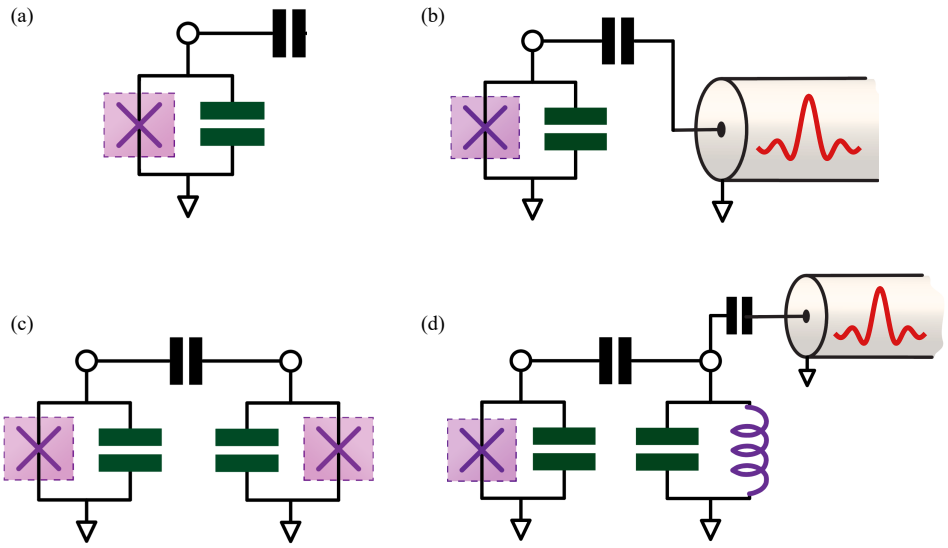


Figura 34 – Representação de circuitos dos elementos de um computador quântico com qubits supercondutores do tipo transmon. (a) Qubit transmon com um canal capacitivo destinado a controle e leitura. (b) Qubit transmon submetido a um pulso de micro-ondas. (c) Dois qubits transmon acoplados. (d) Qubit transmon acoplado a um ressonador de leitura. Figuras adaptadas de [41].

Por último, o processo de medição sobre um qubit transmon é realizado acoplando-se um ressonador a ele, chamado ressonador de leitura (*readout resonator*). O ressonador é construído de forma similar ao qubit, exceto que ao invés da junção Josephson, ele é composto por um indutor convencional, como ilustrado na figura 34(d). A frequência de ressonância do ressonador é bem distante da frequência de transição do qubit. Porém, o acoplamento ao transmon produz uma variação na frequência do ressonador dependente do estado do qubit. Detectando-se essa variação, é possível determinar se o estado do qubit é $|0\rangle$ ou $|1\rangle$. Para isso, o ressonador recebe um pulso com frequência próxima à sua; o pulso é refletido pelo ressonador e a voltagem de saída permite distinguir entre os estados da base computacional. Para minimizar a perturbação no qubit, os pulsos de leitura são da

ordem de poucos femtowatts, o que introduz ruído no sinal de saída e pode levar a erros de medição, como discutido na seção anterior.

3.2.4 Exemplo de chip quântico

Os dispositivos da IBM têm a forma mostrada no lado esquerdo da figura 35. Este grande "abajur" dourado é um refrigerador de diluição, instrumento capaz de atingir baixíssimas temperaturas, como mostra o esquema ao lado dele; a temperatura de operação do dispositivo é de 15 mK (−273.13 °C). Os cabos direcionam os pulsos para controle e leitura dos qubits transmon que se encontram em um chip na ponta do dispositivo, dentro de uma cápsula de blindagem. O exemplo de um chip quântico real da IBM é mostrado no lado direito da figura 35. A imagem mostra um chip de cinco qubits destacando seus principais elementos: em vermelho temos os qubits transmon, em azul os ressonadores de controle e leitura e em verde e amarelo os ressonadores de acoplamento. Os chips são compostos por uma pastilha de silício que contém qubits e ressonadores feitos de alumínio e nióbio. Os qubits são fabricados por meio da técnica de litografia de feixe de elétrons [42]. Como podemos ver na figura cada qubit está associado a um ressonador que fornece os pulsos que irão realizar as portas quânticas e as medições. Os ressonadores de acoplamento conectam os qubits, possibilitando a implementação das portas de dois qubits (**CNOT**). O design do dispositivo permite que os qubits operem em condições controladas, facilitando a implementação de operações quânticas e explorando as propriedades únicas da mecânica quântica, apesar do ruído presente.

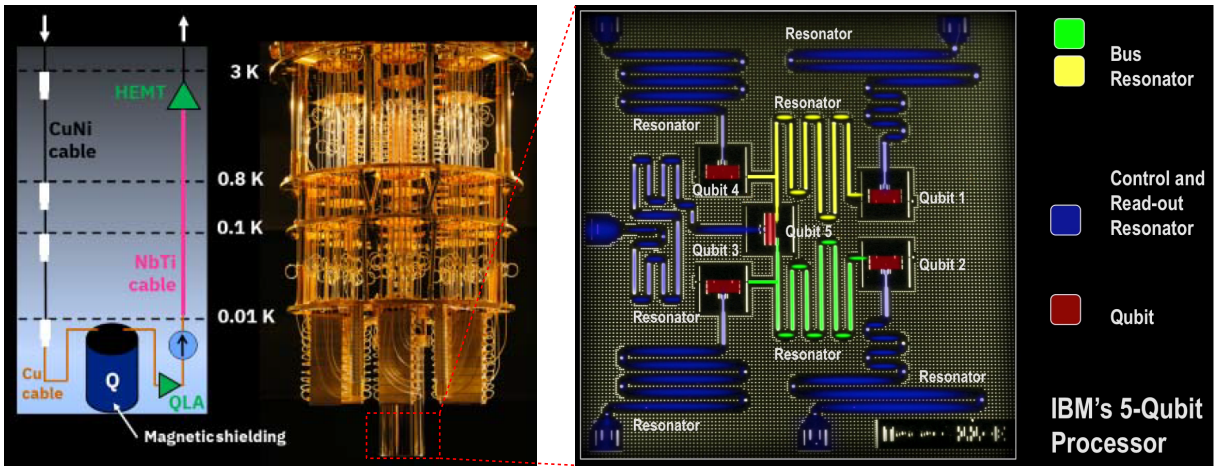


Figura 35 – Computador quântico da IBM. O "abajur" dourado do lado esquerdo é um refrigerador de diluição que resfria o sistema até 15 mK; o chip quântico se encontra na ponta desse dispositivo e é operado através dos pulsos que percorrem os cabos. O lado direito mostra um exemplo de chip quântico da IBM com 5 qubits transmon. Figuras adaptadas de [42].

4 Tomografia de estados quânticos

Um estado quântico desconhecido pode ser estimado (ou reconstruído) através de um processo conhecido como *tomografia de estados quânticos*. Neste capítulo, nós vamos discutir os elementos básicos deste processo, considerando sistemas de n qubits. Primeiramente, vamos generalizar a descrição de estados quânticos através do formalismo de operador densidade. Em seguida, descreveremos um método tomográfico baseado em medições de Pauli para a reconstrução do operador densidade. Finalmente, discutiremos sucintamente a tomografia de estados puros, que é o tipo de tomografia que vamos estudar neste trabalho.

4.1 O operador densidade

No capítulo 2, apresentamos os estados quânticos como vetores de norma 1, representados na notação de Dirac por um ket $|\psi\rangle$ ($\||\psi\rangle\| = \langle\psi|\psi\rangle = 1$). Na mecânica quântica estes são os chamados estados puros. Agora, suponha que um sistema quântico seja preparado aleatoriamente em um de N estados puros $|\psi_j\rangle$, com probabilidade p_j ($\sum_j p_j = 1$). A menos que $p_j = \delta_{ij}$, o estado deste sistema não poderá ser descrito por um vetor. Neste caso, o estado será descrito por um operador, conhecido como operador densidade ou matriz densidade [2, 7, 8]:

$$\hat{\rho} = \sum_{j=1}^{N} p_j |\psi_j\rangle\langle\psi_j|, \tag{4.1}$$

onde $|\psi_j\rangle\langle\psi_j|$ é o operador de projeção sobre o estado $|\psi_j\rangle$. Em contraste com os estados puros, estados na forma da equação (4.1) são chamados de mistos. Como exemplo, considere um qubit preparado em um dos estados da base computacional $\{|0\rangle, |1\rangle\}$ com probabilidade $p_0 = p_1 = 1/2$. O estado deste qubit será $\hat{\rho} = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|) = \frac{1}{2}\mathbf{I}$, que representa um estado maximamente misto.

Além de descrever *ensembles* de estados puros $\{p_j, |\psi_j\rangle\}$, o operador densidade pode descrever também partes de um sistema composto. Considere um sistema bipartido no estado puro $|\psi_{AB}\rangle$. O estado de cada subsistema será dado por

$$\hat{\rho}_A = \mathrm{Tr}_B|\psi_{AB}\rangle\langle\psi_{AB}| \quad \text{e} \quad \hat{\rho}_B = \mathrm{Tr}_A|\psi_{AB}\rangle\langle\psi_{AB}|, \tag{4.2}$$

onde $\mathrm{Tr}_{A(B)}$ representa o traço parcial sobre o sistema A (B). Como exemplo, dado um sistema de 2 qubits no estado de Bell $|\psi_+\rangle$ (equação (2.6)), o estado de cada qubit será $\hat{\rho}_A = \hat{\rho}_B = \frac{1}{2}\mathbf{I}$.

4.1.1 Propriedades do operador densidade

A partir da equação (4.1) é fácil verificar que o operador densidade satisfaz as seguintes propriedades:

- **Hermiticidade**:
$$\hat{\rho} = \hat{\rho}^{\dagger}. \tag{4.3}$$
- **Positividade**:
$$\langle\phi|\hat{\rho}|\phi\rangle \geq 0, \ \ \forall\, |\phi\rangle. \tag{4.4}$$
- **Normalização**:
$$\mathrm{Tr}(\hat{\rho}) = 1. \tag{4.5}$$

Também é fácil verificar que se um operador densidade $\hat{\rho}$ descreve um estado puro, então $\hat{\rho}^2 = \hat{\rho}$; caso contrário, teremos $\hat{\rho}^2 \neq \hat{\rho}$. Assim, uma medida de quão puro (ou quão misto) é um estado quântico pode ser definida como $\mathrm{Tr}(\hat{\rho}^2)$. Esta medida é chamada de *pureza* e satisfaz
$$\frac{1}{d} \leq \mathrm{Tr}(\hat{\rho}^2) \leq 1, \tag{4.6}$$
onde d é a dimensão do espaço de Hilbert associado ao sistema quântico. Um estado com $\mathrm{Tr}(\hat{\rho}^2) = 1$ é puro, enquanto um estado com $\mathrm{Tr}(\hat{\rho}^2) < 1$ é misto; para o limite inferior da pureza, $\mathrm{Tr}(\hat{\rho}^2) = 1/d$, o estado é maximamente misto e dado por[1] $\frac{1}{d}\mathbf{I}_d$, como vimos nos exemplos acima para um qubit.

4.1.2 O operador densidade de 1 qubit

Uma matriz 2×2 arbitrária pode sempre ser escrita como uma combinação linear das matrizes de Pauli (equações (2.11), (2.13) e (2.15)) juntamente com a identidade. Portanto, o conjunto $\mathcal{S}_1 = \{\mathbf{I}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$ forma uma base para o espaço das matrizes 2×2. Nesta base, o operador densidade de um qubit será escrito como
$$\hat{\rho} = \frac{1}{2}\left(\mathbf{I} + r_x\mathbf{X} + r_y\mathbf{Y} + r_z\mathbf{Z}\right), \tag{4.7}$$
onde r_x, r_y e r_z são reais e definem um vetor $\vec{r} = (r_x, r_y, r_z)$ que satisfaz $|\vec{r}| \leq 1$. Com isto, podemos verificar que as propriedades de hermiticidade, positividade e normalização dadas pelas equações (4.3)–(4.5) são satisfeitas; em particular, a postividade de $\hat{\rho}$ é garantida por $\det\hat{\rho} = (1 - |\vec{r}|^2)/4 \geq 0$.

[1] O operador $\mathbf{I}_d$ representa a identidade em um espaço d-dimensional.

O vetor $\vec{r}$ é conhecido como vetor de Bloch[2] e cada uma de suas componentes é dada pelo valor esperado do operador de Pauli correspondente; por exemplo, pode-se verificar que $\langle \mathbf{X} \rangle \equiv \mathrm{Tr}(\hat{\rho}\mathbf{X}) = r_x$. Assim, podemos reescrever o operador densidade de um qubit como

$$\hat{\rho} = \frac{1}{2}\left(\mathbf{I} + \langle \mathbf{X} \rangle \mathbf{X} + \langle \mathbf{Y} \rangle \mathbf{Y} + \langle \mathbf{Z} \rangle \mathbf{Z}\right). \tag{4.8}$$

4.1.3 O operador densidade de n qubits

Para um sistema de n qubits, podemos generalizar os resultados anteriores obtidos para um qubit. O operador densidade deste sistema, dado por uma matriz $2^n \times 2^n$, pode ser expandido em uma base formada pelo produto tensorial da base $\mathcal{S}_1$ associada a cada qubit, ou seja, $\mathcal{S}_n = \{\mathbf{I}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\}^{\otimes n}$. Como exemplo, para 2 qubits temos

$$\begin{aligned}\mathcal{S}_2 = \{&\mathbf{I}\otimes\mathbf{I}, \mathbf{I}\otimes\mathbf{X}, \mathbf{I}\otimes\mathbf{Y}, \mathbf{I}\otimes\mathbf{Z}, \mathbf{X}\otimes\mathbf{I}, \mathbf{X}\otimes\mathbf{X}, \mathbf{X}\otimes\mathbf{Y}, \mathbf{X}\otimes\mathbf{Z},\\ &\mathbf{Y}\otimes\mathbf{I}, \mathbf{Y}\otimes\mathbf{X}, \mathbf{Y}\otimes\mathbf{Y}, \mathbf{Y}\otimes\mathbf{Z}, \mathbf{Z}\otimes\mathbf{I}, \mathbf{Z}\otimes\mathbf{X}, \mathbf{Z}\otimes\mathbf{Y}, \mathbf{Z}\otimes\mathbf{Z}\}.\end{aligned} \tag{4.9}$$

Para simplificar a notação, vamos definir cada um dos 4^n elementos de $\mathcal{S}_n$ como $\mathbf{\Gamma}_j$. Desta forma, $\mathcal{S}_n = \{\mathbf{\Gamma}_j\}_{j=1}^{4^n}$, onde $\mathbf{\Gamma}_1 = \mathbf{I}^{\otimes n}$ e $\mathbf{\Gamma}_{4^n} = \mathbf{Z}^{\otimes n}$. Generalizando a equação (4.8), podemos escrever o operador densidade de n qubits como

$$\hat{\rho} = \frac{1}{2^n}\sum_{j=1}^{4^n} \langle \mathbf{\Gamma}_j \rangle \mathbf{\Gamma}_j, \tag{4.10}$$

onde $\langle \mathbf{\Gamma}_j \rangle = \mathrm{Tr}(\hat{\rho}\mathbf{\Gamma}_j)$.

4.2 Tomografia de estados quânticos de n qubits

Quando falamos de tomografia a ideia que vem à mente é da técnica usada na medicina para realizar diagnósticos e planejamento de tratamento de enfermidades a partir de imagens detalhadas de uma região do corpo humano. Nesta técnica, uma fonte de raios X realiza uma varredura na região de interesse e produz um conjunto de imagens bidimensionais que são combinadas para formar uma imagem mais detalhada da região.

Para estabelecer uma analogia mais direta com a tomografia quântica, vamos considerar o problema de inferir a forma de um objeto a partir das sombras que ele projeta quando iluminado em diferentes ângulos. O exemplo ilustrado na figura 36 mostra um objeto iluminado em três direções perpendiculares definidas pelos eixos x, y e z. Neste caso, as três sombras produzidas nos permitem inferir perfeitamente a forma tridimensional do

[2] Em coordenadas esféricas temos $\vec{r} = (r\sin\theta\cos\phi, r\sin\theta\sin\phi, r\cos\theta)$, onde $r = |\vec{r}|$. Este vetor determina o ponto na esfera de Bloch (figura 3) que corresponde a um estado arbitrário de 1 qubit: se $r = 1$, o estado é puro e localiza-se na superfície da esfera, como vimos no capítulo 2; se $r < 1$, o estado é misto e localiza-se no interior da esfera. Para $r = 0$, o estado é maximamente misto, representado pelo ponto no centro da esfera.

objeto. Observe que se utilizássemos somente uma ou duas das sombras, a inferência seria imperfeita. Por outro lado, se o objeto tivesse uma forma mais complexa, seria necessário realizar iluminações em um número maior de ângulos.

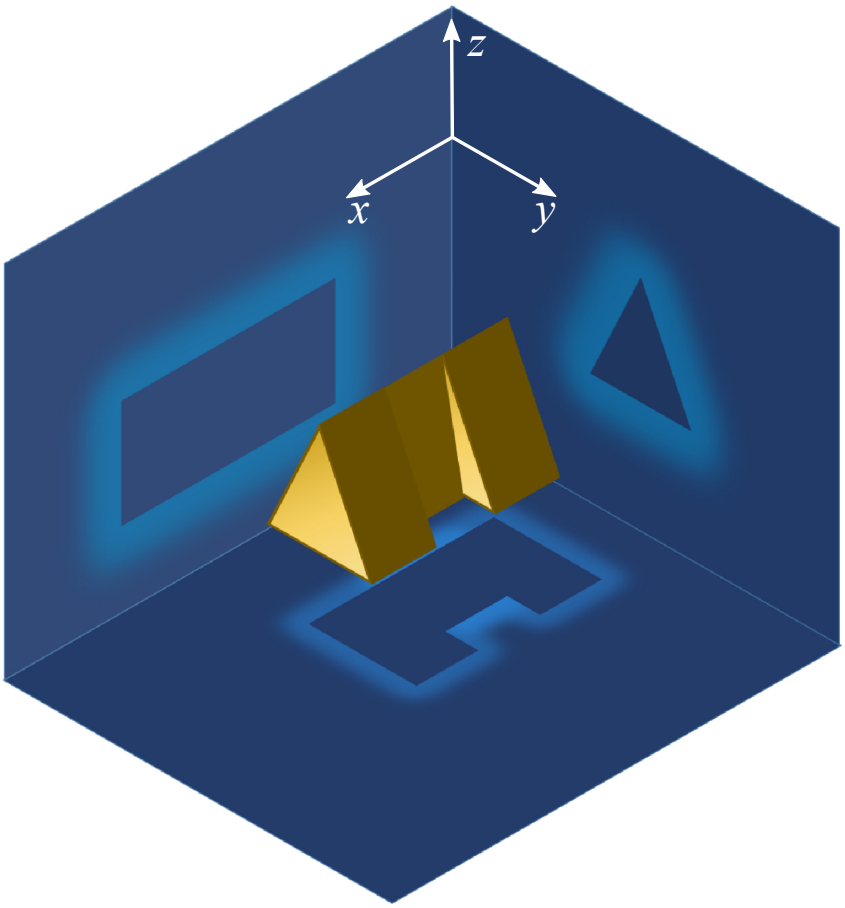


Figura 36 – Objeto caracterizado com a informação das suas sombras produzidas pela iluminação nas direções x, y e z. Imagem adaptada de [43].

Na analogia com a tomografia de estados, o objeto que queremos inferir é um operador densidade desconhecido; a iluminação em diferentes ângulos e as sombras produzidas correspondem à medição representada por um conjunto de operadores e aos resultados gerados, respectivamente. O estado pode ser determinado de forma inequívoca, desde que o conjunto de operadores utilizados seja *informacionalmente completo*, isto é, eles devem formar uma base de operadores atuando no espaço de Hilbert do sistema.[3] Um ponto crucial da tomografia de estados é que ela deve ser realizada sobre um *ensemble* de cópias identicamente preparadas do sistema quântico; como a medição é um processo que modifica o estado (se ele não é um autoestado do observável medido), não seria possível realizar tomografia medindo apenas uma cópia do sistema.

No contexto da computação quântica, a tomografia de estados pode ser útil em

[3] Vimos exemplos disso na seção anterior com os operadores de Pauli e a identidade para o espaço de um qubit e o produto tensorial desta base para o espaço de n qubits.

diversas situações. Por exemplo, podemos usá-la para reconstruir o estado de entrada e caracterizar se ele está adequado ou não ao algoritmo que se quer implementar; podemos também reconstruir o estado resultante da operação implementada no circuito e caracterizar se sua evolução foi de acordo com o esperado. A tomografia de estados é útil também para reconstruir a própria operação implementada no circuito, o que é conhecido como tomografia de processos [2]. Podemos utilizá-la para caracterizar desde portas de um e dois qubits até operações globais de algoritmos mais complexos.

4.2.1 Tomografia com medições de Pauli

A tomografia baseada em medições de Pauli para reconstrução de estados de n qubits parte da decomposição do operador densidade mostrada na equação (4.10). Os coeficientes da decomposição são determinados por medições nas bases associadas aos operadores $\mathbf{\Gamma}_j$, os quais são produtos tensoriais dos operadores de Pauli e identidade. Ao determinarmos esses coeficientes, reconstruímos o operador densidade do sistema [44]. Do total de 4^n bases, é necessário medir em apenas 3^n delas, que são as bases cujo operador $\mathbf{\Gamma}_j$ não inclui uma ou mais identidades no produto tensorial. Para entender isso, vamos escrever a decomposição espectral dos operadores de Pauli:

$$\mathbf{X} = \boldsymbol{\pi}_x^+ - \boldsymbol{\pi}_x^-, \tag{4.11a}$$

$$\mathbf{Y} = \boldsymbol{\pi}_y^+ - \boldsymbol{\pi}_y^-, \tag{4.11b}$$

$$\mathbf{Z} = \boldsymbol{\pi}_z^+ - \boldsymbol{\pi}_z^-, \tag{4.11c}$$

onde $\boldsymbol{\pi}_\xi^\pm$ ($\xi = x, y, z$) é o projetor associado ao autovalor ± 1. Cada elemento $\mathbf{\Gamma}_j$ será dado por uma combinação de 2^n projetores e o valor esperado $\langle \mathbf{\Gamma}_j \rangle$ será obtido determinando-se as probabilidades para cada projetor. Como $\mathbf{I} = \boldsymbol{\pi}_\xi^+ + \boldsymbol{\pi}_\xi^-$ para qualquer ξ, as probabilidades para os elementos $\mathbf{\Gamma}_j$ que incluem a identidade serão redundantes. Por exemplo, considere $n = 2$ e $\mathbf{\Gamma}_8 = \mathbf{X} \otimes \mathbf{Z} = \boldsymbol{\pi}_x^+ \otimes \boldsymbol{\pi}_z^+ - \boldsymbol{\pi}_x^+ \otimes \boldsymbol{\pi}_z^- - \boldsymbol{\pi}_x^- \otimes \boldsymbol{\pi}_z^+ + \boldsymbol{\pi}_x^- \otimes \boldsymbol{\pi}_z^-$; neste caso, $\langle \mathbf{\Gamma}_8 \rangle = p_{xz}^{++} - p_{xz}^{+-} - p_{xz}^{-+} + p_{xz}^{--}$, onde $p_{xz}^{ab} = \mathrm{Tr}[\hat{\rho}(\boldsymbol{\pi}_x^a \otimes \boldsymbol{\pi}_z^b)]$, $\langle \mathbf{\Gamma}_4 \rangle = \langle \mathbf{I} \otimes \mathbf{Z} \rangle = p_{xz}^{++} - p_{xz}^{+-} + p_{xz}^{-+} - p_{xz}^{--}$ e $\langle \mathbf{\Gamma}_5 \rangle = \langle \mathbf{X} \otimes \mathbf{I} \rangle = p_{xz}^{++} + p_{xz}^{+-} - p_{xz}^{-+} - p_{xz}^{--}$, mostrando a redundância das probabilidades.

O método tomográfico descrito acima utiliza 6^n probabilidades (2^n probabilidades para cada uma das 3^n bases de medição) para reconstruir o estado. Este é um número maior que os $4^n - 1$ números reais que caracterizam o operador densidade de um sistema de n qubits.[4] Por esta razão, o conjunto das 3^n medições de Pauli para a tomografia é chamado de conjunto supercompleto.

No caso idealizado de um número infinito de cópias identicamente preparadas, as probabilidades obtidas no processo tomográfico seriam exatas e o estado seria reconstruído de forma perfeita. Porém, em um cenário realístico, a tomografia é realizada sobre um

[4] Consequência das propriedades de hermiticidade e normalização de um operador densidade.

número finito de cópias do sistema. Neste caso, as probabilidades extraídas das medições[5] apresentarão flutuações em relação aos valores exatos e essas flutuações serão maiores quanto menor for o número de cópias utilizado. Em geral, o operador reconstruído neste processo apresentará autovalores negativos (ou seja, não será positivo) e, portanto, não representará um estado físico. Diversos algoritmos de otimização foram desenvolvidos para se extrair um operador densidade compatível com os dados experimentais obtidos [44, 45]. No entanto, como o aumento do número de qubits leva a um aumento exponencial do número de medições e, consequentemente, da quantidade de dados a serem processados, o custo computacional da tarefa se torna cada vez maior. Atualmente, os algoritmos mais rápidos para esta tarefa consomem da ordem de 10^4 s ($\approx$ 2,8 horas) para estimar estados de 10 qubits [46].

Circuitos tomográficos com medições de Pauli

Em um computador quântico, a tomografia de estados de n qubits via medições de Pauli é implementada por um conjunto de 3^n circuitos, cada um correspondendo à medição em uma das bases descritas anteriormente. No caso de um qubit, os 3 circuitos são mostrados na figura 12. Para dois qubits, os 9 circuitos tomográficos são mostrados na figura 37 juntamente com os valores esperados $\langle \mathbf{\Gamma}_j \rangle$ que são estimados em cada um.

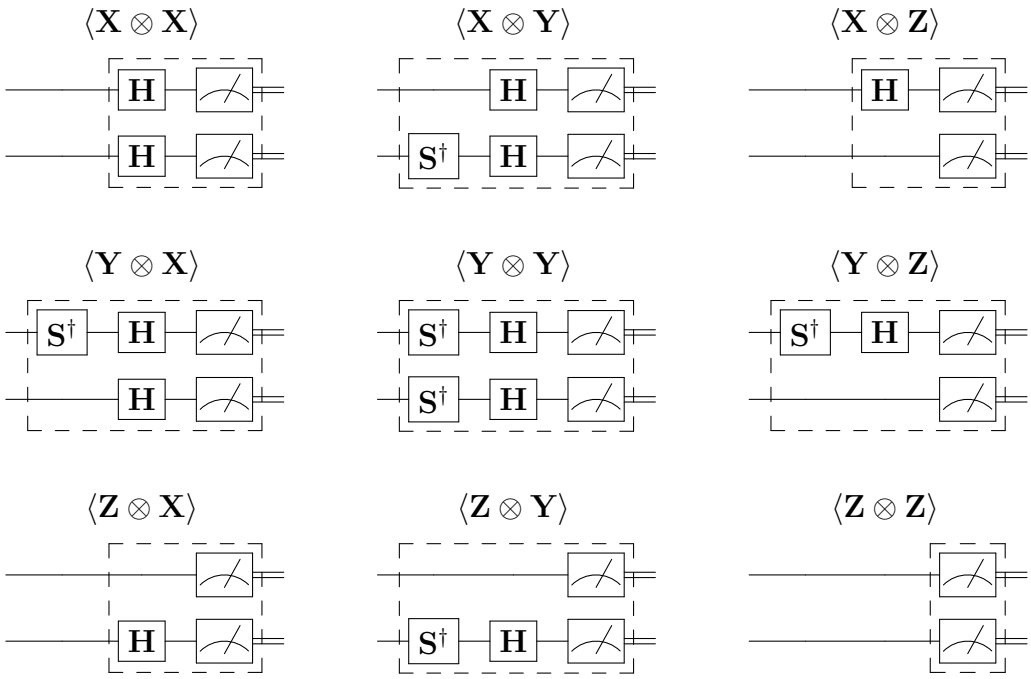


Figura 37 – Circuitos para tomografia de estados de 2 qubits via medições de Pauli.

Na seção 2.5, discutimos as propriedades que caracterizam o "tamanho" de um circuito e, de certa maneira, sua complexidade para operar em dispositivos ruidosos. Deste

[5] Na realidade, temos acesso às frequências normalizadas que são dadas pelo número de contagens de uma dada projeção dividido pelo número total de contagens da medição projetiva implementada.

ponto de vista, os circuitos da tomografia com medições de Pauli podem ser considerados bastante simples e, a príncipio, seriam menos afetados por ruído. Como as medições são realizadas em bases fatoráveis (veja seção 2.4.3), todos eles apresentam o número de fatores tensoriais $T = n$ e profundidade[6] que varia de $P = 1$ a $P = 3$. Porém, o aumento exponencial do número de circuitos necessários para a tomografia pode reduzir o impacto dessa baixa complexidade, além do alto custo experimental e computacional já mencionados.

4.2.2 Tomografia de estados puros

O custo do processo tomográfico aumenta com a dimensão do espaço de Hilbert do sistema, d; no caso de n qubits, onde $d = 2^n$, vimos que o operador densidade é caracterizado por $4^n - 1$ números reais. Porém, é possível reduzir esse custo quando há informação prévia sobre o estado a ser reconstruído. Por exemplo, vamos supor que se saiba de antemão que uma fonte prepara estados *puros* de n qubits (equação (2.3)). Neste caso, o estado será caracterizado por $2^{n+1} - 2$ números reais,[7] uma quantidade muito menor que a de um operador densidade. A contagem de parâmetros em cada caso leva à conclusão natural que a reconstrução de estados puros será um processo menos custoso. De fato, os autores de [47] mostram de forma rigorosa que, para estados puros, um conjunto de operadores informacionalmente completo terá, no mínimo, $2d$ elementos, em contraste com o mínimo de d^2 elementos para reconstruir um operador densidade.

Uma questão importante quando se lida com estados quânticos puros é que, na prática, eles são apenas uma idealização; só seria possível um estado puro de fato se o sistema quântico que ele representa estivesse completamente isolado de interações com o ambiente. Porém, em diversas plataformas experimentais uma fonte pode preparar estados aproximadamente puros e esta aproximação justifica o desenvolvimento de métodos para reduzir a complexidade da tomografia. Nos últimos anos, vários métodos foram propostos e demonstrados experimentalmente [6, 48–52], entre os quais vamos destacar dois daqueles que discutem de forma explícita a reconstrução de estados puros de n qubits [6, 52]. Nesta subseção, descreveremos sucintamente o método desenvolvido por Pereira, Zambrano e Delgado [52]; no próximo capítulo, descreveremos em detalhes o método de Fernandes e Neves [6], que é aquele que vamos aplicar para a tomografia de n qubits em um computador quântico.

[6] Na prática, a profundidade pode ser ligeiramente maior dependendo do conjunto de portas-base do computador quântico. Como vimos na seção 3.1, em termos das portas-base da IBM os circuitos tomográficos deste método teriam profundidade máxima de $P = 4$ (veja figura 21).

[7] Um estado puro em um espaço d-dimensional é especificado por d amplitudes complexas, que correspondem a $2d$ números reais. No entanto, devido à normalização e à irrelevância da fase global, temos que $2d - 2$ números reais independentes são suficientes para caracterizar estados puros.

O método de Pereira, Zambrano e Delgado (PZD) [52]

O método tomográfico apresentado em [52], que vamos chamar de PZD, utiliza medições projetivas em $mn + 1$ bases fatoráveis (incluindo a base computacional) ou m bases não-fatoráveis mais a base computacional, para um número inteiro $m \geq 2$. No primeiro caso, o número de bases aumenta linearmente com o número de qubits, enquanto no segundo caso o número de bases é fixo para qualquer n; em ambos os casos, o método se mostra bastante vantajoso em relação a uma tomografia cujo número de medições aumenta exponencialmente. As figuras 38 e 39 mostram os circuitos tomográficos correspondentes para 2 qubits.

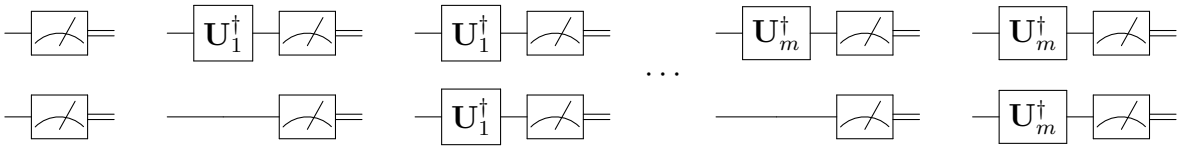


Figura 38 – Circuitos para tomografia de estados puros de 2 qubits via medições em $mn+1$ bases fatoráveis ($m \geq 2$); as portas $\{\mathbf{U}_j\}_{j=1}^m$ são escolhidas pelo usuário [52].

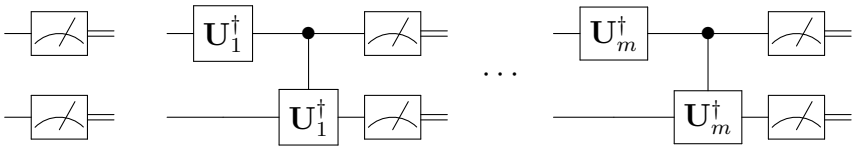


Figura 39 – Circuitos para tomografia de estados puros de 2 qubits via medições na base computacional mais $m \geq 2$ bases não-fatoráveis; as portas $\{\mathbf{U}_j\}_{j=1}^m$ são escolhidas pelo usuário [52].

A medição na base computacional permite a determinação dos módulos dos coeficientes α_j na equação (2.3). As medições nas mn bases fatoráveis ou nas m bases não-fatoráveis são utilizadas para determinar as fases relativas ($\arg \alpha_j$) através da resolução iterativa de sistemas de equações lineares. A resolução desses sistemas de equações será bem sucedida com uma escolha adequada de quantas e quais bases serão utilizadas, o que é definido por m e pelas unitárias $\{\mathbf{U}_j\}_{j=1}^m$ (figuras 38 e 39), respectivamente.

O método proposto foi simulado numericamente e testado experimentalmente nos dispositivos da IBM. Em ambos os casos, os testes foram realizados com *ensembles* de $2^{13} = 8192$ qubits, ou seja, cada medição foi repetida 2^{13} vezes, que era o limite de *shots* nos dispositivos da IBM na época que o trabalho foi realizado. Tanto nas simulações quanto nos experimentos, as reconstruções por medições em bases fatoráveis apresentaram melhores resultados que aquelas realizadas por medições em bases não-fatoráveis. De acordo com os autores, dois fatores podem explicar isto: primeiro, o tamanho total do *ensemble* é da ordem de $mn \times 2^{13}$ e $m \times 2^{13}$, respectivamente, o que favorece o processo com bases fatoráveis; segundo, a necessidade de se utilizar portas **CNOT** nas bases não-fatoráveis

aumentam o ruído no processo, como vimos no capítulo anterior. Os resultados mostraram também que, a medida que o número de bases de medição aumenta ($m > 2$), há um aumento na fidelidade das reconstruções para os dois tipos de bases. Finalmente, em termos computacionais o método se mostrou bastante rápido: simulações com 20 qubits consumiram aproximadamente 3 minutos para estimar o estado.

Nos capítulos 6 e 7, vamos discutir com mais detalhes alguns dos resultados deste método comparando-os com os nossos resultados numéricos e experimentais.

5 Pticografia quântica

Neste capítulo, vamos descrever o método de reconstrução de estados puros proposto por Fernandes e Neves [6], baseado em uma técnica de imageamento computacional conhecida como pticografia. Iniciamos com uma breve descrição da pticografia clássica e, na sequência, apresentamos a sua versão quântica. Em seguida, discutimos a implementação da pticografia quântica em circuitos quânticos e finalizamos o capítulo comparando o método de reconstrução pticográfica com os métodos discutidos no capítulo anterior.

5.1 O que é pticografia?

Pticografia é uma técnica de imageamento computacional [53, 54] utilizada, principalmente, em microscopia óptica [55] e eletrônica [56]. Nesta técnica, um feixe de radiação coerente se desloca relativamente a um objeto de interesse e ilumina, sequencialmente, várias regiões deste objeto. Ao "varrer" o objeto, as iluminações adjacentes devem possuir algum grau de interseção entre si e cada iluminação gera um padrão de difração que é registrado por um detector. Os padrões obtidos são fornecidos a um algoritmo iterativo que fará uma reconstrução computacional da função-transmissão complexa associada ao objeto. A figura 40 ilustra o processo.

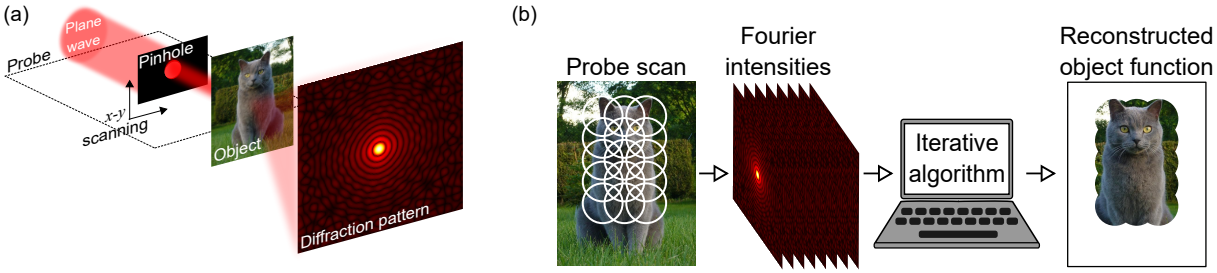


Figura 40 – Pticografia clássica. (a) Esquema de uma configuração típica da pticografia. (b) Visão geral do processo pticográfico. Figura adaptada de [6].

Na formulação do método por Faulkner e Rodenburg em 2004 [53], que inspirou a versão quântica, considerou-se um arranjo simples em que um *pinhole* circular filtra uma onda plana gerando uma iluminação direcionada a uma região do objeto, como mostra a figura 40(a). A luz é difratada por esta região e o padrão de difração é registrado no campo distante, que corresponde à transformada de Fourier do campo incidente no objeto [57]; na realidade, o detector registra o módulo ao quadrado deste padrão, que corresponde à sua distribuição de intensidade, chamada de intensidade de Fourier. O pinhole é deslocado no plano $x - y$ de acordo com o mapeamento mostrado na figura 40(b), gerando uma

intensidade de Fourier para cada posição. A informação das regiões iluminadas (dadas pela posição e forma do pinhole) juntamente com as intensidades de Fourier são as entradas para um algoritmo de recuperação de fase[1] conhecido como PIE (*ptychographic iterative engine*) que vai estimar a função-transmissão do objeto. O algoritmo é iniciado com uma estimativa da função (aleatória, uniforme, etc.) e irá atualizá-la iterativamente, impondo as intensidades de Fourier medidas. Neste processo, as fases da função vão sendo corrigidas devido à imposição dos módulos no domínio conjugado. A estimativa inicial converge para a função do objeto devido à diversidade e redundância dos dados obtidas pelas múltiplas iluminações com interseção parcial.[2]

5.2 Versão quântica da pticografia

5.2.1 O método

Na versão quântica da pticografia [6], o objeto de interesse é um estado quântico puro; no nosso caso, um estado puro de n qubits que é escrito na base computacional de acordo com a equação (2.3), reproduzida abaixo:

$$|\psi\rangle = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle. \tag{5.1}$$

Para determinar este estado, é necessário determinar o conjunto de coeficientes complexos $\{\alpha_j\}_{j=0}^{2^n-1}$ que o caracteriza completamente. Vamos agora descrever como isso é feito através método pticográfico. Em analogia com a pticografia clássica descrita acima, o papel das iluminações com interseção parcial que varrem o objeto é executado por um conjunto de m projetores $\{\mathbf{\Pi}_\ell\}_{\ell=0}^{m-1}$ de *rank* r, onde $1 < r < 2^n$, que deve satisfazer às seguintes condições:

(i) Para qualquer $\mathbf{\Pi}_\ell$ existe pelo menos um $\mathbf{\Pi}_{\ell'}$, tal que $0 < \mathrm{Tr}(\mathbf{\Pi}_\ell \mathbf{\Pi}_{\ell'})/r < 1$. Isto significa que cada projetor deve ter uma interseção parcial com pelo menos um outro projetor do conjunto.

(ii) Todos os níveis do espaço de estados devem ser selecionados pelo menos uma vez.

Estes projetores selecionam partes do estado $|\psi\rangle$, projetando-o sobre subespaços de dimensão r do espaço do sistema, assim como a luz que passa pelo *pinhole* ilumina parte do objeto na pticografia clássica. A condição (i) garante algum grau de interseção entre as partes do estado selecionadas, enquanto a condição (ii) garante que o estado será completamente "varrido". A forma dos projetores será discutida mais adiante. Seguindo a analogia, o

[1] Um algoritmo de recuperação de fase (*phase retrieval*) é aquele que tenta solucionar o chamado problema da fase. Por exemplo, dada uma função complexa $f(u) = |f(u)|e^{i\phi(u)}$ onde $|f(u)|$ é conhecido, a recuperação de fase consiste em determinar $\phi(u)$ a partir da informação disponível.

[2] Se não houvesse interseção, teríamos um problema de recuperação de fase para cada parte iluminada independente das demais, e assim não seria possível obter a relação de fase entre as partes.

estado pós-projeção é submetido a uma operação unitária dada pela transformada quântica de Fourier (QFT[3]) e uma medição na base computacional é realizada, o que equivale à medição da intensidade de Fourier da parte iluminada do objeto no caso clássico.

Dado um *ensemble* de cópias identicamente preparadas descrito pelo estado $|\psi\rangle$, o protocolo pticográfico procede da seguinte forma: primeiro, aplica-se o ℓ-ésimo projetor sobre o *ensemble*, gerando um sub-*ensemble* descrito pelo estado (não-normalizado)

$$|\psi_\ell\rangle = \mathbf{\Pi}_\ell|\psi\rangle. \tag{5.2}$$

Em seguida, aplica-se a QFT neste sub-*ensemble*, obtendo

$$|\tilde{\psi}_\ell\rangle = \mathbf{F}_n|\psi_\ell\rangle = \sum_{j=0}^{2^n-1} \tilde{\alpha}_{j\ell}|j\rangle, \tag{5.3}$$

onde $\mathbf{F}_n$ é a QFT atuando no espaço de n qubits e $\{\tilde{\alpha}_{j\ell}\}_{j=0}^{2^n-1}$ é o conjunto das transformadas de Fourier discretas das amplitudes de $|\psi_\ell\rangle$. Por último, implementa-se uma medição projetiva na base computacional. Este procedimento é repetido para cada um dos m projetores $\mathbf{\Pi}_\ell$ e ao final produz um conjunto de m distribuições de contagens (ou *shots*), que são os dados pticográficos:

$$\{\mathcal{D}_\ell = \{|\tilde{\alpha}_{k\ell}|^2\}_{k=0}^{2^n-1}\}_{\ell=0}^{m-1}, \tag{5.4}$$

onde $|\tilde{\alpha}_{k\ell}|^2 = |\langle k|\mathbf{F}_n\mathbf{\Pi}_\ell|\psi\rangle|^2$. Esses dados, juntamente com o conjunto de projetores, serão as entradas para o algoritmo de reconstrução do estado, que será descrito a seguir. A figura 41 ilustra o processo de pticografia quântica.

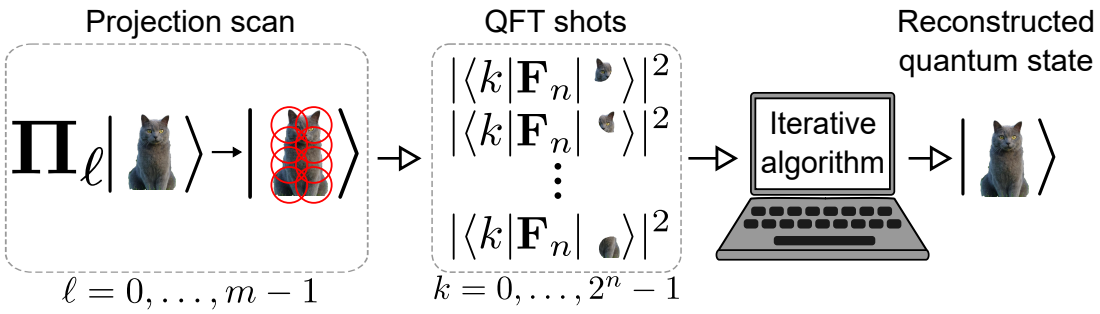


Figura 41 – Pticografia quântica. Figura adaptada de [6].

5.2.2 Algoritmo pticográfico para reconstrução de estados

O algoritmo iterativo que estima o estado $|\psi\rangle$ a partir do conjunto de projetores e dados pticográficos é uma adaptação do PIE [53, 54]. Ele atua levando a estimativa pelas mesmas etapas do processo de medição descrito acima, seguido da imposição dos dados medidos e aplicação de uma regra de atualização. O quadro a seguir detalha os passos do algoritmo.

[3] Do inglês, *quantum Fourier transform*. Veja a seção 5.3.2 para mais detalhes sobre a QFT.

Ptychographic iterative engine (PIE): os passos do algoritmo iterativo

(i) Inicie com uma estimativa aleatória,

$$|\varphi\rangle = \sum_{k=0}^{2^n-1} \gamma_k |k\rangle. \tag{5.5}$$

(ii) Aplique o projetor $\mathbf{\Pi}_\ell$ em $|\varphi\rangle$:

$$|\varphi_\ell\rangle = \mathbf{\Pi}_\ell |\varphi\rangle = \sum_k \gamma_{k\ell} |k\rangle. \tag{5.6}$$

(iii) Aplique a QFT em $|\varphi_\ell\rangle$:

$$|\tilde{\varphi}_\ell\rangle = \mathbf{F}_n |\varphi_\ell\rangle = \sum_{k=0}^{2^n-1} \tilde{\gamma}_{k\ell} |k\rangle. \tag{5.7}$$

(iv) Use o ℓ-ésimo dado pticográfico, $\sqrt{\mathcal{D}_\ell}$, para corrigir os módulos dos coeficientes de $|\tilde{\varphi}_\ell\rangle$, mantendo suas fases:

$$|\widetilde{\varphi}'_\ell\rangle = \sum_{k=0}^{2^n-1} |\tilde{\alpha}_{k\ell}| e^{i \arg \tilde{\gamma}_{k\ell}} |k\rangle. \tag{5.8}$$

(v) Aplique a QFT inversa em $|\widetilde{\varphi}'_\ell\rangle$:

$$|\varphi'_\ell\rangle = \mathbf{F}_n^{-1} |\widetilde{\varphi}'_\ell\rangle. \tag{5.9}$$

(vi) Atualize a estimativa atual do estado de entrada:

$$|\varphi'\rangle = |\varphi\rangle + \beta \mathbf{\Pi}_\ell (|\varphi'_\ell\rangle - |\varphi_\ell\rangle), \tag{5.10}$$

onde β é um parâmetro dentro do intervalo $(0, 2]$; este parâmetro pode ser fixo ou variável e controla o tamanho do passo da atualização, podendo ser ajustado para melhorar a convergência.

(vii) Use a estimativa atualizada como entrada para repetir os passos de (ii) a (vi) com um novo valor de ℓ.

Essa sequência é resumida no diagrama da figura 42. Uma iteração completa do algoritmo, chamada de iteração PIE, consiste em percorrer o *loop* m vezes (passos (ii) a (vii)), onde a cada iteração deste *loop* o projetor e os dados pticográficos correspondentes são usados uma vez para atualizar a estimativa do estado.

Para analisar a convergência do algoritmo, nós vamos usar neste trabalho a distância do traço entre as versões normalizadas das estimativas corrente ($|\varphi_{\text{norm}}\rangle = |\varphi\rangle / \sqrt{\langle\varphi|\varphi\rangle}$) e

atualizada ($|\varphi'_{\text{norm}}\rangle = |\varphi'\rangle/\sqrt{\langle\varphi'|\varphi'\rangle}$) do estado, definida como

$$D(|\varphi_{\text{norm}}\rangle, |\varphi'_{\text{norm}}\rangle) = \sqrt{1 - |\langle\varphi_{\text{norm}}|\varphi'_{\text{norm}}\rangle|^2}. \tag{5.11}$$

Essa distância pode ser calculada a cada iteração do *loop* bem como a cada iteração PIE. A distância do traço é uma medida da distinguibilidade entre dois estados [2] e, com o progresso do algoritmo, ela diminuirá à medida que as estimativas se tornem cada vez mais próximas (ou cada vez menos distinguíveis). Como critério de parada, podemos estabelecer que o algoritmo termine quando D atingir um valor suficientemente pequeno ou após executar um número máximo de iterações PIE (determinado previamente de forma empírica). Ao final, o algoritmo estimará um estado puro que deverá ser normalizado.

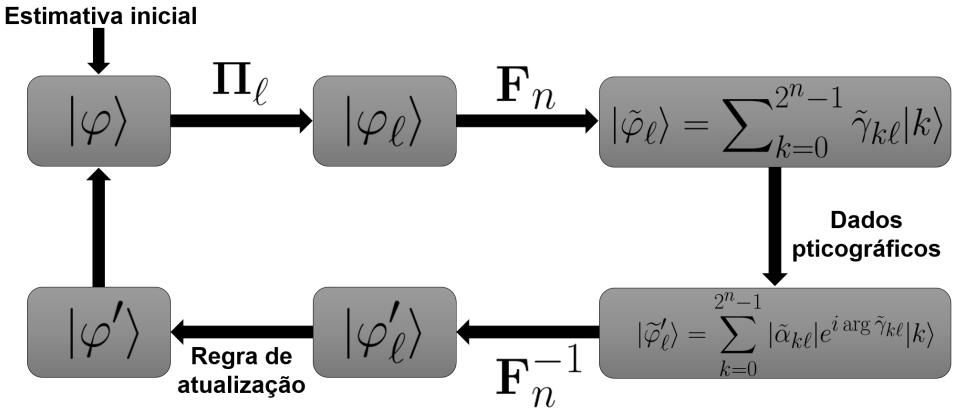


Figura 42 – Diagrama do algoritmo PIE para pticografia quântica.

Para finalizar esta subseção, há alguns pontos importantes a serem comentados:

- Em comparação com os métodos tomográficos discutidos no capítulo anterior, o algoritmo de reconstrução da pticografia quântica é de mais fácil entendimento e implementação. Além disso, veremos no próximo capítulo que ele é muito mais rápido que os algoritmos utilizados em tomografia de estados sem informação prévia.

- O parâmetro β que aparece na etapa de atualização da estimativa é responsável por controlar a convergência do algoritmo. Em geral, nos trabalhos de pticografia clássica que adotam o PIE [53,54], assim como nos trabalhos de pticografia quântica [6,51], os autores utilizaram um valor fixo para β, estimado empiricamente dentro do intervalo $(0, 2]$. No entanto, nós verificamos neste trabalho que, ao variarmos o parâmetro β ao longo das iterações PIE, tanto a convergência do algoritmo quanto a qualidade das reconstruções melhoraram significativamente, como veremos no próximo capítulo.

- O algoritmo PIE proposto por Faulkner e Rodenburg em 2004 [53, 54] passou por inúmeras modificações e melhorias ao longo dos anos [58]. Neste trabalho, o algoritmo de reconstrução utilizado foi aquele adaptado por Fernandes e Neves [6] a partir

dessa versão mais "primitiva" do PIE. Portanto, esta é uma etapa da pticografia quântica que ainda pode ser significativamente melhorada, especialmente para lidar com dados ruidosos gerados pelos dispositivos NISQ. Uma melhoria já foi obtida aqui com a utilização de um β variável.

5.2.3 Projetores $\mathbf{\Pi}_\ell$

Na pticografia quântica existe uma certa liberdade na escolha do conjunto de projetores $\{\mathbf{\Pi}_\ell\}_{\ell=0}^{m-1}$, tanto na forma desses projetores quanto na sua quantidade m. A única exigência ao se construir o conjunto é que as condições (i) e (ii) descritas na subseção 5.2.1 sejam cumpridas. Após a escolha, o conjunto pode ser testado para verificar se é adequado ou não para a pticografia. Para estados de n qubits, Fernandes e Neves [6] propuseram e testaram um conjunto de $m = 6n$ projetores de *rank* $r = 2^{n-1}$ dados por

$$\mathbf{\Pi}_{\xi j}^{\pm} = \boldsymbol{\pi}_{\xi j}^{\pm} \otimes \mathbf{I}^{\otimes n-1}, \tag{5.12}$$

onde o índice ℓ é substituído pelos índices $\xi = x, y, z$ e "$\pm$" que denotam o projetor associado ao autovalor ± 1 do operador de Pauli correspondente (veja equações (4.11)), e $j = 0, \ldots, n-1$ denota o qubit em que ele atua. Estes projetores são fatoráveis, o que favorece sua implementação em dispositivos ruidosos.

Definindo $\boldsymbol{\pi}_{\xi}^{\pm} = |\xi^{\pm}\rangle\langle\xi^{\pm}|$, a aplicação de $\mathbf{\Pi}_{\xi j}^{\pm}$ em um estado de n qubits $|\psi\rangle$ produz o seguinte estado (não-normalizado):

$$\mathbf{\Pi}_{\xi j}^{\pm}|\psi\rangle = |\psi_{\xi j}^{\pm}\rangle = |\xi^{\pm}\rangle_j \otimes |\phi_{\xi j}^{\pm}\rangle_{n-1}, \tag{5.13}$$

onde $|\phi_{\xi j}^{\pm}\rangle_{n-1}$ é o estado dos $n-1$ qubits onde atuaram as identidades. Assim, vemos que o estado do j-ésimo qubit é projetado no autoestado do operador de Pauli correspondente. Na próxima seção, veremos que isto é importante para construir o circuito das projeções intermediárias da pticografia de forma adequada. É importante ressaltar também que cada par de projetores $\mathbf{\Pi}_{\xi j}^{\pm}$ será implementado através de um único circuito, pois se trata de uma medição projetiva de Pauli no j-ésimo qubit (veja seção 2.4.3). Isto levará a um total de $3n$ circuitos pticográficos, como será discutido na última seção do capítulo.

5.3 Circuitos quânticos pticográficos

De maneira geral, um circuito quântico pticográfico para n qubits pode ser representado como na figura 43. Ele contém um registrador clássico (C) de $n+1$ bits, onde serão armazenados os resultados das medições. Primeiramente, o estado a ser reconstruído, $|\psi\rangle$, é preparado com seu circuito específico (veja o apêndice A para maiores detalhes). Na sequência, aplicamos a medição inicial representada pelo par de projetores $\mathbf{\Pi}_{\xi j}^{\pm}$ (equação (5.12)) sobre o j-ésimo qubit; o bit resultante é armazenado em C. Por

último, aplicamos a QFT no estado pós-medição, $|\psi^{\pm}_{\xi j}\rangle$, e realizamos a medição final na base computacional, armazenando os n bits resultantes em C. A seguir, detalhamos cada etapa da construção desses circuitos.

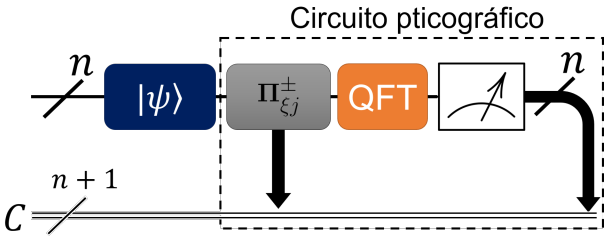


Figura 43 – Circuito quântico pticográfico para n qubits com projeção no j-ésimo qubit; C representa os bits clássicos do registrador que totaliza $n+1$ bits.

A seguir, detalhamos cada etapa da construção dos circuitos quânticos pticográfico.

5.3.1 Circuitos para os projetores $\mathbf{\Pi}^{\pm}_{\xi j}$

Na seção 2.4.3 descrevemos como as medições projetivas de Pauli são implementadas em um circuito quântico onde só há disponibilidade da medição na base computacional (veja figura 12). Em qualquer caso, quando o qubit não é destruído no processo, o seu estado após a medição será sempre $|0\rangle$ ou $|1\rangle$, o que não representa problema quando só há interesse na estatística dos resultados. Porém, como foi visto na seção anterior, para a pticografia será necessário implementar medições de Pauli que projetem o estado do qubit de entrada nos autoestados $\boldsymbol{\pi}^{\pm}_{\xi} = |\xi^{\pm}\rangle\langle\xi^{\pm}|$ do operador de Pauli correspondente, como mostra a equação (5.13). Os circuitos desejados para essas medições estão ilustrados no lado esquerdo da figura 44, onde um estado arbitrário de entrada $\hat{\rho}$ é projetado em um desses autoestados $|\xi^{\pm}\rangle$ na saída. Para obter estes circuitos, devemos acrescentar portas que atuem na base computacional e preparem os autoestados $|\xi^{\pm}\rangle$ adequados, após a medição de Pauli. Com isso, é fácil verificar que os circuitos desejados para as projeções pticográficas terão a forma mostrada no lado direito da figura 44.

5.3.2 Circuito para a transformada quântica de Fourier

A QFT é uma operação importante para diversos algoritmos de computação quântica [2, 33]. Aqui, nós vamos apresentar os resultados principais que nos permitem construir o seu circuito. O passo-a-passo dos cálculos para se chegar a esses resultados pode ser visto em [2, 59].

Como vimos na seção 2.2, podemos escrever a base computacional de um sistema de n qubits usando a representação decimal de um número binário, isto é,

$$|j\rangle = |b_{n-1}\dots b_1 b_0\rangle, \tag{5.14}$$

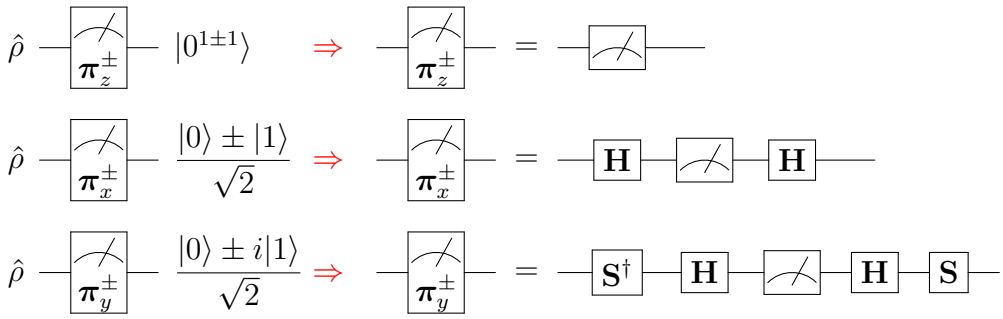


Figura 44 – Circuitos para medições de Pauli que projetam o estado de entrada do qubit no autoestado associado aos autovalores ± 1 do operador medido.

onde $j = 0, \ldots, 2^n - 1$ e $b_k \in \{0, 1\}$ $\forall k$. A QFT no espaço de n qubits é uma operação unitária cuja representação matricial na base computacional é dada por

$$\mathbf{F}_n = \frac{1}{\sqrt{2^n}} \sum_{k,l=0}^{2^n-1} \omega^{kl} |k\rangle\langle l|, \tag{5.15}$$

onde $\omega = \exp(2\pi i/2^n)$; é fácil verificar que $\mathbf{F}_n \mathbf{F}_n^\dagger = \mathbf{F}_n^\dagger \mathbf{F}_n = \mathbf{I}_n$. Aplicando a QFT na base computacional, obtém-se

$$\mathbf{F}_n |j\rangle = \frac{1}{\sqrt{2^n}} \left[|0\rangle + e^{2\pi i 0.b_0}\ |1\rangle\right] \otimes \left[|0\rangle + e^{2\pi i 0.b_1 b_0}\ |1\rangle\right] \otimes \cdots \otimes \left[|0\rangle + e^{2\pi i 0.b_{n-1}\cdots b_1 b_0}\ |1\rangle\right], \tag{5.16}$$

onde $0.b_l b_{l-1} \ldots b_1 b_0 = \sum_{r=1}^{l} b_{r-l}/2^r$ é a expressão de uma fração binária. A partir deste resultado, é possível construir o circuito. Primeiramente, o resultado da operação abaixo sobre o qubit de índice $n-1$ em (5.14) pode ser facilmente verificado

$$\prod_{c=0}^{n-2} [\mathbf{P}_{c,n-1}^{\text{ctrl}}]_{n-c} \mathbf{H}_{n-1} |j\rangle = \frac{1}{\sqrt{2}} \left[|0\rangle + e^{2\pi i 0.b_{n-1}\cdots b_1 b_0} |1\rangle\right] \otimes |b_{n-2} \ldots b_1 b_0\rangle, \tag{5.17}$$

onde $[\mathbf{P}_{c,n-1}^{\text{ctrl}}]_{n-c} \equiv \mathbf{P}_{c,n-1}^{\text{ctrl}}(2\pi/2^{n-c})$ é a porta de fase controlada (equações (2.32)) entre os qubits de índices c e $n-1$. Aplicando um procedimento similar aos demais qubits em ordem decrescente de índices, o resultado final será

$$|j\rangle \to \frac{1}{\sqrt{2^n}} \left[|0\rangle + e^{2\pi i 0.b_{n-1}\cdots b_1 b_0}\ |1\rangle\right] \otimes \cdots \otimes \left[|0\rangle + e^{2\pi i 0.b_1 b_0}\ |1\rangle\right] \otimes \left[|0\rangle + e^{2\pi i 0.b_0}\ |1\rangle\right]. \tag{5.18}$$

Permutando os estados dos qubits $n-1$ e 0, $n-2$ e 1, etc., por meio de portas **SWAP**, esta equação ficará idêntica à (5.16), ou seja, a operação produzirá a QFT. Portanto, o circuito quântico da QFT para n qubits é composto por n portas **H**, $n(n-1)/2$ portas $\mathbf{P}^{\text{ctrl}}$ e $\lfloor n/2 \rfloor$ portas **SWAP**,[4] distribuídas conforme ilustrado na figura 45. A figura 46 mostra o circuito da QFT para $n = 3$.

[4] Vamos ver no próximo capítulo que as **SWAP**s não são necessárias quando a QFT é realizada imediatamente antes de uma medição na base computacional, como é o caso da pticografia quântica.

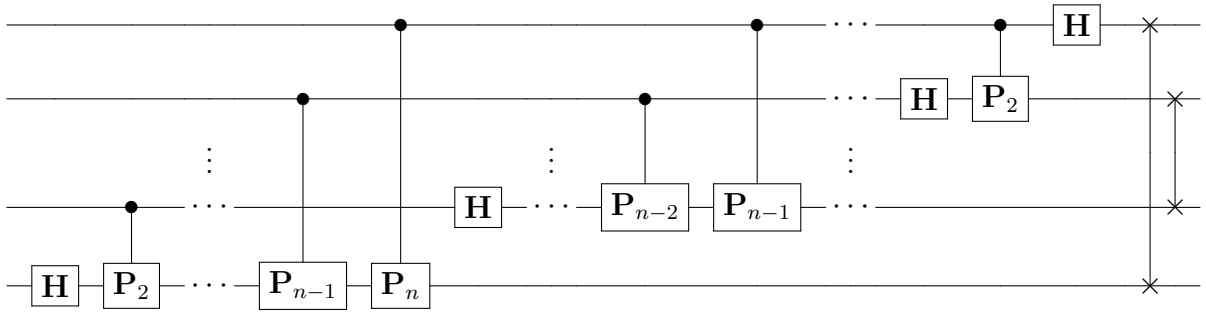


Figura 45 – Circuito para QFT de n qubits; $\mathbf{P}_k \equiv \mathbf{P}(\pi/2^{k-1})$ é a porta de fase dada pela equação (2.25).

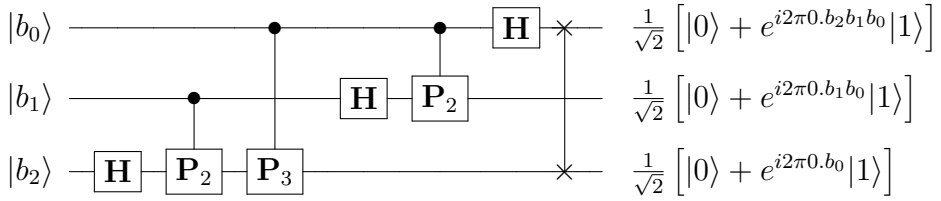


Figura 46 – Circuito para QFT de 3 qubits.

5.3.3 Exemplo: Circuitos pticográficos para 2 qubits

Com todas as etapas para a construção dos circuitos pticográficos apresentadas, podemos mostrar um conjunto completo desses circuitos considerando o caso mais simples que é um sistema de 2 qubits. A figura 47 mostra os 6 circuitos para implementação das medições pticográficas descritas na seção 5.2 em um computador quântico. As caixas tracejadas indicam a medição na base $\{\mathbf{F}_2^\dagger|b_1b_0\rangle\}$ implementada após as projeções $\mathbf{\Pi}_{\xi j}^{\pm}$; acima das caixas estão indicados os dados pticográficos coletados de cada circuito.

5.4 Comparativo com outros métodos tomográficos

Número de circuitos

Como foi visto na seção 5.2.1, a pticografia quântica é implementada usando um conjunto de m projetores $\{\mathbf{\Pi}_\ell\}$ definidos pelo usuário, seguido de uma medição projetiva em uma única base. Portanto, no caso de um sistema de n qubits, o número de circuitos pticográficos será determinado pelo conjunto de projetores utilizado. Com o conjunto $\{\mathbf{\Pi}_{\xi j}^{\pm}\}$ (equação (5.12)) proposto em [6], vimos que serão necessários $3n$ circuitos, um crescimento linear com o número de qubits, em contraste com o crescimento exponencial (3^n) da tomografia com medições de Pauli, discutida na seção 4.2.1. O gráfico da figura 48 compara os dois métodos pelo critério do número de circuitos: é nítida a grande vantagem da pticografia em relação ao custo experimental, que também refletirá em um menor

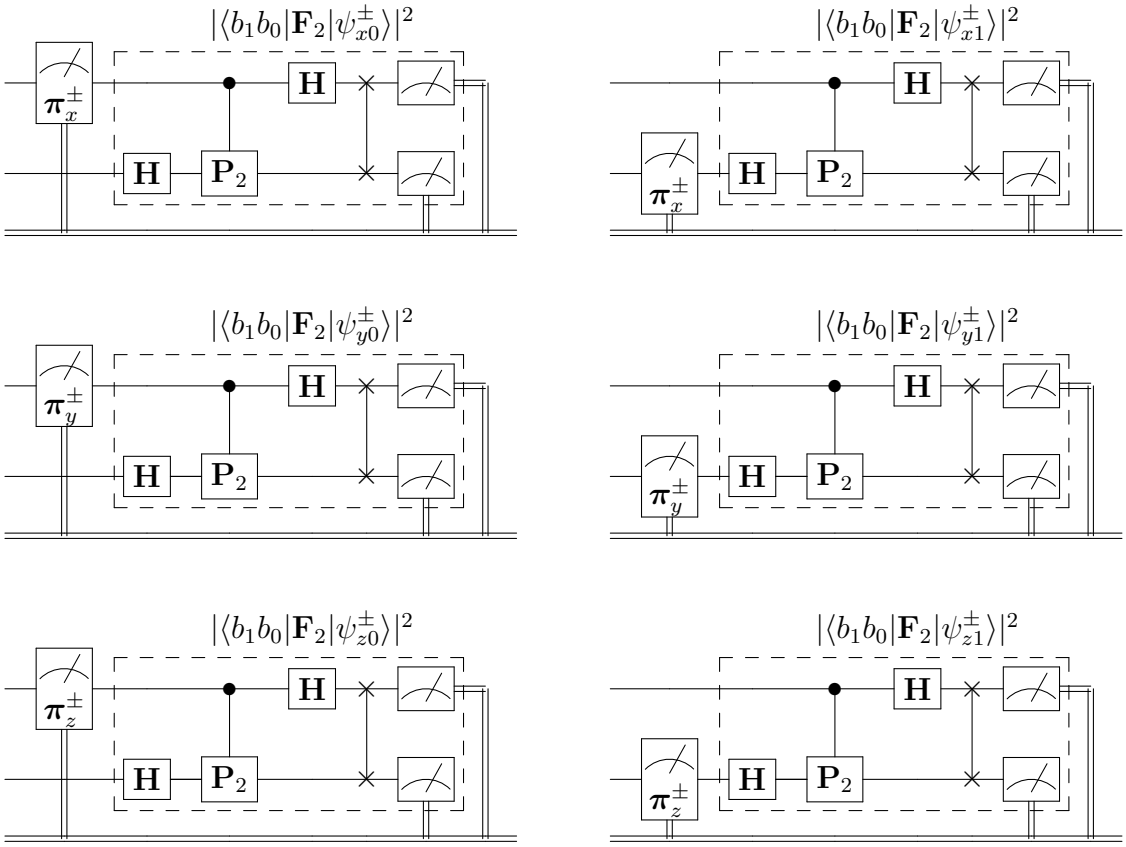


Figura 47 – Circuitos para reconstrução pticográfica de estados puros de 2 qubits.

custo computacional para o pós-processamento, como veremos no próximo capítulo. Por este critério, o método pticográfico se equivale ao método PZD com medições em bases fatoráveis [52], onde o número de circuitos também aumenta linearmente com n (veja seção 4.2.2).

A escolha de outros conjuntos $\{\mathbf{\Pi}_\ell\}$ adequados para a pticografia pode levar a um número menor ou maior de circuitos; no segundo caso, uma escolha inteligente certamente não levará a um aumento significativo. Este é um tópico aberto para futuras investigações.

Complexidade dos circuitos

No capítulo anterior vimos que, tanto a tomografia com medições de Pauli quanto o método PZD com medições em bases fatoráveis são realizados em circuitos de baixa complexidade: eles apresentam baixa profundidade e um número máximo de fatores tensoriais $T = n$, o que favorece o seu desempenho em processadores NISQ. Os circuitos de cada método para reconstrução de estados de 2 qubits são mostrados nas figuras 37 e 38, respectivamente.

No caso da pticografia, a medição ocorre em duas etapas: a projeção intermediária, seguida de uma medição projetiva final. A projeção intermediária com os projetores $\mathbf{\Pi}^{\pm}_{\xi j}$ da equação (5.12) geram circuitos de baixa complexidade, pois incluem apenas portas de um

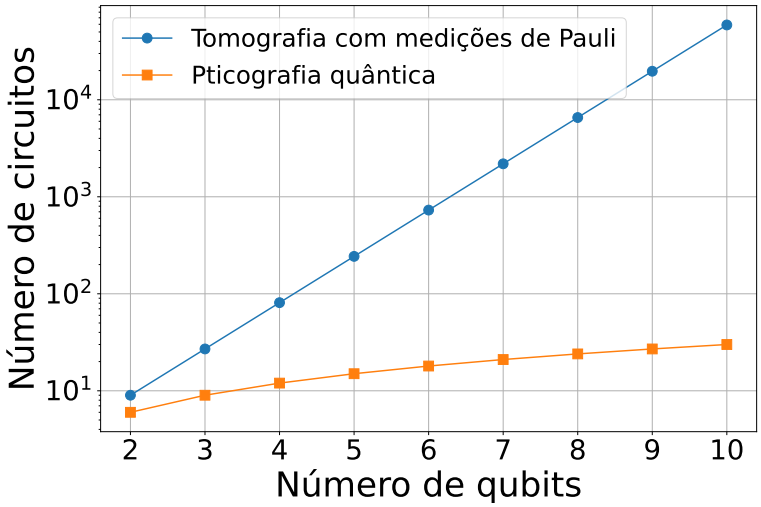


Figura 48 – Número de circuitos para a tomografia com medições de Pauli e a pitcografia quântica usando o conjunto de projetores $\{\mathbf{\Pi}^{\pm}_{\xi j}\}$ (equação (5.12)).

qubit (veja figura 44). Porém, a medição projetiva final é realizada na base não-fatorável $\{\mathbf{F}_n^\dagger|j\rangle\}_{j=0}^{2^n-1}$, que requer a implementação da QFT. Como se observa nas figuras 45 e 46, o circuito da QFT possui o número mínimo de fatores tensoriais, $T = 1$, e sua profundidade aumenta com o número de qubits (é possível verificar que $P = 2n$). Desconsiderando as **SWAP**s, ele requer um total de $n(n+1)/2$ portas quânticas, sendo n portas de um qubit (**H**) e $n(n-1)/2$ portas de dois qubits ($\mathbf{P}^{\text{ctrl}}$). Portanto, os circuitos pticográficos que utilizam esta base de medição serão bem mais complexos que os circuitos dos métodos mencionados acima. Isto fica nítido comparando a figura 47 com as figuras 37 e 38.

No cenário atual de dispostivos NISQ, este é um ponto desfavorável para a aplicação da pticografia quântica. Porém, assim como existe uma flexibilidade na escolha dos projetores $\{\mathbf{\Pi}_\ell\}$, os autores do método [6] sugerem que há também uma flexibilidade na escolha da base de medição final.[5] Neste trabalho, nós vamos testar esta hipótese escolhendo operações unitárias diferentes da QFT, incluindo versões aproximadas da QFT, que reduzem o número de portas de dois qubits do circuito, e também operações unitárias que geram bases fatoráveis.

[5] Esta sugestão foi motivada pelo fato da pticografia clássica ter evoluído em diferentes direções desde a formulação de Faulkner e Rodenburg em 2004 [53]. Uma dessas, é a utilização de propagadores diferentes da transformada de Fourier [60] para a iluminação do objeto. No análogo quântico, isto corresponderia à utilização de operações unitárias diferentes da QFT para a medição projetiva final.

6 Pticografia em um computador quântico: Simulações

Neste capítulo, apresentaremos os resultados obtidos em nossas simulações do método pticográfico, considerando cenários ideais sem ruído e cenários mais realísticos com ruído. Primeiramente, descrevemos os procedimentos gerais que foram adotados para a implementação das simulações. Nas seções subsequentes, apresentaremos os resultados obtidos: inicialmente, exploramos a reconstrução de estados de 2 qubits e, posteriormente, estados de n qubits. Mostramos como a variação do parâmetro β no algoritmo PIE pode resultar em melhores reconstruções. Além disso, comparamos o método pticográfico com os métodos tomográficos discutidos no capítulo anterior. Finalmente, com o objetivo de buscar operações alternativas à QFT que levem a circuitos menos ruidosos, exploraremos o uso da QFT aproximada e de bases aleatórias fatoráveis para a pticografia.

6.1 Preliminares

6.1.1 Transpilação dos circuitos para pticografia quântica

Em um cenário ideal, a implementação de um método de reconstrução de estados requer que as seguintes condições sejam atendidas: primeiro, o *ensemble* de sistemas quânticos deve ser identicamente preparado, pois o que se quer determinar é o estado que descreve este *ensemble*. Segundo, as condições experimentais não devem se alterar durante o processo; por exemplo, em um computador quântico, uma dada porta deve operar da mesma maneira em circuitos diferentes. Em cenários reais é possível atender estas condições, pelo menos de forma aproximada. Aqui, nós vamos discutir como fazer isso para a pticografia em um computador quântico.

Os circuitos para aplicação do método de reconstrução pticográfica podem ser divididos em três sub-circuitos: (i) preparação do estado $|\psi\rangle$, (ii) projeção intermediária $\mathbf{\Pi}^{\pm}_{\xi j}$ e (iii) QFT seguida da medição final na base computacional, como ilustrado na figura 43. Cada um dos $3n$ circuitos construídos para determinar um estado de n qubits será primeiramente transpilado antes de ser executado. Como vimos na seção 3.1.2, um mesmo circuito de entrada transpilado repetidamente vai gerar diferentes circuitos de saída. Isso quer dizer que se transpilarmos cada um dos $3n$ circuitos para a pticografia de forma global (ou seja $\texttt{transpile}\{\texttt{i} + \texttt{ii} + \texttt{iii}\}$), certamente vários deles terão configurações diferentes: por exemplo, os mapas de acoplamento podem ser diferentes, assim como o circuito de preparação, a QFT de um circuito pode apresentar muito mais portas que a

de outro, etc. Com isso, as condições descritas acima para a implementação do método deixariam de ser atendidas. Para evitar isso, adotamos os seguintes procedimentos:

- Escolhemos o *layout*, ou seja, a forma em que os qubits virtuais serão mapeados nos qubits físicos, usando como critério a máxima conectividade entre eles.
- Fixado o *layout*, construímos cada sub-circuito separadamente.
- Os sub-circuitos (i) e (iii) são transpilados individualmente diversas vezes e selecionamos os circuitos ótimos em cada caso.[1] Como critério de seleção, verificamos os circuitos com menos portas de dois qubits, seguido de menos portas de um qubit.

Com esses procedimentos, os sub-circuitos (i) e (iii) selecionados serão os mesmos para os $3n$ circuitos da pticografia, satisfazendo as condições para a implementação do método. Após concluí-los, os três sub-circuitos são concatenados, formando assim o circuito global para a pticografia. A figura 49 ilustra o processo.

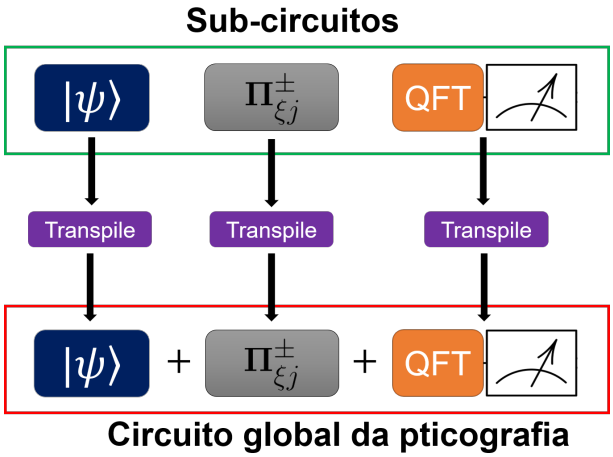


Figura 49 – Transpilação individual dos sub-circuitos seguida da concatenação.

6.1.2 Mitigação de erros

Há diversas formas de tentar lidar com erros nos computadores quânticos. Uma dessas formas é a chamada mitigação de erros, para a qual vários métodos foram propostos recentemente [61–63]. Em geral, os métodos mais robustos são também os mais complexos e, para aplicar algum deles no nosso trabalho, seria necessário um tempo dedicado exclusivamente ao estudo de mitigação de erros. Como isso estava além dos propósitos iniciais deste trabalho, buscamos um método de fácil entendimento e simples implementação para, pelo menos, ilustrar o processo. O método de mitigação de erros que descrevemos a seguir será aplicado nas simulações com ruído e nos experimentos.

[1] Para o sub-circuito (ii) este processo é desnecessário pois ele possui apenas operações sobre um qubit, de forma que uma única transpilação é suficiente.

Mitigando erros de medição

Seja $\mathcal{P}$ um vetor de probabilidades na base computacional resultante de medições repetidas realizadas em um circuito de n qubits. Vamos denotar esse vetor como $\mathcal{P}_{\text{ruído}}$ para o caso de um computador quântico com ruído e $\mathcal{P}_{\text{ideal}}$ para o caso ideal sem ruído. Podemos relacionar esses dois vetores considerando uma matriz de calibração M tal que

$$\mathcal{P}_{\text{ruído}} = M\mathcal{P}_{\text{ideal}}. \tag{6.1}$$

Ao inverter a matriz M teríamos

$$\mathcal{P}_{\text{ideal}} = M^{-1}\mathcal{P}_{\text{ruído}}. \tag{6.2}$$

Portanto, com o conhecimento da matriz de calibração e o cálculo da sua inversa, poderíamos obter um conjunto ideal de probabilidades a partir das probabilidades ruidosas, mitigando o erro de leitura. Obviamente, este é um método bastante simplificado que considera apenas erros na leitura, ignorando todas as demais fontes de ruído no dispositivo. Na prática, ao invés de um $\mathcal{P}_{\text{ideal}}$ o máximo que podemos obter é um $\mathcal{P}_{\text{mitigado}}$.

Para obter a matriz M construímos 2^n circuitos de calibração onde em cada um deles um estado da base computacional $\{|j\rangle\}$ é preparado e medido também na base computacional $\{|j'\rangle\}$. A matriz de calibração $2^n \times 2^n$ será formada pelos vetores coluna das probabilidades condicionais $\mathcal{P}_j = \{p(j'|j)\}_{j'=0}^{2^n-1}$. Observe que na ausência de ruído teríamos $p(j'|j) = \delta_{jj'}$ e M seria a matriz identidade.

O Qiskit possui ferramentas que simplificam o processo de mitigação de erros de leitura.[2] Essas ferramentas geram a matriz de calibração e sua inversa realizando todo o processo descrito acima (o usuário deve apenas ter o cuidado de fornecer o mapa de acoplamento compatível com o do seu circuito). Ao aplicar M^{-1} no vetor de entrada, o usuário obtém os resultados com os erros mitigados.

Aplicação do método nos dados pticográficos

Na pticografia quântica, cada um dos $3n$ circuitos possui uma medição intermediária sobre um qubit, seguido de uma medição final sobre os n qubits. Portanto, cada um deles gera um vetor $2^{n+1} \times 1$ de número de *shots* associados às projeções em $\{|0\rangle|j\rangle, |1\rangle|j\rangle\}_{j=0}^{2^{n-1}}$. Neste caso, para obter uma matriz de calibração $2^{n+1} \times 2^{n+1}$ e sua inversa, nós utilizamos um qubit auxiliar, ou seja, um dos qubits disponíveis no circuito e que não estão sendo usados. Em seguida, aplicamos a mitigação sobre os resultados de cada circuito e obtemos o conjunto de dados pticográficos com os erros de leitura mitigados.

[2] Para mais detalhes veja: https://qiskit.org/documentation/stable/0.19/tutorials/noise/3_measurement_error_mitigation.html.

6.1.3 Procedimentos

As simulações da pticografia em um computador quântico são inteiramente realizadas através do Qiskit, utilizando códigos desenvolvidos em Python em todas as etapas, desde a criação dos circuitos até a análise dos resultados. Para implementá-las, adotamos a seguinte sequência de procedimentos:

1. Definimos um conjunto de estados quânticos que serão usados para os testes. No apêndice A, mostramos a construção dos circuitos de vários dos estados testados de n qubits (alguns exemplos para 2 e 3 qubits já foram vistos na seção 2.4.2).
2. Para cada estado $|\psi\rangle$, construímos os $3n$ circuitos para a pticografia seguindo os procedimentos descritos na seção 6.1.1.
3. Fixamos o número de *shots* (N_s), ou seja, o número de vezes que cada circuito será executado.
4. A simulação é realizada em dois cenários: sem ruído e com ruído.
5. Os dados pticográficos gerados são organizados para alimentar o algoritmo iterativo de reconstrução, PIE. No caso da simulação com ruído, os dados são duplicados e um dos conjuntos é submetido à mitigação de erros de leitura descrita na subseção anterior.
6. O algoritmo PIE é executado e os resultados obtidos são analisados.

No apêndice B mostramos exemplos dos códigos de todas essas etapas.

Análise dos resultados

Como foi visto na seção 5.2.2, a cada iteração do *loop* e a cada iteração PIE, nós computamos a distância do traço (equação (5.11)) entre os estados estimados pelo algoritmo. Esta medida avalia a convergência do algoritmo e não pressupõe nenhum conhecimento do estado que está sendo reconstruído. Porém, para avaliar um método tomográfico, os estados que serão reconstruídos devem ser conhecidos de antemão para que seja possível determinar a qualidade das reconstruções. Uma figura de mérito usual para esta avaliação é a fidelidade. No nosso caso, a fidelidade entre o estado que queremos reconstruir, $|\psi\rangle$, e o estado estimado pelo algoritmo PIE, $|\varphi_{\text{PIE}}\rangle$, é dada por

$$F(|\psi\rangle, |\varphi_{\text{PIE}}\rangle) = |\langle\psi|\varphi_{\text{PIE}}\rangle|^2, \tag{6.3}$$

onde $0 \leq F \leq 1$. A fidelidade é uma medida de quão próximos dois estados quânticos são. Portanto, uma reconstrução perfeita leva a $F = 1$. As fidelidades também serão computadas a cada iteração do *loop* e do PIE.

Além de estimar o estado, as distâncias e as fidelidades, outro dado que vamos obter do algoritmo é o tempo total de execução, para que seja possível compará-lo com outros métodos por esse critério. Tudo isso vale tanto para as simulações, quanto para os experimentos (que serão discutidos no próximo capítulo).

Simulações com ruído

O Qiskit possui ferramentas que permitem ao usuário realizar simulações com ruído.[3] Uma dessas ferramentas gera modelos automáticos de ruído compatíveis com o dispositivo real que será usado. Esses modelos são gerados levando em conta as fontes de erro que discutimos na seção 3.1.3 e, para isto, usam as informações de calibração do dispositivo (por exemplo, veja as figuras 28 a 31).

Neste trabalho, buscando um cenário mais realístico para o estudo do método pticográfico, nós realizamos simulações com ruído usando os modelos automáticos do Qiskit; um exemplo de como implementá-las pode ser visto no apêndice B. Aqui, nós apenas aplicamos o modelo, mas não nos aprofundamos na sua construção: de acordo com o desenvolvedor, ele é apenas uma aproximação dos erros que ocorrem nos dispositivos reais; é possível construir modelos muito mais realistas,[3] mas isso estava além dos nossos propósitos.

6.2 Simulações para 2 qubits

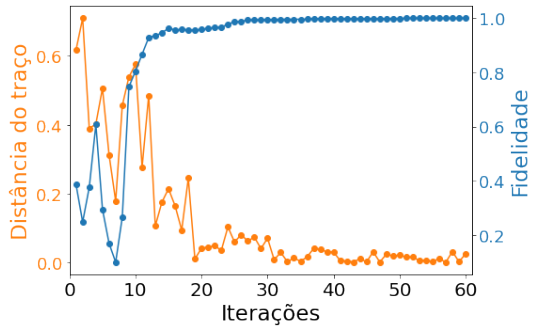
Nosso ponto de partida neste trabalho foi a aplicação da reconstrução pticográfica a estados de 2 qubits em um cenário livre de ruídos. Em seguida, emulamos o ruído para análise mais realista.

6.2.1 Simulações sem ruído

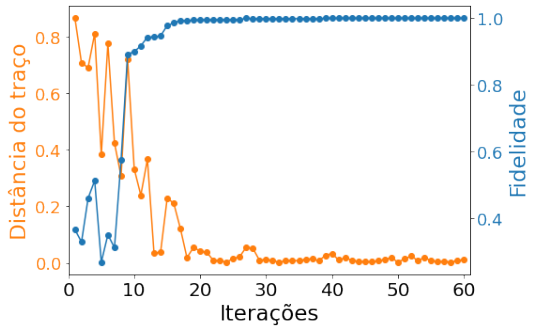
Os resultados que serão apresentados nesta subseção foram obtidos com um número fixo de $N_s = 2 \times 10^4$ *shots*, que era o limite das máquinas de livre acesso da IBM na época em que começamos a implementar as simulações. Nesta época também, nós adotamos um valor fixo de $\beta = 1{,}5$ para o parâmetro de controle das atualizações do algoritmo de reconstrução PIE (veja seção 5.2.2), conforme sugerido na proposta original da pticografia quântica [6]. Na ausência de ruído, nós simulamos a reconstrução de inúmeros estados e em todos os casos observamos a convergência do algoritmo em, no máximo, 5 iterações PIE, que equivalem a 60 iterações do *loop* mostrado no diagrama da figura 42. Ao convergir, a distância do traço (D) entre as estimativas do algoritmo chega a zero enquanto a fidelidade (F) da estimativa com o estado preparado chega a 1. Esta relação entre D e F mostra que, em princípio, o método pticográfico pode reconstruir estados puros desconhecidos.

[3] Veja discussões e tutoriais em https://qiskit.org/ecosystem/aer/tutorials/index.html.

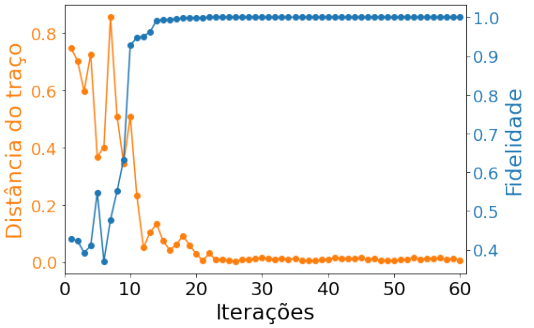
Para ilustrar a discussão, vamos considerar quatro estados: (a) $|\psi\rangle = \left(\frac{|0\rangle+e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes 2}$, (b) $|\psi\rangle = \left(\frac{|0\rangle-e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes 2}$, (c) $|\psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ e (d) $|\psi\rangle =$ Estado aleatório. Os estados (a) e (b) são separáveis, enquanto (c) e (d) são emaranhados (veja apêndice A). As figuras 50 e 51 mostram as evoluções de D e F em função do número de iterações do *loop* e iterações PIE, respectivamente, para cada um desses estados. Observa-se na figura 50 que, devido à estimativa inicial aleatória, D e F oscilam bastante nas primeiras iterações do *loop*, mas logo após a primeira entrada de todos os dados pticográficos (12 iterações do *loop* ou uma iteração PIE), as oscilações praticamente desaparecem e a convergência é rapidamente alcançada, como se vê na figura 51.



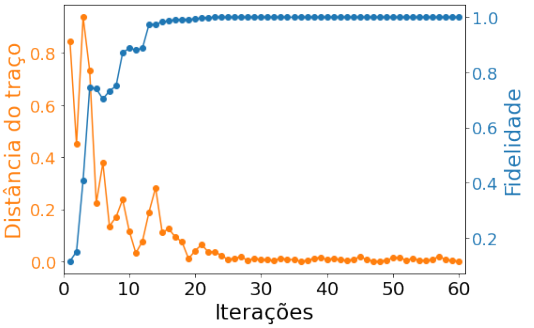
(a) $|\psi\rangle = \left(\frac{|0\rangle+e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes 2}$.



(b) $|\psi\rangle = \left(\frac{|0\rangle-e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes 2}$.



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



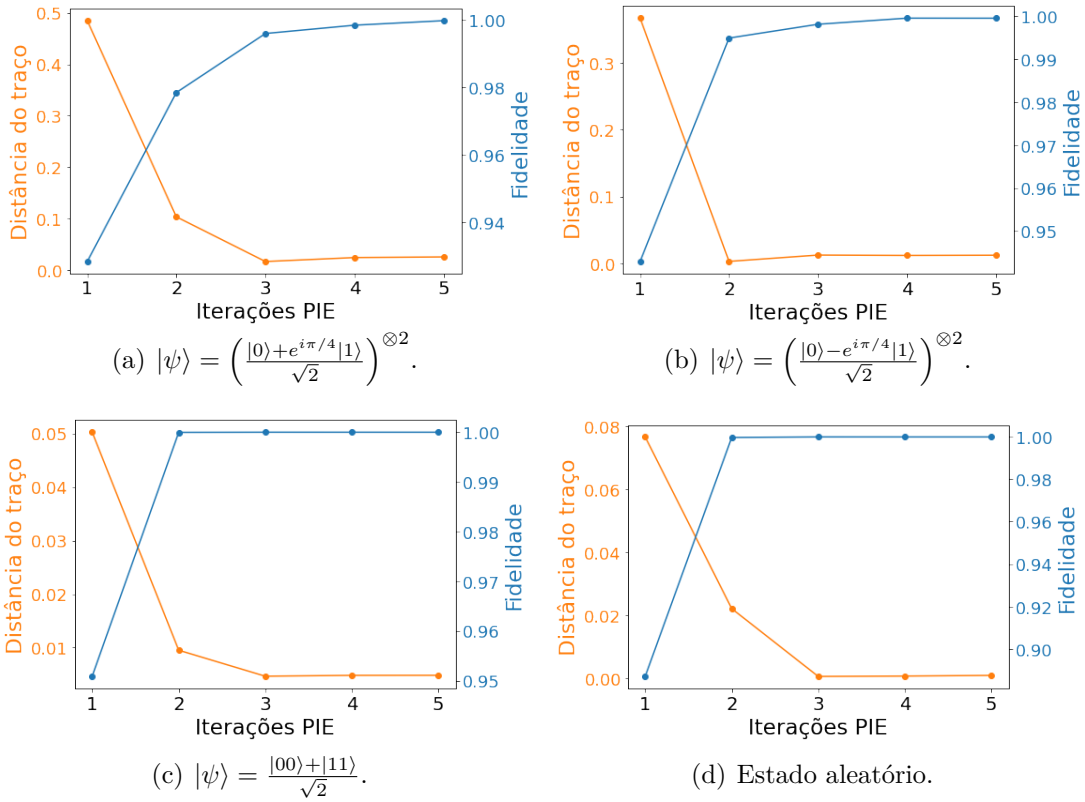
(d) Estado aleatório.

Figura 50 – Fidelidade e distância do traço em função das iterações do *loop* (diagrama da figura 42). As simulações foram realizadas sem ruído, com $N_s = 2 \times 10^4$ *shots* e $\beta = 1{,}5$; os estados reconstruídos estão indicados nas legendas.

6.2.2 Simulações com ruído

Parâmetro β variável

Usando os recursos do Qiskit discutidos na seção 6.1.3, iniciamos as simulações com ruído mantendo as mesmas condições das simulações sem ruído: $N_s = 2 \times 10^4$ e $\beta = 1{,}5$. Em alguns casos, o comportamento do algoritmo iterativo era similar ao da simulação sem ruído: nestes casos, as reconstruções eram bem sucedidas, mas demandavam um número maior de iterações, que fixamos em 20 iterações PIE. Porém, em geral, o que se observava



(a) $|\psi\rangle = \left(\frac{|0\rangle+e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes 2}$.

(b) $|\psi\rangle = \left(\frac{|0\rangle-e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes 2}$.

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

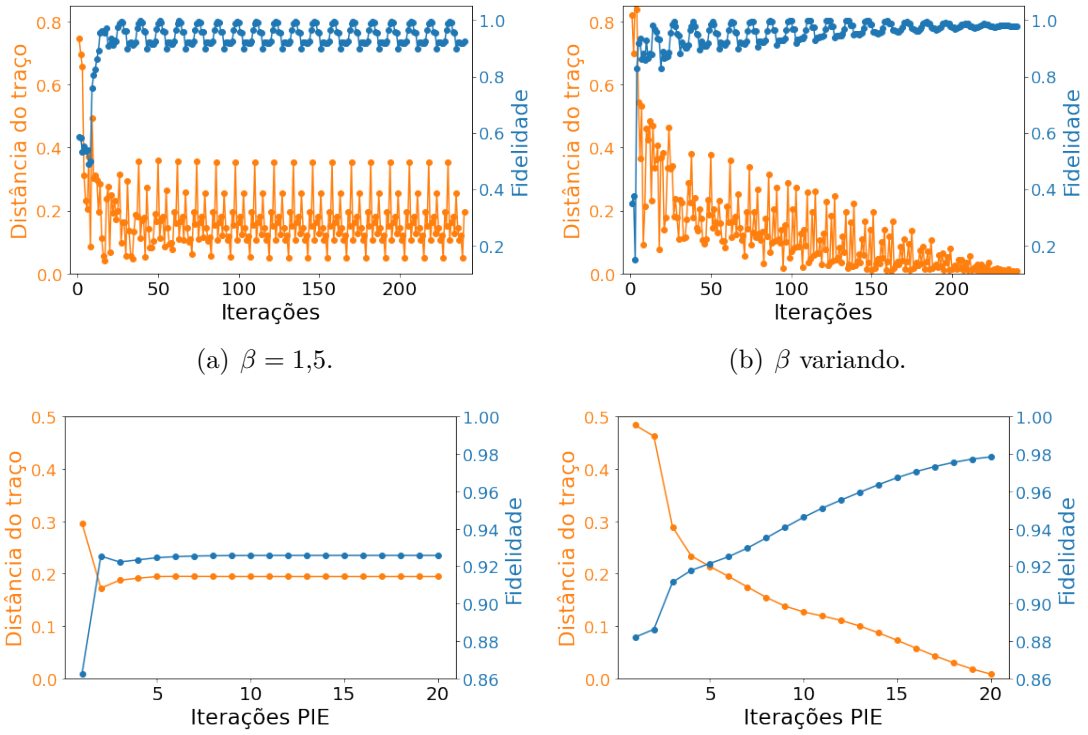
(d) Estado aleatório.

Figura 51 – Fidelidade e distância do traço em função das iterações PIE (cada iteração PIE corresponde a 12 iterações do *loop*) para os resultados mostrados na figura 50.

era uma estagnação do algoritmo de reconstrução: a distância do traço diminuía nas primeiras iterações, mas estagnava e não chegava próximo de zero nas iterações seguintes. Com isso, o mesmo efeito se observava em relação à fidelidade da reconstrução. Como exemplo, os gráficos (a) e (c) da figura 52 mostram a progressão do algoritmo em função do número de iterações do *loop* e PIE, respectivamente, para a reconstrução do estado de Bell $|\psi\rangle = |\psi_+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Pode-se observar na figura 52(a) que por volta de 50 iterações, D e F apresentam uma oscilação periódica que continuaria indefinidamente com a sequência de iterações. Esta é a característica de um quadro de estagnação que pode ser visto claramente no gráfico de iterações PIE da figura 52(c).

Como foi discutido em [6], o algoritmo PIE é equivalente a um algoritmo de gradiente descendente alternado e o parâmetro β, que controla o tamanho do passo da atualização do estado estimado a cada iteração, corresponde à taxa de aprendizagem do algoritmo.[4] Em linhas gerais, o gradiente descendente é um algoritmo iterativo de otimização que tem como objetivo encontrar o mínimo de uma função. A estagnação significa que o algoritmo ficou preso em um mínimo local da função de interesse (que

[4] O gradiente descendente é uma ferramenta bastante utilizada em ciência da computação, especialmente na área de aprendizado de máquina. Veja: http://cursos.leg.ufpr.br/ML4all/apoio/Gradiente.html.



(a) $\beta = 1{,}5$.

(b) β variando.

(c) $\beta = 1{,}5$.

(d) β variando.

Figura 52 – Fidelidade e distância do traço em função das iterações do *loop* [(a) e (b)] e iterações PIE [(c) e (d)], para a reconstrução de $|\psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. As simulações foram realizadas com ruído, com $N_s = 2 \times 10^4$ *shots* e $\beta = 1{,}5$ [(a) e (c)] ou β variável [(b) e (d)].

no nosso caso é a distância do traço). Estudando esses algoritmos [64], vimos que para evitar estagnação, a taxa de aprendizagem deve variar em função das iterações. Com isso, após diversos testes, adotamos um parâmetro β que decresce com as iterações. Mais especificamente, o algoritmo se inicia com $\beta = 2$ e este valor é diminuído de $\Delta\beta$ a cada iteração PIE. O valor de $\Delta\beta$ é uma fração decimal qualquer no conjunto $\{1, 2, 4, 5\}$ e a escolha inicial fixa o número de iterações PIE em $2/\Delta\beta$.

Considerando novamente o estado de Bell $|\psi_+\rangle$, os gráficos (b) e (d) da figura 52 mostram a progressão do algoritmo em função do número de iterações do *loop* e PIE, respectivamente, usando $\Delta\beta = 0{,}1$ (que produz 20 iterações PIE). No gráfico da figura 52(b) observa-se um efeito de amortecimeno das oscilações de D e F direcionando essas quantidades para seu valor mínimo e máximo, respectivamente. Neste exemplo obtivemos $D = 0{,}008$ e $F = 0{,}978$. A convergência também é vista de forma clara no gráfico da figura 52(d).

A partir desses resultados, todos os dados pticográficos gerados em simulações e experimentos passaram a ser analisados executando o algoritmo com β fixo e variável (veja apêndice B). As fidelidades alcançadas foram sempre maiores no segundo caso, como veremos nas próximas seções.

Simulações para um conjunto de dez estados de 2 qubits

No quadro abaixo, listamos um conjunto de dez estados de 2 qubits que serão utilizados em simulações e experimentos. Neste conjunto, selecionamos estados separáveis ($|\psi_1\rangle$ a $|\psi_4\rangle$) e estados emaranhados, incluindo os quatro estados de Bell ($|\psi_5\rangle$ a $|\psi_8\rangle$), um estado arbitrário ($|\psi_9\rangle$) e um aleatório ($|\psi_{10}\rangle$). A preparação desses estados é discutida no apêndice A.

Estados testados de 2 qubits

$$|\psi_1\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes 2}$$

$$|\psi_2\rangle = \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)^{\otimes 2}$$

$$|\psi_3\rangle = \left(\frac{|0\rangle + e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes 2}$$

$$|\psi_4\rangle = \left(\frac{|0\rangle - e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes 2}$$

$$|\psi_5\rangle = |\psi_+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

$$|\psi_6\rangle = |\psi_-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$

$$|\psi_7\rangle = |\phi_+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$

$$|\psi_8\rangle = |\phi_-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

$$|\psi_9\rangle = \begin{bmatrix} -0.09 + 0.477i \\ -0.353 - 0.0759i \\ -0.316 - 0.659i \\ 0.295 - 0.118i \end{bmatrix}$$

$$|\psi_{10}\rangle = \text{Estado aleatório}$$

A pticografia foi aplicada a cada um dos estados simulando o ruído de nove dispositivos da IBM mostrados na tabela 1, com os respectivos números de *shots* para cada simulação.[5] Isto seria equivalente a realizar o experimento em cada um dos dispositivos.

[5] Este número corresponde ao máximo permitido para realizar experimentos em cada dispositivo.

Na implementação dos circuitos pticográficos consideramos dois cenários para a QFT: a utilização ou não da porta **SWAP** ao final do circuito (veja seção 5.3.2). O uso da **SWAP** antes de uma medição na base computacional pode ser substituído por uma inversão nos bits do registrador clássico onde os dados são armazenados. Com isso, visamos diminuir o número de portas **CNOT**s e reduzir os efeitos do ruído no protocolo. Em todos os casos, os dados pticográficos produzidos, com e sem mitigação de erros de leitura, são enviados ao algoritmo PIE onde usamos β variável e $\Delta\beta = 0{,}04$. Portanto, o algoritmo será executado com 50 iterações PIE.

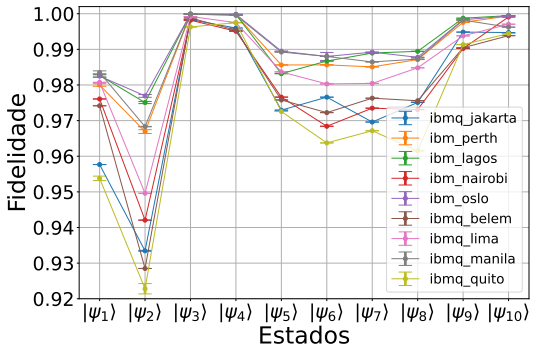
Tabela 1 – Dispositivos da IBM que foram simulados e o número de *shots* usados para cada um.

Dispositivos simulados	ibm_nairobi, ibm_oslo, ibmq_belem, ibmq_lima, ibmq_manila, ibmq_quito	ibm_lagos	ibmq_jakarta ibm_perth
Número de *shots*	2×10^4	$3{,}2 \times 10^4$	10^5

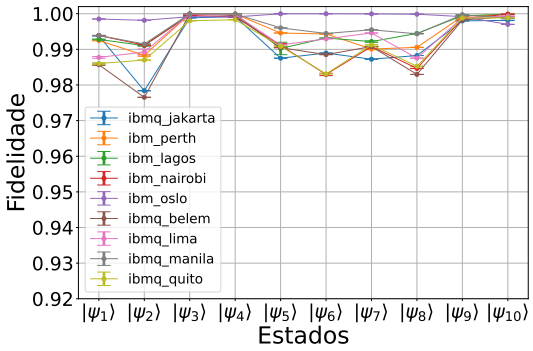
Para os quatro cenários de simulação possíveis (com/sem **SWAP**, com/sem mitigação), o algoritmo PIE foi executado 100 vezes para cada estado; os valores médios e os desvios-padrões das fidelidades foram computados e os resultados estão apresentados na figura 53. Em geral, todos os gráficos mostram bons resultados para as reconstruções, os quais vão melhorando à medida que o cenário fica mais favorável. No caso mais desfavorável (com **SWAP**/sem mitigação), o gráfico da figura 53(a) mostra que, com exceção do estado $|\psi_2\rangle$ em alguns dispositivos, as fidelidades ficaram acima de 0,97. Uma pequena melhora das fidelidades é notada quando a **SWAP** é removida, como mostra o gráfico da figura 53(c) (sem **SWAP**/sem mitigação). Ao implementarmos a mitigação de erros de leitura, obtemos uma melhora sensível dos resultados, como mostram os gráficos das figuras 53(b) (com **SWAP**/com mitigação) e 53(d) (sem **SWAP**/com mitigação), com todas as fidelidades acima de 0,98 e 0,99, respectivamente. Para ilustrar esta discussão de forma quantitativa, nós calculamos a variação percentual de cada fidelidade dos gráficos 53(b), 53(c) e 53(d) em relação àquelas de 53(a). Os resultados obtidos são mostrados na figura 54. De forma mais global, podemos também observar a melhora gradual dos resultados computando a mediana das fidelidades obtidas em cada cenário, como mostra a tabela 2.

Tabela 2 – Mediana das fidelidades mostradas em cada gráfico da figura 53.

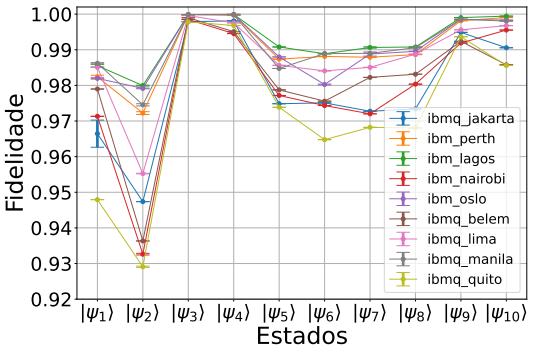
	sem mitigação	**com mitigação**
com SWAP	0,9872	0,9945
sem SWAP	0,9884	0,9965



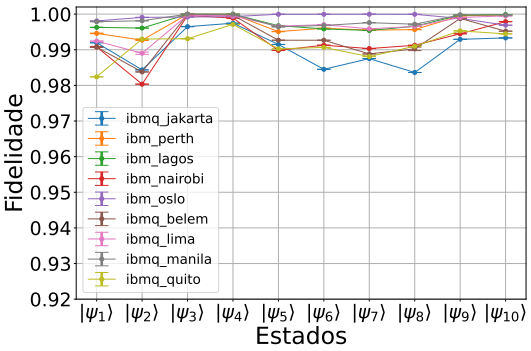
(a) Simulação com **SWAP**.



(b) Simulação com **SWAP** e com mitigação.



(c) Simulação sem **SWAP**.



(d) Simulação sem **SWAP** e com mitigação.

Figura 53 – Fidelidades médias para reconstruções pticográficas dos estados $\{|\psi_j\rangle\}_{j=1}^{10}$ (veja o quadro no texto) simuladas com ruído em 9 dispositivos da IBM indicados nas legendas. O algoritmo iterativo foi implementado com 50 iterações PIE usando β variável. Os cenários das simulações estão indicados abaixo dos gráficos e discutidos no texto.

Comparativo com a tomografia via medições de Pauli

Na caixa de ferramentas do Qiskit há funcionalidades[6] para realizar a tomografia de estados com medições de Pauli que discutimos na seção 4.2.1. Para comparar com nossos resultados, realizamos simulações deste método tomográfico para o mesmo grupo de estados $\{|\psi_j\rangle\}_{j=1}^{10}$ estudados na pticografia. Escolhemos a máquina ibm_lagos e nos limitamos ao número de 2×10^4 *shots*. Como estávamos interessados somente nas fidelidades de reconstrução, nós nos limitamos às funções básicas da ferramenta e não usamos filtros ou algoritmos de reconstrução mais sofisticados que levariam a um tempo mais longo de pós-processamento.

Na simulação, foram gerados 10 conjuntos de dados tomográficos por estado; a tomografia sobre cada conjunto foi realizada e a fidelidade média calculada. Os resultados obtidos são mostrados na figura 55, onde se observa que as fidelidades ficaram próximas

[6] Veja: https://qiskit.org/ecosystem/experiments/manuals/verification/state_tomography.html.

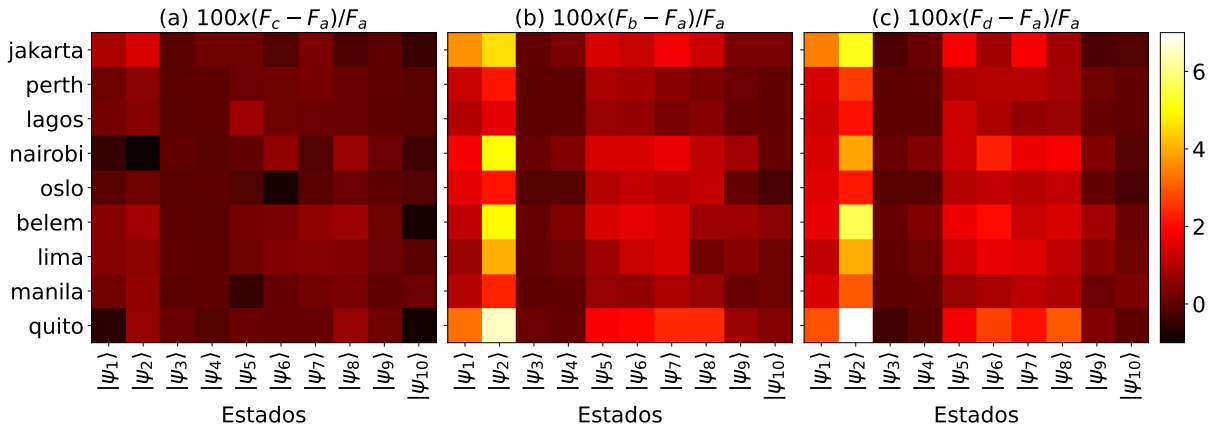


Figura 54 – Variação percentual das fidelidades mostradas nos gráficos das figuras 53(b), 53(c) e 53(d) em relação àquelas de 53(a) obtidas no cenário com **SWAP** e sem mitigação de erros. (a) sem **SWAP**/sem mitigação, (b) com **SWAP**/com mitigação e (c) sem **SWAP**/com mitigação.

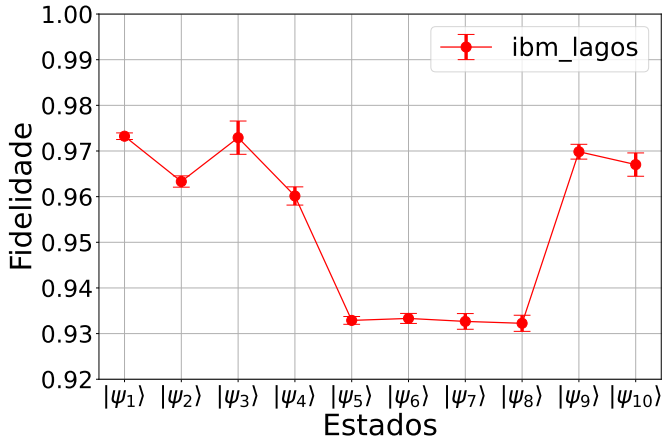


Figura 55 – Fidelidades médias para reconstrução tomográfica via medições de Pauli dos estados de 2 qubits $\{|\psi_j\rangle\}_{j=1}^{10}$ (veja o quadro no texto) simuladas com ferramenta do Qiskit.[6] As simulações foram realizadas com ruído emulado do dispositivo ibm_lagos e usando 2×10^4 *shots*.

ou abaixo de 0,97. As mais baixas foram para os estados de Bell ($\approx 0{,}935$). Todas estas fidelidades ficaram bem abaixo daquelas obtidas pela pticografia, em todos os cenários, como pode ser verificado por comparação com a curvas verdes da figura 53.

6.3 Simulações para *n* qubits

Nesta seção, estudaremos a performance do método pticográfico para a reconstrução de estados puros de n qubits. Assim como na seção anterior, primeiramente apresentamos os resultados para simulações sem ruído e, em seguida, para simulações com ruído. Nossos

resultados serão comparados com aqueles obtidos pelos métodos tomográficos discutidos no capítulo 5. Aqui, vamos prosseguir da mesma forma que na seção anterior, apresentando nossos resultados por meio de simulações sem ruído, seguidas por simulações com ruído.

6.3.1 Simulações sem ruído

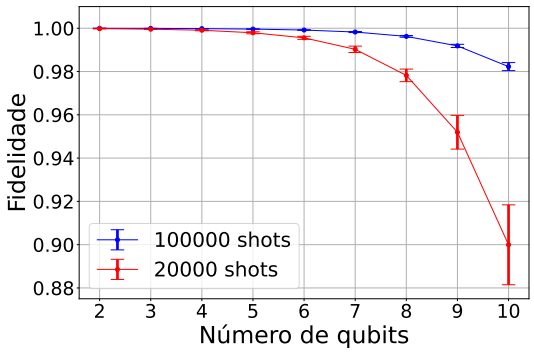
As simulações sem ruído são realizadas com ausência total de erros nos circuitos, ou seja, o funcionamento das portas quânticas e das medições é perfeito, e não há decoerência afetando os qubits. Todavia, neste cenário ainda há um elemento que pode comprometer a tomografia quântica: o tamanho finito do *ensemble* utilizado no processo. Um *ensemble* finito introduz erros nas estimativas das probabilidades (ou contagens) associadas às medições, e, consequentemente, na estimativa do estado. O seu tamanho é relativo à dimensão do sistema quântico e quanto menor for, maiores serão os erros gerados.

No nosso caso, o tamanho do *ensemble* é definido pelo número de *shots* utilizados. Como os computadores da IBM operam com um número fixo de *shots* (veja tabela 1), achamos interessante estudar como isso afeta a performance do método pticográfico na ausência de ruído e em função do número de qubits. Para realizar este estudo, consideramos a reconstrução de dois conjuntos de estados:

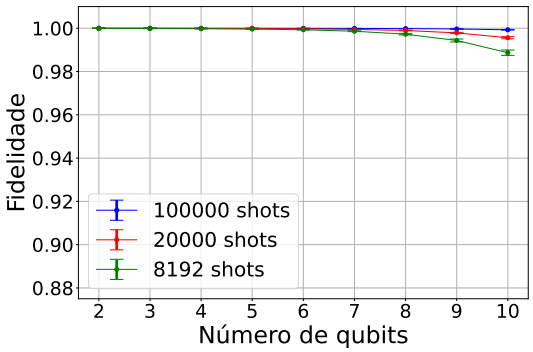
1. *Estados aleatórios separáveis*: estados gerados por portas de um qubit aleatórias.
2. *Estados aleatórios arbitrários*: estados gerados a partir de um vetor normalizado de 2^n números complexos aleatórios.

O método de preparação desses estados aleatórios é discutido no apêndice A. A diferença entre os conjuntos é que no primeiro, os estados serão sempre separáveis, enquanto no segundo, os estados serão, com grande probabilidade, emaranhados.

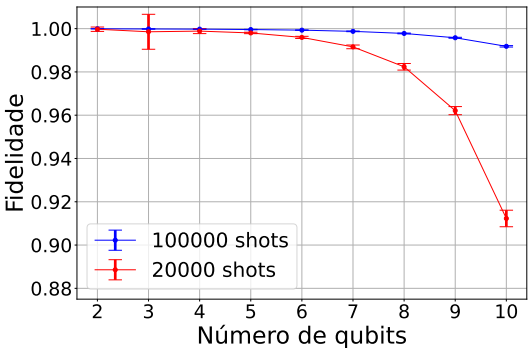
As simulações foram implementadas para $n = 2, 3, \ldots, 10$ qubits. Para cada n geramos 100 estados aleatórios de cada conjunto e simulamos as medições com três valores para o número de *shots*: $N_s = 2^{13} = 8192$, $N_s = 2 \times 10^4$ e $N_s = 10^5$. O primeiro valor é compatível com o limite anterior da IBM usado no método PZD [52] (veja seção 4.2.2), o que nos permite uma comparação mais justa entre os métodos. Os outros dois valores foram os limites disponíveis durante a realização do nosso trabalho (tabela 1). Os dados pticográficos produzidos foram analisados no algoritmo com 20 iterações PIE usando β fixo em 1,5 e β varíavel com $\Delta\beta = 0{,}1$. Para cada estado, foram realizadas 100 reconstruções, das quais calculamos a fidelidade média; em seguida calculamos a média das fidelidades sobre os 100 estados simulados e o seu desvio padrão. Os resultados obtidos são mostrados na figura 56. Em uma análise geral, observa-se que as fidelidades diminuem à medida que o número de qubits aumenta, mostrando o efeito do tamanho finito do *ensemble* e que este é relativo à dimensão do sistema. Por outro lado, a taxa de diminuição fica



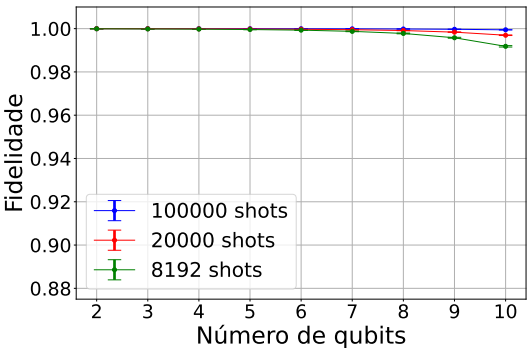
(a) Estados aleatórios separáveis - $\beta = 1{,}5$.



(b) Estados aleatórios separáveis - β variando.



(c) Estados aleatórios arbitrários - $\beta = 1{,}5$.



(d) Estados aleatórios arbitrários - β variando.

Figura 56 – Fidelidades médias para reconstrução pticográfica de estados aleatórios (veja detalhes no texto) na ausência de ruído e com número de *shots* indicado nas legendas. Para cada n simulamos a reconstrução de 100 estados implementando 20 iterações PIE com β fixo [(a) e (c)] e β variável [(b) e (d)].

menor à medida que o número de *shots* aumenta, mostrando que é possível contornar este problema. Observa-se também que, com poucas exceções, os desvios padrões são muito baixos, mostrando que as fidelidades obtidas pelo algoritmo são similares para todos os estados simulados. Comparando os gráficos entre si, confirmamos novamente que a implementação do PIE com β variável (figuras 56(a) e 56(c)) produz resultados bem melhores em relação ao uso do β fixo (figuras 56(b) e 56(d)), mesmo na ausência de ruído. Finalmente, uma comparação dos gráficos das figuras 56(b) e 56(d) mostra que a performance do método pticográfico, na ausência de ruído, não difere significativamente para estados separáveis e emaranhados, respectivamente.

Comparativo com o método PZD

O estudo apresentado acima foi inspirado em um estudo similar realizado no método PZD [52]. Lá, os autores também consideram conjuntos de 100 estados aleatórios (arbitrários e separáveis) para cada n e implementam as simulações sem ruído, com um número fixo de $2^{13} = 8192$ *shots*; os resultados que eles obtêm em termos das medianas das

fidelidades[7] estão reproduzidos na figura 57. As figuras 57(a) e 57(b) correspondem ao uso de $mn+1$ bases fatoráveis enquanto 57(c) e 57(d) ao uso de m bases não-fatoráveis mais a base computacional (veja seção 4.2.2), para $m = 2$ (azul), $m = 3$ (amarelo) e $m = 4$ (roxo). Além disso, 57(a) e 57(c) correspondem à reconstrução de estados aleatórios arbitrários e 57(b) e 57(d) a estados aleatórios separáveis.

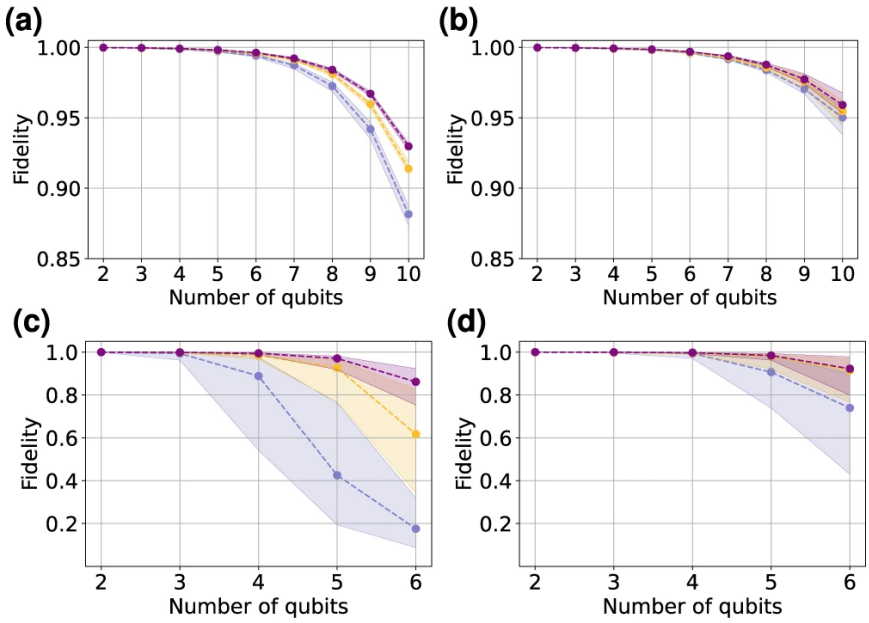


Figura 57 – Fidelidade de estados aleatórios reconstruídos pelo método PZD, em função do número de qubits. (a) e (b) reconstrução com $2n+1$ (azul), $3n+1$ (amarelo) e $4n+1$ (roxo) bases fatoráveis. (c) e (d) reconstrução com 2 (azul), 3 (amarelo) e 4 (roxo) bases não-fatoráveis + base computacional. (a) e (c) [(b) e(d)] estados aleatórios arbitrários (separáveis). Figura adaptada de [52].

Os resultados da figura 57 devem ser comparados às curvas verdes da figura 56(b) e 56(d). É nítido que o método pticográfico mostra melhores resultados, especialmente para $n > 4$ qubits. Por exemplo, para $n = 10$, obtivemos fidelidades médias de aproximadamente 0,99 para ambos os conjuntos aleatórios, enquanto no PZD as fidelidades ficaram, no melhor dos casos ($m = 4$), em torno de 0,93 e 0,96 para os estados arbitrários e separáveis, respectivamente. Além disso, o método PZD apresentou diferenças na reconstrução de estados separáveis (figuras 57(b) e 57(d)) e emaranhados (figuras 57(a) e 57(c)), com melhores fidelidades para os primeiros, algo que não observamos no método pticográfico.

Tempo de execução do algortimo PIE

Na implementação do algoritmo PIE nas simulações e experimentos, nós adotamos como critério de parada o número de iterações fixado *a priori*. Este número determinará o tempo de execução do algoritmo. Nas simulações que discutimos acima, onde utilizamos

[7] As áreas sombreadas representam as medida de dispersão dos dados em torno da mediana.

20 iterações PIE, nós também computamos o tempo médio de execução do algoritmo em função do número de qubits. Os resultados obtidos estão mostrados na curva azul da figura 58 (“Máquina 1”, que corresponde ao computador onde realizamos as simulações que geraram a figura 56). Para 10 qubits, a reconstrução durou em média 8 s, o que representa uma grande vantagem sobre os algoritmos de pós-processamento utilizados em tomografia quântica que podem durar algumas horas [46]. Por outro lado, o método PZD se mostrou mais rápido, com um tempo de reconstrução de aproximadamente 0,5 s [52].

Nós também realizamos simulações nas mesmas condições anteriores em outro computador (“Máquina 2”) para mostrar o efeito de equipamentos com capacidade de processamento diferente no tempo de execução do algoritmo. Neste caso fizemos até 11 qubits e o resultado é mostrado na curva vermelha da figura 58. Em geral, as reconstruções foram mais rápidas (para 10 qubits o tempo foi de aproximadamente 6 s) e observa-se que a curva apresenta uma tendência de crescimento mais lento com o número de qubits.

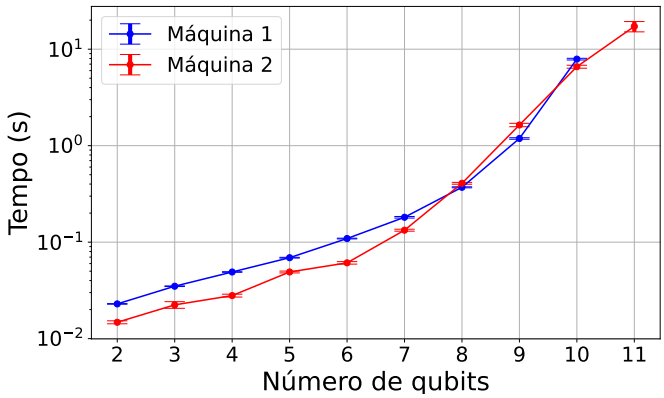


Figura 58 – Tempo médio em função do número de qubits para execução do algoritmo iterativo de reconstrução considerando 20 iterações PIE. Cada curva foi obtida em um computador doméstico com capacidades de processamento diferentes.

Além de computadores com maior capacidade, é possível otimizar o próprio algoritmo PIE para reduzir o tempo de processamento. Como foi dito na seção 5.2.2, o algoritmo que nós utilizamos é uma versão mais “primitiva” do PIE. Ele pode ser melhorado se tornando ainda mais rápido, por exemplo, ao inserirmos algum critério de parada mais refinado sem comprometer a reconstrução do estado. Tais modificações podem ser exploradas e estão em aberto.

6.3.2 Simulações com ruído

Assim como no caso de 2 qubits, realizamos simulações com ruído para n qubits considerando um conjunto selecionado de estados listado no quadro abaixo.

Estados testados de n qubits

$$|\psi_1^n\rangle = \left(\frac{|0\rangle + e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes n}$$

$$|\psi_2^n\rangle = \left(\frac{|0\rangle - e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes n}$$

$$|\psi_3^n\rangle = \text{Estado aleatório separável}$$

$$|\psi_4^n\rangle = |GHZ\rangle = \frac{|0\rangle^{\otimes n} + |1\rangle^{\otimes n}}{\sqrt{2}}$$

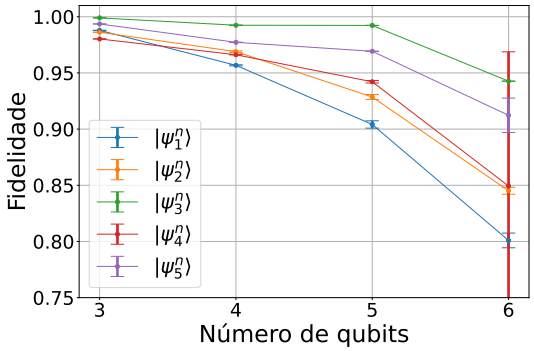
$$|\psi_5^n\rangle = |W\rangle = \frac{1}{\sqrt{n}}(|100\ldots0\rangle + |010\ldots0\rangle + \cdots + |000\ldots1\rangle)$$

O conjunto inclui estados separáveis, $|\psi_1^n\rangle$ a $|\psi_3^n\rangle$, e estados emaranhados, $|\psi_4^n\rangle$ e $|\psi_5^n\rangle$, cujos métodos de preparação são discutidos no apêndice A. Para emular o ruído dos dispositivos da IBM, selecionamos apenas três máquinas das simulações anteriores que mostraram melhores resultados: ibm_perth, ibm_oslo e ibm_lagos. Todas elas são máquinas de 7 qubits e operam com o número máximo de *shots* mostrado na tabela 1. Diante dos resultados obtidos para 2 qubits, as simulações aqui foram todas realizadas usando a QFT sem as portas **SWAP** ao final e os dados pticográficos foram analisados no PIE, com e sem mitigação de erros de leitura, usando β variável, com $\Delta\beta = 0{,}04$, o que nos dá 50 iterações PIE.

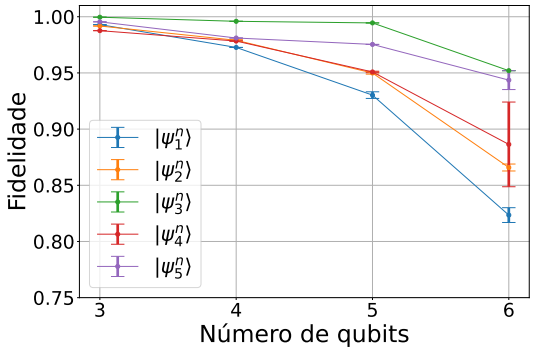
As simulações foram implementadas para $n = 3, 4, 5, 6$ qubits.[8] Para cada um dos estados da lista, realizamos 100 reconstruções e calculamos o valor médio da fidelidade e o seu desvio padrão. Os resultados obtidos são mostrados na figura 59: a primeira coluna apresenta os resultados sem mitigação de erros e a segunda, com mitigação; cada linha corresponde a um dos dispositivos simulados. Em geral, as reconstruções apresentaram ótimas fidelidades. Observa-se que elas diminuem com o número de qubits, o que é natural pelo aumento do ruído e do efeito de *ensemble* finito com a dimensão do sistema. O aumento das fidelidades também é claramente observado com a aplicação da mitigação de erros: a variação percentual das fidelidades antes e depois da mitigação é mostrada na figura 60. Em termos de performance, o dispositivo ibm_lagos foi superior, seguido por ibm_oslo. Uma análise individual dos estados reconstruídos mostra que as melhores fidelidades foram alcançadas por $|\psi_3^n\rangle$ e $|\psi_5^n\rangle$. Como o primeiro é um estado separável, que possui circuito de preparação menos ruidoso, o resultado era esperado. Porém, o segundo é o estado W, cujo circuito de preparação é o mais complexo do conjunto testado

[8] Apesar dos dispositivos simulados terem 7 qubits, nós só utilizamos 6 para que fosse possível usar um qubit auxiliar na mitigação de erros.

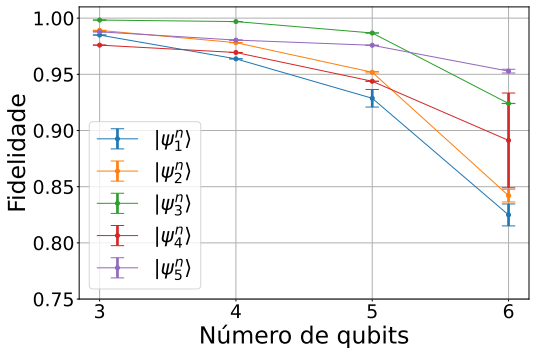
(apêndice A), e as fidelidades obtidas foram surpreendentes. Da mesma forma, as piores fidelidades foram alcançadas para o estado separável $|\psi_1^n\rangle$, que também foi surpreendente.



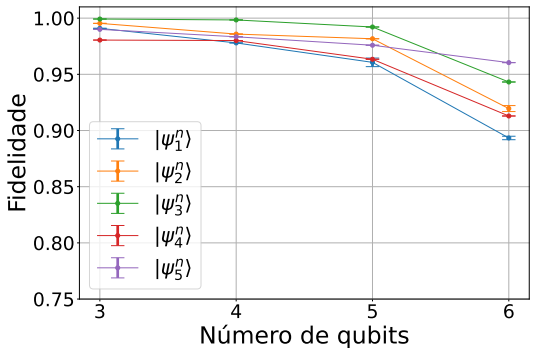
(a) ibm_perth (sem mitigação).



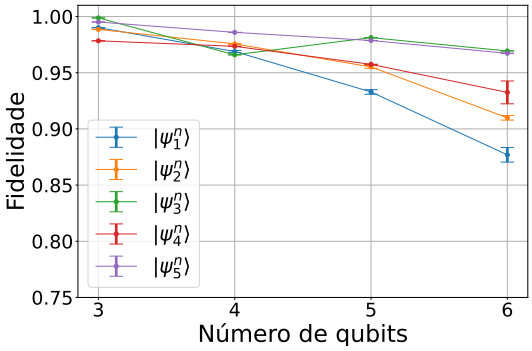
(b) ibm_perth (com mitigação).



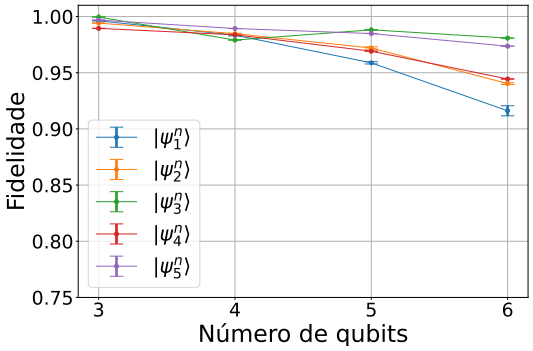
(c) ibm_oslo (sem mitigação).



(d) ibm_oslo (com mitigação).



(e) ibm_lagos (sem mitigação).



(f) ibm_lagos (com mitigação).

Figura 59 – Fidelidades médias para reconstruções pticográficas dos estados $\{|\psi_j^n\rangle\}_{j=1}^5$ (veja o quadro no texto) em função do número de qubits. As simulações foram realizadas com ruído emulado dos dispositivos da IBM indicados abaixo dos gráficos. A primeira (segunda) coluna mostra os resultados sem (com) mitigação de erros de leitura.

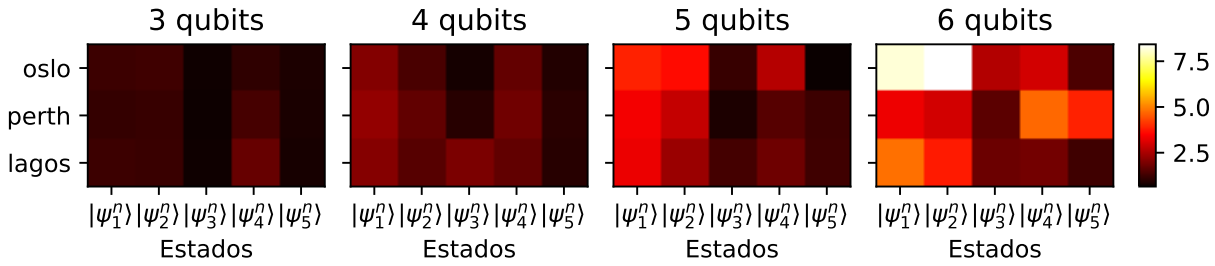


Figura 60 – Variação percentual das fidelidades antes e depois da mitigação de erros de leitura, mostradas na primeira e segunda coluna da figura 59, respectivamente.

Comparativo com a tomografia via medições de Pauli

Novamente, utilizamos a ferramenta do Qiskit para tomografia via medições de Pauli para compararmos os métodos de reconstrução de estados. Neste caso, simulamos a tomografia dos estados $\{|\psi_j^n\rangle\}_{j=1}^5$, para $n = 3, \ldots, 7$ qubits, considerando as mesmas condições do caso de 2 qubits (ruído emulado do dispositivo ibm_lagos e uso de 2×10^4 *shots*). As fidelidades médias obtidas em função de n são mostradas na figura 61. Todas elas ficaram bem abaixo das fidelidades obtidas na pticografia, sem ou com mitigação de erros, como se vê nas figuras 59(e) e 59(f), respectivamente. É possível notar também na figura 61 uma clara discrepância entre as reconstruções dos estados separáveis ($|\psi_1^n\rangle$ a $|\psi_3^n\rangle$) e emaranhados ($|\psi_4^n\rangle$ e $|\psi_5^n\rangle$), mostrando que o método é menos resistente ao ruído do que o método pticográfico.

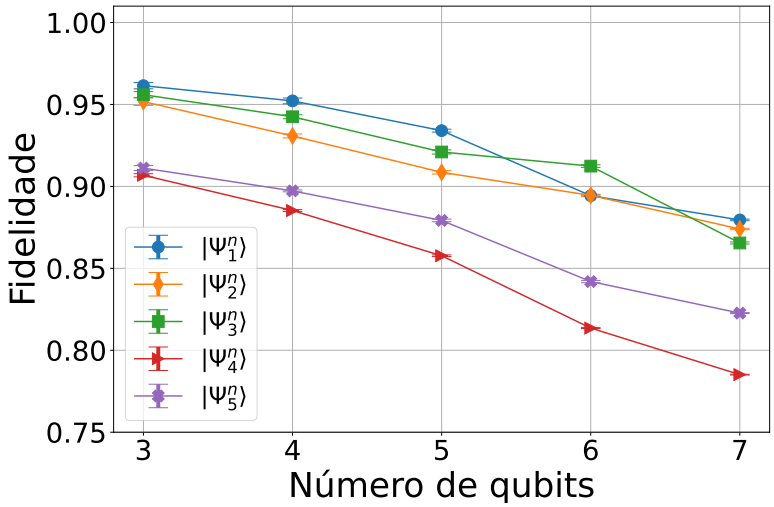


Figura 61 – Fidelidades médias para reconstrução tomográfica via medições de Pauli dos estados de n qubits $\{|\psi_j\rangle\}_{j=1}^5$ (veja o quadro no texto) simuladas com ferramenta do Qiskit. As simulações foram realizadas com ruído emulado do dispositivo ibm_lagos e usando 2×10^4 *shots*.

6.4 Pticografia quântica com medição final em outras bases

Até aqui, nós implementamos a pticografia quântica com as projeções intermediárias $\{\mathbf{\Pi}^{\pm}_{\xi j}\}$ (equação (5.12)) e medição final na base $\{\mathbf{F}^{\dagger}_{n}|j\rangle\}$ que requer o uso da QFT. Tudo isto seguiu rigorosamente a proposta original [6]. No entanto, como mencionado na seção 5.4, os autores do método sugerem que há uma flexibilidade na escolha da base para a medição final. Para dispositivos NISQ seria interessante buscar uma alternativa ao uso da QFT, uma vez que a complexidade do seu circuito (seção 5.3.2) poderia inviabilizar a aplicação do método devido às altas taxas de erro introduzidas. Nesta seção, nós apresentamos resultados de alguns estudos preliminares onde testamos a pticografia com medições em outras bases.

6.4.1 Pticografia com a QFT aproximada: Simulações

Transformada quântica de Fourier aproximada

Uma primeira alternativa à QFT é a sua versão aproximada (AQFT). A AQFT para n qubits é caracterizada por um parâmetro $m = 1, \ldots, n$ que controla o grau de aproximação. Para um dado m, sua atuação na base computacional é obtida da QFT (equação (5.16)), eliminando-se os termos b_k da k-ésima casa onde $k > m$, o que nos fornece [65]

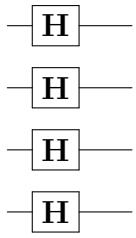
$$|j\rangle \to \frac{1}{\sqrt{2^n}} \left[|0\rangle + e^{2\pi i 0.b_0}\ |1\rangle\right] \otimes \left[|0\rangle + e^{2\pi i 0.b_1 b_0}\ |1\rangle\right] \otimes \cdots \otimes \left[|0\rangle + e^{2\pi i 0.b_{m-1}\cdots b_0}\ |1\rangle\right] \otimes \\ \otimes \left[|0\rangle + e^{2\pi i 0.b_m \cdots b_1}\ |1\rangle\right] \otimes \cdots \otimes \left[|0\rangle + e^{2\pi i 0.b_{n-1}\cdots b_{n-m}}\ |1\rangle\right]. \tag{6.4}$$

No circuito da QFT temos as portas de fase controladas com índices $2, \ldots, n$, configuradas como mostrado na figura 45. O efeito da AQFT de grau m no circuito é eliminar todas essas portas com índice maior que m. Para $m = 1$, todas as portas controladas são eliminadas e só resta uma porta Hadamard sobre cada qubit; esta operação é conhecida como transformada de Hadamard. Para $1 < m < n$, temos as aproximações da QFT com menos portas controladas entre os qubits mais separados no circuito. Para $m = n$, temos a própria QFT. Portanto, quanto menor o valor de m, pior será a aproximação; porém, teremos menos operações controladas no circuito, o que é vantajoso para reduzir o ruído. Como exemplo, a figura 62 mostra os circuitos da AQFT para 4 qubits; nós omitimos as portas **SWAP** pois elas não foram usadas nas simulações e experimentos.

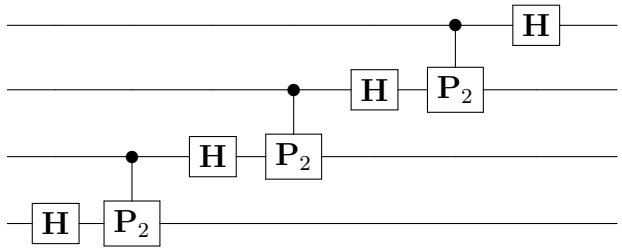
Resultados

Para analisar os efeitos da AQFT na pticografia quântica, implementamos simulações para $n = 3, \ldots, 7$ qubits considerando o mesmo conjunto de cinco estados apresentados no quadro da seção 6.3.2. Além disso, usamos AQFTs de graus $m = 1, 2, 3, 4$, o que nos

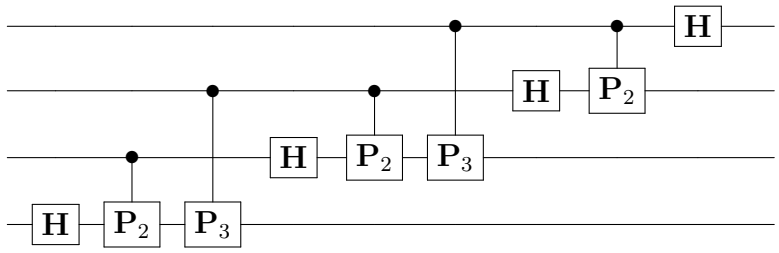
(a) $m = 1$



(b) $m = 2$



(c) $m = 3$



(d) $m = 4$

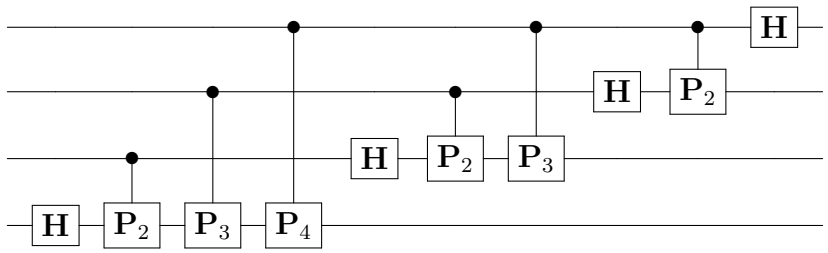
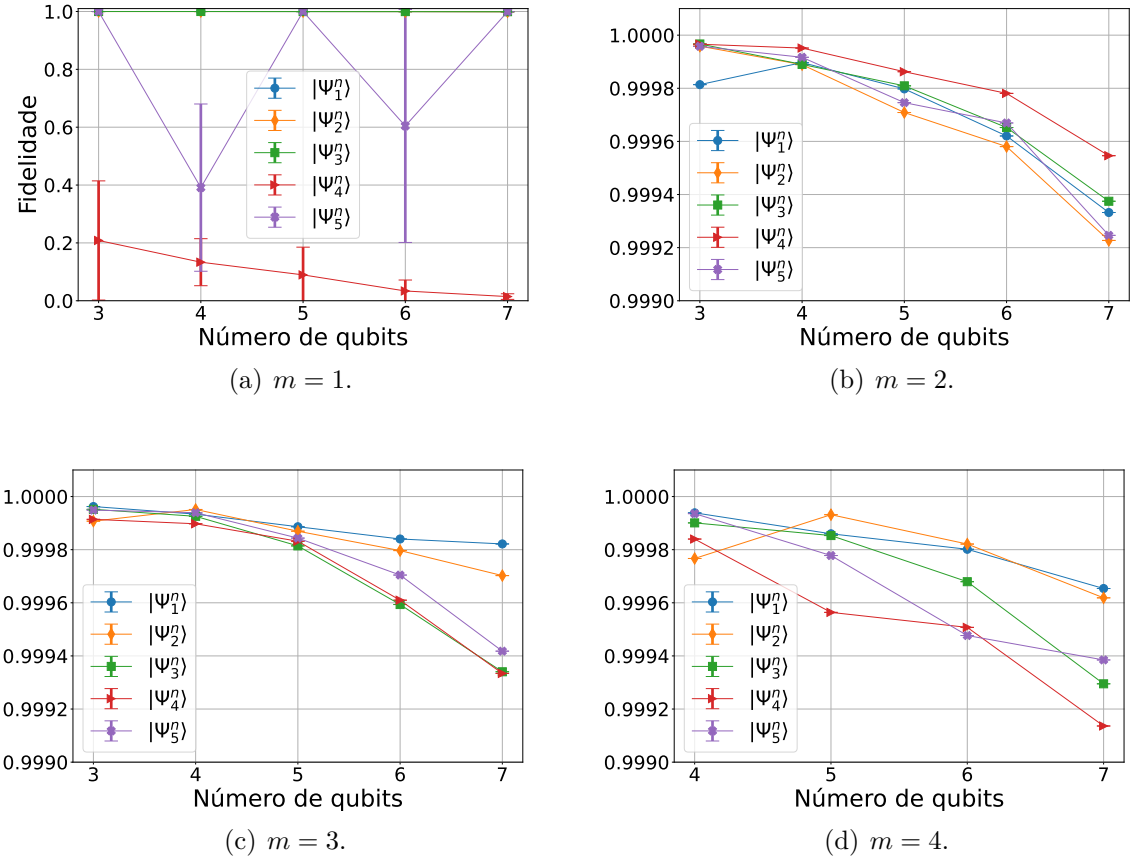


Figura 62 – Circuitos para AQFT (sem **SWAP**s) de 4 qubits e diferentes valores de m.

permite cobrir todas as aproximações para $n = 3, 4, 5$. No algoritmo de reconstrução, utilizamos β variável com $\Delta\beta = 0{,}04$, o que nos dá 50 iterações PIE. Realizamos 10 reconstruções para cada estado e computamos a fidelidade média e o seu desvio padrão.

Primeiramente, consideramos um cenário livre de ruído e simulamos a pticografia com 2×10^4 *shots*. Os resultados obtidos são mostrados na figura 63. Para $m = 1$, as reconstruções foram praticamente perfeitas para os estados separáveis e muito ruins para os estados emaranhados, especialmente o estado GHZ ($|\psi_4^n\rangle$); para o estado W ($|\psi_5^n\rangle$),



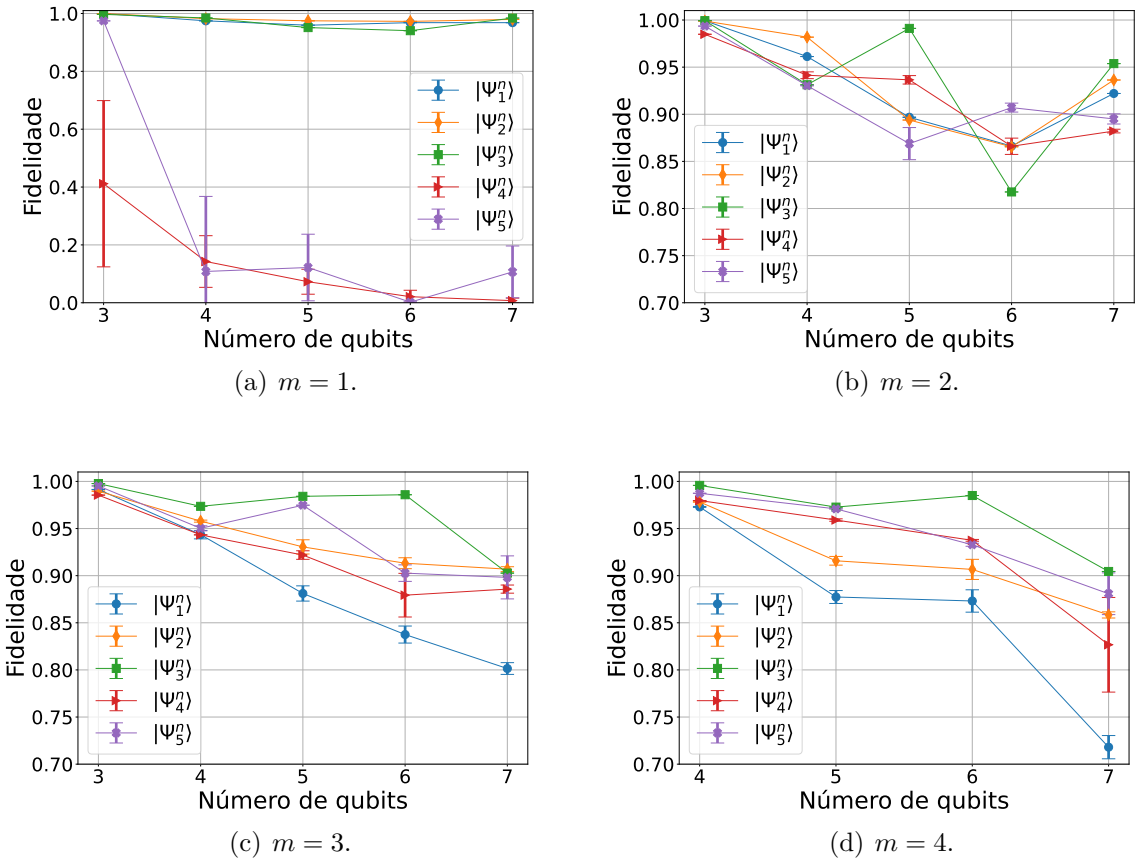
(a) $m = 1$. (b) $m = 2$.

(c) $m = 3$. (d) $m = 4$.

Figura 63 – Resultados para simulações sem ruído usando a AQFT. Fidelidades médias para reconstruções pticográficas dos estados $\{|\psi_j^n\rangle\}_{j=1}^5$ (veja o quadro na seção 6.3.2) em função do número de qubits. As simulações foram realizadas usando 2×10^4 *shots*. O grau m da AQFT está indicado abaixo de cada gráfico.

excelentes reconstruções foram obtidas somente para um número ímpar de qubits, e o motivo para isso não sabemos explicar. Esses resultados nos levam a concluir que a transformada de Hadamard não é uma boa candidata para substituir a QFT na pticografia. Por outro lado, as aproximações de grau $m > 1$ mostraram excelentes resultados, com reconstruções quase perfeitas independentemente do valor de m. Os valores das fidelidades diferem somente na quarta casa decimal e a diminuição com o número de qubits resulta, provavelmente, do efeito de *ensemble* finito. Com isso, podemos dizer que a AQFT com $m > 1$ tem potencial, em princípio, para subsituir a QFT na pticografia. Obviamente serão necessários testes mais robustos para chegarmos a uma conclusão.

Em seguida, avançamos para análises com ruído, onde escolhemos e emulamos o ruído da máquina ibm_lagos, que havia apresentado melhor performance para o caso de n qubits. As simulações foram realizadas com $3{,}2 \times 10^4$ *shots* e não aplicamos a mitigação de erros de leitura nos dados gerados. A figura 64 mostra os resultados obtidos. Novamente, para $m = 1$ observamos bons resultados apenas para os estados separáveis. Para $m > 1$, os resultados ficaram razoavelmente bons e, para um dado n, não se observam grandes



(a) $m = 1$. (b) $m = 2$.

(c) $m = 3$. (d) $m = 4$.

Figura 64 – Resultados para simulações com ruído usando a AQFT. Fidelidades médias para reconstruções pticográficas dos estados $\{|\psi_j^n\rangle\}_{j=1}^5$ (veja o quadro na seção 6.3.2) em função do número de qubits. As simulações foram realizadas com ruído emulado do dispositivo ibm_lagos usando $3{,}2 \times 10^4$ *shots*. O grau m da AQFT está indicado abaixo de cada gráfico.

discrepâncias entre as fidelidades alcançadas em função do grau m da AQFT. Os gráficos para $m > 1$ e $n < 7$ podem ser comparados com o gráfico da figura 59(e), onde usamos a QFT e as simulações partiram das mesmas condições iniciais. Comparando as fidelidades para cada n, observamos resultados melhores com o uso da QFT completa para $n > 3$. Como as simulações foram realizadas em períodos bastante distintos, não se pode descartar que parte disso seja consequência de piores dados de calibração do dispositivo que usamos para emular ruído. De qualquer forma, os resultados das simulações com o uso da AQFT são promissores e uma investigação nesta direção pode ser aprofundada.

6.4.2 Pticografia com bases fatoráveis: Simulações

A AQFT com $m > 1$ mostrou resultados promissores para a pticografia quântica. Porém, mesmo no caso mais simples de $m = 2$, o circuito para a operação envolverá $n - 1$ portas de dois qubits e terá a mesma profundidade da QFT, $P = 2n$, o que ainda pode ser problemático para dispositivos NISQ. Por outro lado, a AQFT com $m = 1$ envolve uma

operação fatorável, $\mathbf{H}^{\otimes n}$, mas os resultados obtidos com ela não foram favoráveis.

Como as medições em bases fatoráveis produzem circuitos muito mais simples e que são menos afetados por ruído, seria interessante investigar se existem outras operações unitárias fatoráveis que sejam adequadas para o método pticográfico. Com esse intuito, nós consideramos uma operação unitária aleatória sobre n qubits dada por

$$\mathbf{U}^{\text{rnd}} = \mathbf{U}_0^{\text{rnd}} \otimes \mathbf{U}_1^{\text{rnd}} \otimes \cdots \otimes \mathbf{U}_{n-1}^{\text{rnd}}, \tag{6.5}$$

onde $\mathbf{U}_j^{\text{rnd}} \equiv \mathbf{U}^{\text{rnd}}(\theta_j, \phi_j, \lambda_j)$ é a uma operação arbitrária sobre o qubit j dada pela equação (2.26) e gerada por valores aleatórios $(\theta_j, \phi_j, \lambda_j)$. Em termos das portas-base da IBM, o circuito para implementação desta operação é mostrado na figura 65.

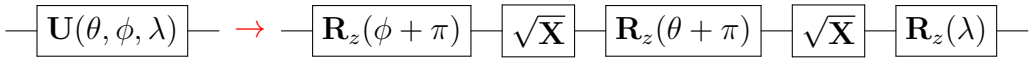
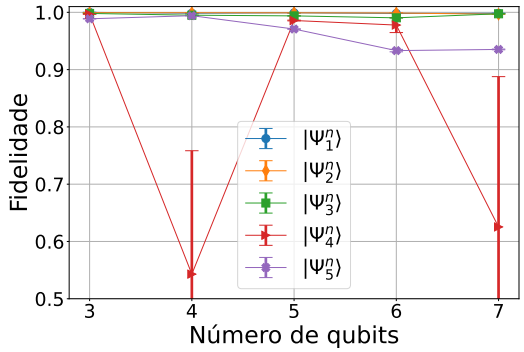


Figura 65 – Decomposição de uma porta de um qubit arbitrária (equação (2.26)) em termos das portas-base da IBM.

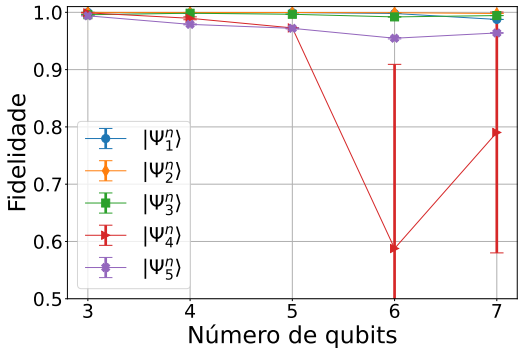
Neste estudo preliminar, realizamos diretamente as simulações com ruído emulado dos dispositivos ibm_lagos e ibm_perth, usando 2×10^4 e 10^5 *shots*, respectivamente. Novamente, simulamos a reconstrução dos cinco estados apresentados no quadro da seção 6.3.2 para $n = 3, \ldots, 7$ qubits. No algoritmo de reconstrução, utilizamos β variável com $\Delta\beta = 0{,}04$ e realizamos 10 reconstruções para cada estado, computando a fidelidade média e o desvio padrão para cada um. As simulações foram realizadas duas vezes para cada dispositivo e não aplicamos a mitigação de erros de leitura nos dados gerados.

Os resultados obtidos são mostrados na figura 66. É importante ressaltar que para cada estado simulado, sorteamos aleatoriamente o conjunto de ângulos $\{(\theta_j, \phi_j, \lambda_j)\}_{j=0}^{n-1}$ que define $\mathbf{U}^{\text{rnd}}$ na equação (6.5), ou seja, cada ponto nos gráficos foi obtido usando-se uma unitária fatorável diferente. Observamos em todos os casos reconstruções excelentes para os estados separáveis, o que foi visto também com o uso de $\mathbf{H}^{\otimes n}$ (figuras 63(a) e 64(a)). As fidelidades para o estado W também foram ótimas; para $n > 3$ qubits ficaram inclusive melhores que no caso da AQFT com $m > 1$ (veja figura 64). No entanto, para o estado emaranhado GHZ, a reconstrução não foi bem sucedida em alguns casos, mas não foi possível extrair um padrão de comportamento deles. O que se observa é que os desvios padrões de tais casos, ao contrário de todos os outros, foram bem altos, indicando que entre as 10 reconstruções, uma ou mais tiveram valores razoáveis ou bons.

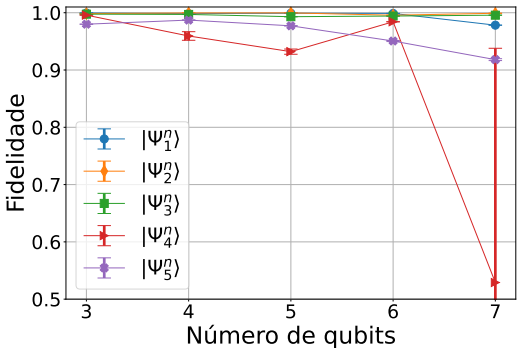
Assim como no caso da AQFT, são necessários estudos mais robustos para se estabelecer a viabilidade da medição final da pticografia em uma base fatorável. Aqui, fizemos apenas testes iniciais com um conjunto bastante restrito de estados. Os indícios apontam que as bases fatoráveis são adequadas para a estimativa de estados separáveis,



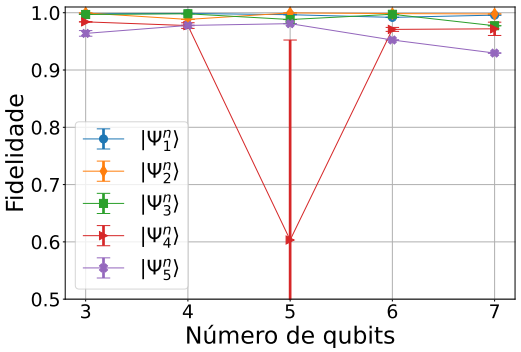
(a) ibm_lagos.



(b) ibm_lagos.



(c) ibm_perth.



(d) ibm_perth.

Figura 66 – Resultados para simulações com ruído usando bases aleatórias fatoráveis. Fidelidades médias para reconstruções pticográficas dos estados $\{|\psi_j^n\rangle\}_{j=1}^5$ (seção 6.3.2) em função do número de qubits. As simulações foram realizadas com ruído emulado dos dispositivos indicados abaixo dos gráficos.

mas apresentam dificuldades na estimativa de estados emaranhados. Além de um estudo mais aprofundado, outra abordagem para o uso de bases fatoráveis seria a sua combinação com a medições intermediárias não-fatoráveis, ao contrário das projeções que usamos aqui.

7 Pticografia em um computador quântico: Experimentos

No capítulo anterior, exploramos a pticografia quântica por meio de simulações. Neste capítulo, apresentaremos nossos resultados experimentais utilizando os computadores quânticos da IBM. Começaremos com uma breve descrição geral de como os experimentos foram conduzidos. Em seguida, apresentamos os resultados para a reconstrução de estados de 2 qubits e n qubits. Para o segundo caso, nós mostramos também os resultados obtidos em experimentos com as bases da QFT aproximada e unitárias aleatórias fatoráveis. Por fim, discutiremos os pontos positivos e negativos de nossos resultados experimentais.

7.1 Preliminares

Os experimentos foram conduzidos nos dispositivos de 7 qubits ibm_oslo, ibm_lagos e ibm_perth, usando o número máximo de *shots* permitidos para cada um: $2{\times}10^4$, $3{,}2{\times}10^4$ e 10^5, respectivamente. Para 2 qubits, implementamos a pticografia para os estados $\{|\psi_j\rangle\}_{j=1}^{10}$ listados no quadro da seção 6.2.2. Para n qubits, os estados testados foram $\{|\psi_j^n\rangle\}_{j=1}^{5}$ listados no quadro da seção 6.3.2; devido ao volume de circuitos por estado ($3n$), à variedade de estados testados e de dispositivos usados, a maior parte dos experimentos foi realizada para $n = 3, 4, 5$ qubits (com exceção daqueles que usam bases fatoráveis, onde fomos até $n = 7$). Os procedimentos para a implementação dos experimentos foram os mesmos das simulações, descritos na lista da seção 6.1.3. A diferença está apenas no item 4, que pode ser substituído por:

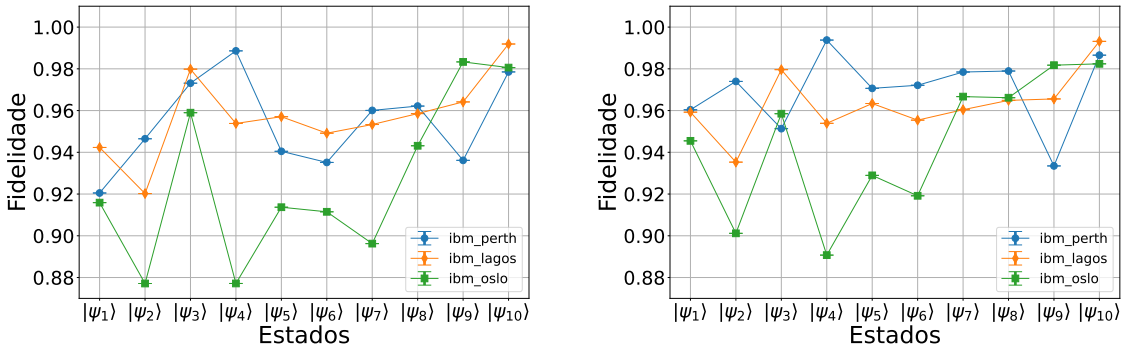
4'. Selecionamos o dispositivo da IBM onde o experimento será executado e enviamos o conjunto de circuitos por estado para a nuvem, através do Qiskit.[1]

Os circuitos para a pticografia foram transpilados conforme descrito na seção 6.1.1. Os experimentos que usaram QFT ou AQFT foram realizados sem as portas **SWAP** ao final do circuito. Os dados pticográficos gerados foram duplicados e uma cópia submetida à mitigação de erros de leitura, descrita na seção 6.1.2. Em todos os casos, implementamos o algoritmo de reconstrução com β variável, usando $\Delta\beta = 0{,}04$ (que implica em 50 iterações PIE). Os resultados obtidos foram avaliados através da fidelidade entre o estado de entrada e a estimativa do algoritmo (equação (6.3)).

[1] Para o caso de 2 qubits, nós enviamos de uma só vez os circuitos de todos os estados. Para n qubits, nós enviamos de uma só vez, para cada estado $|\psi_j^n\rangle$, os $3n$ circuitos para cada valor de n.

7.2 Resultados experimentais para estados de 2 qubits

Para o conjunto de estados de 2 qubits, nossos resultados são mostrados na figura 67. As fidelidades em 67(a) foram obtidas sem a mitigação dos erros de leitura, e ficaram acima 0,87. Claramente, os dispositivos ibm_lagos e ibm_perth apresentaram melhor performance, com fidelidades que variaram entre 0,92 e 0,99, com uma média sobre os estados de 0,957 e 0,954, respectivamente, em contraste com a média de 0,926 para a ibm_oslo. Com a mitigação de erros, podemos notar na figura 67(b) uma melhora das fidelidades na maioria dos casos; esta melhora é quantificada pela variação percentual entre as fidelidades antes e depois da mitigação de erros, como mostrado na figura 68. As médias alcançadas para os dispositivos ibm_lagos, ibm_perth e ibm_oslo após a mitigação foram de 0,963, 0,970 e 0,944, respectivamente.



(a) Sem mitigação.

(b) Com mitigação.

Figura 67 – Fidelidades médias para reconstruções pticográficas dos estados de 2 qubits $\{|\psi_j\rangle\}_{j=1}^{10}$ (veja o quadro da seção 6.2.2) implementadas experimentalmente nos dispositivos da IBM indicados nas legendas. Resultados sem (a) e com (b) mitigação de erros de leitura.

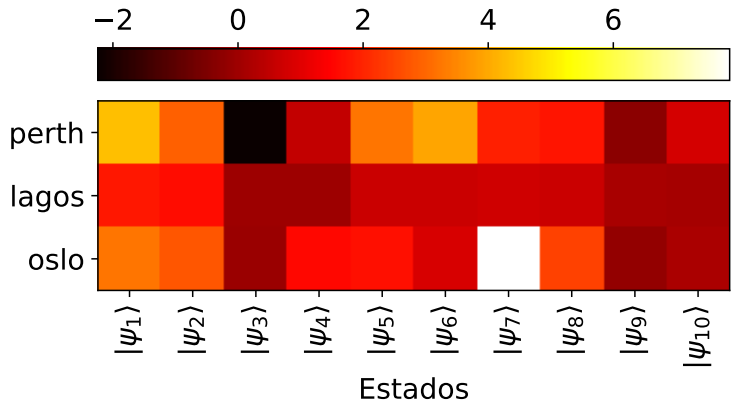


Figura 68 – Variação percentual das fidelidades antes e depois da mitigação de erros de leitura, obtidas dos gráficos da figura 67.

Em uma análise geral, obtivemos bons resultados, especialmente para dois dos dispositivos utilizados. Porém, as fidelidades experimentais ficaram abaixo daquelas obtidas através de simulações com ruído (figura 53). Além disso, embora a mitigação de erro de leitura tenha produzido melhores fidelidades, o seu efeito foi mais discreto em comparação com as simulações. As possíveis causas de tudo isso serão discutidas na seção 7.4.

7.3 Resultados experimentais para estados de n qubits

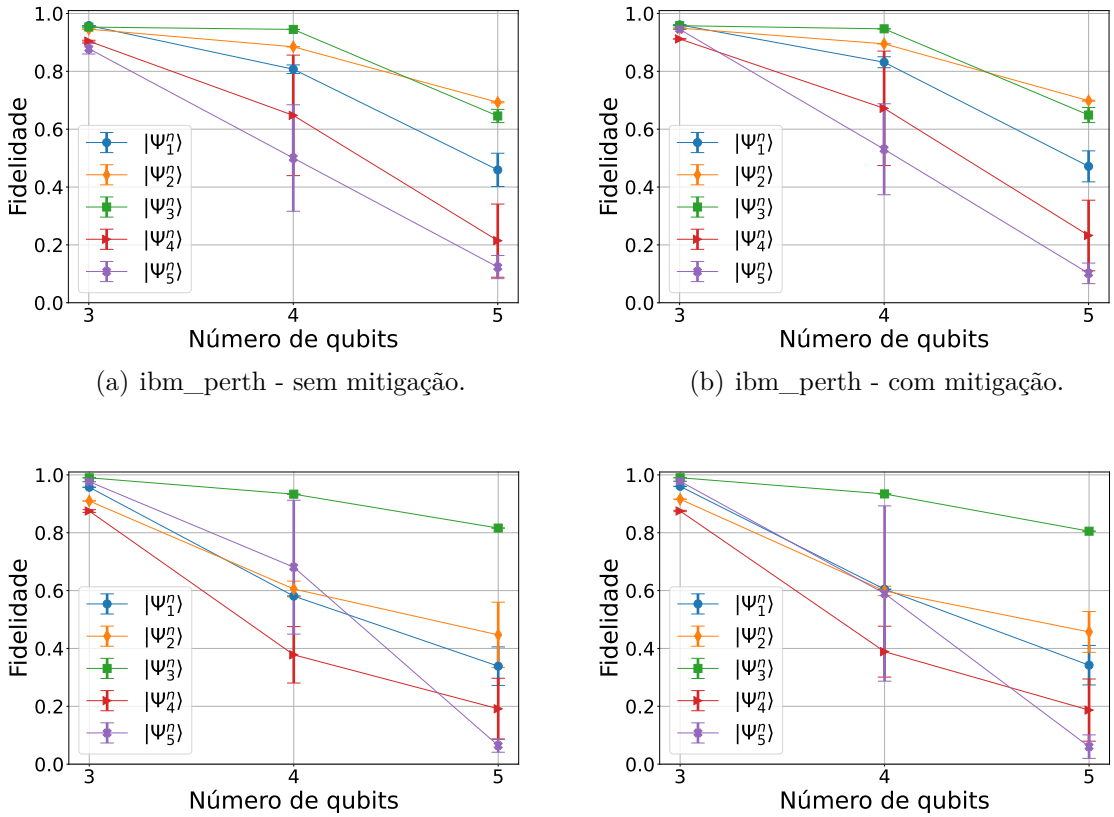
Para o conjunto de estados de $n > 2$ qubits os experimentos foram realizados nos dispositivos ibm_perth e ibm_lagos, que mostraram melhor performance no caso de 2 qubits. Os resultados experimentais usando a QFT são mostrados na figura 69, sem mitigação de erros (primeira coluna) e com mitigação (segunda coluna); cada linha corresponde à máquina utilizada. Apenas para o caso de $n = 3$ qubits tivemos bons resultados, os quais estão compilados na tabela 3. Para $n = 4$, a ibm_perth apresentou bons resultados somente para os estados separáveis; já na ibm_lagos, somente o estado separável aleatório $|\psi_3^n\rangle$ teve boas reconstruções. Este estado também teve reconstruções boas para $n = 5$. Nos demais casos, as fidelidades foram ruins. Especialmente para estados emaranhados $|\psi_4^n\rangle$ e $|\psi_5^n\rangle$ (GHZ e W, respectivamente), as fidelidades apresentam um forte decaimento a partir de $n = 4$, e este comportamento em comparação aos separáveis se deve à complexidade muito maior dos circuitos de preparação envolvidos (veja apêndice A).

Tabela 3 – Compilação dos resultados para $n = 3$ qubits.

	ibm_perth		**ibm_lagos**	
	sem mitigação	com mitigação	sem mitigação	com mitigação
F_{min}	0,878	0,912	0,875	0,875
F_{max}	0,958	0,961	0,990	0,990
$F_{\text{média}}$	0,928	0,945	0,942	0,943

Outro ponto que chama a atenção é a mitigação de erros de leitura. Ao comparar os resultados antes e depois da mitigação, vemos que o seu efeito é irrisório na maioria dos casos, ao contrário das simulações com ruído (figura 59). Para $n = 3$ isto pode ser visto claramente na tabela 3 para a ibm_lagos; no caso da ibm_perth o aumento da fidelidade média pós-mitigação é produzido principalmente pelo estado $|\psi_4^3\rangle$, cuja fidelidade aumenta de 0,878 para 0,945. O efeito da mitigação em termos da variação percentual das fidelidades é mostrado na figura 70; nesses gráficos nós filtramos os casos em que as fidelidades eram muito baixas (antes e depois) para evitar distorções.

Na seção 7.4 vamos tentar explicar as possíveis causas da performance ruim do método pticográfico para $n > 3$ nos computadores quânticos da IBM.



(a) ibm_perth - sem mitigação.

(b) ibm_perth - com mitigação.

(c) ibm_lagos - sem mitigação.

(d) ibm_lagos - com mitigação.

Figura 69 – Fidelidades médias para reconstruções pticográficas dos estados de n qubits $\{|\psi_j\rangle\}_{j=1}^{5}$ (veja o quadro da seção 6.3.2) implementadas experimentalmente nos dispositivos da IBM indicados abaixo dos gráficos. A primeira (segunda) coluna mostra os resultados sem (com) mitigação de erros de leitura.

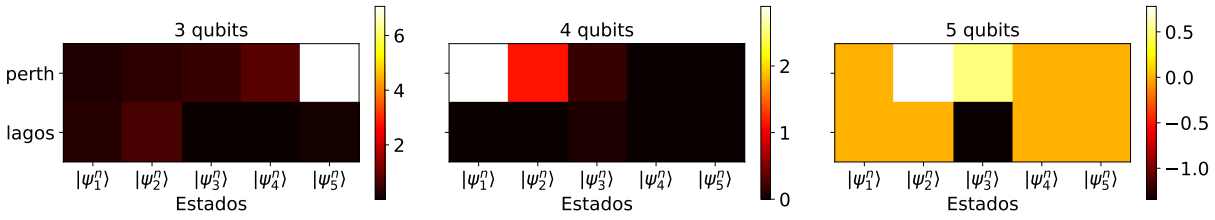
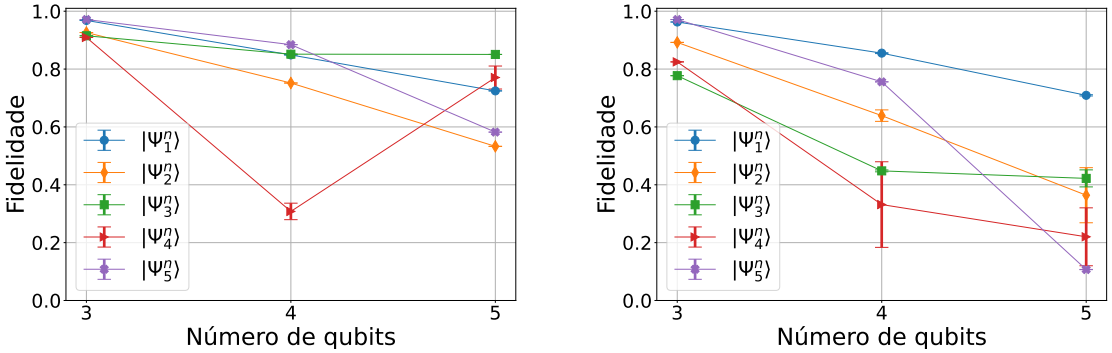


Figura 70 – Variação percentual das fidelidades antes e depois da mitigação de erros de leitura, mostradas na primeira e segunda coluna da figura 69, respectivamente.

Resultados com AQFT e bases fatoráveis

Nos experimentos com a AQFT usamos aproximações de grau $m = 2$ e $m = 3$ (veja seção 6.4.1). Na figura 71 mostramos os resultados obtidos com a ibm_lagos sem mitigação de erros de leitura; novamente, a mitigação de erros foi ineficaz para este dispositivo. Estes resultados devem ser comparados com aqueles mostrados na figura 69(c). Nesta comparação, observa-se o uso da AQFT com $m = 2$ produziu fidelidades um pouco

melhores para $n = 3$ e bem melhores para $n = 4$ e 5, com exceção do estado GHZ, que ficou ruim em ambos os casos. Para $m = 3$, os resultados foram similares àqueles mostrados em 69(c).



(a) $m = 2$ - sem mitigação.

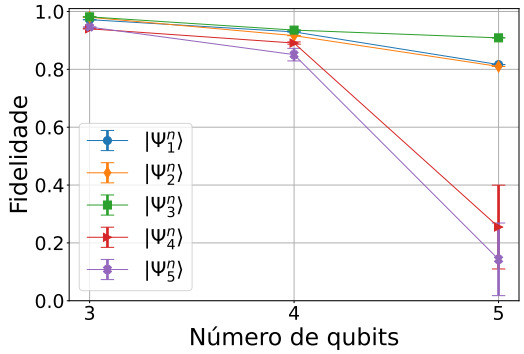
(b) $m = 3$ - sem mitigação.

Figura 71 – Resultados para AQFT obtidos no dispositivo ibm_lagos. Fidelidades médias para reconstruções pticográficas dos estados de n qubits $\{|\psi_j\rangle\}_{j=1}^{5}$ (veja o quadro da seção 6.3.2). O grau de aproximação m da AQFT usada é indicado abaixo de cada gráfico.

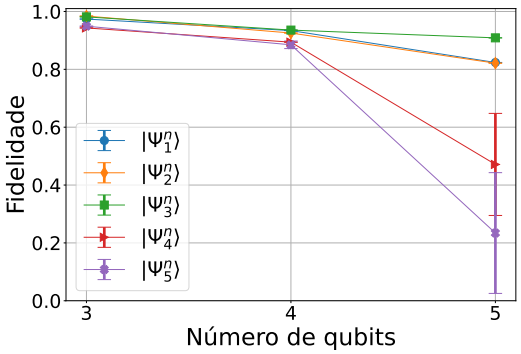
Os resultados mais interessantes com a AQFT foram obtidos na ibm_perth e são mostrados na figura 72, sem mitigação de erros (primeira coluna) e com mitigação (segunda coluna), cada linha corresponde à aproximação m utilizada. Para a AQFT com $m = 2$, observa-se nitidamente uma melhora em relação aos resultados mostrados nas figuras 69(a) e 69(b). Em particular, para $n = 3$ e $n = 4$ qubits obtivemos ótimas fidelidades conforme mostra a tabela 4; nesta tabela se vê novamente que a mitigação de erros teve pouca eficácia. Para $n = 5$, somente os estados separáveis apresentaram boas reconstruções. Para a AQFT com $m = 3$, houve uma melhora na maioria dos casos em comparação com o uso da QFT, mas os resultados não foram tão bons como no caso de $m = 2$. Podemos observar por esses resultados o impacto crescente do ruído, pois à medida que m aumenta, as fidelidades se tornaram piores. Além disso, eles indicam novamente o potencial de uso da AQFT para a pticografia.

Tabela 4 – Compilação dos resultados obtidos para reconstrução de estados de 3 e 4 qubits usando a AQFT com $m = 2$ na ibm_perth (figuras 72(a) e 72(b)).

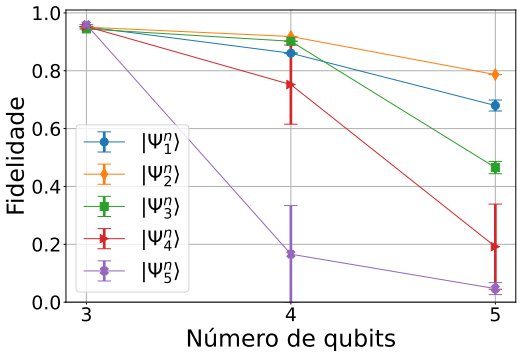
	ibm_perth ($m = 2$)			
	$n = 3$		$n = 4$	
	sem mitigação	com mitigação	sem mitigação	com mitigação
F_{min}	0,940	0,943	0,851	0,885
F_{max}	0,981	0,984	0,936	0,936
$F_{\text{média}}$	0,964	0,966	0,905	0,915



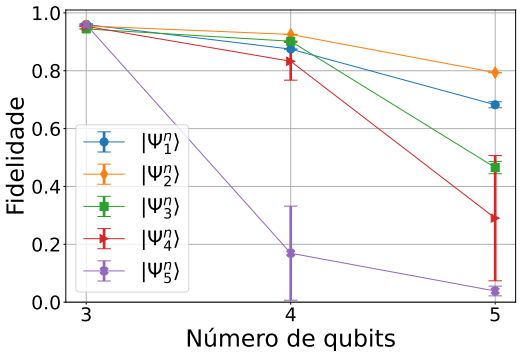
(a) $m = 2$ - sem mitigação.



(b) $m = 2$ - com mitigação.



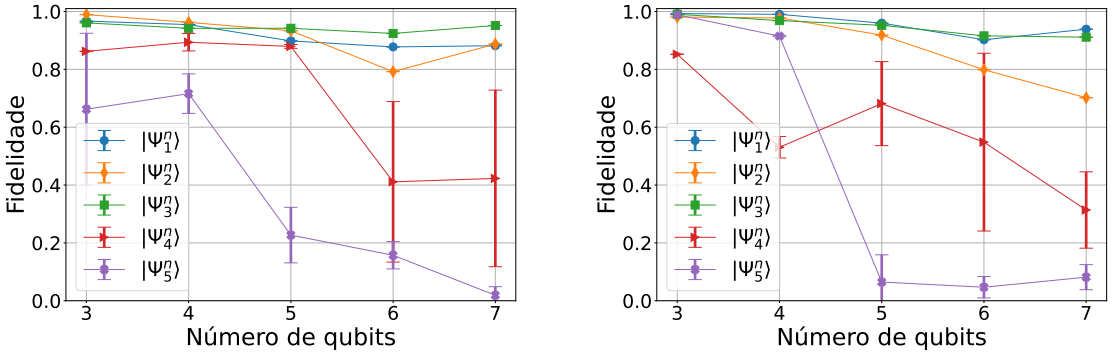
(c) $m = 3$ - sem mitigação.



(d) $m = 3$ - com mitigação.

Figura 72 – Resultados para AQFT obtidos no dispositivo ibm_perth. Fidelidades médias para reconstruções pticográficas dos estados de n qubits $\{|\psi_j\rangle\}_{j=1}^{5}$ (veja o quadro da seção 6.3.2). O grau de aproximação m da AQFT usada é indicado abaixo de cada gráfico. A primeira (segunda) coluna mostra os resultados sem (com) mitigação de erros de leitura.

Os últimos resultados que vamos apresentar neste trabalho foram obtidos com o uso de bases fatoráveis aleatórias para a medição final da pticografia, que discutimos na seção 6.4.2. Devido à baixa complexidade dos circuitos envolvidos para estes tipos de base, realizamos os experimentos para $n = 3, \ldots, 7$ qubits nos dispositivos ibm_perth e ibm_lagos. Os resultados obtidos são mostrados na figura 73. Assim como em nossas simulações (figura 66), as reconstruções dos estados separáveis foram, em geral, bem sucedidas, enquanto os estados emaranhados foram bem reconstruídos apenas em algumas situações. Na ibm_perth, observa-se que as fidelidades do estado W ($|\psi_5^n\rangle$) foram ruins em todos os casos, enquanto que para o estado GHZ ($|\psi_4^n\rangle$) foram boas para 3, 4 e 5 qubits. Na ibm_lagos, a situação praticamente se inverte, confirmando que ainda há muito que se estudar a respeito do uso de bases fatoráveis no método pticográfico.



(a) ibm_perth. (b) ibm_lagos.

Figura 73 – Resultados para bases aleatórias fatoráveis. Fidelidades médias para reconstruções pticográficas dos estados de n qubits $\{|\psi_j\rangle\}_{j=1}^{5}$ (veja o quadro da seção 6.3.2) implementadas experimentalmente nos dispositivos (a) ibm_perth e (b) ibm_lagos. Cada ponto corresponde à medição em uma base aleatória fatorável gerada conforme descrito na seção 6.4.2 Não houve mitigação de erros de leitura.

7.4 Análise dos resultados experimentais

Em uma análise geral dos nossos resultados experimentais, é nítido que eles ficaram abaixo das expectativas geradas pelos resultados das simulações com ruído apresentadas no capítulo anterior. Isto mostra que o ruído simulado é apenas uma aproximação, as vezes grosseira, dos erros que afetam um dispositivo real. Outra evidência disso é que, ao contrário do que ocorreu nas simulações, a mitigação de erros de leitura teve um baixíssimo impacto nos resultados experimentais. Ou seja, existem outras fontes de erros não captadas pela simulação que, em conjunto, são mais representativas que os erros de leitura sozinhos. Mesmo neste cenário ruidoso, nós obtivemos alguns resultados que merecem destaque, como as reconstruções para $n = 2$ e 3 qubits usando a QFT completa e as reconstruções para $n = 3$ e 4 qubits usando a AQFT com $m = 2$, onde todos os estados testados foram reconstruídos com boa/ótimas fidelidades.

Comparativo com o método PZD

O método PZD [52] também foi implementado experimentalmente nos computadores quânticos da IBM.[2] Os autores testaram um conjunto de quatro estados, três deles compatíveis com aqueles dos nossos conjuntos de teste: $|\psi_1^n\rangle$ e $|\psi_2^n\rangle$ (estados uniformes separáveis), e $|\psi_4^n\rangle$ (estado de Bell $|\psi_+\rangle$ e GHZ). Para os estados separáveis, os experimentos foram realizados com até 10 qubits usando bases fatoráveis, e até 4 qubits com bases não-fatoráveis; para o estado Bell/GHZ, o experimento foi realizado com até 4 qubits

[2] Na época, os autores utilizaram o ibm_manhattan e ibm_montreal, dispositivos já "aposentados" de 57 e 27 qubits, respectivamente.

nos dois casos. Vamos comparar os resultados para os casos de estados equivalentes e número de qubits correspondentes. Vale ressaltar que os experimentos foram realizados em dispositivos e tempos diferentes, e isso tem efeito sobre os resultados, como vimos neste capítulo e no anterior.

- Em relação aos estados separáveis, o método PZD com bases fatoráveis obteve fidelidades acima de 0,98 para todos os casos de $n = 2$ a 5, e ficaram acima dos nossos melhores resultados. Com as bases não-fatoráveis, o método PZD apresenta fidelidades melhores para 2 qubits, equivalentes para 3 qubits e muito piores para 4 qubits ($< 0{,}3$) em comparação com todos os nossos resultados.
- Para o estado de Bell/GHZ os nossos melhores resultados ficaram, em geral, acima dos resultados de PZD em ambos os casos de bases fatoráveis e não-fatoráveis, como mostra a tabela 5.

Tabela 5 – Fidelidades de reconstrução dos estados Bell/GHZ para $n = 2$ a 4 qubits pelo método PZD e pela pticografia quântica.

	Método PZD[a]		**Pticografia quântica**	
	Bases fatoráveis	Bases não-fatoráveis	QFT	AQFT ($m = 2$)
$\|\psi_+\rangle$	0,94	0,94	0,975	–
$\|GHZ_3\rangle$	0,90	0,90	0,912	0,943
$\|GHZ_4\rangle$	0,86	0,69	0,670	0,894

[a]Valores aproximados da fidelidade.

Por que os resultados pioram com o aumento do número de qubits?

Nos sub-circuitos das projeções intermediárias $\{\mathbf{\Pi}^{\pm}_{\xi j}\}$ e da medição projetiva final $\{\mathbf{F}^{\dagger}_n|j\rangle\}$ que constituem os circuitos pticográficos, a maior complexidade está no segundo, que requer o uso da QFT. O circuito da QFT, que já é naturalmente complexo, se torna ainda mais após o processo de transpilação (veja seção 3.1.2). Para verificar o efeito da transpilação do circuito da QFT nos dispositivos da IBM que utilizamos e a importância da otimização do processo, vamos considerar o caso de $n = 4$ qubits. A figura 74(a) ilustra o circuito virtual da QFT. Um processo de transpilação automática, realizado com baixo nível de otimização, poderia gerar o mapa de acoplamento mostrado na figura 74(b) e, consequentemente, o circuito transpilado da QFT mostrado abaixo, contendo 24 portas **CNOT**, 20 $\mathbf{R}_z$ (portas vermelhas) e 4 $\sqrt{\mathbf{X}}$ (portas verdes), cuja profundidade é $P = 40$. Uma transpilação com máximo nível de otimização, gera o mapa de acoplamento mostrado na figura 74(c) e o circuito mostrado abaixo, com 18 portas **CNOT**, 20 $\mathbf{R}_z$, 4 $\sqrt{\mathbf{X}}$ e $P = 37$. A redução é considerável, entretanto, o número de portas de 1 e 2 qubits e a profundidade do circuito ainda são significativos em comparação com o circuito virtual.

A tabela 6 mostra os números dos circuitos da QFT transpilados com máximo nível de otimização, em função do número de qubits.

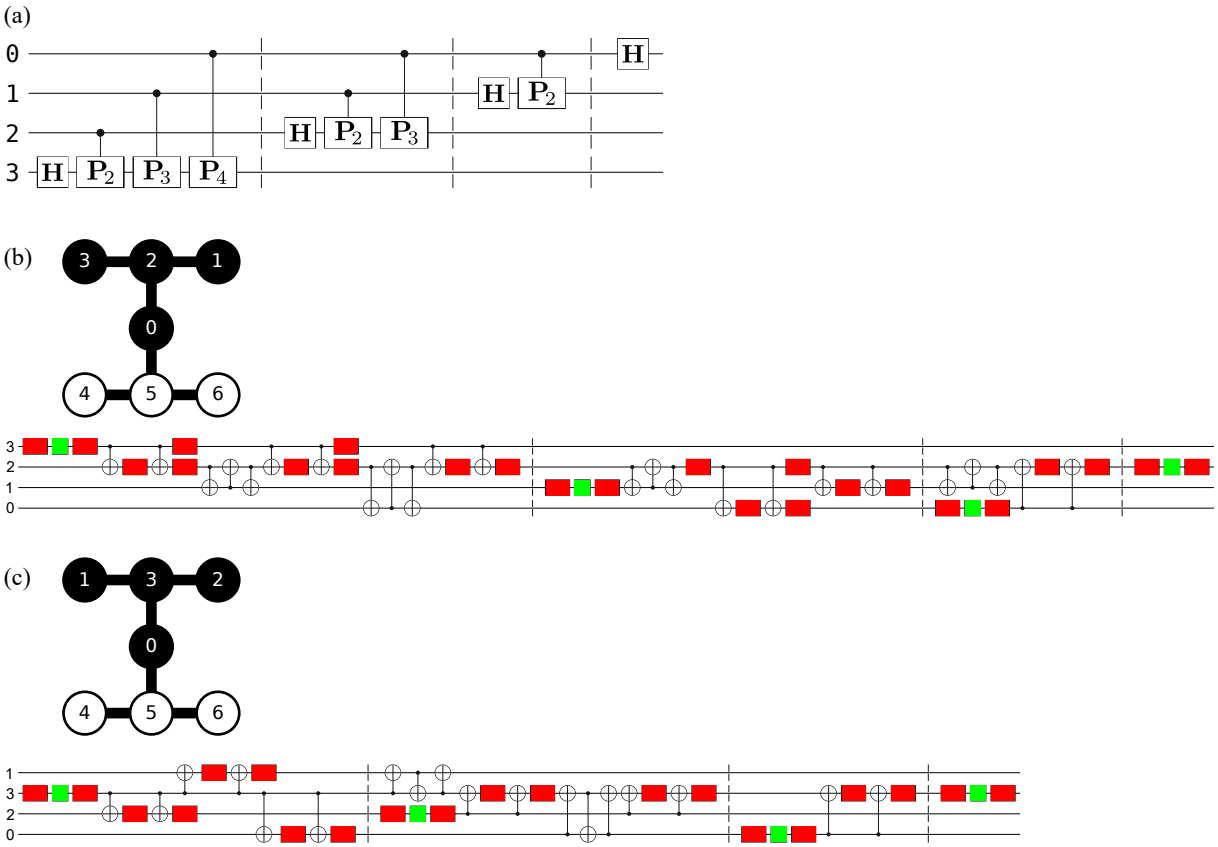


Figura 74 – Ilustração do circuito da QFT (sem **SWAP**s) para 4 qubits antes [(a)] e depois da transpilação [(b) e (c)]. Os retângulos verdes correspondem à porta $\sqrt{\mathbf{X}}$ e os vermelhos à portas $\mathbf{R}_z$ com ângulos específicos; as linhas verticais tracejadas separam o circuito em blocos. Em (b), uma transpilação com baixo nível de otimização gera o *layout* indicado no mapa de acoplamento e produz um circuito com 24 portas **CNOT**s, 24 $\mathbf{R}_z$ e 4 $\sqrt{\mathbf{X}}$. Em (c), a transpilação com nível máximo de otimização gera o *layout* indicado no mapa e produz um circuito com 18 portas **CNOT**s, 20 $\mathbf{R}_z$ e 4 $\sqrt{\mathbf{X}}$.

Tabela 6 – Número de portas e profundidade de um circuito da QFT transpilado com máximo nível de otimização.

	3 qubits	**4 qubits**	**5 qubits**	**6 qubits**	**7 qubits**
CNOT	9	18	32	54	71
$\mathbf{R}_z$	12	20	31	45	66
$\sqrt{\mathbf{X}}$	3	4	5	6	11
P	20	37	57	87	110

Como vimos na seção 3.1.3, o ruído tem efeito crescente com o aumento do número de portas de 2 qubits e a profundidade do circuito. Com isso, e observando os números da tabela 6, podemos entender o declínio acentuado das fidelidades na reconstrução pticográfica de estados de 4 ou mais qubits. Esse declínio é ainda mais nítido para os

estados emaranhados, os quais são mais afetados pelo ruído em sua preparação (veja apêndice A). É interessante observar que, mesmo neste cenário, o método pticográfico com a QFT completa apresentou resultados comparáveis ou melhores que o método PZD com bases fatoráveis e não-fatoráveis, conforme discutimos acima.

O efeito do ruído nos resultados obtidos com a QFT também foi confirmado com o uso de unitárias mais simples para executar a medição final. Obtivemos melhoras substanciais nas fidelidades com o uso da QFT aproximada, especialmente para $m = 2$, que produz circuitos (transpilados) com menos portas de 2 qubits e menor profundidade. Da mesma forma, o uso de bases fatoráveis, que geram circuitos sem portas de 2 qubits e de baixa profundidade, mostrou ótimos resultados para a reconstrução de estados separáveis de até 7 qubits; a dificuldade com estados emaranhados pode estar mais ligada às características da base do que com efeitos de ruído. Tendo em consideração todos esses aspectos, observamos que há boas perspectivas para o aprimoramento do método pticográfico voltado para a aplicação em dispositivos NISQ.

8 Conclusão

Nós estudamos a pticografia quântica aplicada à reconstrução de estados puros de n qubits. Nosso estudo se dividiu entre simulações e experimentos. As simulações foram realizadas através da ferramenta Qiskit e construídas através de códigos Python. Primeiramente, implementamos as simulações em um cenário livre de ruídos para uma grande diversidade de estados de até 10 qubits e considerando um número fixo de *shots*. Mostramos que quanto maior este número, mais próximo de 1 serão as fidelidades à medida que n cresce. Além disso, mostramos que na ausência de ruído, o método pticográfico apresentou melhores resultados que o método PZD para lidar com o tamanho finito do *ensemble*. Nas simulações com ruído, começamos com os estados de 2 qubits. Selecionamos um conjunto de 10 estados que incluía estados separáveis e emaranhados, entre eles os quatro estados de Bell. O método foi testado com ruído emulado de 9 dispositivos da IBM e considerando 4 cenários distintos. Obtivemos ótimas fidelidades, as quais melhoravam significativamente quando aplicávamos a mitigação de erros de leitura. Neste estudo, observamos que a variação do parâmetro β no algoritmo PIE, ao contrário de um valor fixo da proposta original, é essencial para sua convergência e a obtenção de melhores reconstruções; isto foi aplicado posteriormente nos demais estudos. As simulações com ruído foram estendidas a estados de até $n = 6$ qubits e executadas com ruído emulado de 3 máquinas. Selecionamos um conjunto diverso de 5 estados, incluindo separáveis e emaranhados (os importantes estados GHZ e W). Novamente, obtivemos ótimas fidelidades e mostramos a superioridade do método pticográfico sobre a tomografia com medições de Pauli. Fizemos também um estudo inicial sobre a utilização de bases de medição para a pticografia distintas daquela que usa a QFT. Testamos versões aproximadas da QFT e operações fatoráveis, onde obtivemos resultados bastante promissores mas que ainda requerem investigações mais aprofundadas. Os experimentos foram realizados remotamente nos computadores quânticos da IBM para $n = 2, 3, 4, 5$ qubits, seguindo os mesmos procedimentos das simulações com ruído. Para 2 e 3 qubits obtivemos ótimos resultados com o uso da QFT, e com o uso da AQFT chegamos a ótimos resultados com até 4 qubits. No comparativo com o método PZD, verificamos que os nossos melhores resultados foram equivalentes ou melhores, apesar de utilizarmos circuitos mais suscetíveis a ruído.

Além de demonstrar a pticografia quântica em um computador quântico e trazer novas contribuições para o método, nosso estudo deixa também várias possibilidades de investigações visando o seu aprimoramento, das quais destacamos:

- Estudar novas famílias de projeções intermediárias que possam oferecer uma reconstrução mais eficaz.

- Juntamente com o ítem anterior, explorar novas bases para a medição final que usem operações unitárias cujos circuitos sejam mais simples que o da QFT.
- Investigar técnicas avançadas de mitigação de erros, visando aprimorar a qualidade dos resultados.

Esse aprimoramento será essencial a curto prazo para que o método seja estendido a mais qubits em dispositivos NISQ e, futuramente, aplicado em computadores quânticos menos ruidosos. Esperamos que os nossos resultados possam ajudar a impulsionar não só o desenvolvimento de computadores quânticos, mas também de outras tecnologias quânticas.

Referências

[1] Moore, Gordon E: *Cramming more components onto integrated circuits*. Electronics, 38(8):114–117, 1965. Reprint: https://www.cs.utexas.edu/~fussell/courses/cs352h/papers/moore.pdf. Citado na página 10.

[2] Nielsen, Michael A e Isaac Chuang: *Quantum computation and quantum information*. American Association of Physics Teachers, 2002. Citado 13 vezes nas páginas 10, 11, 14, 18, 20, 21, 22, 24, 25, 46, 50, 59 e 61.

[3] Arute, Frank, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell *et al.*: *Quantum supremacy using a programmable superconducting processor*. Nature, 574(7779):505–510, 2019. https://www.nature.com/articles/s41586-019-1666-5. Citado na página 11.

[4] Zhong, Han Sen, Hui Wang, Yu Hao Deng, Ming Cheng Chen, Li Chao Peng, Yi Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao Yan Yang, Wei Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai Le Liu, Chao Yang Lu e Jian Wei Pan: *Quantum computational advantage using photons*. Science, 370(6523):1460–1463, 2020. https://www.science.org/doi/abs/10.1126/science.abe8770. Citado na página 11.

[5] Schlosshauer, Maximilian: *Quantum decoherence*. Physics Reports, 831:1–57, 2019, ISSN 0370-1573. https://www.sciencedirect.com/science/article/pii/S0370157319303084, Quantum decoherence. Citado na página 11.

[6] Fernandes, Mário Foganholi e Leonardo Neves: *Ptychography of pure quantum states*. Scientific Reports, 9(1):16066, Nov 2019. https://doi.org/10.1038/s41598-019-52415-y. Citado 12 vezes nas páginas 13, 52, 55, 56, 57, 59, 60, 63, 65, 70, 72 e 85.

[7] Barnett, Stephen: *Quantum information*, volume 16. Oxford University Press, 2009. Citado 3 vezes nas páginas 14, 37 e 46.

[8] Oliveira, Ivan S: *Física quântica: fundamentos, formalismo e aplicações*. Editora Livraria da Física, 2020. Citado 2 vezes nas páginas 14 e 46.

[9] Stallings, William: *Arquitetura e Organização de Computadores 8a Edição*. São Paulo: Prentice Hall do Brasil, 2010. Citado na página 14.

[10] Clarke, John e Frank K Wilhelm: *Superconducting quantum bits*. Nature, 453(7198):1031–1042, 2008. https://www.nature.com/articles/nature07128. Citado 2 vezes nas páginas 14 e 43.

[11] Kjaergaard, Morten, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I. J. Wang, Simon Gustavsson e William D. Oliver: *Superconducting Qubits: Current State of Play*. Annual Review of Condensed Matter Physics, 11(1):369–395, 2020. https://doi.org/10.1146/annurev-conmatphys-031119-050605. Citado na página 14.

[12] Ballon, Alvaro: *Quantum computing with superconducting qubits*, 2022. https://pennylane.ai/qml/demos/tutorial_sc_qubits.html. Citado 2 vezes nas páginas 14 e 44.

[13] Schindler, Philipp, Daniel Nigg, Thomas Monz, Julio T Barreiro, Esteban Martinez, Shannon X Wang, Stephan Quint, Matthias F Brandl, Volckmar Nebendahl, Christian F Roos, Michael Chwalla, Markus Hennrich e Rainer Blatt: *A quantum information processor with trapped ions*. New Journal of Physics, 15(12):123012, dec 2013. https://dx.doi.org/10.1088/1367-2630/15/12/123012. Citado na página 14.

[14] Fernandes, Gabriel P. L. M., Alexandre C. Ricardo, Fernando R. Cardoso e Celso J. Villas-Boas: *Íons Aprisionados como Arquitetura para Computação Quântica*. Revista Brasileira de Ensino de Física, 45(Rev. Bras. Ensino Fís., 2023 45):e20220218, 2023, ISSN 1806-1117. https://doi.org/10.1590/1806-9126-RBEF-2022-0218. Citado na página 14.

[15] Takeda, S. e A. Furusawa: *Toward large-scale fault-tolerant universal photonic quantum computing*. APL Photonics, 4(6):060902, 2019. https://doi.org/10.1063/1.5100160. Citado na página 14.

[16] Slussarenko, Sergei e Geoff J. Pryde: *Photonic quantum information processing: A concise review*. Applied Physics Reviews, 6(4):041303, 2019. https://doi.org/10.1063/1.5115814. Citado na página 14.

[17] Deutsch, David e Roger Penrose: *Quantum theory, the Church–Turing principle and the universal quantum computer*. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 400(1818):97–117, 1985. https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1985.0070. Citado na página 14.

[18] Deutsch, David e Richard Jozsa: *Rapid solution of problems by quantum computation*. Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences, 439(1907):553–558, 1992. https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1992.0167. Citado na página 14.

[19] Carvalho, L.M., C. Lavor e V.S. Motta: *Caracterização Matemática e Visualização da Esfera de Bloch: Ferramentas para Computação Quântica*. Trends in Computational and Applied Mathematics, 8(3):351–360, 2007, ISSN 2676-0029. https://tema.sbmac.org.br/tema/article/view/196. Citado na página 14.

[20] Bell, J. S. e Alain Aspect: *Speakable and Unspeakable in Quantum Mechanics: Collected Papers on Quantum Philosophy*. Cambridge University Press, 2ª edição, 2004. https://doi.org/10.1017/CBO9780511815676. Citado na página 16.

[21] Bennett, Charles H. e Stephen J. Wiesner: *Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states*. Phys. Rev. Lett., 69:2881–2884, Nov 1992. https://link.aps.org/doi/10.1103/PhysRevLett.69.2881. Citado na página 16.

[22] Bennett, Charles H., Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres e William K. Wootters: *Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels*. Phys. Rev. Lett., 70:1895–1899, Mar 1993. https://link.aps.org/doi/10.1103/PhysRevLett.70.1895. Citado na página 16.

[23] Żukowski, M. , A. Zeilinger, M. A. Horne e A. K. Ekert: *"Event-ready-detectors" Bell experiment via entanglement swapping*. Phys. Rev. Lett., 71:4287–4290, Dec 1993. https://link.aps.org/doi/10.1103/PhysRevLett.71.4287. Citado na página 16.

[24] Greenberger, Daniel M., Michael A. Horne, Abner Shimony e Anton Zeilinger: *Bell's theorem without inequalities*. American Journal of Physics, 58(12):1131–1143, 1990. https://doi.org/10.1119/1.16243. Citado na página 16.

[25] Dür, W., G. Vidal e J. I. Cirac: *Three qubits can be entangled in two inequivalent ways*. Phys. Rev. A, 62:062314, Nov 2000. https://link.aps.org/doi/10.1103/PhysRevA.62.062314. Citado na página 16.

[26] Coffman, Valerie, Joydip Kundu e William K. Wootters: *Distributed entanglement*. Phys. Rev. A, 61:052306, Apr 2000. https://link.aps.org/doi/10.1103/PhysRevA.61.052306. Citado na página 16.

[27] Cabello, Adán: *Bell's theorem with and without inequalities for the three-qubit Greenberger-Horne-Zeilinger and W states*. Phys. Rev. A, 65:032108, Feb 2002. https://link.aps.org/doi/10.1103/PhysRevA.65.032108. Citado na página 16.

[28] D'Hondt, Ellie e Prakash Panangaden: *The Computational Power of the W And GHZ States*. Quantum Info. Comput., 6(2):173–183, mar 2006, ISSN 1533-7146. https://dl.acm.org/doi/10.5555/2011665.2011668. Citado na página 16.

[29] Jung, Eylee, Mi Ra Hwang, You Hwan Ju, Min Soo Kim, Sahng Kyoon Yoo, Hungsoo Kim, DaeKil Park, Jin Woo Son, S. Tamaryan e Seong Keuck Cha: *Greenberger-Horne-Zeilinger versus W states: Quantum teleportation through noisy channels*. Phys. Rev. A, 78:012312, Jul 2008. https://link.aps.org/doi/10.1103/PhysRevA.78.012312. Citado na página 16.

[30] Li, Na, Jian Li, Lei Lei Li, Zheng Wang e Tao Wang: *Deterministic Secure Quantum Communication and Authentication Protocol based on Extended GHZ-W State and Quantum One-time Pad*. International Journal of Theoretical Physics, 55(8):3579–3587, Aug 2016. https://doi.org/10.1007/s10773-016-2986-y. Citado na página 16.

[31] Hua, Fei, Yuwei Jin, Yanhao Chen, Suhas Vittal, Kevin Krsulich, Lev S. Bishop, John Lapeyre, Ali Javadi-Abhari e Eddy Z. Zhang: *Exploiting Qubit Reuse through Mid-circuit Measurement and Reset*, 2023. https://arxiv.org/abs/2211.01925. Citado na página 24.

[32] *Transpiler*. https://qiskit.org/documentation/apidoc/transpiler.html. Acessado: 2022-12-27. Citado na página 35.

[33] J., Abhijith, Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Alexander Malyzhenkov, David Mascarenas, Susan Mniszewski, Balu Nadiga, Daniel O'malley, Diane Oyen, Scott Pakin, Lakshman Prasad, Randy Roberts, Phillip Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter J. Swart, James G. Wendelberger, Boram Yoon, Richard Zamora, Wei Zhu, Stephan Eidenbenz, Andreas Bärtschi, Patrick J. Coles, Marc Vuffray e Andrey Y. Lokhov: *Quantum Algorithm Implementations for Beginners*. ACM Transactions on Quantum Computing, 3(4), jul 2022, ISSN 2643-6809. https://doi.org/10.1145/3517340. Citado 2 vezes nas páginas 37 e 61.

[34] Jesus, Gleydson Fernandes de, Maria Heloísa Fraga da Silva, Teonas Gonçalves Dourado Netto, Lucas Queiroz Galvão, Frankle Gabriel de Oliveira Souza e Clebson Cruz: *Computação quântica: uma abordagem para a graduação usando o Qiskit*. Revista Brasileira de Ensino de Física, 43:e20210033, 2021, ISSN 1806-1117. https://doi.org/10.1590/1806-9126-RBEF-2021-0033. Citado na página 40.

[35] Delft, Dirk van e Peter Kes: *The discovery of superconductivity*. Physics Today, 63(9):38–43, setembro 2010, ISSN 0031-9228. https://doi.org/10.1063/1.3490499. Citado na página 42.

[36] Costa, Marconi B.S. e Antonio C. Pavão: *Supercondutividade: um século de desafios e superação*. Revista Brasileira de Ensino de Física, 34(2):2602–2615, Apr 2012,

ISSN 1806-1117. https://doi.org/10.1590/S1806-11172012000200017. Citado na página 42.

[37] JOSEPHSON, B. D.: *Coupled Superconductors*. Rev. Mod. Phys., 36:216–220, Jan 1964. https://link.aps.org/doi/10.1103/RevModPhys.36.216. Citado na página 42.

[38] Krantz, P., M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson e W. D. Oliver: *A quantum engineer's guide to superconducting qubits*. Applied Physics Reviews, 6(2):021318, junho 2019. https://doi.org/10.1063/1.5089550. Citado na página 43.

[39] Heya, Kentaro e Naoki Kanazawa: *Cross-Cross Resonance Gate*. PRX Quantum, 2:040336, Nov 2021. https://link.aps.org/doi/10.1103/PRXQuantum.2.040336. Citado na página 44.

[40] Tripathi, Vinay, Mostafa Khezri e Alexander N. Korotkov: *Operation and intrinsic error budget of a two-qubit cross-resonance gate*. Phys. Rev. A, 100:012301, Jul 2019. https://link.aps.org/doi/10.1103/PhysRevA.100.012301. Citado na página 44.

[41] *Lectures and Tutorials*. https://www.zlatko-minev.com/education. Acessado: 2023-09-25. Citado na página 44.

[42] Olson, Eric: *How quantum computers work*, mar 2019. https://electronics360.globalspec.com/article/13553/how-quantum-computers-work. Citado na página 45.

[43] Toninelli, Ermes, Bienvenu Ndagano, Adam Vallés, Bereneice Sephton, Isaac Nape, Antonio Ambrosio, Federico Capasso, Miles J. Padgett e Andrew Forbes: *Concepts in quantum state tomography and classical implementation with intense light: a tutorial*. Adv. Opt. Photon., 11(1):67–134, Mar 2019. https://opg.optica.org/aop/abstract.cfm?URI=aop-11-1-67. Citado na página 49.

[44] Altepeter, Joseph B., Daniel F.V. James e Paul G. Kwiat: *Qubit Quantum State Tomography*, páginas 113–145. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. https://doi.org/10.1007/978-3-540-44481-7_4. Citado 2 vezes nas páginas 50 e 51.

[45] Kaznady, Max S. e Daniel F. V. James: *Numerical strategies for quantum tomography: Alternatives to full optimization*. Phys. Rev. A, 79:022109, Feb 2009. https://link.aps.org/doi/10.1103/PhysRevA.79.022109. Citado na página 51.

[46] Shang, Jiangwei, Zhengyun Zhang e Hui Khoon Ng: *Superfast maximum-likelihood reconstruction for quantum tomography*. Phys. Rev. A, 95:062336, Jun 2017. https:

//link.aps.org/doi/10.1103/PhysRevA.95.062336. Citado 2 vezes nas páginas 51 e 81.

[47] Flammia, Steven T, Andrew Silberfarb e Carlton M Caves: *Minimal informationally complete measurements for pure states*. Foundations of Physics, 35:1985–2006, 2005. https://link.springer.com/article/10.1007/s10701-005-8658-z. Citado na página 52.

[48] Goyeneche, D., G. Cañas, S. Etcheverry, E. S. Gómez, G. B. Xavier, G. Lima e A. Delgado: *Five Measurement Bases Determine Pure Quantum States on Any Dimension*. Phys. Rev. Lett., 115:090401, Aug 2015. https://link.aps.org/doi/10.1103/PhysRevLett.115.090401. Citado na página 52.

[49] Ma, Xian, Tyler Jackson, Hui Zhou, Jianxin Chen, Dawei Lu, Michael D. Mazurek, Kent A. G. Fisher, Xinhua Peng, David Kribs, Kevin J. Resch, Zhengfeng Ji, Bei Zeng e Raymond Laflamme: *Pure-state tomography with the expectation value of Pauli operators*. Phys. Rev. A, 93:032140, Mar 2016. https://link.aps.org/doi/10.1103/PhysRevA.93.032140. Citado na página 52.

[50] Pears Stefano, Quimey, Lorena Rebón, Silvia Ledesma e Claudio Iemmi: *Determination of any pure spatial qudits from a minimum number of measurements by phase-stepping interferometry*. Phys. Rev. A, 96:062328, Dec 2017. https://link.aps.org/doi/10.1103/PhysRevA.96.062328. Citado na página 52.

[51] Fernandes, M. F., M. A. Solís-Prosser e L. Neves: *Ptychographic reconstruction of pure quantum states*. Opt. Lett., 45(21):6002–6005, Nov 2020. https://opg.optica.org/ol/abstract.cfm?URI=ol-45-21-6002. Citado 2 vezes nas páginas 52 e 59.

[52] Pereira, Luciano, Leonardo Zambrano e Aldo Delgado: *Scalable estimation of pure multi-qubit states*. npj Quantum Information, 8(1):57, 2022. https://www.nature.com/articles/s41534-022-00565-9#Sec2. Citado 8 vezes nas páginas 52, 53, 64, 78, 79, 80, 81 e 97.

[53] Faulkner, H. M. L. e J. M. Rodenburg: *Movable Aperture Lensless Transmission Microscopy: A Novel Phase Retrieval Algorithm*. Phys. Rev. Lett., 93:023903, Jul 2004. https://link.aps.org/doi/10.1103/PhysRevLett.93.023903. Citado 4 vezes nas páginas 55, 57, 59 e 65.

[54] Rodenburg, J. M. e H. M. L. Faulkner: *A phase retrieval algorithm for shifting illumination*. Applied Physics Letters, 85(20):4795–4797, novembro 2004, ISSN 0003-6951. https://doi.org/10.1063/1.1823034. Citado 3 vezes nas páginas 55, 57 e 59.

[55] Thibault, Pierre, Martin Dierolf, Andreas Menzel, Oliver Bunk, Christian David e Franz Pfeiffer: *High-Resolution Scanning X-ray Diffraction Microscopy*. Science,

321(5887):379–382, 2008. https://www.science.org/doi/abs/10.1126/science.1158573. Citado na página 55.

[56] Jiang, Yi, Zhen Chen, Yimo Han, Pratiti Deb, Hui Gao, Saien Xie, Prafull Purohit, Mark W Tate, Jiwoong Park, Sol M Gruner *et al.*: *Electron ptychography of 2D materials to deep sub-ångström resolution*. Nature, 559(7714):343–349, 2018. https://www.nature.com/articles/s41586-018-0298-5. Citado na página 55.

[57] Goodman, Joseph W: *Introduction to Fourier optics*. Roberts and Company publishers, 2005. Citado na página 55.

[58] Maiden, Andrew, Daniel Johnson e Peng Li: *Further improvements to the ptychographical iterative engine*. Optica, 4(7):736–745, Jul 2017. https://opg.optica.org/optica/abstract.cfm?URI=optica-4-7-736. Citado na página 59.

[59] *Quantum Fourier Transform*. https://learn.qiskit.org/course/ch-algorithms/quantum-fourier-transform. Acessado: 2023-10-05. Citado na página 61.

[60] Stockmar, Marco, Peter Cloetens, Irene Zanette, Bjoern Enders, Martin Dierolf, Franz Pfeiffer e Pierre Thibault: *Near-field ptychography: phase retrieval for inline holography using a structured illumination*. Scientific reports, 3(1):1927, 2013. https://www.nature.com/articles/srep01927. Citado na página 65.

[61] Bravyi, Sergey, Sarah Sheldon, Abhinav Kandala, David C. Mckay e Jay M. Gambetta: *Mitigating measurement errors in multiqubit experiments*. Phys. Rev. A, 103:042605, Apr 2021. https://link.aps.org/doi/10.1103/PhysRevA.103.042605. Citado na página 67.

[62] Nation, Paul D., Hwajung Kang, Neereja Sundaresan e Jay M. Gambetta: *Scalable Mitigation of Measurement Errors on Quantum Computers*. PRX Quantum, 2:040326, Nov 2021. https://link.aps.org/doi/10.1103/PRXQuantum.2.040326. Citado na página 67.

[63] Kim, Youngseok, Andrew Eddins, Sajant Anand, Ken Xuan Wei, Ewout Van Den Berg, Sami Rosenblatt, Hasan Nayfeh, Yantao Wu, Michael Zaletel, Kristan Temme *et al.*: *Evidence for the utility of quantum computing before fault tolerance*. Nature, 618(7965):500–505, 2023. https://www.nature.com/articles/s41586-023-06096-3. Citado na página 67.

[64] Neves, Juliana Ramos: *Hamiltonian Design of Quantum Gates*. Tese de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte, 2022. Citado na página 73.

[65] Barenco, Adriano, Artur Ekert, Kalle Antti Suominen e Päivi Törmä: *Approximate quantum Fourier transform and decoherence*. Phys. Rev. A, 54:139–146, Jul 1996. https://link.aps.org/doi/10.1103/PhysRevA.54.139. Citado na página 85.

[66] Cruz, Diogo, Romain Fournier, Fabien Gremion, Alix Jeannerot, Kenichi Komagata, Tara Tosic, Jarla Thiesbrummel, Chun Lam Chan, Nicolas Macris, Marc André Dupertuis e Clément Javerzac-Galy: *Efficient Quantum Algorithms for GHZ and W States, and Implementation on the IBM Quantum Computer*. Advanced Quantum Technologies, 2(5-6):1900015, 2019. https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.201900015. Citado na página 112.

[67] Shende, V.V., S.S. Bullock e I.L. Markov: *Synthesis of quantum-logic circuits*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(6):1000–1010, 2006. https://ieeexplore.ieee.org/document/1629135. Citado na página 113.

APÊNDICE A – Preparação de estados de n qubits

Nos dispositivos da IBM, os n qubits de um circuito a ser implementado são inicializados no estado $|0\rangle^{\otimes n}$. Para prepararmos um estado puro qualquer $|\psi\rangle$ a partir desse estado inicial, devemos aplicar um conjunto apropriado de portas lógicas representado por $\mathbf{U}_n^{\text{prep}}$, tal que $\mathbf{U}_n^{\text{prep}}|0\rangle^{\otimes n} = |\psi\rangle$. Na seção 2.4.2, vimos exemplos de circuitos para preparação de estados emaranhados de 2 e 3 qubits, mais especificamente os estados de Bell (figura 9) e os estados GHZ e W de 3 qubits (figura 10), respectivamente. Além desses, o método pticográfico foi testado para diversos outros estados quânticos, separáveis e emaranhados, de até 10 qubits. Neste apêndice, nós vamos discutir a preparação e mostrar os circuitos correspondentes para alguns desses estados.

Preparação de estados separáveis: uniformes e aleatórios

Em um circuito quântico, os estados separáveis podem ser preparados usando apenas portas de um qubit. Nós utilizamos dois tipos de estados separáveis: uniformes e aleatórios. Os estados uniformes separáveis de n qubits são caracterizados por coeficientes na base computacional que satisfazem $|\alpha_j| = |\langle j|\psi\rangle| = 1/\sqrt{2^n}\ \forall j$ e podem diferir apenas nas fases relativas. Dois exemplos de estados uniformes que foram testados são

$$|\psi_1^n\rangle = (\mathbf{TH})^{\otimes n}|0\rangle^{\otimes n} = \left(\frac{|0\rangle + e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes n}, \tag{A.1}$$

$$|\psi_2^n\rangle = (\mathbf{THX})^{\otimes n}|0\rangle^{\otimes n} = \left(\frac{|0\rangle - e^{i\pi/4}|1\rangle}{\sqrt{2}}\right)^{\otimes n}, \tag{A.2}$$

cujos circuitos estão representandos na figura 75(a) e 75(b), respectivamente. Os estados aleatórios separáveis são preparados usando portas de um qubit $\mathbf{U}(\theta, \phi, \lambda)$ (veja a equação (2.26)) em cada qubit do circuito, tal que os ângulos de entrada θ, ϕ e λ são gerados aleatoriamente. A figura 75(c) mostra o circuito correspondente.

Os circuitos para esses três estados, ao serem implementados em um computador da IBM, passam por uma transpilação que resulta em uma sequência de portas-base, conforme ilustrado na figura 76. Assim, a profundidade que anteriormente era $P = 2$ na figura 75(a), $P = 3$ na figura 75(b), e $P = 1$ na figura 75(c), passa a ser $P = 3$, $P = 3$, e $P = 5$, respectivamente. Em todos os casos, o número de fatores tensoriais é $T = n$.

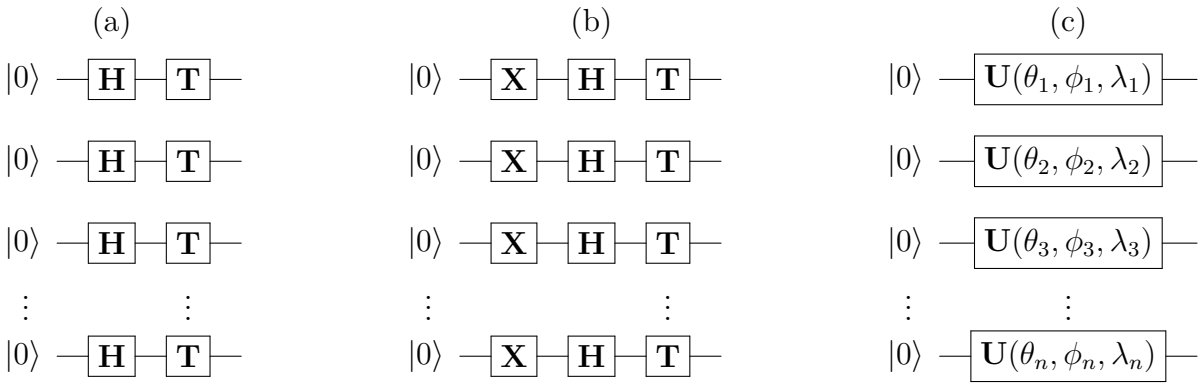


Figura 75 – Circuitos de preparação dos estados uniformes (a) $|\psi_1^n\rangle$ e (b) $|\psi_2^n\rangle$ de n qubits. Em (c) temos o circuito de um estado separável aleatório.

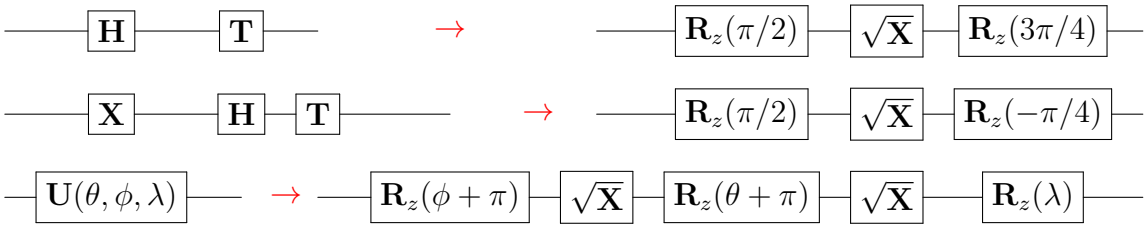


Figura 76 – Decomposição da sequência de portas da figura 75 em termos das portas-base da IBM.

Preparação de estados emaranhados: GHZ e W

A preparação do estado GHZ de n qubits é usualmente definida como:

$$|GHZ\rangle_n = \mathbf{CNOT}_{0,n-1}\cdots\mathbf{CNOT}_{0,2}\mathbf{CNOT}_{0,1}\mathbf{H}_0|0\rangle^{\otimes n} = \frac{|0\rangle^{\otimes n}+|1\rangle^{\otimes n}}{\sqrt{2}}, \tag{A.3}$$

e o circuito correspondente é mostrado na figura 77(a). No entanto, é fácil verificar que este mesmo estado pode ser gerado por

$$|GHZ\rangle_n = \mathbf{CNOT}_{n-2,n-1}\cdots\mathbf{CNOT}_{1,2}\mathbf{CNOT}_{0,1}\mathbf{H}_0|0\rangle^{\otimes n} = \frac{|0\rangle^{\otimes n}+|1\rangle^{\otimes n}}{\sqrt{2}}, \tag{A.4}$$

cujo circuito é mostrado na figura 77(b). Nós usamos esse método pois ele favorece a preparação em circuitos com conectividade restrita, minimizando a quantidade de **SWAP**s. Vale ressaltar que em qualquer caso o circuito do GHZ sempre terá $T = 1$ e quando otimizado sua profundidade miníma é $P = n$.

Uma discussão detalhada de como otimizar o circuito de preparação dos estados GHZ e W para n qubits pode ser encontrada em [66]. De acordo com este trabalho, uma possível maneira de preparar o estado $|W\rangle_n$ é dada por

$$\begin{aligned}|W\rangle_n &= \mathbf{CNOT}_{n-1,n-2}\mathbf{R}_z(\theta_{n-1})_{n-1}\mathbf{P}^{\text{ctrl}}_{n-2,n-1}(\pi)\mathbf{R}_z(-\theta_{n-1})_{n-1}\cdots \\ &\quad\cdots\mathbf{CNOT}_{2,1}\mathbf{R}_z(\theta_2)_2\mathbf{P}^{\text{ctrl}}_{1,2}(\pi)\mathbf{R}_z(-\theta_2)_2\mathbf{CNOT}_{1,0}\mathbf{R}_z(\theta_1)_1\mathbf{P}^{\text{ctrl}}_{0,1}(\pi)\mathbf{R}_z(-\theta_1)_1\mathbf{X}_0|0\rangle^{\otimes n} \\ &= \frac{1}{\sqrt{n}}\left(|100\ldots0\rangle + |010\ldots0\rangle + \cdots + |000\ldots1\rangle\right),\end{aligned} \tag{A.5}$$

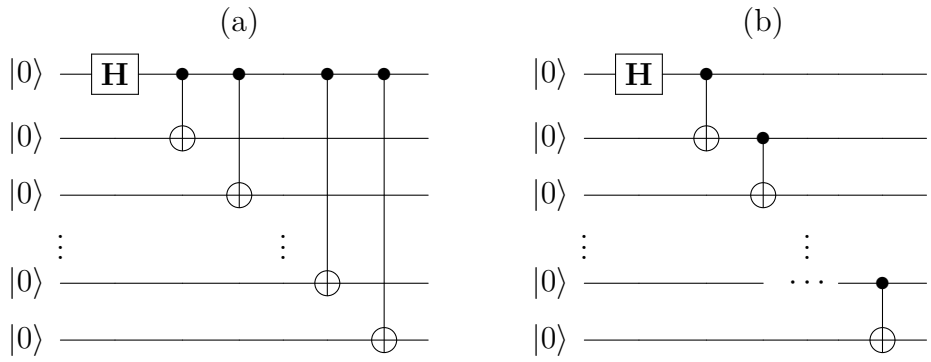


Figura 77 – Circuitos de preparação do estado GHZ para n qubits definido pela equação (A.3) (a) equação (A.4) (b).

onde $\theta_k = \sqrt{1/(n-k)}$. Com base neste resultado, construímos nossos circuitos para o estado W. O número de fatores tensoriais é $T = 1$ e quando otimizado sua profundidade miníma é $P = 3n - 2$. A figura 78 mostra um exemplo do circuito para 4 qubits.

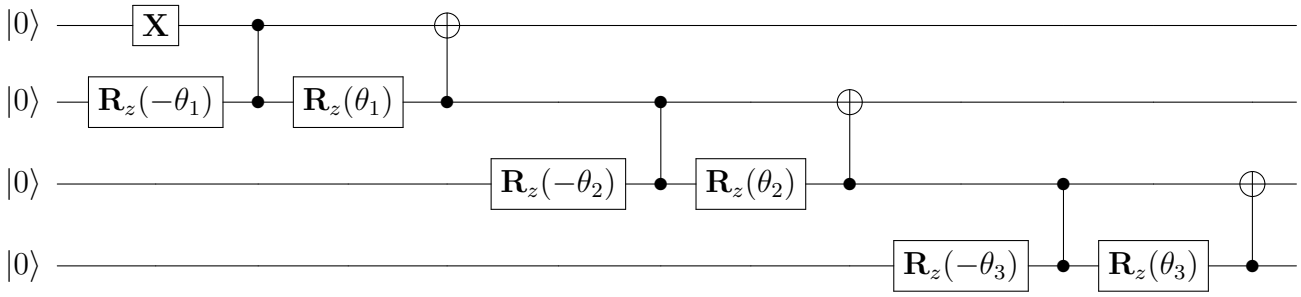


Figura 78 – Circuito de preparação do estado W para 4 qubits.

Preparação de estados arbitrários e aleatórios pelo método *initalize*

Através do Qiskit podemos preparar estados arbitrários ou aleatórios em um circuito usando o método *initialize*.[1] Por exemplo, para gerar um estado $a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$ normalizado, criamos um vetor de números complexos $[a, b, c, d]$ e para gerar um estado aleatório, utilizamos a função *random_statevector*, que cria um vetor de números complexos aleatórios correspondentes às amplitudes de probabilidade do estado, devidamente normalizadas. Nos dois casos, enviamos o vetor para o *initialize* que construirá o circuito de preparação do estado. O artifício por trás do *initialize* foi proposto em [67]: um estado multi-qubit qualquer é preparado por meio de um conjunto/sequência portas lógicas que vão codificar as amplitudes e as fases dos seus coeficientes.

Além dos estados descritos anteriormente, nós também testamos a pticografia quântica para estados gerados pelo método *initialize*, especialmente para estados aleató-

[1] Veja: https://qiskit.org/documentation/locale/ta_IN/stubs/qiskit.circuit.QuantumCircuit.initialize.html. Um tutorial mais didático pode ser consultado em https://www.youtube.com/watch?v=tOWV94-ydL4

rios. É importante ressaltar que, diferetemente dos estados aleatórios separáveis que nós preparamos manualmente, os estados aleatórios gerados pelo *initialize* pertencerão, com grande probabilidade, ao conjunto dos estados emaranhados.

APÊNDICE B – **Códigos Python para a pticografia quântica**

Pticografia quântica com 2 qubits

Nesta seção, apresentamos o código que implementa a pticografia quântica para estados de 2 qubits. Este código é referente à simulação com ruído, onde escolhemos uma máquina e simulamos a reconstrução dos 10 estados listados no quadro da seção 6.2.2.

```
# Bibliotecas necessárias
from qiskit import *
from qiskit.providers.aer import AerSimulator
from qiskit.providers.aer.noise import NoiseModel
from qiskit.ignis.mitigation.measurement import complete_meas_cal, CompleteMeasFitter
import numpy as np
from numpy import pi

# Conectado na IBM Quantum:
IBMQ.enable_account('Cole seu token aqui')
provider = IBMQ.get_provider(hub='ibm-q', group='open', project='main')

# Escolha o dispositivo:
maquina = 'ibm_perth'
backend = provider.get_backend(f'{maquina}')

# Escolha o número de shots:
SHOTS = 100000

# Escolha o layout inicial para a pticografia quântica e
# na sequência o layout inicial para a mitigação:
initial_layout_1 = [0, 1]
initial_layout_2 = [0, 1, 2]

# Função para ordenar os dados de contagens
def RESULT_IBM_ORDINATION(RESULT_GET_COUNTS):

    Contagens = [0, 0, 0, 0, 0, 0, 0, 0]

    DIC_get_counts = RESULT_GET_COUNTS

    for chave, valor in DIC_get_counts.items():
        if chave == '000':
```

```
            Contagens[0] = valor
    for chave, valor in DIC_get_counts.items():
        if chave == '001':
            Contagens[1] = valor
    for chave, valor in DIC_get_counts.items():
        if chave == '010':
            Contagens[2] = valor
    for chave, valor in DIC_get_counts.items():
        if chave == '011':
            Contagens[3] = valor
    for chave, valor in DIC_get_counts.items():
        if chave == '100':
            Contagens[4] = valor
    for chave, valor in DIC_get_counts.items():
        if chave == '101':
            Contagens[5] = valor
    for chave, valor in DIC_get_counts.items():
        if chave == '110':
            Contagens[6] = valor
    for chave, valor in DIC_get_counts.items():
        if chave == '111':
            Contagens[7] = valor

    return Contagens

# Função que calcula a distância do traço
def Trace_distance(PSI, PHI):
    Td = np.sqrt(1-abs(np.inner(PSI.T.conjugate(),PHI.T))**2)
    return Td.real

# Função que calcula a fidelidade
def Fidelity(PSI, PHI):
    return abs(np.inner(PSI.T.conjugate(), PHI.T))**2

# Função que retorna o estado por meio de um array do numpy
def ESTADOS(s):
    #-------------------------------------------------------------------------
    if s == 0:
        psi = np.array([1, 1, 1, 1])
    if s == 1:
        psi = np.array([1, -1, -1, 1])
    if s == 2:
        psi = (1/2)*np.array([1, np.exp(1j*np.pi/4), np.exp(1j*np.pi/4),
        np.exp(1j*np.pi/2)])
    if s == 3:
        psi = (1/2)*np.array([1, -np.exp(1j*np.pi/4), -np.exp(1j*np.pi/4),
        np.exp(1j*np.pi/2)])
    #-------------------------------------------------------------------------
```

```
    if s == 4:
        psi = np.array([1, 0, 0, 1])
    if s == 5:
        psi = np.array([1, 0, 0, -1])
    if s == 6:
        psi = np.array([0, 1, 1, 0])
    if s == 7:
        psi = np.array([0, 1, -1, 0])
    #-----------------------------------------------------------------------
    if s == 8:
        psi = np.array([-0.09+0.477j,-0.353-0.0759j,-0.316-0.659j,0.295-0.118j])
    #-----------------------------------------------------------------------
    if s == 9:
        from qiskit.quantum_info import Statevector, random_statevector
        psi_random = random_statevector(4) # <-- RANDOM STATE
        psi = np.array(psi_random)
    #-----------------------------------------------------------------------
    psi = psi/np.linalg.norm(psi)

    return psi
# Armazenando os estados da função acima em um lista
STATEs_LIST = []
for K in range(10):
    STATEs_LIST.append(ESTADOS(K))

##############################################################################################
# CIRCUITO DA QFT (SEM SWAP) + MEDIÇÃO NA BASE COMPUTACIONAL
##############################################################################################
Q = QuantumRegister(2, name = 'q')
C = ClassicalRegister(3, name = 'c')
QFT = QuantumCircuit(Q, C)
#-------------------------------------------------------------
QFT.h(Q[1])
QFT.cp(pi/2, Q[0], Q[1]) # CROT from qubit 0 to qubit 1
QFT.barrier()
QFT.h(Q[0])
QFT.barrier()
QFT.measure([0,1], [1,0]) # MEDINDO NA BASE COMPUTACIONAL
#-------------------------------------------------------------
# Transpilando o circuito
#-------------------------------------------------------------
QFT_REAL = transpile(QFT, backend=backend, optimization_level=3,
                     initial_layout=initial_layout_1)
##############################################################################################

##############################################################################################
# CIRCUITOS DOS PROJETORES
##############################################################################################
```

```
130 def PROJETOR_CIRCUITO(base, i_qubit):
131     #-------------------------------------------------------------
132     Q  = QuantumRegister(2, name = 'q')
133     C  = ClassicalRegister(3, name = 'c')
134     QC = QuantumCircuit(Q, C)
135     #-------------------------------------------------------------
136     QC.barrier()
137     if base == 0: #sigma_Z
138         QC.measure(Q[i_qubit], C[2])
139     if base == 1: #sigma_X
140         QC.h(Q[i_qubit])
141         QC.measure(Q[i_qubit], C[2])
142         QC.h(Q[i_qubit])
143     if base == 2: #sigma_Y
144         QC.sdg(Q[i_qubit])
145         QC.h(Q[i_qubit])
146         QC.measure(Q[i_qubit], C[2])
147         QC.h(Q[i_qubit])
148         QC.s(Q[i_qubit])
149     QC.barrier()
150 
151     return QC
152 
153 # Armazenando os projetores em uma lista
154 PROJETOR_CIRCUITO_REAL = []
155 for k in range(3):
156     for p in range(1, -1, -1):
157         PJ = PROJETOR_CIRCUITO(k, p)
158         PROJETOR_REAL = transpile(PJ, backend=backend, optimization_level=3,
159                                   initial_layout=initial_layout_1)
160         PROJETOR_CIRCUITO_REAL.append(PROJETOR_REAL)
161 ##############################################################################################
162 
163 ##############################################################################################
164 # CIRCUITOS DE PREPARAÇÃO DOS ESTADOS
165 ##############################################################################################
166 STATE_REAL_LIST = []
167 for S in range(10):
168     #-------------------------------------------------------------
169     Q = QuantumRegister(2, name = 'q')
170     C = ClassicalRegister(3, name = 'c')
171     #-------------------------------------------------------------
172     # PREPARAÇÃO DO ESTADO INCIAL
173     #-------------------------------------------------------------
174     STATE = QuantumCircuit(Q, C)
175     #-------------------------------------------------------------
176     if S == 0:
177         STATE.h(Q[0])
```

```
178         STATE.h(Q[1])
179     if S == 1:
180         STATE.x(Q[0])
181         STATE.x(Q[1])
182         STATE.h(Q[0])
183         STATE.h(Q[1])
184     if S == 2:
185         STATE.h(Q[0])
186         STATE.h(Q[1])
187         STATE.t(Q[0])
188         STATE.t(Q[1])
189     if S == 3:
190         STATE.x(Q[0])
191         STATE.x(Q[1])
192         STATE.h(Q[0])
193         STATE.h(Q[1])
194         STATE.t(Q[0])
195         STATE.t(Q[1])
196     if S == 4:
197         STATE.h(Q[0])
198         STATE.cx(Q[0], Q[1])
199     if S == 5:
200         STATE.h(Q[0])
201         STATE.cx(Q[0], Q[1])
202         STATE.z(Q[1])
203     if S == 6:
204         STATE.h(Q[0])
205         STATE.x(Q[1])
206         STATE.cx(Q[0], Q[1])
207     if S == 7:
208         STATE.h(Q[0])
209         STATE.x(Q[1])
210         STATE.cx(Q[0], Q[1])
211         STATE.z(Q[1])
212     if S == 8:
213         STATE.initialize(STATEs_LIST[8])
214     if S == 9:
215         STATE.initialize(STATEs_LIST[9])
216     #------------------------------------------------------------
217     STATE_REAL = transpile(STATE, backend=backend, optimization_level=3,
218                            initial_layout=initial_layout_1)
219     #------------------------------------------------------------
220     STATE_REAL_LIST.append(STATE_REAL)
221 ##############################################################################################
222
223 ##############################################################################################
224 # CONCATENAÇÃO DOS CIRCUITOS: ESTADO + PROJETOR + (QFT +MEDIÇÃO NA BASE COMPUTACIONAL)
225 ##############################################################################################
```

```
CIRCUITO_PTICOGRAFICO = []
for k in range(10):
    for i in range(6):
        qc = STATE_REAL_LIST[k].compose(PROJETOR_CIRCUITO_REAL[i])
        qcf = qc.compose(QFT_REAL)
        CIRCUITO_PTICOGRAFICO.append(qcf)
###############################################################################################

###############################################################################################
# SIMULAÇÃO DE TODOS OS 10 ESTADOS
###############################################################################################
noise_model = NoiseModel.from_backend(backend)
coupling_map = backend.configuration().coupling_map
basis_gates = noise_model.basis_gates

sim_device = AerSimulator.from_backend(backend)

DATA_RESULT = sim_device.run(CIRCUITO_PTICOGRAFICO,
                             coupling_map=coupling_map,
                             basis_gates=basis_gates,
                             noise_model=noise_model,
                             shots=SHOTS).result()
###############################################################################################

###############################################################################################
# MITIGANDO OS ERROS DE LEITURA
###############################################################################################
qr = QuantumRegister(3)
meas_calibs, state_labels = complete_meas_cal(qr=qr, circlabel='mcal')

cal_results = execute(meas_calibs, Aer.get_backend('qasm_simulator'),
                  coupling_map=coupling_map,
                  basis_gates=basis_gates,
                  noise_model=noise_model,
                  optimization_level=3,
                  initial_layout=initial_layout_2,
                  shots=SHOTS).result()

meas_fitter = CompleteMeasFitter(cal_results, state_labels, circlabel='mcal')
meas_filter = meas_fitter.filter

mitigated_result = meas_filter.apply(DATA_RESULT)
###############################################################################################

###############################################################################################
# MATRIZES DOS PROJETORES COM A IDENTIDADE, QFT E QFT INVERSA
###############################################################################################
i = complex(0,1)
```

```
274
275 PZ10 = np.array([[ 1, 0, 0, 0],          #|0><0| (produto tensorial) I (I=identidade)
276                  [ 0, 1, 0, 0],
277                  [ 0, 0, 0, 0],
278                  [ 0, 0, 0, 0]])
279
280 PZ11 = np.array([[ 0, 0, 0, 0],             #|1><1| (produto tensorial) I
281                  [ 0, 0, 0, 0],
282                  [ 0, 0, 1, 0],
283                  [ 0, 0, 0, 1]])
284
285 PZ20 = np.array([[ 1, 0, 0, 0],             #I (produto tensorial) |0><0|
286                  [ 0, 0, 0, 0],
287                  [ 0, 0, 1, 0],
288                  [ 0, 0, 0, 0]])
289
290 PZ21 = np.array([[ 0, 0, 0, 0],             #I (produto tensorial) |1><1|
291                  [ 0, 1, 0, 0],
292                  [ 0, 0, 0, 0],
293                  [ 0, 0, 0, 1]])
294
295 PX1p = (1/2)*np.array([[ 1, 0, 1, 0],       #|+><+| (produto tensorial) I
296                        [ 0, 1, 0, 1],
297                        [ 1, 0, 1, 0],
298                        [ 0, 1, 0, 1]])
299
300 PX1m = (1/2)*np.array([[ 1, 0,-1, 0],       #|-><-| (produto tensorial) I
301                        [ 0, 1, 0,-1],
302                        [-1, 0, 1, 0],
303                        [ 0,-1, 0, 1]])
304
305 PX2p = (1/2)*np.array([[ 1, 1, 0, 0],       #I (produto tensorial) |+><+|
306                        [ 1, 1, 0, 0],
307                        [ 0, 0, 1, 1],
308                        [ 0, 0, 1, 1]])
309
310 PX2m = (1/2)*np.array([[ 1,-1, 0, 0],       #I (produto tensorial) |-><-|
311                        [-1, 1, 0, 0],
312                        [ 0, 0, 1,-1],
313                        [ 0, 0,-1, 1]])
314
315 PY1R = (1/2)*np.array([[ 1, 0, i, 0],       #|R><R| (produto tensorial) I
316                        [ 0, 1, 0, i],
317                        [-i, 0, 1, 0],
318                        [ 0,-i, 0, 1]])
319
320 PY1L = (1/2)*np.array([[ 1, 0,-i, 0],       #|L><L| (produto tensorial) I
321                        [ 0, 1, 0,-i],
```

```
                        [ i, 0, 1, 0],
                        [ 0, i, 0, 1]])

PY2R = (1/2)*np.array([[ 1, i, 0, 0],      #I (produto tensorial) |R><R|
                        [-i, 1, 0, 0],
                        [ 0, 0, 1, i],
                        [ 0, 0,-i, 1]])

PY2L = (1/2)*np.array([[ 1,-i, 0, 0],      #I (produto tensorial) |R><R|
                        [ i, 1, 0, 0],
                        [ 0, 0, 1,-i],
                        [ 0, 0, i, 1]])

#-------------------------------------
QFT  = (1/2)*np.array([[ 1, 1, 1, 1],
                        [ 1, i,-1,-i],
                        [ 1,-1, 1,-1],
                        [ 1,-i,-1, i]])

IQFT = (1/2)*np.array([[ 1, 1, 1, 1],
                        [ 1,-i,-1, i],
                        [ 1,-1, 1,-1],
                        [ 1, i,-1,-i]])
#-------------------------------------

P_l = np.array([PZ10, PZ11, PZ20, PZ21, PX1p, PX1m,
                PX2p, PX2m, PY1L, PY1R, PY2L, PY2R])
##############################################################################################

##############################################################################################
# PROCESSANDO OS DADOS E RECONSTRUINDO OS ESTADOS
##############################################################################################

lista_files = ['uniform1', 'uniform2', 'uniform3', 'uniform4',
               'bell1', 'bell2', 'bell3', 'bell4', 'arbitrary', 'random']

for s in range(10):
    psi = STATEs_LIST[s]
    #-----------------------------------------------------------------------------
    DADOS_0 = RESULT_IBM_ORDINATION( DATA_RESULT.get_counts(0 + s*6))
    DADOS_1 = RESULT_IBM_ORDINATION( DATA_RESULT.get_counts(1 + s*6))
    DADOS_2 = RESULT_IBM_ORDINATION( DATA_RESULT.get_counts(2 + s*6))
    DADOS_3 = RESULT_IBM_ORDINATION( DATA_RESULT.get_counts(3 + s*6))
    DADOS_4 = RESULT_IBM_ORDINATION( DATA_RESULT.get_counts(4 + s*6))
    DADOS_5 = RESULT_IBM_ORDINATION( DATA_RESULT.get_counts(5 + s*6))
    #-----------------------------------------------------------------------------
    DADOS_0_mitig = RESULT_IBM_ORDINATION( mitigated_result.get_counts(0 + s*6))
    DADOS_1_mitig = RESULT_IBM_ORDINATION( mitigated_result.get_counts(1 + s*6))
```

```
370     DADOS_2_mitig = RESULT_IBM_ORDINATION( mitigated_result.get_counts(2 + s*6))
371     DADOS_3_mitig = RESULT_IBM_ORDINATION( mitigated_result.get_counts(3 + s*6))
372     DADOS_4_mitig = RESULT_IBM_ORDINATION( mitigated_result.get_counts(4 + s*6))
373     DADOS_5_mitig = RESULT_IBM_ORDINATION( mitigated_result.get_counts(5 + s*6))
374     #-------------------------------------------------------------------------------
375     data = open(f'{lista_files[s]}.txt', 'w')
376     data_mitig = open(f'{lista_files[s]}.txt', 'w')
377     #-------------------------------------------------------------------------------
378     data.write(str(DADOS_0))
379     data.write('\n')
380     data.write(str(DADOS_1))
381     data.write('\n')
382     data.write(str(DADOS_2))
383     data.write('\n')
384     data.write(str(DADOS_3))
385     data.write('\n')
386     data.write(str(DADOS_4))
387     data.write('\n')
388     data.write(str(DADOS_5))
389     data.write('\n')
390
391     DADOS_0 = np.array(DADOS_0).reshape(8,1)
392     DADOS_1 = np.array(DADOS_1).reshape(8,1)
393     DADOS_2 = np.array(DADOS_2).reshape(8,1)
394     DADOS_3 = np.array(DADOS_3).reshape(8,1)
395     DADOS_4 = np.array(DADOS_4).reshape(8,1)
396     DADOS_5 = np.array(DADOS_5).reshape(8,1)
397
398     DADOS = np.array([DADOS_0[:4], DADOS_0[4:], DADOS_1[:4], DADOS_1[4:],
399                       DADOS_2[:4], DADOS_2[4:], DADOS_3[:4], DADOS_3[4:],
400                       DADOS_4[:4], DADOS_4[4:], DADOS_5[:4], DADOS_5[4:]])
401
402     ###################### ALGORITMO PIE - DADOS SEM MITIGAÇÃO DE ERROS ###############
403     D_trace_global  = []
404     Fidelity_global = []
405     for m in range(100):
406         phi = np.random.rand(4,1) + np.random.rand(4,1)*i
407         for beta in np.arange(2.0, 0.0, -0.04):
408             for l in range(12):
409                 phi_l    = P_l[l]@phi
410                 phi_ol   = QFT@phi_l
411                 phi_cl   = np.sqrt(DADOS[l])*np.exp(i*np.angle(phi_ol))
412                 psi_cl   = IQFT@phi_cl
413                 psi_ol   = phi
414                 phi      = phi + beta*(P_l[l]@(psi_cl-phi))
415                 #-----------------------------------------
416                 phi_N    = phi/np.linalg.norm(phi)
417                 psi_olN  = psi_ol/np.linalg.norm(psi_ol)
```

```
418                 #----------------------------------------
419             D_trace_global.append(Trace_distance(phi_N, psi_olN))
420             Fidelity_global.append(Fidelity(phi_N, psi.reshape(4,1)))
421         ###############################################################################
422         data.write('\n Fidelidade - media - devio padrao \n')
423         data.write('\n')
424         data.write(str(np.mean(Fidelity_global)))
425         data.write('\n')
426         data.write(str(np.std(Fidelity_global)))
427         data.write('\n')
428         data.write('\n Distancia do traco - media - devio padrao \n')
429         data.write('\n')
430         data.write(str(np.mean(D_trace_global)))
431         data.write('\n')
432         data.write(str(np.std(D_trace_global)))
433         data.write('\n\n')
434         data.write(str(psi))
435         data.close()
436         ###############################################################################
437
438         data_mitig.write(str(DADOS_0_mitig))
439         data_mitig.write('\n')
440         data_mitig.write(str(DADOS_1_mitig))
441         data_mitig.write('\n')
442         data_mitig.write(str(DADOS_2_mitig))
443         data_mitig.write('\n')
444         data_mitig.write(str(DADOS_3_mitig))
445         data_mitig.write('\n')
446         data_mitig.write(str(DADOS_4_mitig))
447         data_mitig.write('\n')
448         data_mitig.write(str(DADOS_5_mitig))
449         data_mitig.write('\n')
450
451         DADOS_0_mitig = np.array(DADOS_0_mitig).reshape(8,1)
452         DADOS_1_mitig = np.array(DADOS_1_mitig).reshape(8,1)
453         DADOS_2_mitig = np.array(DADOS_2_mitig).reshape(8,1)
454         DADOS_3_mitig = np.array(DADOS_3_mitig).reshape(8,1)
455         DADOS_4_mitig = np.array(DADOS_4_mitig).reshape(8,1)
456         DADOS_5_mitig = np.array(DADOS_5_mitig).reshape(8,1)
457
458         DADOS_mitig = np.array([DADOS_0_mitig[:4], DADOS_0_mitig[4:], DADOS_1_mitig[:4],
459                                 DADOS_1_mitig[4:], DADOS_2_mitig[:4], DADOS_2_mitig[4:],
460                                 DADOS_3_mitig[:4], DADOS_3_mitig[4:], DADOS_4_mitig[:4],
461                                 DADOS_4_mitig[4:], DADOS_5_mitig[:4], DADOS_5_mitig[4:]])
462
463         ###################### ALGORITMO PIE - DADOS COM MITIGAÇÃO DE ERROS ###############
464         D_trace_global  = []
465         Fidelity_global = []
```

```
    for m in range(100):
        phi = np.random.rand(4,1) + np.random.rand(4,1)*i
        for beta in np.arange(2.0, 0.0, -0.04):
            for l in range(12):
                phi_l    = P_l[l]@phi
                phi_ol   = QFT@phi_l
                phi_cl   = np.sqrt(DADOS_mitig[l])*np.exp(i*np.angle(phi_ol))
                psi_cl   = IQFT@phi_cl
                psi_ol   = phi
                phi      = phi + beta*(P_l[l]@(psi_cl-phi))
                #----------------------------------------
                phi_N    = phi/np.linalg.norm(phi)
                psi_olN  = psi_ol/np.linalg.norm(psi_ol)
                #----------------------------------------
        D_trace_global.append(Trace_distance(phi_N, psi_olN))
        Fidelity_global.append(Fidelity(phi_N, psi.reshape(4,1)))
    ##############################################################################################
    data_mitig.write('\n Fidelidade - media - devio padrao \n')
    data_mitig.write('\n')
    data_mitig.write(str(np.mean(Fidelity_global)))
    data_mitig.write('\n')
    data_mitig.write(str(np.std(Fidelity_global)))
    data_mitig.write('\n')
    data_mitig.write('\n Distancia do traco - media - devio padrao \n')
    data_mitig.write('\n')
    data_mitig.write(str(np.mean(D_trace_global)))
    data_mitig.write('\n')
    data_mitig.write(str(np.std(D_trace_global)))
    data_mitig.write('\n\n')
    data_mitig.write(str(psi))
    data_mitig.close()

```

Pticografia quântica com n qubits

Nesta seção, apresentamos o código que implementa a pticografia quântica para estados de n qubits. Este código é referente à simulação com ruído, onde escolhemos uma máquina e simulamos a reconstrução dos 5 estados listados no quadro da seção 6.3.2.

Circuitos de preparação dos estados de n qubits

Código que gera os vetores e circuitos do estado uniforme $|\psi_1^n\rangle$ (veja a figura 75(a)):

```
# Vetores do estado uniforme 1:
def uniform_phase1_4pi(n_qubits):
    v = np.array([1, np.exp(1j*pi/4)])
    a = np.array([1, np.exp(1j*pi/4)])
    for n in range(n_qubits-1):
        v = np.kron(v, a)
    return v
psi = uniform_phase1_4pi(n_qubits)
psi = psi/np.linalg.norm(psi)

# Circuitos do estado uniforme 1:
MIN_QUBITS = 3 # Mínimo (estado com 3 qubits)
MAX_QUBITS = 6 # Máximo (estado com 6 qubits)
STATE_REAL_LIST = []
for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
    N_bits = n_qubits + 1 # <-- NUMERO DE BITS CLASSICOS
    Q = QuantumRegister(n_qubits, name = 'q')
    C = ClassicalRegister(N_bits, name = 'c')
    STATE = QuantumCircuit(Q, C)
    for i in range(n_qubits):
        STATE.h(Q[i])
        STATE.t(Q[i])
    #---------------------------------------------------------------
    STATE_REAL = transpile(STATE, backend=backend, optimization_level=3,
                           initial_layout=LAYOUT_LIST[n_qubits-MIN_QUBITS])
    #---------------------------------------------------------------
    STATE_REAL_LIST.append(STATE_REAL)
```

Código que gera os vetores e circuitos do estado uniforme $|\psi_2^n\rangle$ (veja a figura 75(b)):

```
# Vetores do estado uniforme 2:
def uniform_phase2_4pi(n_qubits):
    v = np.array([1, -np.exp(1j*pi/4)])
    a = np.array([1, -np.exp(1j*pi/4)])
    for n in range(n_qubits-1):
        v = np.kron(v, a)
    return v
psi = uniform_phase2_4pi(n_qubits)
psi = psi/np.linalg.norm(psi)

# Circuitos do estado uniforme 2:
MIN_QUBITS = 3 # Mínimo (estado com 3 qubits)
MAX_QUBITS = 6 # Máximo (estado com 6 qubits)
STATE_REAL_LIST = []
for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
```

```
    N_bits = n_qubits + 1 # <-- NUMERO DE BITS CLASSICOS
    Q = QuantumRegister(n_qubits, name = 'q')
    C = ClassicalRegister(N_bits, name = 'c')
    STATE = QuantumCircuit(Q, C)
    for i in range(n_qubits):
        STATE.x(Q[i])
        STATE.h(Q[i])
        STATE.t(Q[i])
    #---------------------------------------------------------------
    STATE_REAL = transpile(STATE, backend=backend, optimization_level=3,
                           initial_layout=LAYOUT_LIST[n_qubits-MIN_QUBITS])
    #---------------------------------------------------------------
    STATE_REAL_LIST.append(STATE_REAL)
```

Código que gera os vetores e circuitos do estado aleatório separável, $|\psi_3^n\rangle$ (veja a figura 75(c)):

```
# Função que gera o conjunto de ângulos aleatórios
def rand_angle_list(N):
    from random import uniform
    from numpy import pi, zeros, array

    R = zeros((N, 3))

    for i in range(N):
        #-----------------------
        a = uniform(0, pi)
        b = uniform(0, 2*pi)
        c = uniform(0, 2*pi)
        #-----------------------
        R[i][0]  = a
        R[i][1]  = b
        R[i][2]  = c
        #-----------------------
    return R

# Função que prepara o estado
def prep_state_rand_sep(N, list_angles):
    from qiskit import QuantumCircuit
    N_bits = N + 1
    prep_circuit = QuantumCircuit(N, N_bits)

    for i in range(N):
        #-----------------------
        A = list_angles[i][0]
        B = list_angles[i][1]
        C = list_angles[i][2]
```

```
        #-----------------------
        prep_circuit.u(A, B, C, i)
        #-----------------------
    return prep_circuit

# Função que gera os vetores dos estados aleatórios
def PSI(N, list_angles):
    from numpy import pi, cos, sin, exp, array, kron

    for i in range(N-1, -1, -1):
        #-----------------------
        A = list_angles[i][0]
        B = list_angles[i][1]
        C = list_angles[i][2]
        #-----------------------
        a11 = cos(A/2)
        a12 = -exp(1j*C)*sin(A/2)
        a21 = exp(1j*B)*sin(A/2)
        a22 = exp(1j*(B+C))*cos(A/2)
        #-----------------------
        V = array([[1], [0]])
        U = array([[a11, a12], [a21, a22]])
        if i == N-1:
            psi_ideal = U@V
        else:
            psi_ideal = kron(psi_ideal, U@V)
            #-----------------------
    return psi_ideal

# Listando os vetores e ciruitos transpilados
STATE_REAL_LIST = []
PSI_LIST = []
for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
    angles_list = rand_angle_list(n_qubits)
    prep_state = prep_state_rand_sep(n_qubits, angles_list)
    #-------------------------------------------------------------
    STATE_REAL = transpile(prep_state, backend=backend, optimization_level=3,
                           initial_layout=LAYOUT_LIST[n_qubits-MIN_QUBITS])
    #-------------------------------------------------------------
    STATE_REAL_LIST.append(STATE_REAL)
    #-------------------------------------------------------------
    psi = PSI(n_qubits, angles_list)
    PSI_LIST.append(psi)
```

Código que gera os vetores e circuitos do estado GHZ, $|\psi_4^n\rangle$:

```
# Vetores do estado GHZ:
def GHZ(n_qubits):
    v = np.array([0, 1])
    a = np.array([0, 1])
    for n in range(n_qubits-1):
        v = np.kron(v, a)
    v[0] = 1
    return v
psi = GHZ(n_qubits)
psi = psi/np.linalg.norm(psi)

# Circuitos do estado GHZ:
MIN_QUBITS = 3 # Mínimo (estado com 3 qubits)
MAX_QUBITS = 6 # Máximo (estado com 6 qubits)
STATE_REAL_LIST = []
for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
    N_bits = n_qubits + 1 # <-- NUMERO DE BITS CLASSICOS
    Q = QuantumRegister(n_qubits, name = 'q')
    C = ClassicalRegister(N_bits, name = 'c')
    STATE = QuantumCircuit(Q, C)
    STATE.h(Q[0])
    for i in range(n_qubits-1):
        STATE.cx(Q[i], Q[i+1])
    #---------------------------------------------------------------
    STATE_REAL = transpile(STATE, backend=backend, optimization_level=3,
                           initial_layout=LAYOUT_LIST[n_qubits-MIN_QUBITS])
    #---------------------------------------------------------------
    STATE_REAL_LIST.append(STATE_REAL)
```

Código que gera os vetores e circuitos do estado W, $|\psi_5^n\rangle$:

```
# Vetores do estado W:
#-------------------------------------------------
def W_state(n_qubits):
    w = np.array([0, 0])
    a = np.array([0, 0])
    for n in range(n_qubits-1):
        w = np.kron(w, a)
    w[1] = 1
    w[2] = 1
    if n_qubits >= 3:
        for n in range(n_qubits-2):
            w[2**(n+2)] = 1
    return w
psi = W_state(n_qubits)
psi = psi/np.linalg.norm(psi)

```

```
# Circuitos do estado W:
MIN_QUBITS = 3 # Mínimo (estado com 3 qubits)
MAX_QUBITS = 6 # Máximo (estado com 6 qubits)
STATE_REAL_LIST = []
for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
    N_bits = n_qubits + 1 # <-- NUMERO DE BITS CLASSICOS
    Q = QuantumRegister(n_qubits, name = 'q')
    C = ClassicalRegister(N_bits, name = 'c')
    STATE = QuantumCircuit(Q, C)
    STATE.x(Q[0])
    STATE.barrier()
    for k in range(n_qubits-1):
        theta = np.arccos(np.sqrt(1/(n_qubits-k)))
        STATE.ry(-theta,Q[k+1])
        STATE.cz(Q[k],Q[k+1])
        STATE.ry(theta,Q[k+1])
        STATE.cx(Q[k+1],Q[k])
    #-------------------------------------------------------------
    STATE_REAL = transpile(STATE, backend=backend, optimization_level=3,
                           initial_layout=LAYOUT_LIST[n_qubits-MIN_QUBITS])
    #-------------------------------------------------------------
    STATE_REAL_LIST.append(STATE_REAL)
```

Pticografia quântica com n qubits: exemplo com o estado W

Código completo para a simulação da pticografia quântica com n qubits; usamos como exemplo a reconstrução do estado W, que pode ser facilmente substituído:

```
# Bibliotecas necessárias
from qiskit import *
from qiskit.providers.aer import AerSimulator
from qiskit.providers.aer.noise import NoiseModel
from qiskit.ignis.mitigation.measurement import complete_meas_cal, CompleteMeasFitter
import numpy as np
from numpy import pi

# Conectado na IBM Quantum:
IBMQ.enable_account('Cole seu token aqui')
provider = IBMQ.get_provider(hub='ibm-q', group='open', project='main')

# Escolha o dispositivo:
maquina = 'ibm_perth'
backend = provider.get_backend(f'{maquina}')

# Escolha o número de shots:
SHOTS = 100000
```

```
19
20  # Intervalo para o número de qubits:
21  MAX_QUBITS = 6
22  MIN_QUBITS = 2
23
24  # Escolhendo o layout para os circuitos pticográficos e para mitigação:
25  # (é necessário ver o mapa de acoplamento para escolher um bom layout)
26  LAYOUT_LIST = []
27  LAYOUT_LIST_MITIG = []
28  for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
29      #-------------------------------------------------------------
30      if n_qubits == 2:
31          initial_layout_1 = [3, 1]
32          initial_layout_2 = [3, 1, 0]
33      if n_qubits == 3:
34          initial_layout_1 = [2, 3, 1]
35          initial_layout_2 = [2, 3, 1, 0]
36      if n_qubits == 4:
37          initial_layout_1 = [2, 0, 3, 1]
38          initial_layout_2 = [2, 0, 3, 1, 5]
39      if n_qubits == 5:
40          initial_layout_1 = [5, 0, 2, 3, 1]
41          initial_layout_2 = [5, 0, 2, 3, 1, 6]
42      if n_qubits == 6:
43          initial_layout_1 = [4, 0, 2, 5, 1, 3]
44          initial_layout_2 = [4, 0, 2, 5, 1, 3, 6]
45      #-------------------------------------------------------------
46      LAYOUT_LIST.append(initial_layout_1)
47      LAYOUT_LIST_MITIG.append(initial_layout_2)
48
49  # FUNÇÃO QUE OTIMIZA OS CIRCUITOS
50  def GREAT_TRANSPILE(QC, BACKEND, INITIAL_LAYOUT):
51  #--------------
52      sim_device = AerSimulator.from_backend(BACKEND)
53  #--------------
54      noise_model = NoiseModel.from_backend(BACKEND)
55      coupling_map = BACKEND.configuration().coupling_map
56      basis_gates = noise_model.basis_gates
57  #--------------
58      c1 = 0
59      c2 = 0
60      min_count_gate_cx = 0
61      min_count_gate_rz = 0
62      min_count_gate_sx = 0
63      #-----------------------------------------------------
64      for k in range(100):
65          circ = transpile(QC, BACKEND, coupling_map=coupling_map,
66                           basis_gates=basis_gates,
```

```
                        initial_layout=INITIAL_LAYOUT,
                        optimization_level=3)
        counts_gates = dict(circ.count_ops())
        #-------------------------------------------------
        if k==0:
            min_count_gate_cx = counts_gates['cx']
            SAVE_CIRC = circ
        else:
            if min_count_gate_cx > counts_gates['cx']:
                min_count_gate_cx = counts_gates['cx']
                SAVE_CIRC = circ
            elif min_count_gate_cx == counts_gates['cx']:
                #-----------------------------------------
                if c1==0:
                    min_count_gate_rz = counts_gates['rz']
                    c1 += 1
                    SAVE_CIRC = circ
                else:
                    if min_count_gate_rz > counts_gates['rz']:
                        min_count_gate_rz = counts_gates['rz']
                        c1 += 1
                        SAVE_CIRC = circ
                    elif min_count_gate_rz == counts_gates['rz']:
                        #-------------------------------------
                        if c2==0:
                            min_count_gate_sx = counts_gates['sx']
                            c2 += 1
                            SAVE_CIRC = circ
                        else:
                            if min_count_gate_sx > counts_gates['sx']:
                                min_count_gate_sx = counts_gates['sx']
                                c2 += 1
                                SAVE_CIRC = circ

    return SAVE_CIRC

# Função que ordena as contagens
def RESULT_IBM_ORDINATION(RESULT_GET_COUNTS, N_KEYS):
    Contagens = []
    for k in range(N_KEYS):
        Contagens.append(0)

    for k in range(N_KEYS):
        for chave, valor in RESULT_GET_COUNTS.items():
            if int(chave, 2) == k:
                Contagens[k] = valor

    return Contagens
```

```
115
116 # Função para gera os vetores do estado W
117 def W_state(n_qubits):
118     w = np.array([0, 0])
119     a = np.array([0, 0])
120     for n in range(n_qubits-1):
121         w = np.kron(w, a)
122     w[1] = 1
123     w[2] = 1
124     if n_qubits >= 3:
125         for n in range(n_qubits-2):
126             w[2**(n+2)] = 1
127     return w
128 # Função que calcula a distância do traço
129 def Trace_distance(PSI, PHI):
130     Td = np.sqrt(1-abs(np.inner(PSI.T.conjugate(),PHI.T))**2)
131     return Td.real
132
133 # Função que calcula a fidelidade
134 def Fidelity(PSI, PHI):
135     return abs(np.inner(PSI.T.conjugate(), PHI.T))
136
137 ##############################################################################################
138 # Gerando os circuitos da QFT
139 ##############################################################################################
140 QFT_REAL_LIST = []
141 for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
142     N_bits = n_qubits + 1 # <-- NUMERO DE BITS CLASSICOS
143     Q = QuantumRegister(n_qubits, name = 'q')
144     C = ClassicalRegister(N_bits, name = 'c')
145     QFT = QuantumCircuit(Q, C)
146     #-------------------------------------------------------------
147     for i in range(n_qubits-1, -1, -1):
148         QFT.h(Q[i])
149         for j in range(i-1, -1, -1):
150             QFT.cp(pi/2**(i-j), Q[i], Q[j])
151         QFT.barrier()
152     QFT.measure(Q, range(n_qubits-1, -1, -1))
153     #-------------------------------------------------------------
154     QFT_REAL = GREAT_TRANSPILE(QFT, backend, LAYOUT_LIST[n_qubits-MIN_QUBITS])
155     #-------------------------------------------------------------
156     QFT_REAL_LIST.append(QFT_REAL)
157 ##############################################################################################
158
159 ##############################################################################################
160 # CIRCUITOS DO ESTADO (W)
161 ##############################################################################################
162 STATE_REAL_LIST = []
```

```
for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
    N_bits = n_qubits + 1 # <-- NUMERO DE BITS CLASSICOS
    Q = QuantumRegister(n_qubits, name = 'q')
    C = ClassicalRegister(N_bits, name = 'c')
    STATE = QuantumCircuit(Q, C)
    #-------------------------------------------------------------
    STATE.x(Q[0])
    STATE.barrier()
    for k in range(n_qubits-1):
        theta = np.arccos(np.sqrt(1/(n_qubits-k)))
        STATE.ry(-theta,Q[k+1])
        STATE.cz(Q[k],Q[k+1])
        STATE.ry(theta,Q[k+1])
        STATE.barrier()
    for k in range(n_qubits-1):
        STATE.cx(Q[k+1],Q[k])
        STATE.barrier()
    #-------------------------------------------------------------
    STATE_REAL = GREAT_TRANSPILE(STATE, backend, LAYOUT_LIST[n_qubits-MIN_QUBITS])
    #-------------------------------------------------------------
    STATE_REAL_LIST.append(STATE_REAL)
##############################################################################################

##############################################################################################
# CIRCUITOS DOS PROJETORES
##############################################################################################
def PROJETOR_CIRCUITO(n_qubits, base, i_qubit):
    #-------------------------------------------------------------
    N_bits = n_qubits + 1 # <-- NUMERO DE BITS CLASSICOS
    Q  = QuantumRegister(n_qubits, name = 'q')
    C  = ClassicalRegister(N_bits, name = 'c')
    QC = QuantumCircuit(Q, C)
    #-------------------------------------------------------------
    QC.barrier()
    if base == 0: #sigma_Z
        QC.measure(Q[i_qubit], C[n_qubits])
    if base == 1: #sigma_X
        QC.h(Q[i_qubit])
        QC.measure(Q[i_qubit], C[n_qubits])
        QC.h(Q[i_qubit])
    if base == 2: #sigma_Y
        QC.sdg(Q[i_qubit])
        QC.h(Q[i_qubit])
        QC.measure(Q[i_qubit], C[n_qubits])
        QC.h(Q[i_qubit])
        QC.s(Q[i_qubit])
    QC.barrier()

```

```
    return QC

PROJETORs_REAL_LIST = []
for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
    PROJETORs_CIRCUITO_QUBIT_n = []
    for k in range(3):
        for p in range(n_qubits-1, -1, -1):
            PJ = PROJETOR_CIRCUITO(n_qubits, k, p)
            PROJETOR_REAL = transpile(PJ, backend=backend, optimization_level=3,
                                      initial_layout=LAYOUT_LIST[n_qubits-MIN_QUBITS])
            PROJETORs_CIRCUITO_QUBIT_n.append(PROJETOR_REAL)
    PROJETORs_REAL_LIST.append(PROJETORs_CIRCUITO_QUBIT_n)
##############################################################################################

##############################################################################################
# CONCATENAÇÃO DOS SUB-CICUITOS
##############################################################################################
CIRCUITOs_PTICOGRAFICO_n = []
for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
    for i in range(3*n_qubits):
        qc = STATE_REAL_LIST[n_qubits-MIN_QUBITS].compose(
             PROJETORs_REAL_LIST[n_qubits-MIN_QUBITS][i])
        qcf = qc.compose(QFT_REAL_LIST[n_qubits-MIN_QUBITS])
        CIRCUITOs_PTICOGRAFICO_n.append(qcf)
##############################################################################################

##############################################################################################
# SIMULAÇÃO DOS CIRCUITOS
##############################################################################################
noise_model = NoiseModel.from_backend(backend)
coupling_map = backend.configuration().coupling_map
basis_gates = noise_model.basis_gates

sim_device = AerSimulator.from_backend(backend)

DATA_RESULTs = sim_device.run(CIRCUITOs_PTICOGRAFICO_n,
                              coupling_map=coupling_map,
                              basis_gates=basis_gates,
                              noise_model=noise_model,
                              shots=SHOTS).result()
##############################################################################################
```

Organização dos dados

Há uma gama de possibilidades para organização dos dados pticográficos. Nós optamos por organizá-los em uma matriz $3n \times 2^{n+1}$, que chamamos de matriz de *shots*: para um dado estado $|\psi\rangle$ ela é representada por $\mathcal{S}_\psi$. Cada linha corresponde aos dados

gerados por um circuito e as colunas 1 a 2^n e $2^n + 1$ a 2^{n+1} correspondem aos dados associados aos projetores $\mathbf{\Pi}^+_{\xi j}$ e $\mathbf{\Pi}^-_{\xi j}$ (equação (5.12)), respectivamente. O ordenamento das linhas seguiu o ordenamento em que os circuitos foram implementados nas simulações e experimentos. Por simplicidade, apresentamos abaixo a transposta da matriz de *shots*

$$\mathcal{S}^{\mathsf{T}}_{\psi} = \begin{bmatrix} \mathcal{D}^+_{z0} & \cdots & \mathcal{D}^+_{zn-1} & \mathcal{D}^+_{x0} & \cdots & \mathcal{D}^+_{xn-1} & \mathcal{D}^+_{y0} & \cdots & \mathcal{D}^+_{yn-1} \\ \mathcal{D}^-_{z0} & \cdots & \mathcal{D}^-_{zn-1} & \mathcal{D}^-_{x0} & \cdots & \mathcal{D}^-_{xn-1} & \mathcal{D}^-_{y0} & \cdots & \mathcal{D}^-_{yn-1} \end{bmatrix}, \tag{B.1}$$

onde $\mathcal{D}^{\pm}_{\xi j}$ é o vetor $2^n \times 1$ de dados pticográficos (equação (5.4)) associado a cada projetor $\mathbf{\Pi}^{\pm}_{\xi j}$. Com os dados de $\mathcal{S}_{\psi}$, o algoritmo de reconstrução é alimentado usando os vetores $\sqrt{\mathcal{D}^{\pm}_{\xi j}}$, dados pela raiz quadrada de cada elemento do vetor de entrada (veja seção 5.2.2). No caso das simulações com ruído e experimentos, nós geramos uma cópia desta matriz e aplicamos o método de mitigação de erros de leitura discutido na seção 6.1.2 para construir uma matriz de *shots* "mitigada", $\mathcal{S}^{\text{mit}}_{\psi}$, a qual também é usada para alimentar o algoritmo de reconstrução.

```
################################################################################################
# PROCESSAMENTO DOS DADOS E RECONSTRUÇÃO
################################################################################################
INITIAL = 0
FINAL = 3*MIN_QUBITS
for n_qubits in range(MIN_QUBITS, MAX_QUBITS+1):
    #-----------------------------------------------------------------------------------------
    psi = W_state(n_qubits)
    psi = psi/np.linalg.norm(psi)
    print(n_qubits)
    #-----------------------------------------------------------------------------------------
    qr = QuantumRegister(n_qubits+1)
    meas_calibs, state_labels = complete_meas_cal(qr=qr, circlabel='mcal')
    #-------------------------------------------------
    cal_results = execute(meas_calibs, Aer.get_backend('qasm_simulator'),
                          coupling_map=coupling_map,
                          basis_gates=basis_gates,
                          noise_model=noise_model,
                          initial_layout = LAYOUT_LIST_MITIG[n_qubits-MIN_QUBITS],
                          optimization_level=3,
                          shots=SHOTS).result()
    meas_fitter = CompleteMeasFitter(cal_results, state_labels, circlabel='mcal')
    meas_filter = meas_fitter.filter
    ############################################################################################
    data = open(f'W_state.txt', 'w')
    data_mitig = open(f'W_state_mitig.txt', 'w')
    ############################################################################################
    N_chaves = 2**(n_qubits+1)
    DADOS = []
    DADOS_mitig = []
```

```
    for i in range(INITIAL, FINAL):
        DATA_RESULT_COUNTS = DATA_RESULTs.get_counts(i)
        #-------------------------------------------------------------------------
        DADOS.append(RESULT_IBM_ORDINATION(DATA_RESULT_COUNTS, N_chaves))
        data.write(str(DADOS[i-INITIAL]))
        data.write('\n')
        #-------------------------------------------------------------------------
        mitigated_result = meas_filter.apply(DATA_RESULT_COUNTS)
        DADOS_mitig.append(RESULT_IBM_ORDINATION(mitigated_result, N_chaves))
        data_mitig.write(str(DADOS_mitig[i-INITIAL]))
        data_mitig.write('\n')
    INITIAL = FINAL
    FINAL += 3*(n_qubits+1)
    ###########################################################################################
    CUT = 2**n_qubits
    for i in range(3*n_qubits):
        DADOS[i] = np.array(DADOS[i]).reshape(2*CUT, 1)
    #Separação dos dados:
    DATA = []
    for i in range(3*n_qubits):
        DATA.append(DADOS[i][:CUT])
        DATA.append(DADOS[i][CUT:])
    #-------------------------------------------------------------------------------
    for i in range(3*n_qubits):
        DADOS_mitig[i] = np.array(DADOS_mitig[i]).reshape(2*CUT, 1)
    #Separação dos dados:
    DATA_mitig = []
    for i in range(3*n_qubits):
        DATA_mitig.append(DADOS_mitig[i][:CUT])
        DATA_mitig.append(DADOS_mitig[i][CUT:])
    ###########################################################################################
    i = complex(0,1)
    #-------------------------------------------------
    # GERANDO OS PROJETORES
    #-------------------------------------------------
    PZ0 = np.array([[1, 0],   #|0><0|
                    [0, 0]])

    PZ1 = np.array([[0, 0],   #|1><1|
                    [0, 1]])

    PXp = (1/2)*np.array([[ 1, 1],   #|+><+|
                          [ 1, 1]])

    PXs = (1/2)*np.array([[ 1,-1],   #|-><-|
                          [-1, 1]])

    PYR = (1/2)*np.array([[ 1, i],   #|R><R|
```

```
                        [-i, 1]])

    PYL = (1/2)*np.array([[ 1,-i],   #|L><L|
                        [ i, 1]])
    #-----------------------------------------------
    P = np.array([PZ0, PZ1, PXp, PXs, PYL, PYR])
    #-----------------------------------------------
    Projetor = []
    c=0
    #-------------------------------------------------------------------
    # Projetores - Z
    #-------------------------------------------------------------------
    for k in range(n_qubits):
        for m in range(2):
            Projetor.append( np.kron(np.eye(2**(k)), P[m]) )
            Projetor[c] = np.kron(Projetor[c], np.eye(2**(n_qubits-1-k)))
            c = c + 1
    #-------------------------------------------------------------------
    # Projetores - X
    #-------------------------------------------------------------------
    for k in range(n_qubits):
        for m in range(2, 4):
            Projetor.append( np.kron(np.eye(2**(k)), P[m]) )
            Projetor[c] = np.kron(Projetor[c], np.eye(2**(n_qubits-1-k)))
            c = c + 1
    #-------------------------------------------------------------------
    # Projetores - Y
    #-------------------------------------------------------------------
    for k in range(n_qubits):
        for m in range(4, 6):
            Projetor.append( np.kron(np.eye(2**(k)), P[m]) )
            Projetor[c] = np.kron(Projetor[c], np.eye(2**(n_qubits-1-k)))
            c = c + 1
    ###################################################################################
    Fidelity_global = []
    D = 2**n_qubits
    idftmtx = np.fft.fft(np.eye(D))/np.sqrt(D)
    dftmtx = idftmtx.T.conjugate()
    for m in range(100):
        phi = np.random.rand(D,1) + np.random.rand(D,1)*i
        for beta in np.arange(2.0, 0.0, -0.04):
            for l in range(6*n_qubits):
                phi_l    = Projetor[l]@phi
                phi_ol   = dftmtx@phi_l
                phi_cl   = np.sqrt(DATA[l])*np.exp(i*np.angle(phi_ol))
                psi_cl   = idftmtx@phi_cl
                psi_ol   = phi
                phi      = phi + beta*(Projetor[l]@(psi_cl-phi))
```

```
                #----------------------------------------
                phi_N    = phi/np.linalg.norm(phi)
                psi_olN  = psi_ol/np.linalg.norm(psi_ol)
                #----------------------------------------
            if Trace_distance(phi_N, psi_olN) < 10.0**(-5.0):
                break
        Fidelity_global.append(Fidelity(phi_N, psi.reshape(D,1)))

    data.write('\n Fidelidade - media - devio padrao \n')
    data.write('\n')
    data.write(str(np.mean(Fidelity_global)))
    data.write('\n')
    data.write(str(np.std(Fidelity_global)))
    data.write('\n')
    data.write('\n Fidelidade - mediana - Interquartile range \n')
    data.write('\n')
    data.write(str(np.median(Fidelity_global)))
    Q1 = np.percentile(Fidelity_global, 25, interpolation = 'midpoint')
    Q3 = np.percentile(Fidelity_global, 75, interpolation = 'midpoint')
    IQR = Q3 - Q1
    data.write('\n')
    data.write(str(IQR))
    data.close()
    ##############################################################################################
    Fidelity_global = []
    for m in range(100):
        phi = np.random.rand(D,1) + np.random.rand(D,1)*i
        for beta in np.arange(2.0, 0.0, -0.04):
            for l in range(6*n_qubits):
                phi_l    = Projetor[l]@phi
                phi_ol   = dftmtx@phi_l
                phi_cl   = np.sqrt(DATA_mitig[l])*np.exp(i*np.angle(phi_ol))
                psi_cl   = idftmtx@phi_cl
                psi_ol   = phi
                phi      = phi + beta*(Projetor[l]@(psi_cl-phi))
                #----------------------------------------
                phi_N    = phi/np.linalg.norm(phi)
                psi_olN  = psi_ol/np.linalg.norm(psi_ol)
                #----------------------------------------
            if Trace_distance(phi_N, psi_olN) < 10.0**(-5.0):
                break
        Fidelity_global.append(Fidelity(phi_N, psi.reshape(D,1)))

    data_mitig.write('\n Fidelidade - media - devio padrao \n')
    data_mitig.write('\n')
    data_mitig.write(str(np.mean(Fidelity_global)))
    data_mitig.write('\n')
    data_mitig.write(str(np.std(Fidelity_global)))
```

```
    data_mitig.write('\n')
    data_mitig.write('\n Fidelidade - mediana - Interquartile range \n')
    data_mitig.write('\n')
    data_mitig.write(str(np.median(Fidelity_global)))
    Q1 = np.percentile(Fidelity_global, 25, interpolation = 'midpoint')
    Q3 = np.percentile(Fidelity_global, 75, interpolation = 'midpoint')
    IQR = Q3 - Q1
    data_mitig.write('\n')
    data_mitig.write(str(IQR))
    data_mitig.write('\n\n')
    data_mitig.write(str(psi))
    data_mitig.close()
```

Circuito da QFT aproximada

Segue abaixo um exemplo de código que cria o circuito da AQFT de grau m (sem **SWAP**) com medição na base computacional:

```
#-------------------------------------------------------------
n_qubits = 4 # NÚMERO DE QUBITS
N_bits = n_qubits + 1 # <-- NÚMERO DE BITS CLÁSSICOS
Q = QuantumRegister(n_qubits, name = 'q')
C = ClassicalRegister(N_bits, name = 'c')
AQFT = QuantumCircuit(Q, C)
#-------------------------------------------------------------
M = m -1
for i in range(n_qubits-1, -1, -1):
    AQFT.h(Q[i])
    if i > M:
        c = i-M
    else:
        c = 0
    for j in range(i-1, c-1, -1):
        AQFT.cp(pi/2**(i-j), Q[i], Q[j])
    AQFT.barrier()
AQFT.measure(Q, range(n_qubits-1, -1, -1))
```