UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Gabriel Turatti Andrade

Implementação de um simulador de circuitos Clifford no Ket

Florianópolis
2025

Gabriel Turatti Andrade

Implementação de um simulador de circuitos Clifford no Ket

Trabalho de Conclusão de Curso do Curso de Graduação em Ciências da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina para a obtenção do título de bacharel em Ciência da Computação.
Orientadora: Profa. Dra. Jerusa Marchi
Coorientador: Me. Evandro Chagas Ribeiro da Rosa

Florianópolis
2025

Ficha de identificação da obra

A ficha de identificação é elaborada pelo próprio autor.

Orientações em:

http://portalbu.ufsc.br/ficha

Gabriel Turatti Andrade

Implementação de um simulador de circuitos Clifford no Ket

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "bacharel em Ciência da Computação" e aprovado em sua forma final pelo Curso de Graduação em Ciências da Computação.

Florianópolis, 03 de Dezembro de 2025.

Profa. Lúcia Helena Martins Pacheco, Dra.
Coordenadora do Curso

Banca Examinadora:

Profa. Dra. Jerusa Marchi
Orientadora

Evandro Chagas Ribeiro da Rosa, Me.
Coorientador
Universidade Federal de Santa Catarina

Eduardo Willwock Lussi, Me.
Avaliador
Universidade Federal de Santa Catarina

Prof. Pedro Belin Castellucci, Dr.
Avaliador
Universidade Federal de Santa Catarina

“Ever Onward, Against the Dark.” - Worth the Candle

AGRADECIMENTOS

Nada na vida nós fazemos sozinhos, por isso, eu quero agradecer a todas as pessoas que estiveram comigo nessa jornada.

Um agradecimento especial a Emily, por todas as nossas conversas de filmes, fofocas, e por me acompanhar nas minhas loucuras. Nossa rotina de graduação é algo que para sempre sentirei falta.

Erva Doce, que apesar de eu nunca ter visto seu rosto, sempre esteve comigo em todos os momentos que eu precisei, principalmente para falar mal de homem. Eu espero que aguentes até 2029 para que nós possamos enterrar um pastel na praia.

Gu, não sei como expressar o quanto ele significa para mim. Não imagino que eu encontrarei outra pessoa com quem eu consiga ser tão próximo.

Alpha Fox, pelas recomendações de música que mudaram o rumo da minha vida. Espero conseguir escutar todas elas eventualmente.

Meu pai, por me dar todo o apoio que eu precisei para chegar aqui. E mais importante do que isso, por ser um bom amigo com quem pude compartilhar histórias de vida.

Arlen, suas lições de vida continuam comigo depois de décadas, e continuarão pelo resto da minha vida.

Os Gabrieis, por aguentarem todos os meus hiperfocos em trens e puzzles.

Minha orientadora Jerusa, pelas suas aulas incríveis que me incentivaram a aprender mais sobre a área de Teoria da Computação. E ao Evandro por me auxiliar tanto nas partes técnicas desse TCC.

O Clube LGBT noturno do LIAA, pelas nossas aventuras e fofocas.

O BEGGS-JEM por ser a minha família na graduação. Bruno pelas piadas hilárias, Eric pelas conversas nerds, Gabriela pelas conversas de música, Samantha pelas conversas de jogos, Julien pelas conversas psicológicas, e Marco, por sempre me ouvir tagarelando sobre a minha vida.

E a todas as pessoas que vieram na minha monitoria, espero que eu tenha conseguido ajudar vocês, pois eu certamente me diverti tentando.

RESUMO

Atualmente, computadores quânticos possuem pouco poder de processamento, pela alta taxa de erros e a decoerência dos qubits, tornando difícil o desenvolvimento de algoritmos quânticos que sejam úteis na pratica. Devido a esse fator, existe uma demanda por programas que simulem classicamente o fenômeno de forma eficiente. Para isso, existe o Ket, uma plataforma de desenvolvimento de algoritmos quânticos que permite a simulação de circuitos quânticos em computadores clássicos. O objetivo desse trabalho foi implementar um novo módulo de simulação no Ket, baseado no formalismo do Tableau. Esse método permite simulações em tempo polinomial em relação a quantidade de qubits, para circuitos cujas portas estejam no grupo de Clifford, utilizando uma estratégia de representação de estados através do grupo de estabilizadores daquele estado, ao invés das amplitudes em si. Com esse projeto, agora é possível realizar simulações no Ket com mais eficiência, permitindo que circuitos de correção de erros, comumente feitos apenas com operadores de Clifford, sejam simulados em tempo polinomial. Também foi implementada a opção de usar portas quânticas fora do grupo de Clifford, como a porta T, que permitem operações universais, mas que aumentam exponencialmente o tempo da simulação para cada porta usada, ou seja, um circuito com várias portas T pode ser inviável de simular. Para simular essas portas, o módulo do Tableau foi estendido para permitir qualquer porta fora do grupo de Clifford. O intuito dessa alteração é que circuitos com poucas portas fora do grupo comparadas com portas dentro do grupo ainda podem ser simulados em um tempo aceitável. Sendo assim, esse módulo pode ser escolhido para simular um dado circuito caso ele seja mais eficiente do que o simulador padrão do Ket. O módulo foi implementado em Python e integrado com o Ket. Depois de implementado, a eficiência desses algoritmos foi medida para vários circuitos, puramente de Clifford com vários qubits, com poucas portas T, e com várias portas de Clifford, analisando também como o custo escala dependendo da quantidade de qubits e portas utilizadas. As simulações com apenas portas de Clifford obtiveram sucesso, conseguindo simular até 2000 qubits, para os circuitos com portas fora do grupo, os tempos de simulação tornam-se rapidamente proibitivos a medida que a quantidade de portas cresce, conforme esperado.

Palavras-chave: Computação Quântica. Simulação Quântica. Grupo de Clifford. Ket. Plataforma Quântica.

ABSTRACT

Currently, quantum computers have low processing power, because of the lack of qubits and high rates of noise, making it difficult to study quantum computing in practice. Because of these factors, there exists a demand for software that can simulate classicaly this phenomenon and in an efficient manner. For that, Ket was created, a platform for quantum programming that allows simulation of quantum circuits in classical computers, this work has the objective of implementing a new simulation module inside Ket, based on the tableau method. This method allows for simulations in polynomial time in relation to the qubits, for circuits that are inside the Clifford group, it uses a strategy of representing the states through the group of stabilizer operators for that state instead of the amplitudes themselves. Therefore, with this project, it is now possible to run simulations inside Ket with more efficiency, allowing for error correction circuits, commonly made of Clifford gates, to be run in polynomial time. The option to use T gates and other arbitrary gates that allow for universal computation was also added, which allows for universal computing, but that have exponential cost relative to the quantity of non-clifford gates used, for that, the tableau module was extended to allow gates outside of the Clifford group. The point of such alteration is that circuits with few non-clifford gates to clifford gates ration can still be simulated in a feasible time. Therefore, this module can be chosen to simulate a given circuit in case the user finds it faster than the default Ket simulator. The module was implemented in Python and integrated with Ket. After implementing it, its efficiency was measured through various circuits, including some with only Clifford gates, with a lot of clifford gates, and with few non-clifford gates, analyzing how the time cost scales depending on the quantity of qubits and gates used. The simulations with only clifford gates were a success, achieving around 2000 qubits simulated in a feasible time, for the simulations with gates outside of the clifford group, the simulation times would quickly become prohibitive as the amount of gates grew, as expected.

Keywords: Quantum Computing. Quantum simulation. Clifford Group. Ket. Quantum Platform.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Um tema relevante da computação é a complexidade temporal de problemas computacionais, isto é, saber o quão demorado um problema se torna à medida que as instâncias dele aumentam de tamanho. Por exemplo, procurar o nome de uma pessoa em uma lista não organizada de nomes possui complexidade linear, pois no pior caso, a ausência do nome na lista, esta precisa ser completamente percorrida.

Esse tipo de problema é classificado como complexidade $O(n)$ (Cormen et al., 2009), pois o problema cresce, em pior caso, tão rápido quanto a função n. Grover mostrou, em 1996, que era possível encontrar um objeto em uma lista não ordenada utilizando apenas $O(\sqrt{n})$ operações (Grover, 1996). Fazendo uso dos fenômenos da mecânica quântica, esse novo paradigma de computação vem se mostrando capaz de resolver problemas de uma forma diferente dos computadores clássicos, e em alguns casos, com complexidades inferiores.

Existe uma procura constante por algoritmos mais eficientes para os problemas matemáticos, alguns problemas são atualmente tão complexos que mesmo instâncias pequenas deles não são computáveis em tempo hábil. Esse é o caso para problemas como o de fatoração de números e logaritmo discreto, que em computadores clássicos custa tempo subexponencial para resolver (Menezes; Oorschot; Vanstone, 1996).

Muitos problemas conhecidos já estavam bem estabelecidos em suas classes de complexidade, a busca em uma lista não organizada, por exemplo, não parecia ser possível computar de modo mais eficiente do que $O(n)$. Atualmente, acredita-se que o problema do caixeiro viajante, assim como outros problemas da classe NP-completo, são problemas que não podem ser resolvidos em tempo polinomial (Sipser, 2012), e até hoje, mesmo com computadores quânticos, nenhum algoritmo foi encontrado para resolver qualquer um deles em tempo polinomial.

Os problemas para os quais não há, até o momento, algoritmo em tempo polinomial, nem tampouco foram demonstrados estarem na classe NP-completo, são chamados de NP-intermediários (Sipser, 2012). Para esses alguns algoritmos quânticos foram desenvolvidos que resolvem o problema em tempo polinomial, ou seja, garantem uma aceleração exponencial em relação aos clássicos, como é o caso do algoritmo de Shor, que consegue resolver, usando computação quântica, o problema de fatoração de números, e o problema de logaritmos discretos (Shor, 1994).

Entretanto, a computação quântica tem suas limitações, pois, para conseguir executar um algoritmo quântico, é necessário modelar o problema alvo de uma forma equivalente no paradigma quântico. Essa tarefa não é trivial, existem diversas restrições na maneira de pensar e representar os problemas, de formas que conflitam com a versão clássica.

Além da computação, na área da química também existe uma demanda por computação quântica, pois a natureza de superposição e entrelaçamento dos qubits permite

simular facilmente o comportamento de moléculas, algo que para computadores clássicos também é visto como um problema de complexidade alta. Diversos trabalhos foram feitos estudando o comportamento de moléculas simples usando computadores quânticos, e é esperado que com computadores melhores, seja possível estudar o comportamento de moléculas mais complexas como aquelas usadas em medicamentos (Cheng et al., 2020).

O avanço nessa área é dificultado pela falta de computadores quânticos que podem ser de fato utilizados. Modelar algoritmos quânticos se torna uma tarefa muito mais difícil quando o objeto da pesquisa existe apenas de forma teórica. Para isso, muitos projetos foram criados com o propósito de auxiliar o desenvolvimento e simulação da computação quântica, tornando ela um objeto mais acessível, entre eles a linguagem de programação Ket (Rosa; Santiago, 2021).

Essa simulação é uma tarefa de complexidade exponencial, o que inviabiliza as simulações de larga escala. Portanto, grande parte da pesquisa na área de simulação quântica se baseia em reduzir a complexidade da simulação, em busca de aumentar a quantidade de qubits simulados.

Dentre as diversas formas de simular computação quântica, há uma técnica que pode ser feita em tempo polinomial, ela é voltada para circuitos de Clifford, os quais são gerados por um subconjunto das portas quânticas possíveis. Esse subconjunto é caracterizado por conter as portas: Hadamard, Phase, e CNOT (Gottesman, 1997).

A implementação de um simulador que use o grupo de Clifford não é novidade, e já foi explorado em outros trabalhos, demonstrando sua eficácia para circuitos dessa sub categoria (Garner et al., 2025). Nesse trabalho, será implementado a mesma estratégia para a plataforma Ket, com uma adição de simular portas fora do grupo.

Esse grupo de portas não é capaz de criar o conjunto universal de operações, o que limita as simulações possíveis, pois vários algoritmos quânticos usam portas fora do grupo de Clifford. Apesar das limitações, esse subconjunto de operações ainda permite o entrelaçamento de qubits e a superposição, o que ajuda, em parte, a modelar algoritmos.

A limitação verdadeira está nas amplitudes arbitrárias de probabilidade. Em um computador quântico, os qubits podem estar em uma superposição de diversos estados, e cada estado pode ter uma probabilidade qualquer de ser medido, desde que todas as probabilidades somem em 1. Por exemplo, um qubit pode ter 10% de chance de ser medido 0 e 90% de chance de medir 1.

No grupo de Clifford, qubits em superposição sempre tem 50% de chance de serem medidos como $|0\rangle$ e 50% de $|1\rangle$. O único jeito de atingir valores arbitrários de probabilidade é inserindo uma porta fora do grupo, como a porta T, que torna a computação universal.

Outra área importante desse trabalho é a ideia de correção de erros. Atualmente, computadores quânticos são extremamente ruidosos, isso é, são suscetíveis a erros, tornando difícil realizar computações demoradas sem os erros inviabilizarem a informação. Para isso, estão surgindo diversos estudos sobre como corrigir os erros do computador, que utilizam

circuitos de correção de erros, formados principalmente por operadores de Clifford, e baseados na ideia de estabilizadores, ambos elementos viáveis de calcular polinomialmente com o método de Tableau (Devitt; Munro; Nemoto, 2013).

Estabilizadores são operadores que não alteram o estado do computador, isso é relevante para correção de erro pois se um estabilizador é aplicado a um estado, e o seu valor é alterado, significa que aquele estado foi afetado por um ruído.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O principal objetivo deste trabalho é implementar um simulador quântico alternativo para o Ket, que poderá ser utilizado opcionalmente pelo usuário, otimizado para simular circuitos de Clifford em tempo polinomial. E que permita o uso de portas fora do grupo de Clifford usando uma extensão do método de Tableau. Permitindo, dessa forma, a simulação de circuitos maiores no Ket com complexidades temporais melhores.

1.1.2 Objetivos Específicos

O1 Estudar a literatura atualizada sobre simulação de circuitos de Clifford.

O2 Implementar um código para a simulação de Tableau seguindo a literatura, no Python.

O3 Adicionar operações fora do grupo de Clifford na simulação.

O4 Integrar o código com a plataforma Ket.

O5 Avaliar o desempenho do código desenvolvido.

1.2 ESTRUTURA DO DOCUMENTO

No Capítulo 2 será introduzido a base da computação quântica, com os conceitos de qubits, os operadores que agem nos qubits, o grupo de operadores de Clifford, o grupo de operadores Pauli, e as representações matemáticas para cada um desses objetos. A última seção do capítulo será focada no conceito de Estabilizadores.

O método de simulação se baseia em representar estados quânticos através do grupo de estabilizadores do estado, utilizando uma tabela que codifica o conjunto gerador desse grupo, o que será explicado no Capítulo 3. Em cada seção será explicado como que os operadores de Clifford podem ser aplicados na tabela, quais mudanças isso implica, e como medir os valores. Por fim serão mostrados os operadores extras: portas T, portas Toffoli e portas genéricas.

Capítulo 4 detalha as informações sobre como foi feita a implementação do tableau no Python, as otimizações feitas para reduzir o tempo de computação, assim como a integração desse código dentro da plataforma Ket. Também informando as adições feitas para que as portas fora do grupo pudessem ser executadas.

Para testar o desempenho dessas implementações, o Capítulo 5 relata quais circuitos foram testados, e como o tempo de execuçao escala de acordo com a quantidade de qubits e de portas lógicas utilizadas para cada circuito. Esses testes se separam em dois tipos, os que tem exclusivamente portas de Clifford, e os que utilizam portas fora do grupo.

Por fim, no Capítulo 6 serão discutidos os resultados do trabalho, e as conclusões obtidas a partir dos testes de desempenho realizados, assim como possíveis trabalhos futuros.

1.3 TRABALHOS CORRELATOS

O formalismo de Tableau é algo que já foi estudado em diversos trabalhos como uma ferramenta para criar circuitos de correção de erro, no entanto, o uso do Tableau para a simulação de circuitos foi muito pouco explorado na literatura.

Há um trabalho que utiliza a ideia de simular circuitos de clifford de forma polinomial, expandindo na ideia do tableau com o conceito de grafos de estado. Nesse caso, ao invés de representar estados pelos estabilizadores, o estado é representado por um grafo de quais qubits interagiram, e em que estão cada um está (Anders; Briegel, 2006).

Existe também Um trabalho que utiliza o conceito de tableau como originalmente proposto, expandindo ele para o uso de portas T com *magic states*, essa implementação é feita de forma aproximada, o real estado da simulação não é possível ser obtido, no lugar existem procedimentos feitos para que a resposta seja próxima do desejado dado uma margem de erro (Bravyi; Gosset, 2016).

Essa mesma ideia é evoluída em outro trabalho, tentando diminuir a complexidade temporal da aplicação de portas T, novamente a simulação não é exata, e apenas uma aproximação do estado real é atingida (Huang; Love, 2019).

Dessa forma, não foi encontrado nenhum trabalho que faça a simulação quântica usando o formalismo de Tableau e que consiga representar com exatidão estados não cliffordianos. Além disso, os trabalhos encontrados se restringem a usar apenas portas T, o que torna lento a simulação de portas de Toffoli e outras portas mais complexas, devido a necessidade de decompor elas em múltiplas portas T.

Nesse trabalho, serão feitas formas de aplicar portas genéricas e portas controladas diretamente no Tableau, sem que seja necessário a decomposição custosa das portas, através de otimizações que serão tratadas no Capítulo 4.

2 COMPUTAÇÃO QUÂNTICA

Nesta seção, são tratados todos os assuntos básicos de computação quântica, necessários para entender o formalismo do tableau. Introduzindo formalmente o conceito de qubit na Seção 2.1, e os operadores de qubits na Seção 2.2, junto com as definições matemáticas por trás de como eles funcionam. Também será brevemente comentado sobre a esfera de Bloch na Seção 2.1.1, uma forma de visualizar um qubit.

Com isso definido, serão explicados, com mais detalhes, os dois principais grupos de operadores desse trabalho, na Seção 2.2.2 sobre o grupo de Clifford, o qual representa as operações que podem ser simuladas pelo tableau. E na Seção 2.2.3 sobre o grupo de Pauli, que é utilizado para simular o tableau em si. Por fim, na Seção 2.3, será explicado o conceito de Estabilizadores que será necessário para entender o tableau.

2.1 QUBITS

A unidade básica da computação quântica é o qubit, utilizado para guardar informação, análogo aos bits de um computador clássico. No entanto, enquanto um bit normal sempre vai ter o valor de 0 ou 1, o qubit se comporta de maneira mais complexa, parecendo estar em 0 e 1 ao mesmo tempo, o que é chamado de superposição. Isso acontece apenas enquanto o qubit não for observado (o que nesse caso equivale a ser medido, ou lido) (Yanofsky; Mannucci, 2008).

Em teoria, essa capacidade de estar em superposição, junto com a de emaranhar qubits, permite ao computador quântico uma vantagem exponencial em relação aos computadores clássicos (Cuffaro, 2013). Isso ocorre porque a informação de cada estado nessa superposição é computada em paralelo. Ou seja, sendo n o número de qubits de um computador quântico, é possível executar 2^n estados diferentes no mesmo passo de computação.

Porém, para resolver os problemas computacionais, precisamos obter valores de retorno, para isso, é preciso medir os qubits. Nesse momento de observação, a superposição deles colapsa, e apenas um estado representado por 0's e 1's é visto, igual ao de um computador clássico. Quando os estados colapsam, a informação de todas as outras execuções diferentes da que foi observada, se perde. Isso estabelece um desafio para fazer uso da computação exponencialmente paralela do computador quântico, pois não é útil computar 2^n estados diferentes se apenas a informação de um deles será vista.

Dizer que um qubit está em um estado de superposição de 0 e 1 significa dizer que ele tem uma certa probabilidade de, quando medido, colapsar em 0, e outra probabilidade, complementar, de colapsar para 1. Tendo isso em mente, é possível representar um qubit com um valor alfa (α) e um valor beta (β), representando os valores da amplitude de probabilidade de 0 e de 1, respectivamente. Essa amplitude pode ser qualquer número complexo, dada a restrição de que $|\alpha|^2 + |\beta|^2 = 1$, isso é, a probabilidade do qubit estar em 0 somado à probabilidade do qubit estar em 1 deve ser igual a 100%. É comum essa

informação ser representada como $\alpha\,|0\rangle + \beta\,|1\rangle$ usando notação Bra-ket, dessa forma, uma superposição de $|0\rangle$ e $|1\rangle$ é representada como uma soma dos vetores kets, ponderada pelas amplitudes.

O motivo de números complexos serem usados, em invés de apenas números reais, se dá pois além do valor de probabilidade, os qubits também podem ter um sinal, chamado fase, com valor complexo unitário. Isso não afeta diretamente no valor observado, mas permite que qubits de determinadas fases interfiram uns com os outros de forma construtiva, intensificando a probabilidade de um dado resultado aparecer, ou de forma destrutiva, em que duas ou mais probabilidades se cancelam, removendo a chance de um determinado estado aparecer.

Essa interferência é o foco de algoritmos quânticos como o Grover, pois permite que a probabilidade da resposta desejada aparecer seja intensificada dentre os 2^n estados possíveis, enquanto os resultados indesejados se cancelem (Grover, 1996).

As representações dos qubits em simulações clássicas e no papel focam principalmente na álgebra linear, que é capaz de representar fielmente o comportamento dos qubits e operadores. Nela, os estados quânticos de n qubits são representados como vetores coluna de tamanho 2^n, em que cada posição do vetor guarda a amplitude de um dos possíveis estados da superposição.

Um único qubit pode ser representado por uma soma dos kets $|0\rangle$ e $|1\rangle$ que representam respectivamente os vetores $\binom{1}{0}$ e $\binom{0}{1}$. Para representar estados quânticos com mais de um qubit, como por exemplo, o estado $|10\rangle$, basta realizar o produto tensorial entre os vetores dos estados $|1\rangle$ e $|0\rangle$, ou seja, $|1\rangle \otimes |0\rangle$. Um exemplo dessa operação é dado na Seção 2.2.1.

Por exemplo, com 3 qubits, existem 8 possibilidades de valores $(000, 001, 010, ..., 111)$, e um estado pode estar em uma superposição de todos esses valores ao mesmo tempo. Por exemplo, um estado quântico que estivesse em uma superposição dos valores 000, 011, 100, 111 e as amplitudes de probabilidade desses estados fossem $\frac{8}{\sqrt{169}}$, $\frac{5}{\sqrt{169}}$, $\frac{8}{\sqrt{169}}$, $\frac{4}{\sqrt{169}}$ respectivamente, seria representado dessa forma:

$$\frac{1}{\sqrt{169}} \left[\begin{array}{c|l} 8 & 000 \\ 0 & 001 \\ 0 & 010 \\ 5 & 011 \\ 8 & 100 \\ 0 & 101 \\ 0 & 110 \\ 4 & 111 \end{array}\right]$$

Para obter a real probabilidade de medir aquele estado, basta multiplicar o valor daquela posição pela sua composta conjugada. No caso dos números reais, isso implica

em obter o valor da amplitude ao quadrado. Nesse exemplo, a chance do estado 100 ser medido, é de $\frac{64}{169}$.

Esse estado também pode ser representado com notação Bra-ket, nessa notação, é explicitado a amplitude apenas dos estados que possuem alguma chance de aparecer. O exemplo anterior na notação braket seria dado como $\frac{8|000\rangle+5|011\rangle+8|100\rangle+4|111\rangle}{\sqrt{169}}$ sendo que $a\,|011\rangle$ nada mais é do que o vetor de 8 dimensões com a 4^{a} posição valendo a e todo resto valendo 0, ou seja, cada ket individualmente representa um pedaço do vetor.

2.1.1 Esfera de Bloch

A esfera de Bloch é um modelo geométrico criado para representar um qubit e as operações que são realizadas nele. Ela auxilia a criar um entendimento mais intuitivo das transformações que ocorrem em um qubit por dadas matrizes.

Um qubit pode ter qualquer valor de amplitude complexo para o $|0\rangle$ e $|1\rangle$, desde que os módulos das amplitudes, ao quadrado, somem em 1, resultando em 4 graus de liberdade. Porém, é sempre possível fatorar a fase global de um estado de forma que a amplitude do $|0\rangle$ seja apenas um número real (Yanofsky; Mannucci, 2008).

Por exemplo, o estado $\frac{1}{\sqrt{2}}(i\,|0\rangle - |1\rangle)$ pode ter o fator imaginário de i removido, resultando no estado $\frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$ equivalente ao anterior.

Ou seja, um qubit pode ser sempre representado em 3 dimensões sem perda de informação, dado isso, o modelo de Bloch foi criado com o intuito de facilitar a visualização de como um qubit se comporta.

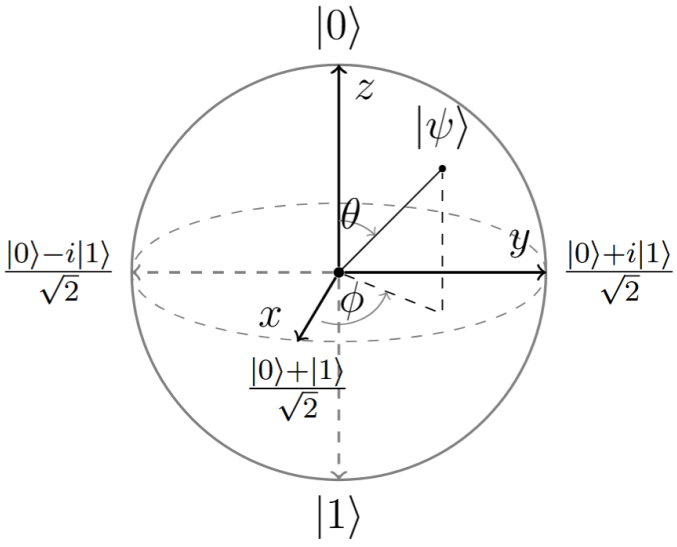


Figura 1 – Esfera de Bloch: Representação 3D de um qubit. Retirado de: (Frisk Kockum; Nori, 2019)

Na Figura 1, um qubit é definido como um ponto na superfície dessa esfera. Se o ponto estiver no polo norte, o qubit representado tem valor $|0\rangle$, se ele estiver no polo sul, tem valor $|1\rangle$. Quando o qubit não está nos polos extremos, ele está em uma superposição

dos dois, o equador da esfera representa todas as superposições de probabilidade igual de ser medido $|0\rangle$ e $|1\rangle$. Quanto mais perto do $|0\rangle$, mais provável é daquele estado ser medido, e vice-versa para o $|1\rangle$.

Os estados $|+\rangle, |-\rangle, |+i\rangle, |-i\rangle$, representados como pontos da esfera nos eixos x e y, são estados com superposição, convenientes de serem nomeados. Eles representam respectivamente: $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, $|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$, e $|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle)$.

As portas de Pauli (X, Y, e Z), que serão vistas na Seção 2.2.3, podem ser interpretadas como rotações do vetor estado nos 3 eixos da esfera, por exemplo, quando a porta Z é aplicada no qubit, o vetor que representa o estado atual vai ser rotacionado 180ž em volta do eixo Z. Vale observar que se o estado estiver exatamente em $|0\rangle$, ele não será afetado pela rotação. Assim como $|+\rangle$ não é afetado por rotações no eixo X, e $|+i\rangle$ não é afetado pelas rotações Y.

2.2 PORTAS LÓGICAS

Análogo ao computador clássico, no computador quântico existem diversas portas lógicas que alteram o estado dos qubits. Porém, além das portas que alteram o valor do qubit, também existem portas que alteram a fase, e portas que levam um estado para uma superposições de estados.

Essas transformações são a base da computação quântica, é com elas que circuitos são montados, e como computadores quânticos atuais são muito custosos de utilizar, existe uma demanda por simulações eficientes desses circuitos.

As principais portas usadas são as de rotação nos eixos tridimensionais, X, Y, Z, a porta de Hadamard, que permite superposição, a porta CNOT que permite o entrelaçamento, além da porta T, que, junto com as outras, permite universalidade na computação (Aaronson; Gottesman, 2004).

A porta H, quando aplicada a um qubit valendo $|0\rangle$, o coloca em uma superposição de $|0\rangle$ e $|1\rangle$ com probabilidade igual, o mesmo acontece quando aplicada a um qubit $|1\rangle$ com a única diferença sendo uma fase negativa relativa no estado $|1\rangle$. Essa operação pode ser representada pela seguinte matriz.

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} \quad = \quad |0\rangle \text{ —}\boxed{H}\text{— } \frac{|0\rangle+|1\rangle}{\sqrt{2}} \tag{1}$$

Cada porta lógica tem uma matriz correspondente, dessa forma, a execução de algoritmos quânticos pode ser descrito através de matrizes que multiplicam vetores, esses vetores representam os estados.

Da mesma forma que um estado de múltiplos qubits pode ser expresso como um produto tensorial de vários qubits. Operadores agindo em vários qubits podem ser expressos com o produto tensorial de operadores de 1 qubit. Por exemplo, a ação de

aplicar a porta X nos 3 últimos qubits do estado $|1111\rangle$ pode ser expressa pela operação $I \otimes X \otimes X \otimes X$. Durante esse trabalho, o símbolo de multiplicação tensorial será omitido, portanto o operador seria expresso como $IXXX\,|1111\rangle = |1000\rangle$.

Para expressar que duas operações foram realizadas sob um estado, uma seguida da outra, é utilizado o $\cdot$, para expressar multiplicação matricial. Por exemplo, para expressar que a porta Z foi aplicada no estado $|0\rangle + |1\rangle$, e depois a porta X, a notação usada nesse trabalho seria $X \cdot Z(|0\rangle + |1\rangle) = -|0\rangle + |1\rangle$.

2.2.1 Exemplo Produto Tensorial

O Produto Tensorial é utilizado tanto para estados quanto para operadores com o propósito de estendê-los para mais qubits, caso seja necessário expressar um sistema de múltiplos qubits, ele é expresso com produtos tensoriais de vetores-estado, e as operações nele são expressas pelos produtos tensorias de operações menores.

Como exemplo, o estado $|01\rangle$ é representado com o seguinte produto tensorial, $|0\rangle \otimes |1\rangle$, o resultado dessa operação é a seguinte.

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 0 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

Para realizar uma operação no estado $|01\rangle$, é necessária uma matriz 4x4, (caso o estado tivesse 3 qubits, seria necessário uma matriz 8x8), essa matriz pode ser montada a partir de matrizes menores que representam operações aplicadas a cada qubit individualmente. Como exemplo, para aplicar uma porta X no primeiro qubit, é necessário realizar a operação X $\otimes$ I, para aplicar uma porta X no segundo, seria I $\otimes$ X. A Equação (3) mostra como seria a matriz para o operador que aplica X no primeiro qubit, e Z no segundo, ou seja, X $\otimes$ Z.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 1 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ 1 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \tag{3}$$

Para operadores com mais qubits são necessários o produto tensorial de mais operadores, na forma que a dimensão do operador tenha que ser a mesma da quantidade de linhas do estado afetado.

2.2.2 Grupo de Clifford

O Grupo de Clifford é definido como todas as operações de matrizes geradas pelo conjunto de portas CNOT, Hadamard e Phase, sob a operação de multiplicação

(Gottesman, 1998).

Esse grupo de operações é relevante para o trabalho, pois qualquer circuito construído apenas a partir de operações do grupo de Clifford pode ser simulado em tempo polinomial usando o método de Tableau (Aaronson; Gottesman, 2004). Será visto em detalhe o que cada operação faz, para depois analisar como simular elas de forma eficiente.

A porta de Hadamard já foi brevemente explicada antes como sendo responsável por superposições. Um exemplo prático, Hadamard aplicada em um estado quântico na superposição $\frac{3|0\rangle+4|1\rangle}{5}$. Para todas as operações quânticas, aplicar ela em uma superposição retorna o mesmo valor que aplicar ela separadamente em cada possibilidade, e somar os resultados. Logo, essa operação resultaria em:

$$\frac{1}{5}(3H\,|0\rangle + 4H\,|1\rangle) = \frac{1}{5}(3\frac{|0\rangle + |1\rangle}{\sqrt{2}} + 4\frac{|0\rangle - |1\rangle}{\sqrt{2}})$$

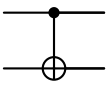
$$=\frac{1}{5\sqrt{2}}(7\,|0\rangle - |1\rangle) \tag{4}$$

A porta Phase permite que uma fase i seja adicionada aos estados de valor $|1\rangle$ no estado quântico. Por exemplo, se a porta Phase é aplicada no estado $(3+2i)\,|0\rangle+(5+4i)\,|1\rangle$, o resultado seria uma fase i aplicada no $|1\rangle$. Ou seja, $(3+2i)\,|0\rangle+(5i-4)\,|1\rangle$. A Equação (5) mostra a matriz correspondente com essa operação.

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = \quad \text{—}\boxed{S}\text{—} \tag{5}$$

A porta CNOT é o que permite o entrelaçamento de qubits, isso é, fazer com que o valor de um qubit dependa do valor de outro. Ela funciona usando dois qubits, um deles funciona como controle, e o outro como alvo. Uma porta X, de negação, que irá trocar o valor do qubit, será aplicada no alvo caso o controle seja igual a $|1\rangle$. Então em um estado quântico do tipo $\frac{1}{\sqrt{14}}(|00\rangle + 3\,|10\rangle + 2\,|11\rangle)$, caso a CNOT seja aplicada com o primeiro qubit sendo o de controle e o segundo sendo o de alvo, o $|10\rangle$ irá se transformar em $|11\rangle$ e vice-versa, transformando o estado em $\frac{1}{\sqrt{14}}(|00\rangle + 3\,|11\rangle + 2\,|10\rangle)$. A Equação (6) mostra a matriz correspondente da operação, como ela afeta dois qubits ela tem tamanho 4 por 4.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \tag{6}$$



2.2.3 Portas de Pauli

As portas de Pauli são as portas X, Y, Z e I, elas também são utilizadas em algoritmos quânticos como a base da computação (Nielsen; Chuang, 2011). Será visto a

definição delas e no Capítulo 3 como que elas se comportam no Tableau. As matrizes que representam elas são as seguintes.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \Rightarrow \quad \text{—}\boxed{X}\text{—} \tag{7}$$

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \Rightarrow \quad \text{—}\boxed{Z}\text{—} \tag{8}$$

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \Rightarrow \quad \text{—}\boxed{Y}\text{—} \tag{9}$$

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow \quad \text{—}\boxed{I}\text{—} \tag{10}$$

Essas portas estão presentes no grupo de Clifford, pois todas elas podem ser derivadas utilizando apenas H, Phase e CNOT, seguindo as equivalências apresentadas na Equação (11), Equação (12) e na Equação (13).

$$\text{—}\boxed{S}\text{—}\boxed{S}\text{—} \Rightarrow \quad \text{—}\boxed{Z}\text{—} \tag{11}$$

$$\text{—}\boxed{H}\text{—}\boxed{Z}\text{—}\boxed{H}\text{—} \Rightarrow \quad \text{—}\boxed{X}\text{—} \tag{12}$$

$$\text{—}\boxed{S^\dagger}\text{—}\boxed{X}\text{—}\boxed{S}\text{—} \Rightarrow \quad \text{—}\boxed{Y}\text{—} \tag{13}$$

Sendo que $S^\dagger$ é a operação inversa de S, que pode ser obtida com três S consecutivos $(S \cdot S \cdot S)$, ou calculando a matriz transposta conjugada de S (Nielsen; Chuang, 2011).

2.2.4 Portas T

Existem infinitos operadores possíveis além daqueles do grupo de Clifford, pois qualquer matriz 2 por 2 unitária representa uma operação válida na computação quântica. No entanto, nem sempre é necessário representar todos eles, pois alguns podem ser descompostos em outras portas. Logo, só é necessário um conjunto finito de portas para que a computação seja universal.

Para este trabalho, sera apresentado uma forma de decompor qualquer operação quântica em varias portas Pauli, e será dado um foco maior para a porta T pois ela é usada na decomposição de diversas portas comumente usadas (Kitaev, 1997).

Uma porta T pode ser interpretada como uma raiz quadrada de uma S, pois ela obedece à identidade $T \cdot T = S$. Outra interpretação para a ação de uma T, é uma transformação que quando aplicada a um estado qualquer, não altera os valores $|0\rangle$ mas aplica uma rotação $\sqrt{i}$ no estado $|1\rangle$. Sendo essa rotação metade da rotação de uma porta

S, um quarto da rotação de uma porta Z, ou um oitavo de uma rotação completa no plano complexo. A matriz é representada na Equação (14).

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{\pi}{4}i} \end{bmatrix} = \text{—}\boxed{T}\text{—} \tag{14}$$

2.3 ESTABILIZADORES

Estabilizadores são a base do método de Tableau, é utilizado o grupo de estabilizadores para representar os qubits, o que ocupa menos espaço que representar um estado pelo seu vetor.

O conceito de estabilizadores é usado para descrever operadores. Um operador é dito como estabilizador de um estado $|\varphi\rangle$ se, quando aplicado ao estado, ele resulta no mesmo estado $|\varphi\rangle$, ou seja, não altera o seu valor. Outra interpretação para essa operação é que o operador é dito como estabilizador de um estado caso aquele estado seja um autovetor do operador com autovalor $+1$.

Um exemplo é o estado $|11\rangle$, ele é estabilizado pelos operadores ZZ e $-IZ$, pois uma porta Z aplicada em um qubit $|1\rangle$, deixa o sinal da fase negativo, mas quando duas portas Z são aplicadas em cada qubit, ambas as portas revertem o sinal, e o estado acaba com amplitude positiva novamente. O operador $-IZ$ tem o mesmo efeito, negando o sinal do qubit $|1\rangle$, mas adicionando uma fase negativa junto, cancelando o efeito no final.

Esses não são os únicos estabilizadores para o estado $|11\rangle$, o operador $-ZI$ também estabiliza esse estado, assim como II, porém, esses operadores podem ser representados como combinações lineares dos operadores anteriores, $-ZI = ZZ \cdot -IZ$, e $II = ZZ \cdot ZZ$. Para cada estado, existe um grupo de estabilizadores, fechado para a operação de multiplicação de matrizes.

No Capítulo 3 onde será apresentado o método de Tableau será visto como o conjunto mínimo gerador desse grupo é relevante. Um conjunto é dito gerador de um grupo quando ele possui as matrizes necessárias para criar todas as matrizes de um dado grupo. Para o estado $|1\rangle$, só as matrizes ZZ e $-IZ$ já conseguem criar todos os operadores estabilizadores.

Dado um operador U_1, é dito que ele comuta com um operador U_2 se e somente se $U_1 \cdot U_2 = U_2 \cdot U_1$, e anti-comuta caso $U_1 \cdot U_2 = -U_2 \cdot U_1$. No formalismo do Tableau, é usado o conceito de portas destabilizadoras, cuja definição é um conjunto em que cada elemento, ou seja, cada operador, anti-comuta com uma matriz do conjunto estabilizador, e comuta com todas as outras (Aaronson; Gottesman, 2004).

Por exemplo, para o estado $|11\rangle$, e geradores de estabilizadores $\{-IZ,\ ZZ\}$, o conjunto de desestabilizadores seria $\{XI,\ XX\}$. O operador XI comuta com $-IZ$, pois $XI \cdot -IZ = -IZ \cdot XI = -XZ$, e anti-comuta com ZZ, pois $ZZ \cdot XI = iYZ$ mas $XI \cdot ZZ = -iYZ$. Já o operador XX comuta com ZZ, pois $ZZ \cdot XX = iYiY = -YY =$

$XX \cdot ZZ$, e anti-comuta com $-IZ$, pois $XX \cdot -IZ = +iXY$, mas $-IZ \cdot XX = -iXY$.

3 MÉTODO DE TABLEAU

A computação quântica pode ser simulada em um computador clássico apenas com vetores e matrizes, essa forma é chamada de simulação de vetor de estados, os vetores representam os qubits, e as matrizes os operadores. Essa simulação pode levar tempo e espaço exponencial para um computador clássico em relação ao número de qubits, portanto, geralmente usam-se outras estratégias para simular qubits de forma mais eficientes.

Esse é o caso do Método de Tableau. Ele se baseia inteiramente na ideia de que estados quânticos podem ser representados unicamente por um grupo de operadores estabilizadores para aquele estado.

Para cada estado quântico, existem diversas combinações de portas estabilizadoras, e esse conjunto tende a crescer exponencialmente conforme a quantidade de qubits cresce. Logo, para representar o grupo de todas as portas estabilizadoras, geralmente se usa apenas um conjunto seleto de portas que juntas, por combinação linear, resulta em todos os operadores possíveis, ou seja, o conjunto gerador daquele grupo.

Para que o grupo seja único para aquele estado, é necessário que o grupo tenha n graus de liberdade, sendo n a quantidade de qubits no estado. Esses graus de liberdade são medidos através do conjunto gerador do grupo. Esse conjunto gerador deve possuir n operadores que são linearmente independentes, ou seja, que não podem ser recriados com os outros operadores do conjunto (Yashin; Elovenkova, 2025).

Esse conceito de portas estabilizadoras e desestabilizadoras é utilizado no Método de Tableau para representar um estado quântico, e dessa forma simulá-lo em um computador clássico (Aaronson; Gottesman, 2004).

Para fazer a simulação, é primeiramente feita uma matriz de tamanho $2n$ por $2n+1$, sendo o n o número de qubits que se deseja simular, e o $+1$ necessário para representar o sinal das portas. A matriz pode ser dividida em 4 quadrantes de tamanho n x n, ignorando a coluna de sinal. Além disso, a matriz é binária, ou seja, os valores de cada célula podem ser apenas 0 ou 1.

	X_1	X_2	X_3	X_4	Z_1	Z_2	Z_3	Z_4	F
D1	0	0	0	0	0	0	0	0	0
D2	0	0	0	0	0	0	0	0	0
D3	0	0	0	0	0	0	0	0	0
D4	0	0	0	0	0	0	0	0	0
S1	0	0	0	0	0	0	0	0	0
S2	0	0	0	0	0	0	0	0	0
S3	0	0	0	0	0	0	0	0	0
S4	0	0	0	0	0	0	0	0	0

Tabela 1 – Exemplo de tabela não preenchida para um estado de n = 4 qubits.

Os dois quadrantes superiores (em que a linha varia de 1 a n) representam os desestabilizadores, já os quadrantes inferiores (de $n+1$ até $2n$) representam os estabilizadores.

Os quadrantes da esquerda representam o uso ou não (baseado no valor binário) da porta X, e os quadrantes da direita representam a aplicação da porta Z. Para aplicar a porta Y, basta que se aplique tanto a X quanto a Z.

As colunas 1 a n representam cada um dos n qubits, as colunas $n+1$ até $2n$ também representam os mesmos n qubits mas em relação a porta Z. Basicamente, sendo uma coluna k, representando a aplicação da porta X para o k-ésimo qubit, a coluna $k+n$ irá representar a porta Z para o mesmo qubit k.

Por convenção, o estado quântico é inicializado em $|0^n\rangle$, e, portanto, a matriz também é inicializada nesse ponto. Sendo assim, os estabilizadores ficam como $III...IZ$, $III...ZI$, ..., $ZII...II$. Para um caso em que $n = 3$, o estado inicial seria $|000\rangle$ e a matriz inicial teria nos quadrantes inferiores os valores 0 para todas as portas X, e apenas uma Z para cada coluna do quadrante inferior direito, geralmente disposto na diagonal principal por conveniência, mas não obrigatoriamente.

Já nos quadrantes superiores, nenhuma porta Z está ativa, e apenas as portas X estão ativas, sendo, por convenção, a diagonal principal da matriz como 1, portanto, o primeiro desestabilizador é a porta X aplicada no primeiro qubit, o segundo é a porta X aplicada no segundo e assim por diante. Conforme a Tabela 2.

Nessa seção, é explicado como funciona o método de tableau e como ele representa estados obtidos através de circuitos de Clifford. Começando na Seção 3.1 em que são explicadas cada uma das portas que geram o grupo de Clifford, Hadamard na Seção 3.1.1, Phase na Seção 3.1.2, e CNOT na Seção 3.1.3. Cada uma dessas operações é seguida por exemplos de que alterações são realizadas na tabela para que ela represente o novo estado obtido por aquele operador.

Depois dessas operações, é explicado como a medição do estado funciona dentro do tableau, introduzindo funções que auxiliam a operação na Seção 3.2.1 para então explicar a medição em si na Seção 3.2. Por fim, encerrando com a porta T na Seção 3.3 e portas genéricas que estão fora do grupo de Clifford, e exigem uma alteração extra além do que a tabela comporta.

3.1 APLICAÇÕES DAS PORTAS

Para cada porta do grupo de Clifford, existe uma série de operações que, quando realizadas na tabela de estabilizadores, retorna a tabela correspondente ao novo estado obtido pela aplicação daquela porta no estado anterior.

As alterações são baseadas no seguinte fato, dado um operador B que estabiliza um estado $|\psi\rangle$, caso seja aplicada uma operação U no estado, o novo estado resultante será estabilizado pela operação $U \cdot B \cdot U^\dagger$. Isso vem da ideia de que se $B|\psi\rangle = |\psi\rangle$ então $U \cdot B \cdot U^\dagger \cdot (U|\psi\rangle) = U|\psi\rangle$. devido a identidade $U \cdot U^\dagger = U^\dagger \cdot U = I$ para toda operação unitária.

Daqui em diante, uma porta é dita “ativa” caso a linha e coluna correspondente

a ela esteja com o bit 1, e "inativa" caso esteja com bit 0. E o ato de "trocar" o valor significa negar o bit.

Para os próximos exemplos dos efeitos das portas, será usado a Tabela 2 abaixo. As colunas Z_1, Z_2, e Z_3, representam as portas Z aplicadas aquele qubit em específico, e as portas X_1, X_2 e X_3 representam a porta X aplicada aquele qubit. As linhas começando com D representam cada um dos desestabilizadores, as linhas começando com S representam cada um dos estabilizadores.

Se uma célula da linha S2, coluna Z_2 esta com o valor 1, isso significa que um dos estabilizadoras desse estado contém uma porta Z aplicada ao qubit 2. Cada linha S representa um dos elementos do conjunto gerador de estabilizadores daquele estado, e qualquer combinação linear desses elementos também irá gerar um operador estabilizador.

	X_1	X_2	X_3	Z_1	Z_2	Z_3	F
D1	1	0	0	0	0	0	0
D2	0	1	0	0	0	0	0
D3	0	0	1	0	0	0	0
S1	0	0	0	1	0	0	0
S2	0	0	0	0	1	0	0
S3	0	0	0	0	0	1	0

Tabela 2 – Exemplo de tabela na base computacional inicial $|000\rangle$. D1 é XII, D2 é IXI, D3 é IIX. S1 é ZII, S2 é IZI, S3 é IIZ.

Nesse caso, a Tabela 2 está indicando que os estabilizadores do estado $|000\rangle$ são IIZ, IZI, e ZII, algo que pode ser facilmente checado pelo fato de que a porta Z aplicada em um qubit $|0\rangle$ retorna ele mesmo. Da mesma forma, os desestabilizadores são IIX, IXI, XII.

3.1.1 Porta Hadamard

As operações correspondentes a Hadamard aplicada num qubit k na Tabela são: Para todo j entre 1 e $2n$:

1. O sinal da linha j é trocado caso essa linha tenha tanto a porta X quanto a Z ativas na coluna correspondente ao qubit k. Isso acontece, pois, se um estado é estabilizado por Y, quando se aplica H, ele será estabilizado por $-Y$, devido a igualdade $H \cdot Y \cdot H = -Y$.

2. O valor da porta X e Z do qubit k são trocados de lugar. Caso a porta Y esteja ativa (ou seja, tanto X e Z), não há efeito. Novamente, isso acontece pela equivalência $H \cdot X \cdot H = Z$ e $H \cdot Z \cdot H = X$.

A intuição por trás dessa transformação é que se um qubit antes era estabilizado pela porta Z ou $-Z$, o valor dele seria $|0\rangle$ ou $|1\rangle$, respectivamente, ambos estados sem

superposição. Após aplicada a porta, ambos esses casos irão se transformar em estados com superposição, que são agora estabilizados pelo X e $-X$, pois $X \cdot (|0\rangle + |1\rangle) = (|1\rangle + |0\rangle)$.

Como exemplo, podemos aplicar a porta Hadamard no primeiro qubit da base computacional. O novo estado passará de $H_1 |000\rangle = \frac{1}{\sqrt{2}} \cdot (|000\rangle + |100\rangle)$. A Tabela 3 abaixo mostra como isso impacta a representação.

	X_1	X_2	X_3	Z_1	Z_2	Z_3	F
D1	0	0	0	1	0	0	0
D2	0	1	0	0	0	0	0
D3	0	0	1	0	0	0	0
S1	1	0	0	0	0	0	0
S2	0	0	0	0	1	0	0
S3	0	0	0	0	0	1	0

Tabela 3 – Bits da coluna X_1 e Z_1 são trocados de lugar, efeito apenas perceptível no D1 e S1. Estado atual $\frac{1}{\sqrt{2}}(|000\rangle + |100\rangle)$

3.1.2 Porta Phase

Quando a porta Phase é aplicada sob um qubit k, para todo j entre 1 e $2n$:

1. O sinal da linha j é trocado caso a linha tenha tanto a porta X quanto a Z ativas para a coluna do qubit k. Isso acontece porque $SYS^\dagger = -X$

2. A porta Z do qubit k é trocada caso a porta X esteja ativa. Isso acontece porque $SXS^\dagger = Y$.

Caso o Y estabilize o estado $\frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$, após ser aplicada S, a porta $-X$ será a nova estabilizadora para o estado $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Na tabela, será aplicado Phase no primeiro qubit estabilizado anteriormente por X, como pode ser visto na Tabela 4. O novo estado agora é estabilizado pela porta Y, isso pode ser melhor visualizado dessa forma:

$$\frac{1}{\sqrt{2}} Y_1(|000\rangle + i\,|100\rangle) = \frac{1}{\sqrt{2}}(i\,|100\rangle + i(-i\,|000\rangle)) = \frac{1}{\sqrt{2}}(i\,|100\rangle + |000\rangle) \tag{15}$$

	X_1	X_2	X_3	Z_1	Z_2	Z_3	F
D1	0	0	0	1	0	0	0
D2	0	1	0	0	0	0	0
D3	0	0	1	0	0	0	0
S1	1	0	0	1	0	0	0
S2	0	0	0	0	1	0	0
S3	0	0	0	0	0	1	0

Tabela 4 – A porta Z do qubit 1 na linha S1 é ativada. Estado atual $\frac{1}{\sqrt{2}}(|000\rangle + i\,|100\rangle)$

3.1.3 Porta CNOT

Quando a porta CNOT é aplicada, usando k como controle, e h alvo. Para todo j entre 1 e $2n$ é feito:

1. Troque o sinal da linha j caso a porta X_h e Z_k estejam iguais (ligadas ou desligadas), e as portas X_k e Z_h estejam ligadas.

2. Troque a ativação da porta X_h caso a porta X_k esteja ativa. Se o qubit controle está em superposição, essa superposição será passada para o alvo.

3. Troque a ativação da porta Z_k caso a porta Z_h esteja ativa. O qubit alvo passa a Z para o controle, isso acontece devido ao fato de que apesar de uma superposição $|0\rangle + |1\rangle$ não ser estabilizada por Z, a superposição $|00\rangle + |11\rangle$ é, pois ZZ aplicado ao $|11\rangle$ multiplica o sinal do estado por -1 duas vezes, cancelando.

Resumidamente, troque o sinal daquele estabilizador caso entre os dois qubits sendo influenciados, o estabilizador deles for YY ou ZX, no primeiro caso, $-YY$ estabiliza qubits da forma $|00\rangle + |11\rangle$, e o ZX estabiliza os da forma $|00\rangle + |10\rangle$. Essa regra serve para trocar um desses estabilizadores pelo outro.

Como exemplo, será aplicado CNOT duas vezes, uma do qubit 1 para o qubit 2, representado na Tabela 5, outra do qubit 2 para o 3 representado na Tabela 6. Para facilitar a compreensão, os qubits com os números correspondentes ao alvo e controle foram momentaneamente alterados para h e k.

	X_k	X_h	X_3	Z_k	Z_h	Z_3	F
D1	0	0	0	1	0	0	0
D2	0	1	0	0	0	0	0
D3	0	0	1	0	0	0	0
S1	1	1	0	1	0	0	0
S2	0	0	0	1	1	0	0
S3	0	0	0	0	0	1	0

Tabela 5 – Primeiro exemplo, $CNOT_{12}$, a superposição foi passada para o segundo qubit também, como resultado, agora $Z_1Z_2I_3$ e $X_1X_2I_3$ estabilizam o estado. Estado atual $\frac{1}{\sqrt{2}}(|000\rangle + i\,|110\rangle)$

	X_1	X_k	X_h	Z_1	Z_k	Z_h	F
D1	0	0	0	1	0	0	0
D2	0	1	1	0	0	0	0
D3	0	0	1	0	0	0	0
S1	1	1	1	1	0	0	0
S2	0	0	0	1	1	0	0
S3	0	0	0	0	1	1	0

Tabela 6 – Segundo exemplo, $CNOT_{23}$. Estado atual $\frac{1}{\sqrt{2}}(|000\rangle + i\,|111\rangle)$

3.2 MEDIÇÃO

A operação de Medição é responsável por colapsar o estado quântico e por retornar informação sobre o estado dos qubits. Antes de mostrar o algoritmo dessa operação, é necessário definir duas novas sub rotinas que serão chamadas pela operação de medida.

3.2.1 Função Rowsum e função g

Quando duas das matrizes de Pauli são multiplicadas, o resultado é sempre uma outra matriz de Pauli multiplicada por um fator (Nielsen; Chuang, 2011). O objetivo da função g() é retornar o fator esperado, em relação a potências de i.

Observando as seguintes multiplicações $X \cdot X = 1 \cdot I$, $X \cdot Y = i \cdot Z$, e $Z \cdot Y = -i \cdot X$. Nesses casos, o fator que eleva o i em cada equação é, respectivamente, 0, 1, e -1. Esses são os valores esperados da função g() quando as matrizes da esquerda da equação forem passadas como parâmetros.

Elaborando as contas das multiplicações de matrizes, esse seria o resultado esperado em cada um dos seguintes casos, $g(X, Y)$ e $g(X, Z)$:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix} = i \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = -i \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{16}$$

Ou seja, no primeiro caso, a multiplicação de matriz retornou $i^1 \cdot Z$, o termo que eleve o i é 1, logo o retorno da função g(X, Y) é definido como 1. No segundo caso tem-se $i^{-1} \cdot Y$, logo, $g(X, Z) = -1$.

Na prática, não é necessário fazer a multiplicação das matrizes para saber qual será a fase, é possível resumir o cálculo a partir da Tabela 7 abaixo. As linhas representam a primeira matriz da função, e as colunas a segunda matriz.

	I	X	Z	Y
I	0	0	0	0
X	0	0	-1	1
Z	0	1	0	-1
Y	0	-1	1	0

Tabela 7 – Representação das entradas e saidas da função g.

Na definição da subrotina Rowsum, é necessário usar o g para calcular o valor do sinal. Quando Rowsum é aplicado no Tableau, ela recebe duas linhas h, j e as altera seguindo algumas regras. O primeiro passo do algoritmo é realizar a seguinte soma:

$$f(h,j) = 2r_h + 2r_j + \sum_{k=1}^{n} g(P_{jk}, P_{hk}) \tag{17}$$

Sendo r_h e r_j, os bits de sinal correspondentes a linha h e j, respectivamente. P_{jk} e P_{hk} as matrizes de Pauli correspondentes com as linhas h e j, e com cada qubit k da linha. Se essa equação retornar um valor que seja 0 mod 4, então o sinal de r_h é alterado para 0, caso contrário, é alterado para 1.

Depois disso, para cada qubit k, e para cada linha de 1 a n, é trocado a ativação da X_{hk} caso o X_{jk} esteja ativa, e o mesmo para a ativação da Z_{hk} caso a Z_{jk} esteja ativa. Como essas alterações acontecem nas linhas dos desestabilizadores, elas não afetam o estado original. Essas alterações servirão na medição para calcular como a tabela fica após um qubit ser medido.

Como exemplo, caso a função rowsum fosse chamada para as linhas com estabilizadores $+XZZ$ e $-IXX$, em um sistema com 3 qubits. o resultado da soma seria $2 \cdot 0 + 2 \cdot 1 + g(I, X) + g(X, Z) + g(X, Z) = 0$. Com esse valor, o sinal de r_h continua como 0. A soma em si é feita na forma de multiplicação de matrizes em que o sinal não importa, pois ele já foi previamente calculado, então $+XZZ$ somado com $-IXX$ resulta em $+XYY$, isso acontece porque $XZ = Y$ ignorando a fase imaginária que já foi acontada na função do sinal.

3.2.2 Cálculo do Valor

Por fim, a ação de medir o sistema, que pode causar o colapso de um estado quântico. Esse colapso é representado da seguinte forma:

Dado uma operação de medição no qubit k. faça o seguinte:

1. Veja se nos estabilizadores existe alguma linha P em que a coluna k esteja com a porta X ativa. Se existir mais que um, use o menor P.

2. Caso exista um P, a medição é aleatória.

3. Caso não exista um P, a medição é determinística.

A lógica por trás do P, é que se nos estabilizadores existe uma porta X, significa que de alguma forma, aquele qubit está em superposição, pois os estados estabilizados por X são aqueles na forma $|0\rangle + |1\rangle$, as vezes sendo necessário também uma mudança de sinal por uma porta Z. Será usado como exemplo uma medição do primeiro qubit da Tabela 6, nesse caso, o P vale 1

Caso exista o P, são feitos os seguintes passos:

1. A função Rowsum é chamada com parametros j e P, para todas as linhas j da tabela que tiverem a porta X ativa para o qubit k, exceto a própria P.

2. O desestabilizador correspondente a linha P é substituído pelo estabilizador P.

3. O estabilizador P volta a ser o valor original dele, em que todos os bits são 0, exceto o da porta Z.

4. O sinal do estabilizador P vai ser 0 ou 1, dependendo do valor que foi medido, que no caso é determinado aleatoriamente com probabilidade igual para ambos os valores. Esse valor corresponde ao valor medido.

Exemplo de medição do qubit 1. Assumindo que depois de medir, o retorno é 1. Para esse caso, o P vale 4, pois a linha 4 é o primeiro estabilizador que possui X no qubit sendo medido. Como nenhuma outra linha possui X no primeiro qubit, a rowsum não é chamada, mas caso fosse, a linha 4 seria alterada somando as linhas que tivessem X.

	X_1	X_2	X_3	Z_1	Z_2	Z_3	F
D1	1	1	1	1	0	0	0
D2	0	1	1	0	0	0	0
D3	0	0	1	0	0	0	0
S1	0	0	0	1	0	0	1
S2	0	0	0	1	1	0	0
S3	0	0	0	0	1	1	0

Tabela 8 – Pós-medição da Tabela 6. Assumindo que o primeiro qubit foi medido como 1. Estado atual $i\,|111\rangle$

A intuição por trás dessas operações é que originalmente, a linha indicava o estabilizador de um qubit com superposição, portanto a porta X estava ativa, depois de feito a medida, ele poderá apenas ser 0 ou 1, e ambos esses estados podem ser estabilizados por Z ou $-Z$, respectivamente.

Logo o desestabilizador do estado irá virar X, e o estabilizador volta a ser Z.

Caso não exista o P, a medida é determinística, e os seguintes cálculos são feitos para determinar qual o valor:

1. É criado uma linha auxiliar na tabela, posição $2n + 1$, ela é inicialmente definida como tendo todas as posições zeradas.

2. É chamado $Rowsum(2n + 1, j)$ sendo j todas as linhas de estabilizadores nas quais as respectivas linhas de desestabilizadores possuem a porta X ativa para aquele qubit. Ou seja, aquele estabilizador tem influência no qubit medido.

3. O valor do sinal da linha $2n + 1$ é o valor do qubit.

Por exemplo, caso os estabilizadores de um estado fossem $-ZZZ$, $+IZZ$ e $-ZIZ$, e os desestabilizadores 1 e 3 tivessem X no qubit relevante. Quando a função de medição fosse chamada para o qubit 2, o resultado seria que na linha extra, seria somado o primeiro e terceiro estabilizador resultando em $+IZI$, o sinal indicaria que o qubit está no estado $|0\rangle$.

Em ambos os casos, o valor da medição é retornado, e a tabela continua com os estabilizadores correspondentes àquele estado quântico.

3.3 PORTAS T E PORTAS U

Todas as operações até agora só se aplicam dentro do grupo de Clifford, para aplicar portas que não estão no grupo, é preciso fazer um algoritmo diferente. Como a tabela não tem a capacidade de mostrar estabilizadores não Cliffordianos, quando uma porta externa é aplicada, é necessário criar mais tabelas. Essa é uma prática que não foi encontrada na literatura, e foi necessário desenvolver novos métodos para manipular e medir sistemas com múltiplas tabelas.

Para mostrar o efeito na prática, será aplicado uma porta T no qubit 1 da Tabela 6, como ela está em superposição, será possível ver o efeito resultante.

Uma porta T pode ser decomposta como a soma ponderada de portas de Pauli, mais especificamente, é possível separar essa porta da seguinte forma na Equação (18).

$$T = \frac{1}{2}((1+\sqrt{i})I + (1-\sqrt{i})Z) \tag{18}$$

Dessa forma, quando a porta T é aplicada no qubit 1 do estado $\frac{1}{\sqrt{2}}(|000\rangle + i\,|111\rangle)$ o resultado pode ser expresso com a Equação (19).

$$\frac{1}{2}((1+\sqrt{i})I + (1-\sqrt{i})Z)\frac{1}{\sqrt{2}}(|000\rangle + i\,|111\rangle) = \frac{1}{\sqrt{2}}(|000\rangle + i\sqrt{i}\,|111\rangle) \tag{19}$$

Da mesma forma como é possível simular o estado quântico $|+\rangle$ por meio da simulação dos vetores $|0\rangle$ e $|1\rangle$ separadamente. Também é possível simular a alteração de uma porta T seguindo essa decomposição, duplicando o estado original e aplicando a porta I e a porta Z separadamente em cada um, com seus respectivos fatores. Desde que, no fim, as amplitudes sejam somadas para que os estados possam interferir construtivamente ou destrutivamente antes de ser feito a medida.

Dentro do Tableau, isso é expresso criando duas tabelas separadas para aquele estado, uma na qual foi aplicada a identidade I e outra tabela em que foi aplicado Z. A primeira tabela terá uma amplitude global de $(1+\sqrt{i})$, e a segunda terá amplitude $(1-\sqrt{i})$. A amplitude da tabela nesse caso se refere a um fator que está multiplicando todos os estados da superposição, basicamente um fator em comum que é colocado em evidência.

As alterações da porta T podem ser vistas no sistema composto das Tabelas 9 e 10 abaixo.

Essa separação apenas acontece quando o estado representado por aquela tabela está em superposição. Caso a tabela não esteja, é necessário verificar o valor estado dela, se o estado for $|0\rangle$, a porta T não afeta o valor, caso o estado seja $|1\rangle$, essa tabela tem sua amplitude multiplicada por $\sqrt{i}$.

	X_1	X_2	X_3	Z_1	Z_2	Z_3	F
D1	0	0	0	1	0	0	0
D2	0	1	1	0	0	0	0
D3	0	0	1	0	0	0	0
S1	1	1	1	1	0	0	0
S2	0	0	0	1	1	0	0
S3	0	0	0	0	1	1	0

Tabela 9 – Tabela em que foi aplicada I_1. Estado atual $\frac{1}{\sqrt{2}}(|000\rangle + i\,|111\rangle)$

	X_1	X_2	X_3	Z_1	Z_2	Z_3	F
D1	0	0	0	1	0	0	0
D2	0	1	1	0	0	0	0
D3	0	0	1	0	0	0	0
S1	1	1	1	1	0	0	1
S2	0	0	0	1	1	0	0
S3	0	0	0	0	1	1	0

Tabela 10 – Tabela que foi aplicada porta Z_1. Estado atual $\frac{1}{\sqrt{2}}(|000\rangle - i\,|111\rangle)$

Todas as operações apresentadas anteriormente se estendem para um sistema com múltiplas tabelas de forma simples. As operações são aplicadas individualmente em cada tabela. Isso inclui outras portas T, dessa forma, é possível que um sistema com 4 tabelas, se torne um sistema com 8 tabelas depois de aplicado uma única porta T.

A função de medida, nesse caso, se torna mais complexa, e será elaborada na Seção 4.3. Após o valor ser obtido, ele é atualizado em todas as tabelas seguindo o mesmo algoritmo de medição apresentado na Seção 3.2.2.

Apesar de ser possível aproximar qualquer porta quântica aplicando diversas portas T, foi decidido criar uma função que permitisse a aplicação de qualquer porta genérica. Como qualquer porta fora do grupo pode resultar em um crescimento exponencial de tabelas, é mais viável para a complexidade que cada porta seja aplicada diretamente ao invés de decomposta em múltiplas portas T.

Dentro do método de Tableau, qualquer porta pode ser decomposta em uma soma ponderada de portas de Pauli, que por fim, podem ser aplicadas de maneira parecida com a das portas T, em que são criadas mais tabelas para cada valor da soma, e a fase da tabela é multiplicada pelo peso daquela operação de Pauli.

Por exemplo, dado uma porta U com uma matriz definida na Equação (20). Essa operação aplica uma transformação, levando o estado $|0\rangle$ para o estado $a\,|0\rangle + c\,|1\rangle$, e o estado $|1\rangle$ para o $b\,|0\rangle + d\,|1\rangle$.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{20}$$

É possível definir uma operação equivalente, utilizando soma de matrizes de Pauli, I, X, Z, e iY, como na Equação (21)

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \frac{a+d}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{c+b}{2}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + \frac{a-d}{2}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} + \frac{c-b}{2}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{21}$$

Ao aplicar esse somatório em um estado, o resultado será um estado equivalente ao de ser aplicado a operação U. Para representar essa soma de matrizes, para cada tabela no sistema, é criado 4 tabelas, cada uma será aplicada uma das matrizes de Pauli, e o fator da tabela é multiplicado pela constante acompanhando a matriz.

4 DESENVOLVIMENTO

Esse capítulo trata da parte prática do trabalho, a Seção 4.1 detalha quais métodos foram utilizados para representar os objetos e operações detalhados nos capitulos anteriores.

Além disso, a Seção 4.2 detalha as principais estratégias que foram utilizadas para diminuir o crescimento do número de tabelas em sistemas não-cliffordianos. Tanto para portas U quanto para portas Toffoli.

Na Seção 4.3 é documentado qual método foi utilizado para realizar a medição em sistemas de múltiplas tabelas, e como que o colapso da superposição.

Por fim, na Seção 4.4 é descritas questões de programação relativas ao trabalho, e na Seção 4.5 a integração do sistema com a plataforma Ket.

4.1 TABELA E TABLEAU

Para simular o comportamento das portas T, foram feitas duas classes, uma para representar cada tabela individualmente, e outra para representar o sistema de múltiplas tabelas formado pelo comportamento das portas não cliffordianas.

Quando uma porta fora do grupo é usada em um qubit no sistema, caso aquele qubit esteja em superposição, as tabelas são duplicadas (ou quadruplicadas), gerando novas versões alternativas em que diferentes portas de Clifford foram aplicadas com diferentes pesos, influenciando na fase da tabela.

Como exemplo, para representar a aplicação da porta T, é possível criar duas tabelas, uma com um fator de amplitude de $\frac{1-\sqrt{i}}{2}$ em que a porta Z foi aplicada, e uma com fator de $\frac{1+\sqrt{i}}{2}$ em que a porta I foi aplicada. Isso acontece apenas caso exista superposição, caso o qubit esteja no estado $|1\rangle$, uma fase de $\sqrt{i}$ é multiplicada no fator da tabela, e para o qubit no estado $|0\rangle$ nada acontece.

Quando o sistema tem mais de uma tabela, todas as operações de Clifford apresentados na Capítulo 3 podem ser aplicadas na classe Tableau, essa classe irá redistribuir a operação, aplicando ela em cada tabela que compõe o sistema.

Portanto, um sistema que inicialmente só possui uma tabela pode gerar, no pior caso, até 4^G tabelas, (sendo G a quantidade de aplicações de portas fora do grupo de Clifford), e cada operação realizada após a aplicação dessas operações serão aplicadas em todas as 4^G tabelas.

A medida é feita em duas partes, a primeira parte faz uma varredura das amplitudes de cada tabela para decidir se o valor medido será $|0\rangle$ ou $|1\rangle$. Isso é, obtendo cada tabela e analisando se o estado dela é $|0\rangle$, $|1\rangle$ ou $|0\rangle \pm |1\rangle$. A segunda etapa, caso haja superposição, é colapsar o estado e propagar o valor obtido para as tabelas abaixo, isso é feito através da função *measurement* da Tabela, que pode se comportar de duas maneiras diferentes.

Caso nenhum valor seja informado como parâmetro, ela irá escolher um valor

aleatório entre 0 e 1, e colapsará para esse estado, retornando o valor. Caso um valor seja informado, ela irá colapsar para esse valor informado. Isso é feito para manter a consistência entre diferentes tabelas, caso o qubit 2 seja colapsado para o estado $|0\rangle$, todas as tabelas terão seu qubit 2 colapsado para esse estado.

Outro detalhe importante na hora de manipular múltiplas tabelas, é que o formalismo do Tableau representa apenas o estado, e não a fase global dele, ou seja, estados que diferem por uma fase global são tratados como sendo o mesmo estado em uma tabela. Por exemplo, A porta Z estabiliza tanto o estado $|0\rangle$ quanto o estado $-|0\rangle$. O estabilizador X pode representar o estado $|0\rangle + |1\rangle$, mas também o estado $e^{i\theta}(|0\rangle + |1\rangle)$, para qualquer valor real de θ.

Isso acontece pois a informação da fase global não interfere na probabilidade do estado, quaisquer dois estados que possuem apenas uma fase global de diferença irão se comportar da mesma forma diante de qualquer combinação de portas. Sendo assim não existe diferença prática entre esses estados.

O problema acontece na hora de representar um sistema com múltiplas tabelas, nesse caso, cada tabela individual está ignorando sua fase global, no entanto, como existem várias tabelas, a fase global de um tabela na verdade representa uma fase relativa para as outras tabelas, algo que pode interferir na medição. Portanto, tratar as duas tabelas como sistemas isolados não funciona, por isso é necessário que cada tabela mantenha registro da sua fase global.

O formalismo de Tableau original não leva isso em conta, para adaptar o sistema, foi necessário adicionar algumas informações a mais nas operações para que a fase global fosse sempre levada em conta. Por exemplo, ao aplicar a porta S_h em um sistema em que todos os estados tem o qubit h no estado $|1\rangle$, originalmente, nada era alterado, pois essa fase global de i não afetava a probabilidade. Agora, no código, quando essa porta é aplicada, a tabela afetada possui sua fase global multiplicada por i.

Outro momento em que a fase global se torna relevante, é nos estados da forma $\frac{1}{2}(|00\rangle - |01\rangle - |10\rangle - |11\rangle)$. Esse estado é estabilizado pelas operações $-XZ$ e $-ZX$, ao aplicar Hadamard no primeiro qubit, isso resulta no estado $|10\rangle - |01\rangle$. No entanto, ao aplicar Hadamard no segundo qubit, resulta em $-|10\rangle + |01\rangle$, esses dois estados tem os mesmos estabilizadores, pois apenas diferem por uma fase de -1.

Casos como esse geralmente acontecem quando há estabilizadores com sinais negativos. Para resolver isso, foi decidido que os estados seriam normalizados em relação ao estado com menor valor binário, e sempre que uma fase estivesse presente no estado de menor valor, um fator seria multiplicado na fase da tabela. No caso anterior $|10\rangle - |01\rangle$ possui um sinal negativo no menor valor $|01\rangle$, o Tableau representaria esse estado com a fase trocada, logo, para manter a conversão, uma fase -1 deverá ser multiplicado na respectiva tabela.

4.2 OTIMIZAÇÕES

O Método de Tableau funciona especificamente para tornar polinomial o tempo de aplicação das portas de Clifford. No entanto, para permitir que esse módulo de simulação seja mais abrangente para diferentes tipos de circuitos, foi introduzido a possibilidade de executar outras portas fora do grupo, essa ação tem um custo exponencial. Essa seção serve para definir todas as formas de otimização utilizadas para reduzir a complexidade dessas portas.

4.2.1 Dedução de estado

Não é possível saber se um qubit tem valor $|0\rangle$ ou $|1\rangle$ olhando diretamente o Tableau, apenas se ele está ou não em superposição. Para aplicar as portas alternativas como a T, é necessário saber se o qubit pode estar no valor $|1\rangle$, para calcular esse valor, a operação rowsum pode ser chamada para todas as linhas da tabela, e o resultado final dessa operação mostra o valor.

No entanto, como essa operação é custosa, foi criado um atalho dentro da tabela. Uma lista de valores para todos os qubits da tabela, indicando se ele esta com o valor 0, valor 1, ou valor 2, indicando superposição. Com essa lista, o valor pode ser consultado em tempo constante sempre que necessário aplicar alguma porta fora do grupo de Clifford, ou outra operação que necessite saber o valor.

As operações de Clifford foram adaptadas para atualizar essa lista após execução. Se um qubit está como $|0\rangle$ e foi aplicado a porta X, a lista será atualizada para $|1\rangle$. Caso um $CNOT$ ou H seja aplicada em um qubit, os estabilizadores resultantes são checados para ver se o qubit afetado está em superposição, ou seja, possui um X nos estabilizadores, se sim, a lista é atualizada para 2.

Essa lista não possui informação de entrelaçamento, então não é possível saber quantos estados existem ou quais qubits estão entrelaçados com quais, apenas se o qubit está ou não em superposição, e caso não esteja, qual o valor dele.

4.2.2 Otimizações da Porta U

Visando diminuir ao máximo a quantidade de tabelas extras criadas por operações não cliffordianas, algumas otimizações foram feitas para detectar quando é desnecessário criar tabelas. Para casos em que o qubit alvo está em superposição, é necessário representar todas as opções, mas para casos em que o valor do qubit é exato, é possível aplicar diretamente a operação na tabela.

Por exemplo, caso uma porta T seja aplicada em um qubit de uma tabela com estado $|0\rangle$, ao invés de criar uma tabela nova, essa operação pode ser ignorada, pois a porta T aplicada no $|0\rangle$ não tem efeito, e caso seja aplicada em um qubit $|1\rangle$, o efeito pode ser resumido como um fator de $\sqrt{i}$ multiplicando a amplitude daquela tabela específica.

Caso a porta U aplicada leve um estado sem superposição para um com superposição, e o valor do qubit seja determinístico, é possível limitar o crescimento para apenas uma tabela a mais. Usando a matriz da Equação (21), um qubit no estado $|0\rangle$ resultaria em duas tabelas, a original seria multiplicada por um fator de a, e a nova tabela teria seu valor trocado para $|1\rangle$ e aplicado uma amplitude de b. Da mesma forma, com um estado determinístico $|1\rangle$, resultaria em uma superposição de $b\,|0\rangle + d\,|1\rangle$.

4.2.3 Portas Toffoli

No formalismo do Tableau, com portas CNOT, é possível representar o entrelaçamento entre dois qubits, controle e alvo, essa conexão é representada pelos estabilizadores com múltiplas entradas X, esses estabilizadores indicam que para estabilizar um qubit, é necessário estabilizar outro, então estados da forma $|00\rangle + |11\rangle$ precisam ser estabilizados por XX.

No entanto, esse entrelaçamento é sempre feito par a par, portas Toffoli, requerem mais de um qubit de controle para afetar um único alvo, ou seja, entrelaçam múltiplos qubits no resultado de um só. O Tableau não consegue representar esse comportamento, isso acontece pois portas Toffoli não estão contidas no grupo de Clifford. Portanto, é necessário mais uma vez criar múltiplas tabelas para representar um único sistema.

Em circuitos quânticos reais, é comum decompor a porta Toffoli em portas menores, usando 7 portas T, isso acontece pois em arquiteturas de computadores quânticos reais, portas Toffoli não são possíveis de implementar devido a restrição física na interação entre qubits. Na parte da simulação quântica, não existe essa restrição, logo é possível tomar atalhos para deixar a execução mais eficiente.

Na tentativa de diminuir o número de ramos diferentes de tabelas, foi pensado a seguinte estratégia. Dado um estado $|+++\,+0\rangle$, em que $|+\rangle$ representa um qubit em superposição $|0\rangle + |1\rangle$. A aplicação de uma Toffoli que usa os 4 primeiros qubits como controle e o último como alvo, resultaria em uma superposição de todas as combinações em que pelo menos um qubit está no estado $|0\rangle$, e nenhuma alteração aconteceu. E um único estado em que todos os qubits estão como $|1\rangle$, e a porta controlada X foi aplicada no último qubit.

Vale notar que antes de qualquer ramificação ser feita, é checado se não existe algum qubit da lista de qubits de controle com valor $|0\rangle$, pois isso invalidaria completamente a execução da porta, que necessita todos os qubits ativos para funcionar. Caso esse qubit seja encontrado, todos os processos de ramificação não são executados, e o efeito da porta é ignorado.

No caso em que a porta pode ser aplicada. Para evitar representar todas as combinações possíveis de estados, é possível tomar o seguinte atalho: O estado irá se separar na soma de dois estados distintos, um que o primeiro qubit é $|0\rangle$, outro em que o primeiro qubit é $|1\rangle$, para o estado em que o qubit é $|0\rangle$. Como exemplo, no caso anterior, o primeiro

qubit poderia ser separado em duas tabelas $|0+++0\rangle + |1+++0\rangle$, para o primeiro estado nada mais precisa ser feito, aquela superposição não é afetada pela Toffoli pois um dos qubits de controle está como 0. Já o segundo estado representa uma superposição na qual o estado $|11110\rangle$ está contido, e esse é afetado pela Toffoli, portanto, fazemos a mesma divisão para o segundo qubit.

Esses passos fariam com que o estado de exemplo com 4 controles se transformasse em 5 tabelas, cada uma codificando uma parte do estado resultante: $|0+++0\rangle + |10++0\rangle + |110+0\rangle + |11100\rangle + |11111\rangle$, analisando as somas, é possível notar que todos os estados anteriormente representados por $|++++0\rangle$ estão contemplados uma única vez. Além disso, a única combinação em que o último qubit foi afetado é aquela em que todos os qubits de controle são $|1\rangle$. Essa operação cria C novas tabelas, sendo C a quantidade de qubits de controle.

Durante o desenvolvimento das operações, outra otimização ainda mais eficiente foi descoberta, uma estratégia que cria um estado novo com o único propósito de servir como cancelamento. Todo o propósito dessa simulação da Toffoli é garantir que o estado em que todos os controles estão ativos seja afetada pela operação, enquanto os outros estados não. Uma forma de garantir isso é criar uma nova tabela, resultante de todos os qubits de controle serem medidos como $|1\rangle$ onde a operação foi aplicada, como feito anteriormente e manter a tabela original em superposição intacta.

O problema é que a tabela original continua representando o estado em que todos os qubits de controle estão ativos, mas que a porta da operação não foi aplicada. Para cancelar a influência da tabela principal, é possível adicionar uma nova tabela, que também representa o estado com todos os qubits ativos, mas em que a porta não foi ativada, e dar a essa tabela uma amplitude oposta a da tabela principal.

No exemplo anterior, isso resultaria em um estado representado por $|++++0\rangle - \frac{1}{4}|11110\rangle + \frac{1}{4}|11111\rangle$. Realizando os cálculos dos vetores é possível notar que o estado extra criado se cancela com um dos estados dentro da superposição, resultando na representação correta do que se espera para a aplicação da Toffoli.

A intuição por trás desse método pode ser melhor entendida através de uma analogia com o comportamento da porta H, quando ela é aplicada no estado $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ o resultado, após os devidos cancelamentos, é $|1\rangle$, no entanto, o resultado também pode ser expresso sem realizar os cancelamentos, na forma de $\frac{1}{2}(|0\rangle + |1\rangle) - \frac{1}{2}(|0\rangle - |1\rangle)$. Essas duas formas são válidas para representar o mesmo estado, desde que os devidos cancelamentos sejam feitos na hora da medição, não existe diferença nessas duas representações. O mesmo acontece com a aplicação da Toffoli no outro exemplo.

Dessa forma, não é necessário desmanchar a tabela principal para extrair cada superposição e combinação, basta continuar simulando o sistema com essas 3 tabelas, e todas as operações deverão se comportar da mesma forma que o sistema se comportaria em um computador quântico real. Todos os efeitos causados pelo estado da tabela principal

serão devidamente cancelados pela tabela extra de sinal oposto.

Esse método traz uma vantagem comparada a outras formas de simular portas Toffoli, por não ser necessário decompor a porta em portas menores. Portanto esse método é extremamente escalável, tanto para uma operação de Toffoli com 2 qubits de controle ou 25 qubits de controle, resultam no mesmo crescimento, em que uma tabela pode, no pior caso, gerar mais 2 tabelas, resultando na triplicação da quantidade de tabelas.

Analisando a execução de múltiplas portas Toffoli, o crescimento das tabelas tem o seguinte comportamento: Depois de aplicado uma Toffoli, uma tabela com superposição pode se tornar 3 tabelas, a original com superposição, e duas tabelas em que todos os qubits de controle estão ativos.

Caso uma segunda porta Toffoli seja aplicada com outros controles, as duas tabelas sem superposição geradas vão se comportar de forma linear e determinística, a operação será aplicada nelas sem que seja necessário criar mais tabelas. Enquanto a tabela original em superposição irá criar mais duas tabelas. Portanto, múltiplas aplicações diretas tem um crescimento linear.

Em algoritmos quânticos, como o de Grover, uma Toffoli é aplicada de forma intercalada com Hadamards em todos os qubits. Nesse caso, o comportamento seria diferente, em que após aplicado a primeira Toffoli, as Hadamards transformariam as duas tabelas com qubits ativos em tabelas de superposição, ao mesmo tempo transformando a tabela original de superposição em uma tabela sem superposição. Quando a próxima Toffoli fosse aplicada, as duas tabelas de superposição irão gerar outras 4 tabelas, enquanto a tabela sem superposição irá se comportar de forma determinística. Logo para cada aplicação intercalada de Toffoli e Hadamards se espera que a quantidade de tabelas dobre.

4.3 MEDIÇÃO

Originalmente, em sistemas de uma única tabela, a medição era feita através do algoritmo de rowsum, que somava as linhas dos estabilizadores e calculava se o qubit estava no estado $|0\rangle$, $|1\rangle$, ou em uma superposição. No caso de uma superposição, um valor seria escolhido aleatoriamente com igual probabilidade, e a tabela seria alterada de acordo. Essa função foi criada para o caso em que apenas portas de Clifford são usadas no circuito.

Para realizar a medição em sistemas com múltiplas tabelas, diversas alterações foram necessárias. A principal delas, foi a criação de uma nova função de medição, que roda em tempo exponencial em relação a quantidade de qubits. Pois não é possível medir o valor das tabelas separadamente, e decidir entre elas baseado na amplitude. Existem interações destrutivas e construtivas que precisam ser consideradas quando se faz a medida que não são representadas usando essa estratégia.

Por exemplo, se uma tabela esta no estado $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ e outra tabela esta no estado $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, caso a medida seja feita em cada tabela separadamente, é possível

que o estado $|1\rangle$ seja medido, mas devido a interferência de sinais opostos, esse sistema de duas tabelas representa apenas o estado $|0\rangle$.

Por um motivo semelhante, não é possível medir um qubit de forma isolada, apenas calculando a interferência individual. Por exemplo, caso uma tabela tenha o estado $\frac{1}{\sqrt{2}}(|00\rangle+|10\rangle)$ e outra tabela tenha o estado $\frac{1}{\sqrt{2}}(|00\rangle-|11\rangle)$. Para medir o valor do primeiro qubit, ignorar o sistema como um todo resultaria em $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ da primeira tabela, e $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ da segunda, novamente resultando em um estado com $|0\rangle$. No entanto os estados $|10\rangle$ e $-|11\rangle$ não deveriam se cancelar, e a real probabilidade é de 50% para medir o primeiro qubit como $|0\rangle$. Isso só é possível notar visualizando o estado completo.

Devido a esses fatores, o jeito encontrado para medir o qubit corretamente foi extrair os vetores estado originais de cada tabela, somando e calculando a probabilidade a partir deles. Isso resulta em um vetor de tamanho exponencial ao número de qubits, mas que corretamente soma o efeito dos estados. Após extrair o vetor de todas as tabelas, basta fazer o cálculo de valor esperado para o qubit que se deseja medir.

A extração do vetor estado original de cada tabela é feita a partir dos n estabilizadores. Para cada estabilizador S_i, é calculado o operador $\frac{I+S_i}{2^n}$. Ao multiplicar os n operadores entre si, o resultado é a matriz densidade daquele estado. Essa matriz pode então ser usada para recriar o estado original, multiplicando ela por algum estado base que não seja ortogonal a matriz. Nesse caso, ortogonal significa que a multiplicação dessa matriz pelo vetor resultaria em um vetor apenas com 0.

Por exemplo, para o estado $|01\rangle - |10\rangle$, os estabilizadores são $-XX$ e $-ZZ$, os operadores resultantes dessas matrizes estão representados na Equação (22) e na Equação (23). A multiplicação delas resulta na Equação (24).

$$\frac{1}{4}(II - XX) = \frac{1}{4}\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \tag{22}$$

$$\frac{1}{4}(II - ZZ) = \frac{1}{4}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{23}$$

$$\frac{1}{4}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{24}$$

Quando a matriz densidade multiplica um estado não ortogonal a ela, nesse caso

$|01\rangle$, o resultado é o vetor estado representado pelos estabilizadores. Nesse caso, quando a matriz multiplica o estado $|10\rangle$, o resultado é o mesmo estado com uma fase global negativa. Para motivos de padronização, nessa implementação, o estado base é sempre o que tiver o menor número binário.

Depois que todos os vetores estados de cada tabela forem extraídos, a classe que controla o sistema pode realizar o cálculo do valor esperado, obtendo o valor absoluto ao quadrado de cada posição do vetor. Depois que a medição é feita, o valor medido é propagado para todas as tabelas, para que elas possam atualizar o valor daquele qubit, repassando o colapso das superposições.

Uma função extra foi criada, chamada dump, ela realiza as mesmas operações de recuperação de estado da medição, mas o retorno dela é um vetor contendo a amplitude de todos os estados do sistema, não apenas o do qubit desejado, essa operação não colapsa o estado. É uma forma mais eficiente de saber o estado do sistema final.

4.4 TABELA E OPERADORES DE CLIFFORD

Descrevendo melhor a parte da programação relativa ao trabalho. Inicialmente, foi criada uma classe para representar o objeto Tabela, com uma variável para a quantidade de qubits e uma matriz representando a própria tabela. Cada linha representando um desestabilizador ou estabilizador, contendo uma lista com os valores das portas para cada qubit. Nessa tabela também foi adicionado, no último índice, um número complexo para representar a amplitude da tabela.

Essa classe foi encapsulada em outra classe chamada Tableau, essa classe representa o sistema como um todo, podendo conter múltiplas tabelas dentro dela. Esse encapsulamento foi feito para facilitar a aplicação dos operadores nas múltiplas tabelas que poderiam existir no sistema.

Para cada operador de Clifford apresentado na Capítulo 3 foi criada uma função na classe Tabela para executar as alterações correspondentes, e uma função na classe Tableau feita para distribuir a aplicação da porta em todas as sub tabelas, e com implementações para portas não cliffordianas, além de cuidar de algumas otimizações extras.

No construtor da Tabela, todos os valores da matriz são marcados como 0 exceto a diagonal principal, que é marcada como 1, simbolizando os desestabilizadores e estabilizadores esperados para o estado inicial $|000...0\rangle$. A tabela tem tamanho $2n$ por $2n + 1$, sendo n a quantidade de qubits. O $2n$ é para os n desestabilizadores e os n estabilizadores. A coluna extra serve para guardar o sinal de cada linha.

Por motivos de depuração, foi criada uma função *string* da classe Tabela, dessa forma, sempre que a função *print* for chamada para esse objeto, o Python irá interpretar como uma string contendo os dados da tabela. Especificamente, foi feito para que os bits da tabela de cada estabilizador fossem lidos, e transformados em uma *string*, retornando todos os estabilizadores em uma forma legível, e com a fase global daquela tabela. Por exemplo,

caso a Tabela 6 fosse transformada em texto dentro do código, a *string* correspondente seria "Phase: 1; +YXX/+ZZI/+IZZ", que indica o conjunto gerador do grupo de estabilizadores do estado e a fase global.

Foram feitos testes com duas formas de representação da tabela. A primeira como uma matriz de valores booleanos, seguindo diretamente a ideia de representação proposta no artigo do Aaronson. A segunda implementação foi feita com uma lista de strings, cada *string* representando um desestabilizador ou estabilizador do estado. Para realizar as operações de Clifford, foram criadas funções que recebem o qubit que se deseja afetar. A matriz de booleanos é alterada de acordo com as especificações do artigo.

A ideia por trás da representação com *strings* era para evitar a complexidade espacial das listas dentro de listas, além de compactar os estabilizadores em um texto. No final essa ideia foi descartada devido ao fato de *strings* serem imutáveis no python, levando a um custo linear para qualquer operação que alterasse o estado, Comparado ao custo constante com matrizes.

4.5 INTEGRAÇÃO COM KET

O novo módulo foi introduzido como uma subclasse do método *LiveExecution*, que encapsula e padroniza as funções para execução das operações do circuito. Para a adaptação funcionar, foi necessário apenas definir quais funções do Tableau seriam chamadas dentro dos métodos do *LiveExecution*, adaptando os tipos de parâmetros entre qubits e listas de qubits quando necessário.

Para algumas funções como a de Phase, que é sempre definida como uma rotação no eixo Z por um dado ângulo θ, foi necessário obter a fase a partir desse ângulo usando a fórmula $e^{i\theta}$, para ângulos bem definidos como 90 e 180, as portas S e Z são chamadas, respectivamente, e para as outras, é necessário decompor em portas menores usando a decomposição de U.

As rotações em X e Y por um ângulo θ, são calculadas da mesma forma para criar as matrizes resultantes, caso essas rotações sejam as de 180 graus, elas podem ser representadas diretamente pelas operações de Clifford, em qualquer outro caso, a função que aplica operações U é chamada.

Para as operações controladas, é checado se elas podem ser decompostas apenas em $CNOT$, caso a operação tenha mais de um qubit de controle, ela é considerada uma porta Toffoli, e portanto, precisa ser decomposta usando o método que cria outras 3 tabelas.

A função de Medida pode incluir mais de um qubit. Mas o Tableau comporta apenas a medição de um qubit por vez, nesse caso, a função encapsuladora chama a função medida do Tableau um qubit por vez, e concatena eles multiplicando pelas potências de dois para criar o número decimal equivalente da representação binária.

Em comparação com o simulador padrão do Ket, a única diferença na forma de usar é que na hora de instanciar um *ket.process*, o parâmetro dele precisa ser uma

variável do tipo *TableauSimulation*. Todas as outras operações no *frontEnd* são feitas igual ao simulador utilizado antes, a única diferença se encontra nas funções executadas no *backEnd*. O simulador ainda não está disponível na versão atual do Ket, mas será liberado para uso nas versões futuras.

Na Figura 2 é mostrado as três classes do sistema e a classe abstratata do Ket que é utilizada como base para implementar o simulador do Tableau. *Table* representa a classe de um tabela, Tableau representa um sistema de múltiplas tabelas, e *TableauSimulation* representa o simulador que é chamado pelo usuário para simular o Tableau.

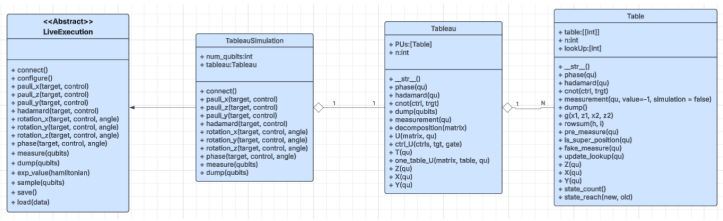


Figura 2 – Diagrama UML detalhando as classes utilizadas no código. Elaborado pelo próprio autor.

5 TESTES E RESULTADOS

CPU	Intel® Xeon™ W-2133
RAM	32 GB
Python	3.12.2
Windows	Windows 10
Ket	0.9.1

Tabela 11 – Configuração do computador usado para os testes.

Para realizar os testes, foi utilizado um computador com as configurações na Tabela 11, alguns circuitos foram executados com diferentes parâmetros de quantidade de portas lógicas e de qubits. A lib Time do Python foi utilizada para medir o tempo de execução da simulação dos circuitos.

Três testes principais foram realizados com o propósito de testar quantos qubits o Tableau conseguiria comportar, quantas operações são possíveis antes que o tempo de execução se torne proibitivo, e como que o Tableau se comporta em circuitos não cliffordianos.

5.1 CÓDIGO DE CORREÇÃO DE ERRO DO SHOR

Foram testados 3 tipos de circuitos, o primeiro foi o circuito de correção de erro com a codificação de Shor (Huang; Brown; Cetina, 2023). Ele foi feito para testar a capacidade do Tableau de simular circuitos puramente de Clifford. Esse tipo de circuito utiliza 9 qubits físicos para representar um único qubit lógico, esse qubit lógico é resistente a erros, pois a redundância dos qubits em sua codificação ajudam a corrigir erros. Algo análogo à estratégia clássica de repetir o mesmo sinal 3 vezes para que a redundância corrija erros de *bit-flip*.

O método de Tableau foi inicialmente desenvolvido com o intuito de auxiliar na representação de circuitos de correção de erro, e portanto, é mais eficiente lidando com circuitos desse formato. No teste, foram utilizados vários qubits lógicos, formados por 9 qubits físicos cada, eles foram preparados fazendo com que os qubits na posição $0, 3, 6$ estivessem em superposição, e entrelaçados com os dois qubits seguintes, criando o estado $(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)$, que é resistente a erros de *bit-flip* e *phase-flip*.

Depois, foram introduzidos erros de *bit-flip* no circuito, aplicando uma X aleatoriamente em um dos 9 qubits pra cada qubit lógico. Logo em seguida, foi aplicado o circuito de correção de erro, que compara os valores dos qubits de cada bloco de 3, para checar se todos os valores estão iguais, e caso não estejam, trocar o valor do qubit que está diferente, esse processo requer 2 qubits auxiliares a mais para registrar a medição do erro para cada qubit lógico.

Qubits	Qubits Lógicos	Tempo (s)
600	50	2,38
1200	100	9,30
1800	150	21,62
2400	200	40,64

Tabela 12 – Resultados da simulação do circuito de correção de erro no Tableau.

Na Tabela 12 é apresentado os resultados dos testes. Como esse circuito usa poucas portas, é possível simular muitos qubits, com um crescimento assintótico quadrático.

5.2 PORTAS ALEATÓRIAS

Outro teste foi montado para testar a capacidade do Tableau de lidar com circuitos com muitas operações de Clifford. Em teoria, como cada operação envolve alterar apenas colunas da tabela, elas tem custo linear em relação a quantidade de qubits.

Esse teste envolve a aplicação de diversas portas aleatórias do grupo de Clifford em cima de um dado número de qubits. Entre essas operações estão as portas X, Z, Y, H e CNOT, no simulador padrão do Ket, todas as operações quânticas tem o mesmo custo, que escala exponencialmente em relação ao número de qubits. No Tableau, essas operações tem custos diferentes, mas todas são assintoticamente proporcionais ao tamanho da coluna, que cresce linearmente em relação a quantidade de qubits. Na Tabela 13 estão os resultados desses testes com diferentes quantidades de qubits e portas.

Qubits	Número de Operações	Tempo (s)
50	12917	1,24
100	51666	9,69
150	116250	36,24
200	206667	91,47
250	322916	184,91

Tabela 13 – Resultados da simulação de circuitos aleatórios no Tableau.

Foi feito uma tentativa de simular a mesma sequência de portas aleatórias no Ket, como o simulador do Ket tem complexidade exponencial, ficou difícil realizar uma comparação entre os dois. O máximo de qubits simulados no Ket foi 26, depois disso o tempo e o espaço de simulação tornou-se proibitivo para o computador. O resultado da comparação é apresentado na Figura 3

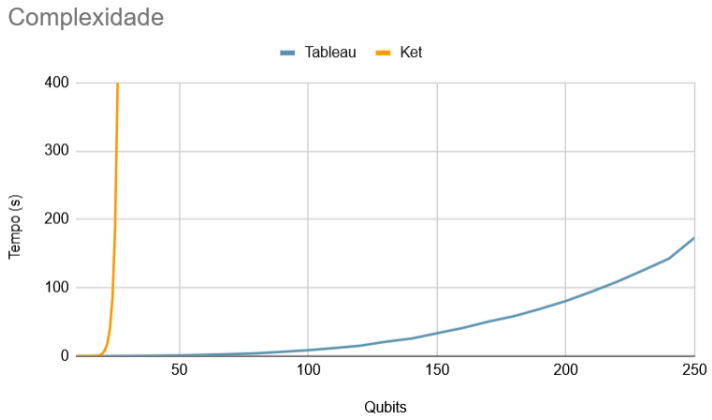


Figura 3 – Comparação simulação do Ket e do Tableau para circuitos com portas aleatórias de Clifford. Elaborado pelo próprio autor.

5.3 ALGORITMO DE GROVER

Por fim, o terceiro circuito foi o algoritmo de busca de Grover. Ele é composto principalmente por portas H, X e portas Toffoli. Esse teste foi feito para testar como que o Tableau lida com um sistema de múltiplas tabelas em comparação com o simulador padrão do Ket. O propósito do circuito é realizar buscas por elementos em listas desorganizadas, sendo capaz de encontrar um elemento desejado na lista em $\sqrt{N}$ operações, sendo N a quantidade de elementos da lista, geralmente representado como 2^q, sendo q a quantidade de qubits usadas para indexar os elementos.

Para um circuito de Grover com n qubits, $q = n - 1$ são usados para representar os itens na lista $0, 1, 2, 3, 4, 5, ..., 2^q - 1$, e o último é usado como qubit auxiliar na busca pelo item. Um item é escolhido arbitrariamente para ser o elemento de busca, no caso, para os testes, foi utilizado sempre o item $2^q - 1$, o elemento escolhido não afeta o desempenho.

Primeiramente, todos os qubits são colocados em superposição com portas H, e o qubit auxiliar tem sua fase trocada por uma porta Z. Depois, duas operações são chamadas, a operação oráculo, e a difusora. O oráculo é uma Toffoli que tem como controle os q qubits, e aplica uma porta X no qubit auxiliar se todos os controles estiverem ativos, a função dela é marcar o estado que se deseja encontrar, análogo a uma *query* de banco de dados clássica que procura por nomes que encaixam em uma determinada *string*.

Algumas portas controle estarão com portas X em volta, isso indica que aquele qubit precisa estar desligado para funcionar, análogo a um "if (not q1)" em um computador clássico. Essas X são postas de acordo com o estado que deseja buscar, caso o estado fosse 11000, teria uma Toffoli com X aplicada em volta dos qubits 3 e 4, com o alvo sendo o quinto qubit.

A operação difusora também é representada por uma Toffoli, com diversas portas H e X aplicadas em todos os qubits de controle antes da Toffoli, e X, H aplicado depois. Caso os controles estejam ativos, a porta aplica uma X no alvo, que é o qubit

auxiliar. Essa operação serve para fazer uma distribuição dos estados, o que faz com que estados não marcados se interfiram destrutivamente entre si, e o estado marcado cresça em probabilidade.

Essas duas operações, do oráculo e do Difusor, podem ser aplicadas diversas vezes no circuito para aumentar a probabilidade do estado desejado ser medido. A quantidade ótima de vezes para se aplicar é $\frac{\pi}{4}\sqrt{2^q}$, mais que isso, e a probabilidade do estado ser medido começa a decair.

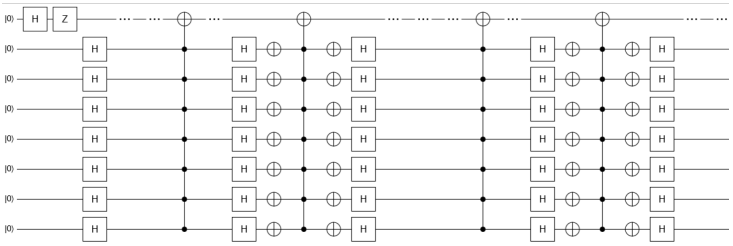


Figura 4 – Circuito de Grover: Inicialização e duas rodadas de oráculo e difusor.

Elaborado pelo próprio autor usando a ferramenta Algassert.

Esse circuito foi replicado em Python usando o Tableau, e usando o simulador padrão do Ket, configurado para o modelo denso. Foram executados vários testes com diferentes quantidades de qubits e rodadas de oráculo e difusor. Os resultados dos experimentos utilizando o simulador padrão do Ket são apresentados na Tabela 14 para motivos de comparação.

Qubits	Rodadas	Tempo (s)
12	36	0,15
15	101	0,91
18	285	7,63
20	569	88,40

Tabela 14 – Resultados da simulação do circuito de Grover no Ket.

Os testes realizados com o Tableau não puderam ser feitos com todas as rodadas recomendadas para garantir a probabilidade do estado ser encontrado. Pois cada rodada envolve a aplicação de 2 portas Toffoli, e cada operação desse tipo pode, no pior caso, triplicar a quantidade de tabelas. Na Tabela 15 é mostrado a capacidade do Tableau de simular esses circuits com diferentes proporções de qubits e rodadas. Além disso, a função dump foi utilizada para recuperar o estado, pois ela é mais eficiente que a função de medição quando se deseja saber o valor de múltiplos qubits em múltiplas tabelas.

Qubits	Rodadas	Portas Toffoli	Tabelas	Tempo (s)
7	6	12	9556	88.28
8	5	10	2388	61.57
9	4	8	596	50.48
10	3	6	148	50.81
11	2	4	36	62.81
12	1	2	8	80.62

Tabela 15 – Resultados da simulação do circuito de Grover no Tableau.

O resultado mostra que a proliferação de tabelas e a operação de medição exponencial torna esse tipo de simulação muito custosa. Sendo assim, é mais útil em circuitos nos quais a quantidade de portas de Clifford é ordens de magnitude maior que a quantidade de portas fora do grupo.

5.4 DISCUSSÃO

Os experimentos mostram basicamente que simular circuitos que estão contidos no grupo de Clifford é uma tarefa trivial para o método de Tableau, conseguindo alcançar quantidades de qubits e portas suficientes para boa parte das pesquisas na área de correção de erro.

Os custos de utilizar o método de Tableau para circuitos fora do grupo de Clifford são proibitivos, e devem ser considerados para casos em que a quantidade de portas de Clifford é muito maior do que a quantidade de portas não Cliffordianas.

Os experimentos com portas de Clifford não puderam ser realizados no Ket, pois não é possível, no estado atual do simulador, conseguir simular mais que 30 qubits em superposição total.

Vale notar que os resultados do experimento utilizando o algoritmo de Grover servem como métrica suficiente para julgar o desempenho do Ket em relação ao Tableau. Considerando que para o método de simulação densa do Ket, não há diferença no desempenho de uma operação baseada no tipo da porta, todas as portas tem o mesmo desempenho.

6 CONCLUSÃO

O1 O método de Tableau foi compreendido.

O2 Foi criado um módulo de simulação para circuitos de Clifford em tempo polinomial em python.

O3 O método foi extendido para que portas fora do grupo de Clifford fossem usadas.

O4 O código foi integrado com o Ket, e agora é possível utilizar esse modo de simulação.

O5 Foram feitos 3 testes para ver como o método de Tableau se comporta em diferentes situações.

O Ket agora possui um novo módulo para simular computação quântica, otimizado especificamente para operações de Clifford, capaz de simular sistemas com muito mais qubits do que o padrão atualmente utilizado.

Dentro do grupo de Clifford, a complexidade temporal do código é polinomial em relação a quantidade de portas e a quantidade de qubits. Sendo n a quantidade de qubits, a tabela tem tamanho $2n$ x $2n$, e cada operação age sob no máximo duas colunas da tabela. A medição pode, no pior caso, realizar somas com o valor de todas as linhas.

De acordo com os testes, essa complexidade baixa é suficiente para simular centenas de qubits e milhares de operações de forma eficaz.

Ao usar portas fora do grupo de Clifford, a complexidade temporal e espacial se torna exponencial em relação a quantidade de portas fora do grupo de Clifford, e a medição se torna exponencial em relação a quantidade de qubits. Para cada tabela existente no sistema, é necessário fazer uma medição de custo 2^n, e a quantidade de tabelas pode no pior caso ser 2^{2G} sendo G a quantidade de portas externas utilizadas.

O algoritmo acaba sacrificando a complexidade da medição para manter as operações do circuito leves. O que pode ser eficiente em circuitos que possuem milhares de portas de Clifford e apenas algumas portas fora de Clifford. Devido ao custo exponencial da medição, simulações com quantidades altas de qubits não são viáveis.

6.1 TRABALHOS FUTUROS

Há espaço para otimizações em algumas operações, como as portas U, que são as mais custosas do sistema. Uma estratégia é encontrar alguma forma de unir tabelas que representem o mesmo estado. Como um estado pode ser codificado por diferentes conjuntos geradores de matrizes, não há como saber diretamente se duas tabelas possuem o mesmo grupo de estabilizadores, e portanto, podem ter seus fatores somados em uma tabela só.

Não foi possível, nesse trabalho, encontrar uma forma de realizar a medição de um estado composto de múltiplas tabelas em tempo polinomial, é possível que exista uma forma de calcular as interferências das tabelas usando apenas os estabilizadores, pois os estados em superposição representados por cada tabela individual são sempre no formato de igual probabilidade. Caso isso seja possível, existiria uma vantagem considerável do método de Tableau para circuitos com poucas portas fora do grupo de Clifford.

Outra opção é utilizar o novo simulador para implementar circuitos mais complexos, com muito mais qubits disponíveis. Como por exemplo, códigos de correção de erro, que codificam vários qubits ruidosos em um único qubit lógico. Estudar tais circuitos antes era uma tarefa difícil com o simulador padrão comportando no máximo entre 20 e 50 qubits. Agora é algo viável, e realizar pesquisas em cima do comportamento desses circuitos foi facilitado.

REFERÊNCIAS

AARONSON, Scott; GOTTESMAN, Daniel. Improved simulation of stabilizer circuits. **Physical Review A**, American Physical Society (APS), v. 70, n. 5, nov. 2004.

ANDERS, Simon; BRIEGEL, Hans J. Fast simulation of stabilizer circuits using a graph-state representation. **Physical Review A**, American Physical Society (APS), v. 73, n. 2, fev. 2006. ISSN 1094-1622. DOI: 10.1103/physreva.73.022334. Disponível em: http://dx.doi.org/10.1103/PhysRevA.73.022334.

BRAVYI, Sergey; GOSSET, David. Improved Classical Simulation of Quantum Circuits Dominated by Clifford Gates. **Physical Review Letters**, American Physical Society (APS), v. 116, n. 25, jun. 2016. ISSN 1079-7114. DOI: 10.1103/physrevlett.116.250501. Disponível em: http://dx.doi.org/10.1103/PhysRevLett.116.250501.

CHENG, Hai-Ping et al. Application of Quantum Computing to Biochemical Systems: A Look to the Future. **Frontiers in Chemistry**, Volume 8 - 2020, 2020. ISSN 2296-2646. DOI: 10.3389/fchem.2020.587143. Disponível em: https://www.frontiersin.org/journals/chemistry/articles/10.3389/fchem.2020.587143.

CORMEN, Thomas H. et al. **Introduction to Algorithms**. 3. ed. [S.l.]: MIT Press, 2009. cap. 2.1.

CUFFARO, Michael E. On the Necessity of Entanglement for the Explanation of Quantum Speedup, 2013.

DEVITT, Simon J; MUNRO, William J; NEMOTO, Kae. Quantum error correction for beginners. **Reports on Progress in Physics**, IOP Publishing, v. 76, n. 7, p. 076001, jun. 2013.

FRISK KOCKUM, Anton; NORI, Franco. Quantum Bits with Josephson Junctions. In: [S.l.*:* s.n.], set. 2019. p. 703–741. ISBN 978-3-030-20724-3. DOI: 10.1007/978-3-030-20726-7_17.

GARNER, Sean et al. **STABSim: A Parallelized Clifford Simulator with Features Beyond Direct Simulation**. [S.l.*:* s.n.], 2025. arXiv: 2507.03092 [quant-ph]. Disponível em: https://arxiv.org/abs/2507.03092.

GOTTESMAN, Daniel. Stabilizer Codes and Quantum Error Correction, 1997. arXiv: quant-ph/9705052 [quant-ph]. Disponível em: https://arxiv.org/abs/quant-ph/9705052.

GOTTESMAN, Daniel. **The Heisenberg Representation of Quantum Computers**. 1998. Tese (Doutorado) – California Institute of Technology. Also published as arXiv:quant-ph/9807006.

GROVER, Lov K. A fast quantum mechanical algorithm for database search. In: PROCEEDINGS of the Twenty-Eighth Annual ACM Symposium on Theory of

Computing. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996. (STOC '96), p. 212–219. DOI: 10.1145/237814.237866. Disponível em: https://doi.org/10.1145/237814.237866.

HUANG, Shilin; BROWN, Kenneth R.; CETINA, Marko. **Comparing Shor and Steane Error Correction Using the Bacon-Shor Code**. [S.l.*:* s.n.], 2023. arXiv: 2312.10851 [quant-ph]. Disponível em: https://arxiv.org/abs/2312.10851.

HUANG, Yifei; LOVE, Peter. Approximate stabilizer rank and improved weak simulation of Clifford-dominated circuits for qudits. **Physical Review A**, American Physical Society (APS), v. 99, n. 5, maio 2019. ISSN 2469-9934. DOI: 10.1103/physreva.99.052307. Disponível em: http://dx.doi.org/10.1103/PhysRevA.99.052307.

KITAEV, A Yu. Quantum computations: algorithms and error correction. **Russian Mathematical Surveys**, IOP Publishing, v. 52, n. 6, p. 1191, 1997.

MENEZES, Alfred J.; OORSCHOT, Paul C. van; VANSTONE, Scott A. **Handbook of Applied Cryptography**. [S.l.]: CRC Press, 1996. 3 and 4.

NIELSEN, Michael A.; CHUANG, Isaac L. **Quantum Computation and Quantum Information: 10th Anniversary Edition**. [S.l.]: Cambridge University Press, 2011. ISBN 9781107002173.

ROSA, Evandro Chagas Ribeiro Da; SANTIAGO, Rafael De. Ket quantum programming. **ACM Journal on Emerging Technologies in Computing Systems (JETC)**, ACM New York, NY, v. 18, n. 1, p. 1–25, 2021.

SHOR, P.W. Algorithms for quantum computation: discrete logarithms and factoring. In: p. 124–134.

SIPSER, Michael. **Introduction to the Theory of Computation**. 3. ed. [S.l.]: Cengage Learning, 2012.

YANOFSKY, Noson S.; MANNUCCI, Mirco A. **Quantum Computing for Computer Scientists**. [S.l.]: Cambridge University Press, 2008.

YASHIN, Vsevolod I.; ELOVENKOVA, Maria A. Characterization of non-adaptive Clifford channels. **Quantum Information Processing**, Springer Science e Business Media LLC, v. 24, n. 3, mar. 2025.

Apêndices

APÊNDICE A – REPOSITÓRIOS

O código para o simulador de Tableau está disponível nos links abaixo.

- Repositório com o código desenvolvido (Códigos@UFSC):
  `https://codigos.ufsc.br/gabriel.turatti.andrade/Tableau-Simulation`
- Repositório com o código desenvolvido (GitHub):
  `https://github.com/Gabriel-Turatti/Tableau-Simulation`

APÊNDICE B – ARTIGO

Artigo no formato especificado pela Sociedade Brasileira de Computação (SBC).

Implementação de um simulador de circuitos Clifford no Ket

Gabriel Turatti Andrade[1], **Jerusa Marchi**[1], **Evandro Chagas Ribeiro da Rosa**[1]

[1]Departamento de Informática e Estatística
Centro Tecnológico – Universidade Federal de Santa Catarina (UFSC)

Resumo. *Atualmente, computadores quânticos possuem pouco poder de processamento, pela alta taxa de erros e a decoerência dos qubits, tornando difícil o desenvolvimento de algoritmos quânticos que sejam úteis na pratica. Devido a esse fator, existe uma demanda por programas que simulem classicamente o fenômeno de forma eficiente. Para isso, existe o Ket, uma plataforma de desenvolvimento de algoritmos quânticos que permite a simulação de circuitos quânticos em computadores clássicos. O objetivo desse trabalho foi implementar um novo módulo de simulação no Ket, baseado no formalismo do Tableau. Esse método permite simulações em tempo polinomial em relação a quantidade de qubits, para circuitos cujas portas estejam no grupo de Clifford, utilizando uma estratégia de representação de estados através do grupo de estabilizadores daquele estado, ao invés das amplitudes em si. Com esse projeto, agora é possível realizar simulações no Ket com mais eficiência, permitindo que circuitos de correção de erros, comumente feitos apenas com operadores de Clifford, sejam simulados em tempo polinomial. Também foi implementada a opção de usar portas quânticas fora do grupo de Clifford, como a porta T, que permitem operações universais, mas que aumentam exponencialmente o tempo da simulação para cada porta usada, ou seja, um circuito com várias portas T pode ser inviável de simular. Para simular essas portas, o módulo do Tableau foi estendido para permitir qualquer porta fora do grupo de Clifford. O intuito dessa alteração é que circuitos com poucas portas fora do grupo comparadas com portas dentro do grupo ainda podem ser simulados em um tempo aceitável. Sendo assim, esse módulo pode ser escolhido para simular um dado circuito caso ele seja mais eficiente do que o simulador padrão do Ket. O módulo foi implementado em Python e integrado com o Ket. Depois de implementado, a eficiência desses algoritmos foi medida para vários circuitos, puramente de Clifford com vários qubits, com poucas portas T, e com várias portas de Clifford, analisando também como o custo escala dependendo da quantidade de qubits e portas utilizadas. As simulações com apenas portas de Clifford obtiveram sucesso, conseguindo simular até 2000 qubits, para os circuitos com portas fora do grupo, os tempos de simulação tornam-se rapidamente proibitivos a medida que a quantidade de portas cresce, conforme esperado.*

Palavras-chave*: Computação Quântica. Simulação Quântica, Grupo de Clifford. Ket. Plataforma Quântica.*

1. Introdução

A computação quântica é uma área de estudo que vem se tornando relevante, por ser capaz de resolver problemas usando estratégias que reduzem a complexidade assintótica do problema para abaixo do que se acreditava estar. Devido a isso, existe uma procura por algoritmos quânticos mais eficientes.

Entretanto, a computação quântica tem suas limitações, pois, para conseguir executar um algoritmo quântico, é necessário modelar o problema alvo de uma forma equivalente no paradigma quântico. Essa tarefa não é trivial, existem diversas restrições na maneira de pensar e representar os problemas, de formas que conflitam com a versão clássica.

O avanço nessa área é dificultado pela falta de computadores quânticos que podem ser de fato utilizados. Modelar algoritmos quânticos se torna uma tarefa muito mais difícil quando o objeto da pesquisa existe apenas de forma teórica. Para isso, muitos projetos foram criados com o propósito de auxiliar o desenvolvimento e simulação da computação quântica, tornando ela um objeto mais acessível, entre eles a linguagem de programação Ket (Rosa; Santiago, 2021).

Essa simulação é uma tarefa de complexidade exponencial, o que inviabiliza as simulações de larga escala. Portanto, grande parte da pesquisa na área de simulação quântica se baseia em reduzir a complexidade da simulação, em busca de aumentar a quantidade de qubits simulados.

Dentre as diversas formas de simular computação quântica, há uma técnica que pode ser feita em tempo polinomial, ela é voltada para circuitos de Clifford, os quais são gerados por um subconjunto das portas quânticas possíveis. Esse subconjunto é caracterizado por conter as portas: Hadamard, Phase, e CNOT (Gottesman, 1997).

A implementação de um simulador que use o grupo de Clifford não é novidade, e já foi explorado em outros trabalhos, demonstrando sua eficácia para circuitos dessa sub categoria (Garner *et al.*, 2025). Nesse trabalho, será implementado a mesma estratégia para a plataforma Ket, com uma adição de simular portas fora do grupo.

Esse grupo de portas não é capaz de criar o conjunto universal de operações, o que limita as simulações possíveis, pois vários algoritmos quânticos usam portas fora do grupo de Clifford. Apesar das limitações, esse subconjunto de operações ainda permite o entrelaçamento de qubits e a superposição, o que ajuda, em parte, a modelar algoritmos. Com a adição de portas arbitrárias e multi-controladas, a computação se torna universal (Gottesman, 1997).

O conjunto de Clifford é relevante também pela área de correção de erros. Atualmente, computadores quânticos são extremamente ruidosos, isso é, são suscetíveis a erros, tornando difícil realizar computações demoradas sem o acumulo de erros inviabilizar a informação. Para isso, estão surgindo diversos estudos sobre como corrigir os erros do computador, que utilizam circuitos de correção de erros, formados principalmente por operadores de Clifford, e baseados na ideia de estabilizadores, ambos elementos viáveis de calcular polinomialmente com o método de Tableau (Devitt; Munro; Nemoto, 2013). Estabilizadores são operadores que não alteram o estado do computador, isso é relevante para correção de erro pois se um estabilizador é aplicado a um estado, e o seu valor é alterado, significa que aquele estado foi afetado por um ruído.

Nesse trabalho, foi implementado um simulador de circuitos de Clifford utilizando o método de Tableau. O método foi também extendido para que portas fora do grupo pudessem ser simuladas com um custo exponencial, usando a estratégia de separar o estado em múltiplas tabelas. Várias otimizações foram feitas visando diminuir a quantidade de tabelas no sistema quando portas fora do grupo são aplicadas.

1.1. Objetivos

O principal objetivo deste trabalho é implementar um simulador quântico alternativo para o Ket, que poderá ser utilizado opcionalmente pelo usuário, otimizado para simular circuitos de Clifford em tempo polinomial. E que permita o uso de portas fora do grupo de Clifford usando uma extensão do método de Tableau. Permitindo, dessa forma, a simulação de circuitos maiores no Ket com complexidades temporais melhores.

Objetivos Específicos

- Estudar a literatura atualizada sobre simulação de circuitos de Clifford.
- Implementar um código para a simulação de Tableau seguindo a literatura, no Python.
- Adicionar operações fora do grupo de Clifford na simulação.
- Integrar o código com a plataforma Ket.
- Avaliar o desempenho do código desenvolvido.

2. Metodologia de Desenvolvimento

Foram realizados três tipos de experimentos, o primeiro com o código de correção de erro do Shor (Huang; Brown; Cetina, 2023) visando testar os limites de simulação de qubits, o segundo com execução de diversas portas aleatórias, visando testar os limites de simulação de portas, e por fim, o algoritmo de Grover, para ver o comportamento do sistema para portas não cliffordianas. Esses experimentos foram rodados multiplas vezes com diferentes parâmetros de qubits e portas.

O primeiro experimento, com correção de erro, foi testado para diferentes quantidades de qubits lógicos, variando de 50 a 200 qubits, para cada qubit lógico, são necessários 9 qubits físicos, além de dois outros qubits auxiliares para fazer a correção de erro de fato, resultando em 11 qubits para cada qubit lógico. No total simulando até cerca de 2400 qubits.

No segundo experimento, diversas portas quânticas de clifford aleatórias foram colocadas na simulação, para testar até que ponto o simulador conseguiria aguentar. A quantidade de qubits também foi variado, de 50 a 250, para cada quantidade, uma quantidade maior de portas foi testadas, para ver como o sistema escala com essas duas variáveis.

O terceiro experimento foi feito com o algoritmo de Grover, composto de duas partes, o oráculo e o difusor, quando esses dois circuitos são executados em sequência, aqui chamado de rodada, eles aumentam a probabilidade de um dado estado ser medido dentro da superposição, esse estado é dito pelo oráculo, para esse trabalho, foi utilizado o estado $|1\rangle^{\otimes n}$. Dependendo de quantos estados existem na superposição, é necessário chamar o oráculo e difusor múltiplas vezes até que a chance do estado ser medido esteja em seu máximo. Para esse teste, não foi possível realizar todas as rodadas recomendadas do Grover no Tableau, pois a complexidade crescia rápido demais em relação ao número de rodadas.

Além disso, para o algoritmo de Grover, o mesmo teste foi feito com o simulador padrão do Ket. Como o simulador denso do Ket não faz distinção na complexidade do

tempo baseado no tipo de operador usado, esse teste é representativo suficiente de como que o simulador se comportaria nas outras situações para ser usado como uma métrica comparativa. No caso do Ket, todas as rodadas recomendadas foram possíveis de serem executadas.

3. Resultados

Nesta seção serão apresentados os resultados, levando em conta principalmente a comparação da complexidade assintótica desse módulo em comparação com o simulador padrão do Ket. Na Tabela 1 são mostrados os parâmetros do computador utilizado para realizar os testes.

CPU	Intel® Xeon™ W-2133
RAM	32 GB
Python	3.12.2
Windows	Windows 10
Ket	0.9.1

Tabela 1. Configuração do computador usado para os testes.

3.1. Código de Correção de Erro do Shor

Foram testados 3 tipos de circuitos, o primeiro foi o circuito de correção de erro com a codificação de Shor (Huang; Brown; Cetina, 2023). Ele foi feito para testar a capacidade do Tableau de simular circuitos puramente de Clifford. Esse tipo de circuito utiliza 9 qubits físicos para representar um único qubit lógico, esse qubit lógico é resistente a erros, pois a redundância dos qubits em sua codificação ajudam a corrigir erros. Algo análogo à estratégia clássica de repetir o mesmo sinal 3 vezes para que a redundância corrija erros de *bit-flip*.

O método de Tableau foi inicialmente desenvolvido com o intuito de auxiliar na representação de circuitos de correção de erro, e portanto, é mais eficiente lidando com circuitos desse formato. No teste, foram utilizados vários qubits lógicos, formados por 9 qubits físicos cada, eles foram preparados fazendo com que os qubits na posição $0, 3, 6$ estivessem em superposição, e entrelaçados com os dois qubits seguintes, criando o estado $(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)$, que é resistente a erros de *bit-flip* e *phase-flip*.

Depois, foram introduzidos erros de *bit-flip* no circuito, aplicando uma X aleatoriamente em um dos 9 qubits pra cada qubit lógico. Logo em seguida, foi aplicado o circuito de correção de erro, que compara os valores dos qubits de cada bloco de 3, para checar se todos os valores estão iguais, e caso não estejam, trocar o valor do qubit que está diferente, esse processo requer 2 qubits auxiliares a mais para registrar a medição do erro para cada qubit lógico.

Qubits	Qubits Lógicos	Tempo (s)
600	50	2,38
1200	100	9,30
1800	150	21,62
2400	200	40,64

Tabela 2. Resultados da simulação do circuito de correção de erro no Tableau.

Na Tabela 2 é apresentado os resultados dos testes. Como esse circuito usa poucas portas, é possível simular muitos qubits, com um crescimento assintótico quadrático.

3.2. Portas Aleatórias

Outro teste foi montado para testar a capacidade do Tableau de lidar com circuitos com muitas operações de Clifford. Em teoria, como cada operação envolve alterar apenas colunas da tabela, elas tem custo linear em relação a quantidade de qubits.

Esse teste envolve a aplicação de diversas portas aleatórias do grupo de Clifford em cima de um dado número de qubits. Entre essas operações estão as portas X, Z, Y, H e CNOT, no simulador padrão do Ket, todas as operações quânticas tem o mesmo custo, que escala exponencialmente em relação ao número de qubits. No Tableau, essas operações tem custos diferentes, mas todas são assintoticamente proporcionais ao tamanho da coluna, que cresce linearmente em relação a quantidade de qubits. Na Tabela 3 estão os resultados desses testes com diferentes quantidades de qubits e portas.

Qubits	Número de Operações	Tempo (s)
50	12917	1,24
100	51666	9,69
150	116250	36,24
200	206667	91,47
250	322916	184,91

Tabela 3. Resultados da simulação de circuitos aleatórios no Tableau.

3.3. Algoritmo de Grover

Por fim, o terceiro circuito foi o algoritmo de busca de Grover. Ele é composto principalmente por portas H, X e portas Toffoli (Grover, 1996). Esse teste foi feito para testar como que o Tableau lida com um sistema de múltiplas tabelas em comparação com o simulador padrão do Ket. O propósito do circuito é realizar buscas por elementos em listas desorganizadas, sendo capaz de encontrar um elemento desejado na lista em $\sqrt{N}$ operações, sendo N a quantidade de elementos da lista, geralmente representado como 2^q, sendo q a quantidade de qubits usadas para indexar os elementos.

Para um circuito de Grover com n qubits, $q = n-1$ são usados para representar os itens na lista $0, 1, 2, 3, 4, 5, ..., 2^q - 1$, e o último é usado como qubit auxiliar na busca pelo item. Um item é escolhido arbitrariamente para ser o elemento de busca, no caso, para os testes, foi utilizado sempre o item $2^q - 1$, o elemento escolhido não afeta o desempenho.

Primeiramente, todos os qubits são colocados em superposição com portas H, e o qubit auxiliar tem sua fase trocada por uma porta Z. Depois, duas operações são

chamadas, a operação oráculo, e a difusora. O oráculo é uma Toffoli que tem como controle os q qubits, e aplica uma porta X no qubit auxiliar se todos os controles estiverem ativos, a função dela é marcar o estado que se deseja encontrar, análogo a uma *query* de banco de dados clássica que procura por nomes que encaixam em uma determinada *string*.

Algumas portas controle estarão com portas X em volta, isso indica que aquele qubit precisa estar desligado para funcionar, análogo a um "if (not q1)" em um computador clássico. Essas X são postas de acordo com o estado que deseja buscar, caso o estado fosse 11000, teria uma Toffoli com X aplicada em volta dos qubits 3 e 4, com o alvo sendo o quinto qubit.

A operação difusora também é representada por uma Toffoli, com diversas portas H e X aplicadas em todos os qubits de controle antes da Toffoli, e X, H aplicado depois. Caso os controles estejam ativos, a porta aplica uma X no alvo, que é o qubit auxiliar. Essa operação serve para fazer uma distribuição dos estados, o que faz com que estados não marcados se interfiram destrutivamente entre si, e o estado marcado cresça em probabilidade.

Essas duas operações, do oráculo e do Difusor, podem ser aplicadas diversas vezes no circuito para aumentar a probabilidade do estado desejado ser medido. A quantidade ótima de vezes para se aplicar é $\frac{\pi}{4}\sqrt{2^q}$, mais que isso, e a probabilidade do estado ser medido começa a decair.

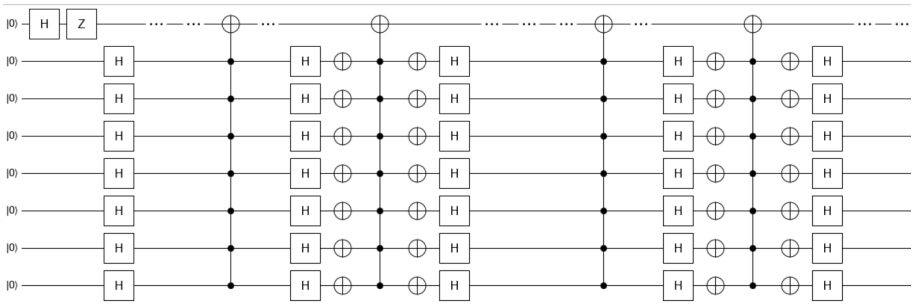


Figura 1. Circuito de Grover: Inicialização e duas rodadas de oráculo e difusor. Fonte: Elaborado pelo próprio autor.

Esse circuito foi replicado em Python usando o Tableau, e usando o simulador padrão do Ket, configurado para o modelo denso. Foram executados vários testes com diferentes quantidades de qubits e rodadas de oráculo e difusor. Os resultados dos experimentos utilizando o simulador padrão do Ket são apresentados na Tabela 4 para motivos de comparação.

Qubits	Rodadas	Tempo (s)
12	36	0,15
15	101	0,91
18	285	7,63
20	569	88,40

Tabela 4. Resultados da simulação do circuito de Grover no Ket.

Os testes realizados com o Tableau não puderam ser feitos com todas as rodadas recomendadas para garantir a probabilidade do estado ser encontrado. Pois cada rodada envolve a aplicação de 2 portas Toffoli, e cada operação desse tipo pode, no pior caso, triplicar a quantidade de tabelas. Na Tabela 5 é mostrado a capacidade do Tableau de simular esses circuits com diferentes proporções de qubits e rodadas. Além disso, a função dump foi utilizada para recuperar o estado, pois ela é mais eficiente que a função de medição quando se deseja saber o valor de múltiplos qubits em múltiplas tabelas.

Qubits	Rodadas	Portas Toffoli	Tabelas	Tempo (s)
7	6	12	9556	88.28
8	5	10	2388	61.57
9	4	8	596	50.48
10	3	6	148	50.81
11	2	4	36	62.81
12	1	2	8	80.62

Tabela 5. Resultados da simulação do circuito de Grover no Tableau.

O resultado mostra que a proliferação de tabelas e a operação de medição exponencial torna esse tipo de simulação muito custosa. Sendo assim, é mais útil em circuitos nos quais a quantidade de portas de Clifford é ordens de magnitude maior que a quantidade de portas fora do grupo.

4. Conclusão

1. O método de Tableau foi compreendido.
2. Foi criado um módulo de simulação para circuitos de Clifford em tempo polinomial em python.
3. O método foi extendido para que portas fora do grupo de Clifford fossem usadas.
4. O código foi integrado com o Ket, e agora é possível utilizar esse modo de simulação.
5. Foram feitos 3 testes para ver como o método de Tableau se comporta em diferentes situações.

O Ket agora possui um novo módulo para simular computação quântica, otimizado especificamente para operações de Clifford, capaz de simular sistemas com muito mais qubits do que o padrão atualmente utilizado.

Dentro do grupo de Clifford, a complexidade temporal do código é polinomial em relação a quantidade de portas e a quantidade de qubits. Sendo n a quantidade de qubits, a tabela tem tamanho $2n$ x $2n$, e cada operação age sob no máximo duas colunas da tabela. A medição pode, no pior caso, realizar somas com o valor de todas as linhas.

De acordo com os testes, essa complexidade baixa é suficiente para simular centenas de qubits e milhares de operações de forma eficaz.

Ao usar portas fora do grupo de Clifford, a complexidade temporal e espacial se torna exponencial em relação a quantidade de portas fora do grupo de Clifford, e a medição se torna exponencial em relação a quantidade de qubits. Para cada tabela existente no sistema, é necessário fazer uma medição de custo 2^n, e a quantidade de tabelas pode no pior caso ser 2^{2G} sendo G a quantidade de portas externas utilizadas.

O algoritmo acaba sacrificando a complexidade da medição para manter as operações do circuito leves. O que pode ser eficiente em circuitos que possuem milhares de portas de Clifford e apenas algumas portas fora de Clifford. Devido ao custo exponencial da medição, simulações com quantidades altas de qubits não são viáveis.

5. Trabalhos futuros

Há espaço para otimizações em algumas operações, como as portas U, que são as mais custosas do sistema. Uma estratégia é encontrar alguma forma de unir tabelas que representem o mesmo estado. Como um estado pode ser codificado por diferentes conjuntos geradores de matrizes, não há como saber diretamente se duas tabelas possuem o mesmo grupo de estabilizadores, e portanto, podem ter seus fatores somados em uma tabela só.

Não foi possível, nesse trabalho, encontrar uma forma de realizar a medição de um estado composto de múltiplas tabelas em tempo polinomial, é possível que exista uma forma de calcular as interferências das tabelas usando apenas os estabilizadores, pois os estados em superposição representados por cada tabela individual são sempre no formato de igual probabilidade. Caso isso seja possível, existiria uma vantagem considerável do método de Tableau para circuitos com poucas portas fora do grupo de Clifford.

Outra opção é utilizar o novo simulador para implementar circuitos mais complexos, com muito mais qubits disponíveis. Como por exemplo, códigos de correção de erro, que codificam vários qubits ruidosos em um único qubit lógico. Estudar tais circuitos antes era uma tarefa difícil com o simulador padrão comportando no máximo entre 20 e 50 qubits. Agora é algo viável, e realizar pesquisas em cima do comportamento desses circuitos foi facilitado.

Referências

DEVITT, Simon J; MUNRO, William J; NEMOTO, Kae. Quantum error correction for beginners. **Reports on Progress in Physics**, IOP Publishing, v. 76, n. 7, p. 076001, jun. 2013.

GARNER, Sean *et al.* **STABSim: A Parallelized Clifford Simulator with Features Beyond Direct Simulation**. [*S. l.: s. n.*], 2025. arXiv: 2507.03092 [quant-ph]. Disponível em: https://arxiv.org/abs/2507.03092.

GOTTESMAN, Daniel. Stabilizer Codes and Quantum Error Correction, 1997. arXiv: quant-ph/9705052 [quant-ph]. Disponível em: https://arxiv.org/abs/quant-ph/9705052.

GROVER, Lov K. A fast quantum mechanical algorithm for database search. *In:* PROCEEDINGS of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996. (STOC '96), p. 212–219. ISBN 0897917855. DOI: 10.1145/237814.237866. Disponível em: https://doi.org/10.1145/237814.237866.

HUANG, Shilin; BROWN, Kenneth R.; CETINA, Marko. **Comparing Shor and Steane Error Correction Using the Bacon-Shor Code**. [*S. l.: s. n.*], 2023. arXiv: 2312.10851 [quant-ph]. Disponível em: https://arxiv.org/abs/2312.10851.

ROSA, Evandro Chagas Ribeiro Da; SANTIAGO, Rafael De. Ket quantum programming. **ACM Journal on Emerging Technologies in Computing Systems (JETC)**, ACM New York, NY, v. 18, n. 1, p. 1–25, 2021.