Vilson Vieira da Silva Junior

Estudo e Simulação de Algoritmos Baseados em Caminhos Aleatórios Quânticos e sua Aplicação em Problemas de Grafos

Joinville

2007

Vilson Vieira da Silva Junior

Estudo e Simulação de Algoritmos Baseados em Caminhos Aleatórios Quânticos e sua Aplicação em Problemas de Grafos

Relatório Final de Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Graduação em Ciência da Computação, da Universidade do Estado de Santa Catarina (UDESC), como requisito parcial a disciplina de Trabalho de Conclusão de Curso.

Orientador: Prof. Alexandre Gonçalves Silva

Co-orientador: Prof. Fernando Deeke Sasse

Joinville

2007

Vilson Vieira da Silva Junior

Estudo e Simulação de Algoritmos Baseados em Caminhos Aleatórios Quânticos e sua Aplicação em Problemas de Grafos

Relatório Final de Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Ciência da Computação da UDESC, como requisito para a obtenção parcial do grau de BACHAREL em Ciência da Computação.

Aprovado em 7 de Dezembro de 2007

BANCA EXAMINADORA

Prof. Alexandre Gonçalves Silva

Prof. Fernando Deeke Sasse

Prof. Rafael Stubs Parpinelli

Prof. Claudiomir Selner

A todos aqueles que valorizam acima de tudo o conhecimento.

Resumo

Este trabalho tem como propósito estudar Caminhos Aleatórios Quânticos aplicados a problemas simples de grafos. Utilizando uma abordagem alternativa à convencional, usamos um simulador quântico, QGAME, e sua linguagem de programação quântica para executar um algoritmo quântico baseado na estratégia de Caminhos Aleatórios. Apresentamos as extensões implementadas em QGAME que proporcionaram a execução destes algoritmos. Aplicamos a versão estendida de QGAME na simulação da forma generalizada do grafo cíclico de N vértices, demonstrando o uso do simulador para este tipo de grafo.

Palavras-chaves: Computação Quântica, Caminhos Aleatórios Quânticos, Teoria da Computação, Teoria de Grafos, QGAME, Linguagens de Programação Quântica

Abstract

The purpose of this work is to study Quantum Random Walks applied to simple graph problems. Using an alternative rather than conventional approach, we use the QGAME quantum simulator and its quantum programming language to execute a quantum algorithm based on the quantum random walk strategy. In order to perform these algorithms we have implemented extensions to QGAME. To demonstrate the use of the simulator at graphs, we have applied the simulator to the general form of cycle graphs with N vertices.

Keywords: Quantum Computing, Quantum Random Walks, Theory of Computation, Graph Theory, QGAME, Quantum Programming Languages

Agradecimentos

Agradeço especialmente a meus pais, por todo seu amor.

À Patrícia, pela cumplicidade e incentivo.

À Fausto, pelas discussões sempre motivadoras.

À Pedro, que me mostrou a *caixa de Pandora* da Ciência da Computação: *Lisp*.

Ao meu co-orientador e amigo Fernando Deeke Sasse, pela paciência e dedicação no desenvolvimento deste trabalho, e por ser um exemplo a ser seguido.

À todos os meus bons e verdadeiros amigos, vocês sabem quem são e quanto significam para mim.

Aos professores do Departamento de Ciência da Computação, Matemática e Física, pelos ensinamentos e amizade.

À Udesc, pelo auxílio que me foi oferecido nestes anos de graduação.

"O que não posso criar, eu não posso compreender"

– Richard P. Feynman

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

1 Introdução

A Ciência da Computação teve seus limites e fundamentos teóricos traçados antes mesmo da construção de um computador propriamente dito. Em 1936, Alan Turing desenvolveu o conceito abstrato de máquina programável [Turing 1936] que mais tarde iria culminar com a tese de Church-Turing, que estabelece que "qualquer processo algorítmico pode ser simulado eficientemente usando uma máquina de Turing" [Sipser 1997].

Esta tese cria implicitamente uma barreira à computação clássica, uma vez que nenhuma máquina poderia ter poder maior de processamento do que a máquina universal de Turing. A partir de então, várias tentativas de se criar máquinas mais poderosas foram feitas, como a computação analógica e a computação aleatória [Nielsen e Chuang 2000].

Em 1982, Richard Feynman [Feynman 1982], motivado pela dificuldade de simulação de sistemas físicos quânticos por computadores clássicos, sugeriu a possibilidade do uso de computadores quânticos. Tais máquinas usariam sistemas quânticos para realizar computações capazes de simular processos físicos quânticos. Em 1985, David Deutsch [Deutsch 1985] provou a universalidade do computador quântico, ou seja, um computador quântico é capaz de simular eficientemente qualquer sistema físico – clássico ou quântico não-relativista – [Nielsen e Chuang 2000]. O *insight* de Feynman era que como sistemas quânticos eram exponencialmente lentos quando simulados classicamente, talvez algoritmos implementados numa máquina quântica poderiam ser exponencialmente mais rápidos do que computadores clássicos na resolução de certos problemas.

A partir de então, foi iniciada a busca por algoritmos para computadores quânticos que resolvessem problemas com eficiência maior que aqueles para computadores clássicos. Esta busca ganhou um impulso notável em 1994, quando Peter Shor [Shor 1994] utilizou da Transformada de Fourier Quântica de Coppersmith [Coppersmith 1994] para formular um algoritmo quântico capaz de fatorar um número inteiro em tempo polinomial e outro para resolver o problema do logaritmo discreto. Um sucesso mais modesto, porém também importante, ocorreu em 1995, quando Lov Grover [Grover 1996] propôs um algoritmo quântico que poderia resolver o problema da busca em uma lista desordenada de dados em um tempo polinomialmente mais rápido que sua contrapartida clássica. A

busca por algoritmos quânticos mais eficientes que os clássicos se revelou mais difícil do que o imaginado. No entanto, novos algoritmos quânticos eficientes têm sido recentemente encontrados em problemas de grafos e problemas que envolvem Transformada de Fourier sobre grupos.

Este trabalho se desenvolve na área que corresponde à generalização quântica da teoria de algoritmos de caminhos aleatórios sobre grafos, utilizando linguagens de programação quântica baseadas na linguagem de programação Lisp. Nas próximas seções iremos discutir os aspectos mais relevantes neste trabalho, assim como seus objetivos e estrutura de apresentação.

1.1 Teoria de Grafos e Caminhos Aleatórios Quânticos

Uma das maiores conquistas da teoria clássica de algoritmos foi a introdução de aleatoriedade e a noção de algoritmo probabilístico. Muitos problemas têm bons algoritmos que usam caminhos aleatórios como subrotina. Por exemplo, o melhor algoritmo para resolver o problema 3-SAT [1] [Schöning 1999] é baseado em caminhos aleatórios.

Com tal motivação em mente o análogo quântico de caminho aleatório foi introduzido em 1993 por Aharonov, Davidovich e Zagury [Aharonov et al. 1993]. Sua contrapartida clássica, os caminhos aleatórios, são definidos como uma sucessão de passos, cada passo sendo dado em uma direção aleatória [Motwani e Raghavan 1995]. Os caminhos aleatórios quânticos seguem esta mesma noção, porém suas propriedades quânticas revelam resultados sem antecedentes clássicos. Isto leva à criação de algoritmos quânticos com eficiência equivalente ou superior àquela de algoritmos clássicos [Kempe 2003].

Os caminhos aleatórios quânticos podem ser aplicados para a solução de uma grande classe de problemas [Kempe 2003]. Neste trabalho buscamos sua aplicação em problemas relacionados à Teoria de Grafos [West 2001], que apresenta importantes desafios à Ciência da Computação, como o clássico problema do caminho mínimo, o problema do Caixeiro Viajante, problema da conectividade de vértices de um grafo e de grafos entre si, problema do isomorfismo de grafos, ou ainda desafios mais recentes ligados à reconstrução de seqüencias de genes ou à distribuição de valores lógicos em variáveis de uma fórmula

[1] Boolean satisfiability problem

lógica (n-SAT), por exemplo [Kempe 2003].

Há dois tipos diferentes de modelos de caminhos quânticos, que são o modelo contínuo, introduzido em [Farhi e Gutmann 1998] e o modelo de tempo discreto [Aharonov et al. 2001, Ambainis et al. 2001]. O modelo contínuo fornece uma transformação unitária diretamente no espaço no qual o caminho ocorre. O modelo discreto introduz um registrador de moeda extra e define um procedimento de dois passos que consiste de um lançamento de moeda quântica e de um passo controlado pela moeda.

As quantidades importantes para o *design* de algoritmos com caminhos aleatórios são *(i)* o seu tempo de mistura (*mixing time*) – o tempo necessário para se atingir a distribuição quase uniforme sobre o domínio – e *(ii)* o tempo final (*hitting time*) – o tempo necessário para se alcançar um certo ponto. Tais quantidades têm sido analisadas para vários grafos nos modelos contínuo e discreto. Um caminho quântico pode diminuir o tempo de mistura até quadraticamente em relação à sua contrapartida clássica, de modo que as performances quântica e clássica são polinomialmente relacionadas. Por outro lado o tempo final quântico é muito diferente do clássico. Já foi mostrado que há grafos tais que para dois vértices o tempo final clássico, de um vértice para outro, é polinomial no número de vértices do grafo, enquanto que o correspondente caminho quântico é exponencialmente mais rápido. Usando tais idéias, Childs et alii [Childs et al. 2003] construíram um problema onde o algoritmo baseado em caminhos aleatórios quânticos resulta num aumento exponencial da velocidade, comparado com o algoritmo clássico probabilístico. Permanece um problema aberto a questão de se tempos finais quânticos podem ser utilizados para aumentar a velocidade relativamente a algoritmos clássicos, para problemas relevantes.

Baseado neste trabalho um algoritmo de caminho quântico foi introduzido em [Shenvi, Kempe e Whaley 2003] para o problema de encontrar um vértice marcado em um grafo. Este algoritmo inicia com a superposição uniforme sobre todos os vértices. Em cada passo ele realiza um caminho aleatório, sendo que há duas regras locais para o caminho: em um vértice não marcado o caminho procede usualmente, mas num vértice marcado uma diferente regra de transição é aplicada (usualmente em um vértice não marcado uma moeda quântica é lançada e em um vértice marcado ela não o é). Depois de algum tempo a amplitude do estado se concentra no item marcado. Ou seja, uma medida encontra o item marcado com grande probabilidade.

Este algoritmo resolve o problema de Grover [Grover 1996] em um grafo. A princípio poderíamos indagar o porquê de se aplicar caminhos quânticos se já existe o algoritmo de Grover para tal. Há, no entanto, situações onde o passo de difusão R_ψ do algoritmo de Grover não pode ser implementado eficientemente, essencialmente devido ao fato de que a topologia local do banco de dados não permitiria, devido a limitações das portas quânticas e porque a inicialização de uma busca é muito custosa. Um caminho quântico faz somente transições locais e pode ser mais vantajosa. Um exemplo é a busca por um item marcado em um banco de dados 2-dimensional. Neste caso o algoritmo de Grover requer $\sqrt{N}$ perguntas, mas para deslocar a amplitude de um item do banco de dados para outro na malha, são necessários adicionalmente $\sqrt{N}$ passos em média por pergunta. A complexidade do algoritmo torna-se $\sqrt{N}\sqrt{N} = N$ e a vantagem quântica é perdida. O algoritmo, por sua vez, é capas de encontrar o item marcado num tempo $O(\sqrt{N}\log N)$ [Ambainis, Kempe e Rivosh 2005].

Um segundo exemplo da superioridade do caminho quântico sobre o algoritmo de Grover é dada em [Ambainis 2004]. Neste caso um caminho quântico é utilizado em um algoritmo para distinção de elementos, que roda em tempo ótimo $O(N^{2/3})$, o que significa uma melhora sobre os algoritmos baseados em Grover, que têm complexidade no tempo $O(N^{3/4})$. Vários novos algoritmos baseados em caminhos aleatórios quânticos com ganhos polinomiais sobre os algoritmos baseados no algoritmo de Grover foram já apresentados.

1.2 Linguagens de Programação Quântica

Para o estudo destes algoritmos baseados em caminhos aleatórios quânticos, dada a presente inexistência de computadores quânticos reais, sua simulação em computadores clássicos torna-se interessante. Isso pode ser realizado usando-se linguagens de programação quântica. Esta necessidade de simulação justifica-se por permitir a experimentação do comportamento dos caminhos aleatórios quânticos, assim como a comparação dos resultados obtidos com seus equivalentes clássicos, identificando assim o ganho ou não de eficiência de processamento.

Atualmente, por se tratar de uma área recente de pesquisa, poucas linguagens de programação quântica estão disponíveis. As simulações geralmente são feitas sem o uso

de uma linguagem quântica específica, mas por rotinas de *software*, dificultando assim a especificação de um algoritmo quântico. Portanto, também pretendemos com este trabalho investigar certas linguagens de programação quântica como QCL [Oemer 2000], QLisp [Desmet 2005] e QGAME [Spector 2004], com pretensão de estender as duas últimas, agregando-lhes novas funcionalidades e facilidade de utilização.

A noção de caminho quântico sobre uma linha introduzida por Aharonov et al. [Aharonov, Davidovich e Zagury 1993] foi generalizada para grafos em geral . Vários aspectos de caminhos aleatórios quânticos sobre grafos em dimensões superiores já são conhecidos [Kempe 2002] e serão aplicados em simulações neste trabalho. O objetivo é utilizar linguagens de programação para computação quântica para a simulação de caminhos quânticos sobre diversos tipos de grafos, como por exemplo QCL [2] [Oemer 2002], QLisp [3] e QGAME (*Quantum Gate and Measurement Emulator*)[Spector 2004]. Estas duas últimas linguagens são escritas em Lisp, extremamente apropriadas para a Computação Quântica. Lisp, mais precisamente o dialeto Common Lisp, é uma linguagem de programação multi-paradigma, permitindo o uso de poderosas técnicas de abstração para modelar circuitos quânticos, sem a limitação de um paradigma em particular. Por exemplo, enquanto que o paradigma funcional é conveniente para a implementação de operações matemáticas lineares, o paradigma orientado a objetos é mais conveniente para a implementação de estruturas de dados. A motivação para a criação de QLisp foi apresentar uma alternativa a simuladores realistas, como QCL, ou seja, simulações que tentam imitar conceitos do mundo real tão perfeitamente quanto possível. Tais simuladores quânticos tipicamente transformam operadores quânticos de alto nível em primitivos que podem ser executados em um hipotético processador quântico. Além disso, simuladores baseados em realidade física proíbem a execução de ações que não permitidas pelos postulados da mecânica quântica. Criadores de QLisp argumentam que simuladores quânticos com flexibilidade para a realização de experimentos não necessariamente permitidos fisicamente são necessários na investigação de novos algoritmos [Desmet 2005].

Por outro lado, QLisp trata simulação como um modelo e traduz computações quânticas diretamente para o formalismo matemático da mecânica quântica. Já um simulador realista transforma operadores de alto nível em um conjunto universal de operadores quânticos primitivos.

[2] `http://www-128.ibm.com/developerworks/linux/library/l-quant.html`

[3] `http://p-cos.net/documents/vub-prog-tr-06-15.pdf`

O emulador QGAME é capaz de manipular matrizes de forma explícita e implícita. Ele possui uma sintaxe para a expressão de programas quânticos e também um interpretador que simula sua execução. QGAME também fornece um modo de especificar algoritmos que incluem chamadas a portas de oráculos com qualquer número de entradas e uma saída. Estas portas são booleanas no sentido de que elas podem ter um dos dois possíveis efeitos nos seus *Qbits* de saída em qualquer chamada particular. Tal emulador é apropriado e tem sido usado para a aplicação de Computação Quântica automática.

1.3 Objetivos

Neste trabalho os principais objetivos são os seguintes:

1. *Aplicação de linguagens de programação quântica a problemas envolvendo caminhos quânticos aleatórios discretos em grafos.* Há diversos problemas abertos na teoria de caminhos quânticos sobre grafos. Por exemplo, o caminho quântico aleatório sobre um círculo de grau par não converge a uma distribuição uniforme (seus autovalores são degenerados). Sua distribuição estacionária depende do ponto de partida. Além disso, há vários grafos que não foram ainda estudados no contexto de caminhos quânticos (como o caminho sobre um grupo simétrico, por exemplo, fundamental em algumas questões interessantes como o isomorfismo de grafos). Também, a conexão entre dois modelos de caminhos quânticos ainda não é clara. Em muitos casos dois caminhos se comportam de forma muito semelhante, em outros, diferente. Seria interessante fazer a conexão entre dois modelos mais precisamente. Procuramos neste trabalho aplicar a linguagem de programação e simulador QGAME a um tipo de grafo específico: o grafo cíclico de N vértices. Nosso objetivo é demonstrar como procedemos para modelar o grafo de tal forma que pudesse ser simulado em um computador clássico. Entendemos por simulação de um grafo a execução de um algoritmo que conduza um "caminhante" por todos os vértices deste grafo – este por sua vez, sendo guiado pelo Caminho Aleatório Quântico – simulando assim uma busca exaustiva por todo o espaço de busca;

2. *Extensão das linguagens de programação para Computação Quântica.* As linguagens de programação quântica e seus simuladores, como QGAME, são relativamente in-

cipientes no que se refere a aplicações destas a problemas envolvendo caminhos aleatórios quânticos, geralmente havendo a carência de operadores e portas (ou funções) quânticas específicas para a simulação dos caminhos aleatórios. Neste sentido esperamos contribuir no desenvolvimento do simulador QGAME, agregando novas funcionalidades que permitam estas simulações.

Em resumo, o trabalho estuda a aplicação de caminhos aleatórios quânticos a grafos, pela discussão desta na forma mais simples de um grafo finito: o grafo cíclico de N vértices [Aharonov et al. 2001] – um grafo não-direcionado com N vértices arranjados em uma linha, sendo os vértices v_0 e v_{n-1} conectados, formando assim um círculo.

1.4 Estrutura de Apresentação

O trabalho inicia no Capítulo 2, onde apresentamos uma introdução concisa à Computação Quântica, de um ponto de vista matemático, partindo da definição dos conceitos fundamentais de *bit quântico* e *portas quânticas* até a união destes conceitos na criação dos *circuitos quânticos*, base para a implementação dos *algoritmos quânticos*.

No Capítulo 3 iniciamos o estudo de uma estratégia para a concepção de *algoritmos quânticos*, os Caminhos Aleatórios Quânticos. O estudo começa pela discussão necessária da versão clássica dos Caminhos Aleatórios. Concluímos o capítulo com a demonstração do uso de uma Linguagem de Programação Quântica (QGAME) para a simulação de uma versão simplista do Caminho de Hadamard. Esta discussão final possibilitará uma visão geral do método de simulação quântico que será utilizado nos próximos capítulos.

O Capítulo 4 contém uma breve introdução à Teoria de Grafos e apresenta de maneira formal o tipo de grafo abordado no estudo, como também sugere algumas aplicações de grafos na modelagem de problemas. O objetivo deste capítulo é agregar o conhecimento necessário sobre esta a Teoria de Grafos, tornando possível a discussão posterior sobre a aplicação do simulador quântico QGAME a grafos.

No Capítulo 5 procuramos introduzir os conceitos inerentes às Linguagens de Programação Quântica, como seus modelos de *hardware quântico* e os tipos de linguagens existentes, assim como uma análise das principais. Finalizamos a discussão dando ênfase ao simulador aplicado neste trabalho: QGAME.

As atividades de desenvolvimento aplicadas para a extensão de QGAME estão relatadas no Capítulo 6, como estas foram implementadas e o diferencial agregado à ferramenta que possibilitou a simulação de Caminhos Aleatórios Quânticos. Os resultados da simulação em um grafo cíclico utilizando a forma estendida de QGAME são detalhados também neste capítulo. Finalizamos este trabalho com o Capítulo 7 de conclusões e expectativas futuras à esta pesquisa.

2 Computação Quântica

A Computação Quântica compreende o estudo das tarefas que podem ser realizadas pelo processamento da informação contida em sistemas quânticos. Portanto, assim como na Computação Clássica, para um sistema quântico poder ser utilizado para computação, é necessária a representação desta informação, assim como a definicão destas tarefas [Nielsen e Chuang 2000].

Neste capítulo apresentamos estes elementos fundamentais. Primeiro a representação da informação através dos *Qbits*, os análogos quânticos dos *bits clássicos*. Em seguida, as operações que irão manipular estas unidades de informação: as portas quânticas.

A construção incremental destes conceitos no decorrer da discussão irá culminar na definição de circuito quântico, a base para a definição dos algoritmos quânticos – entre eles os algoritmos baseados em caminhos aleatórios quânticos – objeto de estudo deste trabalho.

2.1 O Bit Quântico

A Computação Clássica é baseada em um conceito abstrato fundamental conhecido como *bit*. Um *bit* (sigla para *Binary digIT*, ou dígito binário) é a menor unidade de informação de um computador clássico. Todo e qualquer dado em um computador digital atual é representado por seqüências desta unidade. Na Computação Quântica isso não é diferente. Existe também um conceito abstrato, uma unidade fundamental de informação chamada de *bit quântico* ou *Qbit*[1]. Portanto, é um conceito com bases matemáticas. Assim, embora os sistemas quânticos – necessários para o processamento de informação quântica – sejam implementados fisicamente, estes possuem um modelo matemático abstrato que permite a construção de uma Teoria Geral da Computação Quântica [Nielsen e Chuang 2000] mesmo antes de termos um computador quântico propriamente

[1]Notação utilizada por Mermin [Mermin 2003]. Também é comum encontrar na literatura o uso do termo *qubit*.

dito.

Isso facilita, e muito, as pesquisas em Computação Quântica, pois podemos manipular os objetos abstratos de um sistema quântico sem a necessidade de construí-los fisicamente. É importante salientar que o mesmo ocorre para os Computadores Clássicos. Antes mesmo da construção de um computador, Alan Turing já havia o tratado como um objeto abstrato, a Máquina de Turing.

Assim como o *bit* clássico (que denominaremos apenas *bit* neste trabalho, sendo o *bit* quântico diferenciado deste denominando-o *Qbit*), o *Qbit* também possui um estado. Um *Qbit* pode estar em dois estados possíveis: $|0\rangle$ ou $|1\rangle$. Esta a notação utilizada para a representação de estados quânticos, conhecida como *notação de Dirac* [Nielsen e Chuang 2000]. Nela, um *Qbit* tem seu estado x representado como $|x\rangle$ que foi denominado por Dirac de *ket*.

A diferença fundamental entre um *bit* e um *Qbit* está em que os *Qbits* podem formar *combinações lineares de estados*, ou seja,

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

onde α e β são escalares complexos. Se esta equação for analizada do ponto de vista da Álgebra Linear, podemos dizer que $|\psi\rangle$ é um *vetor* sendo $\{|0\rangle, |1\rangle\}$ uma base ortonormal deste espaço vetorial complexo.

Desta forma, um *Qbit* revela uma propriedade que fere o senso comum. Por exemplo, imaginemos uma moeda quântica: que podemos entender como um estado de dois níveis discretos, cara e coroa, zero ou um, porém este estado ao invés de ser representado computacionalmente por um *bit*, representamos este por um *Qbit*. Uma moeda quântica não lançada está numa superposição de cara e coroa. Isto está relacionado a uma diferença crucial entre um *bit* e um *Qbit* que é a incapacidade de se examinar um estado quântico individualmente. Em um Computador Clássico isto é possível e ocorre continuamente. Quando se recupera uma seqüencia de *bits* do registrador de um Computador Clássico, o estado está sendo examinado, ou seja, é realizada uma medição. Porém, isso não pode ser feito em um computador quântico. Não podemos examinar um *Qbit* a fim de determinar seu estado quântico. Ao invés disso, quando um estado quântico é examinado (ou medido) haverá uma probabilidade $|\alpha|^2$ deste estar no estado 0, e uma probabilidade $|\beta|^2$ deste estar no estado 1 [Nielsen e Chuang 2000]. As chamadas ampli-

tudes de probabilidade α e β podem ser determinadas somente estatisticamente, ou seja, através de um *ensemble* de experimentos idênticos. Isso implica que

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.2}$$

É importante notar que após a medida o estado não será mais uma superposição e sim $|0\rangle$ ou $|1\rangle$. Assim, após a medida o sistema quântico *colapsa*, saindo do "mundo quântico" para entrar no "mundo clássico".

Por exemplo, se um *Qbit* estiver no estado

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, \tag{2.3}$$

então após uma medida o estado do sistema será ou $|0\rangle$ ou $|1\rangle$ com probabilidade 1/2 para cada valor.

Assim, diferenciar o que pode ou não ser observado em um *Qbit* torna-se um dos principais desafios em Computação Quântica. A ausência de uma correspondência direta entre o estado quântico real e o estado observável torna esta disciplina não intuitiva. Porém, existe uma correspondência indireta que pode ser obtida pela manipulação dos estados de *Qbits*, levando-os aos resultados desejados, o que é suficiente para se obter o poder de processamento da Computação Quântica.

Embora os *Qbits* revelem propriedades estranhas, estes podem ser implementados fisicamente e foram exaustivamente validados por experimentos. Exemplos de sistemas físicos que os implementam são: o alinhamento de um *spin* nuclear em um campo magnético uniforme e os dois estados de um *elétron* orbitando ao redor de um átomo [Nielsen e Chuang 2000].

2.2 Múltiplos Bits Quânticos

Assim como a representação de um *bit quântico*, a representação de dois ou mais *Qbits* é análoga aos *bits* [Nielsen e Chuang 2000]. Desta forma, se para dois *bits* existem quatro estados possíveis escritos como 00, 01, 10 e 11, também existem quatro estados possíveis para dois *Qbits*, escritos por sua vez como $|00\rangle$, $|01\rangle$, $|10\rangle$ e $|11\rangle$.

Assim como para um único *Qbit*, dois ou mais *Qbits* podem estar em superposição, existindo para cada estado da base um coeficiente complexo chamado de *amplitude* [Nielsen e Chuang 2000]. Portanto, o vetor de estado que descreve dois *Qbits* é escrito como

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle. \tag{2.4}$$

Naturalmente, ao se medir este *Qbit* podemos obter um dos quatro estados possíveis (00, 01, 10 e 11) com uma probabilidade $|\alpha_x|^2$ para cada estado x. E portanto, também está presente a relação

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1. \tag{2.5}$$

2.3 Produto Tensorial

É interessante notar – mais uma vez a partir do ponto de vista da Álgebra Linear – que dois ou mais *Qbits* podem ser representados como vetores, formados pelo produto tensorial de seus pares. Podemos definir informalmente o produto tensorial como uma ferramenta para unir espaços vetoriais, criando espaços vetoriais ainda maiores. Para defini-lo formalmente, imaginemos V e W sendo espaços vetoriais de Hilbert com dimensões m e n, respectivamente. Então, $Z = V \otimes W$ é um espaço vetorial de dimensão $m.n$. Os elementos pertencentes a Z são obtidos pelo produto tensorial $v \otimes w$ de cada elemento $v \in V$ e $w \in W$. Ou pela notação de Dirac, $|v\rangle \otimes |w\rangle$. Assim, se $V = \{0, 1\}$, $|0\rangle \otimes |1\rangle + |0\rangle \otimes |0\rangle$ seria um elemento válido do espaço vetorial $V \otimes V$. Concluímos esta definição apresentando as propriedades básicas que devem ser satisfeitas pelo produto tensorial [Nielsen e Chuang 2000]:

1. Para um escalar z qualquer e elementos $|v\rangle \in V$ e $|w\rangle \in W$, $z(|v\rangle \otimes |w\rangle) = z(|v\rangle) \otimes |w\rangle = |v\rangle \otimes z(|w\rangle)$.

2. Para qualquer $|v_1\rangle, |v_2\rangle \in V$ e $|w\rangle \in W$, $(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle$

3. Para um elemento $|v\rangle$ qualquer e $|w_1\rangle, |w_2\rangle \in W, |v\rangle \otimes (|w_1\rangle + |w_2\rangle) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle$

Portanto, tendo duas matrizes $A = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ e $B = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$, seu produto tensorial é equivalente a

$$A \otimes B = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \otimes \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1.3 \\ 1.4 \\ 2.3 \\ 2.4 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 6 \\ 8 \end{bmatrix}. \tag{2.6}$$

Para o caso de dois *Qbits* podemos escrever [Mermin 2003]

$$|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle \tag{2.7}$$

onde podemos omitir o operador $\otimes$ tornando a notação mais compacta

$$|0\rangle|0\rangle, |0\rangle|1\rangle, |1\rangle|0\rangle, |1\rangle|1\rangle \tag{2.8}$$

ou ainda mais compacta, facilitando sua leitura

$$|00\rangle|01\rangle|10\rangle|11\rangle \tag{2.9}$$

e se os valores dos estados forem representados em base decimal, a notação assume uma forma ainda menor

$$|0\rangle_2|1\rangle_2|2\rangle_2|3\rangle_2 \tag{2.10}$$

esta forma pode ser generalizada através da representação

$$|x\rangle_n, 0 \leq x < 2^n \tag{2.11}$$

onde x é um número na base decimal e n indica a quantidade de *Qbits*. Por exemplo, o *Qbit* $|19\rangle_6$ pode ser representado através formas:

$$|19\rangle_6 = |010011\rangle = |0\rangle|1\rangle|0\rangle|0\rangle|1\rangle|1\rangle = |0\rangle \otimes |1\rangle \otimes |0\rangle \otimes |0\rangle \otimes |1\rangle \otimes |1\rangle \tag{2.12}$$

Todas estas formas tornam-se úteis na representação de *Qbits*. Porém, para a realização de operações em *Qbits*, é bastante útil a representação matricial, que discutiremos na seção seguinte.

2.4 Representação Matricial

Qbits podem ser representados matricialmente, por exemplo, através da associação

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.13}$$

Assim, o produto tensorial aplicado a dois vetores possui a seguinte forma matricial [Mermin 2003]

$$\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \otimes \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 y_0 \\ x_0 y_1 \\ x_1 y_0 \\ x_1 y_1 \end{bmatrix}. \tag{2.14}$$

Em conjunto com a notação que apresentamos na subseção anterior, podemos ter agora as seguintes possíveis representações para um *Qbit* qualquer

$$|5\rangle_3 = |101\rangle = |1\rangle|0\rangle|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.15}$$

onde o valor x em $|x\rangle_n$ especifica a linha da matriz do produto tensorial que contém o valor 1. No exemplo, refere-se à 5^a linha (contando de 0) da matriz resultante.

Após termos definido os objetos abstratos fundamentais da Computação Quântica, podemos analisar os operadores que irão manipular estes objetos. Estes operadores serão analisados na próxima seção.

2.5 Portas de Um Qbit

Assim como os computadores clássicos são construídos a partir de portas lógicas que formam circuitos lógicos e operam em *bits*, os computadores quânticos também são constituídos de circuitos quânticos, formados pela união de portas quânticas, que manipulam os *Qbits*. Estas portas quânticas podem operar tanto em um quanto em vários *Qbits*. Esta seção inicia a discussão das portas quânticas aplicadas em um único *Qbit*.

As portas lógicas clássicas de um *bit* fornecem apenas duas operações possíveis: *(1)* a operação de *identidade* e *(2)* a operação de *negação* [Mermin 2003]. Estas portas lógicas clássicas podem ter suas ações definidas a partir da execução destas em um *bit*

$$ID(0) \rightarrow 0 \qquad ID(1) \rightarrow 1, \tag{2.16}$$

$$NOT(0) \rightarrow 1 \qquad NOT(1) \rightarrow 0. \tag{2.17}$$

Portas lógicas podem ser representadas como sendo funções aplicadas ao *bit*. Neste caso, a aplicação da porta ID não provoca nenhuma mudança no *bit*, conservando seu estado. Porém, a porta NOT inverte o estado do *bit*.

Diferente das portas clássicas, as portas quânticas de um *Qbit* fornecem mais do que apenas duas operações. Até mesmo operações sem correspondente clássico podem ser implementadas.

Para a representação de portas quânticas, assim como *Qbits*, utilizamos matrizes. Discutimos a representação de *Qbits* como matrizes na Seção 2.4. A discussão da forma matricial das portas quânticas é aqui iniciada a partir da porta quântica de um *Qbit* para a operação de negação, a qual convencionou-se denominar porta X. A representação matricial desta porta quântica é dada pela matriz

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2.18}$$

É fácil perceber o comportamento desta porta quântica a partir de sua aplicação em um *Qbit* $\alpha|0\rangle + \beta|1\rangle$ representado em sua forma matricial

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}. \tag{2.19}$$

A ação da porta X é equivalente à porta clássica de negação. Porém, ao invés de se inverter o valor de um *bit* de 0 para 1 e vice-versa, são trocados os valores das amplitudes α por β. Para reforçar o entendimento, demonstramos a aplicação da porta quântica X aos *Qbits* $|0\rangle$ e $|1\rangle$ [Mermin 2003]:

$$X \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{2.20}$$

$$X \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{2.21}$$

Assim como existe um equivalente quântico para a porta lógica clássica de negação, existe também um equivalente para a porta clássica de identidade. A porta quântica para a operação de identidade é chamada I e sua representação matricial é dada por

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{2.22}$$

Novamente é fácil perceber o comportamento desta porta quântica a partir de sua aplicação em um *Qbit* $\alpha|0\rangle + \beta|1\rangle$ qualquer representado em sua forma matricial

$$I \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{2.23}$$

A partir da construção destas duas portas quânticas podemos concluir que qualquer porta quântica de um *Qbit* pode ser representada por matrizes de dimensão 2×2. Porém, nem todas as matrizes desta dimensão são portas quânticas válidas.

Na Seção 2.1 definimos a relação $|\alpha|^2 + |\beta|^2 = 1$ para um estado quântico $\alpha|0\rangle + \beta|1\rangle$ qualquer. Portanto, esta relação deve ser mantida mesmo após a atuação da

porta sobre o estado quântico original. Desta forma, podemos definir uma condição para a especificação de uma matriz U para uma porta quântica qualquer [Nielsen e Chuang 2000]:

1. U deve ser uma matriz unitária
2. $U^\dagger U = I$, onde $U^\dagger$ é a matriz adjunta de U, obtida transpondo-se e tomando-se o complexo conjugado de U, e I a matriz identidade de dimensão 2×2

Tendo que uma matriz adjunta é dada tomando uma matriz quadrada A, e sendo A' uma matriz onde cada elemento $a_{i,j}$ é substituído pelo determinante da matriz A excluindo-se as linhas i e colunas j, multiplicando ainda cada elemento por $(-1)^{i+j}$. A matriz adjunta é a matriz transposta de A'. Por exemplo, para a matriz

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}, \tag{2.24}$$

temos A' definida como

$$A' = \begin{bmatrix} a_{2,2}.(-1)^{1+1} & a_{2,1}.(-1)^{1+2} \\ a_{1,2}.(-1)^{2+1} & a_{1,1}.(-1)^{2+2} \end{bmatrix}, \tag{2.25}$$

e portanto, a matriz adjunta de A é dada por

$$adj(A) = (A')^T = \begin{bmatrix} a_{2,2} & -a_{1,2} \\ -a_{2,1} & a_{1,1} \end{bmatrix}. \tag{2.26}$$

Desta forma, a condição apresentada é a única restrição para que uma matriz seja uma porta quântica e desta forma, diferente das portas clássicas, podemos ter muitas portas quânticas interessantes de apenas um *Qbit*, sem correspondente clássico. Por exemplo, a porta quântica Z [Nielsen e Chuang 2000], definida pela matriz

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.27}$$

não altera o estado de $|0\rangle$ mas muda o sinal de $|1\rangle$ para $-|1\rangle$, um comportamento sem correspondência clássica.

A outra porta, essa ainda mais interessante, é denominada *porta Hadamard* e representada pela letra maiúscula H. Sua forma matricial é

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{2.28}$$

a porta Hadamard é também denominada *raiz quadrada de NOT* – pois a dupla aplicação desta porta resulta na operação NOT propriamente dita, ou seja $H^2 = NOT$ – e transforma o estado quântico $|0\rangle$ na superposição $(|0\rangle + |1\rangle)/\sqrt{2}$ e o estado $|1\rangle$ na superposição de estados quânticos $(|0\rangle - |1\rangle)/\sqrt{2}$.

Para facilitar a compreensão, a aplicação destas duas portas aos dois possíveis estados de um *Qbit* é dada por

$$Z|0\rangle = Z \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle, \tag{2.29}$$

$$Z|1\rangle = Z \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -|1\rangle, \tag{2.30}$$

$$H|0\rangle = H \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \tag{2.31}$$

$$H|1\rangle = H \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{2.32}$$

Outro detalhe interessante que faz parte das portas quânticas é a sua *reversibilidade*. Todas as portas quânticas descritas até aqui são reversíveis, ou seja, se uma porta quântica qualquer for aplicada duas vezes ao mesmo estado quântico, este irá retornar ao seu estado original [Mermin 2003].

Para demonstrar tal característica, tomamos como exemplo duas aplicações consecutivas das portas Z e H. Como a primeira das duas aplicações já demonstramos acima, a segunda aplicação é dada a seguir por

$$Z|0\rangle = Z \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle, \tag{2.33}$$

$$Z\left(-|1\rangle\right) = Z\begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle, \tag{2.34}$$

$$H\frac{|0\rangle + |1\rangle}{\sqrt{2}} = H\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle, \tag{2.35}$$

$$H\frac{|0\rangle - |1\rangle}{\sqrt{2}} = H\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle. \tag{2.36}$$

Note que todos os estados quânticos retornaram ao seu estado original, ou em outras palavras, seu estado quântico foi *revertido* ao seu estado original.

A mesma propriedade é válida para as portas quânticas X e I estudadas no início desta seção. Esta propriedade revela outra sutileza das portas quânticas comparadas com as portas lógicas clássicas. Como vimos, portas quânticas importantes são reversíveis. Isso não é verdade para as portas clássicas, cujas únicas operações reversíveis são as triviais operações de identidade e negação [Mermin 2003]. Portas lógicas clássicas importantes, como a porta lógica universal NAND não são reversíveis, pois por exemplo, a partir do estado 1 não podemos determinar os valores de entrada da porta que levaram a este estado: seriam os valores 0 e 1, 1 e 0 ou 0 e 0 ?

É importante notar que as matrizes

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},\ Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.37}$$

também denominadas matrizes de Pauli, junto com a matriz identidade

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.38}$$

formam uma base ortonormal para o espaço de Hilbert[2] real das matrizes hermitianas complexas 2×2. Uma matriz hermitiana é toda matriz quadrada de números complexos

[2]O espaço de Hilbert é um espaço vetorial sobre os complexos dotado de um produto interno

A, onde $A = A^*$, ou seja, a matriz A é equivalente à matriz complexa conjugada dela própria. Por exemplo, tendo a matriz A definida como

$$A = \begin{bmatrix} 1 & 3+i \\ 3-i & 2 \end{bmatrix}, \tag{2.39}$$

podemos dizer que A é uma matriz hermitiana, pois A é equivalente à sua matriz conjugada A^*.

Desta forma, a ação de qualquer operador sobre um vetor no espaço de estados bidimensional, pode ser expressa em termos destes operadores de base. É comum a notação $\sigma_x = X$, $\sigma_y = Y$, $\sigma_z = Z$, $\boldsymbol{\sigma} = (\sigma_x, \sigma_y, \sigma_z)$.

A partir da definição de portas de um *Qbit* podemos partir para a generalização destas portas, possibilitando a manipulação de múltiplos *Qbits*.

2.6 Portas de Múltiplos Qbits

A Computação Clássica possui um princípio chamado de *universalidade da porta NAND* [Nielsen e Chuang 2000]. Este princípio garante que qualquer porta lógica pode ser construída a partir da combinação de portas *NAND*, sejam elas de um ou mais *bits*.

Um princípio de universalidade de portas quânticas também existe. Este princípio define que qualquer porta lógica de múltiplos *Qbits* pode ser construída a partir da porta *CNOT* e de portas de um *Qbit* como as que vimos na seção anterior [Nielsen e Chuang 2000].

A porta quântica *CNOT* trata-se de uma porta de múltiplos *Qbits*. Mais especificamente, esta porta possui dois *Qbits* como valores de entrada, chamados de *Qbit de controle* e *Qbit alvo*. A Figura 2.1 mostra o circuito que implementa esta porta quântica.

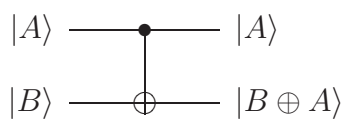

Figura 2.1: Circuito para porta quântica *CNOT*

A linha superior possui o *Qbit* de controle $|A\rangle$ e a linha inferior, por sua vez,

o *Qbit* alvo $|B\rangle$. O funcionamento desta porta se dá da seguinte forma: quando o *Qbit* de controle apresenta o estado 1, o *Qbit* alvo tem seu estado invertido; quando o *Qbit* de controle apresenta o estado 0, nada ocorre no *Qbit* alvo. Note que o estado do *Qbit* de controle é preservado. Este funcionamento é melhor compreendido pela demonstração da aplicação desta porta em todos os possíveis estados quânticos dos dois *Qbits* de entrada:

$$|00\rangle \rightarrow |00\rangle \tag{2.40}$$

$$|01\rangle \rightarrow |01\rangle \tag{2.41}$$

$$|10\rangle \rightarrow |11\rangle \tag{2.42}$$

$$|11\rangle \rightarrow |10\rangle \tag{2.43}$$

Assim como as portas de um *Qbit*, a porta *CNOT* também possui sua representação matricial dada por

$$N_C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.44}$$

Lembrando da representação matricial de *Qbits* que introduzimos na Seção 2.3, é possível aplicar esta porta a qualquer estado de um sistema quântico de dois *Qbits*. Por exemplo, para o caso em que o *Qbit* de controle tem seu estado quântico igual a 1 e o *Qbit* alvo possui o seu também igual a 1, temos a aplicação

$$N_C|11\rangle = N_C \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle \tag{2.45}$$

onde a coluna x (base 2) descreve a transformação que ocorre ao se multiplicar a matriz pela representação matricial do *Qbit* $|x\rangle$. Ou seja, a coluna 0 da matriz N_C descreve a transformação que sofrerá $|00\rangle$ (valor 0 em decimal), a coluna 1 descreve a transformação que sofrerá $|01\rangle$ (valor 1 em decimal), e assim sucessivamente. Esta característica é útil,

pois podemos mapear facilmente uma operação em uma matriz unitária, nos baseando somente em suas entradas e respectivas saídas [Nielsen e Chuang 2000].

Após termos uma visão de todos os elementos que compõe um circuito quântico (*Qbits* e portas quânticas) podemos discutir os circuitos quânticos em si.

2.7 Circuitos Quânticos

Na seção anterior apresentamos um circuito quântico simples (Figura 2.1) para a implementação da porta quântica *CNOT*. Porém, é importante ressaltar alguns detalhes sobre os circuitos quânticos.

Um circuito quântico é representado por um diagrama. Cada linha horizontal que interliga as portas quânticas e os demais elementos do circuito são chamadas de *fios*. Porém, diferente dos circuitos clássicos, estes fios não precisam necessariamente serem um objeto físico, como um fio de cobre [Nielsen e Chuang 2000]. Ao contrário, estas linhas podem representar por exemplo uma partícula de luz (*fóton*) se movendo de um local para outro no espaço. Mais uma vez, a abstração auxilia a esconder estes detalhes, que não precisam serem levados em conta no projeto de um circuito ou algoritmo quântico.

As entradas de um circuito quântico são sempre *Qbits* cuja base ortonormal é igual a $|0\rangle$ e $|1\rangle$.

As portas quânticas são representadas por caixas, com n fios de entrada e m fios de saída, sendo identificadas por uma letra (geralmente maiúscula) em seu interior, como mostra a Figura 2.2. A porta *CNOT* possui uma representação alternativa, como visto na Figura 2.1.

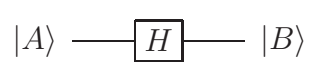

Figura 2.2: Circuito para porta quântica *Hadamard*

Embora os circuitos quânticos apresentem similaridades com os circuitos clássicos, certas operações não são permitidas em um circuito quântico [Nielsen e Chuang 2000]:

- Não podem haver *laços* de uma parte do circuito para outra, ou seja, os circuitos quânticos devem ser *acíclicos*

- Não podemos juntar fios de um circuito quântico. Esta operação é permitida em um circuito clássico (chamada *FAN-IN*) onde fios são unidos através de uma porta lógica *OU*. Como esta operação é não-reversível (e portanto, não-unitária) não podemos implementá-la em um circuito quântico

- Não podemos copiar *Qbits*. A operação inversa à anterior (*FAN-OUT*), onde um *bit* é copiado múltiplas vezes e transmitido em outros fios é impossível de ser realizada em um circuito quântico, já que a mecânica quântica proíbe a cópia de *Qbits*

Além dos *Qbits*, das portas quânticas e dos fios que interligam os elementos, um circuito quântico pode sofrer *medidas* no decorrer de sua execução. Como discutimos na Seção 2.1, a medida converte um estado quântico qualquer $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ em um estado clássico X, onde X poderá ter o valor 0 com uma probabilidade dada por $|\alpha|^2$ ou o valor 1 com uma probabilidade dada por $|\beta|^2$. A operação de medida também possui um símbolo no diagrama de circuitos quânticos [Nielsen e Chuang 2000]. A conversão de $|\psi\rangle$ em um estado clássico X pode ser descrita através do circuito visto na Figura 2.3. Note que o fio após o símbolo do medidor é composto por duas linhas horizontais paralelas ao invés de uma só. Isto diferencia um canal quântico (linha simples) de um canal clássico (linha dupla).

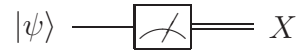

Figura 2.3: Representação de medida: estado quântico $|\psi\rangle$ é convertido em um estado clássico X

Para garantir a compreensão do conceito de circuito quântico, mostraremos um exemplo ligeiramente mais complexo, um circuito quântico gerador de estados de Bell[3].

O circuito é mostrado na Figura 2.4. Trata-se de uma operação Hadamard aplicada no primeiro *Qbit* ($|x\rangle$) sucedida de uma operação *CNOT* aplicada nos dois *Qbits* ($|x\rangle$ e $|y\rangle$), resultando nos dois *Qbits* $|\beta_x\rangle \times |\beta_y\rangle = |\beta_{xy}\rangle$ que corresponderão aos pares de Bell.

Como feito para o circuito quântico da porta *CNOT*, podemos descrever a

[3]Os estados de Bell, ou estados EPR, são estados quânticos importantes em mecânica quântica e possuem este nome em homenagem a Bell, Einstein, Podolsky e Rosen, os primeiros a estudar suas estranhas propriedades [Nielsen e Chuang 2000]

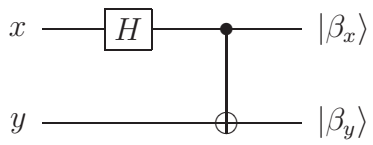

Figura 2.4: Circuito quântico gerador de estados de Bell

operação deste circuito através de sua aplicação em todos os possíveis estados quânticos dos dois *Qbits* de entrada:

$$|00\rangle \rightarrow (|00\rangle + |11\rangle)/\sqrt{2} \tag{2.46}$$

$$|01\rangle \rightarrow (|01\rangle + |10\rangle)/\sqrt{2} \tag{2.47}$$

$$|10\rangle \rightarrow (|00\rangle - |11\rangle)/\sqrt{2} \tag{2.48}$$

$$|11\rangle \rightarrow (|01\rangle - |10\rangle)/\sqrt{2} \tag{2.49}$$

Novamente, é possível a realização das operações do circuito quântico através de manipulação de matrizes [Nielsen e Chuang 2000]. Mostremos como exemplo, como a transição (2.40) é gerada.

O primeiro passo do circuito quântico (ou podemos chamar de algoritmo quântico) é a aplicação da porta Hadamard somente no primeiro *Qbit*

$$H|0\rangle \otimes |1\rangle = H \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes |1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes |1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |1\rangle, \tag{2.50}$$

no segundo passo aplicamos a porta $CNOT$ aos dois *Qbits*. É importante dar atenção à representação matricial dos *Qbits* em superposição do estado quântico $(|01\rangle + |11\rangle)/\sqrt{2}$

$$C_N \frac{|01\rangle + |11\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \frac{|01\rangle + |10\rangle}{\sqrt{2}}. \tag{2.51}$$

Este exemplo ilustra o fato de que qualquer algoritmo quântico pode ser implementado através de um circuito quântico.

Os circuitos quânticos normalmente são uma ferramenta para a criação de algoritmos quânticos. No exemplo anterior apresentamos um algoritmo quântico que gera estados de Bell através da especificação de um circuito. Em geral são utilizados tipos específicos de circuitos quânticos para a construção de novos algoritmos, restringindo o estudo dos algoritmos quânticos a classes específicas destes. Uma destas classes é a dos Caminhos Aleatórios Quânticos, que é utilizada neste trabalho como base para a criação de algoritmos quânticos. O estudo desta estratégia é o objetivo do próximo capítulo.

3 Caminhos Aleatórios Quânticos

Na Ciência da Computação, muitas vezes a utilização de métodos de resolução de problemas não-determinísticos se mostram eficientes. O Caminho Aleatório é um destes métodos, tornando característico seu não-determinismo por utilizar da aleatoridade na tomada de decisão para execução dos passos de um algoritmo.

Diversas aplicações podem ser encontradas em Ciência da Computação para Caminhos Aleatórios, uma das mais importantes é o uso desta estratégia para a solução do problema de satisfiabilidade de variáveis booleanas conhecido como 3-SAT [Schöning 1999]. Caminhos Aleatórios aparecem também em outros contextos como modelagem de movimentos aleatórios de moléculas em líquidos, gases e polímeros, além de modelos para processos cerebrais [Oliveira 2007].

Esta grande abrangência de aplicações motivou o estudo do análogo quântico dos Caminhos Aleatórios. Em 1993, Aharonov, Davidovich e Zagury [Aharonov et al. 1993] propusaram o análogo quântico dos Caminhos Aleatórios, definindo o termo e criando as bases para as atuais pesquisas deste método.

Neste capítulo revisaremos os conceitos fundamentais de Caminhos Aleatórios, iniciando com a versão clássica e, em seguida abordando a generalização quântica. Finalmente, apresentaremos a simulação de uma versão simplificada do Caminho de Hadamard, dando uma visão geral do método de simulação por linguagem de programação quântica que será empregado nos próximos capítulos.

3.1 Caminhos Aleatórios Clássicos

Para introduzir o conceito de Caminho Aleatório, imaginemos uma partícula qualquer situada na posição $x = 0$ de uma reta numérica, sendo $x \in R$, como mostra a Figura 3.1 [Oliveira 2007].

Esta partícula pode mover-se nesta reta, para a direita, quando somamos uma unidade l a x, e para a esquerda, quando subtraímos uma unidade l de x. A decisão de ir para a esquerda ou direita é determinada por uma moeda. Se a saída da moeda for cara,

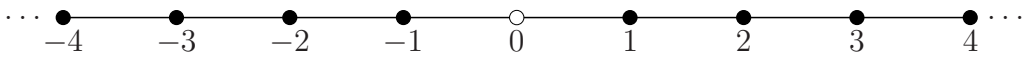

Figura 3.1: Partícula situada na posição $x = 0$ de uma reta numérica de números Reais

vai para a esquerda, se for coroa vai para a direita. Podemos assim definir um algoritmo simples: *(1)* lança a moeda e observa a saída; *(2)* conforme o valor da moeda, vai para a esquerda ou direita.

Assim, após repetirmos N vezes este algoritmo teremos a partícula em x localizada agora em

$$x = ml, \tag{3.1}$$

estando m no intervalo $-N \leq m \leq N$. Ou seja, a partícula "caminhou" m vezes o comprimento l.

O que nos interessa é determinar a probabilidade de após N passos encontrar a partícula em uma posição $x = ml$ qualquer. Partimos do raciocínio de que a quantidade N de passos dados pela partícula é, de maneira óbvia, equivalente a soma da quantidade de passos dados a esquerda (N_e) e a direita (N_d), ou seja

$$N = N_e + N_d, \tag{3.2}$$

e o deslocamento líquido da partícula é dado por

$$m = N_e - N_d. \tag{3.3}$$

Denotamos por p a probabilidade da partícula ir para a direita e por q a probabilidade de ir para a esquerda, de modo que

$$p + q = 1. \tag{3.4}$$

Como temos N_d passos para a direita e N_e passos para a esquerda, a probabilidade associada a uma seqüência de passos é dada pela multiplicação das probabilidades de todos os passos, ou seja

$$p^{N_d}.q^{N_e}. \tag{3.5}$$

Há um grande número de possibilidades de termos em N passos, N_d passos para a direita e N_e passos para a esquerda, sendo este número dado por [Oliveira 2007]

$$\frac{N!}{N_d!N_e!}. \tag{3.6}$$

Portanto, a probabilidade $P_N(N_d)$ num total de N passos, termos N_d para a direita e $N_e = N - N_d$ para a esquerda, é dada por

$$P_N(N_d) = \frac{N!}{N_d!N_e!}\left(p_d^N.q_e^N\right). \tag{3.7}$$

Como a partícula efetua N_d passos para a direita de um total de N passos, o deslocamento m fica totalmente determinado. Portanto,

$$P_N(m) = \frac{N!}{\left(\frac{N+m}{2}\right)!.\left(\frac{N-m}{2}\right)!}.p\left(\frac{N+m}{2}\right).q\left(\frac{N-m}{2}\right). \tag{3.8}$$

Na Figura 3.2 apresentamos a distribuição de probabilidade para um total de passos $N = 20$ e $p = q = 1/2$. Podemos notar uma curva característica de uma distribuição de probabilidade binomial, quando $p = q = 1/2$. Fica explícito que se trata de uma curva bem comportada. Além disso, podemos notar que após N passos, a probabilidade da partícula estar a uma distância N da origem é muito pequena, enquanto que a probabilidade da partícula estar nas redondezas da origem é bastante grande. Veremos em seguida que os Caminhos Aleatórios Quânticos diferem muito desta distribuição clássica.

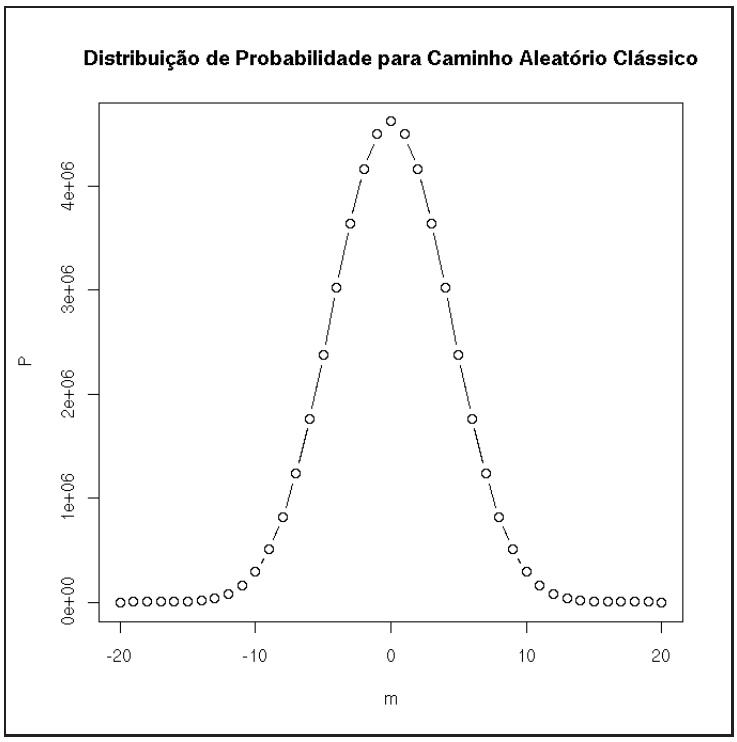

Figura 3.2: Distribuição de probabilidade binomial para o Caminho Aleatório Clássico com $N = 20$ e $p = q = 1/2$

3.2 Caminhos Aleatórios Quânticos

Faremos agora uma breve revisão de alguns resultados fundamentais da teoria Caminhos Aleatórios Quânticos tomando como base o artigo fundamental de Aharonov, Davidovich e Zagury [Aharonov et al. 1993] , que deu origem à área, e o artigo de revisão de Julia Kempe [Kempe 2003]. Apesar de não considerarmos todo o conteúdo de ambos artigos, apresentaremos em detalhe o desenvolvimento matemático não trivial, omitido nos trabalhos. Acreditamos que tal exposição tem valor pedagógico, pois não é facilmente encontrada na literatura da área, normalmente dedicada a especialistas.

Consideremos o exemplo original de Aharonov et al. Seja uma partícula de spin-$\frac{1}{2}$ situada em uma linha cuja posição x_0 é definida por um pacote de onda $|\psi_{x_0}\rangle$, onde a função

$$\psi_{x_0}(x) = \langle x|\psi_{x_0}\rangle \tag{3.9}$$

corresponde a um pacote de onda localizado em x_0 .

Consideremos o operador unitário[1]

$$U_l = \exp\left(\frac{-iPl}{\hbar}\right) , \tag{3.10}$$

onde P é o operador momento linear e l um deslocamento espacial. Tal operador é denominado operador translação gera um deslocamento l no espaço de estados de posição:

$$U_l|\psi_{x_0}\rangle = |\psi_{x_0+l}\rangle , \tag{3.11}$$

Além de ser caracterizado por uma posição, um sistema quântico de uma partícula pode ser caracterizado também por um grau de liberdade interna, como spin, que é interpretado como um momento magnético intrínseco. Esta quantidade aparece sempre quantizada em múltiplos da constante $\hbar/2$, onde $\hbar = h/2\pi$ e h é a constante de Planck. Partículas que possuem dois possíveis estados para uma medida de momento magnético, correspondendo a spin up ou down, são denominadas férmions. O formalismo já desenvolvido para sistemas de dois níveis é suficiente para tratar sistemas quânticos de férmions. O operador associado à medida destes estados de spin up ou down é dado por Z, definido pelas operações (2.29-2.30). O autoestados $|0\rangle$ e $|1\rangle$ correspondem a estados de spin up e down respectivamente, de modo que os autovalores +1 e -1 correspondem a medidas de autovalores de momento magnético up e down, respectivamente. Para termos autovalores associados aos valores físicos medidos, com dimensões de momento angular definimos $S_z = \hbar Z/2$, de modo que

$$S_z = \frac{\hbar}{2}\left(|0\rangle\langle 0| - |1\rangle\langle 1|\right) . \tag{3.12}$$

Portanto $S_z|0\rangle = \frac{\hbar}{2}|0\rangle$ e $S_z|1\rangle = \frac{-\hbar}{2}|1\rangle$. Na notação adimensional, em termos das matrizes de Pauli X, Y, Z, temos $Z|0\rangle = |1\rangle$ e $Z|1\rangle = -|1\rangle$.

Um sistema quântico de uma partícula numa dada posição, numa superposição de estados de spin up e down pode ser representada por

$$|\Psi\rangle = (\alpha_+|0\rangle + \alpha_-|1\rangle) \otimes |\psi_{x_0}\rangle. \tag{3.13}$$

. Veremos a seguir como é possível, através de operações físicas sobre o sistema (operações unitárias) correlacionar o spin da partícula com a direção do seu movimento. A idéia é gerar movimento randômico à esquerda ou direita através de medições de spin da partícula.

[1]Um operador unitário é aquele que preserva a norma dos vetores sobre os quais ele atua, ou seja, $U^\dagger U = UU^\dagger = I$, onde $U^\dagger$ é o hermitiano conjugado de U. Todos os operadores quânticos devem ser unitários.

Consideremos a ação do seguinte operador sobre o sistema descrito por (3.13):

$$U = exp\left[\frac{-i}{\hbar} Z \otimes Pl\right] . \tag{3.14}$$

Provaremos [2] a seguir que

$$U|\Psi\rangle = U(\alpha_+|0\rangle + \alpha_-|1\rangle) \otimes |\psi_{x_0}\rangle = \alpha_+|0\rangle \otimes |\Psi_{x_0+l}\rangle + \alpha_-|1\rangle \otimes |\psi_{x_0-l}\rangle. \tag{3.15}$$

Usando (3.12) e expandindo o operador no argumento da exponencial em (3.15) temos

$$\begin{aligned} U &= e^{\frac{-i}{\hbar}\sigma_z \otimes Pl} && (3.16)\\ &= exp\left[\frac{-i}{\hbar}\left(|0\rangle\langle 0| - |1\rangle\langle 1|\right) \otimes Pl\right] && (3.17)\\ &= exp\left[\frac{-i}{\hbar}|0\rangle\langle 0| \otimes Pl + \frac{i}{\hbar}|1\rangle\langle 1| \otimes Pl\right]. && (3.18) \end{aligned}$$

Usaremos agora o seguinte fato: se A e B comutam, $[A, B] = 0$, então

$$e^{A+B} = e^A e^B. \tag{3.19}$$

Como os operadores $|0\rangle\langle 0| \otimes Pl$ e $|1\rangle\langle 1| \otimes Pl$ comutam, (3.18) pode ser reescrita como

$$U = e^{\frac{-i\sigma_z}{\hbar} \otimes Pl} = e^{\frac{-i}{\hbar}|0\rangle\langle 0| \otimes Pl} e^{\frac{i}{\hbar}|1\rangle\langle 1| \otimes Pl}. \tag{3.20}$$

A exponencial de um operador é formalmente definida por

$$e^A = I + A + \frac{A^2}{2!} + \ldots, \tag{3.21}$$

que vale também para um produto tensorial de operadores:

$$e^{A \otimes B} = \sum_{k=0}^{\infty} \frac{[A \otimes B]^k}{k!}. \tag{3.22}$$

Portanto,

[2] Tal prova não é apresentada em forma detalhada nos artigos e teses pesquisados relativos assunto, de modo que sua apresentação aqui é conveniente.

$$e^{\frac{-i}{\hbar}|0\rangle\langle 0|\otimes Pl} = \sum_{k=0}^{\infty} \frac{\left[\frac{-i}{\hbar}|0\rangle\langle 0|\otimes Pl\right]^k}{k!} \tag{3.23}$$

$$= I - \frac{i}{\hbar}|0\rangle\langle 0|\otimes Pl + \left(\frac{-i}{\hbar}\right)^2 \frac{1}{2}\left(|0\rangle\langle 0|\otimes Pl\right)^2 + \dots . \tag{3.24}$$

Notemos agora que $(|a\rangle\langle a|)^k = |a\rangle\langle a|$, de modo que

$$e^{\frac{-i}{\hbar}|0\rangle\langle 0|\otimes Pl} = I + |0\rangle\langle 0|\otimes\left[\frac{-i}{\hbar}Pl + \left(\frac{-i}{\hbar}\right)^2\frac{(Pl)^2}{2!} + \left(\frac{-i}{\hbar}\right)^3\frac{(Pl)^3}{3!} + \dots\right]. \tag{3.25}$$

Como

$$e^{\frac{-i}{\hbar}Pl} = I + \left(\frac{-i}{\hbar}\right)Pl + \left(\frac{-i}{\hbar}\right)^2\frac{(Pl)^2}{2!} + \left(\frac{-i}{\hbar}\right)^3\frac{(Pl)^3}{3!} + \dots, \tag{3.26}$$

a eq. (3.25)- torna-se

$$e^{\frac{-i}{\hbar}|0\rangle\langle 0|\otimes Pl} = I + |0\rangle\langle 0|\otimes\left[e^{\frac{-i}{\hbar}Pl} - I\right]. \tag{3.27}$$

Por analogia, concluímos que

$$e^{\frac{i}{\hbar}|1\rangle\langle 1|\otimes Pl} = I + |1\rangle\langle 1|\otimes\left[e^{\frac{i}{\hbar}Pl} - I\right] \tag{3.28}$$

de modo que (3.20) torna-se

$$U = e^{-\frac{i}{\hbar}\sigma_z Pl} \tag{3.29}$$

$$= e^{-\frac{i}{\hbar}|0\rangle\langle 0|Pl}e^{\frac{i}{\hbar}|1\rangle\langle 1|Pl} \tag{3.30}$$

$$= \left[I + |0\rangle\langle 0|\otimes(e^{-\frac{i}{\hbar}Pl} - I)\right]\left[I + |1\rangle\langle 1|\otimes(e^{-\frac{i}{\hbar}Pl} - I)\right] \tag{3.31}$$

$$= I + |1\rangle\langle 1|\otimes\left(e^{\frac{i}{\hbar}Pl} - I\right) + |0\rangle\langle 0|\otimes\left(e^{\frac{i}{\hbar}Pl} - I\right) \tag{3.32}$$

$$= I - (|0\rangle\langle 0| + |1\rangle\langle 1|) + |0\rangle\langle 0|\otimes\left(e^{-\frac{i}{\hbar}Pl} + I\right) + |1\rangle\langle 1|\otimes\left(e^{\frac{i}{\hbar}Pl} - I\right) \tag{3.33}$$

$$= |0\rangle\langle 0|\otimes e^{-\frac{i}{\hbar}Pl} + |1\rangle\langle 1|\otimes e^{\frac{i}{\hbar}Pl}, \tag{3.34}$$

Portanto,

$$U|\Psi\rangle = \left[|0\rangle\langle 0|\otimes e^{-\frac{i}{\hbar}Pl} + |1\rangle\langle 1|\otimes e^{\frac{i}{\hbar}Pl}\right]\left[(\alpha_+|0\rangle + \alpha_-|1\rangle)\otimes|\psi_{x_0}\rangle\right] \tag{3.35}$$

$$= \alpha_+|0\rangle\otimes|\psi_{x_0+l}\rangle + \alpha_-|1\rangle\otimes|\psi_{x_0-l}\rangle, \tag{3.36}$$

o que prova nossa afirmação.

A equação acima mostra que U emaranha os diferentes graus de liberdade: a partícula move-se para a posição $x_0 + l$ sempre com o estado de spin $|0\rangle$ (*spin-up*) ou $x_0 - l$ com o estado de spin $|1\rangle$ (*spin-down*). A medição da componente z do spin corresponde a jogar uma moeda quântica: a cada passo jogamos a moeda para obtermos cara (spin up) e movimento a direita ou coroa (spin down) e movimento para a esquerda. O processo do Caminho Aleatório Quântico consiste na evolução do sistema através do repetido lançamento de moedas quânticas, ou seja, na repetição sucessiva do experimento.

Uma medida do spin da partícula relativamente à base $\{|0\rangle, |1\rangle\}$ produz os seguintes resultados:

1. *spin up*, posição $x_0 + l$ e estado $|0\rangle \otimes |\Psi_{x_0+l}\rangle$, com probabilidade $P_+ = |\alpha_+|^2$;
2. *spin down*, posição $x_0 - l$ e estado $|1\rangle \otimes |\Psi_{x_0-l}\rangle$, com probabilidade $P_- = |\alpha_-|^2$.

Uma nova aplicação de U sobre o estado resultante, após a primeira de spin resulta num deslocamento médio da partícula $l(p_+ - p_-)$. Uma repetição do procedimento T vezes (sempre medindo o spin na base $\{|0\rangle, |1\rangle\}$ e reinicializando o estado de spin como $\alpha_+|0\rangle + \alpha_-|1\rangle$), a partícula se moverá em média por uma quantidade

$$\langle x\rangle = Tl(P_+ - P_-) = Tl(|\alpha_+|^2 - \alpha_-|^2) \tag{3.37}$$

Consideremos agora uma modificação deste experimento, seguindo o artigo fundamental de Aharonov et al. [Aharonov, Davidovich e Zagury 1993], que consiste em passar a medir o spin agora ao longo da direção (θ, ϕ), onde

$$\phi = arg(\alpha_-/\alpha_+)\,, \tag{3.38}$$

ou seja,

$$\frac{\alpha_-}{\alpha_+} = \left|\frac{\alpha_-}{\alpha_+}\right| e^{i\phi}\,. \tag{3.39}$$

Os autoestados agora são:

$$|\theta, \phi, 0\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle\,, \tag{3.40}$$

$$|\theta, \phi, 0\rangle = \sin(\theta/2)|0\rangle + e^{i\phi}\cos(\theta/2)|1\rangle \tag{3.41}$$

Os autovalores são novamente ± 1 e o spin é agora representado pelo observável

$$\sigma_u := \sigma \cdot \hat{\mathbf{u}} = \sigma_1 u_1 + \sigma_2 u_2 + \sigma_3 u_3\,, \tag{3.42}$$

onde σ_i são as matrizes de Pauli (2.37) e

$$\hat{\mathbf{u}} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)\,. \tag{3.43}$$

A medição das componentes do spin do sistema $U|\psi\rangle$ relativamente à nova base é representada matematicamente pela aplicação do projetor $P(\theta,\phi) = |\theta,\phi,0\rangle\langle\theta,\phi,0| + |\theta,\phi,1\rangle\langle\theta,\phi,1|$ sobre $U|\psi\rangle$. Após a medida podemos encontrar *spin up* $+\hbar/2$, ou *spin down* $-\hbar/2$, sendo que o sistema colapsa, respectivamente para autoestados

$$|\psi_+\rangle = |\theta,\phi,0\rangle\langle\theta,\phi,0|U\psi\rangle \tag{3.44}$$

ou

$$|\psi_-\rangle = |\theta,\phi,1\rangle\langle\theta,\phi,1|U\psi\rangle\,. \tag{3.45}$$

Explicitamente temos

$$\begin{aligned}
&\langle\theta,\phi,0|U|\psi\rangle = \langle\theta,\phi,0|\left(\alpha_+|0\rangle e^{-\frac{iPl}{\hbar}} + \alpha_-|0\rangle e^{\frac{iPl}{\hbar}}\right)\otimes|\psi_{x_0}\rangle && (3.46)\\
=\ &\left(\cos(\theta/2)\langle 0| + e^{-i\phi}\sin(\theta/2)\langle 1|\right)\left(\alpha_+|0\rangle e^{-\frac{iPl}{\hbar}} + \alpha_-|0\rangle e^{\frac{iPl}{\hbar}}\right)\otimes|\psi_{x_0}\rangle && (3.47)\\
&= \left(\alpha_+\cos(\theta/2)e^{-\frac{iPl}{\hbar}} + \alpha_- e^{-i\phi}\sin(\theta/2)e^{\frac{iPl}{\hbar}}\right)|\psi_{x_0}\rangle\,. && (3.48)
\end{aligned}$$

Similarmente,

$$\langle\theta,\phi,1|U|\psi\rangle = \left(\alpha_+\sin(\theta/2)e^{-\frac{iPl}{\hbar}} - \alpha_- e^{-i\phi}\cos(\theta/2)e^{\frac{iPl}{\hbar}}\right)|\psi_{x_0}\rangle\,. \tag{3.49}$$

Portanto, os estados de *spin-up* e *spin-down* (3.44-3.45) podem ser descritos por

$$|\psi'_\pm\rangle = Z_\pm\left(\alpha_\pm e^{\mp\frac{iPl}{\hbar}} \pm e^{\mp i\phi}\tan(\theta/2)e^{\pm\frac{iPl}{\hbar}}\right)|\psi_{x_0}\rangle\,, \tag{3.50}$$

onde $Z_\pm$ são constantes de normalização.

Se a largura Δx do pacote inicial é muito maior do que o comprimento do passo l, podemos usar a aproximação

$$e^{\pm\frac{iPl}{\hbar}} = \left(I \pm \frac{iPl}{\hbar}\right)|\psi_{x_0}\rangle\,. \tag{3.51}$$

Portanto, (3.50) torna-se

$$\begin{aligned}
|\psi'_\pm\rangle &= Z_\pm\left[\alpha_\pm\left(I \mp \frac{iPl}{\hbar}\right) \pm \alpha_\mp e^{\mp i\phi}\tan(\theta/2)\left(I \pm \frac{iPl}{\hbar}\right)\right]|\psi_{x_0}\rangle && (3.52)\\
&= Z_\pm\left[\left(\alpha_\pm \pm \alpha_\mp e^{\mp i\phi}\tan(\theta/2)\right)I \pm \left(\mp\alpha_\pm + \alpha_\mp e^{\mp i\phi}\tan(\theta/2)\right)\frac{iPl}{\hbar}\right]|\psi_{x_0}\rangle\,. && (3.53)
\end{aligned}$$

A menos de uma constante de normalização podemos também escrever

$$|\psi'_{\pm}\rangle = \left[I \mp \frac{\alpha_{\pm} \mp \alpha_{\mp} e^{\mp i\phi} \tan(\theta/2)}{\alpha_{\pm} \pm \alpha_{\mp} e^{\mp i\phi} \tan(\theta/2)} \frac{iPl}{\hbar} \right] |\psi_0\rangle \tag{3.54}$$

$$= \left(I + \frac{iP}{\hbar} \delta l_{\pm} \right) |\psi_0\rangle \,, \tag{3.55}$$

onde

$$\delta l_{\pm} = \frac{\mp \alpha_{\pm} + \alpha_{\mp} e^{\mp i\phi} \tan(\theta/2)}{\alpha_{\pm} \pm \alpha_{\mp} e^{\mp i\phi} \tan(\theta/2)} \, l \,. \tag{3.56}$$

Notemos que estas equações são equivalentes às eqs. (11) de [Kempe 2003], com $\phi = 0$, $\alpha_+ = \alpha^{\downarrow}$, etc.

Substituindo (3.39) na expressão (3.56) para δl_+, obtemos no numerador

$$-\alpha_+ + \alpha_- \left(\frac{\alpha_-}{\alpha_+} \right) \left| \frac{\alpha_+}{\alpha_-} \right| \tan(\theta/2) \,. \tag{3.57}$$

Escolhendo

$$\tan(\theta/2) = - \left| \frac{\alpha_+}{\alpha_-} \right| (1 + \epsilon) \,, \tag{3.58}$$

temos

$$\delta l_+ = \frac{-\alpha_+ - \alpha_+(1+\epsilon)}{\alpha_+ - \alpha_+(1+\epsilon)} l = \left(1 + \frac{2}{\epsilon} \right) \simeq \frac{2l}{\epsilon} \,. \tag{3.59}$$

Portanto,

$$l \ll |\delta_+| \ll \Delta x \,. \tag{3.60}$$

3.3 Modelo Discreto e o Caminho de Hadamard

Apresentaremos aqui o modelo de tempo discreto dos Caminhos Aleatórios Quânticos. Este modelo apareceu já nos trabalhos de Feynman, tendo sido redescoberto mais tarde por Meyer, Watrous e mais recentemente por Aharonov et al. [Kempe 2003].

Nosso modelo terá uma única dimensão, uma reta. Novamente, como fizemos anteriormente (Seção 3.1) para os Caminhos Aleatórios Clássicos, imaginemos uma partícula em uma reta. H_p então será o espaço de Hilbert povoado pelas possíveis posições desta partícula. Desta forma teremos $H_p = \{ \, |i\rangle \, | \, i \in Z \, \}$, onde $|i\rangle$ corresponde a partícula localizada na posição i, se tomarmos como regra a função de onda $|\psi\rangle$.

Adicionamos ao espaço de posições H_p um outro espaço H_m, que tem como bases os possíveis estados da moeda quântica $|0\rangle, |1\rangle$, tomando um espaço de partículas

em spin-$\frac{1}{2}$. Assim, o sistema quântico para o modelo discreto terá como espaço de estados $H = H_p \otimes H_m$.

Após definir os espaços de estados do sistema quântico, podemos definir os operadores quânticos que irão atuar para a evolução do Caminho Aleatório. O operador de translação pode ser descrito como uma operação unitária

$$S = |0\rangle|0\rangle \otimes \left(\sum_i |i+1\rangle|i\rangle \right) + |1\rangle|1\rangle \otimes \left(\sum_i |i-1\rangle|i\rangle \right) \tag{3.61}$$

onde o índice i itera nos valores de Z. A aplicação deste operador S leva um estado $|0\rangle \otimes |i\rangle$ para $|0\rangle \otimes |i+1\rangle$ e $|1\rangle \otimes |i\rangle$ para $|1\rangle \otimes |i-1\rangle$.

Precisamos agora da operação de rotação da moeda quântica. Usaremos a clássica porta de Hadamard

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{3.62}$$

que caracterizará nosso Caminho Aleatório Quântico, podendo chamá-lo de Caminho de Hadamard. Unindo o operador de translação S e a porta de Hadamard H – ainda com o auxílio de um operador identidade I, que manterá o estado da moeda durante a translação – podemos definir o operador U que implementa o Caminho de Hadamard

$$U = S \times (H \otimes I). \tag{3.63}$$

Assim como para o Caminho Aleatório Clássico, podemos obter a distribuição de probabilidades para o Caminho de Hadamard, e realizar um comparativo com a distribuição binomial do Caminho Aleatório Clássico (Figura 3.2) demonstrado na Figura 3.3 [Kendon 2006].

Analisando o gráfico, podemos perceber um grande contraste entre a versão clássica e quântica do Caminho Aleatório. O caminhante quântico se propaga no espaço de busca quadraticamente mais rápido que o clássico. Podemos aproveitar esta característica para a criação de algoritmos possivelmente mais eficientes que seus correspondentes clássicos. Na próxima seção iremos aplicar uma linguagem de programação quântica para a simulação de uma versão ainda mais simplificada deste Caminho de Hadamard,

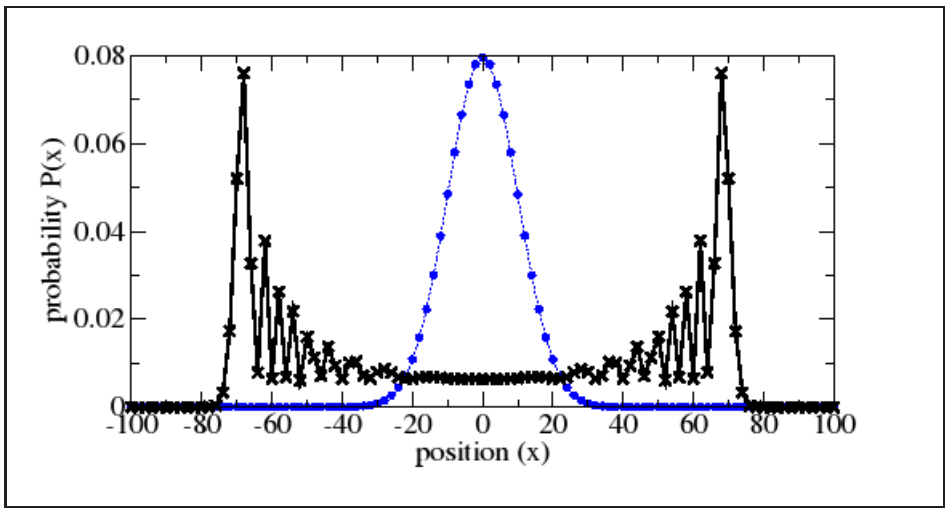

Figura 3.3: Comparação entre distribuição binomial do Caminho Aleatório Clássico (pontos) e da distribuição do Caminho Aleatório Quântico de Hadamard (cruzes) [Kendon 2006]

dando uma idéia de como atuaremos nas simulações dos próximos algoritmos e em outros problemas, introduzindo também o Caminho Aleatório em um grafo cíclico.

3.4 Simulação do Caminho de Hadamard

Após termos discutido os Caminhos Aleatórios Quânticos em sua abordagem geral e discreta, assim como o uso da porta de Hadamard para a obtenção do caminho que leva este nome, vamos introduzir aqui a simulação de algoritmos quânticos através de uma linguagem de programação quântica. O objetivo é mostrar uma visão geral sobre o método de simulação que será empregado nos próximos capítulos, para a simulação de algoritmos quânticos mais complexos.

Desta forma, simularemos uma versão mais simples do Caminho de Hadamard utilizando o simulador de Lee Spector, *QGAME* (*Quantum Gate and Measurement Emulator*) [Spector 2004].

QGAME é um simulador de algoritmos quânticos que utiliza, por padrão, da manipulação de matrizes de forma implícita para a evolução do sistema quântico em simulação. Ele define um conjunto básico de portas quânticas, representadas na forma clássica de expressões simbólicas de Lisp. Algumas portas são definidas como:

```
(hadamard qbit-alvo)
(swap qbit-controle qbit-alvo)
```

Estas operações implementam as portas de Hadamard e uma porta que inverte o valor do *Qbit* alvo, se o *Qbit* de controle for igual a $|1\rangle$. Algumas destas portas já foram discutidas no Capítulo 2. Podemos notar que cada porta especifica o *Qbit* em que opera (*qbit-alvo*), permitindo assim um futuro mapeamento do algoritmo em sua representação gráfica de circuitos. Além disso, podemos especificar novas portas quânticas que não as já definidas por padrão. Uma estratégia para tal é criar "oráculos", que são portas quânticas em forma de uma "caixa-preta". Ou seja, precisamos somente especificar a matriz que mapeia as entradas da porta em suas saídas, deixando a decomposição da porta em função de portas quânticas unitárias para mais tarde. Esta é uma das grandes vantagens de se utilizar linguagens de programação quânticas para a simulação de algoritmos quânticos.

A nossa versão simplista do Caminho de Hadamard compreenderá apenas duas operações:

1. Aplicação da porta de Hadamard ao *Qbit* de moeda;
2. Aplicação do operador de transição S aos *Qbits* de posição.

Portanto, precisaremos além da porta Hadamard, especificar um oráculo, que implementará o operador de transição S. Para isso, precisamos definir sua matriz de mapeamento. Vamos delimitar o espaço de estados do nosso caminhante quântico, fazendo-o caminhar por um grafo cíclico, onde cada vértice corresponde a um estado quântico. Para este exemplo, tomemos um grafo com número de vértices $V = 4$. Portanto, precisaremos de N *Qbits*, sendo $2^N = 4$. Somando ao sistema quântico mais um *Qbit* para ser utilizado como moeda quântica, teremos um sistema quântico de $N = 3$ *Qbits*.

Através da análise do grafo direcionado apresentado na Figura 3.4, podemos inferir as relações de transição: onde para todo estado quântico $|\alpha\beta\gamma\rangle$, os dois primeiros *Qbits* $|\alpha\beta\rangle$ representam os vértices, e o último *Qbit* $|\gamma\rangle$ representa o estado corrente da moeda quântica, e por conseguinte, a aresta tomada para chegar ao estado seguinte – 0 para a aresta a direita e 1 para a aresta a esquerda:

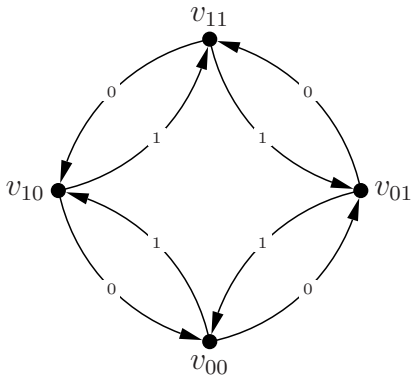

Figura 3.4: Grafo direcionado de vértices $V = 4$ para uma versão simplificada do Caminho de Hadamard

$$
\begin{aligned}
|000\rangle &\rightarrow |010\rangle, \\
|001\rangle &\rightarrow |101\rangle, \\
|010\rangle &\rightarrow |110\rangle, \\
|011\rangle &\rightarrow |001\rangle, \\
|100\rangle &\rightarrow |000\rangle, \\
|101\rangle &\rightarrow |111\rangle, \\
|110\rangle &\rightarrow |100\rangle, \\
|111\rangle &\rightarrow |011\rangle,
\end{aligned}
$$

a partir destas relações, estas transições podem ser representadas matricialmente por

$$
S = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}. \tag{3.64}
$$

Este operador é implementado a seguir em *QGAME*, conforme o Algoritmo 3.1.

```
(defun transicao (qsys q1 q2 q3)
  (apply-operator qsys
                  #2A((0 0 0 0 1 0 0 0)
                      (0 0 0 1 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 1 0)
                      (0 1 0 0 0 0 0 0)
                      (0 0 1 0 0 0 0 0)
                      (0 0 0 0 0 1 0 0))
                  (list q1 q2 q3)))
```

Algoritmo 3.1: Definição de operador de transição como uma função em QGAME

Por fim, utilizamos a função *RUN-QSYS* de *QGAME* para simular o algoritmo definido. Esta função necessita que seja instanciado um sistema quântico que irá simular o algoritmo ou programa. Note que especificamos no Algoritmo 3.2 a quantidade de *Qbits* do sistema quântico em 3.

```
(defun caminho-hadamard-simples ()
   (run-qsys
      (make-instance 'quantum-system
                     :number-of-qubits 3
                     :program '((hadamard 0)
                                (transicao 1 2 0)
                                . . .
                                (hadamard 0)
                                (transicao 1 2 0)))))
```

Algoritmo 3.2: Definição do caminho de Hadamard como uma função em QGAME

O algoritmo foi simplificado para facilitar a exibição. Porém, o que ocorre é a repetição da aplicação dos operadores de Hadamard e de transição N vezes. Para a execução deste algoritmo com $N = 10$ passos, obtivemos a Tabela 3.1 de amplitudes.

Analisando as amplitudes, podemos perceber que nosso caminhante quântico – partindo do vértice inicial, $|00\rangle$, em superposição – alcançou o vértice mais distante – $|11\rangle$ – já na segunda iteração do algoritmo. A partir deste exemplo, é possível ter uma idéia do método de simulação a ser aplicado nos próximos capítulos, onde investigaremos

Passo	$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
0	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	0	0	0	0	0	0
1	0	0	0	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	0	0	0
2	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	$\frac{1}{2}$	$-\frac{1}{2}$
3	0	0	0	0	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	0	0
4	0	0	0	0	0	0	1	0
5	0	0	$\frac{1}{\sqrt{2}}$	0	0	$\frac{1}{\sqrt{2}}$	0	0
6	$\frac{1}{2}$	$-\frac{1}{2}$	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$
7	0	0	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	0	0	0	0
8	1	0	0	0	0	0	0	0
9	0	0	0	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	0	0	0
10	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	$\frac{1}{2}$	$-\frac{1}{2}$

Tabela 3.1: Amplitudes do sistema quântico de 3 *Qbits* após $N = 10$ passos de iteração.

a utilização de simulações utilizando linguagens de programação quânticas, aplicados a versões mais complexas do Caminho Aleatório Quântico e problemas de grafos. Esta investigação nos permitirá verificar a correção e eficiência deste método. Portanto, uma investigação das principais linguagens de programação quântica e seus simuladores torna-se necessária. Porém, antes é preciso conhecer o domínio dos problemas tratados por estas ferramentas, a Teoria dos Grafos, assunto do próximo capítulo.

4 Introdução a Grafos

A Teoria de Grafos abrange o estudo dos modelos de estruturas combinatórias chamados grafos [Yellen 1998]. Estas estruturas estão presentes em um vasto número de aplicações, podendo representar redes físicas, como circuitos elétricos, rodovias, dutos de fiação e até mesmo moléculas orgânicas. Ainda, interações pouco perceptíveis como presentes em nossa sociedade e meio-ambiente – como os relacionamentos pessoais – ou em áreas de estudo como bancos de dados e controle, visto que autômatos, por exemplo, têm toda sua dinâmica modelada através de grafos [Yellen 1998].

Pretendemos com este breve capítulo introduzir os conceitos fundamentais ligados à Teoria de Grafos, viabilizando o entendimento dos grafos cíclicos abordados no trabalho.

4.1 Grafos e Grafos Cíclicos

Podemos definir um grafo G qualquer como sendo *(i)* um conjunto de elementos não-vazio V em união com *(ii)* um conjunto R de elementos que representam as relações entre os vértices pertencentes a V. O conjunto R é simétrico, ou seja, para cada par ordenado $(u, v) \in R$, também $(v, u) \in R$. V é designado *conjunto de vértices de* G, e cada elemento $v \in V$ é chamado *vértice de* G. O conjunto R é chamado de *conjunto de arestas de* G, onde cada elemento $(u, v) ou (v, u) \in R$ denominamos *aresta de* G [Chartrand 1984]. Por exemplo, podemos definir um grafo G_1 sendo seus vértices $V = \{v_0, v_1, v_2, v_3, v_4\}$ e $R = \{(v_0, v_4), (v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1)\}$.

Por representar relações entre elementos, utilizamos uma representação gráfica de linhas e pontos para facilitar a visualização de grafos. Cada vértice é representado como um ponto, e as linhas representam as arestas que conectam um ponto ao outro. Desta forma, para o grafo G_1 definido, teremos sua representação gráfica dada pela Figura 4.1:

Ainda, grafos podem ter suas arestas orientadas, ou seja, podemos especificar para cada aresta, quais são seus vértices de origem e destino. Desta forma, o conjunto R perde sua característica de simetria, pois a ordem dos pares $(u, v) \in R$ determinam

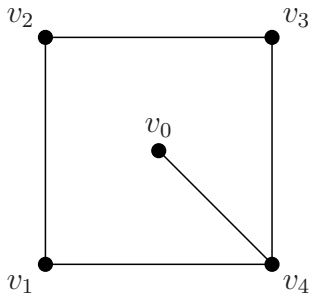

Figura 4.1: Representação gráfica para grafo G_1

os vértices de origem e destino, equivalente a u e v, respectivamente. A representação gráfica de grafos direcionados utiliza para a representação das arestas, setas ao invés de linhas, partindo do vértice de origem e tendo como alvo o vértice de destino. Por exemplo, para o grafo G_2 definido pelos conjuntos $V = \{v_0, v_1, v_2\}$ e $R = \{(v_0, v_1), (v_1, v_2), (v_2, v_0)\}$, teremos sua representação gráfica equivalente a Figura 4.2.

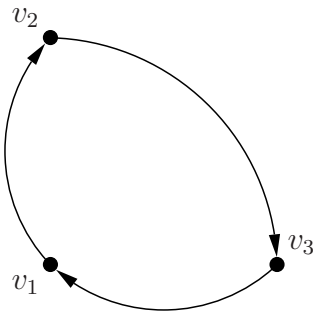

Figura 4.2: Representação gráfica para grafo orientado G_2

Como prova de conceito, procuramos implementar o caminhante guiado pelo algoritmo de Caminho Aleatório Quântico por um grafo cíclico de N vértices. Definimos este tipo de grafo pelos conjuntos

$$V = \{v_0, v_1, v_2, ...v_{n-1}\} \tag{4.1}$$

$$R = \{(v_0, v_1), (v_1, v_2), ..., (v_{n-2}, v_{n-1}), (v_{n-1}, v_0)\} \tag{4.2}$$

onde notamos que o último vértice possui conexão com o primeiro vértice, caracterizando portanto o ciclo. Neste caso, o grafo G_2 definido anteriormente, pode ser classificado como um grafo cíclico de N vértices, com $N = 3$. É comum utilizar a notação C_n para designar um grafo cíclico com n vértices. Em um grafo cíclico, o número de vértices $v \in V$ tem o mesmo número de arestas $r \in R$. Desta forma, se temos que $len(A)$ é definida como uma operação que retorna o número de elementos $x \in A$, então para um grafo cíclico C_n temos que $len(V) = len(R) = n$.

Podemos identificar algumas propriedades inerentes a grafos cíclicos:

1. *Conectado.* Em um grafo G qualquer, se dois vértices u e v possuem uma aresta que os conecte, são ditos então conectados. Um grafo é portanto dito conectado, se todos os seus pares de vértices u e v são conectados. Grafos cíclicos possuem pares de vértices conectados e portanto são ditos grafos conectados;

2. *Eureliano.* Um caminho Eureliano compreende um conjunto de vértices onde é possível a um caminhante no grafo, percorrer todos os vértices $v \in V$, visitando estes vértices somente uma única vez. Um grafo cíclico é dito Eureliano pois permite a ocorrência de um caminho Eureliano como o definido;

3. *Hamiltoniano.* Um ciclo Hamiltoniano compreende um conjunto de vértices onde é possível a um caminhante no grafo, percorrer todos os vértices $v \in V$, visitando-os somente uma única vez e permitindo ao caminhante partir do primeiro vértice v_0 e voltar a este mesmo vértice. Pelo fato de o grafo cíclico permitir a ocorrência de ciclos Hamiltonianos, é dito um grafo Hamiltoniano;

4. *Regular de grau 2.* Um grafo regular em Teoria de Grafos é todo aquele no qual cada vértice $v \in V$ possui o mesmo número de vizinhos. Portanto, um grafo cíclico C_n é dito como grafo regular de grau 2, visto que todos os vértices $v \in V$ de C_n possuem o número de vizinhos igual a 2.

4.2 Principais Aplicações

Grafos estão presentes em problemas de diversas áreas: ligados à transporte, como o problema da ponte de Königsberg e o problema do caixeiro viajante; relacionados também a conexão entre locais, como o problema do conector mínimo; satisfação de requisitos, como o caso do problema 3-SAT; distribuição de tarefas, como o problema de *schedulling*, além de diversos problemas de jogos e *quebra-cabeças*, como o problema do percurso do cavalo em um tabuleiro de jogo de xadrez, e as Torres de Hanói [Chartrand 1984]. Os grafos cíclicos, como sendo um subgrupo de grafos, também apresentam aplicações interessantes.

Grafos cíclicos estão relacionados com problemas de busca. Por exemplo, o problema de determinar a conectividade de dois vértices em um grafo G qualquer en-

volve um algoritmo de busca, geralmente aplicando algoritmos como o *breadth-first search* [Norvig 2003]. Trata-se de um problema de complexidade $O(log\, n)$, provado por Omer Reingold [Reingold 2005]. O problema clássico do caixeiro viajante (ou *TSP, Traveling Salesman Problem*) também pode ser modelado como um grafo cíclico. Este por sua vez, pertence à família dos problemas *NP-hard* [Sipser 1997], e portanto não possuem uma solução clássica que consiga resolvê-los em tempo polinomial, tornando-os fortes candidatos à aplicação de algoritmos quânticos.

Em Teoria de Grupos, os grafos cíclicos também influenciam aplicações. Grupos Cíclicos são grupos G de elementos gerados por um único elemento $g \in G$, chamado de *gerador*. Ou seja, os elementos de um grupo podem ser representados como uma potência de g. Colocando de outra forma, para um grupo G qualquer de 5 elementos $g \in G$, poderíamos ter

$$G = \{g^0, g^1, g^2, g^3, g^4\} \tag{4.3}$$

o que nos leva a perceber que tal conjunto é um clico e portanto, pode ser modelado por um grafo cíclico. Além disso, grafos cíclicos podem ser aplicados para codificar estruturas de grupos, onde tais grafos são conhecidos como *grafos de Cayley* [Wolfram 2002].

Tendo definido formalmente os tipos de grafos abordados neste trabalho, no próximo capítulo caminharemos para a implementação em si, que inicia com uma discussão das principais linguagens de programação quântica, apresentando os motivos que levaram à escolha de QGAME para o desenvolvimento.

5 Linguagens de Programação Quântica

As linguagens de programação buscam criar paradigmas ou abstrações de alto nível para que possamos pensar na solução de problemas, sem nos preocuparmos com detalhes de baixo nível. É esta abstração que torna transparente detalhes do *hardware* que estamos utilizando para desenvolver determinado *software*, por exemplo. Ou ainda, são estes modelos que proporcionam o aproveitamento de certas estruturas de dados e controle complexas, como as criadas para os populares paradigmas de programação imperativo, orientado a objetos, funcional e afins.

Os sistemas quânticos também possuem detalhes que muitas vezes poderiam ser abstraídos do desenvolvedor, ou ainda, seria interessante a definição de um conjunto bem definido de propriedades somente encontradas em sistemas físicos quânticos. O desenvolvimento das linguagens de programação quântica é motivado por esta criação de abstrações que modelem características como o emaranhamento, superposição e paralelismo quântico, além de permitirem a criação de um *framework* para a especificação formal de operações quânticas e sua execução, favorecendo sua visualização e análise, antes mesmo de sua implementação em um *hardware* físico [Selinger 2004].

Como o objetivo deste trabalho é a utilização de linguagens de programação quântica para a modelagem e simulação de algoritmos quânticos, a discussão destas torna-se necessária. Este capítulo busca portanto, a apresentação dos conceitos ligados às linguagens de programação quântica, os modelos de *hardware* em que estas linguagens seriam executadas, para então analisar as linguagens tomadas para o desenvolvimento.

5.1 Modelos de Hardware

Existem atualmente, segundo [Selinger 2004], três modelos de *hardware* conceituais para a execução de programas escritos em uma linguagem de programação quântica:

1. **Modelo de Circuito Quântico:** assim como os circuitos clássicos são compostos pelo arranjo de portas lógicas, os circuitos quânticos utilizam portas quânticas. Porém estas portas (como discutido no Capítulo 2) são sempre reversíveis e corres-

pondem a transformações unitárias em um espaço vetorial complexo. Na seqüência de execução das operações, ou seja, durante a execução do algoritmo (ou programa) quântico, as operações de medida são deixadas sempre como o último passo da computação, preservando suas propriedades quânticas;

2. **Modelo QRAM:** este modelo proposto por Knill [Knill 1996] facilita sua utilização para a interpretação de linguagens de programação quântica, já que o *hardware* consiste em uma memória de *Qbits* indexada aleatoriamente – da mesma maneira que as memórias de acesso aleatório clássicas RAM (*Random Access Memory*) – operada por um computador clássico que faz chamadas à realização das operações, como: "aplique a operação *Hadamard* nos *Qbits* i e j". Desta forma, as medições e as operações unitárias podem se dar durante toda a execução do programa, sem a perda do estado global do sistema;

3. **Máquina de Turing Quântica:** é bastante utilizada para estudo em Análise de Complexidade. As medidas nunca são realizadas neste modelo, e toda a operação da máquina (fita, cabeça de leitura/gravação e controle finito) é assumida como unitária. Embora seja teoricamente equivalente aos outros dois modelos, acredita-se que não é uma aproximação realística do que um computador quântico pode vir a ser.

Portanto, os modelos de circuito quântico e QRAM apresentam-se como as principais alternativas para a interpretação de programas escritos em linguagens de programação quântica. Sendo que a definição de uma linguagem completa para a computação quântica ainda apresenta-se como um desafio, atualmente o que existem são protótipos de linguagens em desenvolvimento, cada uma focando em um paradigma e modelo de *hardware* específico, o que torna necessária uma análise das principais linguagens, apontando suas diferenças e permitindo a escolha de uma opção para o desenvolvimento do trabalho.

5.2 Linguagens Analisadas

Pelo fato de haver diversas linguagens de programação quântica, cada qual operando sobre um determinado paradigma ou *hardware* quântico, é interessante que analisemos as principais implementações existentes:

5.2.1 QLisp

Desenvolvida por Brecht Desmet et al. [Desmet 2005], QLisp é um simulador integrado à linguagem *Common Lisp*. Todas as computações quânticas realizadas em QLisp são traduzidas para seu formalismo matemática, e podem ser observadas em tempo de execução, facilitando o acompanhamento da execução destas computações. Além disso, QLisp possui técnicas de otimização de código implementadas, dando-lhe uma melhor performance na simulação. QLisp também é uma das linguagens utilizada como fonte de informação para o desenvolvimento deste trabalho, já que acaba por dividir certas características com QGAME: mesma linguagem de implementação, mesmo modelo de *hardware* virtual, implementado como uma extensão à *Common Lisp* e não como uma nova linguagem.

5.2.2 QCL

Desenvolvida por Bernhard Ömer durante vários anos [Oemer 1998, Oemer 2000, Oemer 2001, Oemer 2002, Oemer 2003], QCL foi a primeira linguagem de programação quântica real. Foi desenvolvida e tem sua sintaxe inspirada na linguagem de programação C. Sua especificação é bastante completa, porém sua implementação criou a necessidade da construção de não só uma linguagem de programação quântica, mas também de uma linguagem de programação clássica. Desta forma, embora QCL possua recursos avançados para gerenciamento de memória e afins, não se compara a maturidade de linguagens já existentes. Assim, a estratégia de extensão de uma linguagem já estável como a utilizada por QGAME e QLisp parece ser mais adequada.

5.2.3 Linguagem de Bettelli et al.

Bettelli et al. [Serafini 2003] criaram uma linguagem que utiliza do modelo QRAM. Sua implementação se dá como uma extensão à linguagem C++ através da criação de uma biblioteca, onde as operações quânticas são tratadas como objetos de primeira classe.

5.2.4 qGCL

Sanders & Zuliani [Sanders e Zuliani 2000, Zuliani 2001] definiram a linguagem qGCL. Baseada em uma linguagem *guarded-command*, qGCL é mais uma especificação formal do que uma linguagem de programação. Permitiu a Zuliani a definição de algoritmos quânticos com não-determinismo e estados misturados [Gay 2006].

5.2.5 QFC, qHaskell e Outras Linguagens Funcionais

Van Tonder [Tonder 2004] definiu um λ-cálculo quântico e a partir de então as pesquisas em linguagens de programação quântica funcionais tomaram fôlego. Valiron & Selinger [Valiron 2004, Selinger 2007] criaram uma linguagem de programação funcional de alta ordem baseada no modelo de controle clássico e dados quânticos (modelo QRAM) e mais tarde a linguagem de primeira ordem QML foi criada por Altenkirch e Grattage [Grattage 2005, Grattage 2005], aonde tanto o controle quanto os dados são quânticos. Haskell também é alvo de diversas investigações no que se refere ao seu uso como base para programação quântica e os trabalhos de Sabry, Vizzoto e da Rocha Costa [Sabry 2003, Costa 2005] são os mais recentes e definem uma extensão à Haskell, possibilitando a criação de operadores e estados quânticos através de *quantum arrows*.

5.2.6 Quantum Gate and Measurement Emulator

QGAME, ou *Quantum Gate and Measurement Emulator* foi desenvolvida por Lee Spector [Spector 2004] como parte de seu sistema de geração automática de código através de programação genética.

Foi desenvolvido originalmente em *Common Lisp*, porém atualmente possui uma versão em *C++*. A linguagem de programação quântica de QGAME consiste em um conjunto de portas quânticas que – utilizando o modelo de circuito quântico – podem ser combinadas para a criação de algoritmos.

Além de sua facilidade de extensão, QGAME destaca-se por permitir, quando utilizada em conjunto com a linguagem PUSHGP[1], a geração de código por programação genética. O fato de QGAME ser uma extensão em nível sintático à linguagem *Common*

[1]The Push Programming Language: `http://hampshire.edu/lspector/push.html`

Lisp permite que todas as funcionalidades de uma linguagem de programação clássica estejam disponíveis, somente agregando a esta, as propriedades quânticas, como os estados quânticos, operadores e afins. Estas características fizeram com que QGAME fosse o simulador escolhido para o desenvolvimento deste trabalho. No próximo capítulo aprofundamos a análise de QGAME, descrevendo sua arquitetura e uso, salientando as alterações realizadas que levaram à extensão deste simulador.

6 Desenvolvimento e Resultados

Neste capítulo pretendemos demonstrar as atividades relacionadas com o desenvolvimento, que constituiu na expansão do simulador QGAME, com o objetivo de fazê-lo suportar a execução de algoritmos baseados em Caminhos Aleatórios Quânticos. Iniciamos com a apresentação de detalhes que tornam QGAME interessante para este domínio de problema, como a definição de portas quânticas como sendo oráculos. Finalizamos o capítulo discutindo, dentro desta arquitetura de *software*, quais foram as contribuições do trabalho, e os resultados do uso da versão expandida de QGAME na simulação do grafo cíclico de N vértices.

6.1 Características e Arquitetura de QGAME

QGAME compreende um simulador quântico que pode ser executado em um computador clássico. Para isso, QGAME define uma linguagem (que pode ser denominada linguagem de programação quântica) e um interpretador, que simula a execução de algoritmos ou programas escritos nesta linguagem. QGAME ainda oferece a capacidade de visualizar o processo de simulação passo-a-passo.

Como qualquer outra linguagem de programação, os programas em QGAME consistem em um conjunto de comandos ou instruções. Por QGAME ser desenvolvida em Common Lisp, estas instruções têm a mesma sintaxe que instruções em expressões simbólicas de Lisp. Cada instrução geralmente compreende o nome de uma porta quântica, seguida do "endereço" dos *Qbits* que sofreram efeito da aplicação desta porta. Por exemplo, se desejarmos aplicar a porta de NOT quântico ao *Qbit* de "endereço" 2, teremos o seguinte comando:

```
(qnot 2)
```

esta representação facilita a criação de novas portas quânticas implementadas como funções em Common Lisp. Por exemplo, a porta NOT acima é implementada através da seguinte função em Common Lisp (Algoritmo 6.1):

```
(defun qnot (qsys q)
  (apply-operator qsys
                  #2A((0 1)
                      (1 0))
                  (list q)))
```

Algoritmo 6.1: Porta quântica NOT como uma função em QGAME

neste caso, a função que implementa a porta NOT recebe dois argumentos: a instância de um sistema quântico criado e o índice do *Qbit* deste sistema quântico que sofrerá a ação da porta. A função então aplica o operador NOT (representado pela matriz quadrada; em QGAME, matrizes são representadas como *arrays*) aos *Qbits* do sistema quântico, especificados em uma lista de índices.

É interessante notar que a porta NOT e a função NOT possuem números diferentes de argumentos. Isso ocorre pelo fato de a porta NOT e a função NOT estarem em domínios diferentes de abstração. A porta NOT é parte de uma sub-linguagem definida encima de Common Lisp (uma linguagem de domínio específico), interpretada por uma função (neste caso, a função RUN-QSYS), que somente fará sentido para este interpretador. Porém, por Common Lisp conseguir representar dado como código-fonte e código-fonte como dado, a porta NOT pode ser muito bem implementada por uma função de Common Lisp, que recebe como argumento extra do interpretador, o sistema quântico que este estará operando.

Uma outra característica marcante de QGAME é a possibilidade de definirmos portas quânticas como sendo oráculos. Oráculos são caixas-preta no sentido de somente ser necessário especificarmos a matriz que mapeia as entradas em saídas. Na Seção 3.4 utilizamos este conceito para criar uma porta oráculo.

QGAME ainda dispõe de um conjunto de operadores auxiliares, como o operador MEASURE, que realiza a medida do estado do sistema quântico em execução. Outros operadores podem ser acrescentados a QGAME, bastando defini-los como funções de Common Lisp, e em raras exceções (como no caso do operador FOR que implementamos) haverá a necessidade de modificar o interpretador em si.

6.2 Expandindo QGAME

A linguagem de programação quântica disponível em QGAME consiste em um conjunto de portas quânticas e operadores para medida. Desta forma, estruturas de controle como laços de repetição – necessários para a simulação de algoritmos baseados em Caminhos Aleatórios Quânticos – não estão disponíveis por padrão. Também torna-se interessante a inclusão de procedimentos para a visualização facilitada das iterações do algoritmo. Assim, propomos a criação destes facilitadores.

Além disso, o *software* estava sendo distribuído sem um pacote de instalação, porém somente como um único arquivo de código-fonte. Realizamos o "empacotamento" de QGAME utilizando o sistema de definição ASDF[1], tornando sua distribuição e portabilidade facilitadas.

Esta seção revela informações conceituais sobre as expansões realizadas. Todos os detalhes sobre a implementação em *software* como um todo estão disponíveis na documentação do código-fonte em anexo ao trabalho.

6.2.1 Estruturas de Controle

No Capítulo 3 um programa em QGAME para uma versão simplificada do Caminho de Hadamard é apresentado. Neste programa já é possível notar a necessidade de uma estrutura de controle para um laço de repetição.

O procedimento FOR é então proposto. Sua especificação é dada por:

```
(for <limite inferior> <limite superior>
    <corpo a iterar>)
```

Assim, uma iteração como a necessária para o Caminho de Hadamard pode ser simplificada pelo Algoritmo 6.2:

```
(for 0 10
    (hadamard 0)
    (transicao 1 2 0))
```

Algoritmo 6.2: Iteração utilizando operador FOR em QGAME

[1] Another System Definition Facility: `http://www.cliki.net/asdf`

Neste caso, estaríamos iterando o Caminho de Hadamard uma quantidade $n = 10$ vezes. Outras estruturas de controle como WHILE e DO-WHILE também podem ser generalizados a partir do procedimento FOR.

A implementação do laço de repetição FOR se deu pela especificação de uma nova palavra-reservada na linguagem de domínio específico – que constitui a linguagem de programação quântica de QGAME – utilizada por QGAME para especificar os programas ou algoritmos quânticos a serem executados. O componente RUN-QSYS por sua vez foi modificado, permitindo o *parsing* desta nova palavra-reservada, e executando recursivamente o corpo do laço FOR tantas vezes for definida pelo valor da diferença do limite superior e inferior, passados como argumentos para o comando.

6.2.2 Visualização Facilitada e ASDF

QGAME oferece como forma de acompanhamento dos resultados gerados pelas simulações, o acesso textual a algumas informações básicas, como os valores do vetor de amplitudes. Implementamos procedimentos para facilitar o acesso à leitura das amplitudes, detalhadas no código-fonte de QGAME.

Também portamos QGAME para ser distribuído como um pacote ASDF, tornando-o multi-plataforma e de fácil instalação.

6.3 Resultados da Simulação para Grafos Cíclicos

Utilizando a mesma metodologia discutida na Seção 3.4, realizamos a simulação de grafos cíclicos, variando seu número de vértices N, buscando avaliar o desempenho do simulador QGAME. O Algoritmo 6.3 descreve o Caminho Aleatório Quântico discreto com moeda de Hadamard.

```
1 Partícula inicia na posição de origem x = 0;
2 Lançar moeda de Hadamard;
3 Mover partícula para direita ou esquerda em uma posição, dependendo do
    valor da moeda (operador swap);
4 Repetir os passos 2 e 3 t vezes;
5 Guardar a nova posição x da partícula;
6 Repetir os passos acima n vezes
```

Algoritmo 6.3: Caminho Aleatório Quântico discreto com mooeda de Hadamard

Modelamos este algoritmo através do Algoritmo 6.4 especificado em QGAME.

```
(for 1 n
   (for 1 t
      (hadamard 0)
      (sv 1 2 0))
   (printamps)))
```

Algoritmo 6.4: Caminho Aleatório Quântico discreto com mooeda de Hadamard em QGAME

E o executamos para um número n e t aleatórios, onde $100 < n, t < 10000$. A execução é dada pelo interpretador de QGAME através da invocação da função RUN-QSYS, como listado no Algoritmo 6.5.

```
(run-qsys (make-instance 'quantum-system
                         :number-of-qubits q
                         :program '((for 1 n
                                         (for 1 t
                                              (hadamard 0)
                                              (sv 1 2 0))
                                              (printamps)))))
```

Algoritmo 6.5: Execução da função RUN-QSYS para Caminho de Hadamard

O resultado das simulações para um grafo de $q > 3$ vértices, $t = 10$ iterações do caminhante e $n = 10000$ repetições do experimento, é resumida na Tabela 6.1.

Podemos notar que o método necessita de um *Qbit* a mais somente para preservar o estado da moeda quântica entre as iterações. Isto faz com que, para os valores entre os máximos da quantidade de vértices – como 4, 8, 16, 32 e assim por diante – exista uma perda de *Qbits*. Por exemplo, para 7 vértices, são desperdiçados 2 estados, ou um *Qbit*. Porém, isto torna-se necessário para a preservação do estado da moeda. Tomamos para os experimentos, somente quantidades que não desperdiçassem *Qbits*, aproveitando ao máximo os passos de execução do algoritmo. Também é possível notar que a quan-

Qbits	Vértices	Arestas	Números Complexos	Dimensão de S_v	Tempo real
3	4	8	64	8×8	4.174 *seg.*
4	8	16	256	16×16	4.705 *seg.*
5	16	32	1024	32×32	6.642 *seg.*
6	32	64	4096	64×64	14.291 *seg.*
7	64	128	16 384	128×128	52.503 *seg.*
8	128	256	65 536	256×256	273.302 ($4min.$)
9	256	512	262 144	512×512	1383.306 ($23min.$)
10	512	1024	1 048 576	1024×1024	11823.117 ($3.3hs.$)

Tabela 6.1: Tabela de resultado para simulações do Caminho Aleatório Quântico de Hadamard.

tidade de arestas é equivalente ao dobro da quantidade de vértices. Mais uma vez, isto se dá pela necessidade que temos em especificar tanto o passo do caminhante para seu vizinho com este mantendo a moeda em $|0\rangle$, quanto o passo que este realiza mantendo o estado da moeda em $|1\rangle$.

A quantidade de *Qbits* é obtida pela soma da quantidade de *Qbits* utilizados para representar os vértices, mais um *Qbit* utilizado para a moeda de Hadamard. A quantidade de *Qbits* cresce exponencialmente, conforme o número de vértices do grafo. É interessante notar um aumento exponencial na quantidade de números complexos necessários para simular cada grafo. Por exemplo, para um grafo de 32 vértices, precisamos de mais de 4 mil números complexos. Esta grande quantidade de números complexos influencia diretamente o tempo de execução do simulador, visto que a cada aplicação de um operador quântico, cada um destes números deverá ser acessado. Além disso, para a implementação do operador S_v, precisamos de matrizes de grau 2^{2n}, onde n é a quantidade de *Qbits* do sistema quântico. Por exemplo, para o mesmo grafo de 32 vértices, precisaríamos de uma matriz quadrada de grau 64. Isto representa uma complexidade espacial de ordem exponencial, visto que para um sistema de 10 *Qbits* (onde poderíamos simular um grafo de 512 vértices) precisaríamos de mais de 1 milhão de números complexos. Este número chegaria a mais de 1 bilhão para um sistema de 15 *Qbits*.

Portanto, esta complexidade espacial terá reflexo sobre a complexidade temporal dos algoritmos, que também revelou-se de ordem exponencial. O gráfico da Figura 6.1 permite visualizar a curva característica deste tipo de função.

Figura 6.1: Tempo de execução da simulação para N *Qbits*, com $t = 10000$ repetições do experimento.

Para a realização dos experimentos, foi utilizado um computador com processador de *duplo-core* de *2.0GHz*, com memória *RAM* de *1024MB*, em plataforma *GNU/-Linux*. Durante toda a execução das simulações, a carga de processamento manteve-se em 100%, e a utilização da memória *RAM* foi de 55% para uso do simulador e 44% permaneceu em *cache*.

7 Conclusões

A Computação Quântica demonstra-se uma área de pesquisa multidisciplinar, e portanto o estudo de conteúdos da Física e Matemática foi necessário. Pretendemos com este estudo ampliar o conteúdo deste trabalho abrangendo a discussão de tópicos necessários à compreensão dos Caminhos Aleatórios Quânticos. Tópicos estes pertencentes a Mecânica Quântica, e não comuns a estudantes de Ciência da Computação, e portanto demonstraram-se como um desafio.

A estratégia de concepção de algoritmos baseando-se em Caminhos Aleatórios Quânticos demonstrou-se aplicável ao grafo cíclico de N vértices, e sua simulação por um computador clássico também foi possível. O método de simulação utilizando a linguagem de programação quântica e simulador QGAME mostra-se aplicável ao problema proposto, como foi demonstrado através das simulações desenvolvidas no trabalho. A sua utilização permitiu compreender melhor a natureza dos algoritmos quânticos. Além disso, com a simulação em um computador clássico é possível executar tais sistemas quânticos sem a necessidade de sua implementação física, o que facilita seu estudo.

As principais contribuições realizadas por este trabalho podem ser definidas por:

1. *Introdução pedagógica à Computação Quântica.* Procuramos tornar o conteúdo acessível a leitores sem conhecimentos em Mecânica Quântica, fazendo deste trabalho uma introdução facilitada à Computação Quântica e a tópicos pouco abordados de maneira pedagógica, como os Caminhos Aleatórios Quânticos e as linguagens de programação quântica;

2. *Comparativo de linguagens de programação quântica.* Por ser uma área recente de pesquisa, com muitos protótipos de linguagens existentes para a prova de conceitos e, geralmente, sem pretensão prática, um comparativo entre as linguagens foi feito onde documentamos as principais vantagens que levaram a escolha de QGAME como linguagem e simulador de desenvolvimento e estudo para este trabalho;

3. *Extensão da linguagem e simulador quântico QGAME.* Desenvolvemos modificações

em QGAME tanto operacionais quanto em sua base. A linguagem foi estendida com o adendo do procedimento de controle FOR, para realização de iterações de um determinado corpo de código. O autor de QGAME, Lee Spector, foi contactado para as publicações das alterações no *software*.

Pelo método ter-se demonstrado de interesse à aplicação em simulações de Caminhos Aleatórios Quânticos, temos como expectativa futura o desenvolvimento de QGAME, buscando aplicá-lo em problemas avançados de grafos – e que continuam em aberto, como Caminhos Aleatórios Quânticos aplicados a grupos simétricos ou na abordagem de questões referentes a isomorfismo de grafos – não ficando restrito aos grafos cíclicos abordados neste trabalho. O estudo e desenvolvimento de QGAME – ou mesmo QLisp, ou outra linguagem que possa vir a se revelar como bom modelo computacional para a especificação de algoritmos quânticos – como uma linguagem de programação quântica completa, e não somente como um simulador de circuitos quânticos, mas sim uma linguagem que permita a codificação de programas em alto nível de abstração, é de grande interesse para trabalhos futuros. Com uma linguagem de programação quântica de alto nível poderíamos, assim como fazemos com as linguagens clássicas, nos preocupar somente com a solução de um problema, abstraindo detalhes do *hardware* quântico que implementa tal solução, por exemplo.

Referências Bibliográficas

[Aharonov et al. 2001]AHARONOV, D. et al. Quantum walks on graphs. In: *Proc. 33th STOC*. New York, NY: ACM, 2001. p. 50–59.

[Aharonov, Davidovich e Zagury 1993]AHARONOV, Y.; DAVIDOVICH, L.; ZAGURY, N. Quantum random walks. *Physical Review A*, v. 48, n. 2, 1993.

[Ambainis 2004]AMBAINIS, A. Quantum walk algorithm for element distinctness. In: *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04)*. Washington, DC, USA: IEEE Computer Society, 2004. p. 22–31. ISBN 0-7695-2228-9.

[Ambainis et al. 2001]AMBAINIS, A. et al. One-dimensional quantum walks. In: *ACM Symposium on Theory of Computing*. [S.l.: s.n.], 2001. p. 37–49.

[Ambainis, Kempe e Rivosh 2005]AMBAINIS, A.; KEMPE, J.; RIVOSH, A. Coins make quantum walks faster. In: *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2005. p. 1099–1108. ISBN 0-89871-585-7.

[Chartrand 1984]CHARTRAND, G. *Introductory Graph Theory*. [S.l.]: Dover Publications, 1984.

[Childs et al. 2003]CHILDS, A. M. et al. Exponential algorithmic speedup by a quantum walk. In: *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM Press, 2003. p. 59–68. ISBN 1-58113-674-9. ArXiv:quant-ph/0209131.

[Coppersmith 1994]COPPERSMITH, D. *An approximate Fourier transform useful in quantum factoring*. 1994. Arxiv.org/pdf/quant-ph/0201067.

[Costa 2005]COSTA, J. K. V. A. C. da R. Concurrent quantum programming in haskell. In: . [S.l.: s.n.], 2005.

[Desmet 2005]DESMET, B. *Simulation of Quantum Computations and Applications*. Tese (Licenciate Thesis) — Programming Technology Lab, Vrije Universiteit Brussel, Jun 2005.

[Deutsch 1985]DEUTSCH, D. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, v. 400, 1985.

[Farhi e Gutmann 1998]FARHI, E.; GUTMANN, S. Quantum computation and decision trees. *Phys. Rev. A*, American Physical Society, v. 58, n. 2, p. 915–928, Aug 1998.

[Feynman 1982]FEYNMAN, R. Simulating physics with computers. *International Journal of Theoretical Physics*, v. 21, 1982.

[Gay 2006]GAY, S. J. Quantum programming languages: Survey and bibliography. *Mathematical Structures in Computer Science*, v. 16, p. 581–600, 2006.

[Grattage 2005]GRATTAGE, T. A. J. A functional quantum programming language. In: . [S.l.: s.n.], 2005. Arxiv.org/pdf/quant-ph/0409065.

[Grattage 2005]GRATTAGE, T. A. J. *QML: Quantum data and control*. 2005. Manuscript.

[Grover 1996]GROVER, L. A fast quantum mechanical algorithm for database search. *Proceedings of 28th Annual ACM Symposium on Theory of Computing*, p. 212–219, 1996. Arxiv.org/pdf/quant-ph/9605043.

[Kempe 2002]KEMPE, J. Quantum random walks hit exponentially faster. *Probability Theory and Related Fields*, v. 133(2), p. 215–235, may 2002.

[Kempe 2003]KEMPE, J. Quantum random walks - an introductory overview. *Contemporary Physics*, v. 44, 2003.

[Kendon 2006]KENDON, V. M. A random walk approach to quantum algorithms. *Phil. Trans. R. Soc. A*, v. 364, p. 3407–3422, 2006. Arxiv.org/pdf/quant-ph/0609035.

[Knill 1996]KNILL, E. *Conventions for quantum pseudocode*. 1996. Disponível em: <citeseer.ist.psu.edu/knill96conventions.html>.

[Mermin 2003]MERMIN, N. D. From cbits to qbits: Teaching computer scientists quantum mechanics. *American Journal of Physics*, v. 71, p. 23–30, 2003.

[Motwani e Raghavan 1995]MOTWANI, R.; RAGHAVAN, P. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

[Nielsen e Chuang 2000]NIELSEN, M. A.; CHUANG, I. L. *Quantum Computation and Quantum Information*. Cambridge, UK: Cambridge University Press, 2000.

[Norvig 2003]NORVIG, S. J. R. P. *Artificial Intelligence: A Modern Approach*. [S.l.]: Prentice Hall, 2003.

[Oemer 1998]OEMER, B. *A Procedural Formalism for Quantum Computing*. Master Thesis — Department of Theoretical Physics, Technical University of Vienna, 1998.

[Oemer 2000]OEMER, B. *Quantum Programming in QCL*. Master Thesis — Institute of Information Systems, 2000.

[Oemer 2001]OEMER, B. Procedural quantum programming. *Computing Anticipatory Systems: CASYS 2001, Fifth International Conference*, v. 627, 2001.

[Oemer 2002]OEMER, B. *Classical Concepts in Quantum Programming*. 2002. ArXiv.org:quant-ph/0211100.

[Oemer 2003]OEMER, B. *Structured Quantum Programming*. Tese (Doutorado) — Technical University of Vienna, 2003.

[Oliveira 2007]OLIVEIRA, A. C. *Simulação de Caminhos Quânticos em Redes Bidimensionais*. Tese (Doutorado) — Laboratório Nacional de Computação Científica, 2007.

[Reingold 2005]REINGOLD, O. Undirected st-connectivity in log-space. In: . [S.l.: s.n.], 2005.

[Sabry 2003]SABRY, A. Modelling quantum computing in haskell. In: . [S.l.: s.n.], 2003. p. 39–49.

[Sanders e Zuliani 2000]SANDERS, J. W.; ZULIANI, P. Quantum programming. In: *Mathematics of Program Construction*. [s.n.], 2000. p. 80–99. Disponível em: <citeseer.ist.psu.edu/article/sanders99quantum.html>.

[Schöning 1999]SCHÖNING, U. A probabilistic algorithm for k-sat and constraint satisfaction problems. In: IEEE. *40th Ann. Symp. on Foudantions of Computer Science*. [S.l.], 1999. p. 410–414.

[Selinger 2004]SELINGER, P. A brief survey of quantum programming languages. In: . [S.l.: s.n.], 2004.

[Selinger 2007]SELINGER, P. Dagger compact closed categories and completely positive maps. In: . [S.l.: s.n.], 2007.

[Serafini 2003]SERAFINI, S. B. T. Calarco L. Toward an architecture for quantum programming. *The European Physical Journal*, v. 25, n. 2, p. 181–200, 2003.

[Shenvi, Kempe e Whaley 2003]SHENVI, N.; KEMPE, J.; WHALEY, K. B. Quantum random-walk search algorithm. *Phys. Rev. A*, American Physical Society, v. 67, n. 5, p. 052307, May 2003.

[Shor 1994]SHOR, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In: . [S.l.: s.n.], 1994.

[Sipser 1997]SIPSER, M. *Introduction to the Theory of Computation*. [S.l.]: PWS Publishing Company, 1997.

[Spector 2004]SPECTOR, L. *Automatic Quantum Computer Programming - A Genetic Programming Approach*. [S.l.]: Springer, 2004.

[Tonder 2004]TONDER, A. van. *Quantum computation, categorical semantics and linear logic*. 2004. Arxiv.org/pdf/quant-ph/0312174.

[Turing 1936]TURING, A. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, v. 42, 1936.

[Valiron 2004]VALIRON, B. *A functional programming language for quantum computation with classical control*. Master Thesis — University of Ottawa, 2004.

[West 2001]WEST, D. B. *Introduction to Graph Theory*. Upper Saddle River: Prentice Hall, 2001.

[Wolfram 2002]WOLFRAM, S. *A New Kind of Science*. [S.l.]: Wolfram Media, 2002.

[Yellen 1998]YELLEN, J. G. J. *Graph Theory and Its Applications*. [S.l.]: CRC Press, 1998.

[Zuliani 2001]ZULIANI, P. *Quantum Programming*. Tese (Doutorado) — University of Oxford, 2001.