UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Teo Haeser Gallarza

Investigação do crossover point do problema da satisfação booleana utilizando o algoritmo quântico de Grover

Florianópolis
2022

Este trabalho é dedicado aos meus amigos colegas e às
minhas duas famílias.

AGRADECIMENTOS

Agradeço a todos que acreditaram e tiveram a calma na espera desse trabalho.

A todos que ajudaram, mesmo que só ouvindo eu reclamando da vida.

Agradeço em especial Chico, Taguma, Chew e Ana por me aguentarem todos esses anos.

Agradeço à minha primeira família, meus pais, meu irmão e cunhada e minhas duas sobrinhas favoritas.

À minha segunda família, as feiticeiras, a casa aonde aprendi a me expressar e não ter medo de ser quem sou.

Cross por ser um dos grande motivos de risadas da minha graduação.

A todos os *palomeiros* que sempre se dedicaram tanto pelo meu bem estar e felicidade, com todas as reuniões de livros ruins que tivemos na pandemia.

Aos *lasanhas de colherzinhas*, que apesar de chegarão tarde nesse momento da minha vida, forão tão marcantes me ouvindo tantas horas falar sobre meus problemas.

Vitor, Marlon e Paulo um trio de amigos que talvez não percebam o quão importantes foram pra mim.

Gabriel, Bruno e Gui que mesmo depois de tanto tempo e com todas as distâncias que a vida nos impôs, ainda gostam tanto de mim.

Ao Renato por fazer incríveis docinhos.

Ao Roque que me apresentou a Esther e que me apresentou o Nero.

Mikael, que depois de o destino nos unir novamente, ainda me aguenta, uma mídia da Marvel de cada vez.

Agradecer ao paulo pelas repentinas companhias nos almocos/RU.

À todas minhas professoras e colegas de dança, principalmente a Bell, o Will, a Pri e a Julia, além é claro, da Paola que me acompanha nessa jornada desde o quase começo e ainda me incentiva a ir muito mais longe.

Thales, Otto e todos os colegas de mesas de RPG.

Pelos moradores da *lan house* por cada hora de conversa jogada fora e todos as aventuras mirabolantes que passamos para aguentar aquele lugar.

Vitt e Paulo por cada momento no *Discord* conversando, que me entreteram por horas e horas com muitas fofocas e jogatinas.

Tiz e Júlio por serem incríveis parceiros de jogos e aonde conheci novas paixões, *AoE2* e revivi antigas, *L4D2*.

Adler, Osnei, Emerson e João que mesmo criando uma distância ainda se dispunham em me receber a qualquer momento para conversar e jogar.

Aos demais colegas de curso e amigos.

“I must not fear.
Fear is the mind-killer.
Fear is the little-death that brings total obliteration.
I will face my fear.
I will permit it to pass over me and through me.
And when it has gone past I will turn the inner eye to see its path.
Where the fear has gone there will be nothing.
Only I will remain.”
(Frank Herbert, Dune)

RESUMO

O crossover point representa um ponto onde as instâncias do problema de satisfação booleana (SAT) são muito mais difíceis de serem resolvidas. O objetivo desse trabalho é fazer uma investigação do comportamento das instâncias ao serem executadas no algoritmo de Grover. O algoritmo de Grover é um algoritmo quântico utilizado para encontrar um elemento em uma lista desordenada que apresenta comportamento assintótico da ordem de $\sqrt{N}$, onde N é a quantidade de elementos na lista. Para sua implementação, fez-se uso da linguagem Ket e do simulador de computação quântica QuBox. A investigação faz uso de instâncias SAT geradas aleatoriamente, porém limitadas no número de qubits. Foram implementados, além do algoritmo de Grover e o de geração de instâncias SAT, um algoritmo para converter tais instâncias em oráculos de Grover e um algoritmo para iterar o algoritmo de Grover, quando o número de respostas para o oráculo é desconhecido. Devido às diversas limitações e ao tempo necessário para a execução das instâncias, não foi possível ter uma visão clara sobre a real complexidade da execução de instâncias SAT no ponto de crossover, quando executadas no algoritmo de Grover.

Palavras-chave: Algoritmo de Grover. Problemas de Satisfação Booleana. Crossover point.

ABSTRACT

The crossover point represent a point where the instances of the satisfaction problem (SAT) are incredibly hard to answer. The objective of this work is to do an investigation of the behaviour of the instances when executed in the Grover algorithm. The Grover algorithm is a quantum algorithm that finds an element in an unordered list that has an asymptotic behavior of $\sqrt{N}$, where N is the number of elements in the list. For it's implementation, it used the language Ket and the quantum computer simulator QuBox. The investigation made use of randomly generated SAT instances, but limited in the number of qubits Besides the Grover algorithm and the generation of SAT instances, it was also implemented an algorithm to convert the SAT instances into quantum oracles and an algorithm to iterate the Grover algorithm when the number of answers of the oracle is unknown. Due to the limitations found and the time needed to execute the instances, it wans't possible to have a clear understanding of the real complexity to run the SAT instances in the crossover point, when excecuted in the Grover algorithm.

Keywords: Grover algorithm. Boolean Satisfaction Problem. Crossover point.

SUMÁRIO

1 INTRODUÇÃO

Desde a criação dos computadores, muitos estudos têm sido dedicados ao aperfeiçoamento de seus componentes e algoritmos, com o intuito de conseguir executar os programas de maneira cada vez mais rápida. A partir desses estudos, vieram grandes avanços tecnológicos, mas a estrutura básica sobre a qual se faz a computação nunca foi alterada.

O uso de silício e código binário é a base da computação clássica e, apesar de suas limitações, principalmente a necessidade de dissipação de energia, os circuitos integrados vinham sendo rapidamente melhorados, chegando a duplicar sua capacidade a cada 2 anos (MACK, 2011). O atingimento do limite físico de calor das peças do computador mostrou que é necessária a pesquisa sobre novas maneiras de se arquitetar computadores, criando interesse em várias novas áreas de pesquisa.

Com esse interesse e os avanços científicos feitos na área da física quântica, foi criada uma nova área de pesquisa chamada *Computação Quântica*, que tem por objetivo tentar criar um computador que tenha mais processamento que os computadores atuais, conhecidos como clássicos, a partir de propriedades físicas apenas vistas no campo quântico.

A área de computação quântica já teve vários avanços teóricos importantes, por exemplo, a criação de algoritmos, como os algoritmo de Shor (SHOR, 1994) e de Grover (GROVER, 1996), sendo eles algoritmos que apresentam ganhos exponencial e quadrático, respectivamente, se comparados com suas contrapartidas clássicas. Porém ainda não foi possível verificar os ganhos esperados da computação quântica, uma vez que os computadores quânticos atuais não suportam a execução de grandes circuitos quânticos. Isso ocorre pois não existe ainda alguma máquina física que consiga fazer esse cálculos com uma taxa de erro aceitável, e as máquinas atuais, chamadas de NISQ (Noisy Intermediate-Scale Quantum) (PRESKILL, 2018), ainda não têm a capacidade de executar problemas grandes e complexos como o deste trabalho. Com o objetivo de superar essas adversidades, foi preciso utilizar um simulador para a execução dos algoritmos. Contudo, ao se utilizar um simulador, não é possível ter efetivamente a aceleração esperada da execução dos algoritmos quânticos. Isso dificulta a medição verídica do custo computacional da execução de problemas, portanto foram utilizados outros tipos de métricas para fazer as avaliações necessárias.

O trabalho é composto inicialmente de uma revisão teórica sobre os assuntos abordados na pesquisa. Primeiro será vista uma revisão do problema de satisfação booleana, em que será explicado o problema e também o conceito do *crossover point*, que será o foco desse trabalho.

Dentro ainda da revisão teórica, serão apresentados conceitos de física quântica, explicados e exemplificados dentro do contexto da computação quântica. Dessa

forma não será necessário entendimento prévio da área para entender o resto do trabalho.

A última parte da revisão é a explicação em detalhes do algoritmo de Grover, apresentando para tal um exemplo mostrando os detalhes matemáticos e as portas lógicas.

Após a revisão, é abordado o problema do trabalho, a relação do problema SAT, mais especificamente o *crossover point*, com o oráculo de Grover, demonstrando como é possível um algoritmo de Grover resolver problemas SAT. Essa relação é baseada na relação entre instâncias SAT e um oráculo quântico utilizado no algoritmo de Grover. São então explicadas as implementações dos algoritmos produzidos para esse trabalho, são eles: o algoritmo de Grover, o algoritmo de geração de instâncias do problemas SAT, o algoritmo de transformação de instâncias SAT em oráculos quânticos e o algoritmo de validação de entradas para instâncias SAT. Também é apresentado um algoritmo clássico para o teste da satisfação de instâncias (MENG, 2017), utilizado para auxiliar a identificar as instâncias satisfazíveis.

Ao final do trabalho, são apresentados os gráficos obtidos, buscando demonstrar a complexidade e a capacidade do algoritmo de Grover de resolver as instâncias SAT geradas. Com esses gráficos, será analisado o efeito do *crossover point* sobre o algoritmo de Grover. A partir desses gráficos, também serão analisadas as limitações atuais de se tentar aplicar o algoritmo de Grover em casos reais, tentando testar ao máximo o simulador utilizado, para ver quais a limitações que ele apresenta. Também será analisado se o algoritmo de Grover, nesse contexto, é útil para resolver instâncias SAT.

1.1 MOTIVAÇÃO

Os estudos na área da computação quântica são importantes para se entender as limitações dessa nova área do conhecimento. Além disso, a aplicação de algoritmos já conhecidos dentro de um computador quântico é vital para se compreenderem as limitações reais que estes podem apresentar. Com esse trabalho, será possível entender melhor o funcionamento do problema SAT no contexto de um computador quântico, dando uma visão de como outros algoritmos NP-completos podem ser executados em tal contexto.

1.2 PERGUNTA DE PESQUISA

O problema SAT é um problema importante para o estudo de classes de algoritmos. Notadamente, instâncias SAT, cuja taxa dada pelo número de cláusulas, dividido pelo número de variáveis, gira em torno de 4,2, região conhecida como *crossover point*, apresentam maior dificuldade de resolução.

Posto isso, esse trabalho tem como objetivo responder a pergunta de qual é o comportamento do problema SAT, ao ser executado em um computador quântico?

1.3 OBJETIVOS

O objetivo principal deste projeto de pesquisa é investigar o funcionamento do algoritmo de Grover, quando utilizado para resolver instâncias do problema SAT próximas do *crossover point*. Desse modo, estabelecendo relações entre a complexidade das instâncias SAT sendo executadas em um algoritmo clássico em relação a um algoritmo quântico.

1.3.1 Objetivos Específicos

1. Compreender e implementar o algoritmo de Grover .
2. Verificar capacidade do algoritmo de Grover de resolver instâncias SAT dada a relação de cláusulas por variável de cada instância.
3. Analisar a complexidade computacional de se adaptar o algoritmo de Grover para resolver instâncias SAT.
4. Analisar quais instâncias apresentam maior dificuldade de serem resolvidas pelo algoritmo de Grover.
5. Analisar a utilidade do algoritmo de Grover nesse contexto ao invés de algoritmos clássicos.

2 REVISÃO TEÓRICA

Este capítulo apresenta uma revisão teórica sobre o problema da satisfação booleana, apresentando o que é o crossover point. Depois são vistos os conceitos básicos de computação quântica necessários para o entendimento do trabalho, bem como serão apresentados os postulados da mecânica quântica. Em seguida, explica-se em detalhes o algoritmo de Grover, utilizando um exemplo para ilustrar o seu funcionamento.

2.1 PROBLEMA DA SATISFAÇÃO BOOLEANA

O problema da satisfação booleana (SAT) consiste em encontrar uma atribuição de valores verdade que satisfaça, ou seja, torne verdadeira, uma fórmula booleana. Uma fórmula booleana é uma expressão envolvendo variáveis booleanas, como p_1, p_2, e operadores booleanos, como $\vee$ (ou), $\wedge$ (e) e $\neg$ (negação). As variáveis recebem valores *TRUE* ou *FALSE*, e, ao se aplicarem os operadores lógicos, seguindo as premissas da lógica clássica, dadas pelas tabelas verdade dos operadores, a avaliação da fórmula retorna um desses valores. Diz-se que uma fórmula booleana é satisfeita ou satisfazível se existe uma entrada que a valida, ou seja, que retorna o valor *TRUE*. Assim uma fórmula é insatisfazível, se não houver entrada qualquer que satisfaça a fórmula. Com isso, podemos definir o problema da satisfação booleana como o problema de determinar se uma fórmula booleana é satisfazível ou não.

O problema SAT é importante dentro da área de estudo de complexidade de problemas Hopcroft *et al.* (2006). Dentre as classes de complexidade, a classe NP assume central importância. Definida como a classe de problemas decidíveis em tempo não determinístico polinomial (SIPSER, 2013), estão nesta classe os problemas para os quais não existe forma conhecida (algoritmo) de se encontrar uma resposta em tempo polinomial, sendo necessário fazer uma busca exaustiva sobre o conjunto de dados da entrada para se determinar a solução do problemas. Essa busca exaustiva pode tomar tempo exponencial ao tamanho da instância de entrada. No caso do problema SAT, a dificuldade é de se descobrir se existe uma atribuição de valores verdade para as variáveis que valide a fórmula, criando a necessidade de testar todas as possíveis interpretações ou se comprovar que não existe uma atribuição possível e, portanto, a fórmula é insatisfazível. Porém, os problemas classificados como NP têm a característica de que eles verificam eficientemente soluções do problema, isso significa que, ao se apresentar uma solução possível, é possível determinar, em tempo polinomial, se essa solução é ou não uma solução para o problema. (HOPCROFT *et al.*, 2006)

O problema SAT é classificado como NP-Completo (SIPSER, 2013), uma subclasse dos problemas NP, sendo este o primeiro a ser classificado como tal pelo teorema de Cook-Levin (COOK, 1971). Ele é usado como base para se provar que ou-

tros problemas são também NP-Completos. Dada a existência de diversos problemas que podem ser reduzidos em problemas na classe dos NP-Completos, é de interesse se descobrir maneiras mais rápidas de se resolver tais problemas e, em particular, o problema SAT. A dificuldade em se demonstrar a impossibilidade da existência de algoritmos em tempo polinomial que resolvam problemas NP-Completos, bem como a dificuldade de se provar que tais algoritmos não existem, caracteriza o mais importante problema ainda em aberto na área da computação, conhecido como $P \stackrel{?}{=} NP$ (FORTNOW, 2009).

Uma fórmula lógica pode ser reescrita, restringindo o conjunto de operadores lógicos e também a forma de suas cláusulas. Uma forma normal conjuntiva - CNF - restringe o conjunto de operadores a apenas $\vee$ (ou), $\wedge$ (e) e $\neg$ (negação) e o formato da fórmula deve ser uma "conjunção de disjunções" onde os literais de uma cláusula são conectados com $\vee$ e as cláusulas são conectadas com $\wedge$ (SIPSER, 2013). Este formato é usado como base para o padrão DIMACS-CNF [1], que é o formato padrão utilizado em competições de algoritmos para solução de SAT.

O padrão DIMACS-CNF consiste em representar a fórmula em um formato textual, onde a primeira linha, conhecida como cabeçalho, tem o formato ***p cnf #v #c***, onde *v* é o número de variáveis e c é o número de cláusulas. Após essa linha, todas as próximas linhas descrevem uma cláusula, indicando os literais que estão dentro da cláusula, terminando a linha com um *0*.

Como um exemplo, tomemos a fórmula $(p2 \vee p3) \wedge (\neg p1 \vee \neg p2 \vee \neg p3) \wedge (p1 \vee \neg p2 \vee p3)$, que tem 3 variáveis e 3 cláusulas. Quando representada em formato DIMACS, tem-se:

```
p cnf 3 3
2 3 0
-1 -2 -3 0
1 -2 3 0
```

Para se validar uma fórmula, quando em uma CNF, pelo menos um literal de cada cláusula deve assumir valor verdade verdadeiro. A fórmula acima tem com uma resposta possível $(\neg p1 \wedge \neg p2 \wedge p3)$, que pode ser facilmente verificada associando a estas valor verdade verdadeiro e constatando que cada variável valida pelo menos uma cláusula da fórmula.

Uma variante do problema SAT é o 3-SAT, também reconhecidamente NP-Completo, onde toda cláusula contém exatamente 3 literais. Essa variação será usada durante o trabalho, por sua simplicidade no número de literais por cláusula e por ser co-

[1] Link para leitura mais extensa do formato DIMACS-CNF: http://www.domagoj-babic.com/uploads/ResearchProjects/Spear/dimacs-cnf.pdf

nhecido que um problema SAT qualquer, com um número maior de literais por cláusula, pode ser transformado, em tempo polinomial, em um problema 3-SAT (HOPCROFT *et al.*, 2006).

Como mencionado, outros algoritmos classificados como NP-Completos são equivalentes em questão de complexidade computacional ao problema SAT, portanto têm a característica de que podem ser reduzidos a um problema SAT. Com isso, é criada a necessidade de se encontrar as maneiras mais rápidas de se resolver um problema SAT e entender seus casos mais difíceis.

2.1.1 Crossover point

Dado que uma fórmula booleana é constituída por um número *c* de cláusulas e por um número *v* de variáveis, pode-se fazer uma relação de $\frac{c}{v}$, e, como demonstrado por Crawford e Auton (1996), existe uma ligação entre essa relação e a dificuldade de resolver uma instância do problema SAT em específico. Ao se avaliar essa relação, é possível entender que, quando o valor da divisão é muito pequeno, ou seja, existe um número próximo de cláusulas e variáveis, é computacionalmente simples de resolver o problema e, quando se têm muitas mais cláusulas que variáveis, é fácil provar que não existe como resolver o problema.

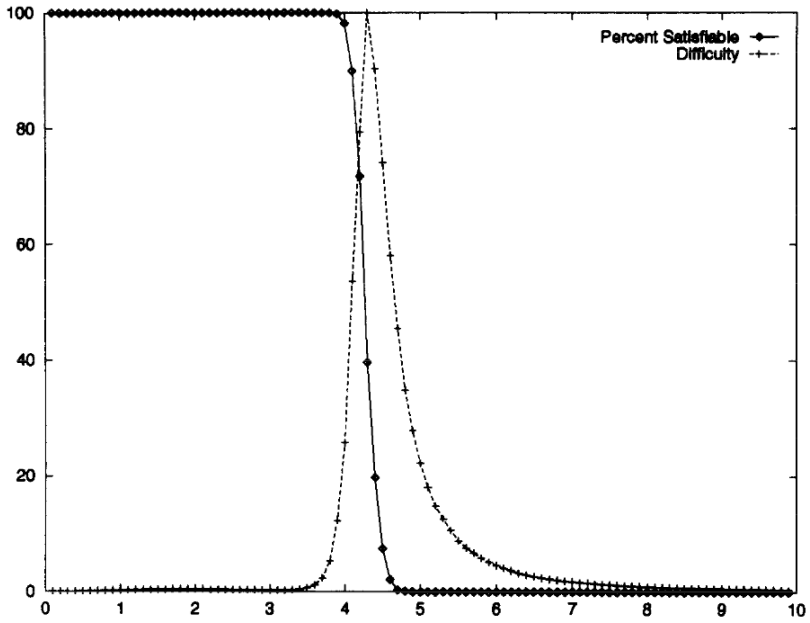

Figura 1 – Relação entre dificuldade de execução e porcentagem de satisfazibilidade. Obtido em (CRAWFORD; AUTON, 1996)

Porém, como é possível ver na figura 1, quando essa relação se aproxima do valor 4,2, existe uma grande dificuldade computacional para encontrar essa resposta, pois nesse ponto se encontram as fórmulas onde é praticamente necessário fazer uma busca exaustiva de todas as possibilidades, para poder provar tanto se existe um resultado possível que satisfaça a teoria lógica, quanto se essa fórmula é insatisfazível.

Esse ponto, batizado de crossover point, tem seu valor exato dependente dos tipos de problemas SAT analisados variando em cada caso, porém ele sempre existe próximo do valor 4,2. Essa análise foi feita utilizando algoritmos clássicos para resolver os problemas SAT, e, como ainda não foram encontradas formas mais eficientes de se resolver problemas NP-completos, todos problemas SAT que se utilizam de algoritmos clássicos para sua resolução devem cair nessa classificação de dificuldade. Porém, o que aconteceria, caso os mesmos problemas fossem resolvidos por algoritmos de outras arquiteturas, como um computador quântico?

2.2 COMPUTAÇÃO QUÂNTICA

Nas próximas seções, serão explicadas noções básicas de computação quântica para o entendimento do algoritmo implementado. Nelas serão revisados conceitos básicos de notações da física quântica, além de seus postulados, os quais serão utilizados para se entender melhor o funcionamento de um computador quântico.

2.2.1 Notação de Dirac

A notação de Dirac (DIRAC, 1939), também conhecida como notação bra-ket, é amplamente usada no campo da computação quântica. Consiste tanto de $\langle\ |$ como $|\ \rangle$, onde '$|\ \rangle$', chamado de *ket*, representa um vetor, também conhecido como vetor coluna, e '$\langle\ |$', chamado de *bra*, representa seu conjugado transposto, representado por †, ou seja um vetor linha. Pode ser interpretado também que '$\langle\ |$' representa um mapeamento linear de um vetor para um número.

A notação tem propriedades importantes como:

- $|\psi\rangle = \langle\psi|^\dagger$;
- $\langle\xi|\psi\rangle$ representa o produto interno dos vetores, mas pode ser interpretado como a aplicação do vetor $|\psi\rangle$ no mapeamento $\langle\xi|$ e é representado como

$$\langle\xi|\psi\rangle = \begin{bmatrix}\xi_0^\dagger & \dots & \xi_n^\dagger\end{bmatrix}\begin{bmatrix}\psi_0\\ \vdots\\ \psi_n\end{bmatrix} = \xi_0^\dagger\psi_0 + \dots + \xi_n^\dagger\psi_n. \quad (1)$$

- $|\psi\rangle\langle\xi|$ representa o produto externo dos vetores.

2.2.2 Postulados da mecânica quântica

Para entender sobre computação quântica, é necessário primeiramente entender sobre a mecânica quântica que rege o funcionamento dos computadores quânticos. Para isso, é importante conhecer os postulados da mecânica quântica que serão usados como base para se entender os conceitos utilizados.

Nielsen e Chuang (2011) listam 4 postulados e, com eles, é possível ter a base necessária para entender o funcionamento de algoritmos quânticos. Cada postulado será explicitado e contará com um tópico subsequente que mostrará a utilização do postulado na prática.

2.2.2.1 Postulado 1 - Espaço do estado

Postulado 1 *Associado a cada sistema físico fechado, existe um espaço de Hilbert* [2] *conhecido como espaço do estado, onde o sistema pode ser completamente descrito por um vetor unitário conhecido por vetor do estado.*

Qubit

O postulado 1 determina uma relação entre vetores e espaço de Hilbert, então o *vetor do espaço* pode ser visto como um vetor da álgebra linear, podendo assim ser utilizado em operações matriciais da álgebra linear. Com essa ideia, foi criado um conceito chamado qubit, o *quantum bit*, que é representado por um *vetor do espaço* e pode ser utilizado em operações similares a portas lógicas para se fazerem computações com os qubits.

Mas qual a motivação de se utilizar qubits, em vez de de bits clássicos? O bit é um conceito fundamental da computação clássica, onde se cria uma base binária para os estados possíveis do bit, sendo eles 0 ou 1. Já, na computação quântica, pelo fato de se utilizarem vetores como a unidade fundamental de computação, é possível fazer cálculos mais complexos que um bit clássico. Então a principal diferença entre qubits e bits clássicos é que os qubits podem estar tanto nos estados 0, 1, como em estados intermediários resultantes da combinação linear do estado 0 e 1.

Para se representar os qubits, utiliza-se a notação de Dirac (introduzida da Seção 2.2.1), onde o vetor utiliza uma base para representar o estado atual do qubit. Comumente se utiliza a chamada base computacional $\{|0\rangle, |1\rangle\}$ para representar os qubits, sendo os vetores representados da forma $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ e $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

Utilizando a base computacional como exemplo, um qubit qualquer pode ser descrito pela equação

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

onde α e β são números complexos.

Dada essa combinação linear descrita pela equação 2, é possível criar uma representação de uma esfera para todos os estados possíveis, colocando os estados de base como dois vetores em mesma direção porém em sentido opostos da esfera, essa representação é chamada de esfera de Bloch [Figura 2], onde cada ponto da

[2] Espaço de Hilbert é um espaço vetorial com produto interno

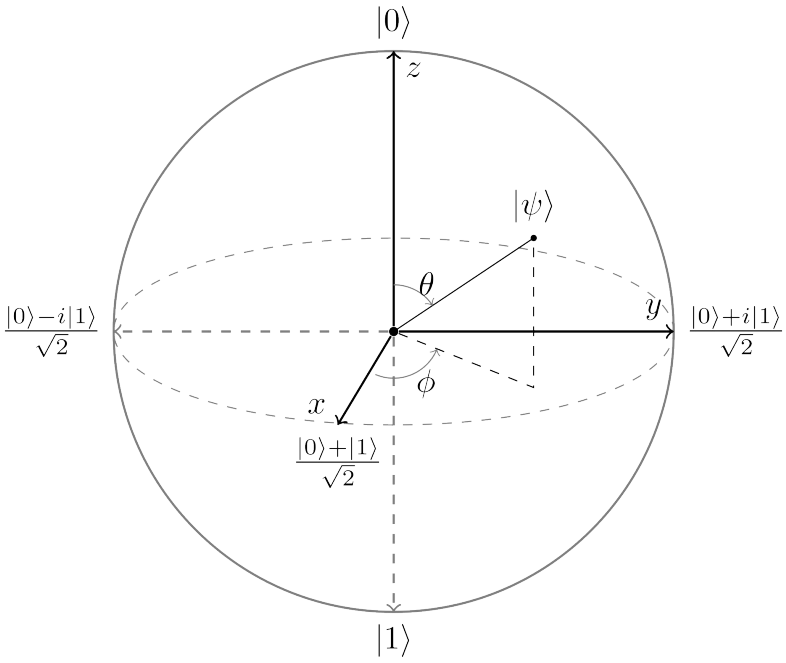

Figura 2 – Representação da esfera de Bloch. Retirado de: https://quantum-ket.gitlab.io/qubox/qc.html

superfície representa um estado possível do qubit. Os estados que se encontram numa posição diferente dos estados base se diz estarem em superposição, onde o seu estado é dado por uma combinação qualquer originária da equação 2.

Dos estados em superposição, existe um notável para esse trabalho, o estado $|-\rangle$. Ele é equivalente a $(|0\rangle - |1\rangle)/\sqrt{2}$ e será utilizado em explicações futuras.

2.2.2.2 Postulado 2 - Evolução

Postulado 2 *A evolução de um sistema quântico fechado pode ser descrita por uma transformação unitária. Logo o estado $|\psi\rangle$ do sistema no tempo t_1 está relacionado ao estado $|\psi'\rangle$ do sistema no tempo t_2 pela operação unitária U, criando a relação*

$$|\psi'\rangle = U|\psi\rangle . \tag{3}$$

Operações quânticas

Pelo fato de os qubits serem representados por vetores, as transformações unitárias podem ser representadas por matrizes de transformação linear, as quais mapeiam o vetor de estado, qubit, para outro vetor de estado. Esses operadores depois serão utilizados como portas lógicas quânticas, para se construir algoritmos quânticos onde as portas terão os nomes das operações quânticas.

Operador Unitário - Essas operações são unitárias, ou seja, a sua inversa são iguais ao seu conjugado transposto, conhecido como operador adjunto

$$U^{-1} = U^{\dagger}. \tag{4}$$

Isso traz a propriedade de que qualquer operador unitário aplicado ao seu adjunto resulta na matriz identidade, portanto qualquer operação unitária pode ser revertida. Além disso, todo operador cujo adjunto é igual à matriz original é o seu próprio inverso, logo, ao se aplicar duas vezes a mesma operação, o vetor de entrada continuará inalterado:

$$UU^{\dagger} = I. \tag{5}$$

I é o operador identidade, que será explicado posteriormente.
Alguns exemplos de operações unitárias são:

1. **Operador X** - O operador X é representado pela matriz:

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

E representa a transformação $\{|0\rangle \rightarrow |1\rangle, |1\rangle \rightarrow |0\rangle\}$ e tem equivalência com a porta lógica clássica *NOT*

2. **Operador de Hadamard** - O operador de Hadamard é representado pela matriz

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{7}$$

E representa a transformação $|0\rangle \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ e $|1\rangle \rightarrow \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. Essa porta é utilizada para criar uma superposição em qubits. Por isso, o operador de Hadamard é comumente utilizado para criar uma inicialização dos valores dos qubits, onde os qubits que passarem por esse processo podem compôr um sistema composto [postulado 4] de qubits em superposição.

Tomando como exemplo o algoritmo de Grover, os qubits têm primeiramente o valor $|0\rangle$ e, após passarem pelo operador H, eles serão transformados para o estado $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$.

3. **Operador Identidade** - O operador identidade representa uma operação que não altera o valor que for operado e é representado por:

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

Dado que qualquer operador pode gerar o operador identidade, é possível perceber que os operadores são reversíveis, caso seja aplicado o Hermitiano do operador, porém é também notável que, caso um operador cujo Hermitiano é igual ao operador original for aplicado duas vezes, a operação também será revertida. Esse conceito é importante para uma técnica que será utilizada chamada descomputação, que será melhor explicada na sub-seção 2.2.5, e com ela é possível reverter operações em qubits, assim limpando os qubits utilizados de modificações de valores e entrelaçamentos [postulado 4] consequentes das operações utilizadas.

2.2.3 Postulado 3 - Medição

Postulado 3 *As medidas quânticas podem ser descritas por uma coleção M_m de operadores de medição. Essas são operações que agem sobre um espaço de estado sendo medido. O índice m se refere ao resultado possível que pode acontecer no experimento. Se o estado do sistema quântico é $|\psi\rangle$ imediatamente antes da medição, então a probabilidade do resultado m acontecer é dada por*

$$p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle . \tag{9}$$

onde o estado do sistema depois da medição é

$$\frac{M_m |\psi\rangle}{\langle\psi| M_m^\dagger M_m |\psi\rangle}. \tag{10}$$

O operador de medição satisfaz a equação de completude

$$\sum_m M_m^\dagger M_m = I. \tag{11}$$

A equação de completude expressa o fato de que todas as probabilidades se somam, chegando em 1

$$1 = \sum_m p(m) = \sum_m \langle\psi| M_m^\dagger M_m |\psi\rangle . \tag{12}$$

2.2.3.1 Colapso do qubit

Como visto anteriormente, os qubits podem estar em superposições, ampliando as possibilidade de representações de estados. Porém, ao serem medidos, o resultado sempre será um dos estados da base dos operadores de medida, logo existe uma probabilidade envolvida na chance de qual estado da base será o resultante da medição da superposição e o cálculo de probabilidade de um qubit colapsar em um dos estados da base é dado pela equação (9). Quando um estado do espaço que se encontra em superposição é medido e seu estado é alterado para algum valor dos autoestados do operador de medida, se diz que ele colapsou. O colapso é a única

ação sobre um estados de que não é reversível, logo não existe operação que retorne um qubit colapsado para o estado que se encontrava antes da medição.

Para exemplificar, consideremos o qubit no estado $\alpha|0\rangle + \beta|1\rangle$. Ele tem como característica que $|\alpha|^2$ é a probabilidade de se medir $|0\rangle$ e $|\beta|^2$ é a probabilidade de se medir $|1\rangle$. É notável pela equação (12) que existe uma relação entre essas duas probabilidades, e essa relação pode ser expressa pela equação $|\alpha|^2 + |\beta|^2 = 1$ e, com essa relação, pode-se chegar à conclusão de que, ao se aumentar a probabilidade de um dos resultados possíveis, o outro irá diminuir também.

2.2.4 Postulado 4 - Sistemas compostos

Postulado 4 *O espaço do estado de um sistema composto é o produto tensorial dos espaços dos componentes do sistema. Além disso, se tivermos os sistemas numerados de 1 até n, e o sistema número i está preparado no estado* $|\psi_i\rangle$*, então o estado conjunto total do sistema é* $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$

2.2.4.1 Sistemas compostos

Sistemas compostos quânticos são sistemas com mais de um qubit que são associados pela operação de produto tensorial entre os qubits individuais, criando assim um vetor que representa todo o sistema composto. Uma maneira de se representar a operação entre dois qubits que estão associados em um sistema composto com maior facilidade é: $|\psi_1\rangle \otimes |\psi_2\rangle = |\psi_1\psi_2\rangle$

Com a união de, por exemplo, $|0\rangle$ e $|0\rangle$ temos, $|0\rangle \otimes |0\rangle = |00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$.

E na união de $|1\rangle$ e $|0\rangle$ temos, $|1\rangle \otimes |0\rangle = |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$.

A representação mais utilizada desses dois exemplos é $|00\rangle$ e $|10\rangle$, porém comumente se faz a transformação dos números de dentro para números decimais, com o objetivo de facilitar a leitura, tendo nesse exemplo $|0\rangle$ e $|2\rangle$.

2.2.4.2 Evolução de sistemas compostos

O sistema composto também pode sofrer uma evolução e, como a evolução mostrada na seção 2.2.2.2, pode ser representada por uma operação linear. Para se criar uma transformação para um sistema composto, pode-se fazer um produto

tensorial entre operações de sistemas menores, onde a dimensão da operação será o produto das dimensões das operações que a compõem

$$U|00\rangle = (U \otimes U)|00\rangle = U|0\rangle \otimes U|0\rangle . \tag{13}$$

As operações então são sobre todos os qubits do sistema composto, a menos que seja explicitado quais são os qubits alvo pela notação U_i, sendo isso equivalente, em um sistema composto de tamanho n, a:

$$U_i = I_0 \otimes I_1 \otimes ... \otimes U_i \otimes ... \otimes I_n. \tag{14}$$

2.2.4.3 Emaranhamento Quântico

O emaranhamento quântico, às vezes descrito como entrelaçamento, acontece quando um sistema composto não pode ser descrito pelo produto tensorial de cada qubit. Os estados Bell, por exemplo, são estados que se encontram emaranhados, sendo um deles:

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{15}$$

A partir desse estado, não há nenhum qubit $|a\rangle$ e $|b\rangle$ tal que $|\psi\rangle = |a\rangle + |b\rangle$.

Com isso não é possível aplicar alguma operação sobre um qubit em específico sem afetar os outros emaranhados, já que não podem ser representados como qubits individualmente. Essa relação entre os qubits se aplica não só quando se faz uma evolução do sistema, mas também quando se faz uma medição dos qubits.

Isso é possível de se notar no estado de Bell $|\psi\rangle$, pois, ao se medir o primeiro qubit, o segundo irá colapsar para o mesmo valor, já que existem apenas duas possibilidades, $|00\rangle$ ou $|11\rangle$.

2.2.5 Descomputação

Como dito na seção 2.2.2.2, a descomputação é uma técnica para “limpar“ os qubits e, com o entendimento do emaranhamento quântico da seção 2.2.4.3, é possível entender melhor como funciona e para que serve a descomputação (BICHSEL *et al.*, 2020).

Algumas operações lineares transformam um sistema composto não emaranhado em um estado emaranhado. Entretanto, pela natureza das operações quânticas de serem reversíveis, é possível reverter o estado de emaranhamento quântico de um sistema, caso a operação inversa da operação que entrelaçou seja aplicada no sistema de novo. Também é possível aplicar operações que resultam em sistemas emaranhados de sistemas já emaranhados, colocando o que poderiam ser chamadas

de camadas de emaranhamento, e, para reverter, é necessário aplicar as operações inversas em ordem reversa, assim voltando ao estado não emaranhado.

Com isso, é possível fazer a descomputação em cima de um sistema que passou por operações que causaram o sistema se tornar emaranhado, algo importante, caso os qubits de um sistema emaranhado precisem ser separados do sistema para passar por operações não locais ou medições, sem alterar outros bits que estejam emaranhados.

É importante ressaltar que, como dito na seção 2.2.3.1, a ação de medir um qubit não é reversível, logo, caso um qubit colapse em algum resultado, não serão possíveis operações para se retornar ao estado anterior. Com isso, a operação de medição de qubit é a única operação que não pode ser descomputada e, portanto, é comumente feita apenas no final da execução dos algoritmos.

2.3 ALGORITMO DE GROVER

O algoritmo de Grover é um dos primeiros algoritmos quânticos a demonstrar vantagem sobre um algoritmo clássico, inventado por Lov Grover (GROVER, 1996). O objetivo do algoritmo é fazer uma busca por itens em uma lista não ordenada, o que, em um computador clássico, demanda $\mathcal{O}(N)$ avaliações, onde N representa a quantidade de itens da lista, pois, no pior dos casos, seria necessário avaliar todos os N itens para encontrar o item buscado (CORMEN *et al.*, 2009). Já o algoritmo de Grover consegue resolver usando apenas $\mathcal{O}(\sqrt{N})$ avaliações.

O Algoritmo em si é dividido em três partes, que serão detalhadas em seguida. Estes passos são a inicialização, a aplicação do iterador de Grover por até $\mathcal{O}(\sqrt{N})$ vezes e, por fim, a medição do resultado final. Para a execução do algoritmo é necessário $n = \log_2(N)$ qubits de entrada que serão utilizados para a busca, mais os qubits auxiliares utilizados para guardar o resultado intermediário das cláusulas avaliadas, além de um oráculo quântico que serve para marcar os itens que estão sendo procurados na lista.

Nesta seção, o funcionamento do algoritmo será explicado em mais detalhes, sendo intercalado com seções de exemplos que serão utilizadas para seguir um exemplo real de uma execução do algoritmo de Grover, nas quais serão apresentadas tanto uma visualização matemática quanto quais portas lógicas seriam necessárias para a implementação da parte do algoritmo sendo explicada.

2.3.1 Inicialização

A primeira parte do algoritmo é conhecida como a inicialização. É uma parte importante, porém que acontece apenas uma vez no início da execução do algoritmo e consiste em distribuir igualitariamente as probabilidades entre os estados da base

computacional possíveis dos qubits de entrada. Os qubits de entrada são inicializados em $|0\rangle$ e nessa etapa são aplicadas em todos eles portas Hadamard, assim criando um sistema em superposição dos qubits onde as amplitudes de todos os estados têm uma mesma amplitude com o valor de $\frac{1}{\sqrt{N}}$. Com os qubits nesse estado, os próximos passos do algoritmo poderão avaliar entre todas as possibilidade representadas pelos qubits.

2.3.1.1 Exemplo - Inicialização

O exemplo que será seguido nessas seções será a busca pelo número 4 na lista de números $[0, 1, ..., 6, 7]$. A lista está em ordem para simplificar a explicação, porém os elementos representados pelos qubits estarão em uma ordem aleatória. Nesse caso o valor de N é igual a 8. Para a criação da lista com todas as possibilidades, são necessários 3 qubits ($\log_2(8)$) e passar cada um individualmente por uma porta Hadamard, assim colocando os qubits em superposição, representando todas as possibilidades das listas.

$$|\psi\rangle = H|0\rangle \otimes H|0\rangle \otimes H|0\rangle, \tag{16}$$

$$= \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right), \tag{17}$$

$$= \frac{1}{\sqrt{8}}(|0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle + |5\rangle + |6\rangle + |7\rangle). \tag{18}$$

Como demonstrado na equação 18, após a aplicação das portas H, o estado do sistema é um vetor de todas as possibilidades, todas com a mesma probabilidade de medição, $\frac{1}{8}$.

2.3.2 Iterador de Grover

O iterador de Grover é a operação que será aplicada $\mathcal{O}(\sqrt{N})$ vezes, uma seguida da outra, tendo como a saída de uma operação, a entrada da outra. A operação é dividida em duas partes, o oráculo e o difusor, sendo o primeiro uma operação única, dependendo da função que o algoritmo está tentando responder, enquanto o difusor aplica uma transformação de rotação de fase no sistema e tem uma forma conhecida, porém não existe apenas uma maneira de se implementar.

2.3.2.1 Oráculo

O oráculo serve para selecionar, entre todas as possibilidades do sistema, os estados que atendem a função que o oráculo representa, fazendo uma rotação em Z

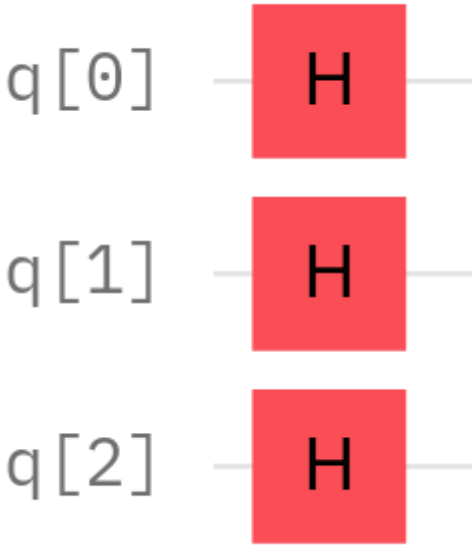

Figura 3 – Aplicação de 3 portas Hadamar nos 3 qubits inicializados em $|0\rangle$

e causando uma troca de sinal nos valores que o oráculo identificar. Esse processo serve para marcar os estados e será necessário para a próxima etapa.

A função que o oráculo representa é da forma

$$f(|\psi\rangle) = \begin{cases} f(|\psi_w\rangle) = 1, & \text{Caso seja um dos valores procurados,} \\ f(|\psi\rangle) = 0, & \text{Para todos os outros estados de } |\psi\rangle\,. \end{cases} \tag{19}$$

A equação recebe um vetor $|\psi\rangle$ qualquer e retorna 1 caso o valor seja um dos que responde à equação do oráculo, marcado como $|\psi\rangle_w$, e retorna 0 para os outros casos.

Depois de ser aplicada a função, será pego o resultado da função e, caso o resultado seja 1, então o estado em questão vai receber a rotação em X. Uma representação dessa operação como um todo em um vetor é: $f(|\psi\rangle\,|y\rangle) = |\psi\rangle\,|y \bigoplus f(\psi)\rangle$, onde o qubit $|y\rangle$ é um qubit auxiliar que será marcado no caso de que $|\psi\rangle$ ser um dos valores procurados, ou seja caso a função f retorne 1.

A implementação real do oráculo pode ignorar a necessidade desse qubit, usando a porta X para fazer a rotação no vetor selecionado, porém ainda é necessário selecionar os estados que atendem a função corretamente e, para esse trabalho, será utilizado um algoritmo base para se criar o circuito corretamente.

2.3.2.2 Exemplo - Oráculo

Depois de aplicada a inicialização dos qubits, é passado o vetor resultante como entrada para o oráculo, que irá marcar o vetor que representa o valor 4 com o sinal negativo.

$$|\psi_1\rangle = O_f(|\psi\rangle) = \frac{1}{\sqrt{8}}(|0\rangle + |1\rangle + ... - |4\rangle + ... + |7\rangle). \tag{20}$$

Portas lógicas

Normalmente o oráculo é representado como uma caixa preta, dentro do qual o conteúdo não é explicitado. Isso se deve ao fato de o circuito ser baseado em uma função onde o resultado não é conhecido, então se deve modelá-la como um circuito quântico, porém, no caso do nosso exemplo, já sabemos qual é o resultado, tornando necessário fazer o caminho ao contrário, tentar achar uma função que retorne o valor desejado. Mas podemos considerar que não sabíamos inicialmente qual valor estamos procurando e apenas temos a função transformada em circuito lógico da imagem 4. Uma mesma expressão pode ser implementada de diversas maneiras em um circuito quântico, e nesse caso foram utilizados dois qubits auxiliares (*q[3]* e *q[4]*), para ajudar a visualização, porém poderiam ser utilizadas maneiras mais simples de se transformar essa função.

As portas em que estão escritas "+", são portas *not*. Algumas delas têm uma linha vertical com mais dois círculos azuis, essas portas são conhecidas como portas de controle, onde os pequenos círculos azuis controlam a ativação de uma porta lógica em outro *qubit*. Nesse caso são utilizados dois *qubits* de controle para a aplicação de uma porta *not* em um dos qubits auxiliares. Para ativar essa porta *not*, é necessário que todos os *qubits* de controle estejam no estado 1, assim alterando o valor do *qubit* alvo. É utilizada essa porta de controle, mais conhecida como *multiple-control-Not*, para fazer a avaliação dos valores e apenas ativar a porta *not* quando a entrada for o valor procurado.

Após a aplicação das portas, é feita uma descomputação das operações, refazendo-as em ordem contrária, menos a operação de marcar, para assim retirar qualquer efeito de emaranhamento que essas portas possam ter criado entre si.

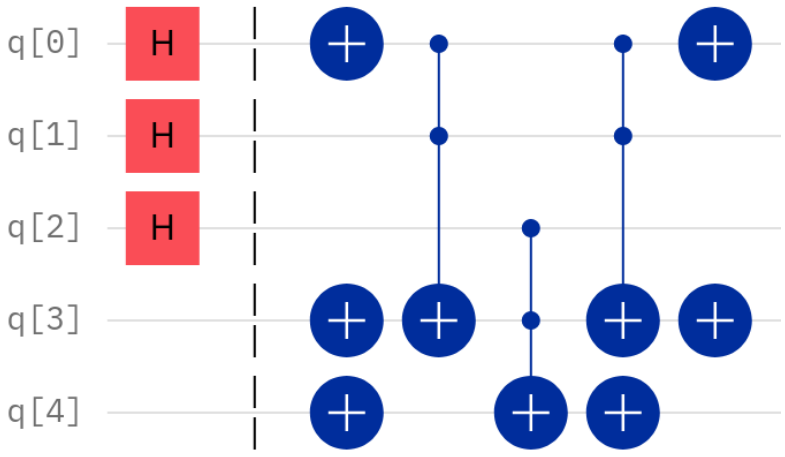

Figura 4 – Implementação do oráculo da função de exemplo. Também está na imagem a parte da inicialização e dois qubits adicionais.

2.3.2.3 Difusor de Fase

Nessa parte será aplicado o difusor de fase representado pela operação

$$D = 2 |\psi\rangle \langle\psi| - I. \tag{21}$$

Ele representa uma rotação em cima da média dos vetores, fazendo com que valores muito próximos da média sejam pouco alterados e valores longe da média tenham uma rotação grande. Como pode ser visto na figura [5].

Esse detalhe é importante, pois, no passo do oráculo, os vetores procurados tiveram os seus sinais invertidos, tendo assim uma diferença notável da média e todos os outros vetores não tiveram alterações, então continuaram muito próximos da média. Logo, o difusor de fase pouco afetará a amplitude desses outros vetores. Uma representação de como essa rotação funciona nos vetores durante uma aplicação do difusor pode ser vista na figura [6]

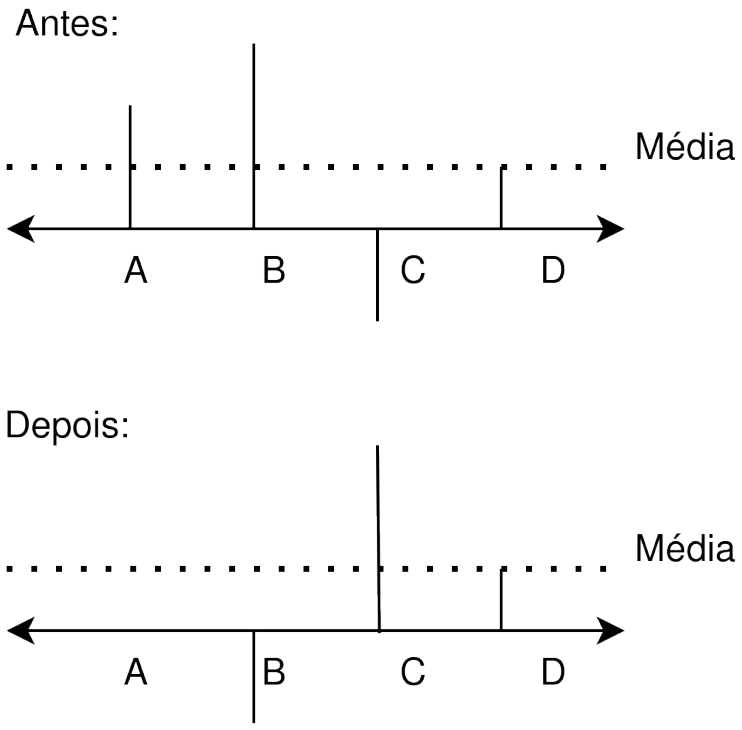

Figura 5 – Rotação de vetores sobre a média.

2.3.2.4 Exemplo - Difusor de fase

O difusor de fase é utilizado no intuito de aumentar a probabilidade de medição dos valores marcados. A equação 21 representa a operação necessária para chegar a esse resultado e abaixo é possível ver como ela afeta os vetores de entrada. Não é necessário entender totalmente a aplicação da fórmula, apenas perceber na equação 28 que o vetor que está sendo procurado, $|4\rangle$, teve sua amplitude aumentada, enquanto

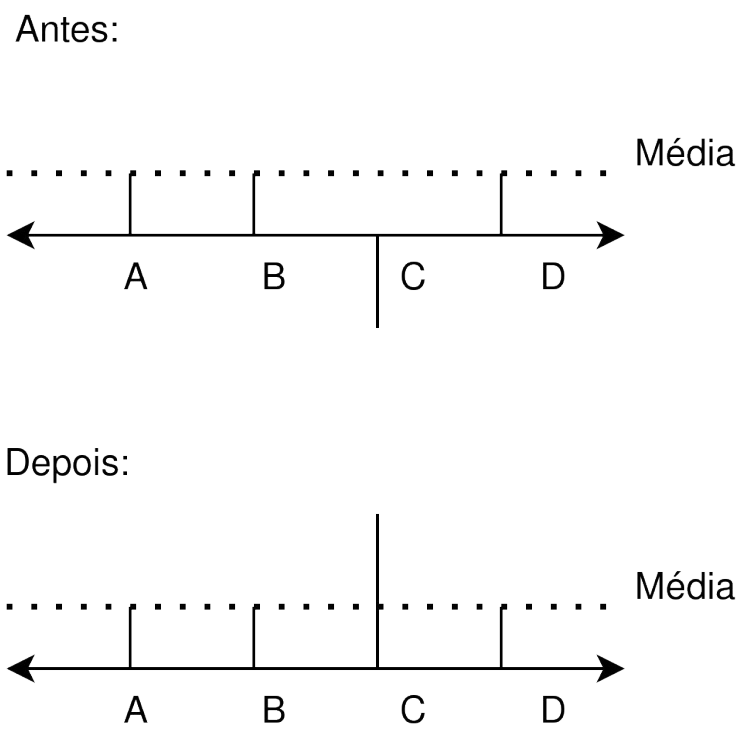

Figura 6 – Rotação do vetor marcado.

a amplitude dos outros vetores diminuiu, criando uma probabilidade muito alta de se medir o valor 4, caso o sistema seja medido após a aplicação do difusor

$$|\psi_2\rangle = D(|\psi_1\rangle) \tag{22}$$
$$= (2\,|\psi\rangle\,\langle\psi| - I)\,|\psi_1\rangle \tag{23}$$
$$= 2\,\langle\psi|\psi_1\rangle\,|\psi\rangle - |\psi_1\rangle \tag{24}$$
$$\text{e dado que: } \langle\psi|\psi_1\rangle = (1+1+1+1-1+1+1+1)/8 = 6/8 \tag{25}$$
$$|psi_2\rangle = \frac{3}{2}\,|\psi\rangle - |\psi_1\rangle \tag{26}$$
$$= \frac{3(|0\rangle\ldots)}{2\sqrt{8}} - \frac{2(|0\rangle\ldots - |4\rangle\ldots)}{2\sqrt{8}} \tag{27}$$
$$= \frac{(|0\rangle + |1\rangle + |2\rangle + |3\rangle + |5\rangle + |6\rangle + |7\rangle)}{2\sqrt{8}} + \frac{5(|4\rangle)}{2\sqrt{8}} \tag{28}$$

2.3.2.5 Número de iterações

Como já estabelecido, o número de iterações do operador de Grover para chegar ao resultado é $\mathcal{O}(\sqrt{N})$, porém ainda é vital ter um número inteiro para a execução real do algoritmo. O valor de iterações é baseado no número de respostas possíveis para

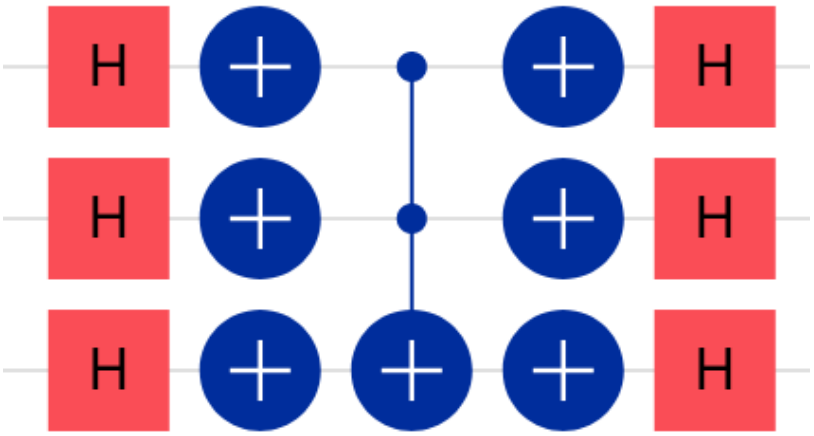

Figura 7 – Implementação do difusor de fase.

o oráculo, sendo o número dado pela fórmula (CRAWFORD; AUTON, 1996)

$$it \leq \frac{\pi}{4}\sqrt{\frac{N}{t}} \tag{29}$$

onde t é o número de resultados que satisfazem a equação do oráculo. Com isso, é possível se determinar um máximo de iterações que, com uma grande chance, terá um resultado possível.

Entretanto, existem casos em que o número de resultados da equação não é conhecido, como, por exemplo, ao se avaliar um problema SAT, já que, para determinar o número de resultados possíveis, é necessário resolver o problema, tornando redundante a execução do algoritmo de Grover.

Como nesses casos não é possível determinar com antecedência o número de iterações necessárias para se encontrar o resultado, deve-se executar o algoritmo de Grover mais de uma vez, com números de iterações diferentes e depois avaliar se o resultado final de cada execução é correto. Essa maneira de resolver o problema funciona bem com o problema SAT, pois a avaliação se uma resposta é válida para a fórmula demanda um tempo polinomial, assim não aumentando a complexidade do algoritmo (HOPCROFT *et al.*, 2006).

2.3.3 Avaliação

No estágio final da execução, é apenas necessário fazer uma medição dos qubits sendo avaliados e, com uma probabilidade muito alta, será registrada uma resposta possível para a função modelada.

Para o exemplo mostrado, foi utilizado um oráculo que tinha apenas uma resposta possível, porém é possível fazer oráculos que têm mais de uma resposta. Nesses casos, o oráculo irá marcar todos esses estados e, no próximo estágio, todos irão au-

mentar igualmente a probabilidade de serem escolhidos como resposta. Com maior número de respostas, também se diminui o número de iterações necessárias para se resolver o problema, criando uma vantagem em utilizar oráculos que têm mais de uma resposta possível, já que serão executados menos vezes e terão maior probabilidade de estarem corretos.

3 DESENVOLVIMENTO

Essa seção é dedicada ao desenvolvimento e execução dos algoritmos propostos, para, assim, se chegar aos resultados finais. Primeiro será explicado cada algoritmo em detalhes, ilustrando seu funcionamento com pseudo-código. Com isso, a próxima seção entrará em detalhes como foi feita a implementação dos algoritmos, explicando o processo de união deles e quais as limitações foram encontradas e impostas para o funcionamento dos programas.

Após a execução do oráculo, todo o resto do algoritmo de Grover segue uma implementação normal do problema, em que a implementação de um problema SAT dentro do algoritmo de Grover não traz nenhuma outra diferença, tornando apenas necessário implementar corretamente a parte de transformar uma instância SAT em um oráculo, para, assim, o algoritmo poder resolver essa instância.

3.1 ALGORITMOS

Essa próxima parte do trabalho será dedicada a entender os algoritmos utilizados durante o projeto. Estes serão o de geração de instâncias SAT, o de transformação de SAT em oráculos, o algoritmo de Grover e o algoritmo de número desconhecido de respostas. Também foi empregado um algoritmo de avaliação de instâncias SAT, porém, por ter sido implementado por outra pessoa (MENG, 2017) e seu funcionamento ser bem conhecido, não foi entrado em muitos detalhes sobre sua implementação. Todos os outros algoritmos serão explicados em suas sub-seções onde contarão com um texto explicativo e um algoritmo propriamente dito, demonstrando o funcionamento de cada um.

Os algoritmos que se utilizam de portas lógicas quânticas foram produzidos utilizando a linguagem de programação Ket (ROSA; SANTIAGO, 2021), uma linguagem que torna possível a simulação de um computador quântico em computadores clássicos. Já os outros algoritmos foram feitos em Python.

3.1.1 Implementação de geração de problemas SAT

Essa implementação foi baseada no artigo de Mitchell *et al.* (1992), onde, dado um número de cláusulas e variáveis, cada cláusula é criada ao se selecionarem três variáveis em aleatório, e negando cada com uma probabilidade de 0.5. Os problemas SAT gerados serão no padrão DIMACS-CNF, em que cada linha representa uma disjunção, ou seja, cada cláusula é separada pela operação *AND*, e cada variável dentro de cada cláusula é separada pela operação OR.

3.1.2 Implementação de transformação de instâncias SAT em oráculos

Para compreender a próxima parte do trabalho, é necessário entender sobre a relação entre o problema de satisfação Booleana e o oráculo de Grover. Como no problema SAT é buscado um valor que consiga satisfazer uma instância SAT, é possível representar as variáveis do problema como um vetor binário, em que cada elemento do vetor guarda se o sinal da variável é positivo ou negativo, sendo apenas necessário procurar qual dessas combinações é suficiente para responder à fórmula.

Com isso, podemos criar uma relação entre uma instância SAT e um oráculo de Grover. Para isso, devemos considerar os qubits de entrada como as variáveis do problema, onde cada qubit corresponde a uma das posições do vetor binário das variáveis. Assim, as avaliações das operações serão aplicadas sobre esses qubits de entrada e o resultado final será uma medição desse vetor das variáveis.

Dadas essas relações, é apenas necessário converter as operações lógicas *OR* e *AND*, utilizadas no problema SAT, para portas lógicas quânticas, assim, podendo transformar uma instância SAT qualquer em um oráculo quântico. A operação *NOT* é facilmente representada pela porta quântica *X*, então não necessita de conversão.

Exemplo utilizando Ket da porta NOT

```
def NOT(qubit):
    X(qubit):
```

Diferentemente da computação clássica, toda operação quântica deve ser reversível, logo não existe uma porta quântica que represente uma porta *AND* na sua totalidade. Porém, ainda é possível modelá-la em um contexto quântico com a porta *multi-controlada-Not*, utilizando os qubits a serem avaliados como os controles e colocando o resultado da operação em outros qubits auxiliares. Assim, o resultado de uma operação *AND* entre os qubits de controle é guardado nesse qubit auxiliar, deixando os qubits avaliados inalterados e, com isso, é possível ter uma equivalência com a porta *AND* clássica. Um detalhe importante de se notar é que, além de guardar o resultado em outro qubit, a aplicação de uma porta *multi-controlada-Not* cria um emaranhamento entre os qubits, então é importante aplicar a descomputação posteriormente, após a utilização do resultado da operação.

Exemplo utilizando Ket da porta AND

```
def AND(avaliados, resultado):
with control(avaliados):
  NOT(resultado)
```

Já a porta *OR* não tem nenhuma porta com uma equivalência direta na computação quântica, então, para recriá-la num algoritmo quântico, foi utilizada a lei de De Morgan $(X \vee Y) = \neg(\neg X \wedge \neg Y)$, a fim de facilitar a criação dessa porta. Para se fazer uma implementação dessas, é utilizada a implementação da porta *AND* acima, sendo necessário negar os qubits de entrada e o qubit de saída, e então aplicar um

AND, assim fazendo uma operação equivalente a aplicar um *OR*. Importante notar que a inversão dos qubits de entrada deve ser revertida logo após a operação *AND*, já que eles foram pontualmente negados para fazer a operação equivalente, então devem ser revertidos a seu estado antes da aplicação da lei de De Morgan, assim, voltando para seu estado antes da operação OR.

Exemplo utilizando Ket da porta OR

```
1 def OR(avaliados, resultado):
2     with around(NOT, avaliados):
3         AND(avaliados,resultado)
4     NOT(resultado)
```

Com as equivalências quânticas das operações lógicas usadas nos problemas SAT, foi criado um algoritmo [algoritmo 1] que transforma problemas SAT em um oráculo de Grover. Este algoritmo é dividido em três partes, a primeira é onde se recebem os qubits que representam as variáveis, os qubits que representam as cláusulas, além de um qubit auxiliar.

O primeiro passo é a análise de cada cláusula [linha 3], onde será visto quais variáveis são negativas [linha 7]. Dessa maneira, em cada qubit que representar uma variável negativa deverá ser aplicada a porta *X*. Após isso, deve ser aplicada a operação *OR* nos qubits que representam as variáveis da cláusula sendo analisada [linha 12]. O resultado é colocado em um qubit adicional e esse qubit representa o valor verdade dessa cláusula, dado o valor de entrada dos qubits.

Depois de aplicar essas operações, o resultado de cada cláusula estará em um dos qubit adicional, então é necessário se retirar o resultado de todas elas, aplicando uma porta *multi-controlada-X* em todas [1] [linha 4]. Como o objetivo final é fazer uma marcação nos vetores que atenderem todas as cláusulas, deve-se aplicar a parte do controle em todas elas e o resultado será guardado em um qubit adicional. Esse qubit representa a saída do oráculo e a troca do seu sinal pela aplicação da porta *X* que faz a marcação dos vetores que terão suas amplitudes aumentadas.

Contudo, antes de seguir para a próxima parte de ampliação de amplitudes, é primeiro necessário fazer a descomputação [linha 5], aplicando todas as portas aplicadas anteriormente, menos a *multi-controlada-X*, na ordem contrária. Como explicado na seção 2.2.5, isso serve para retirar os entrelaçamentos entre os qubits que as portas lógicas quânticas colocam, assim limpando os qubits de se interferirem no resto do algoritmo. Após a descomputação, os qubits que representam as variáveis e as cláusulas são enviados para o difusor de fase.

[1] O resto da implementação desse algoritmo foi feito no algoritmo 2

Algorithm 1: Cnf_Oracle

Input : Expressão SAT, qubits $|var\rangle$ e $|cla\rangle$
Output : Primeira metade do oráculo de Grover

```
1  x_state ← True ∀v ∈ Var;
2  X(cla);
3  foreach cláusula c in expressão do
4      list_var ← Lista vazia;
5      foreach variável v in cláusula do
6          if estado de v (positivo ou negativo) for diferente de x_state_var then
7              X(var);
8              x_state_var ← not x_state_var;
9          end
10         list_var ← list_var ∪ var;
11     end
12     Aplicar porta X em |cla_c⟩ usando de controle |list_var⟩;
13 end
```

3.1.3 Implementação do Algoritmo de Grover

A implementação do algoritmo de Grover foi bastante similar a uma implementação padrão do algoritmo, porém foi utilizado o algoritmo 1 internamente para criar os oráculos. Com isso, foi necessário colocar de entrada para o algoritmo a expressão SAT a ser verificada pelo algoritmo. Como explicado na seção anterior, algumas partes do algoritmo 1 foram feitas dentro desse algoritmo, por motivos de simplificar a implementação. Essas são a parte da porta *X* no qubit auxiliar [linha 4] e da descomputação [linha 5]. Dado isso, foi apenas necessário fazer a inicialização dos valores, aplicar as linhas já comentadas acima e depois aplicar o difusor de fase.

Com essa implementação base, obtém-se a maior parte dos elementos do algoritmo de Grover, apenas faltando a decisão de quantas execuções do operador de Grover deverão acontecer. Em uma execução normal do Grover, utiliza-se a fórmula 29, sendo *m* o número de execuções, *N* o número total de elementos da lista e *t* o número de soluções, a fim de saber o número de execuções necessárias para ter uma grande probabilidade de se encontrar uma resposta para o oráculo. Porém, como visto na seção 2.3.2.5, não é possível saber o número de soluções de um problema SAT sem resolvê-lo parcialmente, então, caso fosse tentado descobrir o número de soluções para assim rodar o algoritmo de Grover, isso tornaria a aceleração do algoritmo de Grover inútil, logo foi necessário usar um método para determinar o número de execuções de um número desconhecido de soluções.

3.1.4 Algoritmo para número desconhecido de respostas

O algoritmo [algoritmo 3] se baseia em uma repetição de chutes aleatórios [linha 4], para determinar o número de execuções do iterador de Grover. Portanto, para

Algorithm 2: Grover

Input : (1) Expressão SAT, (2) *it* como o número de iterações, (3) $|var\rangle$ e $|cla\rangle$ são listas de qubits com tamanho do número de variáveis e do número de cláusulas no estado $|0\rangle$, respectivamente, e (4) $|aux\rangle$ é um qubit inicializa no estado $|1\rangle$

Output : Medição do estado quântico no final do algoritmo

```
1 H(|var, aux⟩);
2 for i ← 0 to it do
3 |   Executar CNF_Oráculo(expressão, |var⟩, |cla⟩);
4 |   Aplicar porta X em |aux⟩ usando de controle |cla⟩;
5 |   Aplicar as portas de CNF_Oráculo na ordem contrária para fazer
  |    descomputação;
6 |   Aplicar difusor de fase;
7 end
```

cada iteração do algoritmo, deve-se executar o algoritmo de Grover [linha 5], utilizando esse número gerado como a quantidade de repetições do iterador de Grover, obtendo um resultado final (BOYER *et al.*, 1998). Esse deve ser avaliado se é um resultado possível para o problema [linha 6], o que é algo que pode ser feito em tempo linear com um problema SAT, então isso não causa nenhum aumento significativo no tempo da execução final. Porém, caso o resultado não seja encontrado nessa execução, o algoritmo avança para a próxima iteração, assim escolhendo outro valor. Esse valor é sempre escolhido dado um máximo m, e este aumenta para cada iteração, dessa forma, gerando números cada vez maiores de execuções do algoritmo de Grover.

Assim, é feita uma busca probabilística por vários valores diferentes e incrementais de iterações do algoritmo de Grover. O valor de m irá aumentar até chegar o caso ao valor $\sqrt{2^v}$, onde será rodado mais uma vez com esse valor e, caso não seja encontrado valor, o algoritmo retornará que não foi encontrada resposta para esse problema. Esse valor da última execução é baseado no caso de que exista apenas uma resposta, pois este é o caso com a maior necessidade de execuções do algoritmo. O algoritmo se baseia na ideia de que, caso existam múltiplas respostas possíveis para o problema, é esperado que alguma seja encontrada, mesmo que o valor de iterações não seja o valor exato esperado. Isso acontece pois existe a possibilidade de se medir o resultado correto, mesmo antes de se chegar no número desejado de iterações.

Por conseguinte, é esperado encontrar uma resposta para o problema em até

$$m_{max} = \frac{9}{2}\frac{N}{2\sqrt{(N-t)t}} \tag{30}$$

onde t é o número de respostas possíveis, e $0 < t \leq 3N/4$ (BOYER *et al.*, 1998). No caso de $t > 3N/4$, o número de respostas é tão alto que apenas uma medição qualquer já deve retornar uma resposta possível para o sistema. No caso em que $t = 0$, ou seja, não houver resposta para o problema, é apenas necessário rodar até o final do

algoritmo e retornar que não foi encontrada resposta para essa instância.

Dado esse máximo, é notável que o tempo de execução do algoritmo ainda é limitado por $\mathcal{O}(\sqrt{N/t})$, logo, com a utilização desse algoritmo, é possível se chegar ao resultado sem aumentar a complexidade do algoritmo como um todo, tornando possível utilizá-lo nesse trabalho.

Algorithm 3: Resolverdor SAT

Input : Expressão SAT, onde *v* é o número de variáveis e *c* o número de cláusulas da expressão
Output : Uma possível solução para a expressão SAT

```
1  m ← 1;
2  lambda ← 6/5;
3  while True do
4  |  j ← um valor aleatório entre 1 e m;
5  |  resultado ← Grover(SAT, j, v, c);
6  |  if Verifier(resultado) retornar positivo then
7  |  |  break;
8  |  end
9  |  if m = √(2^v) then
10 |  |  break;
11 |  end
12 |  m ← min(lambda*m, √(2^v))
13 end
```

3.1.5 Algoritmo de avaliação

O algoritmo de avaliação foi feito para fazer avaliações em tempo linear das respostas geradas pelo algoritmo de Grover, e, para funcionar, são necessárias tanto a expressão quanto a resposta a ser testada. A entrada da resposta é dada como uma lista booleana, com tamanho igual ao número de variáveis da instância SAT, onde cada posição dessa lista representa uma das variáveis e o seu valor booleano representa se a variável será negada ou não. O motivo de ser representado assim é pela saída do algoritmo de Grover, onde serão medidos os qubits que representam as variáveis e assim será retornada justamente uma lista boolena que será passada para esse algoritmo 4.

Com essa representação das variáveis e dado que as fórmulas são CNF, portanto os elementos de dentro das cláusulas são separados pela operação *OR*, é apenas necessário passar por cada cláusula [linha 1] e avaliar se algumas das variáveis [linha 2] têm o valor igual à resposta sendo testada [linha 8]. Caso algumas das variáveis seja validada pela entrada, a cláusula é considerada validada e o algoritmo irá seguir para a próxima cláusula que será avaliada [linha 9].

Caso nenhuma das 3 variáveis da cláusula seja validada, o algoritmo retorna *False*, pois é necessário que todas as cláusulas sejam validadas, já que todas são ligadas pela operação *AND*. Já, caso o algoritmo tenha passado por todas as cláusulas e, portanto, validado todas elas, a resposta sendo testada é considerada uma resposta para essa instância SAT e o algoritmo irá retornar *True*.

Algorithm 4: SAT Verifier

```
   Input  : Expressão SAT, res lista binária representando a resposta
   Output : Verdadeiro ou Falso se a resposta valida a fórmula
 1 foreach cláusula c in expressão do
 2     foreach variável v in cláusula do
 3         if v tem o sinal '-' then
 4             sinal ← True;
 5         else
 6             sinal ← False
 7         end
 8         if sinal for igual ao valor de v em res then
 9             Vá para a próxima cláusula;
10         end
11     end
12     else
13         return False;
14 end
15 return True;
```

3.1.6 Implementação dos testes

Após serem feitas essas implementações, foi apenas necessário colocá-las dentro de um algoritmo com o objetivo de fazer os testes. Para isso, foi feito um algoritmo que acessa o arquivo com as instâncias SAT e, para cada uma delas, executa o algoritmo 3 e guarda os resultados, junto com o número de cláusulas e variáveis da instância, em um arquivo.

Algorithm 5: Algoritmo principal

```
   Output : Lista com o resultado para cada teste
 1 lista_sats ← leitura do arquivo de SAT;
 2 foreach SAT S in lista_sats do
 3     resultado ← resultado ∪ Resolvedor SAT(s)
 4 end
```

3.2 EXECUÇÕES DOS ALGORITMOS

Os algoritmos anteriormente apresentados foram então implementados e executados, porém foram necessárias algumas restrições sobre a execução, devidas ao

simulador. A principal limitação foi o número de qubits que podiam ser utilizados, então foi colocado um limite de 27 qubits, mais um auxiliar, na execução, pois, com mais qubits, o aumento de tempo da execução seria muito maior.

Esse limite foi estabelecido com a ideia de escolher um número grande o suficiente para poder executar instâncias SAT que se encontram na região do crossover point, porém que não demorassem tanto para executarem. Para determinar esse limite, foi necessário levar dois aspectos em consideração, primeiro a quantidade de instâncias geradas que tenham valores distintos na relação *clausula/variavel*, isso é importante, pois, como explicado na seção 2.1.1, essa relação é necessária para se metrificar a dificuldade de resolver uma instância SAT, usando algoritmos clássicos. Logo, é desejável gerar várias instâncias com valores distintos de *cláusula/variável*.

	# Qubits					
Regiões Cla/Var	25	26	27	28	29	30
0.0,0.5	21	23	25	27	30	32
0.5,1.0	39	43	47	51	55	60
1.0,1.5	22	24	26	29	31	34
1.5,2.0	14	15	17	18	20	22
2.0,2.5	9	10	11	12	13	14
2.5,3.0	6	7	8	9	9	10
3.0,3.5	4	5	6	6	6	7
3.5,4.0	4	4	4	4	5	6
4.0,4.5	2	2	3	4	4	4
4.5,5.0	2	2	2	2	3	4

Tabela 1 – Número de frações alcançáveis em cada região, dado o número de qubits.

Contudo, existe uma limitação das frações alcançáveis, dado um valor máximo de Qubits. Para se visualizar essa limitação, foi considerado que

$$\#cla + \#var \leq \#Qubits \tag{31}$$

onde # representa a quantidade de cada elemento. Essa relação é verdadeira, pois é utilizado 1 qubit para cada cláusula e 1 qubit para cada variável, logo, para se chegar na limitação de 27 Qubits, é necessário limitar o número de cláusulas e variáveis das instâncias SAT, seguindo essa equação. Então, para se chegar em números de *cla/var* próximos de 5, é necessário um número de cláusulas, *#cla*, que seja 5 vezes maior que o número de variáveis, $\#cla \approx 5 \times \#var$, e ainda os dois valores somados devem ser menores que 27, $\#cla + var \leq 27$.

A tabela 1 demonstra essa limitação de números alcançáveis, dado um valor máximo de qubits. Como um dos focos do trabalho é analisar instâncias SAT que tenham essa relação próxima de 4.2, é vantajoso escolher um número de Qubits que tenha uma boa quantidade na região *[4.0,4.5]*. Na tabela, é possível ver que, com 27

Qubits, existe um aumento de 2 para 3 números de relações alcançáveis, porém, com 28 qubits, seria possível gerar até 4 instâncias com relações diferentes. O motivo de não ter sido escolhido um número maior se deve à limitação temporal da execução, já que ainda era necessário um qubit auxiliar também na execução.

Todas execuções com essa quantidade de qubits têm uma demora muito grande para serem simuladas, pois, a cada novo qubit, existe um crescimento exponencial de algumas operações, então é importante não escolher um número muito alto de qubits. Assim, foi feita a escolha arbitrária de 3 instâncias na região a ser analisada.

Essa escolha é enfatizada pelo tempo de execução dos testes, em que, para cada iteração onde se resolvem todos os SAT gerados, foram necessárias, em média, 34.5 horas para executar em um computador com as especificações da tabela 2.

OS	Manjaro Linux x86_64
Kernel	5.15.49-1-MANJARO
CPU	Intel i3-6100 (4) @ 3.700GHz
GPU	NVIDIA GeForce GTX 1060 3GB
Memory	7887MiB

Tabela 2 – Especificações da máquina utilizada.

Com o número de qubits escolhido, foram então geradas as instâncias SAT, onde foi colocado um máximo de 20 instâncias por região, já que as regiões com menores valores acabam tendo várias combinações, como, por exemplo, a região [0.5,0.1] que tem 47 possíveis combinações, mas não apresenta necessidade de se criarem tantas instâncias, já que isso aumentaria bastante o tempo de execução.

As instâncias geradas [2] foram então passadas por um algoritmo de validação de satisfabilidade (MENG, 2017). O programa recebe um SAT no formato DIMACS e utiliza uma busca *brute-force*, testando todas as possibilidades, para retornar se aquela instância é satisfazível ou não. Essa análise foi importante para poder observar a capacidade do algoritmo de Grover de realmente resolver o problema SAT e ter certeza de que as instâncias que não foram resolvidas não têm solução. Porém, essa análise não deveria ser utilizada em um caso real, já que o objetivo é utilizar o ganho do algoritmo de Grover para resolver instâncias SAT.

Dessa forma, foi possível ver que, de todo o conjunto, apenas uma das fórmulas é insatisfazível. Esse resultado é esperado já que as instâncias são muito pequenas, então não é muito provável que uma instância pequena gerada de maneira randômica seja insatisfazível.

Com as instâncias geradas e analisadas, foi então colocada cada uma para rodar no algoritmo 4. Esse algoritmo irá retornar a soma do número de iterações que foram executadas para cada problema, que será guardado em um arquivo de

[2] SATs gerados podem ser vistos no link https://github.com/Teyal/TCC

saída, para poder depois gerar gráfico e análises sobre os resultados. Esse número é o acúmulo de cada número de iterações testado em cada iteração do algoritmo 3. Também será retornado se, após todas as tentativas, foi encontrada uma resposta para a instância sendo analisada. Isso será utilizado para saber se a implementação como um todo consegue resolver tais problemas e poder analisar em quais tipo de instâncias foram encontrados mais problemas de resolução.

Como o algoritmo é baseado em várias aleatoriedades, tanto do algoritmo 3, como das características quânticas de probabilidade de se retornar cada resposta, foi executado múltiplas vezes esse teste. Assim, se torna possível concluir com mais certeza sobre a capacidade do algoritmo de Grover de resolver as instâncias SAT e também poder ter melhor noção da média de iterações necessárias para resolver cada tipo de instância.

4 ANÁLISE DOS RESULTADOS & CONCLUSÃO

Os itens a seguir apresentarão as análises e conclusões do projeto. As análises serão feitas sobre os resultados e suas limitações. Já a seção de conclusão será sobre quais objetivos foram alcançados, bem como algumas perspectivas de trabalhos futuros que poderiam ser feitos a partir desse projeto.

4.1 ANÁLISE DOS RESULTADOS

A partir dos algoritmos implementados, foi construído o gráfico 8, fazendo uma relação entre a divisão de cláusulas e variáveis de uma instância SAT, com o número de iterações necessárias para se encontrar o resultado dentro da execução dos algoritmos. Além disso, as execuções são marcada como azuis, caso tenha sido encontrada uma resposta para aquela instância, e vermelho, caso não tenha sido encontrada.

Esse gráfico tinha o objetivo de tentar visualizar a figura 1 num contexto quântico, onde o número de iterações é usado como métrica para a dificuldade de se resolver uma instância. Porém, é perceptível que há uma diferença entre as duas plotagens. Isso se deve, em grande parte, ao tamanho das instâncias geradas, onde no artigo (BOYER *et al.*, 1998) foram utilizadas instâncias de 200 variáveis, já as instâncias desse trabalho tiveram um máximo de 17 variáveis.

Para executar instâncias de 200 variáveis no algoritmo de Grover, seriam necessárias mais de 1000 qubits para conseguir uma relação de cláusula/variável mais que 4. Isso demonstra como ainda são necessários avanços nos simuladores ou até computadores quânticos de propósito geral com mais de 1000 qubits, para fazer uma comparação melhor entre a dificuldade de resolver as instâncias SAT num computador clássico e um quântico.

Ainda analisando a plotagem, é possível perceber que alguns problemas não tiveram respostas, após a execução do algoritmo. A motivação disso não é óbvia apenas com essa única execução, já que todas instâncias, menos uma, são satisfazíveis, então era de se esperar que todas as instâncias, possíveis de serem resolvidas, o fossem.

Uma interpretação possível para essa dificuldade de resolver algumas instâncias é a característica probabilística dos algoritmos, tanto das escolhas aleatórias de execuções do algoritmo 3, quanto das propriedades quânticas de probabilidade das medições. Assim sendo, foram feitas múltiplas execuções e então compiladas em um novo gráfico.

Ao se analisar o gráfico resultante, visualizado na figura 9, onde foram executadas 55 vezes o mesmo algoritmo, é possível notar que os problemas que não tiveram respostas em uma das iterações conseguiram encontrar uma resposta em outra iteração. Isso demonstra que os algoritmos implementados têm um funcionamento correto,

dado um grau de aleatoriedade.

Uma exceção dessa análise é a instância com a relação de cláusulas/variáveis = 3.4, onde nas 4 execuções não foi encontrada nenhuma resposta. Porém esse resultado é condizente com o esperado, pois essa instância é a única que realmente não tem resposta para a equação lógica.

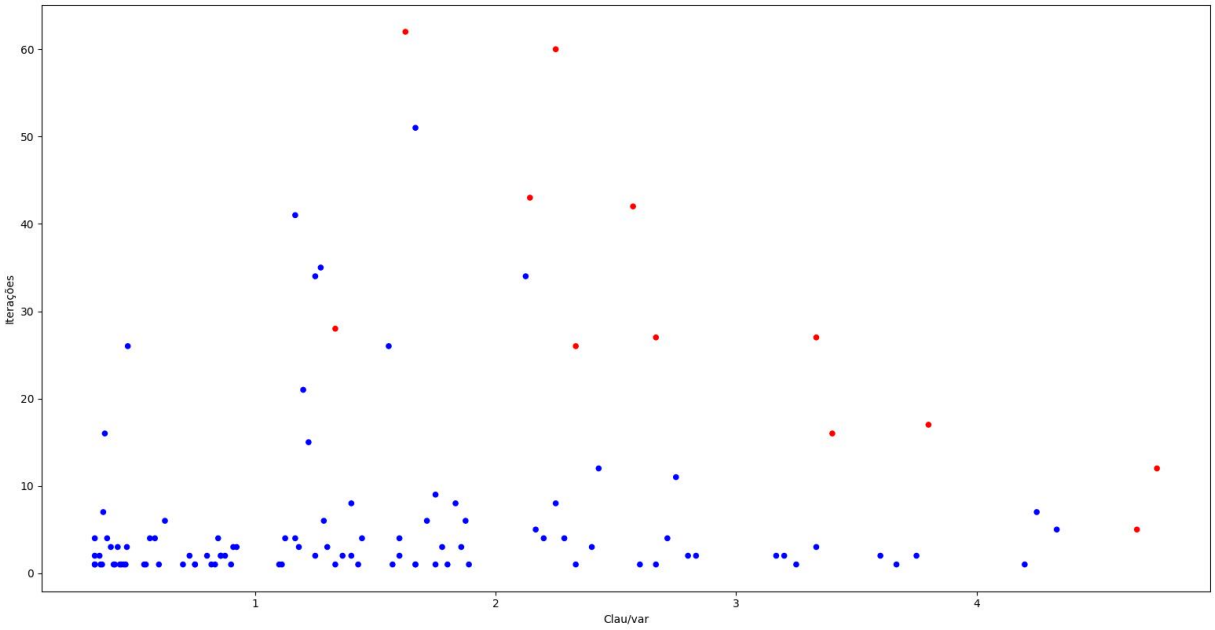

Figura 8 – Gráfico mostrando a relação de Cláusulas/Variáveis com o número de iterações necessárias para chegar ao resultado com 1 iteração.

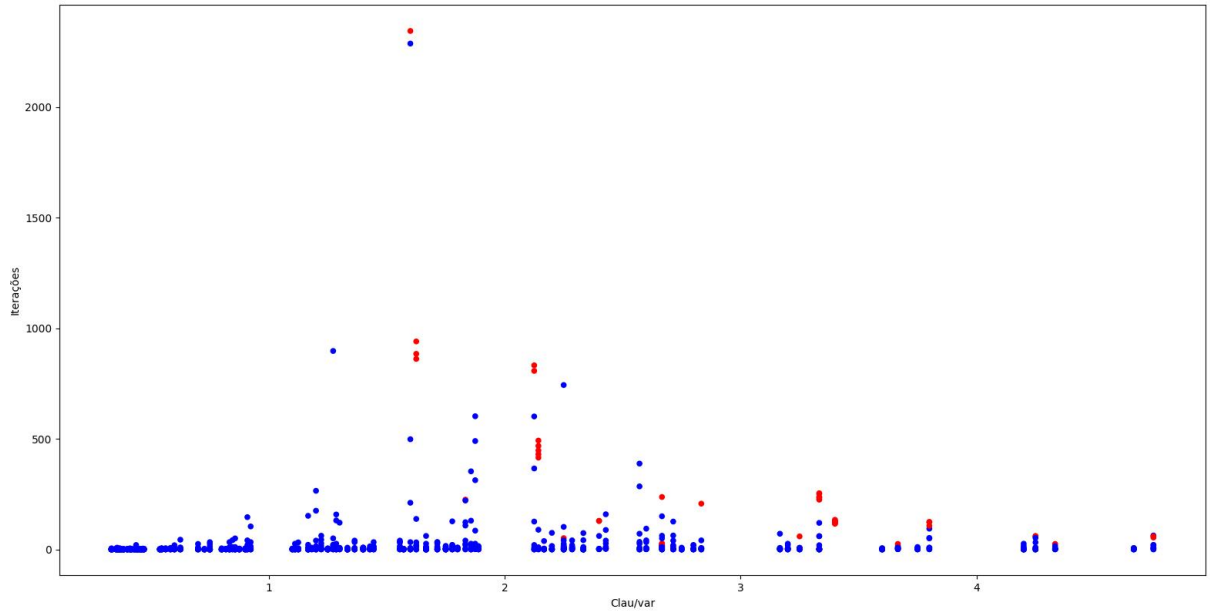

Figura 9 – Gráfico mostrando a relação de Cláusulas/Variáveis com o número de iterações necessárias para chegar ao resultado com 55 iterações.

Regiões Cla/Var	Média de iterações
[0.0,0.5]	2.098663102
[0.5,1.0]	5.032323232
[1.0,1.5]	18.95284091
[1.5,2.0]	57.98787879
[2.0,2.5]	60.24949495
[2.5,3.0]	33.79480519
[3.0,3.5]	46.36
[3.5,4.0]	18.03636364
[4.0,4.5]	15.44848485
[4.5,5.0]	15.15454545

Tabela 3 – Média de iterações por regiões

4.2 CONCLUSÃO

Com esse projeto, foi possível fazer uma investigação sobre o crossover point dentro de um contexto quântico e, para chegar a essa investigação, foi necessário se alcançar alguns objetivos propostos.

O objetivo 1, de fazer a implementação dos algoritmos, foi alcançado fazendo a análise e implementação de vários algoritmos propostos. E, pelos resultados mostrados, as implementações foram todas realizadas com êxito, tanto os algoritmos base, como as interações que eles têm entre si.

Com as implementações corretas, as execuções tiveram de sofrer algumas limitações por escolha para poderem ser simuladas em tempo hábil, porém todos os algoritmos estão aptos a serem testados com maior números de qubits, o maior limitador, caso sejam realizadas melhorias no simulador utilizado para acelerar o processamento de qubits.

Dada a figura 9, é possível concluir que o objetivo 2, de verificar a capacidade do algoritmo de Grover de resolver instâncias SAT, foi alcançado demonstrando como o algoritmo de Grover é capaz de resolver todos os tipos de instâncias SAT. Porém, com a tabela 2, é notável a existência de uma maior dificuldade em algumas regiões do que outras e, das regiões com maior dificuldade, o ponto de crossover não está incluído. Isso pode ser devido ao tamanho das instâncias ser muito pequeno, uma vez que as instâncias nessa região têm, em média, apenas 4 variáveis.

Com instâncias assim, mesmo com instâncias pertencendo ao crossover point, não é possível visualizar a complexidade esperada desses problemas. Já a motivação do porquê outras regiões apresentam dificuldades não é clara, a partir desses resultados, demonstrando que o objetivo 4 não foi completamente alcançado e poderia ser mais extensivamente avaliada em trabalhos futuros.

Verificando o objetivo 3, é perceptível que a inclusão de outros algoritmos não criou um aumento significativo na complexidade do algoritmo de Grover, que ainda tem

como limitante $\mathcal{O}(\sqrt{N})$. Isso se deve pelo algoritmo 1 ser executado em tempo $\mathcal{O}(\sqrt{N})$ porém é apenas necessário executar uma vez por análise então apenas se soma na complexidade. Já o algoritmo 3 conseguir chegar numa resposta ainda limitado por $\mathcal{O}(\sqrt{N})$ e o algoritmo 4 ser também executado em tempo linear. Assim, demonstrou-se que o algoritmo de Grover realmente tem uma utilidade para resolver instâncias SAT.

Como as instâncias geradas são muito pequenas e todas as execuções utilizaram de um simulador quântico, o tempo para resolver um problema nesse experimento é muito maior que o tempo de resolver em um computador clássico. Isso evidencia como ainda não é possível tirar algum tipo de vantagem sobre a execução do algoritmo de Grover em relação a algoritmos clássicos de resolução de SAT, assim alcançando o objetivo 5. Uma metrificação possível da dificuldade real do tempo que levaria para resolver essas instâncias em um computador quântico é a quantidade de iterações que foram necessárias, porém esse dado é facilmente influenciado por várias questões dessa implementação. A principal é o tamanho das instâncias, que, por serem pequenas, normalmente apresentam várias respostas possíveis e são na maioria satisfazíveis. Esses fatos podem ter diminuído a média de iterações para a avaliação de cada instância, portanto não é possível tirar conclusões enfáticas sobre como realmente seriam as execuções de problemas maiores em um computador real. Além de que, em um computador real, poderiam existir outros tipos de elementos que poderiam dificultar uma execução, como complexidade de implementar portas *multi-control*, por exemplo, tudo isso tornando mais complexo de se fazer uma análise realista de comparação do tempo de execução de instâncias SAT em um algoritmo clássico com um algoritmo quântico.

Como trabalhos futuros, também poderiam ser analisadas outras maneiras de representar uma instância SAT em um oráculo, pois a quantidade de qubits necessários aumenta junto com o número de variáveis e cláusulas, de acordo com essa implementação. Caso fossem encontradas outras maneiras de se utilizar menos qubits, isso tornaria mais fácil de se executarem instâncias maiores e seria possível chegar a novos resultados.

Outra possibilidade de trabalho futuro seria a avaliação mais detalhada dos casos onde o algoritmo não encontrou resposta, mesmo elas existindo. Essa análise poderia ser tanto sobre a corretude da implementação, quanto uma análise sobre a amplitude das possíveis respostas para aquela instância. A análise avaliaria se as respostas não foram escolhidas por causa da aleatoriedade de resposta do algoritmo de Grover, ou se não foram selecionadas devido a uma má escolha de valores no algoritmo 3. A partir disso, será possível avaliar a utilidade dessa combinação de algoritmos e concluir se o algoritmo 3 é adequado para resolver o problema SAT.

REFERÊNCIAS

BICHSEL, Benjamin; BAADER, Maximilian; GEHR, Timon; VECHEV, Martin. Silq: A High-Level Quantum Language with Safe Uncomputation and Intuitive Semantics. *In*: PROCEEDINGS of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation. London, UK: Association for Computing Machinery, 2020. (PLDI 2020), p. 286–300. DOI: 10.1145/3385412.3386007. Disponível em: https://doi.org/10.1145/3385412.3386007.

BOYER, Michel; BRASSARD, Gilles; HØYER, Peter; TAPP, Alain. Tight Bounds on Quantum Searching. **Fortschritte der Physik**, Wiley, v. 46, n. 4-5, p. 493–505, jun. 1998. ISSN 1521-3978. DOI: https://doi.org/10.48550/arXiv.quant-ph/9605034. Disponível em: http://dx.doi.org/10.1002/(SICI)1521-3978(199806)46:4/5%3C493::AID-PROP493%3E3.0.CO;2-P.

COOK, Stephen A. The Complexity of Theorem-Proving Procedures. *In*: PROCEEDINGS of the Third Annual ACM Symposium on Theory of Computing. Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971. (STOC '71), p. 151–158. DOI: 10.1145/800157.805047. Disponível em: https://doi.org/10.1145/800157.805047.

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. **Introduction to Algorithms, Third Edition**. 3rd. [*S.l.*]: The MIT Press, 2009. ISBN 0262033844.

CRAWFORD, James M.; AUTON, Larry D. Experimental results on the crossover point in random 3-SAT. **Artificial Intelligence**, v. 81, n. 1, p. 31–57, 1996. Frontiers in Problem Solving: Phase Transitions and Complexity. ISSN 0004-3702. DOI: https://doi.org/10.1016/0004-3702(95)00046-1. Disponível em: https://www.sciencedirect.com/science/article/pii/0004370295000461.

DIRAC, P. A. M. A new notation for quantum mechanics. **Proceedings of the Cambridge Philosophical Society**, v. 35, n. 3, p. 416, jan. 1939. DOI: 10.1017/S0305004100021162.

FORTNOW, Lance. The Status of the P versus NP Problem. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 52, n. 9, p. 78–86, set. 2009. ISSN 0001-0782. DOI: 10.1145/1562164.1562186. Disponível em: https://doi.org/10.1145/1562164.1562186.

GROVER, Lov K. **A fast quantum mechanical algorithm for database search**. [*S.l.*: *s.n.*], 1996. eprint: arXiv:quant-ph/9605043.

HOPCROFT, John E.; MOTWANI, Rajeev; ULLMAN, Jeffrey D. **Introduction to Automata Theory, Languages, and Computation (3rd Edition)**. USA: Addison-Wesley Longman Publishing Co., Inc., 2006. ISBN 0321455363.

MACK, Chris A. Fifty Years of Moore's Law. **IEEE Transactions on Semiconductor Manufacturing**, v. 24, n. 2, p. 202–207, 2011. DOI: 10.1109/TSM.2010.2096437.

MENG, Yibai. **DPLL-SAT-solver**. [*S.l.*]: GitHub, 2017. https://github.com/YibaiMeng/DPLL-SAT-solver.

MITCHELL, David; SELMAN, Bart; LEVESQUE, Hector. Hard and Easy Distributions of SAT Problems. *In*: PROCEEDINGS of the Tenth National Conference on Artificial Intelligence. San Jose, California: AAAI Press, 1992. (AAAI'92), p. 459–465.

NIELSEN, Michael A.; CHUANG, Isaac L. **Quantum Computation and Quantum Information: 10th Anniversary Edition**. 10th. USA: Cambridge University Press, 2011. ISBN 1107002176.

PRESKILL, John. Quantum Computing in the NISQ era and beyond. **Quantum**, Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, v. 2, p. 79, ago. 2018. ISSN 2521-327X. DOI: 10.22331/q-2018-08-06-79. Disponível em: https://doi.org/10.22331/q-2018-08-06-79.

ROSA, Evandro Chagas Ribeiro da; SANTIAGO, Rafael de. Ket Quantum Programming. **J. Emerg. Technol. Comput. Syst.**, Association for Computing Machinery, New York, NY, USA, a ser publicado, 2021. ISSN 1550-4832. DOI: 10.1145/3474224.

SHOR, P.W. Algorithms for quantum computation: discrete logarithms and factoring. *In*: PROCEEDINGS 35th Annual Symposium on Foundations of Computer Science. [*S.l.*: *s.n.*], 1994. P. 124–134. DOI: 10.1109/SFCS.1994.365700.

SIPSER, Michael. **Introduction to the Theory of Computation**. Third. Boston, MA: Course Technology, 2013. ISBN 113318779X.

ANEXO A – CÓDIGO FONTE

Código usado está disponível em: https://github.com/Teyal/TCC

ANEXO B – ARTIGO

Investigação do crossover point do problema da satisfação booleana utilizando o algoritmo quântico de Grover

Teo H. Gallarza[1], **Jerusa Marchi**[1], **Evandro C. Rosa**[1]

[1]Instituto de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

{teo.gallarza,jerusa.marchi}@ufsc.br, ev.crr97@gmail.com

Abstract. *In the crossover point, the instances of the boolean satisfability problem (SAT) are harder to resolve. This work investigates the behaviour of the Grover algorithm, which finds an element in an unordered list with a asymptotic of order of $\sqrt{N}$ - N being the number of elements. The implementation use the language Ket and the simulator of quantum computing QuBix. It was used SAT instances randomly generated with a limitation in the number of qubits. It was also implemented the algorithm to convert the instances into Grover oracles and another to iterate the Grover algorithm with an unkown number of answers. The results didn't left a clear vision of the real complexity of this execution.*

Resumo. *No crossover point, as instâncias do problema de satisfação booleana (SAT) são mais dificilmente resolvidas. Este trabalho investiga seu comportamento no algoritmo de Grover, que encontra um elemento em uma lista desordenada e de comportamento assintótico da ordem de $\sqrt{N}$ — N é a quantidade de elementos. A implementação usou a linguagem Ket e o simulador de computação quântica QuBox. Faz-se uso de instâncias SAT geradas aleatoriamente e limitadas no número de qubits. Implementaram-se também um algoritmo para converter tais instâncias em oráculos de Grover e outro, para iterá-lo quando o número de respostas para o oráculo é desconhecido. Não se obteve uma visão clara da real complexidade dessa execução.*

1. Introdução

Desde a criação dos computadores, muitos estudos têm sido dedicados ao aperfeiçoamento de seus componentes e algoritmos, com o intuito de conseguir executar os programas de maneira cada vez mais rápida. O uso de silício e código binário é a base da computação clássica e, apesar de suas limitações, principalmente a necessidade de dissipação de energia, os circuitos integrados vinham sendo rapidamente melhorados, chegando a duplicar sua capacidade a cada 2 anos [Mack 2011]. O atingimento do limite físico de calor das peças do computador criou interesse em várias novas áreas de pesquisa.

Com esse interesse e os avanços científicos feitos na área da física quântica, foi criada uma nova área de pesquisa chamada *Computação Quântica*. A área de computação quântica já teve vários avanços teóricos importantes, por exemplo, a criação de algoritmos, como o de Grover [Grover 1996], que apresenta ganhos quadrático se comparados com suas contrapartidas clássicas.

Porém ainda não foi possível verificar os ganhos esperados da computação quântica, uma vez que os computadores quânticos atuais não suportam a execução de grandes circuitos quânticos. Isso ocorre pois as máquinas atuais, chamadas de NISQ (Noisy Intermediate-Scale Quantum) [Preskill 2018], ainda não têm a capacidade de executar problemas grandes e complexos como o deste trabalho. Com o objetivo de superar essas adversidades, foi preciso utilizar um simulador para a execução dos algoritmos. Contudo, ao se utilizar um simulador, não é possível ter efetivamente a aceleração esperada da execução dos algoritmos quânticos. Isso dificulta a medição verídica do custo computacional da execução de problemas, portanto foram utilizados outros tipos de métricas para fazer as avaliações necessárias.

Após a revisão, é abordado o problema do trabalho, a relação do problema SAT, mais especificamente o *crossover point*, com o oráculo de Grover, demonstrando como é possível um algoritmo de Grover resolver problemas SAT. Essa relação é baseada na relação entre instâncias SAT e um oráculo quântico utilizado no algoritmo de Grover. São então explicadas as implementações dos algoritmos produzidos para esse trabalho, são eles: o algoritmo de Grover, o algoritmo de geração de instâncias do problemas SAT, o algoritmo de transformação de instâncias SAT em oráculos quânticos e o algoritmo de validação de entradas para instâncias SAT. Também é apresentado um algoritmo clássico para o teste da satisfação de instâncias [Meng 2017], utilizado para auxiliar a identificar as instâncias satisfazíveis.

Ao final do artigo, são apresentados os gráficos obtidos, buscando demonstrar a complexidade e a capacidade do algoritmo de Grover de resolver as instâncias SAT geradas. Com esses gráficos, será analisado o efeito do *crossover point* sobre o algoritmo de Grover. A partir desses gráficos, também serão analisadas as limitações atuais de se tentar aplicar o algoritmo de Grover em casos reais, tentando testar ao máximo o simulador utilizado, para ver quais a limitações que ele apresenta. Também será analisado se o algoritmo de Grover, nesse contexto, é útil para resolver instâncias SAT.

O objetivo principal deste projeto de pesquisa é investigar o funcionamento do algoritmo de Grover, quando utilizado para resolver instâncias do problema SAT próximas do *crossover point*. Desse modo, estabelecendo relações entre a complexidade das instâncias SAT sendo executadas em um algoritmo clássico em relação a um algoritmo quântico.

Para isso, é necessário fazer uma implementação do algoritmo de Grover que consiga resolver instâncias do Problema SAT. Para isso, além da implementação do algoritmo de Grover, é necessário implementar um algoritmo de geração de instâncias SAT junto com um algoritmo que transforma as instâncias geradas em oráculos quânticos. Dessa maneira será possível executar o problema SAT dentro o algoritmo de Grover. A geração de instâncias deve seguir as limitações do simulador, logo é necessário descobrir este limite, para, assim, tentar ao máximo gerar as instâncias do Problema SAT que estejam próximas ao *crossover point*. Com essas implementações, serão feitas análises sobre os resultados encontrados que serão comparados com os resultados de computadores clássicos, para assim saber qual o efeito do *crossover point* em relação ao algoritmo de Grover.

2. Desenvolvimento

Para se fazer a avaliação da dificuldade de resolução das instâncias SAT pelo algoritmo de Grover, estas serão transformadas em oráculos. Para tanto é necessário implementar, além do algoritmo de Grover, algoritmos auxiliares para criação e transformação de instâncias SAT em oráculos.

O algoritmo de criação de SAT é o mais simples, pois ele deve criar instâncias de 3SAT de maneira aleatória, para tentar explorar diferentes taxas de cláusulas por variáveis sem ter uma heurística para a criação. Logo, o algoritmo deve receber um número de variáveis e cláusulas e gerar uma lista aleatória de 3 variáveis por cláusula no formato DIMACS. Esse processo é feito para cada cláusula e, após todas serem geradas, a instância SAT final é armazenada em um arquivo juntamente com todas as outras instâncias geradas, que forma o arquivo de instâncias a serem analisadas.

Após todas as instâncias serem criadas, é necessário então torná-las possíveis de serem executadas no algoritmo de Grover. Para fazer isso, foi necessário implementar um algoritmo que transforma as instâncias em oráculos quânticos. Esse algoritmo segue uma lógica de equivalência das operações lógicas da instância SAT com portas lógicas quânticas. As operações de *AND* foram transformadas na porta quântica *multiple-control-not*, já a operação *OR* não tem uma equivalência tão direta com uma porta, então foi feita uma transformação, utilizando a lei de De Morgan, $(X \vee Y) = \neg(\neg X \wedge \neg Y)$, para poder utilizar a mesma equivalência da porta *AND* com as portas *OR*, apenas necessitando colocar algumas negações nas entradas e no qubit de saída. Com esse algoritmo é necessário tanto um qubit para cada variável, para assim medir qual é a resposta do sistema, quanto um qubit para cada cláusula, para poder armazenar o resultado de todas as avaliações das cláusulas e depois avaliar todas elas em conjunto para chegar ao resultado final. Como é necessário fazer a avaliação de todas juntas, é necessário também mais um qubit onde será armazenada a resposta final do oráculo.

Após feita essa transformação do problema SAT em oráculo quântico, ele é colocado para executar dentro do algoritmo de Grover. O algoritmo de Grover recebe a fórmula a ser avaliada e um número de iterações, *i*, e com isso faz a inicialização dos qubits e depois executa i vezes a função de transformação de instâncias SAT em oráculo, aplicando depois a sua descomputação e o difusor de fases. Com isso será gerado um sistema de portas lógicas que, ao ser executado, retornará uma possível resposta para o problema, dado que o número de iterações seja escolhido corretamente. Porém, existe um problema sobre essa escolha, pois, ao se gerar uma instância SAT da maneira aleatória, não é possível saber o número de respostas para o problema sem resolvê-lo previamente, então foi necessária a utilização de outro algoritmo para saber quantas iterações de Grover serão executadas.

Esse outro algoritmo foi baseado na descrição do algoritmo de número desconhecido de soluções [Boyer et al. 1998], onde é tentado encontrar a resposta para o problema mesmo sem saber o número de iterações correto. Para se chegar ao valor, o algoritmo executa várias vezes o algoritmo de Grover, onde cada vez é escolhido um valor diferente para o número de iterações, assim esperando que, em uma das execuções, seja encontrado o valor esperado. A ideia do algoritmo se baseia num cálculo probabilístico de que, caso existam muitas respostas para o sistema, escolhendo aleatoriamente o número de iterações, existe uma chance muito alta de se medir o valor correto. E, caso tenham poucos

valores, depois de se executarem vários números aleatórios de iterações, é feita mais uma execução como se tivesse apenas um resultado. Mesmo todas essas execuções adicionais, o algoritmo garante que o custo computacional de execução será ainda $\mathcal{O}(\sqrt{N/t})$, assim não aumentando a complexidade de maneira significativa. Além disso, depois de executar ele todo, existe uma probabilidade alta de se encontrar uma resposta para o sistema, porém, para saber se a resposta foi realmente encontrada, o algoritmo necessitou de um algoritmo de avaliação das respostas. Isso se deve porque o algoritmo de Grover sempre irá retornar uma resposta, não sabendo avaliar se a resposta realmente valida o sistema modelado no oráculo, então essa avaliação da resposta se torna necessária. Felizmente o problema de satisfação boolena, por ser NP-completo, tem um tempo de avaliação linear, logo essas avaliações podem ser feitas sem aumentar a complexidade do código.

O algoritmo de avaliação de uma possível resposta, dada uma instância SAT, foi criado com a relação de avaliar resposta do algoritmo Grover em mente. Ele recebe a instância SAT e a resposta da execução de Grover como entrada, fazendo uma equivalência com o vetor de saída dos qubits que representam as variáveis no algoritmo de Grover com as entradas da instância SAT. Com essa relação em mente, é feita uma análise de cada cláusula, onde é visto quais as 3 variáveis se encontram naquela cláusula e é verificado se algumas delas aparece com o mesmo sinal, representado como 0 = positivo e 1 = negativo no vetor booleano), assim validando ou não a cláusula. Caso algumas das cláusulas não seja validada, a entrada analisada não é uma resposta daquela instância, porém, caso todas sejam validadas, ela é uma resposta. Esse resultado é retornado ao algoritmo anterior, para saber se é necessário fazer mais análises ou se o resultado já foi encontrado.

Com todos esses algoritmos implementados, foi apenas necessário executá-los, para se chegar às conclusões do problema. Ao final das execuções, os resultados foram guardados em arquivos e então foram criadas tabelas e gráficos para se fazer análise sobre os resultados e assim fazer uma reavaliação em relação aos objetivos alcançados por este trabalho.

3. Análise dos Resultados & Conclusão

Os itens a seguir apresentarão as análises e conclusões do projeto. As análises serão feitas sobre os resultados e suas limitações. Já a seção de conclusão será sobre quais objetivos foram alcançados, bem como algumas perspectivas de trabalhos futuros que poderiam ser feitos a partir desse projeto.

3.1. Análise dos resultados

A partir dos algoritmos implementados, foi construído o gráfico 1, fazendo uma relação entre a divisão de cláusulas e variáveis de uma instância SAT, com o número de iterações necessárias para se encontrar o resultado dentro da execução dos algoritmos. Além disso, as execuções são marcada como azuis, caso tenha sido encontrada uma resposta para aquela instância, e vermelho, caso não tenha sido encontrada.

Esse gráfico tinha o objetivo de tentar visualizar a figura do crossover point [Crawford and Auton 1996] num contexto quântico, onde o número de iterações é usado como métrica para a dificuldade de se resolver uma instância. Porém, é perceptível que há uma diferença entre as duas plotagens. Isso se deve, em grande parte, ao tamanho

das instâncias geradas, onde no artigo [Boyer et al. 1998] foram utilizadas instâncias de 200 variáveis, já as instâncias desse trabalho tiveram um máximo de 17 variáveis.

Para executar instâncias de 200 variáveis no algoritmo de Grover, seriam necessárias mais de 1000 qubits para conseguir uma relação de cláusula/variável mais que 4. Isso demonstra como ainda são necessários avanços nos simuladores ou até computadores quânticos de propósito geral com mais de 1000 qubits, para fazer uma comparação melhor entre a dificuldade de resolver as instâncias SAT num computador clássico e um quântico.

Ainda analisando a plotagem, é possível perceber que alguns problemas não tiveram respostas, após a execução do algoritmo. A motivação disso não é óbvia apenas com essa única execução, já que todas instâncias, menos uma, são satisfazíveis, então era de se esperar que todas as instâncias, possíveis de serem resolvidas, o fossem.

Uma interpretação possível para essa dificuldade de resolver algumas instâncias é a característica probabilística dos algoritmos, tanto das escolhas aleatórias de execuções do algoritmo de número desconhecido de respotas, quanto das propriedades quânticas de probabilidade das medições. Assim sendo, foram feitas múltiplas execuções e então compiladas em um novo gráfico.

Ao se analisar o gráfico resultante, visualizado na figura 2, onde foram executadas 55 vezes o mesmo algoritmo, é possível notar que os problemas que não tiveram respostas em uma das iterações conseguiram encontrar uma resposta em outra iteração. Isso demonstra que os algoritmos implementados têm um funcionamento correto, dado um grau de aleatoriedade.

Uma exceção dessa análise é a instância com a relação de cláusulas/variáveis = 3.4, onde nas 4 execuções não foi encontrada nenhuma resposta. Porém esse resultado é condizente com o esperado, pois essa instância é a única que realmente não tem resposta para a equação lógica.

3.2. Conclusão

Com esse projeto, foi possível fazer uma investigação sobre o crossover point dentro de um contexto quântico e, para chegar a essa investigação, foi necessário se alcançar alguns objetivos propostos.

O objetivo de fazer a implementação dos algoritmos, foi alcançado fazendo a análise e implementação de vários algoritmos propostos. E, pelos resultados mostrados, as implementações foram todas realizadas com êxito, tanto os algoritmos base, como as interações que eles têm entre si.

Com as implementações corretas, as execuções tiveram de sofrer algumas limitações por escolha para poderem ser simuladas em tempo hábil, porém todos os algoritmos estão aptos a serem testados com maior números de qubits, o maior limitador, caso sejam realizadas melhorias no simulador utilizado para acelerar o processamento de qubits.

Dada a figura 2, é possível concluir que o objetivo de verificar a capacidade do algoritmo de Grover de resolver instâncias SAT, foi alcançado demonstrando como o algoritmo de Grover é capaz de resolver todos os tipos de instâncias SAT. Porém, com

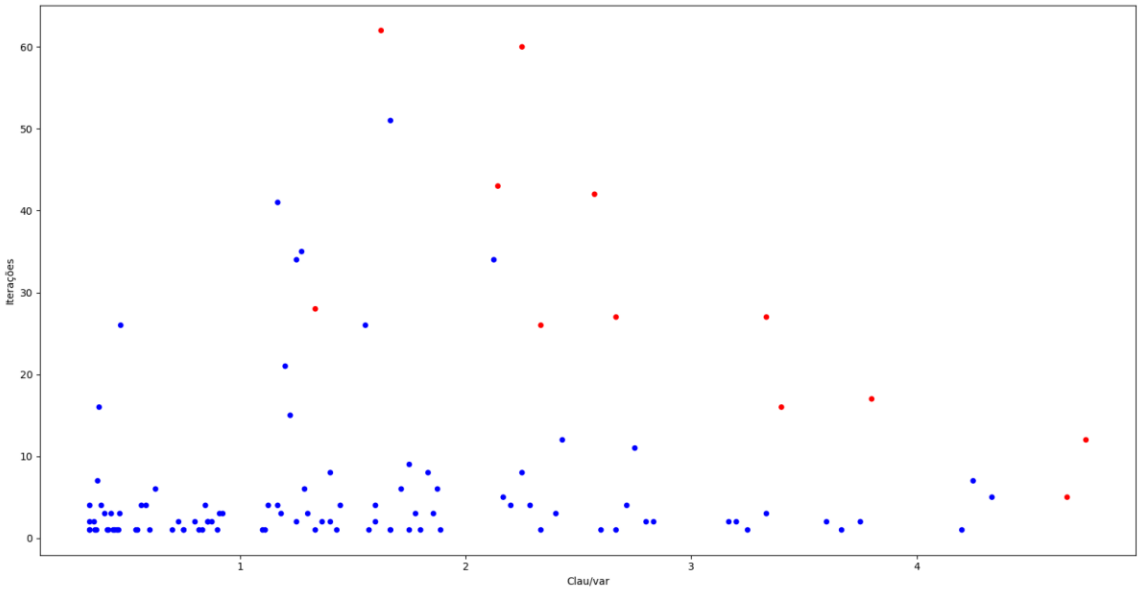

Figure 1. Gráfico mostrando a relação de Cláusulas/Variáveis com o número de iterações necessárias para chegar ao resultado com 1 iteração.

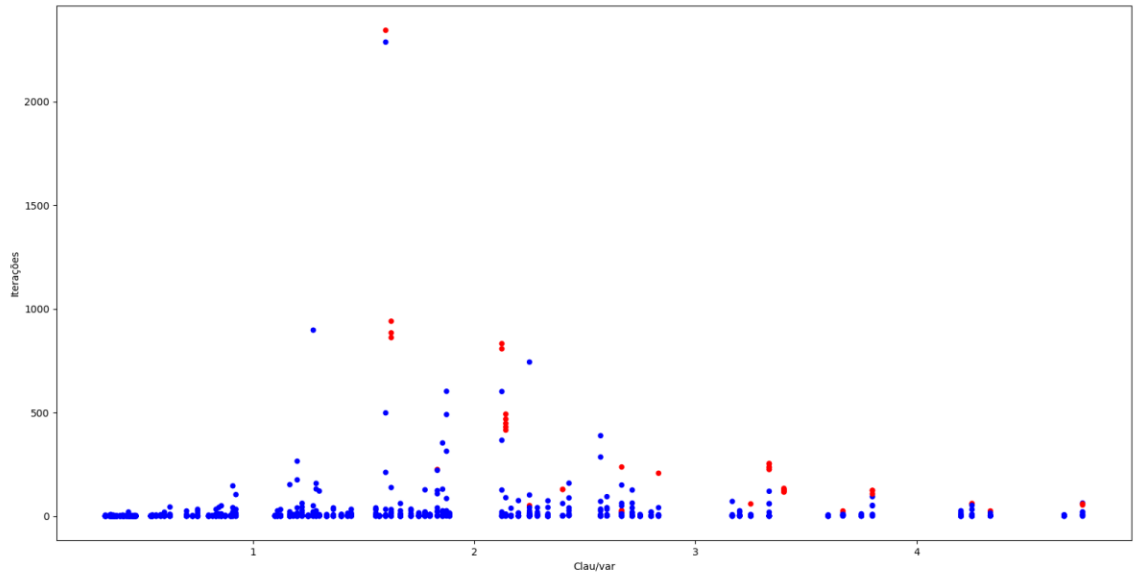

Figure 2. Gráfico mostrando a relação de Cláusulas/Variáveis com o número de iterações necessárias para chegar ao resultado com 55 iterações.

a tabela 1, é notável a existência de uma maior dificuldade em algumas regiões do que outras e, das regiões com maior dificuldade, o ponto de crossover não está incluído. Isso pode ser devido ao tamanho das instâncias ser muito pequeno, uma vez que as instâncias nessa região têm, em média, apenas 4 variáveis.

Com instâncias assim, mesmo com instâncias pertencendo ao crossover point, não é possível visualizar a complexidade esperada desses problemas. Já a motivação do porquê outras regiões apresentam dificuldades não é clara, a partir desses resultados, demonstrando que o objetivo de avaliar a dificuldade de executar instâncias SAT difíceis,

Regiões Cla/Var	Média de iterações
[0.0,0.5]	2.098663102
[0.5,1.0]	5.032323232
[1.0,1.5]	18.95284091
[1.5,2.0]	57.98787879
[2.0,2.5]	60.24949495
[2.5,3.0]	33.79480519
[3.0,3.5]	46.36
[3.5,4.0]	18.03636364
[4.0,4.5]	15.44848485
[4.5,5.0]	15.15454545

Table 1. Média de iterações por regiões

não foi completamente alcançado e poderia ser mais extensivamente avaliada em trabalhos futuros.

Verificando o objetivo da analise de complexidade, é perceptível que a inclusão de outros algoritmos não criou um aumento significativo na complexidade do algoritmo de Grover, que ainda tem como limitante $\mathcal{O}(\sqrt{N})$. Isso se deve pelo algoritmo de transformação de SAT em oráculos ser executado em tempo $\mathcal{O}(\sqrt{N})$ porém é apenas necessário executar uma vez por análise então apenas se soma na complexidade. Já o algoritmo de número desconhecido de respostas conseguir chegar numa resposta ainda limitado por $\mathcal{O}(\sqrt{N})$ e o algoritmo de avaliação das instâncias ser também executado em tempo linear. Assim, demonstrou-se que o algoritmo de Grover realmente tem uma utilidade para resolver instâncias SAT.

Como as instâncias geradas são muito pequenas e todas as execuções utilizaram de um simulador quântico, o tempo para resolver um problema nesse experimento é muito maior que o tempo de resolver em um computador clássico. Isso evidencia como ainda não é possível tirar algum tipo de vantagem sobre a execução do algoritmo de Grover em relação a algoritmos clássicos de resolução de SAT, assim alcançando o objetivo de analisar a utilidade de usar o algoritom de Grover para resolver instâncias SAT. Uma metrificação possível da dificuldade real do tempo que levaria para resolver essas instâncias em um computador quântico é a quantidade de iterações que foram necessárias, porém esse dado é facilmente influenciado por várias questões dessa implementação. A principal é o tamanho das instâncias, que, por serem pequenas, normalmente apresentam várias respostas possíveis e são na maioria satisfazíveis. Esses fatos podem ter diminuído a média de iterações para a avaliação de cada instância, portanto não é possível tirar conclusões enfáticas sobre como realmente seriam as execuções de problemas maiores em um computador real. Além de que, em um computador real, poderiam existir outros tipos de elementos que poderiam dificultar uma execução, como complexidade de implementar portas *multi-control*, por exemplo, tudo isso tornando mais complexo de se fazer uma análise realista de comparação do tempo de execução de instâncias SAT em um algoritmo clássico com um algoritmo quântico.

Como trabalhos futuros, também poderiam ser analisadas outras maneiras de representar uma instância SAT em um oráculo, pois a quantidade de qubits necessários au-

menta junto com o número de variáveis e cláusulas, de acordo com essa implementação. Caso fossem encontradas outras maneiras de se utilizar menos qubits, isso tornaria mais fácil de se executarem instâncias maiores e seria possível chegar a novos resultados.

Outra possibilidade de trabalho futuro seria a avaliação mais detalhada dos casos onde o algoritmo não encontrou resposta, mesmo elas existindo. Essa análise poderia ser tanto sobre a corretude da implementação, quanto uma análise sobre a amplitude das possíveis respostas para aquela instância. A análise avaliaria se as respostas não foram escolhidas por causa da aleatoriedade de resposta do algoritmo de Grover, ou se não foram selecionadas devido a uma má escolha de valores no algoritmo de número desconhecido de respostas. A partir disso, será possível avaliar a utilidade dessa combinação de algoritmos e concluir se o algoritmo de número desconhecido de respostas é adequado para resolver o problema SAT.

References

Boyer, M., Brassard, G., Høyer, P., and Tapp, A. (1998). Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505.

Crawford, J. M. and Auton, L. D. (1996). Experimental results on the crossover point in random 3-sat. *Artificial Intelligence*, 81(1):31–57. Frontiers in Problem Solving: Phase Transitions and Complexity.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search.

Mack, C. A. (2011). Fifty years of moore's law. *IEEE Transactions on Semiconductor Manufacturing*, 24(2):202–207.

Meng, Y. (2017). Dpll-sat-solver. `https://github.com/YibaiMeng/DPLL-SAT-solver`.

Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79.