Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Programa de Pós-Graduação em Informática Aplicada

Utilização do Algoritmo de Grover para o Treinamento de Redes Neurais Artificiais Clássicas

Ronaldo Ramos da Silva

Recife

Fevereiro de 2014

Ronaldo Ramos da Silva

Utilização do Algoritmo de Grover para o Treinamento de Redes Neurais Artificiais Clássicas

Orientador: Tiago Alessandro Espíndola Ferreira

Co-Orientador: Wilson Rosa de Oliveira Junior

Dissertação de mestrado apresentada ao Curso de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do grau de Mestre em Informática Aplicada.

Recife

Fevereiro de 2013

Resumo

As Redes Neurais Artificiais (RNA's), são modelos com habilidade de adaptar, aprender, generalizar, agrupar ou organizar dados onde as operações são baseadas em processamento paralelo. As RNA's são aplicadas em diversos campos como modelagem, análise de séries temporais, reconhecimento de padrões, etc. O treinamento das RNA's podem se dar de forma supervisionada, não supervisionada ou por reforço. O teorema de Convergências do Perceptron garante encontrar uma solução opara o treinamento da rede quando as classes são linearmente separáveis, mas não dá um limite superior para a convergência.

Este trabalho propõe uma abordagem quântica alternativa a abordagem clássica no treinamento do Perceptron. Na abordagem proposta, o adaptação dos pesos sinápticos se dá através do Algoritmo de busca de *Grover* promovendo, ainda que probabilística, um limite superior a convergência do *perceptron*.

Como aplicação, é feita a simulação do treinamento do *perceptron* utilizando o algoritmo quântico *BBHT* na tentativa encontrar um conjunto de pesos sinápticos para a solução do problema.

Abstract

Artificial Neural networks (RNA's) are models with the ability to adapt, learn, generalize, cluster or organize data where operations are based on parallel processing. The RNA's are applied in various fields such as modeling, time series analysis, pattern recognition, etc. The training of ANN's can take the form of supervised, unsupervised or reinforcement. The Perceptron Convergence Theorem guarantees finding a solution for network training when the classes are linearly separable, but not of an upper limit for convergence. This paper proposes an alternative approach to quantum classical approach in the Perceptron training. In the proposed approach, the adaptation of the synaptic weights occurs are through the search algorithm of Grover promoting, albeit probabilistic, an upper limit to the perceptron convergence. As an application, and made a simulation of the perceptron training using the quantum algorithm BBHT in trying to find a set of synaptic weights for the solution of the problem.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Uma rede Neural Artificial (RNA) pode ser descrita como modelo inspirado no processamento que ocorre em estruturas neurais biológicas ao realizar uma tarefa particular ou uma determinada função delegada[16]. RNAs são sistemas paralelos distribuídos compostos por unidades de processamento simples, ligadas por nós, que calculam funções matemáticas (normalmente não-lineares)[7]. A inspiração para as RNAs advem do exame das estruturas do cérebro, particularmente da análise dos neurônios e suas conexões.

As RNAs podem ser caracterizadas como modelos com habilidade de adaptar, aprender, generalizar, agrupar ou organizar dados onde as operações são baseadas em processamento paralelo. As raízes desta tecnologia têm diversas origens tais como: a neurociência, matemática, estatística, física, ciência da computação e engenharia.

Em virtude da habilidade de aprender de forma supervisionada ou não, são diversos os campos de aplicação, tais como: modelagem, analise de séries temporais, reconhecimento de padrões, etc. Apesar do grande número de problemas que podem ser abordados pela ferramenta RNA para gerar uma possível solução, há limitações comprometem a eficiência deste processo. Uma destas limitações é o tempo de treinamento da RNA, que utilizam principalmente algoritmos de gradiente descendente, e o algoritmo de aprendizagem backpropagation [35] e suas variações.

O *Teorema de Convergência do Perceptron* demostrado por *Rosenblatt* [35] dá a garantia de que o algoritmo de treinamento sempre encontra uma solução em um número finito de iterações caso as classes em questão sejam linearmente separáveis.

A Computação Quântica é um paradigma computacional baseado nos postulados da Mecânica Quântica e, portanto, fenômenos da Física Quântica serão considerados na computação [17], todavia, é possível ter uma visão puramente algébrica da mesma. A computação quântica utiliza elementos de três áreas bem conhecidas: matemática, física e ciência da computação. Sua compreensão passa pelo entendimento dos modelos matemáticos envolvidos como nos postulados da mecânica quântica.

De forma geral, existem três classes de algoritmos quânticos que oferecem vantagens sobre os algoritmos clássicos conhecidos[29]. Os algoritmos baseados na *Transformada Quântica de Fourier* [10], entre eles o algoritmo de Deutch[11] e o algoritmo de Shor[36], os algoritmos de busca ao qual pertence o algoritmo de Grover[14] e os algoritmos de simulação onde um computador quântico é usado para simular um processo computacional quântico. Devido às restrições impostas pela mecânica quântica, há um impedimento em estender os resultados obtidos pelos códigos clássicos para os códigos quânticos, devendo ser feita uma análise previa a fim de verificar se as restrições quânticas não estão sendo violadas.

Uma grande motivação para o estudo da computação quântica é a busca de algoritmos eficientes para problemas em que a solução em tempo polinomial não é classicamente conhecida. Visto que RNAs tem em sua formação diversas disciplinas somada a visão de Roger Penrose [18] o qual argumentou que uma nova física de unificação de fenômenos quânticos com a relatividade geral, pode explicar as habilidades mentais , tais como a compreensão, percepção e consciência[18], nascendo desta forma o interesse de pesquisas em Redes Neurais que incorporam conceitos de Física Quântica.

Uma Rede Neural Artificial Quântica (RNAQ), é um tipo de RNA que pode ser construída usando os princípios de processamento de informação quântica[1]. RNAQ mostra-se como uma grande área a ser explorada no ramo da computação e informação quântica. No entanto os modelos ainda são muito simplistas, hávendo pouco entendimento dos componentes essenciais de uma RNA baseada em técnicas e conceitos quânticos[15].

Lloyd [21], mostra que o aprendizado de maquinas quânticas pode fornecer ganhos exponenciais sobre algoritmos clássicos, visto que a aprendizagem de maquinas se dá sobre grandes quantidades de dados que se apresentam como vetores. Computadores quânticos são eficientes na em manipulação de vetores e de produtos de tensores em espaços de alta dimensionalidade.

Como um grande número de dados vem sendo gerado por dia o termo *Big Data* tornou-se corriqueiro. A necessidade de processamento destes dados tem levado grandes instituições a financiar estudos na área, pesquisadores vem produzindo algoritmos [33] a espera dos primeiros processadores quânticos com número de *qubits* suficiente para começar a testar na prática a inteligência artificial quântica.

Este trabalho propõe uma abordagem quântica alternativa às abordagens clássicas para o treinamento de redes neurais artificiais RNAs com pesos utilizando o *Algoritmo de Busca de Grover* [14] afim de encontrar uma rede com um erro mínimo com complexidade da ordem de $\Theta(\sqrt{N})$.

1.1 Relevância

O trabalho realizado colabora para as áreas de RNA e de Computação Quântica. O grande número de dados gerados no dias atuais (*Big Data*), produz também a necessidade maquinas de alto poder de processamento. A natureza paralela da computação quântica mostra-se um caminho teoricamente possível para o processamento desses dados. A utilização da Computação Quântica para solução de problemas do mundo clássico mostra-se bastante estimulador na busca de sua utilização desta abordagem em diversos problemas, uma vez que os efeitos quânticos permitem, em determinados casos, um ganho computacional expressivo. A larga utilização prática das RNAs justifica a importância das pesquisas na área em busca de métodos, técnicas e algoritmos que facilitem seu uso.

Além disso, o arcabouço proposto para implementar a busca quântica utilizando a linguagem de circuitos quânticos é bastante genérico [29]. Para implementar a abordagem proposta em função das diferentes redes neurais basta alterar no algoritmo quântico o operador que identifica os conjuntos de pesos da RNA, alteração esta que dependerá apenas do número de perceptrons e do número de conexões. A segunda alteração necessária é a determinação do número de *qubits*, definindo-se a representação e a precisão dos valores manuseados.

1.2 Objetivo do Trabalho

O objetivo deste trabalho é a proposição de treinamento de RNAs a partir de uma abordagem quântica de busca afim de encontrar um conjunto de pesos e bias que satisfaçam o erro mínimo desejado. Para tanto algums objetivos específicos foram contemplados:

1. Caracterização adequada do problema tratado e identificação de contribuições e trabalhos relacionados;
2. Investigação da possibilidade de adaptação de algoritmos quânticos existentes para busca do conjunto de dados desejado;
3. Definição de algoritmos quânticos que executem a tarefa de busca em tempo satisfatório;
4. Investigação da existência de ganhos na abordagem quântica proposta em relação à sua contrapartidas clássicas.

A solução proposta passa dela definição de um operador unitário que executa a tarefa de identificar os conjuntos de pesos e bias adequados a rede e por seguinte o uso do algoritmo BBHT[6] ou o algoritmo de contagem quântica para definição do número de aplicações do operador de Grover.

1.3 Organização do trabalho

Capítulo 2

Redes Neurais Artificiais

O primeiro trabalho sobre Rede Neurais foi realizado por *Warren MccCulloch* e *Walter Pitts* em 1943 [24]. Neste trabalho, foi desenvolvido um modelo matemático que simula o comportamento do neurônio biológico, (Figura 2.1), este trabalho concentrou-se muito mais em descrever um modelo artificial de um neurônio e apresentar suas capacidades computacionais do que em apresentar técnicas de aprendizado. As conclusões desta pesquisa foram de grande importância para a implementação computacional do neurônio.

2.1 Neurônio Biológico

O neurônio pode ser considerado a unidade básica da estrutura do cérebro e do sistema nervoso. A membrana exterior de um neurônio toma a forma de vários ramos extensos chamados dendritos, que recebem sinais elétricos de outros neurônios, e de uma estrutura a que se chama um axônio que envia sinais elétricos a outros neurônios. De forma geral, os eventos em um circuito integrado de silício acontecem na ordem de nanossegundos ($10^{-9}s$), enquanto que eventos neurais acontecem na ordem de milissegundos ($10^{-3}s$). Porem a aparente lentidão de um neurônio é compensada em umas rede neural pelo grande número de neurônios, (cerca de 10 bilhões), e conexões, (cerca de 60 trilhões) resultando numa extrema eficiência[16]. Recentemente pesquisas demostraram que os dendritos, as partes menores dos neurônios, também são componentes ativos que fazem suas próprias "computações"[37].

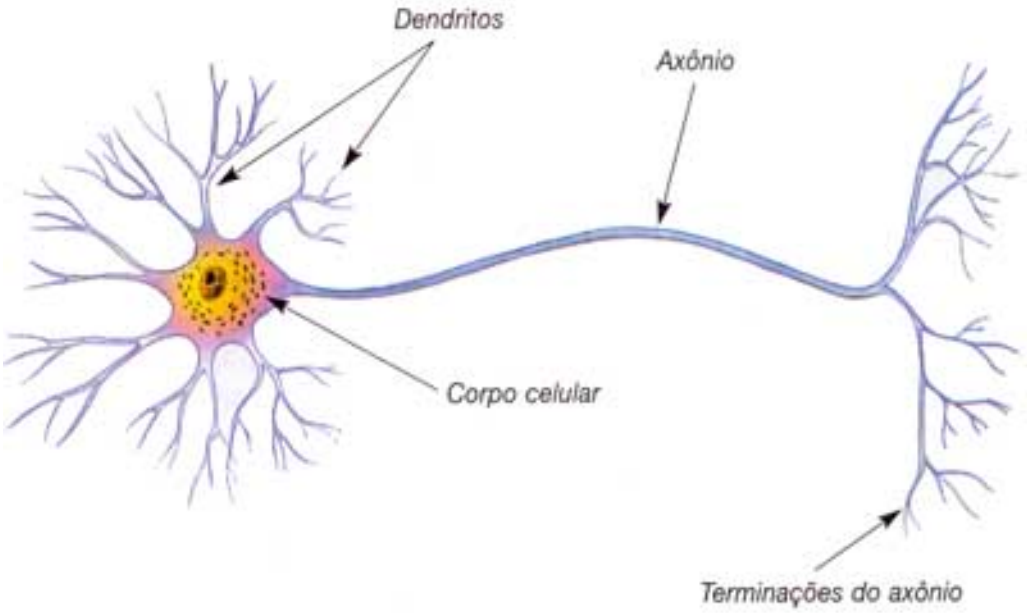

Figura 2.1: Neurônio

Fonte: `http://www.rc.unesp.br/biosferas/mat0002.php`

2.2 Neurônios artificiais

A proposta do modelo de *McCulloch* e *Pitts* (MCP), é uma simplificação do modelo biológico conhecido. Sua modelagem se apresenta como um nodo com n terminais de entrada (dentritos), $x_1, x_2, ..., x_n$, e apenas um terminal de saída y (axônio). Foram acoplados aos terminais de entrada pesos reais referentes ao processo sináptico real, gerando um sinal resultante inibitório ou excitatório causando o efeito $x_i w_i$ na sinapse i Figura 2.2.

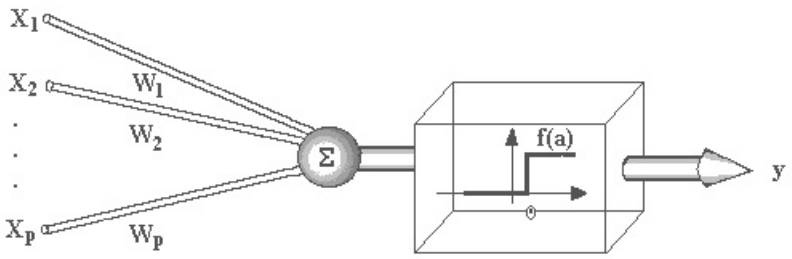

Figura 2.2: Modelo de neurônio artificial proposto por McCulloch e Pitts

Fonte: `http://www.din.uem.br/ia/neurais/`

Similar ao neurônio biológico que dispara quando a soma dos impulsos recebidos ultrapassa um limiar de excitação (*threshold*), o neurônio do modelo MCP é ativado através da aplicação de uma "função de ativação"que dependendo da soma ponderada das entradas

ativa ou não a saída. Este terá sua saída ativada quando:

$$\sum_{i=0}^{n} x_i w_i \geq \theta \tag{2.1}$$

Onde n é o numero de entradas do neurônio, w_i é o peso associado a entrada x_i e θ é o limiar (*threshold*) do neurônio.

O modelo original apresentado por *MccCulloch* e *Pitts* foi proposto com pesos fixos, não ajustáveis, o que não permite aprendizado, e disparos inibidores são melhor representados por pesos negativos, além de que redes com apenas uma camada só conseguirem implementar funções linearmente separáveis[7].

A partir do modelo MCP, foram propostos diversos modelos tais como a adição de um bias aplicado externamente representado por b_k que produz o efeito de uma transformação afim na função de ativação linear.

O neurônio pode ser descrito da seguinte forma matemática:

$$u_k = \sum_{i=i}^{n} x_i w_{ki} \tag{2.2}$$

e

$$y_k = \varphi(u_k + b_k) \tag{2.3}$$

A função de ativação φ proposta originalmente por *McCuullock* e *Pitts*, é uma função de limiar, isto é,

$$y_k = \varphi(u_k + b_k) = \begin{cases} 1 & \text{se } u_k + b_k \geq 0 \\ 0 & \text{se } u_k + b_k < 0 \end{cases} \tag{2.4}$$

Onde u_k é a saída do combinador linear, b_k é o bias (Figura 2.3), $\varphi(.)$ é a função de ativação e y_k é o sinal de saída do neurônio.

Outra modificação foi a utilização de diferentes funções de ativação das quais pode-se citar a função de limiar, a saída de um neurônio assume o valor 1, se $y_k \geq 0$ e 0 caso contrário; a função Sigmoide, esta sendo a função mais comum nas aplicações de redes neurais, e a função tangente hiperbólica que permite que a função se estenda de -1 até $+1$[16].

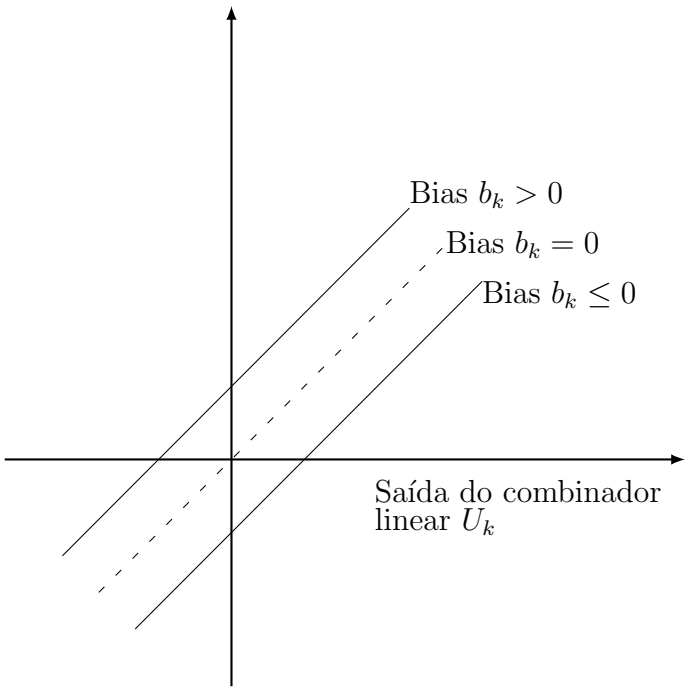

Figura 2.3: Transformação afim produzida pela presença de um bias

2.3 Aprendizado

Uma RNA tem a capacidade de aprender por exemplos e fazer interpolações ou extrapolações do que foi aprendido. O algoritmo de aprendizado baseado no conceito conexionista é um conjunto de procedimentos que se preocupa em determinar a intensidade das conexões entre os neurônios de maneira que a RNA possa aprender uma determinada função. Não há um único algoritmo de aprendizagem, nem são todos baseados no mesmo conceito, cada algoritmo tem suas vantagens e desvantagens, mas basicamente diferem apenas pela forma que ajustam os pesos. Os diversos algoritmos de aprendizado podem ser classificados em três classes principais: supervisionado ,não-supervisionado e por reforço. O aprendizado supervisionado é o mais comum no treinamento de RNAs, o instrutor fornece as entradas e saídas desejadas para a rede e confere o quanto a rede está próxima de uma solução aceitável, adaptando durante o treinamento os pesos entre os neurônios, de modo a prover uma menor diferença entre as saídas desejadas e as saídas geradas pela RNA. A desvantagem é que sem o instrutor, a rede não conseguirá aprender novas estratégias para problemas não contemplados pelos exemplos fornecidos a rede. No aprendizado não supervisionado, não há a figura do instrutor, inicialmente, as saídas da rede não são conhecidas, apenas os padrões de entradas são fornecidos, funcionando de modo a distinguir classes de padrões diferentes dos dados apresentados à rede, através de algoritmos de aprendizado baseados geralmente em conceitos de vizinhança e agrupamento. A rede é ajustada de acordo com regularidades estatísticas dos dados de entrada, criando uma harmonia interna de tal forma que ela cria categorias, otimizando em relação aos parâmetros livres da rede uma medida da quantidade que é independente da tarefa a ser executada[23]. Os estágios iniciais da audição e visão dos seres humanos ocorrem através do aprendizado não supervisionado. O aprendizado por reforço pode ser visto como um caso particular de aprendizado supervisionado, onde a principal diferença é a medida de desempenho usada em cada um dos sistemas [7].

2.4 Redes Perceptron e Adaline

2.4.1 Portas de limiar lineares

As portas de limiar lineares são definidas conforme a equação 2.4, são semelhantes ao nodo MCP.

Modelada desta forma, as portas de limiar lineares estão restritas à solução de problemas cuja solução pode ser obtida pela separação de duas regiões por meio de um hiperplano, isto é, são linearmente separáveis. Pode-se visualizar o caso bidimensional. Seja x_1 e x_2 as entradas do nodo e w_1 e w_2 os pesos referentes as entradas e θ e y o limiar e a saída respectivamente, executando uma função qualquer. A condição de disparo do nodo (y = 1) é então definida por $x_1w_1 + x_2w_2 \geq 0$. Então pode-se descrever de forma geral a equação da reta onde $x_2 = f(x_1)$, conforme mostra a Equação 2.5. Desta forma a superfície de decisão de uma porta de limiar linear está restrita a uma reta, ou um hiperplano para o caso n-dimensional[7]. A Figura 2.4 mostra a solução para o problema do *OU* lógico através de uma porta de limiar linear.

$$x_2 = -\left(\frac{w_1}{w_2}\right) x_1 + \left(\frac{\theta}{w_2}\right) \tag{2.5}$$

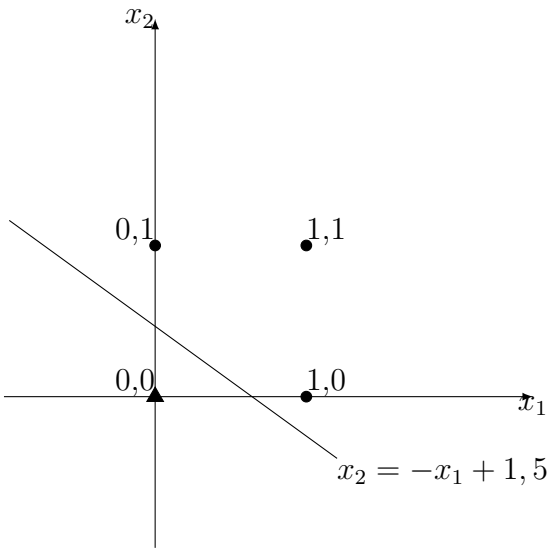

Figura 2.4: Solução para o problema do E através de unia porta de limiar linear. Para este caso, tem-se $w_1 = w_2 = 1$ e $\theta = 1,5$.

2.4.2 Perceptron

O perceptron é a forma mais simples de uma rede neural usada para a classificação de padrões ditos linearmente separáveis. O algoritmo usado para ajustar os parâmetros livres desta rede neural foi inicialmente proposto em um procedimento de aprendizagem desenvolvido por *Rosenblatt* [34] para seu modelo cerebral do perceptron. Este algoritmo

apresenta algumas diferenças em relação a regra de Hebb (1949, p.62 apud [16]). A função de ativação passou a poder ter um limiar θ diferente de zero. Foi introduzido um fator de aprendizado $\eta(0 < \eta \leq 1)$ que regula a velocidade de modificação dos pesos. *Rosenblatt* demonstrou que se padrões usados para treinar o perceptron são retirados de duas classes linearmente separáveis, então o algoritmo converge e posiciona a superfície de decisão na forma de um hiperplano entre duas classes. Esta demonstração é conhecido como *Teorema de Convergência do Perceptron.*

De uma maneira geral, durante o processo de adaptação, ou aprendizado, o que se deseja obter é o valor da variação Δw a ser aplicado ao vetor de pesos w de tal forma que o seu valor atualizado $w(t+1) = w(t) + \Delta w$ e $\Delta w = \eta x$ onde x é a entrada, esteja mais próximo da solução desejada do que w(t). Portanto, os algoritmos de aprendizado em RNAs visam ao desenvolvimento de técnicas para a obtenção do valor de Δw mais apropriado para a obtenção da solução do problema em questão.

2.4.3 Algoritmo de aprendizagem do Perceptron

O objetivo do algoritmo é obter o incremento Δw a ser aplicado ao vetor de peso w atualizando o valor $w(t+1) = w(t) + \Delta w$ deixando $w(t+1)$ mais próximo da solução desejada do que $w(t)$. A condição de disparo é $w.x = 0$ onde $w = -\theta, w_1, w_2, ..., {w_n}^T ex = 1, x_1, x_2, ..., {x_n}^T$. Selecionar um exemplo de treinamento (x, d), padrão de entrada e sua respectiva saída. Isto pode ser feito de maneira cíclica (em ordem) através dos exemplos de treinamento ou pegando um exemplo aleatoriamente.

Se w classifica d_k corretamente, isto é, se: $w.x_k \geq 0$ e $d_k = +1$ ou se $w.x_k < 0$ e $d_k = 0$ Então: não fazer nada.

Senão Passo de alteração: Modificar w somando ou subtraindo x_k de acordo com a saída correta ser +1 ou 0: Neste caso o erro $(e = d - w.x_k)$ e pode ser +1 ou -1 e $w' = w + e_k x_k$.

Uma modificação plausível seria somar ao vetor w um vetor que estivesse na direção de x com o vetor ηx. Assim a equação de atualização de w pode ser escrita como $w(t+1) = w(t) + \eta ex$.

Capítulo 3

Informação Clássica

O computador clássico é descrito como uma máquina que lê um conjunto de dados, executa cálculos sobre estes e gera uma saída. Estes dados comumente são codificados com zeros e uns. A peça mais básica de informação é chamado de *bit*, este representa a menor quantidade de informação digital capaz de ser armazenada podendo estar em um de dois estados mutuamente exclusivos, 0 ou 1. É fácil ver que é possível armazenar um bit em um circuito elétrico através da distinção entre dois valores como tensão ou corrente elétrica onde 0 é um estado de Baixo potencial elétrico e 1 é um estado de alto potencial elétrico. Matematicamente, um bit está associado a uma variável $x \in \{0,1\}$, onde o conjunto $\{0,1\}$ é dito espaço de estados do bit. Assim, um bit x pode ser 0 ou 1, mas nunca uma sobreposição destes valores. Para representar uma quantidade maior de estados, um registro, é preciso de uma área mais extensa da memória, é feita então uma justaposição de uma quantidade finita de k bits. Isto é, um registro está associado a uma *k-tupla* $(x_{k-1}, ..., x_0) \in \{0,1\}^k$, onde o produto cartesiano

$$\{0,1\}^k = \underbrace{\{0,1\} \cdot \{0,1\} \cdot ... \cdot \{0,1\}}_{k} \tag{3.1}$$

É chamado de espaço de estados desse registro. Um registro pode estar a cada instante em apenas um dos 2^k diferentes estados do seu espado de estados. Existe uma bijeção natural θ entre o conjunto das *k-uplas* $(x_{k-1}, ..., x_0); x_i \in \{0,1\}$ e o conjunto $\{N \in Z; 0 \leq N \leq 2^{k-1}\}$

definida por:

$$\theta(x_{k-1}, ..., x_0) = x_{k-1}.2^{k-1} + \ldots + x_1.2^1 + x_0.2^0 \tag{3.2}$$

Que vem a ser a representação binária de N. Vamos supor então que se deseja representar os números 0, 1, 2, 3 em binário. Tem-se quatro itens ($2^2 = 4$). Portanto, é preciso de pelo menos dois bits para representar esses números.

Tabela 3.1: Representação Binária dos números 0-3

Decimal	Binário
0	00
1	01
2	10
3	11

É fácil ver que se é desejável avaliar a quantidade mínima de k bits de um número inteiro positivo N, será necessário um registro com $\lceil \log_2 N \rceil \leq k$ bits, de forma que k deverá satisfazer $N \leq 2^k - 1 < 2^k$, donde $\log_2 N < k$.

3.1 Portas Lógicas

A *Álgebra Booleana*, conhecida também como álgebra de chaveamento[9], é o modelo matemático mais adequado para descrever o funcionamento dos circuitos lógicos digitais. Uma função booleana , é uma função da forma $f : \{0,1\}^k \to \{0,1\}^m$, cuja entrada é um estado $x = (x_{k-1}, \ldots, x_0) \in \{0,1\}^k$ e cuja saída é um estado $y = (f_{m-1}(x), \ldots, f_0(x)) \in \{0,1\}^m$. De forma geral, f é descrita por blocos elementares chamadas de portas lógicas, *AND*, *OR* e *NOT*, conhecidas como portas universais ou elementares por serem estas suficientes para construção de qualquer outra porta lógica através de combinações adequadas. Estes componentes podem ser implementados fisicamente por transistores e outros componentes eletrônicos[32].

3.1.1 Porta Not

A porta *Not* ou inversora tem como símbolo "¬" e implementa a função booleana $\neg : \{0,1\} \to \{0,1\}$ definida por $\neg(x) = 1 - x$. Ou seja, a função *Not* possui apenas uma entrada e uma saída atuando sobre um único bit retornando 0 se a entrada for 1 e 1 se a entrada for 0. Esta porta tem a seguinte representação:

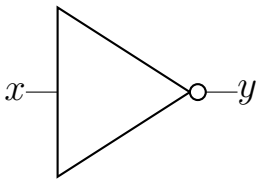

Figura 3.1: Porta: *Not*

x	y
0	1
1	0

Tabela 3.2: Tabela Verdade: Porta Not

3.1.2 Porta And

A porta *And* também chamada de disjunção tem como símbolo " $\wedge$ " e implementa a função booleana $\wedge : \{0,1\} \to \{0,1\}, k \geq 2$, definida por:

$$\wedge(x_{k-1}, \ldots, x_0) = \begin{cases} 1 & \text{se } (x_{k-1}, \ldots, x_0) = (1, 1, \ldots, 1) \\ 0 & \text{se } (x_{k-1}, \ldots, x_0) \neq (1, 1, \ldots, 1) \end{cases}$$

A função *And* tem no mínimo duas entradas, mas uma única saída. Esta porta tem a seguinte representação:

3.1.3 Porta Or

A porta *Or* também chamada de conjunção tem como símbolo " $\vee$ " e implementa a função booleana $\vee : \{0,1\} \to \{0,1\}, k \geq 2$, definida por:

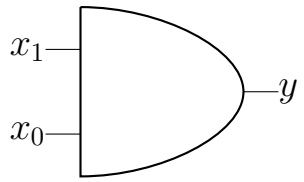

Figura 3.2: Porta And

x_1	x_0	y
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 3.3: Tabela Verdade: Porta And

$$\wedge(x_{k-1},\ldots,x_0) = \begin{cases} 0 & \text{se } (x_{k-1},\ldots,x_0) = (0,0,\ldots,0) \\ 1 & \text{se } (x_{k-1},\ldots,x_0) \neq (0,0,\ldots,0) \end{cases}$$

Semelhante a porta *And*, a função *Or* tem no mínimo duas entradas, mas uma única saída. Esta porta tem a seguinte representação:

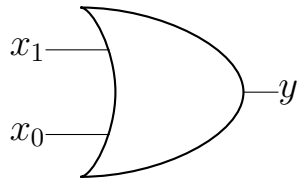

Figura 3.3: Porta Or

3.1.4 Reversibilidade

O *Princípio de Landauer* [4] afirma que que qualquer manipulação logicamente irreversível de informações, como o apagamento de um *bit* ou a fusão dos dois *bits*, deve ser acompanhada de um aumento de entropia. Por outro lado, é geralmente aceito que qualquer transformação logicamente reversível de informação pode, em princípio, operar de uma forma termodinamicamente reversível por um mecanismo físico apropriado[5]. Uma porta lógica reversível está associada a uma função booleana inversível. Dentre as portas lógicas

x_1	x_0	y
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 3.4: Tabela Verdade: Porta Or

elementares, apenas a porta *Not* é reversível, de forma que a inversa da função $\neg(x) = y$ é a função $\neg^{-1}(y) = x$, definida por $\neg^{-1}(y) = 1 - y$. A função *Not* é dita auto inversa, isto é, $\neg^{-1} = \neg$. De forma mais clara, uma porta lógica será reversível se conhecido o valor de saída (*output*) é possível determinar o valor de entrada (*input*). De posse deste conceito, é possível ver que conhecida a saída das portas *And* ou *Or*, os valores da entrada não podem ser determinados.

3.2 Estado, Evolução e Medição

3.2.1 Estado e Evolução

O estado de um registro de *k-bits* pode ser definido como uma *k-tupla* $(x_{k-1}, \ldots, x_0 \in \{0,1\}^k$, dito o seu espaço de estado. è possível visualizar o espaço de estados como os 2^k vértices de um cubo de aresta unitária e dimensão k. Como exemplo temos o cubo que representa um registro composto por 3 *bits*, ver Figura 3.4.

Assim, para o caso de $k = 3$, existem oito combinações para a tripla $(x_2, x_1, x_0), x_i \in \{(0,1)\}$ formada pelos oito vértices. O conjunto formado pela união disjunta desses oito vértices é o que denominamos de *espaço de estados*.

A evolução de um registro da-se quando o estado x passa ao estado $y = f(x)$, onde $f : \{0,1\}^k$ é uma função booleana. Esta evolução acontece devido a ação de uma porta lógica ou função booleana. Assim considerando o espaço de estados anterior com 3 bits e um registro no estado inicial $(0,0,0)$, a aplicação sucessiva de duas portas lógicas A e B formadas pelos blocos elementares fará o registro inicial evoluir para os estado $(1,0,0)$ e por fim para o estado $(1,1,0)$, Figura 3.5.

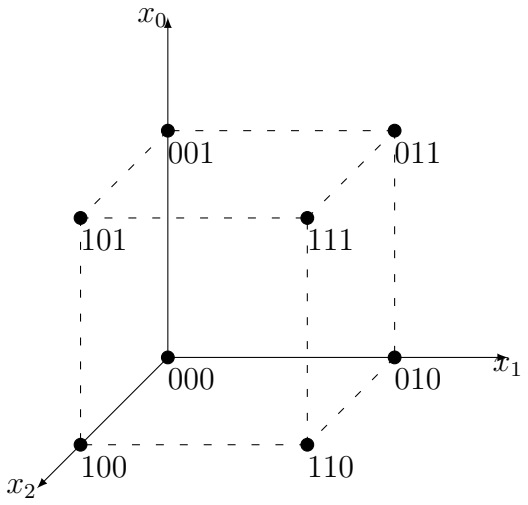

Figura 3.4: Representação do espaço de estados: 3 *bits*

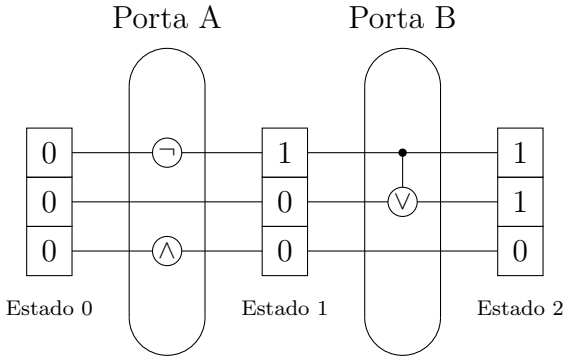

Figura 3.5: Evolução do espaço de estados de 3 *bits*

3.2.2 Medição

Na computação clássica, medir um registro consiste em observar o estado no qual este se encontra através de processos computacionais adequado ao sistema. A observação do registro consiste em observar o estado individual de cada *bit* pertencente ao registro, isto é, após a aplicação das portas lógicas da Figura 3.5, a leitura do primeiro *bit* menos significativo será 0 enquanto que o segundo e terceiro *bit* será 1 resultando no número 6 em notação binária. É Possível observar a a medida do registro é um processo determinístico e que a sua observação não alterou o seu estado.

Capítulo 4

Computação Quântica

A *máquina de Turing* (MT) [39] foi a primeira noção formal de computação, onde se definiu um modelo de como os algoritmos da *Computação Clássica* devem ser construídos ajudando a entender quais os limites desta forma de computação. Segundo *Turing* se um algoritmo pode ser realizado em qualquer tipo de sistema físico então há um algoritmo equivalente computável na MT. Esta afirmação é a denominada tese de *Church-Turing* [39].

Apesar de ser capaz de descrever algoritmos para diversos problemas, a MT apresenta algumas limitações, dais quais, o custo computacional é um deles, evidenciado por alguns problemas de ordem prática como o problema da busca de um elemento em um vetor desordenado.

Existem muitas variações do modelo básico da MT. Uma das variantes é a introdução da aleatoriedade denominada *Maquina de Turing Probabilista* [38], contudo, mesmo essa variante não muda a capacidade essencial das MTs como modelo de computação. A MT determinística pode ser muito menos eficiente do que o modelo aleatório, mas a classe de funções computáveis não muda introduzindo-se a aleatoriedade no modelo básico [29].

Benioff [3] em 1980 criou um modelo de MT com uma fita quântica para aproveitar vantagens de determinados fenômenos quânticos como a superposição de estados [3], mas está, por utilizar estados clássicos, ao ler a fita quântica, fazia com que os estados da fita também se tornassem clássicos. Na tentativa de simular sistemas quânticos *Richard Feynman* provou em 1982 que nenhuma MT conhecida poderia simular de forma eficiente este tipo de sistema, pois poderia haver um aumento exponencial no número de estados inviabilizando

a computação em tempo hábil [12]. A Computação Quântica proposta por *David Deutsth* em 1985 [11] é baseada inteiramente na *Física Quântica*, neste modelo *Deutsth* propôs que portas quânticas poderiam funcionar de maneira similar a computação digital tradicional baseada em portas logicas binárias.

A Computação Quântica é definida então como um paradigma computacional baseado nos *postulados da Mecânica Quântica* e, portanto, é possível ter uma visão puramente algébrica [17]. Fenômenos da física quântica serão considerados na Computação.

A Computação Quântica muitas vezes afugenta potenciais pesquisadores em ciência da computação, devido à aparente necessidade de ir fundo em *Mecânica Quântica*. A complexidade da Computação Quântica ainda é mal compreendida e neste capitulo Introduziremos alguns conceitos e definições da *Mecânica Quântica* adequados ao entendimento dos conteúdos tratados neste documento.

4.1 Postulados da Mecânica Quântica

A *Mecânica Quântica* é uma estrutura matemática para o desenvolvimento de teorias físicas [29]. Por ser regido por postulados (axiomas) que fornecem uma conexão entre o mundo físico e o formalismo matemático da mecânica Quântica, analisando de uma forma matemática, podemos classificar a *Mecânica Quântica* como uma teoria. Os postulados necessários ao entendimento da *Computação Quântica* são descritos e comentados a seguir.

4.1.1 Primeiro Postulado

O *espaço de Hilbert* [20] é usado para modelagens de sistemas infinitos. A evolução dos estados em um sistema quântico são modelados matematicamente no *espaço de Hilbert* das funções de ondas. Porém, para o entendimento da *Computação Quântica*, é necessário apenas o uso de sistemas quânticos finitos, em outras palavras, não é necessário modelar sistemas quânticos no *espaço de Hilbert* das funções de ondas, considera-se somente os *espaços de Hilbert* que são espaços vetoriais complexos de dimensão finita, assim tem-se a vantagem de não se preocupar com funções de onda. Um espaço vetorial complexo é um espaço vetorial cujos vetores possuem coordenadas (e consequentemente magnitudes), descritos por números

complexos.[40].

Postulado 4.1.1. *[29] Associado a qualquer sistema físico isolado existe um espaço vetorial complexo com produto interno (espaço de Hilbert) conhecido como espaço de estados do sistema. O sistema é completamente descrito por seu vetor de estado, que é um vetor unitário no espaço de estados do sistema.*

A unidade básica de informação na Computação Clássica é o *bit*, que assume valor 0 ou 1. Na *Computação Quântica*, a unidade básica de representação da informação é um sistema quântico com espaços de estados com duas dimensões, o *qubit* (*quantum bit*), este é o sistema quântico mais simples. Usando a notação de *Dirac*, um *qubit* $|\psi\rangle$ é representado por meio de um vetor bidimensional em um espaço de *Hilbert*, de acordo com a seguinte expressão geral:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4.1}$$

e o estado geral de um qubit $|\psi\rangle$ tem a seguinte notação:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{4.2}$$

Sendo $|0\rangle$ e $|1\rangle$ a base ortonormal para o espaço de estados do sistema conhecido como *base computacional* e α e β são números complexos arbitrários ($\alpha, \beta \in \mathbb{C}$), a relação de normatização deve ser satisfeita:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{4.3}$$

A interpretação física do *qubit*, na expressão 4.2, é que ele está simultaneamente nos estados $|0\rangle$ e $|1\rangle$. Como os vetores $|0\rangle$ e $|1\rangle$ são ortonormais, a soma não dá o vetor nulo. As possibilidades excludentes classicamente coexistem quanticamente. A quantidade possível de informação que se pode armazenar no estado $|\psi\rangle$ é infinita. Porém, essa afirmação está apenas a nível quântico. A coexistência é destruída quando se tenta observa-la, onde para torná-la acessível a nível clássico é preciso fazer uma medida.

4.1.2 Interpretação Geométrica

Um qubit pode ser interpretado geometricamente em três dimensões, como um vetor contido numa esfera ($\mathbb{R}^3$). Essa representação é denominada de *Esfera de Bloch*. Pode-se expressar genericamente o estado de um *qubit* por:

$$|\psi\rangle = e^{i\gamma}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle\right) + e^{i\varphi}\left(\sin\left(\frac{\theta}{2}\right)|1\rangle\right) \tag{4.4}$$

onde γ, θ, e $\varphi \in \mathbb{R}$. O termo $e^{i\gamma}$ é denominado fator de fase global e não possui efeito físico observável (uma vês que $|e^{i\gamma}| = 1$) sendo, mais usada a expressão:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}(\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{4.5}$$

sendo $\alpha = \cos\frac{\theta}{2}$ e $\beta = e^{i\varphi}\sin\frac{\theta}{2}$, estando assim o *qubit* já normalizado.

Os números θ e φ definem um ponto sobre a superfície da *esfera de Bloch* como mostra a Figura 4.1. A representação na *esfera de Bloch* mostra-se bastante importante no auxilio da interpretação dos resultados das operações.

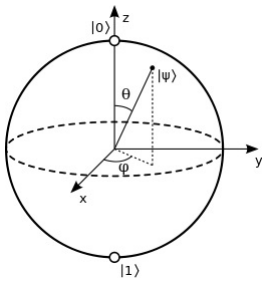

Figura 4.1: Esfera de Bloch

Fonte: `http://es.wikipedia.org/wiki/Esfera_de_Bloch`

4.1.3 Evolução Unitária

A mecânica quântica descreve matematicamente a evolução no tempo de um sistema quântico isolado por uma transformação linear [29]. A descrição de sistemas quânticos isolados na álgebra linear são vetores unitários, e, as funções que transformam vetores unitários

em vetores unitários do mesmo espaço vetorial são as transformações unitárias. Transformações lineares unitárias U podem ser definidas como aquelas que atendam à seguinte propriedade:

$$U^{\dagger}U = UU^{\dagger} = I \tag{4.6}$$

Onde $U^{\dagger}$ é a conjugada transposta de U e I é a matriz identidade.

Postulado 4.1.2. *[29] A evolução de um sistema quântico fechado é descrita por uma transformação unitária. Ou seja, o estado $|\psi\rangle$ do sistema no tempo t_1 está relacionado ao estado $|\psi'\rangle$ do sistema no tempo t_2 por um operador unitário U o qual depende somente dos tempos t_1 e t_2,*

$$|\psi'\rangle = U|\psi\rangle \tag{4.7}$$

Logo, a consequência imediata da definição da evolução de um sistema quântico como um processo unitário é que ele é reversível e a energia do processo se conserva. Analisando a *esfera de Bloch* [26], a transformação unitária corresponde a uma rotação no espaço vetorial que preserva o comprimento do vetor, isto garante que a probabilidade total de um conjunto de estados sempre permaneça a mesma, isto é, considerando o *qubit* normalizado sua magnitude será igual a 1.

A equação de *Schrodinger* descreve a relação temporal entre os estados $|\psi'\rangle$ e $|\psi\rangle$.

$$\frac{d}{dt}|\psi(t)\rangle = -i\widehat{H}|\psi(t)\rangle \tag{4.8}$$

Onde $\widehat{H}$ é um operador auto-adjunto chamado de Hamiltoniano do sistema fechado. A equação 4.7 representa o equivalente discreto da equação de *Schrodinger*, obtida considerando-se quantidades infinitesimais *dt*.

$$|\psi(t)\rangle = e^{-it\widehat{H}}|\psi(0)\rangle = U(t)|\psi(0)\rangle \tag{4.9}$$

4.1.4 Medição

As transformações unitárias descrevem a evolução de sistemas quânticos isolados, isto se dá para sistemas que não interagem com outros sistemas, mas há momentos que é necessário obter acesso a informação contida no estado $|\psi\rangle$, em outras palavras, um sistema físico externo deverá interagir com o sistema para obter um resultado. Na *Computação Clássica* a extração do resultado do estado dos bits é conceitualmente simples, diferentemente da Computação Quântica onde este processo denominado de medição não é uma tarefa trivial. A medição afeta o sistema isolado provocando um colapso no espaço de estado deste sistema após a medição. O processo de medição altera o estado do sistema $|\psi\rangle$ em 4.2 fazendo-o assumir o estado $|0\rangle$ com probabilidade $|\alpha|^2$, ou o estado $|1\rangle$ com probabilidade $|\beta|^2$. Matematicamente falando, uma medição corresponde a uma projeção do vetor de estados representando o *qubit* em um dos estado da base computacional. A medição é uma operação irreversível[1], uma vez que não há meios de faze-lo voltar ao estado anterior à medição [25].

Postulado 4.1.3. *[29] As medidas quânticas são descritas por uma coleção $\{M_m\}$ de operadores de medida. Esses operadores atuam no espaço de estados do sistema que está sendo medido. O índice m se refere aos efeitos das medidas que podem ocorrer em um experimento. Se o estado do sistema quântico for $|\psi\rangle$ imediatamente antes da medida, então a probabilidade de um resultado m ocorrer é dada por:*

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle, \tag{4.10}$$

e o estado do sistema depois da medida será

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}} \tag{4.11}$$

Os operadores de medida satisfazem a equação de completude,

$$\sum_{i=m} M_m^\dagger M_m = I. \tag{4.12}$$

[1]Contudo, observa-se que ainda há conservação de energia

4.1.5 Sistemas Compostos

Na computação clássica, um bit é capaz de representar dois valores, 0 e 1, e um conjunto de *n bits*, pode representar 2^n valores diferentes, um por vez. Na Computação Quântica, um registrador de *n qubits* também pode armazenar 2^n valores, porém, graças à superposição dos estados, todos são representados simultaneamente. O postulado a seguir descreve este sistema:

Postulado 4.1.4. *[29] O espaço de estados de um sistema físico composto é o produto tensorial dos espaços de estados dos sistemas físicos que o compõem. Além disso, se houver sistemas numerados de 1 até n, e o sistema número i for preparado no estado $|\psi_i\rangle$, então o estado do sistema composto será $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$.*

O produto tensorial de dois *qubits* $|a\rangle$ e $|b\rangle$, representado por $|a\rangle \otimes |b\rangle = |ab\rangle = |a\rangle|b\rangle$ é descrito como:

$$|a\rangle \otimes |b\rangle = \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix} \tag{4.13}$$

$$= \begin{bmatrix} a_1 \cdot |b\rangle \\ a_2 \cdot |b\rangle \\ \cdots \\ a_n \cdot |b\rangle \end{bmatrix} \tag{4.14}$$

Como não houve interação na criação dos estados $|a\rangle$ e $|b\rangle$, estes estão em *espaços de Hilbert* distintos $\mathcal{H}_1$ e $\mathcal{H}_2$, respectivamente. De forma que uma alteração em qualquer dos estados não alterará o outro.

O sistema composto constituído pelos estados $|a\rangle$ e $|b\rangle$ pode ser representado como:

$$|ab\rangle = |a\rangle \otimes |b\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2 \tag{4.15}$$

$$
\begin{aligned}
|\psi\rangle &= |\psi_1 \otimes |\psi_2\rangle \\
&= (\alpha_1|0\rangle + \beta_1|1\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle) \\
&= \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle \\
&= \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \\
&= \alpha'_0|0\rangle + \alpha'_1|1\rangle + \alpha'_2|2\rangle + \alpha'_3|3\rangle \\
&= \sum_{i=0}^{2^2-1} \alpha'_i|i\rangle
\end{aligned} \tag{4.16}
$$

Onde α'_i é a amplitude associada.

De forma genérica, o estado geral de um sistema de n *qubits* pode ser denotado por:

$$
|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i|i\rangle \tag{4.17}
$$

com $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$ e a base computacional $\{|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle\}$.

4.2 Processamento da Informação

O processamento da informação na computação clássica se da através de operações sobre os bits que armazenam a informação, estes operadores são os circuitos lógicos, que são agrupamentos de dispositivos mais simples denominados de portas lógicas. Analogamente, na Computação Quântica o processamento da informação também é realizado por operadores, estes são os circuitos quânticos que são formados por portas quânticas que realizam operações unitárias sobre os *qubits*. Como descreve o postulado da evolução, uma porta quântica simples aplica uma operação unitária U sobre um *qubit* no estado $|\psi\rangle$ fazendo-o evoluir para $U|\psi\rangle$. Por serem matrizes unitárias, os operadores preservam a norma dos vetores e como toda matriz unitária, por definição, possui inversa, as operações definidas sobre os *qubits* são inversíveis. A representação destas portas é dada por matrizes unitárias, que devem possuir

a seguinte propriedade:

$$U \cdot U^{\dagger} = U^{\dagger} \cdot U = \mathbb{I} \tag{4.18}$$

4.2.1 Portas Quânticas de um Qubit

Portas quânticas que atua sobre um *qubit* podem ser descritas por matrizes $\mathbb{C}^2 \to \mathbb{C}^2$, Como existem infinitas matrizes unitárias $\mathbb{C}^2 \to \mathbb{C}^2$, também há infinitas portas quânticas que atuam sobre um *qubit* [29]. Dentre os operadores unitários destacam-se as *matrizes de Pauli*:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{4.19}$$

A porta quântica mais simples é a porta $I : \mathbb{C}^2 \to \mathbb{C}^2$ (identidade) definidas nos vetores da base ortonormal, que atuando sobre um *qubit* não altera o seu estado.

$$I|0\rangle = |0\rangle \qquad I|1\rangle = |1\rangle \tag{4.20}$$

A porta $U_{NOT} : \mathbb{C}^2 \to \mathbb{C}^2$ definidas nos vetores da base ortonormal, simbolizada por X, atua sobre um *qubit* trocando seu estado, ou seja:

$$X|0\rangle = |1\rangle \qquad X|1\rangle = |0\rangle \tag{4.21}$$

A porta $Z : \mathbb{C}^2 \to \mathbb{C}^2$ definidas nos vetores da base ortonormal, também chamada de inversão de fase, executa a troca a fase do estado $|1\rangle$ de 180 graus($e^{i\pi} = -1$) ao passo que sua atuação sobre $|0\rangle$ o deixa neutro.

$$Z|0\rangle = |0\rangle \qquad Z|1\rangle = -|1\rangle \tag{4.22}$$

Um operador que tem grande importância na Computação Quântica é o *operador de Hadamard*, denotado por $H : \mathbb{C}^2 \to \mathbb{C}^2$. Sua ação leva um estado da base a uma superposição de estados, e uma segunda aplicação leva a superposição ao estado original. Sua representação matricial em relação a base canônica ortonormal de $\mathbb{C}^2$ é a matriz $H \in \mathbb{M}_{2,2}(\mathbb{C})$ dada por:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.23}$$

O operador *Hadamard* aplicado aos *qubits* $|0\rangle$ e $|1\rangle$ cria as seguintes superposições:

$$H|0\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |0\rangle = |+\rangle \tag{4.24}$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle - \frac{1}{\sqrt{2}} \cdot |0\rangle = |-\rangle \tag{4.25}$$

De forma geral, há infinitas portas quânticas para um único qubit, estas podem ser generalizadas como rotações em um dos eixos x, y e z da esfera de *Bloch*,

$$U_R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4.26}$$

e deslocamento de fase,

$$U_P(\phi_1, \phi_2) = \begin{bmatrix} e^{i\phi_1} & 0 \\ 0 & e^{i\phi_2} \end{bmatrix} \tag{4.27}$$

4.2.2 Circuitos Quânticos

Os circuitos lógicos clássicos são utilizados para descrever a aplicação dos operados (portas lógicas), sobre os diversos *bits*. A notação adotada para circuitos clássicos permite

expressar a ordem em que as operações são realizadas, mas esta notação não restringe como tais operações devem ser realizadas fisicamente.

O circuito quântico é o análogo ao circuito lógico clássico, este é um dispositivo que consiste de portas quânticas conectadas umas às outras e cujos passos computacionais são sincronizados ao longo do tempo. Como os circuitos lógicos clássicos, os circuitos quânticos são mais comumente apresentados e entendidos através de um gráfico ou diagrama. A Figura 4.2 apresenta o diagrama de um circuito consistindo de duas aplicações sucessivas da porta H a um *qubit*:

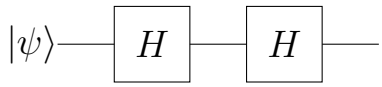

Figura 4.2: Circuito Quântico com duas portas *Hadamard*

Como primeiro elemento temos a entrada, representada na posição mais a esquerda. A entrada do circuito é o vetor de estado de um *qubit* ou um produto tensorial de *n-qubits* ($|\psi\rangle$) podendo estar agrupados ou em um ou mais registradores representando diferentes partes de entradas ou diferenciando *qubits* auxiliares. As linhas horizontais são análogas aos fios de um circuito clássico e representam a evolução do estado do *qubit* (passagem do tempo ou deslocamento de uma partícula). O sentido de leitura de um circuito quântico é da esquerda para direita.

As portas quânticas são representadas por retângulos e definem operações sobre os *qubits*. A saída é a posição mais a direita do circuito, podendo ser ou não medida. É o novo vetor de estado do *qubit* após duas aplicações sucessivas da porta H é $H^2|\psi\rangle$.

A Figura 4.3 apresenta um circuito cuja entrada consiste de dois *qubits* nos estados iniciais $|0\rangle$ e $|0\rangle$, respectivamente, duas portas *Hadamards* são aplicadas a cada *qubit* . A saída do circuito é o estado dos dois *qubits* após a aplicação das operações indicadas: $((|0\rangle + |1\rangle)/\sqrt{2})((|0\rangle + |1\rangle)/\sqrt{2}) = (|00\rangle + |01\rangle + |10\rangle + |11\rangle)/2$.

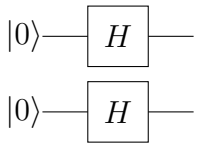

Figura 4.3: Circuito Quântico: Geração de estados superpostos com duas portas *Hadamard*

A ação do circuito da Figura 4.3 pode ser descrita como:

$$
\begin{aligned}
(H \otimes H)(|0\rangle|0\rangle) &= H^{\otimes 2}|00\rangle = H|0\rangle \otimes H|0\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\
&= \frac{1}{2}(|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle) \\
&= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \qquad (4.28)
\end{aligned}
$$

Em notação decimal:

$$
= \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle) \qquad (4.29)
$$

4.2.3 Portas Quânticas de Multi-Qubit

Até aqui, foram descritas portas que atuam apenas sobre um *qubit.*, contudo, as operações quânticas necessitam também portas que atuem sobre múltiplos *qubits*. Pode-se estender a noção de portas quânticas de 1-qbuit para portas quânticas *multi-qubit* por meio do produto tensorial dos operadores implementados pelas portas de *1-qubit*.

Para que a porta H de 1-qubit atue em um estado de *n-qubits*, denotado por $|\psi^n\rangle$ conforme Figura 4.4, efetuam-se o produto tensorial deste operador sucessivamente, obtendo $H^{\otimes n}$ e, consequentemente, a porta correspondente.

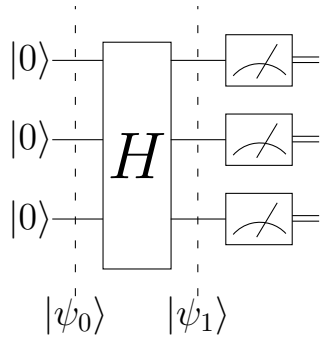

Figura 4.4: Aplicação da Porta *Hadamard* em *n-qubits*

O estado inicial do circuito da Figura 4.4 é dado por:

$$\begin{aligned} |\psi_0\rangle &= |0\rangle \otimes |0\rangle \otimes |0\rangle \\ &= |000\rangle \end{aligned} \tag{4.30}$$

O estado $|\psi_1\rangle$ corresponde à aplicação da porta *Hadamard* aos três *qubits* de entrada:

$$\begin{aligned} |\psi_1\rangle &= H^{\oplus 3}|\psi_0\rangle \\ &= H|0\rangle \otimes H|0\rangle \otimes H|0\rangle \\ &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \\ &= \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |010\rangle + \ldots + |111\rangle) \\ &= \frac{1}{\sqrt{8}}(|0\rangle + |1\rangle + |2\rangle + \ldots + |7\rangle) \end{aligned} \tag{4.31}$$

Como o estado $|\psi_1\rangle$ tem amplitudes igual para qualquer valor de 0 a 7, uma medição provocará um colapso para qualquer dos valores com igual probabilidade. O efeito provocado por esta porta faz dela um recurso de grande utilização por diversos algoritmos quântico consagrados [25].

De forma geral, pode-se usar portas quânticas de *1-qubit* para transformar o estado $|0\ldots0\rangle$ de n-qubits em qualquer estado do tipo $|\psi_1\rangle|\psi_2\rangle\ldots|\psi_n\rangle$, onde cada $|\psi_i\rangle$ é uma superposição arbitrária $\alpha|0\rangle + \beta|1\rangle$. O estado obtido por esta operação é do tipo produto-direto (ou separáveis). Para obtensão de estados emaranhados, precisa-se das portas Quânticas controladas.

4.2.4 Portas Quânticas Controladas

É possível construir combinações de portas logicas quânticas com *qubits* de controle. Suponha que U seja uma porta lógica quântica atuando sobre um ou mais *qubits*, se os *qubits* de controle estão no estado $|1\rangle$, então deve ser aplicada aos qubits alvos, caso os controles estejam no estado $|0\rangle$, não há qualquer alteração nos qubits alvo. Chamamos de

U-controlada a combinação de qualquer porta U combinada a um ou mais controles. Na Figura 4.5, $|\psi\rangle$ e $|\phi\rangle$ são controle e alvo respectivamente.

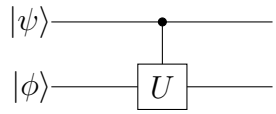

Figura 4.5: Porta U-controlada

Uma das portas de *multi-qubits* de grande utilização é a porta U_{CNOT} ou (NOT controlada) é uma porta que atua sobre dois *qubits* (controle e alvo) está representada na Figura 4.6. A porta U_{CNOT} atua da seguinte maneira: estando o *qubit* de controle no estado $|1\rangle$, então o *qubit* alvo é invertido; estando o *qubit* de controle no estado $|0\rangle$, então o *qubit* alvo permanece inalterado.

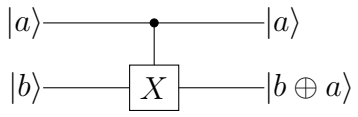

Figura 4.6: Porta CNOT a dois *qubits*, onde $|a\rangle$ representa o *qubit* de controle e $|b\rangle$ representa o *qubit* alvo.

A ação da porta $U_{CNOT} : \mathbb{C}^2 \otimes \mathbb{C}^2 \rightarrow \mathbb{C}^2 \otimes \mathbb{C}^2$ pode ser definida pelas transformações operadas nos estados da base canonica da seguinte forma:

$$\begin{aligned} U_{CNOT}|00\rangle &= |00\rangle \\ U_{CNOT}|01\rangle &= |01\rangle \\ U_{CNOT}|10\rangle &= |11\rangle \\ U_{CNOT}|11\rangle &= |10\rangle \end{aligned} \tag{4.32}$$

Observando os efeitos das transformações, pode-se concluir que o efeito da porta U_{CNOT} no *qubit* alvo pode ser representada por $X^a|b\rangle$, ou ainda por $a \oplus b$, em que $\oplus$ denota a soma

módulo 2. A matriz associada à porta U_{CNOT} é a seguinte:

$$U_{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.33}$$

O conjunto de portas de 1-*qubit* mais a porta U_{CNOT} compõem um conjunto de portas quânticas universais, ou seja, qualquer operação unitária pode ser representada por um circuito quântico que é uma combinação apenas de portas deste conjunto.

4.2.5 A porta quântica U_{CCNOT}

A porta U_{CNOT} pode ser generalizada acrescentando um ou mais *qubits* de controle, esta porta é denominada *operador de Toffoli* (Figura 4.7). A porta U_{CCNOT} *(controled-controled-NOT)* atua sobre um registro quântico de três *qubits* e o seu efeito é negar o estado do terceiro *qubit* se e somente se o estado dos dois primeiros é 11. O operador linear de *Toffoli* U_{CCNOT}: $\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$ é assim definido em relação ao vetor da base:

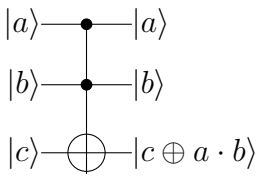

Figura 4.7: Porta *Toffoli* Quântica

$$U_{CCNOT}|000\rangle = |000\rangle \qquad U_{CCNOT}|100\rangle = |100\rangle \tag{4.34}$$

$$U_{CCNOT}|001\rangle = |001\rangle \qquad U_{CCNOT}|101\rangle = |101\rangle \tag{4.35}$$

$$U_{CCNOT}|010\rangle = |010\rangle \qquad U_{CCNOT}|110\rangle = |111\rangle \tag{4.36}$$

$$U_{CCNOT}|011\rangle = |011\rangle \qquad U_{CCNOT}|111\rangle = |110\rangle \tag{4.37}$$

Sua representação em relação à base ortonormal de $\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$ é a matriz unitária

$multi_qubit \in M_{8,8}(\mathbb{C})$ dada por:

$$U_{CCNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.38}$$

4.2.6 Módulos Lógicos Básicos Reversíveis

Os portas lógicas como AND, OR e XOR, entre outros, podem ser simulados a partir da porta *Toffoli*. Como exemplo temos a porta OR ($\vee$). A porta *Toffoli* aqui é usada com controles ativados em $|0\rangle$ e o qubit alvo no estado $|1\rangle$ A saída deste componente assume o estado $|1\rangle$ quando um dos *qubits* de entrada estiverem em $|1\rangle$ conforme figura 4.8.

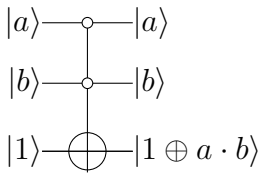

Figura 4.8: Simulação da porta *OR* Quântica a partir da Porta *Toffoli* Quântica

Capítulo 5

Algoritmos Quânticos

A maioria das portas lógicas clássicas são irreversíveis, no entanto, toda porta quântica é reversível. Surge naturalmente a questão. São os algoritmos clássicos quanticamente implementáveis? Ou seja, sempre existirão operadores lineares que possam implementar uma computação classicamente exequível?

Deutcsh [11] demonstrou que para toda função que possa ser classicamente implementada, é possível construir portas quânticas reversíveis. Qualquer circuito clássico pode ser substituído por um circuito equivalente contendo somente elementos reversíveis, fazendo uso da porta *Toffoli* [29]. Podemos simular as portas clássicas NAND e FAN-OUT utilizando a porta clássica *Toffoli*, estas duas portas são suficientes para todos os outros elementos de um circuito clássico, desta forma, vimos que circuitos clássicos podem ser simulados por circuitos reversíveis equivalentes. A porta *Toffoli* quântica pode ser usada para simular portas lógicas clássicas irreversíveis, similar a porta *Toffoli* clássica. Isso nos permite afirmar que toda operação realizada por um computador clássico pode ser pode ser realizada por um computador quântico.

Todavia, se simular computadores clássicos fosse a única característica dos computadores quânticos, não justificaria o esforço empreendido para o entendimento dos efeitos quânticos. A vantagem da computação quântica é o uso dos efeitos quânticos, tais como paralelismo e emaranhamento, no número de passos para a computação.

5.1 Paralelismo Quântico

O paralelismo quântico é uma característica essencial de muitos algoritmos quânticos. A melhor maneira de explicar o paralelismo quântico é visualizar o experimento da dupla fenda [2]. Nesse experimento, um canhão de elétrons é posicionado à frente de um anteparo com duas fendas atrás do anteparo é colocado um detector de elétrons, conforme Figura 5.1. Caso uma fenda seja fechada, o elétron comporta-se como partícula, apresentando o padrão $I_1(x)$ ou $I_2(x)$ da Figura 5.1. Quando as duas fendas estão abertas, o elétron comporta-se como onda, apresentando o padrão $I(x)$ de interferência entre ondas, Figura 5.1.

$$I(x) \neq I_1(x) + I_2(x) \tag{5.1}$$

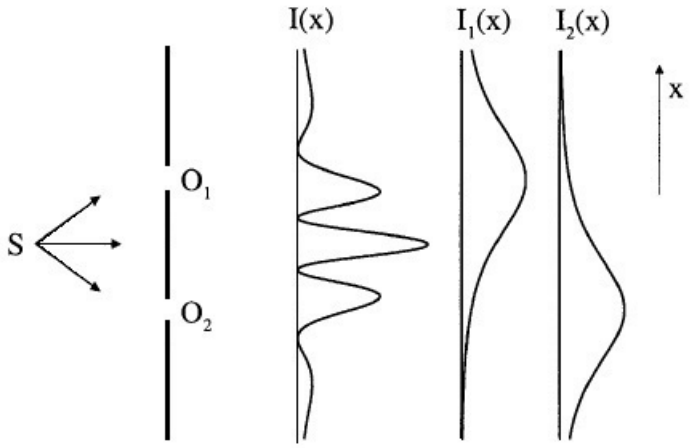

Figura 5.1: Diagrama esquemático do experimento da dupla-fenda. Uma fonte S emite luz, que pode passar por duas fendas O_1 e O_2antes de atingir uma tela. O padrão $I_1(x)$ é produzido quando apenas a fenda O_1 esta aberta e o padrão $I_1(x)$ apenas quando O_2 esta aberta. A intensidade $I(x)$ da luz na tela quando ambas as fendas estão abertas é diferente da soma algébrica das intensidades $I_1(x)$ e $I_1(2)$

Diferentemente do paralelismo clássicos onde vários circuitos diferentes são acionados simultaneamente para calcular $f(x)$, no paralelismo quântico, um único circuito é empregado para avaliar a função para múltiplos valores de x simultaneamente, explorando a capacidade de um computador quântico de poder estar em superposição de estados diferentes.

Este procedimento pode ser facilmente generalizado para funções com um número arbitrário de bits, usando a porta *Hadamard*. Essa operação consiste de n portas *Hadamard* atuando simultaneamente sobre n *qubits*. para o estado de dois *qubits* temos:

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} \tag{5.2}$$

de forma geral o resultado da transformação de *Hadamard* sobre n *qubits* inicializados no estado $|0\rangle$ é:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \tag{5.3}$$

A transformação de *Hadamard* produz uma superposição de todos os estados da base computacional, todas com a mesma amplitude. Note que isso é feito de forma extremamente eficiente, com 2^n estados sendo superpostos usando-se apenas n portas.

A avalização paralela de uma função de n bits, $f(x)$, pode ser feita da seguinte maneira, conforme Figura 5.2. Prepara-se o seguinte estado de $n+1$ *qubits*: $|0\rangle^{\otimes n}|0\rangle$, e então aplica-se a transformação de *Hadamard* nos primeiros n *qubits*, isso produz o estado:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|0\rangle \tag{5.4}$$

Em seguida aplica-se o circuito que implementa U_f como mostra a Figura 5.2.

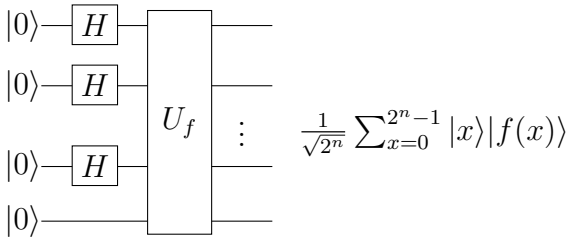

Figura 5.2: Computando valor de f para todos os valores de x.

Obtem-se então:

$$
\begin{aligned}
U_f|x\rangle|0\rangle &= U_f\left(\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle|0\rangle\right) \\
&= \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}U_f|x\rangle|0\rangle \\
&= \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle|f(x)\rangle \qquad (5.5)
\end{aligned}
$$

O circuito realizará a computação de todos os 2^n valores $f(0, f(1), \ldots, f(2^n - 1)$ ao mesmo tempo com uma única aplicação de U_f que implementa a função f. No entanto, esse paralelismo não pode ter seu uso de forma imediata, essa informação encontra-se no nível quântico. Uma medida do estado do registrador na saída do circuito, obtém-se apenas o valor da função num único ponto, não refletindo toda a informação contida na superposição dos estados.

5.2 Módulos Aritméticos e Lógicos Reversíveis

5.2.1 Somador

Uma versão reversível de um somador completo (*full adder*), pode ser construída a partir de 4 portas lógicas (U_{CNOT} e U_{CCNOT}) quânticas conforme Figura 5.3.

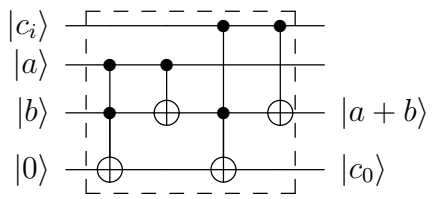

Figura 5.3: Somador Completo de um bit

A expansão deste módulo pode ser realizada em série ou cascata, para adição entre operadores de tamanho n. O primeiro *carry-in* é inicializado em 0 , e o *carry-out* da última soma corresponde ao s_n do resultado $Soma = |s_n s_{n-1} \ldots s_0\rangle$ [31].

Sua representação em relação à base ortonormal de $\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$ é a matriz unitária

$A \in M_{16,16}(\mathbb{C})$ dada por:

$$Somador = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix} \tag{5.6}$$

A aplicação do somador reversível a dois *qubits* iniciados em $|0\rangle$ e colocados em estados de superposição através da porta *Hadamard* produz:

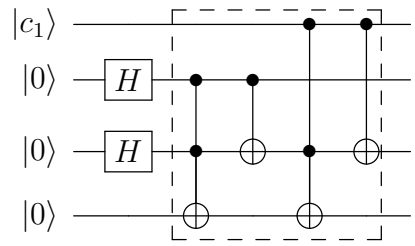

Figura 5.4: Aplicação do Somador Completo com qubits em estado de superposição

$$\begin{aligned} |\psi_0\rangle &= |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle \\ |\psi_1\rangle &= |0\rangle \otimes H|0\rangle \otimes H|0\rangle \otimes |0\rangle = \frac{1}{2}(|0000\rangle + |0010\rangle + |0100\rangle + |0110\rangle) \\ |\psi_2\rangle &= \frac{1}{2}(|0000\rangle + |0010\rangle + |0110\rangle + |0101\rangle) \end{aligned}$$

5.2.2 Subtrator

Para a criação do subtrator reversível, basta executar o somador de modo reverso.

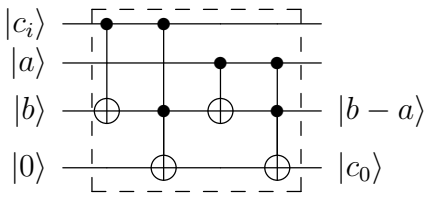

Figura 5.5: Subtrator Completo

Também é possível a expansão em série com subtratores reversíveis para palavras de tamanho n. O último borrow representa s_n. O bit mais significativo do resultado $Subtracao = |s_n s_{n-1} \dots s_0\rangle$, e seu estado em $|1\rangle$ indica resultado negativo da operação. Neste caso, a saída da subtração equivale a $2^{n-1} - (A - B)$ [31].

Sua representação em relação à base ortonormal de $\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$ é a matriz unitária $A \in M_{16,16}(\mathbb{C})$ dada por:

$$Somador = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix} \tag{5.7}$$

5.2.3 Multiplicador

O multiplicador reversível pode ser construído a partir de somadores controlados. Estes realizam a soma se o estado do controle for $|1\rangle$, ou retornam o segundo operando caso o estado do controle for $|0\rangle$ [31].

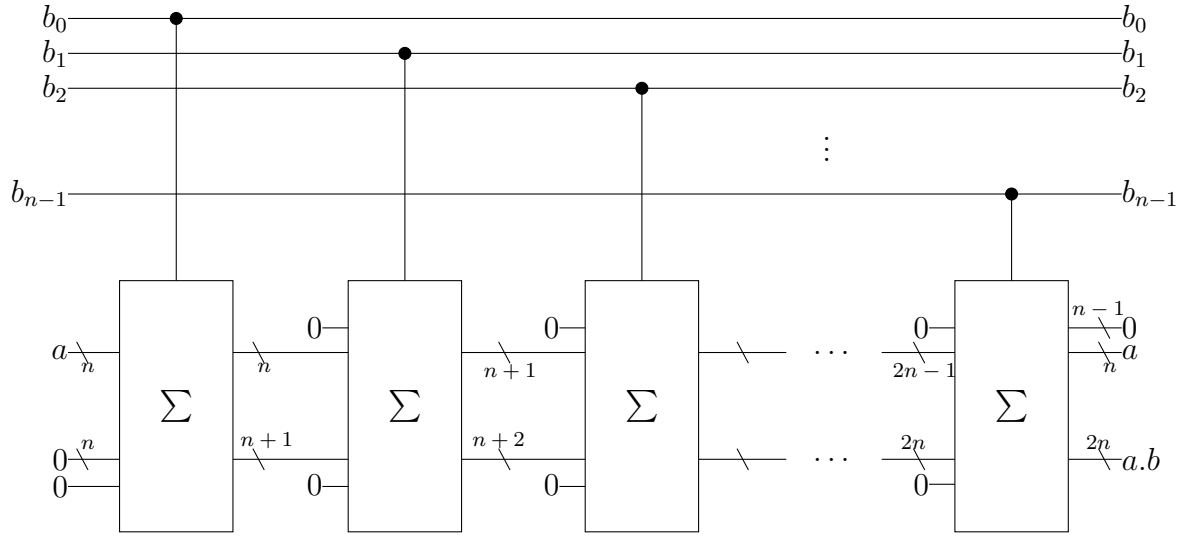

Figura 5.6: Circuito Multiplicador executando $a.b$, onde $b = |b_{n-1} \dots b_0\rangle$. Algumas linhas de entrada são agrupadas em uma única linha de saída [19].

A primeira soma controlada inicia-se com os estados $|2^0.a\rangle$ e $|0\rangle$. Se $|b_1\rangle$, a operação é realizada, e o resultado $|2^0.a\rangle$, equivalente a a multiplicado por b_0, será o segundo operando do próximo somador. Quando $b_i = |0\rangle$, a multiplicação prossegue com o resultado anterior ao somador controlado por b_i, uma vez que o segundo operando é simplesmente repassado quando o controle não é ativado [27].

A aplicação da superposição a uma das entradas de um multiplicador reversível produz os múltiplos da entrada não superposta inicialmente, isto é, dado $|A\rangle_n$ o registro superposto e $|B\rangle_m$ o registro não superposto, O registro quântico resultante da multiplicação contém todos os múltiplos de $|B\rangle_m$ na forma $|A.B\rangle_{n+m}$.

5.2.4 Comparador

O comparador reversível proposto em [27] é um sistema combinacional que indica a relação entre magnitudes das palavras de entrada. Este comparador analisa se $A > B$ e

acrescenta uma porta para indicar se $A \neq B$. O comparador utiliza o modulo subtrator 5.5 e uma porta *Toffoli* conforme figura 5.7.

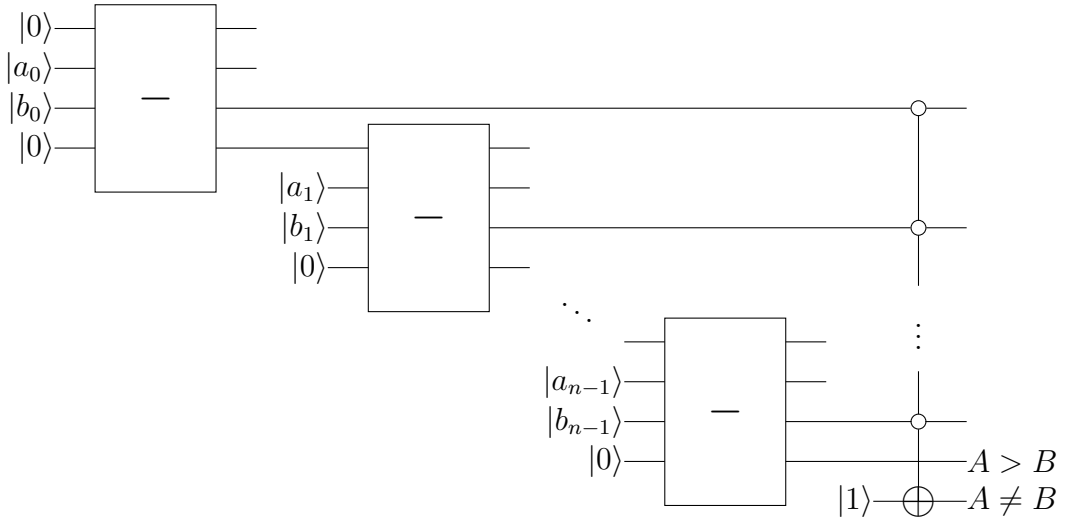

Figura 5.7: Circuito Comparador reversível. Se todas as subtrações correspondentes resultarem no estado $|0\rangle$, as duas palavras não são diferentes [27].

5.3 O Algoritmo de Grover

Em computação, problemas de busca se resumem em como encontrar um elemento em um conjunto de dados. Se os dados estão estruturados, é possível explorar essa estrutura afim de construir algoritmos eficientes para encontrar o elemento. Em um conjunto de dados não-estruturados, não se conhece nada a respeito de como os elementos do conjunto estão organizados. O *algorítimo de Grover* [14], foi desenvolvido inicialmente para a busca em um conjunto não-estruturado (lista não ordenada), mas seu uso mais eficiente não será desta forma [29]. A aplicação do *algoritmo de Grover* proporciona uma melhora quadrática no desempenho em comparação ao algoritmo clássico de busca, diferentemente de outros algoritmos quânticos, que proporcionam melhoras exponenciais em relação a seus similares clássicos. A complexidade do *algoritmo de Grover* é $O(\sqrt{n})$.

O *algoritmo de Grover* pode ser utilizado para resolver problemas NP-completos com procuras exaustivas por todo o espaço de soluções. Isto resulta em um ganho de desempenho considerável em relação aos algoritmos clássicos, mas não alcança tempo polinomial para problemas NP-completos.

5.3.1 Problema de Busca

De acordo com a definição da *Maquina de Turing* [39], de forma geral, um procedimento computacional pode ser visto como um conjunto de operações aplicadas sobre uma sequência de *bits* retornando outra sequência de *bits*. este procedimento pode ser descrito por uma função booleana $f : \{0,1\}^n \to \{0,1\}^m$, cuja entrada é um estado $x = (x_{k-1}, \ldots, x_0) \in \{0,1\}^n$ e cuja saída é um estado $y = (f_{m-1}(x), \ldots, f_0(x)) \in \{0,1\}^m$ sendo que f possui um custo associado ao número de operações necessárias para retomar o resultado para cada entrada possível.

Um problema de grande importância é, dada uma função f e um valor k, encontrar uma possível entrada x tal que $f(x) = k$. Caso a função seja bijetora, basta avaliar a função inversa $f^{-1}(k)$. Porem de forma geral há varias soluções possíveis sendo valida a escolha de qualquer uma delas. O problema resume-se a procurar as soluções no conjunto imagem da função. Caso não haja informação alguma sobre a estrutura ou propriedade interna da função como, por exemplo, distribuição das soluções, o caminho que pode-se seguir é analisar todo o domínio da função comparando os resultados. Esta metodologia é conhecida como ataque de força bruta ou busca exaustiva.

Suponha que o domínio da funçãof possui tamanho N. Ao avaliarmos $f(x)$, a única maneira de encontramos um valor x tal que $f(x) = k$ é fazer uma busca linear. Se tentarmos encontrar o valor x no domínio que contém N elementos examinando apenas K elementos, temos a probabilidade $\frac{K}{N}Q$ de encontrar o valor de x, onde Q é a probabilidade do elemento x estar presente no conjunto. Temos então que examinar N elementos afim de encontramos o elemento procurado com uma probabilidade Q.

Num conjunto de dados não estruturados, no melhor caso temos que analisar a função uma única vês. No pior caso teremos que fazer N avaliações e em média $\frac{N+1}{2}$ avaliações.

5.4 Descrição do Algoritmo

Em seu artigo, *Grover* [14] aborda a busca para encontrar um elemento específico dentro de uma lista não-ordenada sendo necessários $O\left(\sqrt{N}\right)$ aplicações do operador G. Em aplicações práticas, depara-se diversas vezes com problemas onde mais de um elemento da

lista satisfaz o critério utilizado na busca. quando o número de soluções que satisfaz o critério é maior que um, podemos usar a generalização descrita por Goong Chen e Shuunhua Sum [8].

Suponha que se tenha um problema de busca definido por uma função

$$f : \{0,1\}^n \to \{0,1\}, \tag{5.8}$$

definida por

$$f(i) = \begin{cases} 1, \text{se } i = i_0, \\ 0, \text{se } i \neq i_0, \end{cases} \tag{5.9}$$

que será utilizada para identificação do elemento procurado.

Suponha que há M elemento na lista onde $i \in \{0, 1, \ldots, N-1\}$ tal que $f(i_0) = 1$, mas $f(i) = 0$ para todos os outros valores. A idéia básica é criar um estado de superposição na entrada e gira-lo em direção a $|x\rangle$ usando o *Operador de Grover* G.

O *Algoritmo de Grover* utiliza dois registradores quânticos. o primeiro registrador composto de $n-qubits$ iniciados no estado $|0\ldots0\rangle$, e o segundo com um qubit iniciado no estado $|1\rangle$.

Ao primeiro registrador inicializado como $|0\rangle^{\otimes n}$, a este é aplicado o operador $H^{\otimes n}$ 4.23 afim de gerar a superposição dos estados. Este registrador esta relacionado aos elementos da lista onde será feita a busca.

A superposição será chamada por $|\psi\rangle$, ou seja,

$$|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{1=0}^{N-1} |i\rangle \tag{5.10}$$

O segundo registrador será inicializado no estado $|1\rangle$ a este, também será aplicado o operador H. Este registrador tera papel fundamental na marcação do elemento procurado.

O estado do segundo registrador será denotado por $|-\rangle$.

$$|-\rangle = H|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \tag{5.11}$$

5.5 Operadores do Algoritmo

5.5.1 Rotação de Fase

O operador linear que representa a função f, em 5.9, utilizada para a identificação do elemento procurado, pode ser visto como um operador linear U_f que transforma $|i\rangle$ em $|f(i)\rangle$, onde $|i\rangle$ é o estado de *n-qubits* do primeiro registrador.

$$|i\rangle|0\rangle \xrightarrow{U_f} |i\rangle|f(i)\rangle \tag{5.12}$$

aqui, o primeiro e o ultimo registrador tem na entrada e na saída, *n-qubits* e um qubit respectivamente. Aplicado $_f$ temos,

$$U_f(|i\rangle|0\rangle) = \begin{cases} |i\rangle|1\rangle, \text{se } i = i_0, \\ |i\rangle|0\rangle, \text{se } i \neq i_0. \end{cases} \tag{5.13}$$

e,

$$U_f(|i\rangle|1\rangle) = \begin{cases} |i\rangle|0\rangle, \text{se } i = i_0, \\ |i\rangle|1\rangle, \text{se } i \neq i_0. \end{cases} \tag{5.14}$$

O operador altera o estado do segundo registrador quando o primeiro registrador representa o elemento procurado. O operador U_f é definido como um *oráculo quântico*, uma caixa preta que tem a capacidade de reconhecer as soluções do problema de busca por meio de um *qubit-oráculo*.

A aplicação de U_f em 5.13 e 5.14 será representada como,

$$U_f(|i\rangle|j\rangle) = |i\rangle|j \oplus f(i)\rangle, \tag{5.15}$$

onde $|i\rangle$ é o estado de *n-qubits* do primeiro registrador, $|j\rangle$ é o estado de um *qubit* do segundo registrador *(qubit-oráculo)*, este é um *qubit* que será invertido se $f(i) = 1$, nada acontecendo caso contrário, $\oplus$ é a soma módulo 2 e $U_f \in \mathbb{C}^{(2^{n+1}\times 2^{n+1})}$.

Quando o tamanho da lista não for uma potência de 2, a superposição de todas as entradas possíveis de f pode ser feita utilizando a Transformada de Fourier Quântica conforme [22].

Após a superposição dos estados, o passo seguinte do *Algoritmo de Grover* é a aplicação do operador U_f sobre o estado $|\psi\rangle|-\rangle$ conforme Figura 5.2,

$$U_f(|\psi\rangle|-\rangle) = U_f\left(\left(\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle\right)|-\rangle\right) \tag{5.16}$$

Utilizando as propriedades do produto tensorial e da linearidade do operador U_f,

$$U_f(|\psi\rangle|-\rangle) = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}U_f(|i\rangle|-\rangle) \tag{5.17}$$

Substituindo 5.11 em 5.17 e novamente da linearidade do operador U_f,

$$U_f(|\psi\rangle|-\rangle) = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}\frac{1}{\sqrt{2}}(U_f(|i\rangle|0\rangle) - U_f(|i\rangle|1\rangle)) \tag{5.18}$$

Aplicando a definição de U_f de 5.15 em 5.18,

$$U_f(|\psi\rangle|-\rangle) = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}\frac{1}{\sqrt{2}}(|i\rangle(|f(i)\rangle - |1 \oplus f(i)\rangle)) \tag{5.19}$$

De f em 5.9 temos,

$$|i\rangle(|f(i)\rangle - |1 \oplus f(i)\rangle) = \begin{cases} |i\rangle(|1\rangle - |0\rangle), \text{se } i = i_0, \\ |i\rangle(|0\rangle - |1\rangle), \text{se } i \neq i_0. \end{cases} \tag{5.20}$$

Substituindo 5.19 em 5.20 e novamente aplicando a definição de $|-\rangle$ de 5.11 temos:

$$U_f(|\psi\rangle|-\rangle) = \left(\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{f(i)} |i\rangle\right) |-\rangle. \tag{5.21}$$

Após a aplicação do operador U_f sobre o estado $|\psi\rangle$, a função f foi avaliada em todos os elementos da lista com uma única aplicação do operador. O estado do segundo registrador não foi alterado, mas este foi essencial na identificação do elemento procurado. O estado do primeiro registrador permanece sendo uma superposição de todos os estados associados aos elementos da lista.

O estado do primeiro registrador será denotado por $|\psi_1\rangle$, isto é,

$$|\psi_1\rangle = \sum_{i=0}^{N-1} (-1)^{f(i)} |i\rangle \tag{5.22}$$

No entanto, a fase dos elementos procurados foram alterados de $\frac{1}{\sqrt{N}}$ para $-\frac{1}{\sqrt{N}}$. Os estados associados foram identificados, mas essas informações estão disponíveis apenas quanticamente. Uma tentativa de medir o primeiro registrador poderá não resultar em sucesso, pois a probabilidade de se obter o elemento procurado é

$$\left|\frac{-1}{\sqrt{N}}\right|^2 = \frac{1}{N} \tag{5.23}$$

O operador que produz a rotação de fase é representada por uma matriz diagonal unitária, com $A_{kl} = 0$ se $k \neq l$ e $A_{kl} = e^{i\phi_k}$, onde ϕ_k é um número real qualquer e $i = \sqrt{-1}$. Abaixo

segue um exemplo para um sistema onde $n = 4$.

$$R = \begin{bmatrix} e^{i\phi_1} & 0 & 0 & 0 \\ 0 & e^{i\phi_2} & 0 & 0 \\ 0 & 0 & e^{i\phi_3} & 0 \\ 0 & 0 & 0 & e^{i\phi_4} \end{bmatrix} \tag{5.24}$$

Da formula de *Euler*,

$$e^{ix} = \cos x + i \sin x \tag{5.25}$$

$$e^{-ix} = \cos x - i \sin x \tag{5.26}$$

as entradas diagonais de 5.24 são equivalentes a $\cos \phi_k + i \sin \phi_k$.

O *Algoritmo de Grover* se utiliza de uma rotação seletiva de fase, ou seja, é necessário uma matriz que rotacione em π radianos apenas a fase do estado marcado. Esta matriz será uma matriz diagonal de com $1's$ ao longo da diagonal com exceção dos k-ésimos elementos da diagonal relacionados aos elementos procurados, estes terão o valor -1.

Uma interpretação geométrica do efeito da aplicação do operador U_f sobre o primeiro registrador. U_f aplicado sobre um vetor unitário qualquer gerado pelos elementos da base computacional $\{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$ resulta numa reflexão desse vetor em relação ao subespaço ortogonal a $|i_0\rangle$, gerado por todos os outros elementos da base computacional [32]. Essa visualização pode ser feita considerando essa reflexão como uma reflexão em relação a projeção de $|v\rangle$ sobre o subespaço ortonormal $|i_0\rangle$. Seja $|u\rangle$ o vetor unitário na direção dessa projeção conforme Figura 5.8, temos:

$$|u\rangle = \frac{1}{\sqrt{N-M}} \sum_{i=0, i \neq i_0}^{N-1} |i\rangle \tag{5.27}$$

$$|i_0\rangle = \frac{1}{\sqrt{M}} \sum_{i=0, i=i_0}^{N-1} |i\rangle \tag{5.28}$$

e $|\psi\rangle$ pode ser escrito como:

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|u\rangle + \sqrt{\frac{M}{N}}|i_0\rangle \tag{5.29}$$

O produto interno dos estados envolvidos no *Algoritmo de Grover* sempre será um número real devido as amplitudes dos estados serem números reais, permitindo a comparação entre ângulos de quaisquer dois pares de estados. O ângulo formado entre $|\psi\rangle$ e $|i_0\rangle$ e entre $|u\rangle$ e $|i_0\rangle$ são respectivamente:

$$\langle\psi|i_0\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}\langle\psi|i_0\rangle = \frac{M}{\sqrt{N}}\langle i_0|i_0\rangle = \frac{M}{\sqrt{N}} \tag{5.30}$$

$$\langle u|i_0\rangle = \frac{1}{\sqrt{N-M}}\sum_{i=0,i\neq i_0}^{N-1}\langle i|i_0\rangle = 0 \tag{5.31}$$

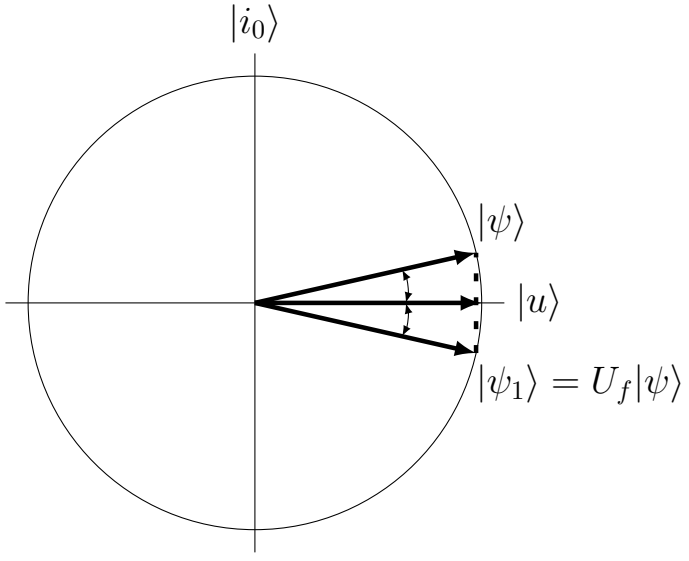

Figura 5.8: Interpretação geométrica da ação de U_f sobre o estado $|\psi\rangle$.

5.5.2 Inversão Sobre a Média

O operador que produz a inversão sobre a média no vetor de estados atua aumentando ou diminuindo a amplitude de um estado.

O ângulo entre $|u\rangle$ e $|i_0\rangle$ é $\frac{\pi}{2}rad$ e entre $|\psi\rangle$ e $|i_0\rangle$ é menor do que $\frac{\pi}{2}rad$ (quanto maior N, mais proximo de $\frac{\pi}{2}rad$).

O aumento da amplitude é conseguida refletindo o vetor $|\psi_1\rangle$ em relação ao vetor $|\psi\rangle$, aumentando a amplitude do elemento procurado em relação a amplitude do estado $|\psi\rangle$ conforme Figura 5.9.

A reflexão de $|\psi_1\rangle$ sobre $|\psi\rangle$ é dada por:

$$2\langle\psi|\psi_1\rangle|\psi\rangle - |\psi_1\rangle \tag{5.32}$$

$$\begin{aligned} 2\langle\psi|\psi_1\rangle|\psi\rangle - |\psi_1 &= (2|\psi\rangle\langle\psi|)|\psi_1\rangle \\ &= (2|\psi\rangle\langle\psi| - I)|\psi_1. \end{aligned} \tag{5.33}$$

Ou seja, o operador procurado é

$$2|\psi\rangle\langle\psi| - I \tag{5.34}$$

Onde I é o operador identidade.

Segue que o estado $|\psi_1\rangle$ 5.21, pode ser reescrito como

$$|\psi_1\rangle = |\psi\rangle - \frac{2M}{\sqrt{N}}|i_0\rangle \tag{5.35}$$

onde $|\psi\rangle$ é escrito como 5.10 e i_0 é o elemento procurado.

Aplicando o operador 5.34 sobre o estado $|\psi_1\rangle$ e usando 5.35, temos:

$$\begin{aligned} (2|\psi\rangle\langle\psi| - I)|\psi_1\rangle &= (2|\psi\rangle\langle\psi| - I)\left(|\psi_1\rangle = |\psi\rangle - \frac{M}{\sqrt{N}}|i_0\rangle\right) \\ &= (2\langle\psi|\psi\rangle)|\psi\rangle - \left(\frac{4M}{\sqrt{N}}\langle\psi|i_0\rangle\right)|\psi\rangle - |\psi\rangle + \frac{2M}{\sqrt{N}}|i_0\rangle \end{aligned} \tag{5.36}$$

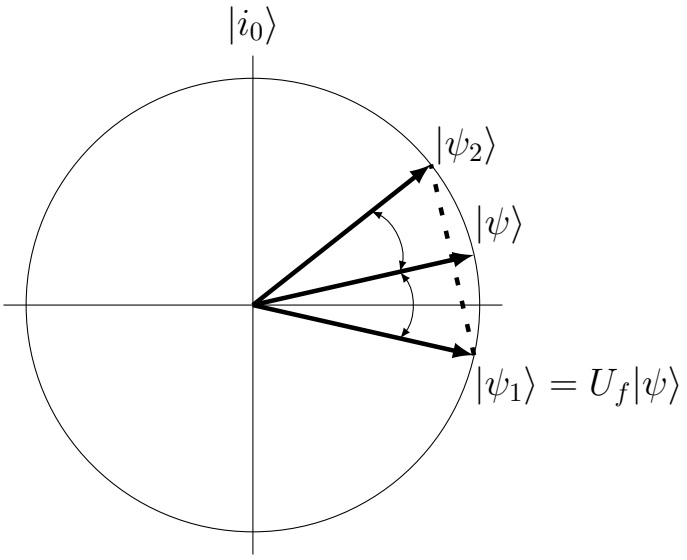

Figura 5.9: Reflexão de $|\psi_1\rangle$ em relação a $|\psi\rangle$.

Substituindo 5.30 em 5.36, temos:

$$(2|\psi\rangle\langle\psi| - I)|\psi_1\rangle = \frac{N-4M^2}{N}|\psi\rangle + \frac{2M}{\sqrt{N}}|i_0\rangle \tag{5.37}$$

Que é o estado resultante dos operadores U_f e $2|\psi\rangle\langle\psi| - I$. A composição destes dois operadores é chamada de *operador de Grover* G e o estado resultante é descrito como $|\psi_G\rangle$

Na base $|\alpha\rangle$,$|\beta\rangle$, a iteração de Grover pode ser escrita como

$$G = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{5.38}$$

A amplitude dos estados $|i_0\rangle$'s logo após a primeira aplicação pode ser de 5.37,

$$\left(\frac{N-4m^2}{N}\right)\left(\frac{1}{N}\right) + \frac{2M}{\sqrt{N}} = \frac{N(2M+1)-4M^2}{N\sqrt{N}} \tag{5.39}$$

O operador de inversão de fase 5.34, é representada pela matriz $N \times N$:

$$D = \begin{bmatrix} \frac{2}{N} - 1 & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} - 1 & \cdots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} - 1 \end{bmatrix} \tag{5.40}$$

Após repetidas aplicações do operador G o vetor de estado se aproxima de $|i_0\rangle$, ponto no qual uma observação na base computacional retorna uma solução para o problema da busca com alta probabilidade. O custo do *algoritmo de Grover* está no número de vezes que o operador G deve ser aplicado para que o ângulo entre $|i_0$ e $G^k|\psi\rangle$ seja o mais próximo de 0.

$$\arccos(\langle\psi|i_0\rangle) - k\theta = 0 \tag{5.41}$$

Onde θ é o ângulo de rotação produzido pelo operador G.

O ângulo θ formado entre os vetores $|i_0$ e $G|\psi\rangle$ pode ser calculado a partir de 5.30 e 5.37,

$$\begin{aligned} \cos\theta &= \langle\psi|\psi_G\rangle \\ &= \frac{N-4M^2}{N}\langle\psi|\psi\rangle + \frac{2}{\sqrt{N}}\langle\psi|i_0\rangle \\ &= \frac{N-4}{N} + \frac{2}{\sqrt{N}}\left(\frac{M}{\sqrt{N}}\right) \\ &= \frac{N-4M^2+2M}{N} \end{aligned} \tag{5.42}$$

$$\theta = \arccos\left(\frac{N-4M^2+2M}{N}\right) \tag{5.43}$$

Substituindo 5.30 e 5.43 em 5.41 temos:

$$\arccos\left(\frac{M}{\sqrt{N}}\right) - k\arccos\left(\frac{N-4M^2+2M}{N}\right) = 0 \tag{5.44}$$

$$k = \frac{\arccos\left(\frac{M}{\sqrt{N}}\right)}{\arccos\left(\frac{N-4M^2+2M}{N}\right)} \tag{5.45}$$

k é o número de aplicações do operador G conforme Figura 5.10. Onde,

$$\lim_{N\to\infty} \frac{k}{\sqrt{N}} = \frac{\pi}{4} \tag{5.46}$$

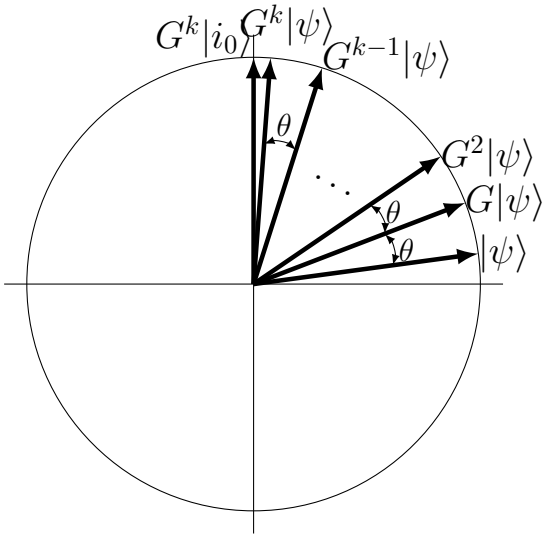

Figura 5.10: Aplicações sucessivas do operador G.

Para uma lista de tamanho N e M itens satisfazendo o critério de busca, são necessárias $O\left(\sqrt{N\backslash M}\right)$ aplicações de G. Onde M é o número de elementos que satisfazem a função 5.9.

5.5.3 Passos do Algoritmo de Grover

O procedimento geral do algoritmo de *Grover* pode ser visto na Figura 5.11 e está descrito no Algoritmo 1 abaixo:

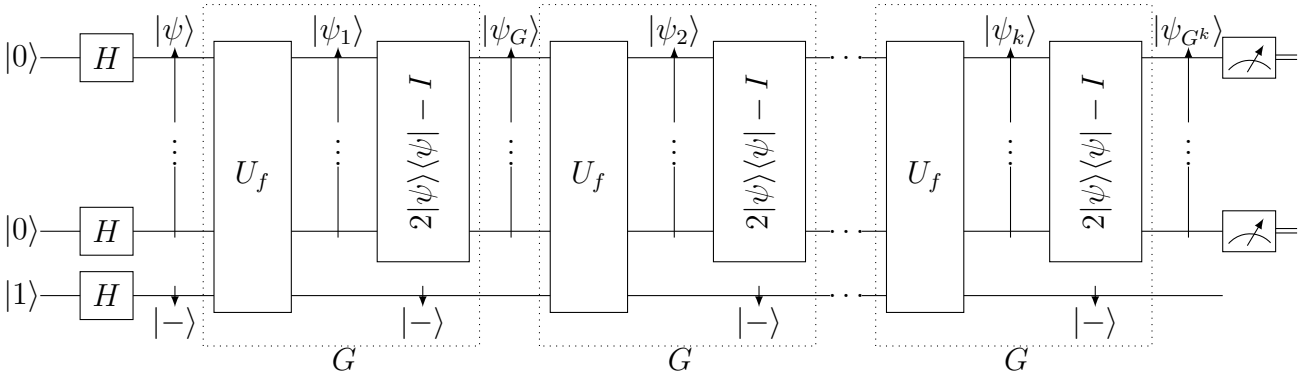

Figura 5.11: Aplicação do *Algoritmo de Grove*

1. É preparado o estado inicial;
2. É Aplicado o operador de *Hadamard* para criação dos estados superpostos;
3. Loop para aplicação dos operadores de rotação de fase e inversão sobre a média;
4. Aplicação da rotação de fase (Oráculo);
5. Inversão sobre a média;
6. Fim do loop;
7. Leitura do estado do primeiro registrador.

Algorithm 1 Algoritmo de Grover

$|\psi_0\rangle \leftarrow |0\rangle_n|1\rangle_1$

$|\psi_1 \leftarrow H^{\otimes n}|0\rangle_n H|1\rangle_1$

para $j \leftarrow 1$ **até** $\lfloor \frac{\pi}{4}\sqrt{N} \rceil$ **faça**

$|\psi_{2j}\rangle \leftarrow U_g|\psi_{(2j-1)}\rangle$

$|\psi_{(2j+1)}\rangle \leftarrow (2|\psi_1\langle\psi_1| - I)|\psi_{2j}\rangle$

fim para

Observe o estado do primeiro registrador

5.5.4 Exemplo $N = 4$

O número de *qubits* necessários para para uma lista de 4 elementos é 2 *qubits*. O numero de aplicações pode ser calculado a partir de 5.45.

$$\begin{aligned} k &= \frac{\arccos\left(\frac{1}{\sqrt{4}}\right)}{\left(\frac{4-4.1^2+2.1}{4}\right)} \\ k &= \frac{\arccos\left(\frac{1}{\sqrt{4}}\right)}{\left(\frac{1}{2}\right)} \\ k &\cong 1,33 \end{aligned}$$

Arredondando o valor de k para o inteiro mais próximo, será aplicado o operador de Grover apenas uma vez.

Antes da aplicação de G, é criada a superposição $|\psi\rangle$ formada por todos os elementos da base.

$$\begin{aligned} |\psi\rangle &= H|0\rangle \otimes H|0\rangle \\ &= \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \\ &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \end{aligned}$$

Em notação decimal temos:

$$|\psi\rangle = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle)$$

O estado $|\psi\rangle$ é descrito pelo vetor

$$= \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$$

supondo que o elemento procurado seja $|i_0\rangle = |11\rangle = |3\rangle$, é aplicado o operador U_f sobre o estado $|\psi\rangle|-\rangle$. O elemento procurado é o único que tem sua amplitude alterada.

$$\begin{aligned} |\psi_1\rangle|-\rangle &= U_f(|\psi\rangle|-\rangle) \\ &= \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle - |3\rangle) \end{aligned}$$

A matriz de inversão de fase será,

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

O passo seguinte é a aplicação do operador $2|\psi\rangle\langle\psi| - I$ sobre o estado $|\psi_1$, produzindo o estado $|\psi_G\rangle$:

$$\begin{aligned} |\psi_G\rangle &= (2|\psi\rangle\langle\psi| - I)|\psi_1\rangle \\ &= \frac{3}{2}|\psi\rangle - |\psi_1\rangle \\ &= 1|3\rangle \end{aligned}$$

A matriz que representa a aplicação do operador $2|\psi\rangle\langle\psi| - I$ é,

$$= \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

A aplicação do operador de Grover dará com 100% de probabilidade o elemento procurado.

As operações matriciais para a aplicação do operador de Grover para o exemplo segue abaixo:

$$\begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

5.5.5 Número Desconhecido de Soluções

Até o momento, as situações descritas, decorrem do conhecimento do número de soluções, um caso muito mais interessante ocorre quando o número de soluções não é conhecida de antemão. Se decidirmos fazer uma iteração $\pi/4\sqrt{N}$ vezes, o que daria quase certeza de encontrar uma solução, se houvesse apenas uma, a probabilidade diminuiria caso o número de soluções fosse maior que um e mantivéssemos o mesmo numero de iterações. Por exemplo, há uma única solução entre 2^{20} possibilidades, serão necessárias 804 iterações. O mesmo número de iterações iria produzir uma solução com uma probabilidade inferior a um em um milhão caso haja quatro soluções. A fim de encontrar uma solução de forma eficiente quando o seu número é desconhecido, precisamos dos seguintes lemas, o primeiro dos quais é provado por álgebra simples.

5.5.6 BBHT

O *BBHT* foi um algoritmo proposto para aplicação quando não se conhece o número de soluções.

Lema 5.1. *[6] Para quaisquer números reais α e β, e qualquer inteiro positivo m,*

$$\sum_{j=0}^{m-1} \cos\alpha + 2\beta j) = \frac{\sin(m\beta)\cos(\alpha + (m-1)\beta)}{\sin\beta} \tag{5.47}$$

En particular, quando $\alpha = \beta$,

$$\sum_{j=0}^{m-1} \cos((2j+1)\alpha) = \frac{\sin(2m\alpha)}{2\sin\alpha} \tag{5.48}$$

Lema 5.2. *[6] Seja t o número desconhecido de soluções e seja θ tal que $\sin^2\theta = t/N$. Seja m um inteiro positivo arbitrário. Seja j um número escolhido randomicamente de acordo com uma distribuição uniforme entre 0 e $m-1$. Se for observado o registro após a aplicação de j interações do* Algoritmo de Grover *a partir do estado inicial $|\psi\rangle = \sum_i 1\sqrt{N}$, a probabilidade de se obter a solução é exatamente*

$$P_m = \frac{1}{2} - \frac{\sin(4m\theta)}{4m\sin(2\theta)} \tag{5.49}$$

Em particular $P_m \geq 1/4$ quando $m \geq 1/\sin(2\theta)$.

O algoritmo *BBHT* [6], foi proposto para encontrar uma solução quando o número t de soluções é desconhecida. Por simplicidade, assume-se inicialmente que $1 \leq t \leq 3N/4$.

Algorithm 2 Algoritmo BBHT

Inicializar $m = 1$ e $\lambda = 6/5$ (λ pode ser qualquer valor entre 1 e 4/3).

Escolher j uniformemente ao acaso entre os inteiros não negativos menores que m.

Aplica-se j interações do algoritmo de Grover a partir do estado inicial $|\psi\rangle = \sum_i \frac{1}{\sqrt{N}}$

Observe o registro: obtendo i como resultado.

Se $T[i] = x$, o problema está resolvido. Fim.

Caso contrário, defina m como $mim(\lambda m, \sqrt{N})$ e volte para o passo 2.

O algoritmo *BBHT*, encontra uma solução em tempo esperado de $O(\sqrt{N/t})$.

5.5.7 Contagem Quântica

É possível estimar o número de soluções em um problema de busca por meio da combinação da iteração de *Grover* com a técnica de estimativa de fase baseada na *Transformada de Fourier Quântica*. O conhecimento da estimativa do número de soluções, permite também encontrar uma solução, mesmo que sendo o número total desconhecido [29].

Sejam $|a\rangle$ e $|b\rangle$ dois autovalores da iteração de Grover, no espaço definido por $|\alpha\rangle$ e $|\beta\rangle$. Seja θ o ângulo de rotação determinado pela iteração de Grover. de 5.38, os autovalores correspondentes são $e^{i\theta}$ e $e^{i(2\pi-\theta)}$. Afim de simplificar, tomemos o espaço de busca expandido para $2N$ acrescentando um *qubit* $|0\rangle$ ao espaço de busca N, assegurando que $sin^2(\theta/2) = M/2N$.

A contagem quântica estima θ com precisão de m *bits* e probabilidade de sucesso de pelo menos $1 - \epsilon$. A Figura 5.12 representa o circuito da contagem quântica, onde o primeiro registrador contém $t \equiv m + \lceil \log(2+1/2\epsilon) \rceil$ *qubits*, e o segundo registrador contém $n+1$ *qubits* referentes ao número de *qubits* necessário ao Algorítimo de Grover no espaço aumentado. A *Transformada de Fourier* retornará uma estimativa de fase de θ ou $2\pi - \theta$ (a estimativa de ambos são equivalentes), com precisão $|\Delta\theta| \leq 2^{-m}$, com probabilidade de pelo menos $1 - \epsilon$.

A partir da equação $sin^2(\theta/2) = M/2N$ e a estimativa de fase de θ, obtém-se uma estimativa do número de soluções, M.

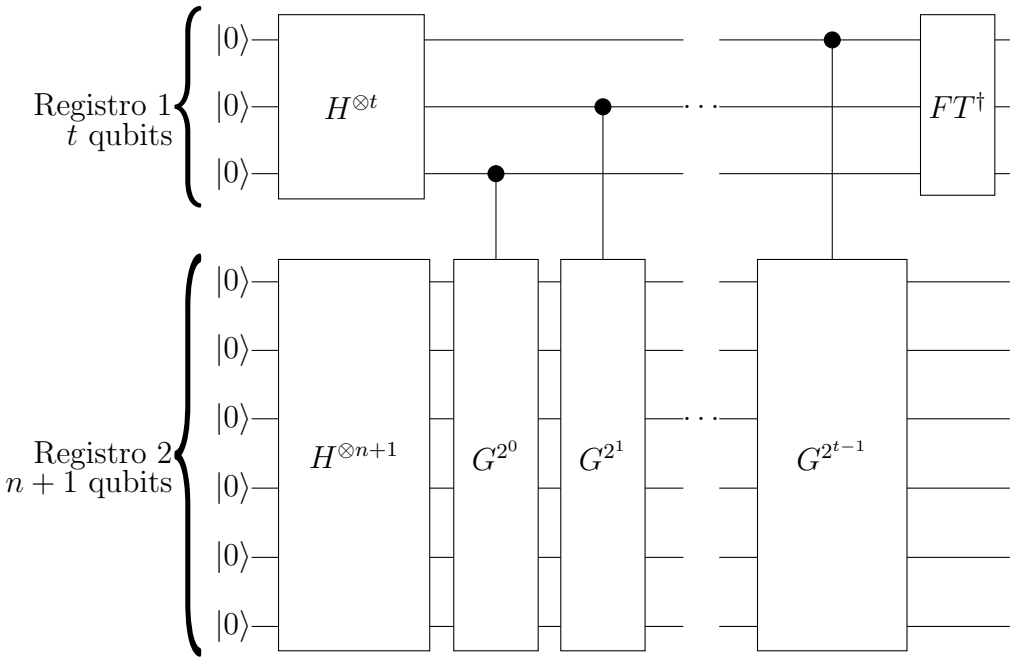

Figura 5.12: Contagem Quântica: Circuito

$$\begin{aligned}\frac{\delta M}{2N} &= \left|sin^2\left(\frac{\theta+\Delta\theta}{2}\right) - \sin^2\left(\frac{\theta}{2}\right)\right| \\ &= sin^2\left(\frac{\theta+\Delta\theta}{2}\right) + \sin^2\left(\frac{\theta}{2}\right)\left|\sin\left(\frac{\theta+\Delta\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)\right| \end{aligned} \tag{5.50}$$

De $|sin(\theta+\Delta\theta)/2 - \sin(\theta/2)| \leq |\Delta\theta|/2$ e de $|sin(\theta+\Delta\theta)/2| < \sin(\theta/2) + |\Delta\theta|/2$, temos,

$$\frac{\delta M}{2N} < \left(2\sin\left(\frac{\theta}{2}\right) + \frac{|\Delta\theta|}{2}\right)\frac{|\Delta\theta|}{2} \tag{5.51}$$

$$|\Delta M| < \left(\sqrt{2MN} + \frac{N}{2^{m+1}}\right)2^{-m} \tag{5.52}$$

A dificuldade em encontrar uma solução para problema de busca quando o número de soluções é desconhecido, recai em saber o número de vezes que a iteração de *Grover* deve ser repetida, usando o algoritmo de estimativa de fase para estimar θ e M com alta precisão, a dificuldade é removida. O erro angular nesse caso será no máximo iqual a $\pi/4(1 + |\Delta\theta|/\theta)$,

de forma que se escolhermos $m = \lceil n/2 \rceil + 1$, o erro angular será no máximo $\pi/4 \times 3/2 = 3\pi/8$, o que corresponde a uma probabilidade de sucesso de pelo menos $\cos^2(3\pi/8) = 1/2 - 1/2\sqrt{2} \approx 0,15$ para o algoritmo de busca. Se a probabilidade de se estimar θ com a precisão acima for $5/6$, a probabilidade total de se obter uma solução para o problema da busca é $5/6 \times \cos^2(3\pi/8) \approx 0,12$, uma probabilidade que pode rapidamente se aproximar de 1 com algumas repetições apenas da rotina combinada "contagem-busca"[29].

Capítulo 6

Aplicação do Operador de Grover no Treinamento do Perceptron

Seja w_i os pesos sinápticos conforme 2.1, e considere encontrar um conjunto de w_i's que gere uma saída desejada do perceptron, numa lista de w_i's. Classicamente teríamos promover uma busca no conjunto de w_i's até encontrar um que satisfaz a saída desejada.

A busca será feita a partir de uma modificação no operador U_f do *Algoritmo de Grover* como descrita na próxima seção.

6.1 Representação Matemática do Problema

Para representar matematicamente o problema, suponha que a busca será realizada sobre uma lista $\{0, 1, \ldots, N-1\}$ onde $N = 2^n$ para algum número natural n e que os w_i's e o bias serão representados por sequências de m *qubits*, onde m define a precisão dos pesos sinápticos e do bias do perceptron. Desta forma n é um múltiplo de m definido pelo número de sinapses mais o bias e a função g é dada por:

$$g(w, E, d) = \begin{cases} 1 & IntegerPart[Pi]/(4ArcSin[1/Sqrt[2^n q]])] se f(w, E_i) = d_i \\ 0 & \text{se } f(w, E_i) \neq d_i \end{cases}$$

A partir desta função, é definido um operador unitário U_g, tal que,

$$U_g : |w\rangle|E_i\rangle|d_i\rangle|-\rangle \rightarrow (-1)^{g(w,E_i,d_i)}|w\rangle|E_i|d_i\rangle|-\rangle \tag{6.2}$$

Ao qual denominaremos como *Oráculo de Grover*, Onde,

$|d_i\rangle$ é a *i-ésima* saída desejada;

$|E_i\rangle$ é o *i-ésimo* Padrões de treinamento;

$|w\rangle$ é o conjunto de pesos (sinapses).

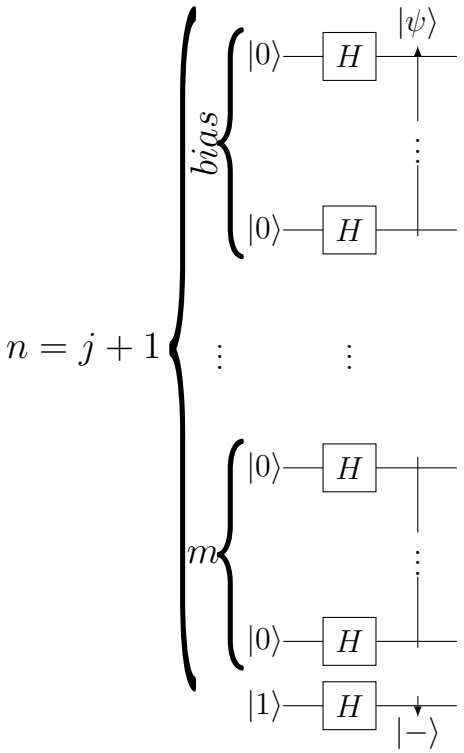

Figura 6.1: Aplicação do operador *Hadamard* aos pesos w_i's e bias onde j é o número de sinapses e m define a precisão dos pesos sinápticos e bias

O objetivo do oráculo é identificar os conjuntos de w_i's que definem os coeficientes das retas que separam as classes como mostrado na Figura 6.2.

A construção do operador U_g pode ser feita a partir do operador U_f conforme a Figura 6.3. A cada linha do operador U_g, está atrelado um conjunto de *qubits*. A linha superior, refere-se ao primeiro registrador, formado por n *qubits* subdivididos em pesos sinápticos e bias. Este

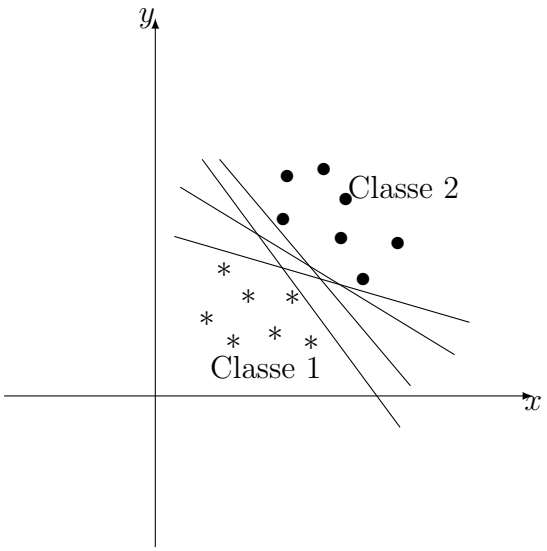

Figura 6.2: Separação das Classes 1 e 2.

registrador armazena os valores de entradas. A segunda linha é o conjunto de *qubits* que armazena os padrões de treinamento e as saídas desejadas e a terceira linha é formada por um único *qubit* auxiliar, é o *qubit-oráculo*. Se a entrada for um estado projetado, a porta Z controlada pelo segundo registrador inverte a amplitude do estado, caso o valor do segundo registrador seja igual a $|0\rangle$ na saída inferior do operador U_f. É usada uma porta *Toffoli* generalizada para criar a porta Z controlada com o *qubit* alvo no estado $|-\rangle$.

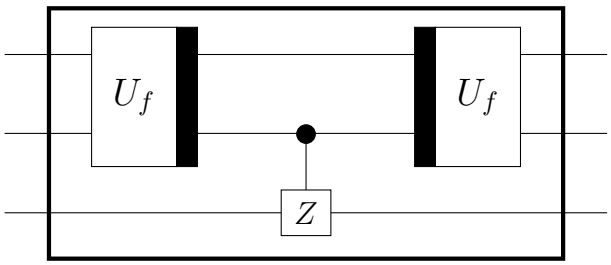

Figura 6.3: Esquema do circuito U_g a partir de U_f.

Os registradores serão organizados de forma que o primeiro registrador está relacionado aos elementos da lista onde será feita a busca e está subdividido em $j+1$ agrupamentos de tamanho m, onde m é a precisão desejada, j o número de sinapses e o "$+1$"referisse ao bias, Figura 6.1.

Os *qubits* cujos valores de saída são iguais aos valores de entrada serão omitidos para um melhor entendimento.

Seja y_k a função de ativação do perceptron como dado em 2.2 e 2.3,

$$y_k = \varphi\left(\sum_{i=i}^{n} x_i w_{ki} + b_k\right) \tag{6.3}$$

Para implementar o operador que calcula y_k para varias entradas em superposição, serão usadas apenas operadores reversíveis. O operador pode ser implementado utilizando algoritmos aritméticos descritos por *Kowada* em [31]. Um possível operador que calcula U_f pode ser implementado utilizando os operadores *somador* Figura 5.3, *multiplicador* Figura 5.6, *comparador* Figura 5.7, *OR* Figura 4.8 e *Z* 4.22. Para gerar $\sum_{i=i}^{n} x_i w_{ki} + b_k$, são usados os operadores *somador* e *multiplicador* e a função de limiar será verificada a partir do operador *comparador* 5.7, comparando o resultado de $\sum_{i=i}^{n} x_i w_{ki} + b_k$ e o estado $|0\rangle$. Será aplicada então uma porta *OR* tendo como entradas, a saída do *comparador* e a saída desejada $|d_i\rangle$ do padrão de treinamento que, caso sejam iguais, o conjunto de w_i's terá sua fase invertida a partir da aplicação da porta Z que é controlada pela saída da porta *OR*, nada acontecendo caso contrário.

A Figura 6.4 ilustra o esquema de portas que produz a inversão de fase utilizando os padrões de treinamento $|E_i\rangle$ e a saída desejada $|d_i\rangle$.

O circuito da Figura 6.4 apresenta a aplicação do oráculo U_g a apenas um padrão de treinamento. É necessário expandir o circuito de forma que todo o conjunto de dados de treinamento da rede seja utilizado. Para isso, usaremos a seguinte abordagem. Serão apresentados todos os padrões de treinamento e saídas desejadas na identificação, pelo oráculo, dos elementos da lista que satisfazem a equação 6.3 afim de aumentar a amplitude destes estados pelo *operador de Grover*.

6.1.1 Utilização de todos os Padrões de Treinamento na definição do Oráculo

O operador U_g conforme esquematizado na Figura 6.4 foi aplicado a um único padrão de treinamento, isto implica em um número grande de soluções das retas que satisfazem a equação 6.3 conforme Figura 6.5. Afim de evitar que M seja maior do que $N/2$, será feita uma modificação no operador na equação 6.2 de U_g para que esta receba todos os padrões

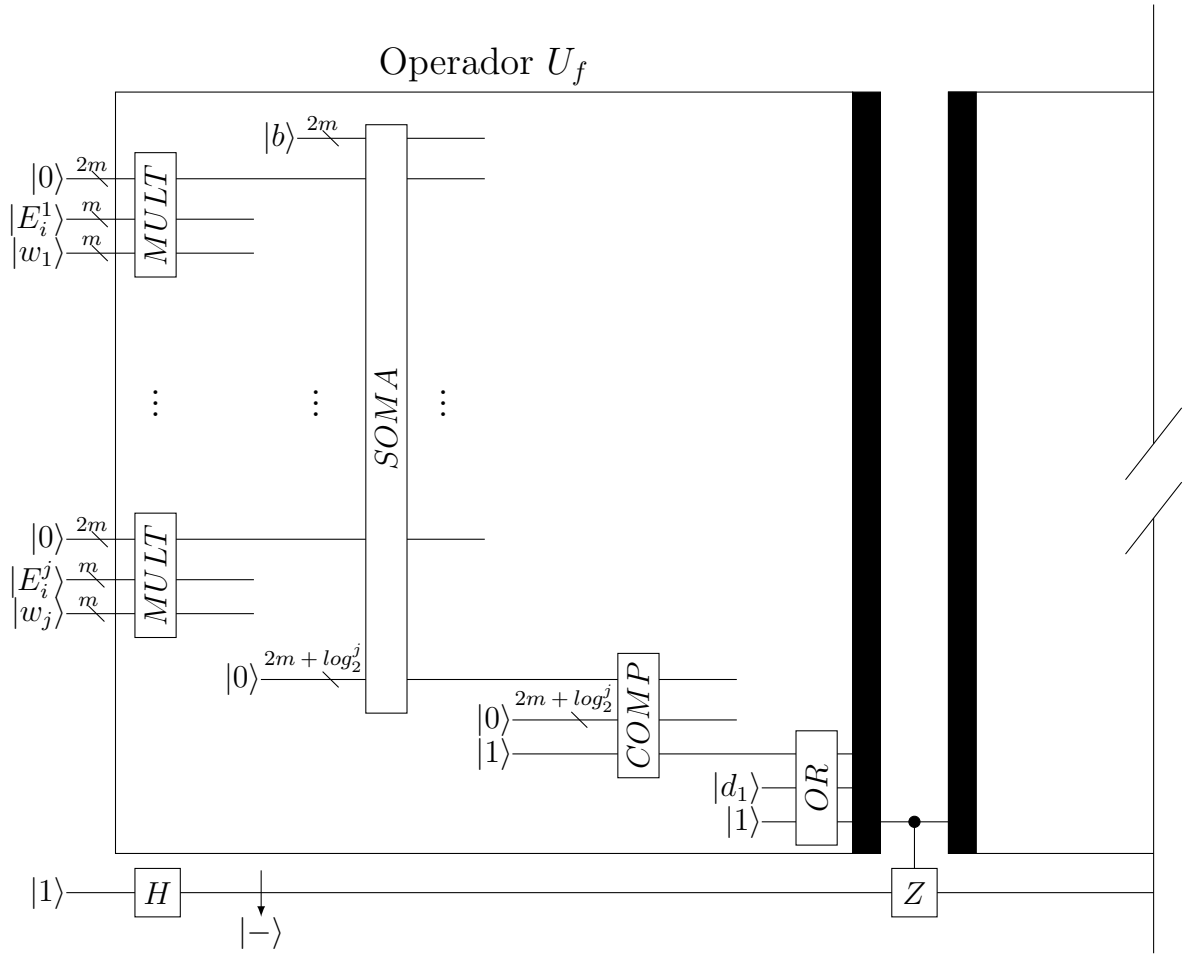

Figura 6.4: Circuito que implementa o oráculo U_f a partir dos módulos *somador*, *multiplicador*, *comparador*, *OR* e *Z*.

de treinamento e suas respectivas saídas desejadas.

Para que o oráculo comporte-se da forma apresentada na Figura 6.2, o operador U_g deve receber todos os padrões de treinamento e saídas desejadas. Segue então que:

$$U_g : |w\rangle|E_1\rangle \dots |E_j\rangle|d_1\rangle \dots |d_j\rangle|-\rangle \rightarrow (-1)^{g(w,E_1,\dots,E_j,d_1,\dots,d_j)}|w\rangle|E_1\rangle \dots |E_j\rangle|d_1\rangle \dots |d_j\rangle|-\rangle \tag{6.4}$$

Este oráculo, pode ser visto como a aplicação do operador U_f em cascata e uma porta Z controlada criada a partir de uma porta *Toffoli* generalizada 6.6. Identificados os elementos e invertidas suas fases, segue a aplicação da inversão sobre a média $2|\psi\rangle\langle\psi| - I$. A aplicação do oráculo seguida da inversão sobre a média é definida como *operador de Grover G*.

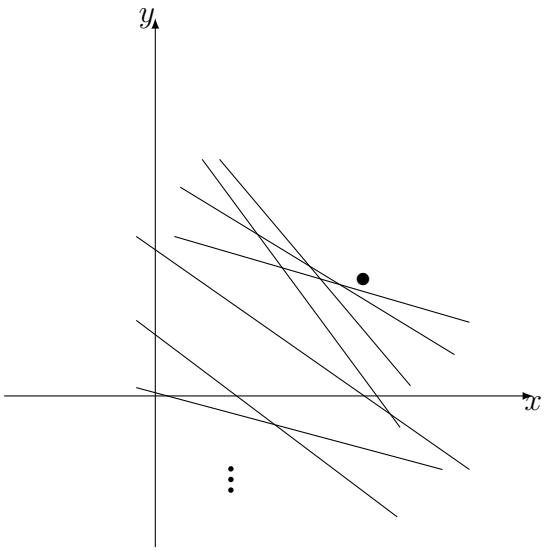

Figura 6.5: Aplicação do oráculo a um único padrão de treinamento.

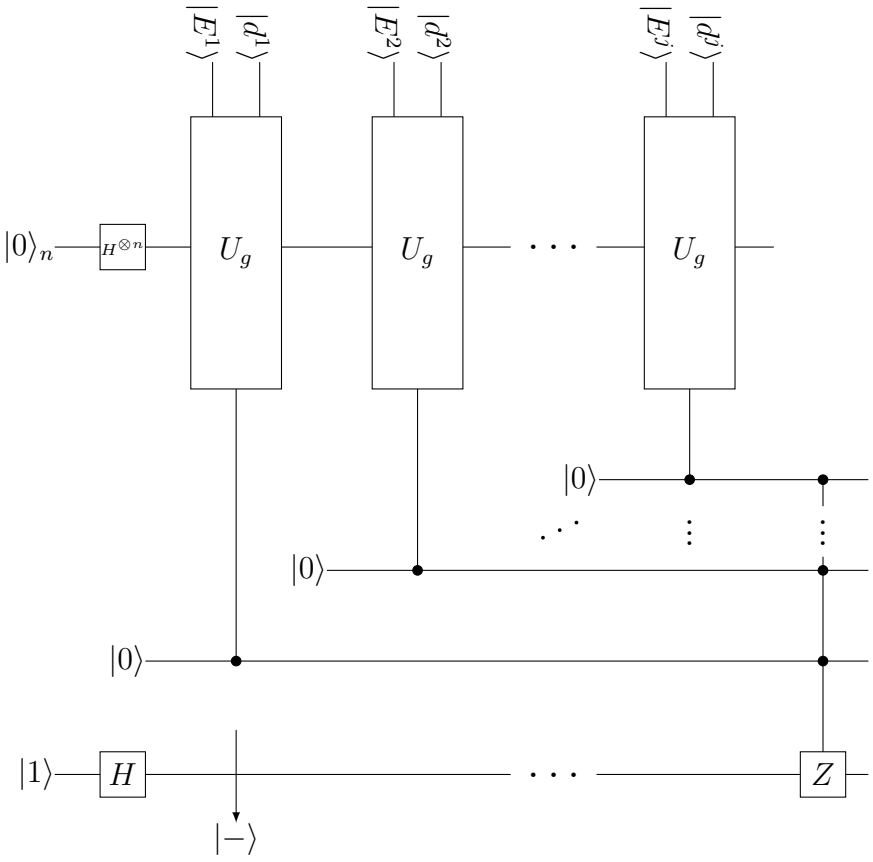

Figura 6.6: Circuito do oráculo construído a partir da equação 6.4.

Até o momento definimos o operador G que identifica e produz a inversão sobre a média, mas não definimos quantas vezes aplicaremos o operador. Note que o número de aplicações do operador de Grover depende de M, mas não da identificação das soluções, de forma

que basta conhecer M para aplicarmos a equação 5.45 e consequentemente o algoritmo quântico de busca. Para definir o número de aplicações do operador de Grover, utilizaremos o algoritmo de contagem quântica descutido na seção 5.5.6.

Capítulo 7

Simulações do Algoritmo de Grover

Por que simular computadores quânticos em máquinas convencionais? De forma geral, vale a pena simular um sistema quando a simulação é imensamente menos custosa e ainda assim permite estudarmos os aspectos de interesse do sistema real. Simular o comportamento dos computadores quânticos permitirá entender as dificuldades a serem enfrentadas no desenvolvimento de novos algoritmos destinados a computadores quânticos. É interessante observar que antes do primeiro computador quântico ser construído, teremos diversos algoritmos específicos para computadores quânticos rodando em simulação. Além disso, através de simulações, teremos uma ideia muito aproximada do que poderemos esperar ao construir computadores quânticos.

Um computador clássico tem a capacidade de simular um computador quântico, aparentemente não há nenhum problema computável que não seja possível resolver num computador clássico, desde que recursos infinitos, tais como memória ou tempo de processamento fossem possível como teorizado por *Turing* [39]. No entanto, não dispomos de recursos infinitos, afim de resolver problemas que demandam estes recursos tais como o problema da fatoração de grandes números que podem necessitar de milhões de anos de processamento, mesmo em super computadores.

A computação quântica torna exequível muitos destes problemas, mas criar um algoritmo quântico que seja mais eficiente que o algoritmo clássico correspondente não é uma tarefa trivial. Apesar de ser possível tirar partido dos fenômenos quânticos, para computar 2^n estados em simultâneo, a extração da informação é limitada a n bits de informação devido ao problema da medição colapsar o estado do sistema. Desta forma, é necessário criar um

algoritmo que execute a computação tirando partido dos fenômenos quântico e ao final, o estado do sistema permita extrair a informação procurada a partir da medição.

Existem diversos pacotes e aplicativos para simulação de computadores quânticos tais como QCL (Quantum Computer Language)[30], Quantum (Mathematica) [13], QuTip (Python) [28], entre outros.

7.1 Simulando o Algoritmo de Grover

Os testes iniciais da simulação do Algoritmo de Grover, utilizou-se o modulo *Quantum* [13] desenvolvido pela *Universidade Tecnológico de Monterrey (México)*. A ferramenta mostrou-se bastante simples, o que permitiu uma rápido aprendizado de suas ferramentas e comandos. Em seus exemplos, encontra-se o *Quantum Circuit Implementing Grover's Search Algorithm*. Esta permitiu entender os processos algébricos atrelados ao algoritmos como também visualizar o circuito lógico quântico no algoritmo de busca de Grover.

Aqui, mostraremos o exemplo da simulação para 4 *qubits*, o que dá um espaço de busca de $N = 2^4$, e a busca será feita ao elemento $|2\rangle$ (Figura 7.3).

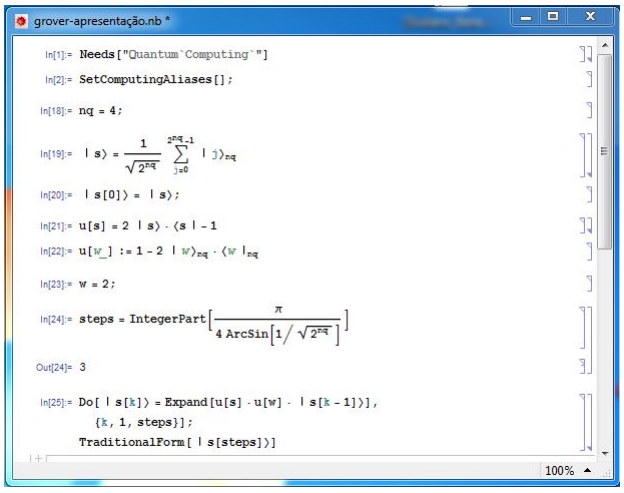

Figura 7.1: Implementação do *Algorítimo de Grover* no *Software Mathematica*

7.1.1 Passos da Implementação

1. Definição do número de qubits do espaço de busca: $nq = 4$;

2. Criação dos estados superpostos: $|s\rangle$;

3. Atribuição ao estado $|s[0]\rangle$ o estado $|s\rangle$;

4. Criação da matriz de inversão sobre a média: $u[s] = 2|s\rangle \cdot \langle| - 1$;

5. Criação da matriz de inversão de fase: $u[w\_]: = 1 - 2|w\rangle_{nq} \cdot \langle w|_{nq}$;

6. Definição do elemento a ser procurado: $w = 2$;

7. Cálculo do número de aplicações do operador de Grover: step = Integerpart$[\frac{\pi}{4 \arcsin[1/\sqrt{2^{nq}}]}]$;

8. Loop de aplicação do operador de Grover: Do$[|s[k]\rangle$ = Expand$[u[s] \cdot u[w] \cdot |s[k - 1]\rangle], \{k, 1, steps\}]$

A utilização do *Mathematica*, permite ao usuário acompanhar passo a passo a evolução do algoritmo, isto foi fundamental para o entendimento da das operações entre ass matrizes e vetores presentes na operação.

A saída do algorítimo pode ser formatada de modo a apresentar-se a amplitude de probabilidade, o estado a ser medido e o produto tensorial dos estados da base conforme a Tabela abaixo.

Probability	Measurement	State
0.00257874	$\begin{pmatrix} 0_1 & 0_2 & 0_3 & 0_4 \end{pmatrix}$	$\|0\rangle \otimes \|0\rangle \otimes \|0\rangle \otimes (-1.\|0\rangle)$
0.00257874	$\begin{pmatrix} 0_1 & 0_2 & 0_3 & 1_4 \end{pmatrix}$	$\|0\rangle \otimes \|0\rangle \otimes \|0\rangle \otimes (-1.\|1\rangle)$
0.961319	$\begin{pmatrix} 0_1 & 0_2 & 1_3 & 0_4 \end{pmatrix}$	$\|0\rangle \otimes \|0\rangle \otimes \|1\rangle \otimes \|0\rangle$
0.00257874	$\begin{pmatrix} 0_1 & 0_2 & 1_3 & 1_4 \end{pmatrix}$	$\|0\rangle \otimes \|0\rangle \otimes \|1\rangle \otimes (-1.\|1\rangle)$
0.00257874	$\begin{pmatrix} 0_1 & 1_2 & 0_3 & 0_4 \end{pmatrix}$	$\|0\rangle \otimes \|1\rangle \otimes \|0\rangle \otimes (-1.\|0\rangle)$
0.00257874	$\begin{pmatrix} 0_1 & 1_2 & 0_3 & 1_4 \end{pmatrix}$	$\|0\rangle \otimes \|1\rangle \otimes \|0\rangle \otimes (-1.\|1\rangle)$
0.00257874	$\begin{pmatrix} 0_1 & 1_2 & 1_3 & 0_4 \end{pmatrix}$	$\|0\rangle \otimes \|1\rangle \otimes \|1\rangle \otimes (-1.\|0\rangle)$
0.00257874	$\begin{pmatrix} 0_1 & 1_2 & 1_3 & 1_4 \end{pmatrix}$	$\|0\rangle \otimes \|1\rangle \otimes \|1\rangle \otimes (-1.\|1\rangle)$
0.00257874	$\begin{pmatrix} 1_1 & 0_2 & 0_3 & 0_4 \end{pmatrix}$	$\|1\rangle \otimes \|0\rangle \otimes \|0\rangle \otimes (-1.\|0\rangle)$
0.00257874	$\begin{pmatrix} 1_1 & 0_2 & 0_3 & 1_4 \end{pmatrix}$	$\|1\rangle \otimes \|0\rangle \otimes \|0\rangle \otimes (-1.\|1\rangle)$
0.00257874	$\begin{pmatrix} 1_1 & 0_2 & 1_3 & 0_4 \end{pmatrix}$	$\|1\rangle \otimes \|0\rangle \otimes \|1\rangle \otimes (-1.\|0\rangle)$
0.00257874	$\begin{pmatrix} 1_1 & 0_2 & 1_3 & 1_4 \end{pmatrix}$	$\|1\rangle \otimes \|0\rangle \otimes \|1\rangle \otimes (-1.\|1\rangle)$
0.00257874	$\begin{pmatrix} 1_1 & 1_2 & 0_3 & 0_4 \end{pmatrix}$	$\|1\rangle \otimes \|1\rangle \otimes \|0\rangle \otimes (-1.\|0\rangle)$
0.00257874	$\begin{pmatrix} 1_1 & 1_2 & 0_3 & 1_4 \end{pmatrix}$	$\|1\rangle \otimes \|1\rangle \otimes \|0\rangle \otimes (-1.\|1\rangle)$
0.00257874	$\begin{pmatrix} 1_1 & 1_2 & 1_3 & 0_4 \end{pmatrix}$	$\|1\rangle \otimes \|1\rangle \otimes \|1\rangle \otimes (-1.\|0\rangle)$
0.00257874	$\begin{pmatrix} 1_1 & 1_2 & 1_3 & 1_4 \end{pmatrix}$	$\|1\rangle \otimes \|1\rangle \otimes \|1\rangle \otimes (-1.\|1\rangle)$
Probability	Measurement	State

7.2 Simulação do Algoritmo de Grover Usando Python

A migração para o Python através do uso do pacote *Qutip* [28], permitiu expandir a simulação até o número de 16 qubits, este numero de qubits é suficiente para um perceptron simular uma porta lógica simples. A porta AND (Figura 7.2 foi escolhida então para a simulação. Todavia, a matriz definida pelo operado U_G foi gerada classicamente devido ao número de qubits exceder o máximo permitido no uso deste pacote, uma vês que os operadores que formam o oráculo, tem inúmeros *qubits* auxiliares.

Os qubits do estado de entrada foi subdividido em 3 subgrupos com 3 qubits cada, de forma que os três primeiros qubits representam o bias, os três qubits intermediários, representam o peso sináptico w_1 e os três últimos qubits representam o peso sináptico de w_2.

A simulação seguiu os passos do algoritmo *BBHT*, partindo do pressuposto que não é

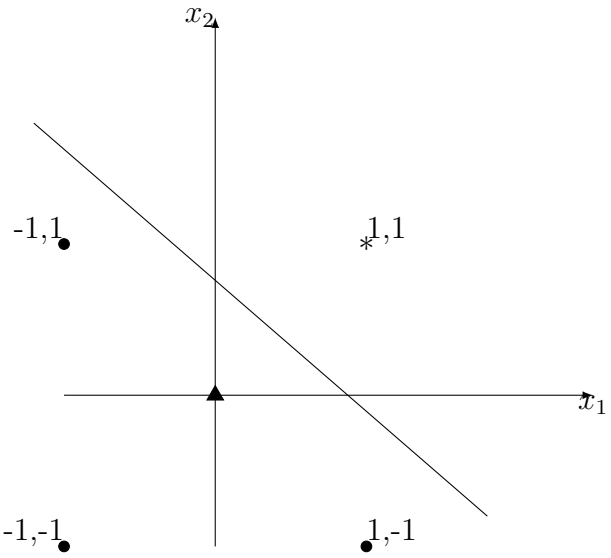

Figura 7.2: Solução para o problema da porta lógica AND onde $Verdadeiro = 1$ e $Falso = -1$.

conhecido o número de soluções, onde no seu loop, o número de aplicações do algoritmo de Grover aumenta em função do crescimento da variável j (Algoritmo 2).

As operações algébricas produzidas pelo simulação do algorítimo de Grover, seguem as operações descritas na seção 5.5.4, porem os vetores e matrizes gerados são da ordem de $2^9 \times 1$ e $2^9 \times 2^9$ respectivamente.

Após a aplicação do operador *Hadamard* aos qubits de entrada no algoritmo, uma leitura do gráfico da Figura 7.3. Uma rápida leitura, mostra que todos os estados estão com a mesma amplitude, o que era de se esperar já que a porta *Hadamard* põe o *qubit* num estado intermediário entre $|0\rangle$ e $|1\rangle$.

É aplicado a primeira iteração do BBHT e as amplitudes dos estados pode ser vista no gráfico da Figura 7.4. Aqui, já pode ser visto um aumento da amplitude dos estados que produzem os coeficientes das retas que separam as classes da porta AND. A probabilidade de uma leitura encontrar um dos estados procurados é de aproximadamente $60,6\%$.

Seguindo o algorítimo *BBHT*, é feita uma leitura dos *qubits*, os estados colapsam e os pesos da saída são testados utilizando os padrões de teste da RNA. Caso o estado colapsado seja um dos estados procurados, o *BBHT* sai do loop e o algorítimo é encerrado. Todavia, caso o estado colapsado não satisfaça os dados de teste, o *BBHT* entra no loop seguinte, desta vez, aplicando duas vezes o *operador de Grover*.

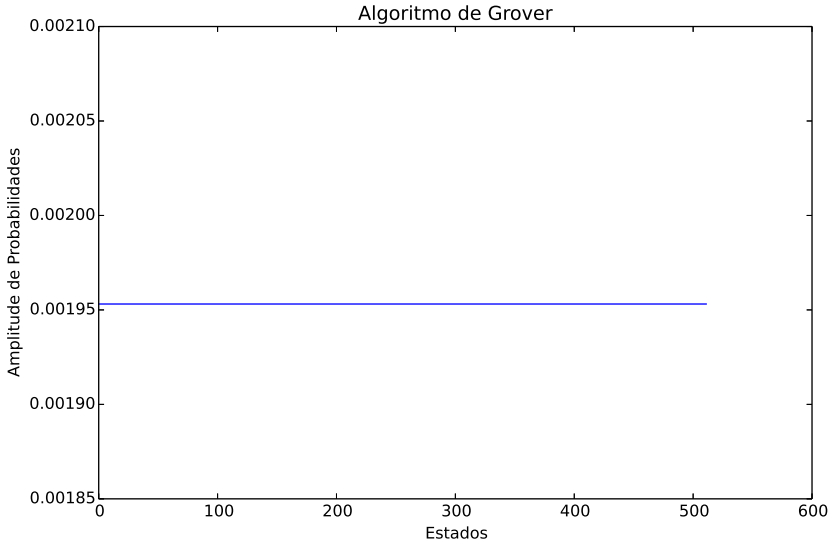

Figura 7.3: Implementação do *Algorítimo de Grover*: Amplitudes dos estados após a aplicação da porta $H^{\otimes n}$.

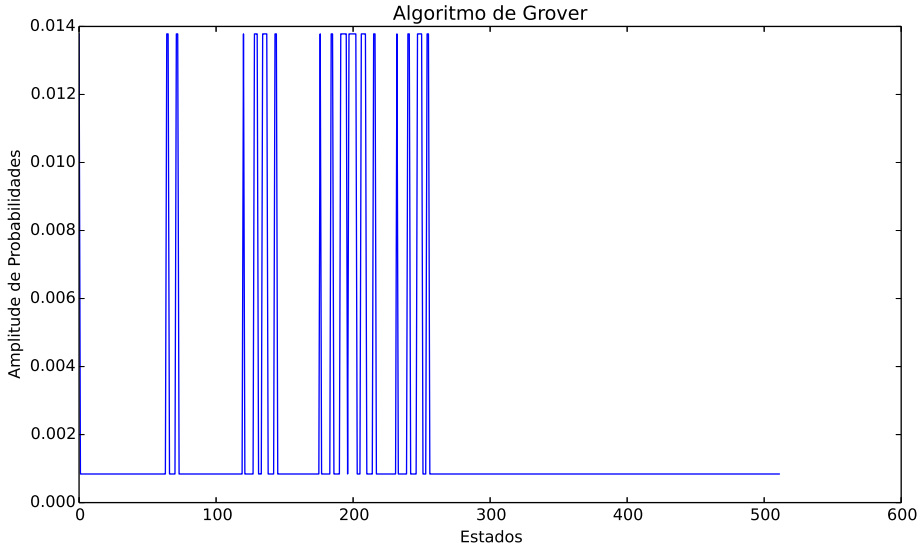

Figura 7.4: Implementação do *Algorítimo de Grover*: Amplitudes dos estados após uma aplicação do *operador de Grover*.

Após a segunda aplicação, conforme o loop do *BBHT*, a amplitude dos estados da base são alterados, e uma análise do gráfico da Figura 7.5 mostra um aumento significativo dos estados que satisfazem a busca. A probabilidade de uma leitura encontrar um dos estados procurados após a segunda aplicação do *operador de Grover* é de aproximadamente $99,3\%$. É feita uma leitura dos *qubits* e novamente, os pesos sinápticos são testados na RNA, o que

deve satisfazer os padrões de teste. O loop do algoritmo *BBHT* é então encerrado.

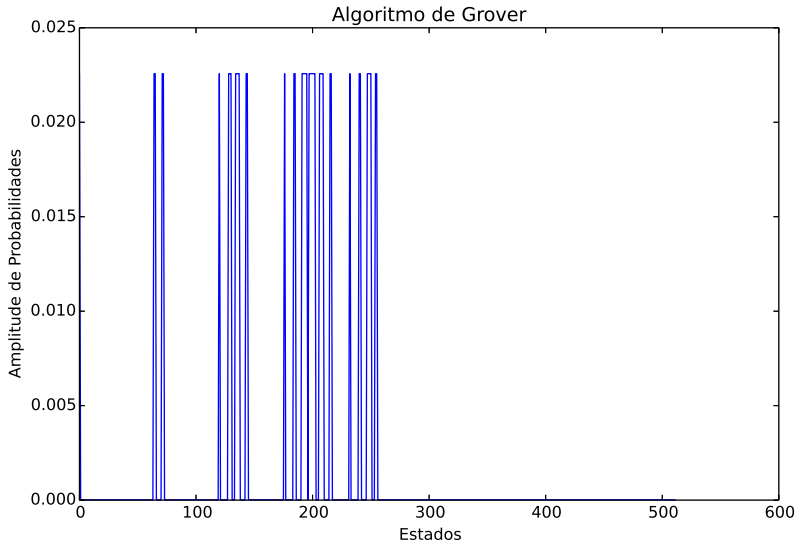

Figura 7.5: Implementação do *Algorítimo de Grover*: Amplitudes dos estados após duas aplicação do *operador de Grover*.

O uso do *BBHT*, pode ser substituído pelo algoritmo de Contagem Quântica, este retornaria o número de soluções para a busca, o qual seria aplicado na equação 5.45 definindo o número de aplicações do operador de Grover.

Capítulo 8

Conclusões e trabalhos Futuros

Neste capítulo são apresentadas as considerações finais desta dissertação e apresenta algumas sugestões de trabalhos futuros que visam contribuir com a solução apresentada neste documento.

8.1 Conclusão

Este trabalho de dissertação visou propor uma abordagem quântica para o treinamento de uma RNA com pesos, em particular o perceptron clássico. Como solução foi proposto o uso do *Algoritmo de Grover* para promover a busca dos pesos sinápticos que compõem os elementos da lista. A abordagem proposta propõe a criação de um operador unitário, que trabalha como o oráculo de Grover na identificação e inversão de fase dos estados que compões os coeficientes da reta que separam duas classes. Como o número de estados da base que satisfazem a condição da busca não era conhecida, o *Algoritmo BBHT* foi aplicado, porem, uma outra abordagem poderia ter sido aplicada, a partir do Algorítimo de Contagem Quântica.

Foram feitas simulações do *Algoritmo de Grover* utilizando o *Software* Mathematica através do pacote Quantum, que tem uma rápida curva de aprendizado, foram feitas as primeiras simulações, porem este mostrou-se inadequado quando o número de *qubits* era maior do que 6, devido ao aumento exponencial das matrizes envolvidas. Como alternativa, migrou-se para a plataforma Python, utilizando o pacote *Qutip*, que apesar de não ser

tão intuitivo quanto o primeiro, tem uma curva de aprendizado melhor do que os pacotes baseados em *C* ou *C++* como o *QCL*. Apesar de não apresentar travamentos, o pacote *Qutip*, limitou-se a processar as matrizes até o número de 16 qubits, o que gerou matrizes de $2^{16} \times 2^{16}$, permitindo simular a porta AND com 9 *qubits*, o mínimo de qubits que se mostrou eficiente para a simulação.

Decorrente da limitação imposta pelo software, a matriz que representa o operador proposto foi gerado de forma clássica, mantendo os demais elementos do *Algoritmo de Grover* e *BBHT*.

Uma consequência da metodologia adotada é que, diante da afirmação de *Rosenblatt* no *Teorema de Convergência do Perceptron* [35], a abordagem quântica permite definir um limite superior para encontrar um conjunto de pesos sinápticos. Este limite é definido pelo *Algoritmo de Grover* que tem tempo de execução da ordem $O(\sqrt{N/M}$ [29], no entanto, a abordagem proposta tem o tempo de processamento é dominado pela multiplicação.

8.2 Trabalhos Futuros

Como consequência da limitação imposta pelo software, a busca por ferramentas que proporcionem maior poder computacional, tanto de software como hardware, visando aumentar a a eficiência da solução, ampliando o número de qubits de forma a aumentar tanto a precisão como o numero de sinapses e neurônios da RNA. Com o desenvolvimentos de novos algoritmos quânticos para manipulação de matrizes e vetores, a linha de pesquisa de aprendizado de maquina será beneficiada com os ganhos exponenciais promovido pela computação quântica. A manipulação de grandes dados (—Big Data) será no futuro um grande incentivador no desenvolvimento dos primeiros processadores quânticos.

Por fim, sugere-se a extensão da pesquisa na área de RNA's para abranger também a pesquisa para aplicação de algoritmos quânticos a treinamento de RNA's não supervisionadas.

Referências Bibliográficas

[1] MV Altaisky. Quantum neural network. *arXiv preprint quant-ph/0107012*, 2001.

[2] G Benenti, G. Casati, and G. Strini. *Principles of quantum computation and information*, volume 1. World scientific, 2004.

[3] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22(5):563–591, 1980.

[4] Charles H Bennett. Notes on landauer's principle, reversible computation, and maxwell's demon. *Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics*, 34(3):501–510, 2003.

[5] Charles H Bennett. Notes on landauer's principle, reversible computation, and maxwell's demon. *Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics*, 34(3):501–510, 2003.

[6] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *arXiv preprint quant-ph/9605034*, 1996.

[7] A. P. Braga, A. P. L. F. Carvalho, and T. B. Ludermir. *Redes neurais artificiais: teoria e aplicações*. LTC - Livros Técnicos e Científicos Editora, Rio de Janeiro-RJ, 2007.

[8] Goong Chen and Shunhua Sun. Generalization of grover's algorithm to multiobject search in quantum computing, part ii: General unitary transformations. *Mathematics of Quantum Computation. Computational Mathematics*, pages 161–168, 2002.

[9] Jacob Daghlian. *Lógica e álgebra de Boole*. Atlas, 1986.

[10] Franklin de Lima Marquezino. *A transformada de fourier quântica aproximada e sua simulação*. PhD thesis, Master's thesis, Laboratório Nacional de Computação Científica, 2006.

[11] David Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[12] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.

[13] José Luis Gómez-Muñoz and Francisco Delgado. Quantun: A free mathematica add-on for dirac bra-ket notation, quantum algebra, quantum computing and the qhd approximation to the heisenberg equations of motion, November 2012.

[14] Lov K Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters*, 79(2):325, 1997.

[15] Sanjay Gupta and RKP Zia. Quantum neural networks. *Journal of Computer and System Sciences*, 63(3):355–383, 2001.

[16] S. Haykin. *Redes neurais: Princípios e prática*. Bookman, Porto Alegre-RJ, 2008.

[17] Bernardo Lula Júnior and Aércio Ferreira de Lima. Circuitos quânticos. *ppginf. ucpel. tche. br/weciq/CD/.../mini-cursobernardo-lula. pdf¿. Acessado em 20/12/1013*, 10, 2006.

[18] Stanley A Klein. Is quantum mechanics relevant to understanding consciousness. *Psyche*, 2(3), 1995.

[19] Luis Kowada, Celina Figueiredo, Renato Portugal, and Carlile Lavor. Aplicação do algoritmo de grover para problemas np-completos. In *2? Workshop-Escola de Computação e Informação Quântica*, 2007.

[20] David C Lay, Ricardo Camelier, and Valéria de Magalhães Iório. *Álgebra linear e suas aplicações*. LTC, 1997.

[21] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

[22] Chris Lomont. A quantum fourier transform algorithm. *arXiv preprint quant-ph/0404060*, 2004.

[23] V. Y. Matsunaga. *curso de redes neurais utilizando matlab*. UEPB, Belém do Para-PA, 2012.

[24] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[25] N David Mermin. *Quantum computer science*, volume 38. Cambridge University Press Cambridge, 2007.

[26] Valéria S Motta, Luiz Mariano Carvalho, and Nelson Maculan. Esfera de bloch: algumas propriedades.

[27] Amanda Leonel Nascimento, Luis Antônio Brasil Kowada, and Wilson Rosa de Oliveira. Uma unidade lógica e aritmética reversível. *A A*, 1(F1):F2, 2006.

[28] PD Nation and JR Johansson. Qutip: Quantum toolbox in python, October 2012.

[29] Michael A Nielsen, Isaac L Chuang, and Ivan S Oliveira. *Computação quântica e informação quântica*. Bookman, 2005.

[30] Bernhard Omer. *Quantum programming in QCL*. 2000.

[31] R Portugal and CMH Figueiredo. Reversible karatsuba's algorithm. *Journal of Universal Computer Science*, 12(5):499–511, 2006.

[32] Renato Portugal, Carlile C Lavor, Luiz M Carvalho, and Nelson Maculan. *Uma introdução à computação quântica*. Sociedade Brasileira de Matemática Aplicada e Computacional (SBMAC), 2012.

[33] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big feature and big data classification. *arXiv preprint arXiv:1307.0471*, 2013.

[34] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[35] Frank Rosenblatt. *Principies of Neurodynamics: Perceptrons and the theory of brain mechanisms . Spartan Books*. Spartan Books, New York-USA, 1962.

[36] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. IEEE, 1994.

[37] Spencer L Smith, Ikuko T Smith, Tiago Branco, and Michael Häusser. Dendritic spikes enhance stimulus selectivity in cortical neurons in vivo. *Nature*, 2013.

[38] R. Solovay and V. Strassen. A fast monte-carlo test for primality. *SIAM Journal on Computing*, 6(1):84–85, 1977.

[39] Alan M Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, 42(2):230–265, 1936.

[40] A. L. Vignatti. Uma introducão 'a computacão quântica. preprint (2004), available at `http://www.ic.unicamp.br/ bit/arquivos/tg.pdf`.