UNIVERSIDADE FEDERAL DE LAVRAS

MARCOS VINICIUS FIGUEIREDO SOUSA

MÉTODOS DE MITIGAÇÃO E SUPRESSÃO DE RUÍDOS QUÂNTICOS EM QPUS APLICADOS AO ALGORITMO DE GROVER.

LAVRAS – MG

2025

MARCOS VINICIUS FIGUEIREDO SOUSA

MÉTODOS DE MITIGAÇÃO E SUPRESSÃO DE RUÍDOS QUÂNTICOS EM QPUS APLICADOS AO ALGORITMO DE GROVER.

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia Física, para a obtenção do título de Bacharel.

Prof. Dr. Moises Porfírio Rojas Leyva
Orientador

LAVRAS – MG
2025

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Sousa, Marcos Vinicius Figueiredo
Métodos de Mitigação e Supressão de ruídos quânticos em QPUs aplicados ao Algoritmo de Grover. / Marcos Vinicius Figueiredo Sousa. – Lavras : UFLA, 2025.
59 p. : il.

Monografia (graduação) –Universidade Federal de Lavras, 2025.
Orientador: Prof. Dr. Moises Porfírio Rojas Leyva.
Bibliografia.

1. Ruído Quântico. 2. Interação de qubits. 3. Computação Quântica. I. Leyva, Moises Porfírio Rojas. II. Título.

MARCOS VINICIUS FIGUEIREDO SOUSA

MÉTODOS DE MITIGAÇÃO E SUPRESSÃO DE RUÍDOS QUÂNTICOS EM QPUS APLICADOS AO ALGORITMO DE GROVER.

METHODS FOR MITIGATION AND SUPPRESSION OF QUANTUM NOISE IN QPUS APPLIED TO THE GROVER'S ALGORITHM.

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia Física, para a obtenção do título de Bacharel.

APROVADA em 30 de Fevereiro de 2016.

Prof. MSc. Antônio Banca Um	UFM
Prof. DSc. João Banca Dois	FCO
Profa. Esp. Eliza Banca Três	BELMIS
Prof. Esp. Carlos Banca Quatro	IBGPLUS

Prof. Dr. Moises Porfírio Rojas Leyva
Orientador

LAVRAS – MG
2025

Espaço reservado a dedicatória.

AGRADECIMENTOS

Espaço reservado aos agradecimentos.

Espaço reservado a epígrafe.
(Autor Desconhecido)

RESUMO

A computação quântica é uma área emergente da ciência da computação que se baseia nos princípios da mecânica quântica para processar informações. Um dos conceitos fundamentais dessa área é a superposição quântica, que permite que os qubits assumam múltiplos estados simultaneamente, possibilitando a realização de cálculos exponencialmente mais rápidos para certos problemas. Um dos algoritmos de destaque nesse contexto, e foco principal deste trabalho, é o Algoritmo de Grover, utilizado para busca em bases de dados não estruturadas, proporcionando uma aceleração quadrática em relação aos métodos clássicos. Este estudo tem como objetivo analisar a estrutura e o funcionamento do Algoritmo de Grover, avaliando sua eficiência teórica e prática por meio de cálculos analíticos, simulações computacionais e implementações em hardwares quânticos reais. Além disso, serão exploradas técnicas para aprimorar a confiabilidade das execuções em dispositivos quânticos físicos, que são susceptíveis a erros devido à presença de ruído quântico, originado principalmente da interação entre os qubits e o ambiente. Para transpor esses desafios, serão investigados métodos de otimização, supressão e mitigação de ruído, buscando melhorar a precisão dos resultados obtidos nas *Quantum Processing Units* (QPUs). Dessa forma, este trabalho pretende não apenas aprofundar o conhecimento sobre a implementação e aplicabilidade do Algoritmo de Grover, mas também avaliar estratégias que aumentem sua robustez em ambientes ruidosos, contribuindo para o avanço da computação quântica.

Palavras-chave: Simulação Quântica; Processamento Quântico; Decoerência.

ABSTRACT

Quantum computing is an emerging field of computer science that is based on the principles of quantum mechanics to process information. One of its fundamental concepts is quantum superposition, which allows qubits to exist in multiple states simultaneously, enabling exponentially faster computations for certain problems. One of the prominent algorithms in this context, and the main focus of this work, is Grover's Algorithm, which is used for searching in unstructured databases, providing a quadratic speedup compared to classical methods. This study aims to analyze the structure and functioning of Grover's Algorithm, evaluating its theoretical and practical efficiency through analytical calculations, computational simulations, and implementations on real quantum hardware. In addition, techniques will be explored to enhance the reliability of executions on physical quantum devices, which are susceptible to errors due to the presence of quantum noise, primarily caused by the interaction between qubits and the environment. To overcome these challenges, methods of noise optimization, suppression, and mitigation will be investigated, seeking to improve the accuracy of results obtained in Quantum Processing Units (QPUs). Thus, this work aims not only to deepen the understanding of the implementation and applicability of Grover's Algorithm but also to evaluate strategies that increase its robustness in noisy environments, contributing to the advancement of quantum computing

Keywords: Quantum Simulation; Quantum Processing; Decoherence.

INDICADORES DE IMPACTO

O presente trabalho promove impactos significativos nos campos tecnológico e educacional, além de contribuir potencialmente para avanços econômicos e sociais a longo prazo. Do ponto de vista tecnológico, a pesquisa auxilia no desenvolvimento de técnicas que aumentam a confiabilidade e, por conseguinte, a aplicabilidade dos computadores quânticos, permitindo uma maior fidelidade nos resultados obtidos e reduzindo as limitações impostas pelos ruídos quânticos intrínsecos às atuais *Quantum Processing Units* (QPUs). Esses avanços são fundamentais para acelerar a adoção da computação quântica em setores estratégicos, como criptografia, inteligência artificial e otimização de processos industriais. No âmbito acadêmico e educacional, o estudo pode agir como um fomentador da formação de novos pesquisadores e profissionais capacitados, ao instigar o desenvolvimento de novas áreas de pesquisa e ainda integrar conhecimentos teóricos e experimentais essenciais para o avanço da computação quântica no Brasil. A pesquisa contribui diretamente para a área temática de Tecnologia e Produção da Política Nacional de Extensão, uma vez que busca aprimorar a aplicabilidade de algoritmos quânticos e desenvolver técnicas de mitigação de erros aplicáveis a diferentes domínios. Além disso, o projeto está alinhado com os Objetivos de Desenvolvimento Sustentável (ODS) da ONU, especialmente o ODS 9, que incentiva a inovação o fortalecimento de pesquisas científicas em países em desenvolvimento, como é o caso do Brasil, e o ODS 4, que visa a educação de qualidade, pois o estudo promove o ensino e a pesquisa em computação quântica no país, uma vez que é apresentado de forma didática e de fácil entendimento. Embora os impactos diretos ainda sejam mais perceptíveis no ambiente acadêmico e de pesquisa, o desenvolvimento de técnicas de mitigação de ruído tem o potencial de beneficiar indústrias e empresas que futuramente empregarão a computação quântica para resolver problemas complexos em menor tempo e com maior eficiência. Dessa forma, este trabalho representa um passo relevante para a evolução da computação quântica e sua aplicabilidade prática, fortalecendo o Brasil como um agente ativo nessa revolução tecnológica global.

IMPACT INDICATORS

This work promotes significant impacts in the technological and educational fields, as well as potentially contributing to long-term economic and social advances. From a technological perspective, the research aids in the development of techniques that enhance the reliability and, consequently, the applicability of quantum computers, enabling greater fidelity in the results obtained and reducing the limitations imposed by the intrinsic quantum noise present in current Quantum Processing Units (QPUs). These advances are essential for accelerating the adoption of quantum computing in strategic sectors such as cryptography, artificial intelligence, and industrial process optimization. In the academic and educational sphere, the study can act as a catalyst for the training of new researchers and skilled professionals by encouraging the development of new research areas while integrating theoretical and experimental knowledge essential for the advancement of quantum computing in Brazil. The research directly contributes to the Technology and Production area of the National Extension Policy, as it seeks to improve the applicability of quantum algorithms and develop error mitigation techniques applicable to different domains. Furthermore, the project aligns with the United Nations Sustainable Development Goals (SDGs), particularly SDG 9, which promotes innovation and strengthens scientific research in developing countries, such as Brazil, and SDG 4, which aims for quality education, as the study fosters teaching and research in quantum computing in the country, presenting the topic in a didactic and easily understandable manner. Although the direct impacts are still more noticeable in the academic and research environment, the development of noise mitigation techniques has the potential to benefit industries and companies that will eventually employ quantum computing to solve complex problems in less time and with greater efficiency. Thus, this work represents a significant step toward the evolution of quantum computing and its practical applicability, strengthening Brazil as an active player in this global technological revolution.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE SÍMBOLOS

SUMÁRIO

1 INTRODUÇÃO

No início do desenvolvimento dos primeiros computadores, com a *Máquina de Turing* em 1936 e, posteriormente, a criação dos transistores nos Laboratórios da *Bell Telephone*, em 1947 (**??**), e sua evolução até passar a ser utilizado na construção de processadores, muito já se pensava a respeito dos incríveis feitos que essa emergente tecnologia poderia proporcionar, e realmente podemos ver tal expectativa sendo concretizada nos dias atuais, com os computadores sendo acessíveis até mesmo da palma da mão, com os *smartphones*, algo inimaginável àquela época.

Contudo, embora ainda seja – e continuarão sendo – demasiadamente útil, existem novos problemas que exigem uma forma diferente de processamento, como Richard Feynman afirmou em 1981 na primeira Conferência da Física da Computação, na qual sugeriu que computadores quânticos poderiam realizar simulações que são impossíveis de serem realizadas pelos computadores clássicos, mesmo aqueles mais robustos. Isso se deve à natureza intrínseca aos próprios problemas, como, por exemplo, a simulação de moléculas grandes ou materiais complexos, que exigem um poder de processamento que cresce exponencialmente com o aumento do número de partículas. Em adição, Feynman também descreveu o mundo físico como “quântico”, e como tal, sistemas físicos (quânticos) apenas seriam adequadamente simulados por meio de computadores que utilizassem princípios quânticos de processamento (**??**).

A partir desse ponto, muitos adeptos e fomentadores da ideia contribuíram com conhecimento, como David Deutsch, que propôs o primeiro algoritmo quântico, mesmo que com aplicações limitadas, foi demonstrada uma eficiência muito superior aos algoritmos clássicos (**??**), além de servir de suporte para o desenvolvimento de outros algoritmos importantes, como os Algoritmos de Simon (**??**) e de Shor . Também Peter Shor mostrou com seu algoritmo quântico de fatoração que um algoritmo quântico poderia fatorar números inteiros exponencialmente mais rápido que algoritmos clássicos (**??**). E, ainda, Lov Grover, que apresentou um algoritmo de busca em listas desordenadas, que opera por meio de uma técnica de amplificação tanto de amplitudes quanto das probabilidades e que possui melhora quadrática na eficiência em relação a métodos clássicos conhecidos (**??**).

Seguindo o viés já apresentado, este trabalho tem por objetivo se envolver no estudo, construção, melhoria de resultados e análise do Algoritmo de Grover aplicado a um sistema de quatro qubits, a fim de melhor compreendê-lo, e ainda conseguir gerar conteúdo que poderá ser base para novos trabalhos. As melhorias propostas neste trabalho estão relacionadas aos métodos de supressão – que é uma estratégia preventiva, aplicada durante a execução do circuito quântico para minimizar a geração de erros – e de mitigação – ocorre após a execução do circuito quântico, com técnicas que tentam corrigir ou compensar os erros nos resultados obtidos – de ruídos quânticos gerados pelas *Quantum Processing Unit* (QPUs) nos Computadores Quânticos (CQs). Ambas estratégias possuem diversos métodos incluídos no *Qiskit*, alguns deles serão empregados, suas influências na qualidade final dos resultados serão analisadas e discutidas no discorrer deste texto.

2 FUNDAMENTAÇÃO TEÓRICA

A computação quântica vem ganhando destaque por sua capacidade de solucionar problemas considerados intratáveis para sistemas clássicos. Entre os algoritmos quânticos, o Algoritmo de Grover se destaca por oferecer uma melhoria quadrática na busca em bases de dados não estruturadas. Entretanto, a presença de ruídos nas QPUs – oriundos de fontes como decoerência, interação com o meio, imperfeições nas operações quânticas, dentre outras – impõe desafios que requerem a implementação de estratégias que tornem os erros menos impactantes no resultado final. Este referencial teórico visa fundamentar os conceitos necessários para a compreensão tanto do funcionamento e posterior implementação do Algoritmo de Grover quanto dos métodos de supressão e mitigação de ruídos que serão aplicados.

Na computação quântica, os *quantum bits*, ou bits quânticos – chamados de *qubits* – são a unidade fundamental de informação. Diferente dos bits clássicos, que assumem apenas os valores 0 ou 1, os qubits podem assumir estados de superposição, que podemos entender como uma condição em que o qubit "pode também assumir os dois valores simultaneamente. Nessa situação, diz-se que um qubit está em superposição de estados, ou coerência" (**??**); além disso, o computador quântico é capaz de realizar o processamento paralelo das informações contidas nos qubits, denominado de paralelismo quântico – quer dizer, o CQ consegue atuar em todos os qubits de forma síncrona, logo, é possível processar toda a informação simultaneamente. Um outro conceito extremamente importante quando se fala não apenas de computação quântica especificamente, mas presente no cerne da mecânica quântica, é o que chamamos de emaranhamento quântico, que é uma "consequência direta da condição de superposição" (**??**), e, em palavras simples, pode ser entendido como uma forte correlação entre diferentes partículas, de modo que o estado de uma não pode ser descrito de forma desvinculada ao estado da outra, independentemente da distância que as separam, devido ao princípio da não localidade, que Einstein chama de "ação fantasmagórica à distância". Ainda segundo Rigolin **??**, "os estados emaranhados podem ainda serem usados para se realizar eficientemente tarefas impossíveis de serem executadas por meio de recursos clássicos [...], como o teletransporte quântico".

Os *chips* quânticos da IBM (QPUs) operam utilizando qubits supercondutores, que são projetados para reduzir a sensibilidade ao ruído de carga e aumentar a estabilidade dos estados quânticos. Para manter a supercondutividade, esses qubits operam a temperaturas extremamente baixas, próximas do zero absoluto. Esses qubits são manipulados pelas portas lógicas quânticas, isto é, o estado de qubit é modificado pela aplicação de pulsos eletromagnéticos de micro-ondas que induzem transições nos níveis de energia dos mesmos (**??**) Dentre as portas lógicas quânticas existentes, podem ser citadas algumas como, por exemplo, as portas *Hadamard* (H), *Pauli-X* (X), *Pauli-Z* (Z), *X-Controlada* (C_x, $CNOT$), *Z-Controlada* (C_Z) e a porta *Toffoli* (*X-Multi-Controlada* de três qubits (MCX), que são portas lógicas quânticas muito conhecidas, e disponíveis no *Qiskit*, e quase todas serão abordadas no decorrer deste documento, visto que serão necessárias para a construção do circuito quântico (mais detalhes sobre construção e operação de cada uma serão dados no Capítulo 4). Essas portas lógicas operam de forma a explorar as peculiaridades da mecânica quântica para efetuar cálculos complexos (**??**).

O Algoritmo de Grover, foco deste objeto, foi proposto por Grover **??** e tem como objetivo acelerar a busca em bases de dados não ordenadas. Seu funcionamento pode ser resumido em três etapas principais: inicialização, em que há a criação do Espaço de Pesquisa, que nada mais é que a superposição uniforme de todos os estados possíveis; aplicação do oráculo, uma operação que marca (adicionando uma fase negativa) o(s) estado(s) que correspondem à solução do problema; e, por fim, aplicação do Operador de Difusão ou de Amplificação, que amplifica a probabilidade do estado marcado, tornando-o mais provável de ser medido (**??**). A eficiência do algoritmo de Grover se evidencia na sua capacidade de reduzir o número de buscas necessárias em comparação com os métodos clássicos, o que o torna uma ferramenta promissora em diversas aplicações, como a quebra de sistemas criptográficos e otimização de buscas (**??**).

Apesar do potencial dos algoritmos quânticos, os computadores quânticos atuais sofrem com a presença de ruído – decorrente de decoerência dos qubits, imperfeições nas portas lógicas, interação com o meio e erros de leitura. Esses fatores podem degradar significativamente a fidelidade dos resultados dos algoritmos, e uma fidelidade alta é crucial para garantir que os estados quânticos sejam preservados com alta precisão durante as operações (**??**). Tomando por base esta asserção, fica evidente a necessidade e a importância da utilização de métodos que possam restringir a influência do ruído quântico, de modo a garantir fidelidades maiores nos estados finais após as operações, gerando, por conseguinte, melhores e mais confiáveis resultados.

O *Qiskit SDK* possui uma gama de ferramentas para supressão e mitigação de ruído quântico, e é intuito deste trabalho aplicar algumas delas ao circuito quântico e analisar suas efetividades, emphi. e., a capacidade de cada método de melhorar o resultado final. São eles: Otimização do circuito compilado, Desacoplamento Dinâmico (DD), Compilação Aleatória (*Pauli Twirling*), Extrapolação de Erro Zero (ZNE), Extinção de Erro de Leitura Giratória (TREX), Amplificação de Erro Probabilístico (PEA), Cancelamento de Erro Probabilístico (PEC).

2.1 Métodos de Supressão e Mitigação

Como discutido por Preskill **??**, o mundo experimenta no cenário atual a era da tecnologia Quântica de Escala Intermediária Ruidosa (NISQ, na sigla em inglês para *Noisy Intermediate-Scale Quantum*), que descreve o "tamanho" dos CQs disponíveis atualmente em termos de quantidade de qubits (entre 50 e algumas centenas), enquanto o termo "ruidoso" faz jus ao objeto de estudo desta produção: ruído quântico, que, ainda segundo ele, advém da incapacidade de se controlar fielmente as interações dos qubits.

> "[...] com esses dispositivos barulhentos não esperamos ser capazes de executar um circuito que contenha muito mais do que cerca de 1000 portas — isto é, 1000 operações fundamentais de dois qubits — porque o ruído sobrecarregará o sinal em um circuito muito maior do que isso. Essa limitação no tamanho do circuito impõe um teto ao poder computacional da tecnologia NISQ. Eventualmente, faremos melhor, usando correção de erro quântico para escalar para circuitos maiores." (**??**).

Dado o exposto, torna-se imprescindível a utilização de pelo menos alguns dos métodos que serão aqui introduzidos, cabendo ao usuário optar por aqueles que mais se adequam às características intrínsecas a cada circuito, específicas de cada aplicação. é esperado que este compêndio seja útil para servir de fonte de pesquisa para auxiliar nessa tomada de decisão.

2.1.1 Supressão

As técnicas de supressão de ruído em computação quântica visam reduzir o surgimento do ruído durante o processamento ou execução do circuito (como decoerência e erros de porta), e, por conseguinte, evitar os efeitos indesejados sem recorrer, necessariamente, a códigos completos de correção de erros. Essas técnicas são especialmente relevantes no contexto dos dispositivos NISQ, onde a presença de ruído ainda é significativa. As técnicas de supressão contempladas pelo *Qiskit* e que aqui também o serão, são:

Otimização do Circuito Quântico

A otimização ocorre na fase de compilação, e é um método que deve ser utilizado durante o processo de mapeamento do circuito quântico virtual[1] para o circuito quântico físico[2] utilizando para isso a função "transpile", que possui quatro possíveis níveis: nível 0, que não há otimização; nível 1, que combina decomposição otimizada de portas de um qubit com cancelamento de portas inversas consecutivas; nível 2, que utiliza um método de cancelamento comutativo em vez de cancelamento de C_X's, resultando na diminuição de portas redundantes; nível 3, que é o nível de otimização mais eficaz, embora mais exigente em se tratando de poder computacional, pois emprega vários métodos, inclusive decomposição otimizada de portas de um qubit, usada no nível um, e cancelamento comutativo aplicado no nível dois, além de outros métodos como coleção de subcircuitos de dois qubits, consolidação de portas lógicas (blocos) consecutivas atuantes em um mesmo qubit para um único bloco mais otimizado, e, por fim, utiliza síntese unitária, que pode fazer uma simplificação das portas por suas portas-base (**??**).

Decomposição Otimizada de Portas de um Qubit (**??**): Esse método visa otimizar cadeias de portas de qubit único combinando-as em uma única porta. As condições empregadas pelo algoritmo para tomar a decisão de substituir o conjunto original de portas por uma nova configuração são: se a corrente original estava fora da base: substituir; se a cadeia original estava na base, mas a ressíntese é menor erro: substituir; se a cadeia original contém uma porta de pulso: não substitua; se a cadeia original equivale à identidade: substituir por nulo; O erro é calculado como uma multiplicação dos erros de portas individuais naquele qubit.

Cancelamento Inverso (**??**): O objetivo desse método é identificar e remover pares de portas quânticas consecutivas que são inversas uma da outra, otimizando assim o circuito ao eliminar

[1] Um circuito quântico virtual refere-se à representação abstrata e idealizada de um circuito quântico, em que não há interferência de ruídos quânticos ou limitações tecnológicas da implementação física.

[2] Um circuito quântico físico é aquele implementado em hardware real, também chamado *backend*, utilizando as portas lógicas que o mesmo suporta.

operações redundantes. Ao aplicá-lo, é possível reduzir a profundidade do circuito e minimizar possíveis fontes de erro, contribuindo para a execução mais eficiente de algoritmos quânticos.

Cancelamento Comutativo (**??**): O objetivo desse método é identificar e remover portas auto-inversas redundantes, aproveitando as relações de comutação no circuito. As portas consideradas incluem: H, X, Y, Z, C_X, C_Y, C_Z. Ao aplicá-lo, é possível reduzir a complexidade do circuito, eliminando operações redundantes e melhorando a eficiência da execução em dispositivos quânticos.

Coleção de Subcircuitos de 2 Qubits (**??**): é uma ferramenta de análise que identifica e agrupa subcircuitos compostos por portas de dois qubits que são adjacentes (vizinhas) e atuam nos mesmos qubits. Esses blocos coletados podem ser posteriormente otimizados ou sintetizados de maneira mais eficiente por outras passagens do *transpile*. Ao utilizá-la, é possível preparar o circuito para otimizações subsequentes, como a consolidação de operações ou a redução da profundidade do circuito, melhorando a eficiência da execução em dispositivos quânticos.

Consolidação de Blocos de Portas Consecutivas (**??**): Essa é uma transformação que substitui sequências de portas consecutivas atuando nos mesmos qubits por um único nó unitário. Geralmente, esses blocos são selecionados anteriormente por outra ferramenta, como a Coleção de Subcircuitos de 2 Qubits. Essa consolidação permite que o subcircuito seja ressincronizado posteriormente, potencialmente resultando em uma implementação mais otimizada. Além disso, é possível reduzir a complexidade do circuito, agrupando operações em unidades únicas que podem ser mais eficientes para execução em hardware quântico.

Síntese Unitária (**??**): O objetivo dessa função é sintetizar operações unitárias, substituindo sequências de portas por implementações equivalentes que sejam mais eficientes ou que atendam a restrições específicas do hardware. Funciona através da análise do circuito em busca de subcircuitos unitários e os reescreve utilizando um conjunto específico de portas base, visando otimizar o circuito de acordo com as características do *backend* (CQ) alvo.

Desacoplamento Dinâmico (DD, *Dynamical Decoupling*)

“Circuitos quânticos são executados em hardware IBM como sequências de pulsos de micro-ondas que precisam ser agendados e executados em intervalos de tempo precisos.” (**??**). Durante uma operação qualquer, alguns qubits podem passar alguns intervalos de tempo sem que estejam sendo utilizados, devido à existência de lacunas nos circuitos; esses qubits podem acabar interagindo entre si de forma não controlada e indevida. Essas interações são responsáveis por gerar quantidades significativas de erros no resultado final. O DD é muito útil para este tipo de circuitos, onde há lacunas, pois "é uma técnica de controle quântico que consiste em aplicar sequências personalizadas de pulsos ao sistema quântico considerado para cancelar (ou fazer a média) da interação com o ambiente" (**??**), ou seja, são inseridas sequências de pulsos de micro-ondas nos qubits ociosos que são propositalmente simétricos, para que sua média total seja cancelada, de modo que se equivalham à operação de identidade, isto é, não causem perturbações no sistema. Esses pulsos controlados agem de modo a suprimir erros de interação indevida, ao impedir que haja a interação entre qubits.

Compilação Aleatória (*Randomized Compiling* ou *Pauli Twirling*)

O método de *Pauli twirling* é uma técnica de supressão de erros que transforma canais de ruído arbitrários em canais do tipo Pauli, os quais são mais fáceis de modelar e corrigir. A ideia central é aplicar, de forma aleatória, operações do grupo de Pauli – como os operadores I, X, Y e Z – antes e depois de um canal de ruído. Ao se fazer a média (ou "*twirling*") sobre essa distribuição aleatória de operações, os termos não diagonais (ou seja, os erros coerentes) tendem a se cancelar, resultando em um canal de ruído que age de forma puramente estocástica[3]. Em outras palavras, o ruído original, que pode apresentar comportamentos complexos e coerentes, é transformado em um ruído que se comporta como uma mistura probabilística de erros simples (por exemplo, erros de *bit-flip* ou *phase-flip*). Essa transformação facilita tanto a análise teórica quanto a mitigação prática dos erros, pois canais de ruído do tipo Pauli são bem compreendidos e possuem uma estrutura que pode ser incorporada em protocolos de correção de erros e técnicas de mitigação, como o *randomized benchmarking*.

> "Realizado sozinho, ele pode mitigar ruído coerente porque o ruído coerente tende a se acumular quadraticamente com o número de operações, enquanto o ruído de

[3] Um erro estocástico ocorre de forma aleatória, seguindo uma distribuição probabilística, ao contrário dos erros coerentes, que se manifestam de maneira sistemática. Esse comportamento possibilita que técnicas de correção e mitigação sejam aplicadas de forma mais eficaz, pois os efeitos do ruído podem ser tratados estatisticamente (**??**)

> Pauli se acumula linearmente. O twirling de Pauli é frequentemente combinado com outras técnicas de mitigação de erros que funcionam melhor com ruído de Pauli do que com ruído arbitrário"(**??**).

Conforme discutido por Emerson et al. **??**, essa técnica é fundamental para converter erros que se acumulam de maneira prejudicial em erros estocásticos, cuja influência pode ser controlada e, em muitos casos, mitigada através de métodos de pós-processamento. Trabalhos posteriores, como o de Temme et al. **??**, demonstram aplicações práticas dessa técnica para melhorar a fidelidade de circuitos quânticos de curta profundidade, tornando os resultados experimentais mais próximos do ideal.

2.1.2 Mitigação

Técnicas de mitigação são utilizadas como complemento às de supressão, aplicadas no pós-processamento com o intuito de compensar os efeitos dos erros induzidos nos resultados medidos pela ação do ruído que não pôde ser totalmente cancelado na supressão.

Extinção de Erro de Leitura Giratória (TREX, *Twirled Readout Error eXtinction*)

O *Twirled Readout Error eXtinction* (TREX) é uma técnica de mitigação de erros, especificamente projetada para reduzir os erros associados à leitura (*readout*) dos qubits durante as medições. Essa abordagem é baseada no conceito de *twirling* de medições, que envolve a aplicação aleatória de operações de Pauli X antes das medições, seguida de uma inversão correspondente dos bits medidos. O objetivo principal do TREX é diagonalizar a matriz de transferência de erro de leitura, facilitando sua inversão e, consequentemente, a correção dos erros de leitura. O funcionamento dessa técnica consiste na substituição (de forma aleatória) do processo de medição, em que uma medição simples é trocada por um processo de medida composto de três etapas: primeiro, os canais são submetidos a Portas X, que invertem o estado dos qubits (*bit-flip*) nos quais atuaram; segundo, após as aplicações das Portas X, é realizada uma medição comum; por fim, é feito o ajuste dos estados, de modo que os canais em que houve atuação de Portas X antes da medida têm seu estado corrigido (invertido) classicamente. Essa sequência de operações transforma o canal de erro de leitura original em uma forma diagonal, onde os elementos fora da diagonal principal são minimizados. Essa diagonalização simplifica a inversão da matriz de erro, permitindo uma mitigação mais eficaz dos erros de leitura. O método TREX apresenta diversas vantagens no contexto da mitigação de erros em medições quânticas. Uma de suas principais qualidades é o fato de não assumir um modelo específico de ruído, o que o torna uma técnica versátil e aplicável a uma ampla variedade de sistemas quânticos. Além disso, ao incorporar uma etapa de randomização com operações de Pauli antes da medição, o TREX contribui para a redução de erros correlacionados entre qubits. Outro ponto favorável é a simplicidade de sua implementação: o método exige apenas a inserção de operações quânticas simples antes das medições e a correção clássica correspondente nos resultados, dispensando o uso de circuitos quânticos adicionais complexos ou ajustes estruturais profundos no experimento.

Extrapolação de Ruído Zero (ZNE, *Zero-Noise Extrapolation*)

O método de *Zero Noise Extrapolation* (ZNE) é uma técnica de mitigação de ruído que tem como objetivo estimar o valor ideal de uma medida quântica – ou seja, o valor que seria obtido caso não houvesse ruído no sistema. Essa estimativa é feita a partir da execução de múltiplas versões do mesmo circuito quântico, cada uma com diferentes níveis de ruído artificialmente aumentados. A ideia central é que, ao entender como o ruído afeta os resultados, é possível extrapolar os valores obtidos para o caso em que o ruído é zero. A implementação do ZNE consiste, em linhas gerais, em três etapas principais. Primeiramente, realiza-se a amplificação do ruído no circuito original. Essa amplificação pode ser feita, por exemplo, repetindo certas portas quânticas, como portas $CNOT$, o que efetivamente aumenta o tempo de execução do circuito e, consequentemente, sua exposição ao ruído. Em seguida, executa-se o circuito várias vezes, cada vez com um fator diferente de amplificação de ruído, e coleta-se os resultados das medidas. Por fim, aplica-se uma técnica matemática de extrapolação – como a extrapolação linear, quadrática ou de Richardson – para estimar o resultado que o circuito teria fornecido na ausência de ruído. Entre as vantagens do ZNE, destaca-se o fato de que a técnica não exige modificações no hardware quântico, podendo ser aplicada com acesso apenas à execução de circuitos e ao controle sobre o tempo de operação ou o número de repetições de determinadas portas. Além disso, é uma abordagem compatível com diversos modelos de ruído, o que a torna bastante versátil. No entanto, essa técnica também apresenta desvantagens. A principal delas é o aumento no número total de execuções necessárias, o que pode ser um obstáculo em dispositivos com tempo de acesso limitado. Além disso, o processo de extrapolação pode ser sensível a flutuações estatísticas nos resultados, especialmente em hardwares instáveis ou com ruído não escalável. Dito isso, embora muitas vezes a ZNE possa realmente melhorar os resultados, não é garantido que os resultados gerados sejam sempre condizentes com a realidade. Mesmo assim, trabalhos como o de Giurgica-Tiron et al. **??** demonstram a eficiência do método para estimativas digitais mitigadas de erro em circuitos quânticos ruidosos.

Amplificação de Erro Probabilística (PEA, *Probabilistic Error Amplification*)

A técnica conhecida como *Probabilistic Error Amplification* (PEA) é uma abordagem que visa amplificar de maneira controlada os efeitos dos erros em circuitos quânticos para facilitar sua quantificação e, posteriormente, a mitigação desses erros. A ideia central é executar versões modificadas do circuito original onde as operações propensas a introduzir erro são replicadas ou escaladas de forma probabilística. Esse procedimento resulta em um cenário no qual os efeitos do ruído são enfatizados, permitindo extrair informações mais precisas sobre a natureza e a magnitude dos erros presentes no sistema. O método se fundamenta na premissa de que, ao amplificar o erro, é possível utilizar técnicas matemáticas de extrapolação para estimar o valor que seria obtido em um cenário ideal, ou seja, na ausência de ruído. Essa abordagem é particularmente útil para dispositivos NISQ, onde o ruído é inevitável e pode comprometer a fidelidade dos resultados dos algoritmos quânticos. Ao amplificar os erros, o PEA torna mais fácil identificar e corrigir os erros coerentes, convertendo-os em erros estocásticos, os quais tendem a se comportar de forma mais aleatória e, portanto, podem ser gerenciados de maneira mais eficaz por técnicas de correção ou mitigação. O PEA compartilha fundamentos com outras técnicas de mitigação de erros, como o *Zero Noise Extrapolation* (ZNE) e o *Probabilistic Error Cancellation* (PEC). No entanto, a diferença reside no foco: enquanto o ZNE e o PEC procuram reduzir o impacto dos erros por meio de estratégias de pós-processamento e inversão do canal de ruído, o PEA intencionalmente amplifica o sinal de erro para que ele se torne mensurável com maior precisão e, então, possa ser compensado. Essa estratégia permite que as variações induzidas pelo ruído sejam mais facilmente analisadas, servindo de base para métodos de correção que ajustem os valores esperados dos observáveis. Trabalhos publicados, como o de Temme et al. **??** e Endo et al. **??**, discutem de maneira aprofundada como a amplificação e o cancelamento probabilístico dos erros podem melhorar a precisão dos resultados em circuitos quânticos ruidosos. De acordo com Temme et al. **??**, amplificar os erros de forma controlada permite uma melhor estimativa dos efeitos do ruído, possibilitando uma correção mais efetiva e elevando a fidelidade dos resultados. Já Endo et al. **??** demonstram a aplicabilidade prática de técnicas de mitigação em dispositivos quânticos de curto tempo de execução, ressaltando a importância de estratégias que convertam erros coerentes em erros estocásticos, os quais são mais fáceis de serem gerenciados. No entanto, em seu trabalho, Kim et al. **??** alertam sobre como alguns "resultados podem desviar arbitrariamente do resultado desejado" quando se tenta aplicar a amplificação de ruídos em modelos que sejam mesmo que ligeiramente mais complexos. Em resumo, o PEA consiste em amplificar os erros de forma probabilística, para então utilizar técnicas de extrapolação que estimem o valor ideal do circuito, ou seja, o valor que se obteria na ausência de ruído. Essa abordagem se mostra promissora para melhorar a precisão dos resultados em dispositivos quânticos atuais e é uma ferramenta complementar importante de mitigação de ruído quântico.

Cancelamento de Erro Probabilístico (PEC, *Probabilistic Error Cancellation*)

O Cancelamento de Erro Probabilístico é uma das técnicas mais potentes de mitigação de erros em sistemas quânticos NISQ, pois permite obter estimativas não enviesadas dos valores esperados de observáveis mesmo na presença de ruído significativo. A ideia central do PEC é representar cada operação quântica ideal como uma combinação linear de operações ruidosas que o hardware realmente implementa, usando quasi-probabilidades η_i que podem assumir valores negativos ou maiores que 1 para "inverter" estatisticamente o efeito do ruído (**??**). Na prática, isso significa primeiro caracterizar o canal de ruído do dispositivo por meio de protocolos de tomografia (por exemplo, *Pauli-Lindblad tomography*), de modo a aprender o comportamento exato de $U_{ruidoso,i}$, o canal ruidoso que substitui a operação ideal U_{ideal} (**????**). Em seguida, cada porta ideal é decomposta como

$$U_{ideal} = \sum_i \eta_i \, U_{ruidoso,i} \ ,$$

onde cada $U_{ruidoso,i}$ é uma implementação ruidosa realizável no *hardware*, e η_i são os coeficientes da distribuição de quasi-probabilidade associada.

Para estimar o valor esperado de um observável O, o PEC executa um conjunto de circuitos ruidosos, amostrados de acordo com a distribuição de probabilidade $P(i) = \frac{|\eta_i|}{\gamma}$, sendo $\gamma = \sum_i |\eta_i|$ o fator de sobrecusto, e computa uma média ponderada que incorpora o sinal $sign(\eta_i)$. Esse procedimento produz um estimador não enviesado, mas requer $O(\gamma^2)$ execuções do circuito, o que costuma crescer exponencialmente com a profundidade do circuito (**??**).

Essa sobrecarga de amostragem é rigorosamente limitada por resultados recentes que estabelecem limites universais para o custo de protocolos de mitigação de erros, mostrando que qualquer método genérico – incluindo o PEC – precisa de um número de amostras proporcional a γ^2 para alcançar uma dada precisão com alta probabilidade (**??**).

Em síntese, o Cancelamento Probabilístico de Erro oferece estimativas estatisticamente exatas dos valores esperados em circuitos quânticos ruidosos, mas demanda um custo de amostragem elevado – governado pelo fator γ – e rigorosa caracterização do canal de ruído. Por essa razão, ele é particularmente indicado para circuitos de baixa profundidade e cenários onde a precisão não enviesada seja crucial para os resultados (**??**).

3 FERRAMENTAS E MÉTODOS

Esta seção visa descrever as ferramentas e os métodos utilizados para o desenvolvimento e a avaliação do algoritmo de Grover em dispositivos quânticos. Inicialmente, apresenta-se o uso de simuladores quânticos, em especial o AerSimulator, que permite a emulação de circuitos quânticos sob condições realistas de ruído, possibilitando uma análise mais próxima do comportamento esperado em hardware físico.

Em seguida, são detalhadas as características dos computadores quânticos utilizados nos testes, especificamente os *backends* disponibilizados pela IBM Quantum. A escolha de múltiplos dispositivos com diferentes configurações visa proporcionar uma avaliação comparativa do desempenho do algoritmo em distintas arquiteturas e condições operacionais.

Por fim, é abordada a estrutura matemática e funcional do Algoritmo de Grover, descrevendo-se as principais etapas necessárias para sua implementação e destacando-se os conceitos fundamentais que sustentam sua eficiência em relação aos métodos clássicos de busca.

Esse conjunto de ferramentas e procedimentos estabelece a base metodológica para os experimentos realizados, permitindo uma análise abrangente dos resultados obtidos.

3.1 Dispositivos Quânticos

3.1.1 Simuladores

Para avaliar o desempenho do algoritmo de Grover em condições que refletem as imperfeições de um dispositivo quântico real, será utilizado o simulador *AerSimulator* (**??**), uma classe pertencente ao *Qiskit Aer* (**??**), que, por sua vez, é um módulo do *Qiskit*. Este simulador possibilita a emulação de circuitos quânticos com a inclusão de modelos de ruído específicos de cada QPU, proporcionando uma análise mais realista do algoritmo desenvolvido. Isso ocorre porque o *AerSimulator* é capaz de reproduzir, de forma aproximada, o comportamento de dispositivos reais, previamente selecionados, permitindo avaliar o desempenho do algoritmo de Grover sob as limitações físicas e operacionais típicas do *hardware* quântico. Para fins de comparação, serão utilizados os modelos de ruído correspondentes aos mesmos dispositivos quânticos (*backends*) reais que, posteriormente, computarão o algoritmo.

3.1.2 Computadores

Para realizar os testes, foram utilizados dois diferentes *backends*: ibm_brisbane e ibm_torino. O *ibm_brisbane* é um processador baseado na tecnologia Eagle r3, com 127 *qubits* e conectividade otimizada para execução de circuitos de médio porte. Já o *ibm_torino* é um dispositivo mais recente, baseado na arquitetura Heron r1, com 133 *qubits* e melhorias no tempo de coerência e fidelidade de portas. Todos são de acesso livre através do serviço de computação em nuvem IBM Quantum (**??**).

Esses *backends* correspondem a Processadores Quânticos (QPUs) supercondutores que operam com qubits físicos implementados por circuitos de transmon. A IBM disponibiliza informações detalhadas sobre cada um de seus dispositivos, incluindo a topologia de acoplamento, as portas quânticas nativas, bem como métricas importantes para a execução de algoritmos, como tempos de coerência (T_1 e T_2), fidelidades de operação e taxas de erro de leitura (**??**).

O uso desses *backends* permite avaliar o desempenho do algoritmo de Grover não apenas em ambientes simulados, mas também em dispositivos físicos sujeitos a ruído real, decoerência e demais limitações tecnológicas inerentes aos atuais computadores quânticos de uso geral (**??**). Além disso, cada *backend* possui configurações distintas de conectividade entre qubits e diferentes níveis de fidelidade operacional, o que possibilita uma análise comparativa do impacto dessas variáveis na execução do algoritmo. O Quadro 3.1 mostra como exemplo, dois *backends* disponíveis na *IBM Quantum Platform*, e que serão utilizados tanto para simulações quanto para as execuções nas QPU's reais.

Quadro 3.1 – Configurações dos Computadores Quânticos utilizados

Backends	N° qubits	CLOPS	QPU	Portas Base
ibm_brisbane	127	180k	Eagle r3	ECR, ID, RZ, SX, X
ibm_torino	133	210k	Heron r1	CZ, ID, RX, RZ, RZZ, SX, X

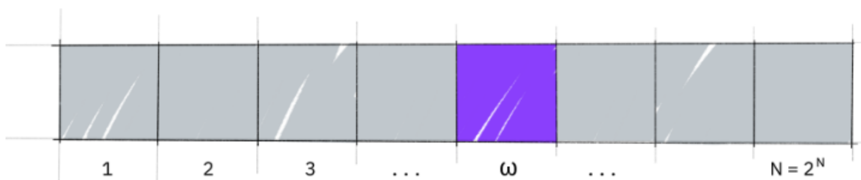
Fonte: (**??**)

Nesse quadro pode-se analisar a quantidade de *qubits* (N° qubits), o número de CLOPS[1] (*Circuit Layer Operations Per Second*), o tipo de processador (QPU) e as portas base[2] de cada *backend*.

3.2 Visão Geral do Algoritmo de Grover

Como já mencionado, o Algoritmo de Grover apresenta um método de busca em listas que não possuem qualquer tipo de ordenação com uma eficiência de ordem quadrática, por meio de uma técnica de amplificação tanto de amplitudes quanto das probabilidades. Seja a ilustração apresentada na Figura 3.1,

Figura 3.1 – Lista com 2^n itens e um elemento ω marcado.



Fonte: Github/*Qiskit*

Aqui tem-se ilustrado um conjunto contendo $N = 2^n$ elementos organizados aleatoriamente. Nele, há um termo ω destacado, nomeado "*winner*", que é justamente o elemento procurado. Se quisesse-se buscar o item marcado por meio de computação clássica, seriam necessárias, em média, $\frac{N}{2}$ iterações, contudo, ao implementarmos o Algoritmo de Grover, esse valor cai para $\sqrt{N}$, que é um avanço significativo, principalmente quando considera-se manipular conjuntos com grandes quantidades de itens (**??**).

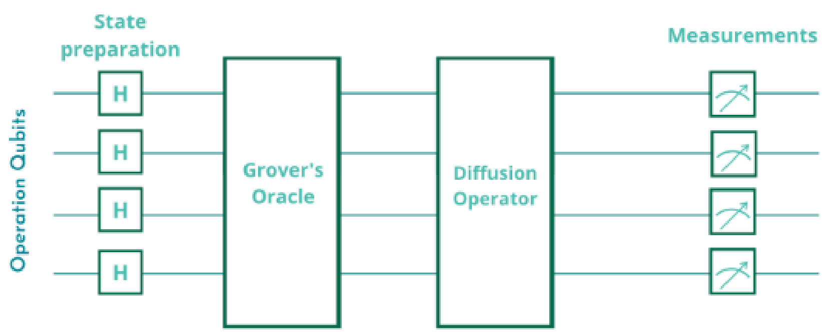
[1] CLOPS é uma métrica criada pela IBM para medir quantas camadas de portas quânticas um processador consegue executar por segundo. Camadas, por sua vez, são as operações que podem ser executadas em paralelo, *i. e.*, portas lógicas atuando em diferentes canais (*qubits*) de forma independente podem ser executados ao mesmo tempo.

[2] Portas Base, ou *basis gates*, representam o conjunto de operações nativas que um processador quântico consegue executar diretamente no *hardware*, *i. e.* as portas que não precisam ser reescritas no processo de compilação do circuito virtual para o físico.

3.2.1 Estrutura

Para que seja possível compreender a atuação do algoritmo, é preciso entender sua estrutura, que conta com três partes necessárias para que possa ser implementado, sendo elas: preparação inicial do estado, determinação do Oráculo (marcação do estado buscado) e amplificação da amplitude e probabilidade (aplicação do Operador de Grover ou Operador de Difusão). Isso pode ser exemplificado pelo esquema da Figura 3.2.

Figura 3.2 – Esquematização da estrutura do Algoritmo de Grover.



Fonte: Github/*Qiskit*.

Na preparação do estado há a criação do que chamaremos de espaço de pesquisa, é o conjunto que contém o elemento marcado e onde ele será procurado.

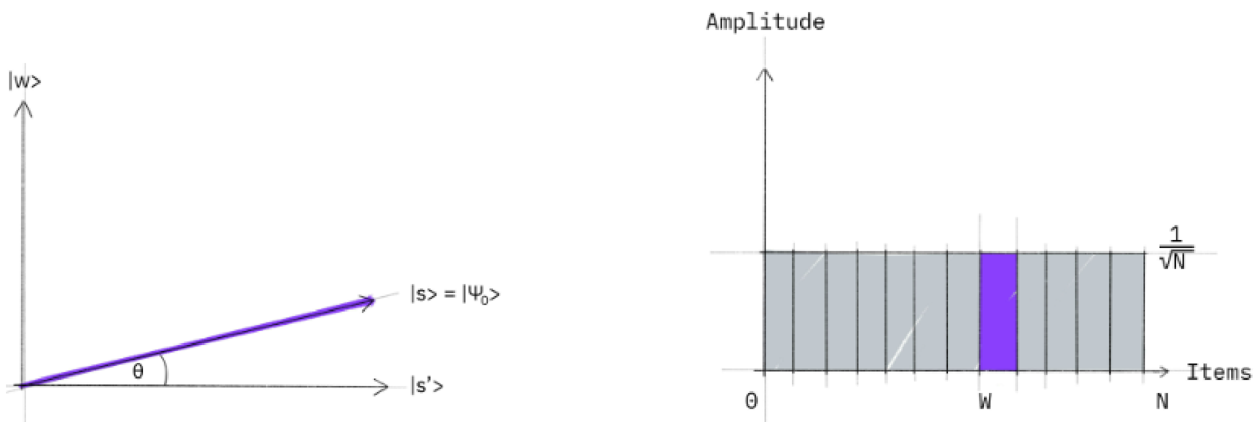
Feito isso, seguiremos para a determinação do item que queremos encontrar, que é feita pelo Oráculo. A atuação do Oráculo consiste em inverter a fase do elemento de interesse, ou seja, ele marca o item buscado. Caso haja mais de um item de interesse, o Oráculo deve agir em todos eles, marcando-os, de modo que sejam destacados pelo difusor.

Por fim, utiliza-se o Operador de Difusão para aumentar a amplitude do estado marcado, enquanto decresce as demais, sendo assim uma garantia de que na medida final, o resultado obtido seja aquele procurado.

Quando trabalha-se com o Algoritmo de Grover, pode-se reduzir o problema a um espaço de duas dimensões, sendo necessário considerar apenas o estado buscado – *winner*, $|\omega\rangle$ –, e a superposição uniforme, $|s\rangle$. Contudo, esses não são vetores perpendiculares entre si, uma vez que $|\omega\rangle$ ocorre em superposição com amplitude $\frac{1}{\sqrt{N}}$. Para contornar isso, cria-se um vetor $|s'\rangle$

que é perpendicular a $|\omega\rangle$, formando assim o plano bidimensional apresentado na Figura 3.3a. Na Figura 3.3b, estão ilustradas as amplitudes dos itens contidos no conjunto, evidenciando o elemento *winner*.

Figura 3.3 – Plano Bidimensional e Amplitude dos estados da base.



(a) Plano formado pelos vetores $|\omega\rangle$ e $|s'\rangle$.

(b) Amplitude dos itens do conjunto, com $|\omega\rangle$ em destaque.

Fonte: Github/*Qiskit*.

3.2.2 Preparação Inicial

A Preparação Inicial é a etapa inicial do algoritmo, na qual o Espaço de Pesquisa é criado, matematicamente descrito como uma superposição uniforme e pela Equação 3.1, e possui tamanho dado por $N = 2^n$, com n sendo o número de qubits.

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |bin(x)\rangle\,, \tag{3.1}$$

com $x \in \{0, 1, 2, \ldots, N-1\}$.

Se uma medição for realizada na base $|x\rangle$, de acordo com o Quinto Postulado da Mecânica Quântica (**??**), a superposição colapsa, podendo resultar em qualquer um dos estados de mesma probabilidade $\frac{1}{N} = \frac{1}{2^n}$.

Em termos de circuito quântico, a superposição explicitada pela Equação 3.1 pode ser construída facilmente aplicando a porta Hadamard em cada um dos qubits, que se iniciam no estado fundamental $|0\rangle$, como representado pela Equação 3.2.

$$|s\rangle = H^{\otimes n}\ |0\rangle^{n} \tag{3.2}$$

O vetor $|s\rangle$ que aparece na Figura 3.3a pode ser escrito em coordenadas polares, como

$$|s\rangle = \sin\theta\,|\omega\rangle + \cos\theta\,|s'\rangle\,, \tag{3.3}$$

em que

$$\theta = \arcsin\langle s|\omega\rangle = \arcsin\frac{1}{\sqrt{N}} \tag{3.4}$$

3.2.3 Oráculo (U_f)

Tomando a premissa de que o elemento ω procurado está em um determinado conjunto contendo $N = 2^n$ itens, $0, 1, 2, ..., N-1$, com $n \in \mathbb{N}$, podemos recorrer a uma função $f : 0, 1, 2, ..., N-1,$ $\rightarrow 1, -1$ que atua na amplitude dos elementos para marcar o buscado, sendo f tal que

$$f(x) = \begin{cases} -1, & \text{se } x = \omega \\ 1, & \text{se } x \neq \omega \end{cases} \tag{3.5}$$

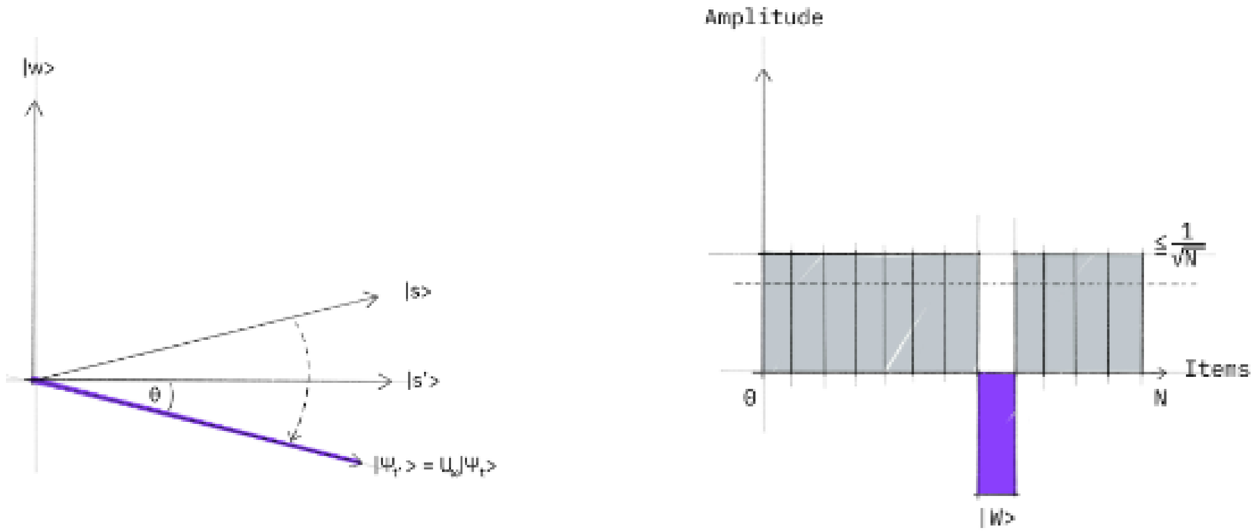
Em outras palavras, o Oráculo, denotado por U_f, nada mais é que uma função que gera a reflexão em um ângulo θ de $|s\rangle$ sobre $|s'\rangle$, bem como a reflexão da amplitude do elemento de interesse $|\omega\rangle$, que pode ser visualizado com a ajuda da Figura 3.4.

De modo geral, a construção dos Oráculos no Algoritmo de Grover pode ser realizada utilizando portas lógicas quânticas do tipo X e MCZ, seguindo a seguinte regra de formação:

1. Para cada qubit cujo valor no estado buscado seja $|0\rangle$, aplica-se uma porta X nesse qubit, realizando assim um *bit-flip*;

2. Em seguida, aplica-se uma porta (MCZ) sobre todos os qubits, marcando o estado desejado com um fator de fase;

Figura 3.4 – Reflexão dos estados $|s\rangle$ e $|\omega\rangle$, respectivamente.



(a) Reflexão de $|s\rangle$ sobre $|s'\rangle$ em um ângulo θ.

(b) Reflexão de π rads na amplitude de $|\omega\rangle$.

Fonte: Github/*Qiskit*.

3. Por fim, aplica-se novamente a porta X nos mesmos qubits que sofreram o *bit-flip* na primeira etapa, revertendo as alterações realizadas inicialmente.

3.2.4 Operador de Difusão (U_s)

A aplicação do Operador de Difusão, denotado por U_s, consiste em operações de amplificação sobre o estado $|s\rangle$, gerando uma reflexão adicional do mesmo.

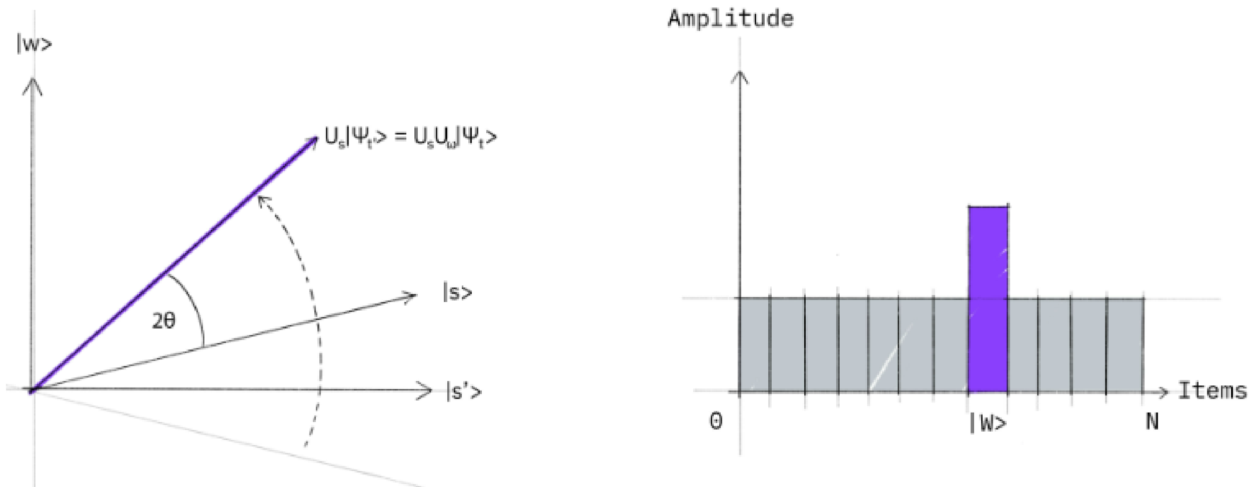
$$U_s = 2\ |s\rangle\langle s| - I \tag{3.6}$$

Essa transformação mapeia o estado de $|s\rangle$ para $(U_s U_f)\,|s\rangle$ e completa a transformação, tal como está apresentado na Figura 3.5.

Duas reflexões sempre resultam em uma rotação, que leva o estado inicial $|s\rangle$ para mais perto do elemento *winner*, $|\omega\rangle$.

Como esta é uma reflexão sobre $|s\rangle$, busca-se adicionar uma fase negativa a cada estado ortogonal a $|s\rangle$. A maneira como pode-se fazer isso é aplicar uma transformaão que leva o estado

Figura 3.5 – Segunda reflexão dos estados $|s\rangle$ e $|\omega\rangle$, respectivamente.



(a) Segunda reflexão de $|s\rangle$ sobre $|s'\rangle$.

(b) Segunda reflexão na amplitude de $|\omega\rangle$.

Fonte: Github/*Qiskit*.

$|s\rangle \rightarrow |0\rangle$, com isso, em vez de U_s, chamar-se-á U_0. Dessa forma, a Equação 3.6 se torna:

$$U_0 = 2\ |0\rangle\langle 0| - I \tag{3.7}$$

Essa operação aplica uma fase negativa a todos os estados ortonormais a $|0\rangle$, como será demonstrado.

[TALVEZ COLOCAR NUM APÊNDICE***]

Aplicação do Operador U_0 em $|s\rangle$, com $|s\rangle$ podendo ser um conjunto arbitrário com $N-1$ itens:

$$U_0\ |s\rangle = (2\,|0\rangle\langle 0| - I)\,\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|bin(x)\rangle\,,$$

Expandindo, tem-se:

$$2\ |0\rangle\langle 0|\left(\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|bin(x)\rangle\right) = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|bin(x)\rangle$$

$$2\ |0\rangle \left(\sum_{x=0}^{15} \langle 0|bin(x)\rangle \right) = \sum_{x=0}^{15} |bin(x)\rangle \tag{3.8}$$

Como $\langle 0|bin(x)\rangle = 1$ apenas se $x = 0$ (enquanto todos os outros termos se tornam 0), o único termo que sobrevive é:

$$2\,|0\rangle\,\langle 0|bin(0)\rangle = 2\,|bin(0)\rangle$$

Então, em síntese, tem-se:

$$\begin{aligned} U_0\ |s\rangle =& 2\ |bin(0)\rangle - \sum_{x=0}^{N-1} |bin(x)\rangle \\ =& 2\ |bin(0)\rangle - |bin(0)\rangle - |bin(1)\rangle - |bin(2)\rangle - \cdots - |bin(N-1)\rangle \\ =& |bin(0)\rangle - \sum_{x=1}^{N-1} |bin(x)\rangle \end{aligned} \tag{3.9}$$

Ou seja, a amplitude $|bin(0)\rangle$ aumenta em relação aos demais que invertem.

Tendo sido demonstrada a forma matemática, passa-se agora para a apresentação da obtenção do circuito quântico que realiza essa operação.

O Operador de Difusão em termos de portas lógicas é dado por:

$$U_s = H^{\otimes n}\ U_0\ H^{\otimes n} \tag{3.10}$$

Como já visto, a atuação de U_0 é aplicar uma fase negativa a todos os estados ortogonais a $|s\rangle$, e, para isso, deve ser realizada a seguinte sequência:

1. Aplicação de portas X em todos canais para *bit-flip* dos estados;

$$|00\cdots 0\rangle \rightarrow |11\cdots 1\rangle$$

2. Aplicação de porta *MCZ*, com alvo no último canal, pois dessa forma o último estado recebe uma inversão de fase;

$$MCZ_{[N\times N]} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{bmatrix}_{[N\times N]} \tag{3.11}$$

3. Novamente aplica-se portas X para desfazer os *bit-flips*.

$$|11\cdots 1\rangle \rightarrow |00\cdots 0\rangle$$

Após a última etapa, o resultado obtido é:

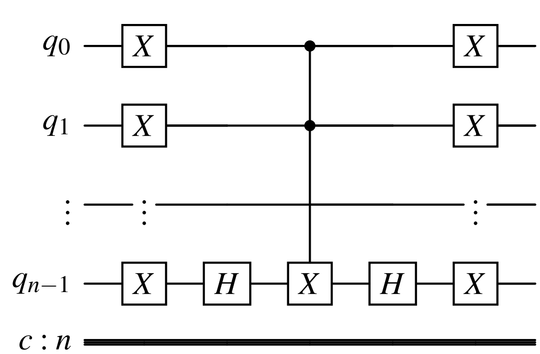
$$\begin{bmatrix} -1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{[N\times N]}$$

Ou seja, existe ainda uma fase global em U_0, de modo que o Operador de Difusão, em termos de portas lógicas, é:

$$U_0 = -X^{\otimes n}\, MCZ\, X^{\otimes n} \tag{3.12}$$

O trecho de circuito descrito em Equação está mostrado na Figura 3.6

Figura 3.6 – Operador de Difusão parcial



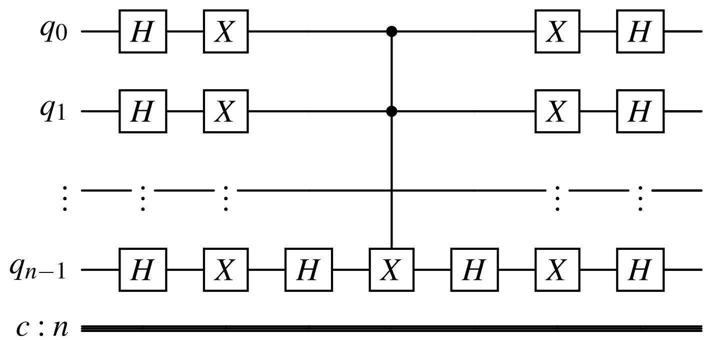
Fonte: do autor

Considerando as Equações 3.10 e 3.12. Tem-se que o Operador de Difusão completo em portas lógicas quânticas são dados pela Equação 3.13 e pela Figura 3.7.

$$U_s = H^{\otimes n}\, X^{\otimes n}\, MCZ\, X^{\otimes n}\, H^{\otimes n} \tag{3.13}$$

Note que a fase negativa não foi acrescentada, pois não precisa ser considerada no circuito quântico.

Figura 3.7 – Operador de Difusão completo



Fonte: do autor

3.2.5 Fator de Otimizacão (k)

Afim de otimizar o resultado, a aplicação do Oráculo (U_f) e do Operador de Difusão (U_s) deverão ser executados um número k de vezes para que o resultado obtido seja o mais próximo possível (com maiores amplitudes e probabilidades) do estado esperado $|\omega\rangle$, e pode ser expresso por

$$|\psi_k\rangle = (U_s U_f)^k\, |s\rangle \tag{3.14}$$

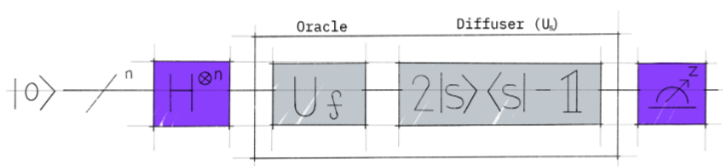
O número ideal k de iterações necessárias para que $|\omega\rangle$ seja obtido é dado por

$$k = \frac{\pi}{4}\sqrt{\frac{N}{m}} \tag{3.15}$$

em que N é o tamanho do espaço de pesquisa, e m é a quantidade de itens procurados.

A Figura 3.8 mostra um esquema do Algoritmo de Grover completo, evidenciando o Oráculo e o Operador de Difusão.

Figura 3.8 – Esquematização do Algoritmo de Grover.



Fonte: Github/*Qiskit*.

4 PROCEDIMENTOS TEÓRICOS E COMPUTACIONAIS

Este capítulo descreve detalhadamente os procedimentos teóricos e computacionais adotados para a implementação do Algoritmo de Grover. Inicialmente, apresenta-se o procedimento teórico formalizado matematicamente e expresso por meio de circuitos quânticos compostos por portas lógicas elementares.

Posteriormente, será apresentado o procedimento computacional, que envolve a modelagem e a execução do algoritmo utilizando ferramentas de simulação e programação quântica. Esta parte tem como objetivo validar teoricamente o funcionamento do algoritmo e analisar, por meio de experimentos práticos, o seu desempenho em ambientes computacionais quânticos. Assim, este capítulo estabelece as bases necessárias para a compreensão e replicação dos resultados obtidos ao longo deste trabalho.

4.1 Criando o Circuito Quântico Virtual

Nesta seção, serão apresentados de forma integrada os aspectos teóricos e computacionais da implementação do Algoritmo de Grover com 4 qubits e que orientaram a construção do circuito quântico utilizado no processo. Serão apresentados, também, trechos de código escritos em *Python* utilizando a biblioteca *Qiskit*. A combinação entre fundamentação teórica e simulação computacional permite visualizar com clareza o funcionamento do algoritmo, cujo procedimento foi dividido em três etapas principais:

1. a preparação do espaço de pesquisa;

2. a aplicação do oráculo;

3. a operação de difusão.

Cada uma dessas etapas é descrita de forma detalhada em uma estrutura que visa garantir clareza na compreensão do funcionamento do circuito e posterior análise dos resultados.

4.1.1 Preparação Inicial do Circuito Quântico

A princípio, foi necessário determinar o Espaço de Pesquisa $|s\rangle$, *i. e.*, criar uma superposição uniforme dos estados da base de 4 qubits. Como sugere a Equação **??**, para $n = 4$ qubits, o Espaço de Pesquisa possui $N = 2^4 = 16$ itens, e é dado por:

$$\begin{aligned} |s\rangle &= \frac{1}{4}\sum_{x=0}^{15} |bin(x)\rangle \\ &= \frac{1}{4}(|0000\rangle + |0001\rangle + |0010\rangle + |0011\rangle + |0100\rangle + |0101\rangle + |0110\rangle + |0111\rangle + \\ &\qquad |1000\rangle + |1001\rangle + |1010\rangle + |1011\rangle + |1100\rangle + |1101\rangle + |1110\rangle + |1111\rangle) \end{aligned} \tag{4.1}$$

Para a implementação do circuito quântico virtual em *Qiskit*, deve-se primeiramente importar as dependências e instanciar as variáveis do circuito. As dependências relacionadas à biblioteca *Qiskit* são apresentadas na Figura 4.1 e a importação e declaração dos recursos na Figura 4.2.

Figura 4.1 – Dependências do Programa

```
{
  "QiskitDependencies": {
    "qiskit": "2.1.2",
    "qiskit-aer": "0.17.1",
    "qiskit-ibm-provider": "0.11.0",
    "qiskit-ibm-runtime": "0.41.1"
  }
}
```

Fonte: do autor

A Figura 4.2 mostra `QC_Grover`, que é um objeto instanciado pela classe `QuantumCircuit`. Nela será adicionado todo o circuito. Por sua vez, `qubits` e `bits` são instâncias das classes `QuantumRegister` e `ClassicalRegister`, respectivamente, que serão responsáveis pelo armazenamento dos dados e importantes no processo de medição.

A próxima etapa é a formação do espaço de pesquisa mencionado anteriormente e, para isso, pode-se recorrer à Equação **??**, que implica no uso de portas *Hadamard* em cada *qubit*. Esse passo é feito usando o módulo `inicializa_s`, apresentado na Figura 4.3.

Figura 4.2 – Inicialização do Circuito Quântico

```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister

n = <num_qubits>
qubits = QuantumRegister(n, 'qubit')
bits= ClassicalRegister(n, 'bit')

QC_Grover = QuantumCircuit(qubits, bits)
```

Fonte: do autor

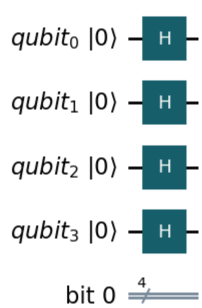
Figura 4.3 – Módulo Preparação Inicial

```
def inicializa_s(qc, qubits):
    for qubit in qubits:
        qc.h(qubit)
    return qc
```

Fonte: do autor

Uma representação visual do circuito quântico virtual relativo a este trecho pode ser acompanhada na Figura 4.4. É relevante notar que todos os *qubits* são inicializados no estado $|0\rangle$.

Figura 4.4 – Circuito Quântico Virtual - Superposição



Fonte: do autor

4.1.2 Oráculo

Para dar prosseguimento, é preciso marcar o estado que se deseja obter, *i. e.*, aplicar o **??** (Equação **??**) ao Espaço de Pesquisa (Equação 4.1). Seja $\omega = |1111\rangle$ o estado arbitrariamente escolhido para ser buscado, a aplicação do Oráculo é tal que

$$
\begin{aligned}
U_f\,|s\rangle = \frac{1}{4}\,U_f\,[&|0000\rangle+|0001\rangle+|0010\rangle+|0011\rangle+|0100\rangle+|0101\rangle+|0110\rangle+|0111\rangle+\\
&|1000\rangle+|1001\rangle+|1010\rangle+|1011\rangle+|1100\rangle+|1101\rangle+|1110\rangle+|1111\rangle]\\
=\frac{1}{4}(&|0000\rangle+|0001\rangle+|0010\rangle+|0011\rangle+|0100\rangle+|0101\rangle+|0110\rangle+|0111\rangle+\\
&|1000\rangle+|1001\rangle+|1010\rangle+|1011\rangle+|1100\rangle+|1101\rangle+|1110\rangle+-|1111\rangle)
\end{aligned} \tag{4.2}
$$

Que pode ser expresso de acordo com a matriz 16×16

$$
(U_f\;\;|s\rangle)_{[16\times16]} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{bmatrix}_{[16\times16]}
$$

O desafio seguinte foi determinar a(s) porta(s) lógica(s) que retorna(m) essa matriz. Para isso, pode-se seguir a regra de formação enunciada na Seção **??** do Capítulo Ferramentas e Métodos. Como o estado buscado é $\omega = |1111\rangle$, não será necessário o uso de portas X, apenas uma MCZ (Equação 3.11) será suficiente, pois o resultado prático que se obtém com a aplicação da porta MCZ é basicamente mudar a fase do último elemento da matriz $N\times N$ à qual ela for aplicada, exatamente o que se intenta obter; portanto, o Oráculo será uma porta MCZ.

Em *Qiskit*, a implementação é feita com o módulo `oraculo_Uw`, mostrado na Figura 4.5. E o circuito quântico virtual relativo a este trecho é mostrado na Figura 4.6[1].

O módulo `mcz_circ` apenas é usado para agrupar o conjunto de portas $H\ MCX\ H$ em um único bloco, para facilitar identificação. Para fins de ilustração, vide Figura 4.6. O parâmetro `winner` representa o estado marcado a ser identificado (por exemplo, $|1111\rangle$).

[1] Como o *Qiskit* não oferece suporte nativo à porta MCZ, ela é construída combinando portas $Hadamard$ e MCX, visto que $MCZ = H\ MCX\ H$ (H's apenas no qubit alvo).

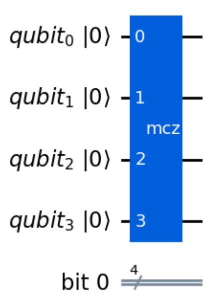
[TALVEZ DEMONSTRAR NO APÊNDICE***]

Figura 4.5 – Módulo Oráculo

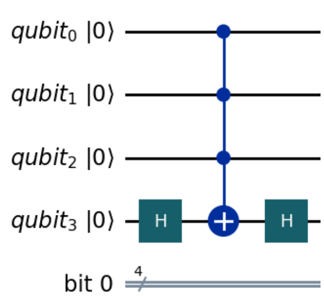
```
def mcz_circ(qc):
    qc.h(3)
    qc.mcx([0, 1, 2], 3)
    qc.h(3)
    return qc

def oraculo_Uw(qc, qubits, winner, QuantumCircuit):
    for i, index in enumerate(winner):
        if index == "0":
            qc.x(i)

    # Para agrupar as portas H MCX H em uma MCZ
    mcz_gate = mcz_circ(qc = QuantumCircuit(4,name="mcz")).to_instruction()
    qc.append(mcz_gate,[i for i in range(4)])

    for i, index in enumerate(winner):
        if index == "0":
            qc.x(i)
```

Fonte: do autor

Figura 4.6 – Circuito Quântico Virtual - Oráculo



(a) Oráculo contraído.



(b) Oráculo expandido.

Fonte: do autor.

4.1.3 Difusão

A etapa seguinte é a de amplificação do estado buscado, aplicando uma segunda reflexão U_s (Equação 3.6) na resultante da etapa anterior. Conforme demonstrado na Seção 3.2.4, Capítulo Ferramentas e Métodos, a operação que se deseja realizar é dada pela Equação 3.9. Essa operação aplica uma fase negativa a todos os estados ortonormais a $|0\rangle$, *i. e.*, a expansão da Equação 3.9 para quatro qubits resulta em:

$$U_0\ |s\rangle = |0000\rangle - |0001\rangle - |0010\rangle - |0011\rangle - \cdots - |1111\rangle \tag{4.3}$$

Ou seja, a amplitude $|0000\rangle$ aumenta em relação aos demais que invertem.

O difusor é representado pela matriz:

$$U_0 = -\begin{pmatrix} -1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{4.4}$$

Para colocar isso em termos de portas lógicas, basta utilizar o conjunto de portas apresentado pela Equação 3.13. Em *Qiskit*, a implementação pode ser feita com o uso do módulo `difusor_Us`, visto na Figura 4.7.

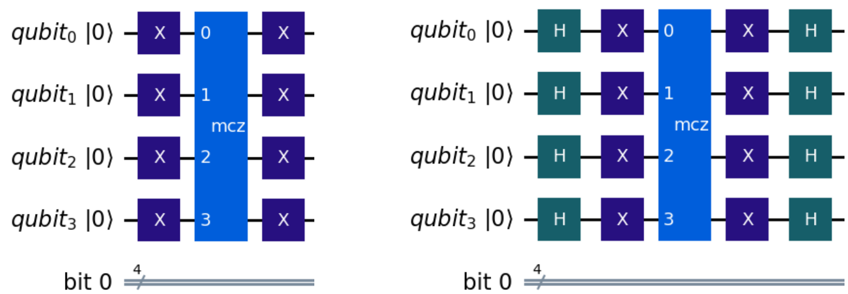
Figura 4.7 – Módulo Operador de Difusão

```
def difusor_Us(qc, qubits):
    for i in qubits:
        qc.h(i)
        qc.x(i)

    # Para agrupar as portas H MCX H em uma MCZ
    mcz_gate = mcz_circ(qc = QuantumCircuit(4,name="mcz")).to_instruction()
    qc.append(mcz_gate,[i for i in range(4)])

    for i in qubits:
        qc.x(i)
        qc.h(i)
    return qc
```

Fonte: do autor

A idealização virtual desse trecho pode ser observada na Figura 4.8. Nela, é possível fazer a equivalência entre U_0 (Equação 3.12) e a Figura 4.8a e entre U_s (Equação 3.13) e a Figura 4.8b.

Figura 4.8 – Circuito Quântico Virtual - Amplificação, U_0 e U_s.



(a) Circuito de U_0 (b) Circuito de U_s

Fonte: do autor

4.1.4 Fator k

Para que a Equação 3.14 seja de fato satisfeita, deve-se determinar o fator de otimização, *i. e.*, a quantidade de vezes que o Oráculo (U_f) e o Operador de Difusão (U_s) devem ser repetidos. O valor de k pode ser obtido com a utilização da Equação 3.15, em que tem-se $N = 16$ e $m = 1$.

$$
\begin{aligned}
k &= \frac{\pi}{4}\sqrt{\frac{N}{m}} \\
&= \frac{\pi}{4}\sqrt{16} \\
&= \pi
\end{aligned}
$$

E como k não pode ser número de ponto flutuante, deve-se arredondar para o inteiro mais próximo, ou seja, para fins práticos,

$$k = 3 \tag{4.5}$$

. Nesse ponto, tem-se que todas as etapas estão concluídas, cabendo, então, uní-las em um único circuito quântico. Em *Qiskit*, isso pode ser feito com o trecho de código mostrado na Figura 4.9, que leva em consideração o valor de k, por meio de um *loop* `for` no intervalo dado por k. Ao final, aplica-se o método `measure()` ao circuito, que realiza a medição dos *qubits*, colapsando o sistema em um dos estados da base computacional, revelando o resultado da busca.

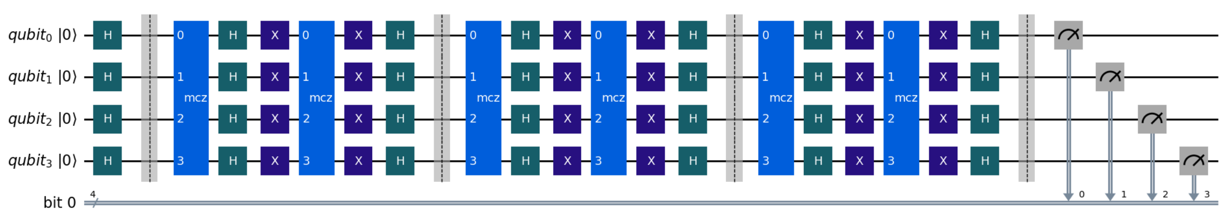
Figura 4.9 – Trecho para formação do Circuito Quântico Virtual.

```
N = 2**n
k = (round((pi/4)*sqrt(N)))
marked_state = "<seu_estado>"
inicializa_s(QC_Grover, range(n))

for _ in range(k):
    QC_Grover.barrier([i for i in range(n)])
    oraculo_Uw(QC_Grover, range(n), marked_state, QuantumCircuit)
    difusor_Us(QC_Grover, (range(n)))

QC_Grover.barrier([i for i in range(n)])
QC_Grover.measure([0,1,2,3],[0,1,2,3])
```

Fonte: do autor

O circuito relativo ao trecho em questão pode ser visto na Figura 4.10. As barreiras (`barrier()`), nesse caso, são apenas separadores visuais do processo de iteração, e os objetos ao final de cada canal indicam o processo de medição do estado final sendo armazenado nos canais clássicos.

Figura 4.10 – Circuito Quântico Virtual - Algoritmo de Grover Completo.



Fonte: do autor

4.2 Plataforma *IBM Quantum*

Nesta seção, descreve-se e elucida-se por meio de trechos de códigos exemplos o procedimento adotado para o acesso e preparação da simulação do circuito do Algoritmo de Grover utilizando os recursos computacionais da *IBM Quantum Platform* (**??**). Essa plataforma online desenvolvida pela IBM permite a execução de algoritmos em computadores quânticos reais e simuladores avançados disponíveis na nuvem.

O primeiro passo consiste na criação de uma conta na plataforma e posterior autenticação com uma organização habilitada. Dentro do ambiente web, o usuário pode configurar projetos e acessar diferentes dispositivos quânticos, incluindo simuladores ideais (sem ruído), simuladores com ruído e *hardwares* reais. Cada projeto está vinculado a uma *instância*, que define os recursos disponíveis, como prioridade de fila e acesso a dispositivos específicos.

Diferentemente dos ambientes locais de simulação, como os disponíveis via `AerSimulator` no *Qiskit*, a *IBM Quantum* exige que até mesmo as simulações ideais sejam submetidas remotamente, exigindo autenticação prévia e conexão à nuvem. Com a migração da plataforma para o novo modelo unificado, o canal de acesso deve ser definido como `ibm_quantum_platform`, e é obrigatória a definição explícita da *instância*.

4.2.1 Procedimento de Autenticação

A seguir, a Figura 4.11 apresenta um exemplo do código utilizado para salvar e autenticar uma conta na plataforma, utilizando a biblioteca `qiskit_ibm_runtime`:

Figura 4.11 – Trecho de acesso à *IBM Quantum Platform*.

```
from qiskit_ibm_runtime import QiskitRuntimeService

QiskitRuntimeService.save_account(
    token="<seu_token>",
    instance="<sua_instancia>",
    channel="ibm_quantum_platform",
    overwrite=True,
    set_as_default=True
)
service = QiskitRuntimeService()
```

Fonte: do autor

O `token` pode ser obtido no perfil do usuário na plataforma, enquanto a `instance` segue o padrão `<hub>/<group>/<project>`, que pode ser consultado no painel de gerenciamento de projetos da *IBM Quantum Platform*. A chamada `set_as_default=True` permite que o serviço seja usado diretamente nos próximos acessos, sem necessidade de repetir a autenticação.

Após essa configuração inicial, torna-se possível listar os dispositivos disponíveis, verificar seus estados (como filas, fidelidades e número de qubits), selecionar o *backend* desejado e submeter os circuitos para execução.

Fluxo de Execução na Nuvem

O processo completo de execução de um circuito na plataforma envolve:

- Construção do circuito com bibliotecas do `Qiskit`;
- Escolha do *backend* apropriado (simulador ou QPU);
- Submissão da tarefa à nuvem via `QiskitRuntimeService`;
- Acompanhamento da execução e análise dos resultados.

4.2.2 Simulação Ideal via `Statevector`

Essa seção traz um exemplo de código que gera uma distribuição de probabilidades ideal, utilizando a classe `Statevector` do `Qiskit`, apresentada na Figura 4.12, que simula a evolução do estado quântico de forma exata e livre de ruído. Essa abordagem fornece a expectativa teórica para o comportamento do algoritmo de Grover.

Figura 4.12 – Trecho para Simulação Ideal.

```
from qiskit.quantum_info import Statevector
ideal_distribution = Statevector.
                     from_instruction(QC_Grover).
                     probabilities_dict()
fig, ax = plt.subplots()
plot_histogram(ideal_distribution,
               title="Distribuição Ideal de Probabilidades",
               ax=ax)
ax.set_ylabel("Probabilidade")
plt.show()
```

Fonte: do autor

Este resultado serve como referência teórica da máxima fidelidade, permitindo a comparação com as demais execuções e a análise da influência dos ruídos nos sistemas quânticos reais.

4.2.3 Simulação com Ruído via *AerSimulator*

Essa seção apresenta uma simulação mais próxima da realidade, pois faz uso de informações de ruído da QPU escolhida (definindo <nome_backend>). Para isso, utiliza a classe `AerSimulator`, da biblioteca `qiskit_aer`. O código exemplo é mostrado na Figura 4.13. Essa é uma forma de criar uma simulação que busca imitar de forma mais precisa o comportamento prático do circuito, uma vez que a simulação com ruído permite observar a degradação de fidelidade causada por imperfeições reais nos dispositivos quânticos.

Figura 4.13 – Trecho para Simulação com Ruído.

```
from qiskit_aer import AerSimulator
from qiskit.compiler import transpile

service = QiskitRuntimeService()
backend = service.backend('<nome_backend>')

# Gerar simulador com modelo de ruído real
backend_sim = AerSimulator.from_backend(backend)
transpiled_circ_sim = transpile(QC_Grover, backend_sim)
result = backend_sim.run(transpiled_circ_sim, shots=<num_shots>).result()
circuits.append(transpiled_circ_sim)
```

Fonte: do autor

Essa abordagem é valiosa para testar e validar circuitos antes da submissão ao *hardware* físico, permitindo ajustes e observação de efeitos de ruído específicos, como erros de porta, leitura e decoerência.

4.2.4 Execução em via **`Sampler`**

Por fim, essa última seção traz os procedimentos a serem realizados para a execução em uma QPU. Antes de o circuito ser enviado a alguma QPU propriamente dita, ele precisa ser reescrito em uma linguagem que ela entenda, *i. e.*, o circuito virtual precisa ser reescrito em termos das portas base do *backend* no qual se pretende enviá-lo. Essa é a etapa de compilação mencionada na Seção 3.1.2 e sua implementação é feita com a função `transpile` conforme mostrado na Figura 4.14.

Figura 4.14 – Trecho para Compilação.

```
from qiskit.compiler import transpile

backend_HW = service.backend("<nome_backend>")
transpiled = transpile(QC_Grover, backend_HW, optimization_level=<optim_lvl>)
```

Fonte: do autor

A definição de <`nome_backend`> é o que garante a compilação ideal de acordo com as portas base de cada *backend*, visto que cada tipo de processador pode possuir portas base diferentes. Pode-se também configurar o nível de otimização, explanado na Seção 2.1.1, por meio de `optimization_level`. Concluída essa etapa, segue-se com o circuito compilado.

A Figura 4.15 mostra um trecho de exemplo para execução em QPU's que possibilita também utilizar as técnicas de supressão – *Dynamical Decoupling* (DD) e *Pauli Twirling* introduzidas na Seção 2.1.1 – disponíveis na classe `Sampler` da biblioteca `qiskit_ibm_runtime`.

Figura 4.15 – Trecho para Execução em QPU via `Sampler`.

```
from qiskit_ibm_runtime import SamplerV2 as Sampler, Batch

def rodar():
    num_shots = <num_shots>
    with Batch(backend=backend_HW):
        sampler = Sampler()
        # Execução com Dynamical Decoupling + Pauli Twirling
        sampler.options.dynamical_decoupling.enable = True
        sampler.options.twirling.enable_gates = True
        job_DD_Twiling = sampler.run([transpiled], shots=num_shots)
try:
    rodar()
except Exception as e:
    print(f"Erro: {e}")
```

Fonte: do autor

A Figura 4.16, por sua vez, mostra um trecho de exemplo para execução em QPU's que possibilita utilizar as técnicas de mitigação – *Twirled Readout Error eXtinction* (TREX), *Zero-Noise Extrapolation* (ZNE), *Probabilistic Error Amplification* (PEA), e *Probabilistic Error Cancellation* (PEA) introduzidas na Seção 2.1.2 – disponíveis na classe `Estimate` da biblioteca `qiskit_ibm_runtime`.

Figura 4.16 – Trecho para Execução em QPU via `Estimate`.

```
COLOCAR CODIGO COM ESTIMATE***
```

Fonte: do autor

A execução real permite não apenas verificar a eficiência do algoritmo em ambiente prático, mas também comparar os efeitos das técnicas de supressão aplicadas, ajudando a entender o comportamento dos sistemas quânticos sob influência de ruído quântico.

5 RESULTADOS E ANÁLISE

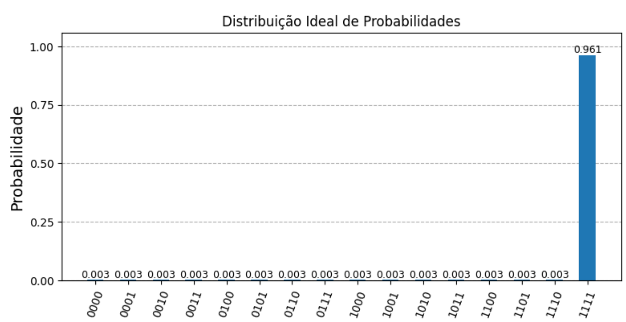
Este capítulo apresentará os resultados obtidos com a execução do Algoritmo de Grover em diferentes contextos computacionais: simulação ideal, simulação com ruído e execução real em hardware quântico. Os resultados são apresentados na forma de gráficos de distribuição de probabilidades dos estados finais, permitindo comparações qualitativas e quantitativas entre os cenários.

5.1 Resultado Ideal (Teórico)

Nesta seção, são apresentados os resultados obtidos a partir da simulação teórica ideal do circuito do Algoritmo de Grover, tal qual exemplificado na Figura 4.12 da Seção 4.2.2. Esta simulação permite visualizar a distribuição de probabilidades do estado final de maneira exata e sem influência de ruídos externos, funcionando como uma referência teórica para as demais simulações.

A Figura 5.1 mostra o histograma gerado após a aplicação do algoritmo, considerando um circuito com 4 qubits e o estado marcado $\omega = |1111\rangle$. Observa-se que o algoritmo amplifica significativamente a probabilidade de ocorrência do estado marcado, como era esperado teoricamente.

Figura 5.1 – Distribuição Ideal de Probabilidades (Simulação via `Statevector`).



Fonte: do autor

Com o valor ideal de iterações $k = 3$ (resultado 4.5), a probabilidade de medir o estado marcado $|1111\rangle$ foi de $96,1\%$, enquanto os demais estados apresentam probabilidades próximas de zero. Este comportamento é compatível com a expectativa teórica do algoritmo de Grover, conforme discutido nas seções anteriores e demonstrado também no Apêndice **??**.

A distribuição obtida reflete, portanto, a eficiência máxima do algoritmo quando executado em condições ideais. Esta simulação serve como base de comparação para as análises subsequentes, em que serão incluídos fatores realistas como ruído, erro de leitura e imperfeições físicas presentes nos dispositivos quânticos atuais.

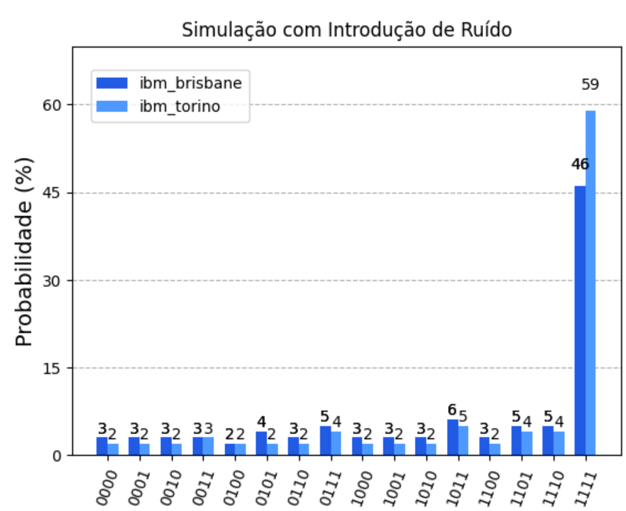
5.2 Resultado com Ruído (`AerSimulator`)

A simulação ideal apresentada na seção anterior assume um sistema quântico livre de ruídos. No entanto, implementações reais são afetadas por diversos tipos de erros, como decoerência, erro de porta, erro de leitura e ruído térmico. Para aproximar a execução do algoritmo à realidade dos dispositivos quânticos atuais, é necessário considerar esses fatores na simulação.

Esta seção tem por premissa a apresentação dos resultados da simulação feita usando dados de ruídos de dois *backends* diferentes, já mencionados no Quadro 3.1. Os modelos de ruídos são fornecidos pela classe `AerSimulator`, e podem ser utilizados como exemplificado na Figura 4.13 da Seção 4.2.3. Este modelo inclui características específicas e atuais do dispositivo real, como fidelidade de portas, taxas de erro de leitura e tempos de decoerência dos qubits, e por isso se faz uma ferramenta valiosa.

A Figura 5.2 mostra o histograma da probabilidade final, em termos de porcentagem, de cada estado.

Figura 5.2 – Distribuição de Probabilidades (Simulação via `AerSimulator`).



Fonte: do autor

Apesar da presença de ruído, observa-se que o estado marcado $|1111\rangle$ ainda possui a maior probabilidade entre os estados possíveis para ambos os modelos de ruído utilizados, com 46% e 59% para *ibm_brisbane* e *ibm_torino*, respectivamente. No entanto, pode-se notar facilmente que a presença do ruído, mesmo em ambiente simulado, já fornece um resultado bastante distorcido com relação à expectativa teórica (ou ideal), na qual a probabilidade para o estado marcado era superior a 96%. Ademais, os outros estados apresentam valores não desprezíveis, evidenciando a dispersão causada pelos erros quânticos.

5.3 Resultado Real em *Hardware* Quântico (`Sampler`)

Após a validação teórica e a simulação com ruído, o Algoritmo de Grover foi executado em dois computadores quânticos reais da *IBM Quantum*: *ibm_brisbane* e *ibm_torino*, introduzidos na Seção 3.1.2. A escolha de dois dispositivos teve como objetivo comparar o desempenho do mesmo circuito em arquiteturas físicas distintas, permitindo observar variações de fidelidade e impacto de parâmetros específicos de cada backend.

Para cada execução, foram configurados 20000 *shots*, e algumas condições específicas, conforme descritas abaixo:

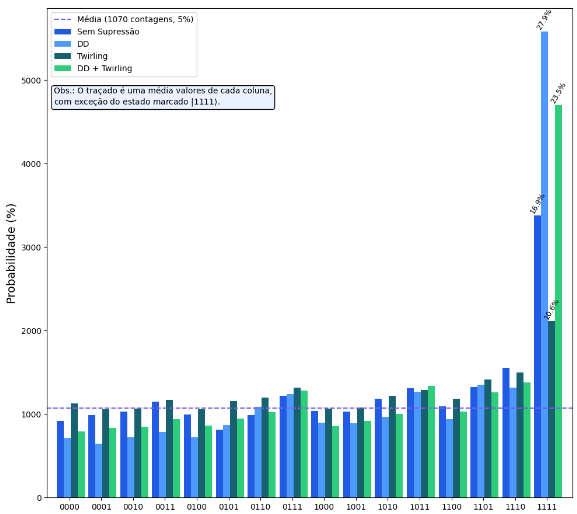
1. Sem nenhum método de supressão.
2. *Dynamical Decoupling* habilitado e *Pauli Twirling* desabilitado.
3. *Dynamical Decoupling* desabilitado e *Pauli Twirling* habilitado.
4. Ambos habilitados.

Esse mesmo padrão foi aplicado com dois níveis diferentes de otimização (2 e 3). Todas essas diferentes preparações foram com o intuito de estudo e avaliação dos efeitos de cada método, inclusive quando aplicados simultaneamente.

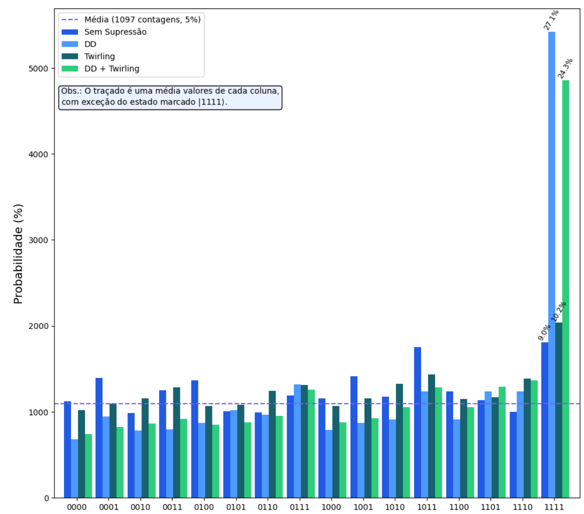
A apresentação dos resultados é feita baseada nessas configurações, em que pode-se analisar em cada gráfico as diferenças causadas pelos métodos de supressão para um mesmo nível de otimização. Dito isso, a Figura 5.3 mostra as quatro configurações de supressão aplicadas aos níveis de otimização 2 e 3 (Figuras 5.3a e 5.3b), enquanto a Figura 5.4 replica as mesmas quatro configurações, também com os dois níveis de otimização aplicados a cada histograma (Figuras 5.4a e 5.4b).

Figura 5.3 – Resultados de *ibm_brisbane*.

(a) Histograma de *ibm_brisbane* com $optim\_lvl = 2$



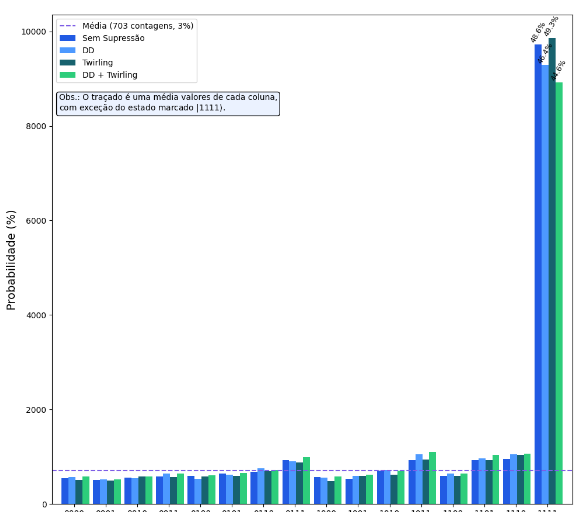
(b) Histograma de *ibm_brisbane* com $optim\_lvl = 3$



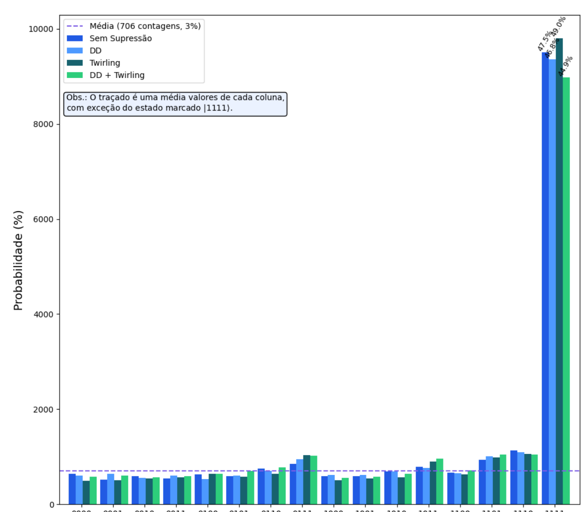
Fonte: do autor

Figura 5.4 – Resultados de *ibm_torino*

(a) Histograma de *ibm_torino* com $optim\_lvl = 2$



(b) Histograma de *ibm_torino* com $optim\_lvl = 3$



Fonte: do autor

6 CONCLUSÃO

APÊNDICE A – Demonstração do Circuito

Com a finalidade de entender o que acontece em cada parte do circuito + será feita a demonstração passo a passo das operações a cada camada.

Seja $|\psi\rangle$ o estado inicial do espaço formado pelos 4 qubits + com valor inicial dado por $|\psi\rangle = |0000\rangle$. Aplicaremos portas *Hadamard* em cada qubit como parte da primeira etapa do Algoritmo de Grover: a preparação inicial. Essa operação resulta em:

1. Preparação inicial: aplicação das portas H's (Figura 1):

$$H^{\otimes 4}|\psi\rangle = |\psi_1\rangle = \frac{1}{4}\Big(|0000\rangle + |0001\rangle + |0010\rangle + |0011\rangle + |0100\rangle + |0101\rangle + |0110\rangle + |0111\rangle + |1000\rangle + |1001\rangle + |1010\rangle + |1011\rangle + |1100\rangle + |1101\rangle + |1110\rangle + |1111\rangle\Big)$$

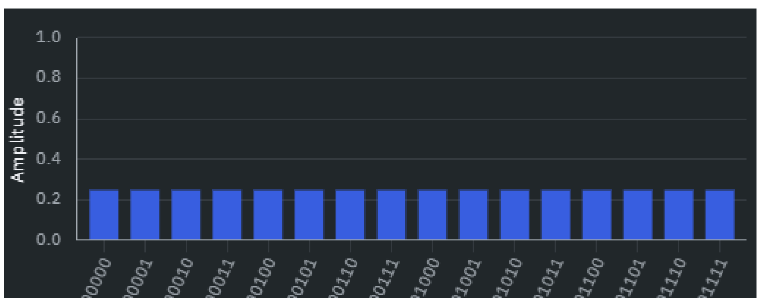


Figura 1 – Aplicação das portas H's na preparação inicial.
Fonte: IBM-*Quantum Learning / Composer*.

2. Aplicar o oráculo U_f que marca o estado $|1111\rangle$ invertendo seu sinal (Figura 2):

$$U_f|\psi_1\rangle = |\psi_2\rangle = \frac{1}{4}\Big(|0000\rangle + |0001\rangle + |0010\rangle + |0011\rangle + |0100\rangle + |0101\rangle + |0110\rangle + |0111\rangle + |1000\rangle + |1001\rangle + |1010\rangle + |1011\rangle + |1100\rangle + |1101\rangle + |1110\rangle - |1111\rangle\Big)$$

3. Início da operação de difusão: aplicar portas *Hadamard* (Figura 3):

$$H^{\otimes 4}|\psi_2\rangle = |\psi_3\rangle = \frac{1}{8}\Big(7|0000\rangle + |0001\rangle + |0010\rangle - |0011\rangle + |0100\rangle - |0101\rangle - |0110\rangle + |0111\rangle + |1000\rangle - |1001\rangle - |1010\rangle + |1011\rangle - |1100\rangle + |1101\rangle + |1110\rangle - |1111\rangle\Big)$$

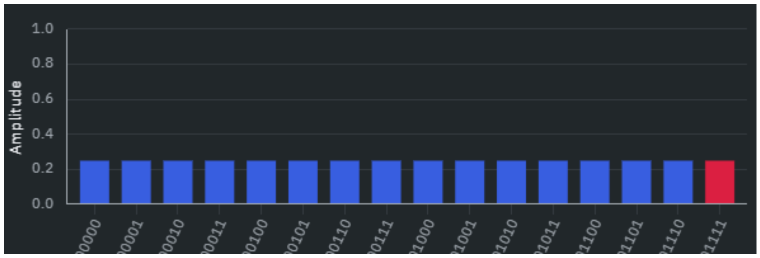


Figura 2 – Aplicação do oráculo U_f, que inverte o sinal do estado $|1111\rangle$.
Fonte: IBM-*Quantum Learning / Composer*.

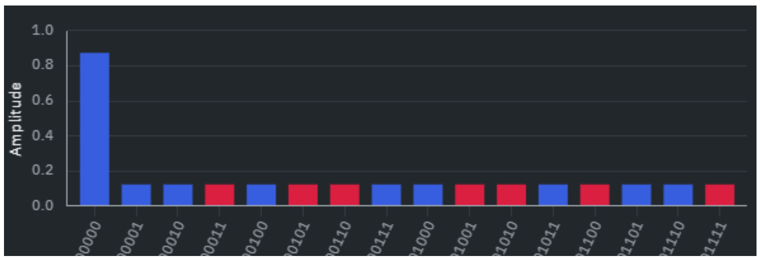


Figura 3 – Aplicação das portas *Hadamard* no início da operação de difusão.
Fonte: IBM-*Quantum Learning / Composer*.

4. Aplicar portas X (Figura 4):

$$
\begin{aligned}
X^{\otimes 4}|\psi_3\rangle = |\psi_4\rangle = \frac{1}{8}\Big(& 7|1111\rangle + |1110\rangle + |1101\rangle - |1100\rangle + |1011\rangle - |1010\rangle - |1001\rangle + |1000\rangle \\
& + |0111\rangle - |0110\rangle - |0101\rangle + |0100\rangle - |0011\rangle + |0010\rangle + |0001\rangle - |0000\rangle\Big) \\
= \frac{1}{8}\Big(& - |0000\rangle + |0001\rangle + |0010\rangle - |0011\rangle + |0100\rangle - |0101\rangle - |0110\rangle + |0111\rangle \\
& + |1000\rangle - |1001\rangle - |1010\rangle + |1011\rangle - |1100\rangle + |1101\rangle + |1110\rangle + 7|1111\rangle\Big)
\end{aligned}
$$

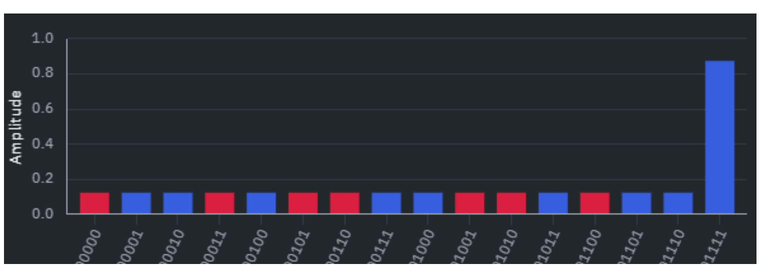


Figura 4 – Aplicação das portas X no operador de difusão.
Fonte: IBM-*Quantum Learning / Composer*.

5. Aplicar a porta *MCZ* (Figura 5):

$$MCZ\,|\psi_4\rangle = |\psi_5\rangle = \frac{1}{8}\Big(-|0000\rangle + |0001\rangle + |0010\rangle - |0011\rangle + |0100\rangle - |0101\rangle - |0110\rangle$$
$$+\,|0111\rangle + |1000\rangle - |1001\rangle - |1010\rangle + |1011\rangle - |1100\rangle + |1101\rangle + |1110\rangle - 7\,|1111\rangle\Big)$$

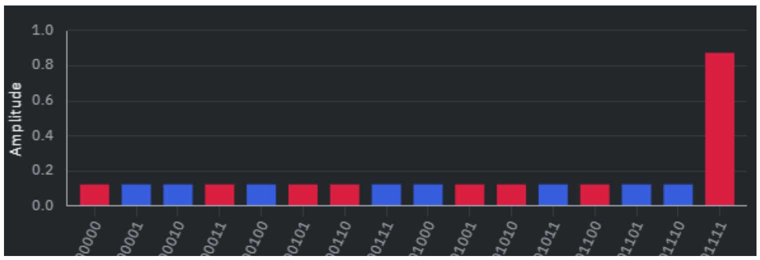


Figura 5 – Aplicação da porta *MCZ* no operador de difusão.
Fonte: IBM-*Quantum Learning / Composer*.

6. Aplicar portas *X* novamente (Figura 6):

$$X^{\otimes 4}\,|\psi_5\rangle = |\psi_6\rangle = \frac{1}{8}\Big(-|1111\rangle + |1110\rangle + |1101\rangle - |1100\rangle + |1011\rangle - |1010\rangle - |1001\rangle$$
$$+\,|1000\rangle + |0111\rangle - |0110\rangle - |0101\rangle + |0100\rangle - |0011\rangle + |0010\rangle + |0001\rangle - 7\,|0000\rangle\Big)$$
$$= \frac{1}{8}\Big(-7\,|0000\rangle + |0001\rangle + |0010\rangle - |0011\rangle + |0100\rangle - |0101\rangle - |0110\rangle + |0111\rangle$$
$$+\,|1000\rangle - |1001\rangle - |1010\rangle + |1011\rangle - |1100\rangle + |1101\rangle + |1110\rangle - |1111\rangle\Big)$$

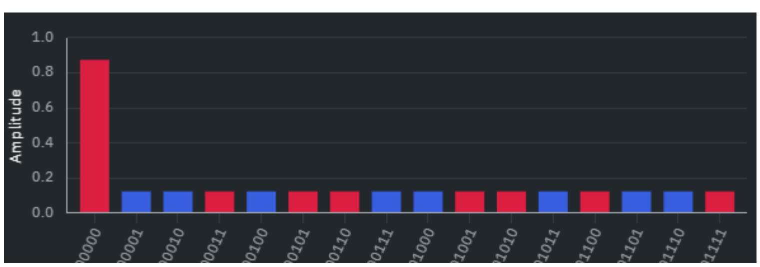


Figura 6 – Segunda aplicação das portas *X* no operador de difusão.
Fonte: IBM-*Quantum Learning / Composer*.

7. Aplicar portas *Hadamard* novamente (Figura 7):

$$H^{\otimes 4}\left|\psi_6\right\rangle = \left|\psi_7\right\rangle = \frac{1}{16}\Big(-3\left|0000\right\rangle - 3\left|0001\right\rangle - 3\left|0010\right\rangle - 3\left|0011\right\rangle - 3\left|0100\right\rangle - 3\left|0101\right\rangle$$
$$- 3\left|0110\right\rangle - 3\left|0111\right\rangle - 3\left|1000\right\rangle - 3\left|1001\right\rangle - 3\left|1010\right\rangle$$
$$- 3\left|1011\right\rangle - 3\left|1100\right\rangle - 3\left|1101\right\rangle - 3\left|1110\right\rangle - 11\left|1111\right\rangle\Big)$$
$$= -\frac{3}{16}\Big(\left|0000\right\rangle + \left|0001\right\rangle + \left|0010\right\rangle + \left|0011\right\rangle + \left|0100\right\rangle + \left|0101\right\rangle + \left|0110\right\rangle + \left|0111\right\rangle$$
$$+ \left|1000\right\rangle + \left|1001\right\rangle + \left|1010\right\rangle + \left|1011\right\rangle + \left|1100\right\rangle + \left|1101\right\rangle + \left|1110\right\rangle + \frac{11}{3}\left|1111\right\rangle\Big)$$

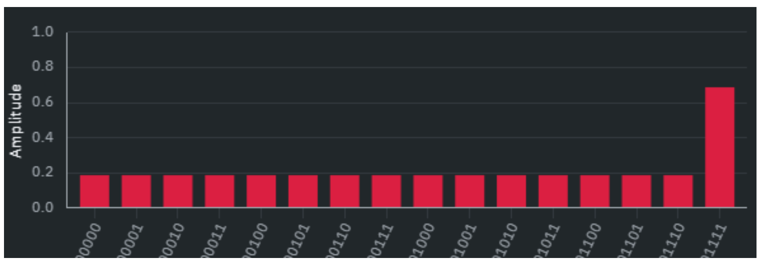


Figura 7 – Aplicação final das portas *Hadamard* na primeira iteração.
Fonte: IBM-*Quantum Learning / Composer*.

O sétimo passo finaliza a primeira iteração, mas precisamos repetir o processo mais duas vezes.

8. Marcar novamente o estado buscado pelo oráculo U_f (Figura 8):

$$U_f\left|\psi_7\right\rangle = \left|\psi_8\right\rangle = -\frac{3}{16}\Big(\left|0000\right\rangle + \left|0001\right\rangle + \left|0010\right\rangle + \left|0011\right\rangle + \left|0100\right\rangle + \left|0101\right\rangle + \left|0110\right\rangle$$
$$+ \left|0111\right\rangle + \left|1000\right\rangle + \left|1001\right\rangle + \left|1010\right\rangle + \left|1011\right\rangle + \left|1100\right\rangle + \left|1101\right\rangle + \left|1110\right\rangle - \frac{11}{3}\left|1111\right\rangle\Big)$$

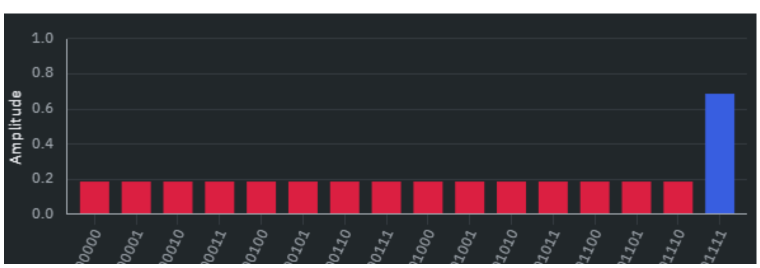


Figura 8 – Aplicação do oráculo U_f na segunda iteração.
Fonte: IBM-*Quantum Learning / Composer*.

Após o oráculo, devemos aplicar o operador de difusão.

9. Aplicar portas *Hadamard* (Figura 9):

$$H^{\otimes 4}|\psi_8\rangle = |\psi_9\rangle = \frac{7}{32}\Big(-\frac{17}{7}|0000\rangle - |0001\rangle - |0010\rangle + |0011\rangle - |0100\rangle + |0101\rangle + |0110\rangle - |0111\rangle - |1000\rangle + |1001\rangle + |1010\rangle - |1011\rangle + |1100\rangle - |1101\rangle - |1110\rangle + |1111\rangle\Big)$$

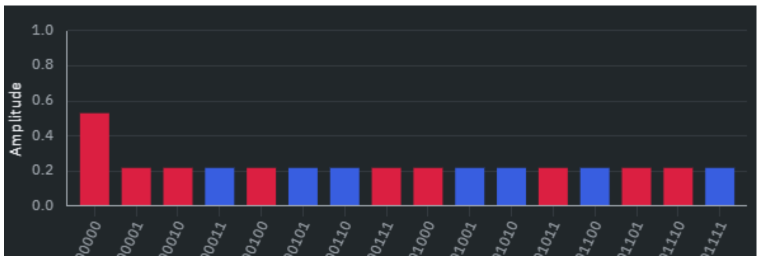


Figura 9 – Aplicação das portas *Hadamard* na operação de difusão da segunda iteração.
Fonte: IBM-*Quantum Learning / Composer*.

10. Aplicar portas X (Figura 10):

$$X^{\otimes 4}|\psi_9\rangle = |\psi_{10}\rangle = \frac{7}{32}\Big(-\frac{17}{7}|1111\rangle - |1110\rangle - |1101\rangle + |1100\rangle - |1011\rangle + |1010\rangle + |1001\rangle - |1000\rangle - |0111\rangle + |0110\rangle + |0101\rangle - |0100\rangle + |0011\rangle - |0010\rangle - |0001\rangle + |0000\rangle\Big)$$

$$= \frac{7}{32}\Big(|0000\rangle - |0001\rangle - |0010\rangle + |0011\rangle - |0100\rangle + |0101\rangle + |0110\rangle - |0111\rangle - |1000\rangle + |1001\rangle + |1010\rangle - |1011\rangle + |1100\rangle - |1101\rangle - |1110\rangle - \frac{17}{7}|1111\rangle\Big)$$

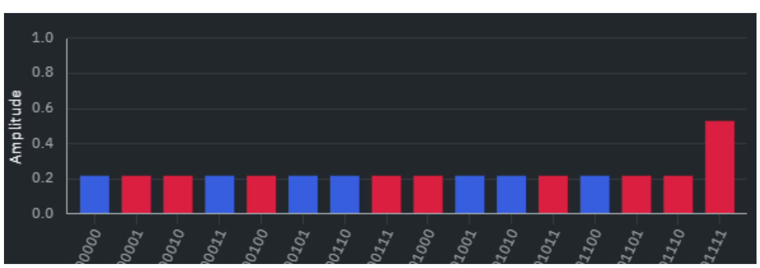


Figura 10 – Aplicação das portas X no operador de difusão da segunda iteração.
Fonte: IBM-*Quantum Learning / Composer*.

11. Aplicar porta *MCZ* (Figura 11):

$$MCZ\,|\psi_{10}\rangle = |\psi_{11}\rangle = \frac{7}{32}\Big(|0000\rangle - |0001\rangle - |0010\rangle + |0011\rangle - |0100\rangle + |0101\rangle + |0110\rangle$$
$$- |0111\rangle - |1000\rangle + |1001\rangle + |1010\rangle - |1011\rangle + |1100\rangle - |1101\rangle - |1110\rangle + \frac{17}{7}|1111\rangle\Big)$$

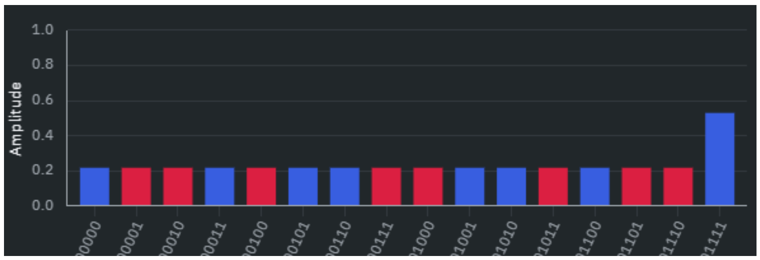


Figura 11 – Aplicação da porta *MCZ* na segunda iteração do operador de difusão.
Fonte: IBM-*Quantum Learning / Composer*.

12. Aplicar portas X novamente (Figura 12):

$$X^{\otimes 4}|\psi_{11}\rangle = |\psi_{12}\rangle = \frac{7}{32}\Big(|1111\rangle - |1110\rangle - |1101\rangle + |1100\rangle - |1011\rangle + |1010\rangle + |1001\rangle$$
$$- |1000\rangle - |0111\rangle + |0110\rangle + |0101\rangle - |0100\rangle + |0011\rangle - |0010\rangle - |0001\rangle + \frac{17}{7}|0000\rangle\Big)$$
$$= \frac{7}{32}\Big(\frac{17}{7}|0000\rangle - |0001\rangle - |0010\rangle + |0011\rangle - |0100\rangle + |0101\rangle + |0110\rangle - |0111\rangle$$
$$- |1000\rangle + |1001\rangle + |1010\rangle - |1011\rangle + |1100\rangle - |1101\rangle - |1110\rangle + |1111\rangle\Big)$$

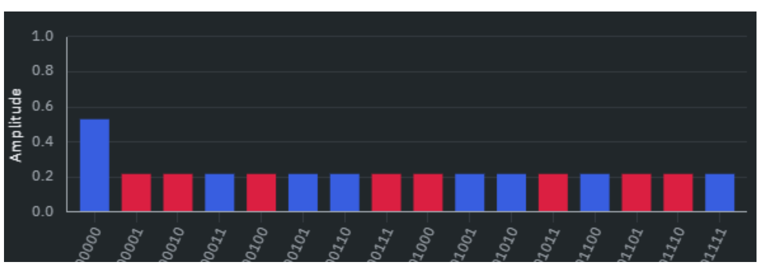


Figura 12 – Segunda aplicação das portas X na segunda iteração do operador de difusão.
Fonte: IBM-*Quantum Learning / Composer*.

13. Para finalizar a segunda iteração, aplicamos novamente as portas *Hadamard*:

$$H^{\otimes 4}\,|\psi_{12}\rangle = |\psi_{13}\rangle = \frac{5}{64}\Big(|0000\rangle + |0001\rangle + |0010\rangle + |0011\rangle + |0100\rangle + |0101\rangle + |0110\rangle$$
$$+\,|0111\rangle + |1000\rangle + |1001\rangle + |1010\rangle + |1011\rangle + |1100\rangle + |1101\rangle + |1110\rangle + \frac{61}{5}\,|1111\rangle\Big)$$

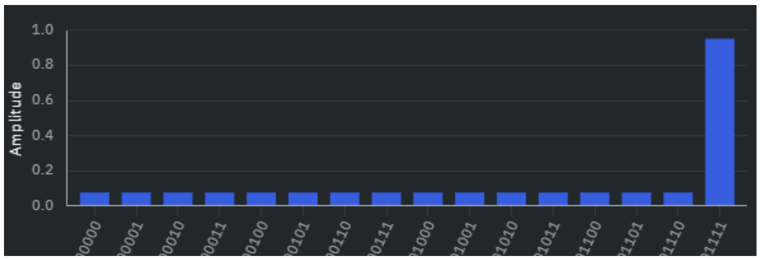


Figura 13 – Aplicação final das portas *Hadamard* na segunda iteração.
Fonte: IBM-*Quantum Learning / Composer*.

O décimo terceiro passo finaliza a segunda iteração, mas é necessário repetir o processo mais uma vez.

14. Aplicar o oráculo U_f mais uma vez (Figura 14):

$$U_f\,|\psi_{13}\rangle = |\psi_{14}\rangle = \frac{5}{64}(|0000\rangle + |0001\rangle + |0010\rangle + |0011\rangle + |0100\rangle + |0101\rangle + |0110\rangle$$
$$+\,|0111\rangle + |1000\rangle + |1001\rangle + |1010\rangle + |1011\rangle + |1100\rangle + |1101\rangle + |1110\rangle - \frac{61}{5}\,|1111\rangle))$$

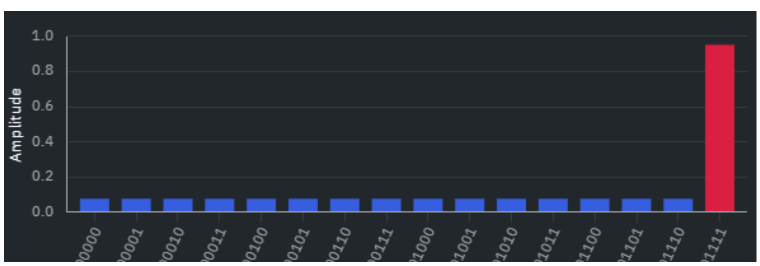


Figura 14 – Aplicação do oráculo U_f na terceira iteração.
Fonte: IBM-*Quantum Learning / Composer*.

15. Aplicar portas *Hadamard* (Figura 15):

$$H^{\otimes 4}\,|\psi_{14}\rangle = |\psi_{15}\rangle = \frac{33}{128}\Big(\frac{7}{33}\,|0000\rangle + |0001\rangle + |0010\rangle - |0011\rangle + |0100\rangle - |0101\rangle - |0110\rangle$$
$$+\,|0111\rangle + |1000\rangle - |1001\rangle - |1010\rangle + |1011\rangle - |1100\rangle + |1101\rangle + |1110\rangle - |1111\rangle\Big)$$

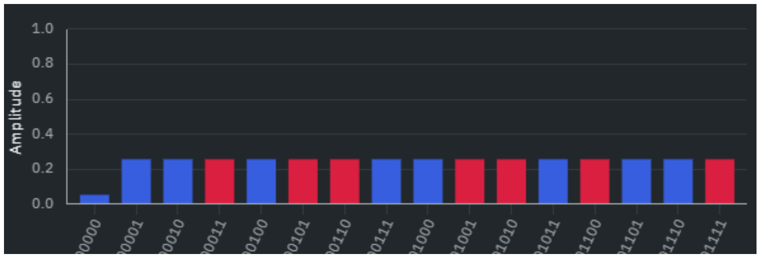


Figura 15 – Aplicação das portas *Hadamard* na operação de difusão da terceira iteração.
Fonte: IBM-*Quantum Learning / Composer*.

16. Aplicar portas X (Figura 16):

$$
\begin{aligned}
X^{\otimes 3}|\psi_{15}\rangle = |\psi_{16}\rangle = \frac{33}{128}\Big(\frac{7}{33}|1111\rangle + |1110\rangle + |1101\rangle - |1100\rangle + |1011\rangle - |1010\rangle - |1001\rangle \\
+ |1000\rangle + |0111\rangle - |0110\rangle - |0101\rangle - |0100\rangle - |0011\rangle + |0010\rangle + |0001\rangle - |0000\rangle\Big) \\
= \frac{33}{128}(-|0000\rangle + |0001\rangle + |0010\rangle - |0011\rangle + |0100\rangle - |0101\rangle - |0110\rangle + |0111\rangle \\
+ |1000\rangle - |1001\rangle - |1010\rangle + |1011\rangle - |1100\rangle + |1101\rangle + |1110\rangle + \frac{7}{33}|1111\rangle)
\end{aligned}
$$

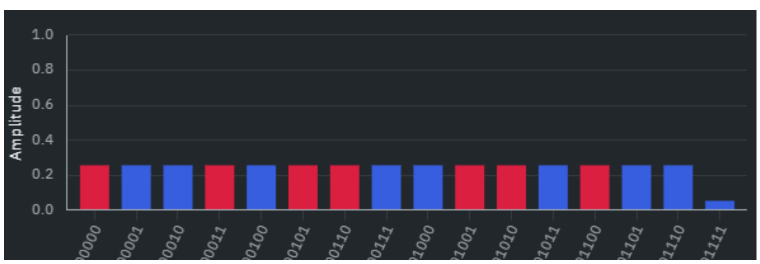


Figura 16 – Aplicação das portas X no operador de difusão da terceira iteração.
Fonte: IBM-*Quantum Learning / Composer*.

17. Aplicar porta MCZ (Figura 17):

$$
\begin{aligned}
MCZ|\psi_{16}\rangle = |\psi_{17}\rangle = \frac{33}{128}\Big(-|0000\rangle + |0001\rangle + |0010\rangle - |0011\rangle + |0100\rangle - |0101\rangle \\
- |0110\rangle + |0111\rangle + |1000\rangle - |1001\rangle - |1010\rangle \\
+ |1011\rangle - |1100\rangle + |1101\rangle + |1110\rangle - \frac{7}{33}|1111\rangle\Big)
\end{aligned}
$$

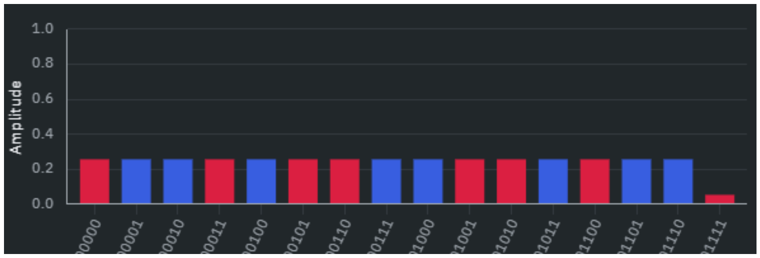


Figura 17 – Aplicação da porta *MCZ* na terceira iteração do operador de difusão.
Fonte: IBM-*Quantum Learning / Composer*.

18. Aplicar portas X (Figura 18):

$$
\begin{aligned}
X^4\,|\psi_{17}\rangle = |\psi_{18}\rangle = \frac{33}{128}\Big(&-|1111\rangle+|1110\rangle+|1101\rangle-|1100\rangle+|1011\rangle\\
&-|1010\rangle-|1001\rangle+|1000\rangle+|0111\rangle-|0110\rangle-|0101\rangle\\
&+|0100\rangle-|0011\rangle+|0010\rangle+|0001\rangle-\frac{7}{33}|0000\rangle\Big)\\
=\frac{33}{128}\Big(&-\frac{7}{33}|0000\rangle+|0001\rangle+|0010\rangle-|0011\rangle\\
&+|0100\rangle-|0101\rangle-|0110\rangle+|0111\rangle+|1000\rangle-|1001\rangle\\
&-|1010\rangle+|1011\rangle-|1100\rangle+|1101\rangle+|1110\rangle-|1111\rangle\Big)
\end{aligned}
$$

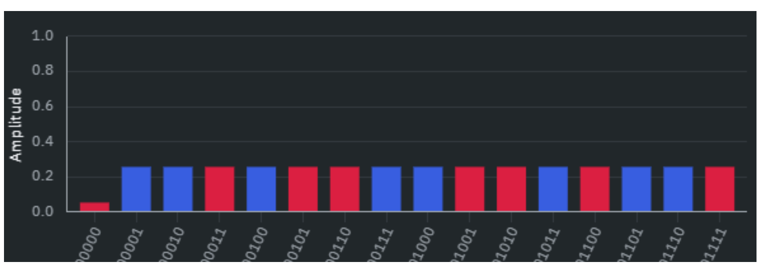


Figura 18 – Última aplicação das portas X na terceira iteração do operador de difusão.
Fonte: IBM-*Quantum Learning / Composer*.

19. Para finalizar a terceira e última iteração, aplicamos novamente as portas *Hadamard* (Figura 19):

$$
\begin{aligned}
H^{\otimes 4}\,|\psi_{19}\rangle = |\psi_{20}\rangle = \frac{13}{256}\Big(&|0000\rangle+|0001\rangle+|0010\rangle+|0011\rangle+|0100\rangle+|0101\rangle\\
&+|0110\rangle+|0111\rangle+|1000\rangle+|1001\rangle+|1010\rangle+|1011\rangle\\
&+|1100\rangle+|1101\rangle+|1110\rangle-\frac{251}{13}|1111\rangle\Big)
\end{aligned}
$$

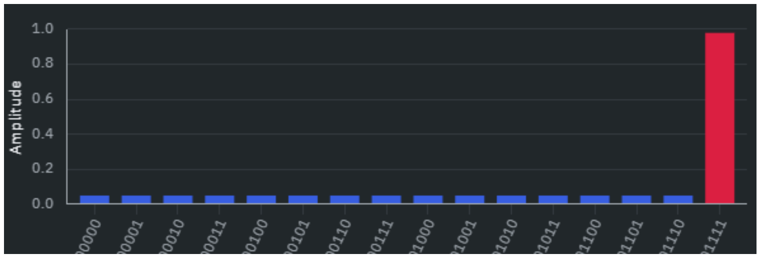
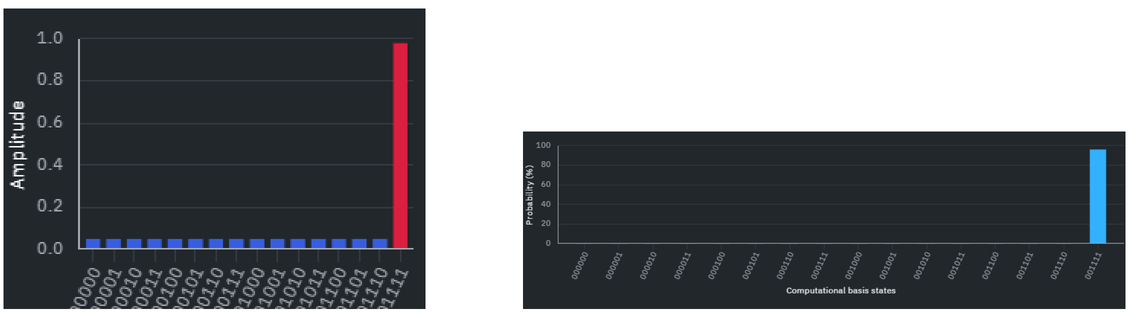


Figura 19 – Probabilidades finais de $|\psi\rangle$ após as três iterações do Algoritmo de Grover.
Fonte: IBM-*Quantum Learning / Composer*.

Ao final do processo, o estado marcado $|1111\rangle$ aparece com amplitude máxima, o que confirma a eficiência do Algoritmo de Grover em aumentar a probabilidade do estado desejado em um espaço de busca não estruturado.

Figura 20 – Amplitude e probabilidades após medição final do circuito.



(a) Amplitude do estado $|1111\rangle$ igual a $0,98$.

(b) Probabilidades finais de cada estado do espaço de busca, com 96,13% para o estado $|1111\rangle$.

Fonte: IBM-*Quantum Learning / Composer*.

APÊNDICE B – Reconstrução de Probabilidades Multi-Qubit a partir do `Estimator`

Ao utilizar o `Estimator` do *Qiskit*, o resultado retornado consiste nos valores esperados dos operadores de medição, tipicamente expressos em termos de combinações de observáveis do tipo Z_i, onde Z é o operador de Pauli-Z.

Para um sistema de n *qubits*, a base computacional $\{|0\rangle, |1\rangle\}$ é associada aos autovalores $+1$ e -1 do operador Z, respectivamente. Assim, a probabilidade de cada estado base pode ser reconstruída a partir das médias dos produtos de operadores Z.

Para o caso de $n = 4$, é necessário obter:

- As expectativas de 1 ponto: $\langle Z_0 \rangle$, $\langle Z_1 \rangle$, $\langle Z_2 \rangle$, $\langle Z_3 \rangle$.
- As expectativas de 2 pontos: $\langle Z_0 Z_1 \rangle$, $\langle Z_0 Z_2 \rangle$, $\ldots$.
- As expectativas de 3 pontos: $\langle Z_0 Z_1 Z_2 \rangle$, $\ldots$.
- A expectativa de 4 pontos: $\langle Z_0 Z_1 Z_2 Z_3 \rangle$.

A fórmula geral para reconstruir a probabilidade $P(b_0 b_1 b_2 b_3)$, onde $b_k \in \{0, 1\}$, é:

$$P(b_0 b_1 b_2 b_3) = \frac{1}{16} \sum_{s_0=0}^{1} \sum_{s_1=0}^{1} \sum_{s_2=0}^{1} \sum_{s_3=0}^{1} \left[\prod_{j=0}^{3} (-1)^{b_j s_j} \right] \langle Z_0^{s_0} Z_1^{s_1} Z_2^{s_2} Z_3^{s_3} \rangle \tag{1}$$

onde $Z_j^0 = I$ (identidade no *qubit* j) e $Z_j^1 = Z_j$.

Este método é uma aplicação direta da transformada inversa de Walsh–Hadamard sobre as expectativas, convertendo-as em probabilidades na base computacional.

APÊNDICE C – Demonstração da Porta *MCZ* a partir de *MCX* e *Hadamard*

O objetivo deste apêndice é demonstrar que a porta Z-multi-controlada (MCZ) pode ser construída a partir de uma porta X-multi-controlada (MCX) e duas portas *Hadamard* (H). A sequência é dada por:

$$MCZ = H \text{ MCX } H \tag{2}$$

onde H é a porta Hadamard aplicada no qubit alvo da MCX.

Demonstração

A porta MCZ que intenta-se criar possui alvo no último *qubit* apenas, sendo assim, a aplicação das H's deve acontecer somente no último canal. Em termos de portas lógicas e matrizes, isso implica que a Equação 2, na verdade, é

$$MCZ_{16x16} = (I \otimes I \otimes I \otimes H)\ MCX_{16x16}\ (I \otimes I \otimes I \otimes H) \tag{3}$$

A matriz resultante da operação $(I \otimes I \otimes I \otimes H)$ é

$$\left(I^{\otimes 3} \otimes H \right)_{16x16} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}_{16x16} \tag{4}$$

Já a matriz de MCX é

$$MCX_{16x16} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}_{16x16} \tag{5}$$

Voltando à Equação 3, precisamos fazer o produto As matrizes correspondentes a X, Z e H são, respectivamente

Para um sistema com múltiplos qubits, considere uma MCX com $n-1$ qubits de controle e 1 qubit alvo:

$$\text{MCX}\,|c_1c_2\ldots c_{n-1}t\rangle = |c_1c_2\ldots c_{n-1}\rangle \otimes X^{c_1\cdots c_{n-1}}\,|t\rangle \tag{6}$$

Aplicando Hadamard antes e depois da MCX no qubit alvo:

$$\begin{aligned} H\cdot MCX\cdot H\,|c_1c_2\ldots c_{n-1}t\rangle &= H\cdot MCX\big(|c_1c_2\ldots c_{n-1}\rangle \otimes H\,|t\rangle\big) \\ &= H\big(|c_1c_2\ldots c_{n-1}\rangle \otimes X^{c_1\cdots c_{n-1}}H\,|t\rangle\big) \\ &= |c_1c_2\ldots c_{n-1}\rangle \otimes \big(HX^{c_1\cdots c_{n-1}}H\big)\,|t\rangle \\ &= |c_1c_2\ldots c_{n-1}\rangle \otimes Z^{c_1\cdots c_{n-1}}\,|t\rangle \\ &= MCZ\,|c_1c_2\ldots c_{n-1}t\rangle \end{aligned}$$

Portanto, a sequência $H\cdot MCX\cdot H$ no qubit alvo é equivalente à porta MCZ.

Representação em Qiskit

No Qiskit, podemos implementar a MCZ usando a MCX e portas Hadamard como segue:

Figura 21 – Implementação de porta *MCZ* a partir de *MCX* e *H*’s

```
from qiskit import QuantumCircuit

qc = QuantumCircuit(4)  # 3 controles + 1 alvo
qc.h(3)                 # Hadamard no qubit alvo
qc.mcx([0,1,2], 3)      # MCX com controles 0, 1 e 2 e alvo 3
qc.h(3)                 # Hadamard no qubit alvo
qc.draw('mpl')
```

APÊNDICE D – Reconstrução de probabilidades multi-qubit a partir de valores esperados

Ao utilizar o `Estimator` do Qiskit, obtém-se diretamente os valores esperados $\langle P \rangle$ de operadores de Pauli sobre o estado quântico final do circuito, sem acessar diretamente as contagens (*counts*). Entretanto, é possível reconstruir a distribuição de probabilidades dos estados computacionais a partir desses valores esperados.

Para um sistema de n qubits, a matriz densidade ρ pode ser expandida na base de Pauli:

$$\rho = \frac{1}{2^n} \sum_{P \in \mathscr{P}^n} \langle P \rangle P$$

onde:

- $\mathscr{P} = \{I, X, Y, Z\}$ é o conjunto dos operadores de Pauli de um único qubit.

- $\mathscr{P}^n$ representa todos os produtos tensoriais possíveis desses operadores para n qubits.

- $\langle P \rangle = \mathrm{Tr}(\rho P)$ é o valor esperado do operador P.

A probabilidade p_x de observar o estado computacional $|x\rangle$ (com x variando de 0 a $2^n - 1$) é obtida por:

$$p_x = \langle x | \rho | x \rangle$$

Substituindo a expansão de ρ, obtém-se:

$$p_x = \frac{1}{2^n} \sum_{P \in \mathscr{P}^n} \langle P \rangle \langle x | P | x \rangle$$

Observa-se que:

- Apenas os operadores de Pauli contendo I e Z contribuem para p_x, pois X e Y possuem elementos fora da diagonal.

- Para cada qubit k:

$$\langle b_k | Z | b_k \rangle = (-1)^{b_k}, \quad \langle b_k | I | b_k \rangle = 1$$

onde $b_k \in \{0, 1\}$ é o bit k do estado $|x\rangle$.

Portanto, para $n = 4$, a fórmula final torna-se:

$$p_x = \frac{1}{16} \sum_{P \in \{I,Z\}^4} \langle P \rangle \prod_{k=0}^{3} \left[\begin{cases} 1, & P_k = I \\ (-1)^{b_k}, & P_k = Z \end{cases} \right]$$

Esse método permite reconstruir a distribuição completa de probabilidades apenas a partir dos valores esperados $\langle P \rangle$ dos produtos tensoriais de operadores I e Z, obtidos via `Estimator`.