2
Fundamentos

2.1
Mecânica Quântica

2.1.1
Introdução

Quando a luz incide sobre uma superfície de metal, alguns elétrons são arrancados desta mesma superfície. Este fenômeno não é satisfatoriamente explicado pela mecânica clássica, devido a três fatores [18]:

- Não importa o quão baixa é a intensidade da luz, os elétrons são arrancados instantaneamente da superfície do metal. De acordo com a física clássica, os elétrons precisariam acumular, durante um certo tempo, a energia necessária para escapar da superfície do metal, já que a luz deveria ter sua energia distribuída uniformemente ao longo da onda.
- Existe uma frequência de corte para o feixe incidente de luz, abaixo da qual nenhum elétron é emitido. De acordo com as leis de Maxwell [19], a energia transportada pela luz depende apenas da amplitude da onda eletromagnética e não da frequência.
- Se a energia cinética máxima dos elétrons emitidos for relacionada através de um gráfico com a frequência da luz incidente, uma linha reta será obtida. Como a energia deveria estar relacionada apenas com a amplitude da onda, de acordo com as leis de Maxwell, não deveria existir uma relação linear entre frequência e energia.

A explicação para estes fenômenos foi proposta em 1900 por Planck [20] ao abandonar o conceito clássico para o comportamento da luz apenas como uma onda eletromagnética [18]. Planck demonstrou que, se a energia da luz estiver concentrada em pacotes (ou *quanta*) de energia $E = h\nu$, onde ν é a freqüência da luz e h é a constante de Planck, os problemas descritos anteriormente podem ser, então, facilmente explicados. Através desta explicação, a luz passa a ter, em determinadas situações, um comportamento de uma partícula, denominada fóton.

2.1.2 Funções de Onda

A ideia de que a luz se comporta em alguns casos como uma onda e em alguns casos como uma partícula leva imediatamente à questão sobre se o fóton tem outras propriedades comuns às partículas. Como energia e massa estão relacionadas pela equação $E = mc^2$, o fóton, que se move à velocidade da luz c, tem uma massa relativística $m = h\nu/c^2$. Portanto, como o fóton tem massa e velocidade, também deve ter um momento $p = mc = h\nu/c = h/\lambda$, onde λ é o comprimento de onda da luz.

De maneira inversa, assim como os fótons têm um momento característico h/λ, outras partículas (como, por exemplo, elétron), que também têm um momento característico, devem possuir um comprimento de onda relacionado a este momento $\lambda = h/p$. Se uma partícula tem um comprimento de onda característico, então deve ser possível representar este comprimento de onda matematicamente da mesma maneira que uma onda é representada, ou seja, usando-se uma função de onda. Na física clássica, uma onda estacionária com comprimento de onda λ, propagando-se na direção positiva do eixo x, pode ser representada por

$$\psi(x) = \exp(i2\pi x/\lambda) = \cos(2\pi x/\lambda) + i\sin(2\pi x/\lambda), \tag{2-1}$$

onde $i = \sqrt{-1}$. Substituindo $\lambda = h/p$, obtém-se

$$\psi(x) = \exp(ipx/\hbar), \tag{2-2}$$

onde $\hbar = h/2\pi$ ($\hbar$: constante de Planck reduzida) [18].

A partir da função de onda, é possível determinar a probabilidade de que a partícula esteja em um determinado ponto do espaço ao se tentar observá-la. A densidade de probabilidade para a localização de uma partícula com função de onda ψ deve ser o quadrado da amplitude da função de onda, ou seja, $|\psi|^2$ [21, 18]. Em uma dimensão, uma partícula, representada pela função de onda $\psi(x)$, terá uma densidade de probabilidade de ser encontrada no intervalo $x + dx$ igual a:

$$|\psi(x)|^2 dx = \psi^*\psi dx, \tag{2-3}$$

onde ψ^* representa o complexo conjugado de ψ. Em três dimensões, a densidade de probabilidade da partícula é $|\psi(\mathbf{r})|^2$, onde $\mathbf{r}$ é um vetor de três dimensões (x,y,z). Isto significa que a probabilidade de se encontrar a partícula no elemento infinitesimal de volume $d\tau = dxdydz$ no ponto $\mathbf{r}$ é $|\psi(\mathbf{r})|^2 d\tau$. A integral deste valor sobre todo o espaço onde a partícula pode ser encontrada fornece a probabilidade de se encontrar esta partícula em algum lugar do espaço e, consequentemente, este valor deve ser igual a 1 (equação 2-4).

$$\int_{-\infty}^{\infty} |\psi|^2 d\tau = 1. \tag{2-4}$$

As funções de onda introduzem um conceito importante e pouco intuitivo: a relação entre a função de onda e a localização da partícula é, por natureza, probabilística [18]. Em um feixe de luz, por exemplo, a função de onda está espalhada por toda a frente de onda de forma homogênea mas, quando se tenta detectar os fótons, nota-se que os mesmos chegam de forma imprevisível ao detector, com uma densidade de probabilidade proporcional à intensidade do feixe de luz ou proporcional ao quadrado da amplitude da onda. Em outras palavras, o valor das propriedades da partícula que podem ser representadas por funções de onda (como, por exemplo, a posição da partícula) não pode ser conhecido *a priori*. Sem efetivamente medir a propriedade que se deseja, tudo que se pode afirmar sobre esta propriedade é que ela tem diferentes probabilidades de assumir diferentes valores quando for observada. Assim, antes de se observar através de algum instrumento, por exemplo, a posição de um elétron confinado dentro de alguma região do espaço, o máximo que se pode determinar é a probabilidade de observá-lo em alguma região deste espaço de confinamento.

As funções de onda devem obedecer as seguintes propriedades [18]:

1. ψ deve ser finita;
2. ψ deve ser contínua;
3. ψ deve ser derivável duas vezes;
4. ψ deve ser integrável em todo o espaço.

2.1.3 Operadores Hermitianos

As informações contidas nas funções de onda são as únicas de que se pode dispor a respeito de um sistema físico de dimensões nanométricas[1] (como, por exemplo, partículas subatômicas, átomos e moléculas). Como visto na seção 2.1.2, é possível manipular estas funções de onda de modo a se obter a distribuição de probabilidade espacial do sistema físico representado por esta função. No entanto, esta informação espacial não é a única que se deseja obter sobre o sistema físico. Deseja-se também ser capaz de medir quantidades, tais como, por exemplo, o momento, a energia e o momento angular de uma partícula. Apesar da função de onda não poder ser medida diretamente, todas essas propriedades mecânicas podem ser medidas diretamente e são chamadas, na mecânica quântica, de "observáveis"

[1] 1 nanômetro = 1 metro $\times 10^{-9}$

[21]. Os observáveis são representados matematicamente por "operadores". Alguns exemplos de observáveis são a posição e o momento de uma partícula.

Um operador é um objeto matemático que age sobre uma função transformando-a em outra função. Um exemplo é o operador $\hat{\partial}_x$, que realiza a operação de diferenciar com respeito à variável x. Este operador transforma uma função $f(x)$ em sua derivada.

Apesar de, em geral, os operadores transformarem uma função em outra, em alguns casos especiais um operador pode transformar uma função nela mesma. Um exemplo é o operador $\hat{\partial}_x$ que transforma a função $e^{\alpha x}$ em $\alpha e^{\alpha x}$, ou seja, a mesma função multiplicada por uma constante α. A função que é transformada nela mesma é chamada de autofunção (ou autovetor) e a constante multiplicativa introduzida pelo operador é chamada de autovalor.

Os conceitos de autofunção e autovalor têm importância primordial na mecânica quântica. Isto se deve ao fato de que todos os observáveis são representados por operadores, sendo que os autovalores, os quais podem ser encontrados por meio da aplicação de um operador a uma função, são os únicos valores que podem efetivamente ser observados ao se fazer uma medida. Em outras palavras, qualquer medição de um observável deve encontrar um dos possíveis autovalores do operador [21]. Por exemplo, os níveis de energia de um átomo são autovalores do operador de energia. Ou seja, qualquer medida realizada para se determinar o nível de energia de um átomo deve encontrar um desses autovalores.

Os operadores que representam quantidades observáveis devem respeitar duas restrições: linearidade e hermiticidade. Um operador $\hat{A}$ é linear quando, para quaisquer valores constantes α e β, e para quaisquer funções de onda $\psi(x)$ e $\phi(x)$, a equação 2-5 for válida, tal que

$$\hat{A}(\alpha\psi(x) + \beta\phi(x)) = \alpha\hat{A}\psi(x) + \beta\hat{A}\phi(x). \tag{2-5}$$

Um operador $\hat{A}$ é hermitiano se, para quaisquer funções de onda $\psi(x)$ e $\phi(x)$, a equação 2-6 for válida, tal que

$$\int_{-\infty}^{\infty} \psi^*(x)\hat{A}\phi(x)dx = \int_{-\infty}^{\infty} \hat{A}\psi^*(x)\phi(x)dx. \tag{2-6}$$

Duas importantes consequências advêm destas restrições: a primeira é que os autovalores de um operador hermitiano são números reais; a segunda é que as autofunções de um operador hermitiano, correspondentes a autovalores diferentes, são ortogonais entre si [21].

Supondo-se então que um observável $\mathcal{A}$ é representado por um operador $\hat{A}$ e que uma função de onda ψ é uma autofunção de $\hat{A}$ com autovalor α (ou seja, $\hat{A}\psi = \alpha\psi$), então o sistema tem um valor definido α para o observável $\mathcal{A}$. Uma medida de $\mathcal{A}$ irá resultar no valor α com probabilidade 1. Por outro lado,

se a função de onda ψ não for uma autofunção de $\hat{A}$, então uma medida de $\mathcal{A}$ pode produzir qualquer um de diferentes autovalores do operador $\hat{A}$ com diferentes probabilidades. O valor esperado de um observável $\mathcal{A}$, denotado por $\langle A \rangle$, é definido pela equação 2-7,

$$\langle A \rangle = \int \psi^* \hat{A} \psi d\tau. \tag{2-7}$$

De modo a simplificar a notação matemática, pode-se usar a notação de Dirac para estes problemas [18]. Neste caso, a função de onda ψ passa a ser representada pelo símbolo $|\psi\rangle$, chamado "ket". O conjugado complexo ψ^* da função de onda é representado pelo símbolo $\langle\psi|$, chamado "bra". As integrais do tipo $\int \phi^* \psi d\tau$ são representadas pelo símbolo $\langle\phi|\psi\rangle$. É importante notar que este símbolo implica na integração da função, ou seja, $\langle\phi|\psi\rangle = \int \phi^* \psi d\tau$. Portanto, uma função de onda ψ está normalizada se $\langle\psi|\psi\rangle = 1$.

A principal característica que se deseja ressaltar, com relação às funções de onda e aos operadores, é a possibilidade de se representar as funções de onda como uma expansão por autofunções. Em outras palavras, além de serem ortogonais, as autofunções de um operador hermitiano formam uma base ortonormal completa que permite escrever qualquer função de onda para o sistema físico como uma superposição (combinação linear) destas autofunções. Ou seja, se um operador $\hat{A}$ possui um conjunto de autofunções $|\psi_i\rangle$ com autovalores α_i (ou seja, $\hat{A}|\psi_i\rangle = \alpha_i |\psi_i\rangle$), então qualquer função de onda ϕ válida para o sistema pode ser representada pela superposição de autofunções $|\psi_i\rangle$ conforme mostrado na equação 2-8, onde $c_i^* c_i$ (ou $|c_i|^2$) é a probabilidade de se medir o autovalor α_i, onde

$$\phi = \sum_i c_i |\psi_i\rangle. \tag{2-8}$$

Isto significa que, ao se fazer uma medida de um estado quântico, a função de onda é projetada em uma das autofunções que formam a base ortonormal e o autovalor correspondente a esta autofunção é o valor observado. Em outras palavras, a função de onda pode ser escrita como uma combinação linear dos autovetores associados a um determinado operador. Os coeficientes associados aos autovetores nesta combinação linear permitem determinar a probabilidade de que a função de onda seja projetada sobre cada um dos autovetores quando a observação for realizada. Vale ressaltar que, como $|c_i|^2$ representa a probabilidade de um autovalor ser observado, e como a base ortonormal é completa, então $\sum_i |c_i|^2 = 1$.

2.2 Computação Quântica

Um computador quântico é um dispositivo que faz uso direto de certos fenômenos da mecânica quântica para realizar operações com dados. Estes fenômenos

permitem construir, em teoria, computadores que obedecem novas leis mais permissivas de complexidade computacional [22]. O termo "computação quântica" é, portanto, usado para descrever processos computacionais que se baseiam nestes fenômenos específicos e que são, portanto, capazes de diminuir o esforço e a complexidade para se resolver determinados problemas. A principal perspectiva que se abre com o uso de computadores quânticos em relação ao seu poder de processamento é o fato de que "as possibilidades contam, mesmo que elas nunca venham a acontecer" [22].

Em um computador clássico, o *bit* é a menor unidade de informação, podendo assumir os valores 0 ou 1. Em um computador quântico, a unidade de informação básica, chamada de *qubit*, pode assumir os estados $|0\rangle$, $|1\rangle$ ou uma superposição dos dois estados. Esta superposição de dois estados é uma combinação linear dos estados $|0\rangle$ e $|1\rangle$ e pode ser representada por

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2-9}$$

onde $|\psi\rangle$ é o estado do *qubit* e, conforme discutido na seção 2.1.3, $|\alpha|^2$ e $|\beta|^2$ são as probabilidades de que os estados 0 ou 1 acarretem da observação do *qubit*.

Ao ser observado, o *bit* quântico será trazido para o nível clássico e o estado observado será o valor 0 ou 1. Esta superposição de estados oferece aos computadores quânticos um grau de paralelismo incomparável que, se devidamente explorado, permite que estes computadores realizem tarefas impraticáveis em computadores clássicos, no que diz respeito ao tempo demandado de processamento. Um exemplo de uma tarefa deste tipo é a fatoração inteira de grandes números. Neste problema, deseja-se encontrar os dois números primos cujo produto seja igual a um número x fornecido. Esta tarefa pode levar milhões de anos para ser resolvida em computadores clássicos usando os algoritmos mais sofisticados conhecidos. Em um computador quântico, no entanto, a mesma tarefa levaria apenas alguns segundos para ser concluída [23].

A dinâmica de um sistema quântico, quando não está sendo medido, é governada pela equação de Schrödinger [18], cuja formulação independente do tempo, para uma partícula em uma dimensão, é mostrada a seguir:

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi(x)}{\partial x^2} + V(x)\psi(x) = E\psi(x), \tag{2-10}$$

onde $\hbar$ é a constante de Planck reduzida, m é a massa da partícula, $\psi(x)$ é a função de onda estacionária da partícula, $V(x)$ determina o valor da energia potencial em função da posição e E é a energia da partícula.

Sendo assim, esta dinâmica deve levar o sistema de um estado para outro de uma forma que preserve a ortogonalidade. Para um espaço vetorial complexo, transformações lineares que preservam a ortogonalidade são transformações unitárias. Por sua vez, qualquer transformação unitária de um espaço de estados quânticos é

uma transformação quântica legítima e vice versa. Ou seja, pode-se encarar as transformações como sendo rotações de um vetor complexo. Na computação quântica, os conjuntos de transformações primitivas de estados quânticos são denominados "portas quânticas". Para computadores clássicos existem conjuntos de portas lógicas que são universais, no sentido que qualquer computação clássica pode ser efetuada usando-se uma combinação dessas portas. De forma similar, existem portas quânticas universais para a computação quântica.

Uma discussão sobre o funcionamento básico dos computadores quânticos pode ser encontrada em [24].

2.2.1 Sistemas Quânticos Multiníveis

De fato, a maioria das abordagens da computação quântica emprega *qubits* codificados em sistemas quânticos de dois níveis. No entanto, os sistemas candidatos à codificação de informação quântica normalmente têm uma estrutura física muito mais complexa, com diversos graus de liberdade diretamente acessíveis [25], tais como átomos [26], íons [27], ou fótons [28].

Recentemente, começou-se a considerar sistemas quânticos de d níveis, com relação à generalização e melhoria de protocolos e algoritmos quânticos baseados em *qubits* [29]. Estes sistemas quânticos utilizam o *qudit* como unidade de informação, o qual pode assumir qualquer um dos d valores, ou uma superposição dos d estados. Surgiram resultados interessantes no caso particular do *qutrit* ($d = 3$). Por exemplo, tem sido demonstrado que os protocolos de criptografia quântica são menos vulneráveis a ataques de terceiros quando são utilizados *qutrits* [30]. Especula-se também sobre a possibilidade da computação quântica com *qutrits* ser mais tolerante a falhas [31].

Apesar de ainda não terem sido construídos computadores quânticos funcionais baseados em *qubits*, em [29] é proposto um *design* de computador quântico com *qutrits*, utilizando armadilha de íons na presença de um gradiente de campo magnético.

2.3 Algoritmos com Inspiração Quântica

Embora a computação quântica ofereça uma boa promessa em termos de capacidade de processamento, atualmente dois problemas a impedem que se torne uma ferramenta útil: a dificuldade de se implementar um computador quântico e a dificuldade de se criar algoritmos que tirem proveito da capacidade de processamento destes computadores. Deste modo, as pesquisas na área de computação quântica se concentram, atualmente, em dois pontos:

– Desenvolvimento de novos algoritmos que sejam mais eficientes nos computadores quânticos do que os algoritmos equivalentes para computadores clássicos.
– Desenvolvimento de um *hardware* que torne viável o uso dos computadores quânticos.

No entanto, em [32] foi proposta uma nova abordagem. Ao invés de se desenvolver novos algoritmos para computadores quânticos ou de se tentar viabilizar o uso destes, a ideia de "computação com inspiração quântica" é apresentada. O objetivo desta nova abordagem é criar algoritmos clássicos, ou seja, executáveis em computadores clássicos, que tirem proveito dos paradigmas da física quântica, de forma a melhorar o desempenho dos mesmos na resolução de problemas.

Um exemplo de um algoritmo com inspiração quântica é o Algoritmo de Ordenação com Inspiração Quântica [32], baseado no Algoritmo de Ordenação por Seleção. Este algoritmo é usado para resolver o problema de ordenação numérica onde se deseja ordenar uma lista L_1 com n elementos em ordem ascendente ou descendente. Quando implementado de maneira eficiente, o tempo de execução deste algoritmo é da ordem de $\frac{1}{2}N\left(N-1\right)$ [33], onde N é o tamanho inicial de L_1. Por exemplo, caso se queira ordenar uma lista com 16 valores, serão necessários, em média, 120 buscas à lista para se encontrar o menor elemento. Usando elementos inspirados na física quântica, este algoritmo pôde ser melhorado de forma que o tempo de execução do novo algoritmo resultante é da ordem de $\sqrt{N}\left[N+2\left(\sqrt{N}-1\right)\right]$ [32]. Portanto, uma lista com 16 elementos precisará, em média, de 88 buscas na lista usando-se a ordenação por seleção, o que é um número significativamente menor de buscas que as 120 necessárias para a execução do algoritmo sem inspiração quântica.

Esta abordagem de computação com inspiração quântica foi aplicada em diversos modelos tradicionais de Inteligência Computacional, tais como em mapas auto-organizados (*self-organizing maps* – SOM) [34], em otimização por algoritmos imunológicos clonais [35], em otimização multiobjetivo por enxame de partículas [36], em evolução diferencial para otimização binária [37], dentre outros.

Quanto à Computação Evolutiva, área à qual o modelo proposto nesta tese pertence, tal abordagem culminou nos, assim denominados, "algoritmos evolutivos com inspiração quântica" (AEIQs). O AEIQ usando representação binária inicialmente proposto em [38] é um exemplo deste tipo de algoritmo e será descrito na seção 2.3.1. Outro exemplo de AEIQ é o algoritmo evolutivo para otimização numérica proposto em [15], o qual é inspirado no princípio de múltiplos universos da física quântica. Este AEIQ possui representação por números reais e apresenta um tempo de convergência menor para problemas *benchmark* quando comparado com algoritmos convencionais [16].

2.3.1
Algoritmo Evolutivo com Inspiração Quântica e Representação Binária

O AEIQ com representação binária (*Quantum-Inspired Evolutionary Algorithm* – QEA) foi proposto inicialmente em [38]. Neste modelo, o algoritmo evolutivo proposto também é caracterizado por um cromossomo, uma função de avaliação e uma dinâmica populacional. Entretanto, ao invés de uma representação binária convencional, este algoritmo usa uma representação especial que simula um cromossomo formado por *qubits*. Esta representação é feita da seguinte forma: um *qubit* é formado por um par de números (α,β), como um vetor

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \tag{2-11}$$

onde $|\alpha|^2 + |\beta|^2 = 1$. O valor dado por $|\alpha|^2$ e o valor dado por $|\beta|^2$ indicam as probabilidades de que o *qubit* terá os valores 0 e 1, respectivamente, quando for observado. Pode-se visualizar graficamente esta relação entre α e β, conforme mostrado na figura 2.1.

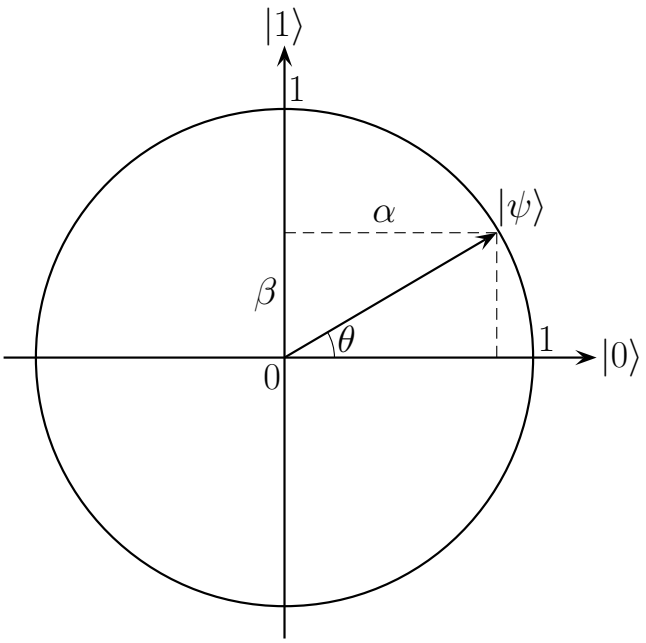

Figura 2.1: Representação gráfica das probabilidades de se observar os valores 0 e 1 para um *qubit* qualquer.

A partir da definição do *qubit* em 2-11, pode-se definir um indivíduo quântico formado por m *qubits*, na forma

$$\left[\begin{array}{c|c|c|c} \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \end{array}\right], \tag{2-12}$$

onde $|\alpha_i|^2 + |\beta_i|^2 = 1$ e $i = 1,2,3,\dots,m$. Deste modo, cada indivíduo quântico representa uma superposição de indivíduos formados por m genes. Seja, por exemplo,

um indivíduo quântico formado por 3 *qubits* com os seguintes valores:

$$\left[\begin{array}{c|c|c} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} \end{array}\right] \tag{2-13}$$

Para se calcular a "amplitude de probabilidade" (o módulo da amplitude de probabilidade ao quadrado é igual à probabilidade se observar o estado) do estado $|000\rangle$, multiplica-se as amplitudes de probabilidade de se observar os estados 0 em cada um dos bits (α_1, α_2 e α_3). Para se calcular a amplitude de probabilidade de se observar o estado $|001\rangle$, multiplica-se as amplitudes de probabilidade de se observar estes bits (α_1, α_2 e β_3). Estendendo-se isto para as outras possibilidades de observação dos estados, a superposição dos mesmos pode ser representada pela equação 2-14, tal que

$$\frac{1}{4}|000\rangle+\frac{\sqrt{3}}{4}|001\rangle+\frac{1}{4}|010\rangle+\frac{\sqrt{3}}{4}|011\rangle+\frac{1}{4}|100\rangle+\frac{\sqrt{3}}{4}|101\rangle+\frac{1}{4}|110\rangle+\frac{\sqrt{3}}{4}|111\rangle . \tag{2-14}$$

Este resultado significa que cada um dos possíveis estados deste indivíduo tem a probabilidade de ser observado segundo os valores listados na tabela 2.1.

Estado	Probabilidade
000	$\frac{1}{16}$
001	$\frac{3}{16}$
010	$\frac{1}{16}$
011	$\frac{3}{16}$
100	$\frac{1}{16}$
101	$\frac{3}{16}$
110	$\frac{1}{16}$
111	$\frac{3}{16}$

Tabela 2.1: Probabilidades de observação de cada um dos possíveis estados do indivíduo quântico.

O AEIQ proposto em [38, 39] é definido como no pseudocódigo exibido na figura 2.2, onde $Q(t)$ é uma população com um ou mais indivíduos quânticos. No passo de iniciação, estes indivíduos são iniciados de modo que os valores α_i e β_i (onde $i = 1,2,3,\ldots,j$ e j é o comprimento do indivíduo quântico) sejam iguais a $\frac{1}{\sqrt{2}}$. Isto significa que, na geração inicial, todos os genes dos indivíduos terão a mesma probabilidade de gerarem os estados 0 ou 1.

A atualização da população, conforme indicado na figura 2.2, é realizada pelo operador *q-gate*, o qual é definido pela seguinte matriz de rotação:

$$U(\Delta\theta_i) = \left[\begin{array}{cc} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{array}\right]. \tag{2-15}$$

Esta matriz deve ser usada para multiplicar cada uma das colunas do indivíduo quântico, ou seja, os valores α_i e β_i são tratados como um vetor bidimensional e

$t \leftarrow 0$
inicia $Q(t)$
gera $P(t)$ observando estados de $Q(t)$
avalia $P(t)$
armazena as melhores soluções de $P(t)$ em $B(t)$
enquanto *não ocorrer condição de parada* **faça**
    $t \leftarrow t + 1$
    gera $P(t)$ observando estados de $Q(t-1)$
    avalia $P(t)$
    atualiza $Q(t)$ aplicando *q-gate*
    armazena as melhores soluções dentre $B(t-1)$ e $P(t)$ em $B(t)$
    armazena a melhor solução b de $B(t)$
fim

Figura 2.2: Pseudocódigo do algoritmo evolutivo com inspiração quântica e representação binária QEA.

sofrem rotação usando-se a matriz especificada. Em outras palavras, considerando-se o indivíduo quântico $Q(t) = \{(\alpha_1,\beta_1), (\alpha_2,\beta_2), \ldots, (\alpha_j,\beta_j)\}$, a atualização deste indivíduo é realizada de acordo com a equação 2-16.

$$(\alpha_i',\beta_i') = U(\Delta\theta_i) \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}. \tag{2-16}$$

Os valores de $\Delta\theta$ são definidos através de uma tabela, de modo que esta matriz de rotação seja capaz de modificar os valores de α_i e β_i aumentando as chances de que os indivíduos com as melhores avaliações sejam observados.

Finalmente, a estrutura $B(t)$ é usada para armazenar os melhores indivíduos gerados pelo algoritmo ao longo do processo evolutivo. Os dois últimos passos do algoritmo servem para armazenar os melhores indivíduos gerados na população atual com os melhores indivíduos criados nas gerações anteriores.

O QEA foi utilizado com sucesso em problemas de otimização combinatória [38, 13] e detecção de faces [40], apresentando resultados superiores aos AGs convencionais em termos de tempo de convergência e qualidade das soluções encontradas. Uma versão modificada do QEA foi utilizada também em problemas de otimização combinatória com múltiplos objetivos [41].

2.4 Programação Genética

O objetivo de se ter computadores resolvendo problemas automaticamente é central nas áreas da Inteligência Artificial, do Aprendizado de Máquina e na ampla área compreendida pelo que Alan Turing chamou de "Inteligência de Máquina"

(*Machine Intelligence*) [42]. Portanto, uma vez que este cientista pioneiro da computação já vislumbrava tal possibilidade no final da década de 1940, pode-se dizer que a idéia de se evoluir programas de computador é quase tão antiga quanto o próprio computador em si [8]. Arthur Samuel, pioneiro do Aprendizado de Máquina, em sua palestra de 1983, intitulada *AI: Where It Has Been and Where It Is Going* [43], declarou que o principal objetivo do Aprendizado de Máquina e da Inteligência Artificial é:

> "fazer com que máquinas exibam comportamento pelo qual, se desempenhado por humanos, se poderia assumir o envolvimento do uso de inteligência." [2]

Neste contexto, a Programação Genética é uma técnica de CE que cria soluções para problemas, automaticamente, sem que o usuário necessite conhecer ou especificar previamente a forma ou a estrutura da solução. No nível mais abstrato, PG é um método sistemático e independente de domínio para se fazer com que computadores resolvam problemas automaticamente, partindo de uma definição de alto nível sobre o que se necessita que seja feito.

O termo *Genetic Programming* foi cunhado independentemente por John Koza e Hugo DeGaris em trabalhos publicados no início da década de 1990. Quando a definição de Koza para este termo começou a predominar, após seu importante livro lançado em 1992 [5], a maioria dos pesquisadores passou a representar a evolução de programas por estruturas em árvore e, mais especificamente, na forma de programas em linguagem LISP. Entretanto, atualmente entende-se por PG toda forma evolutiva de indução de programas [8].

Os AGs, que foram introduzidos por Holland [44], representam a forma mais conhecida de CE. Em AGs, os indivíduos de uma população podem ser representados e interpretados de várias formas, enquanto que em PG são tratados explicitamente como programas escritos em um subconjunto ou variação de uma linguagem de programação convencional.

O algoritmo básico de um AG, de forma geral, é mantido em PG: a busca é feita avaliando-se iterativamente a aptidão dos indivíduos em uma população e aplicando-se operadores genéticos (como, por exemplo, cruzamento e mutação) aos indivíduos mais aptos, de forma a explorar diferentes áreas promissoras do espaço de busca. Nos AGs, a avaliação da aptidão pode tomar várias formas, enquanto que em PG um indivíduo tem sua aptidão avaliada, pelo menos em parte, pela execução do programa e pelo acesso à qualidade de sua(s) saída(s) resultante(s).

[2] Tradução livre.

2.4.1 Algoritmo Básico

Há basicamente duas formas de se conduzir uma execução de PG: uma abordagem com gerações e uma abordagem denominada *steady-state*. Na PG com gerações, toda uma nova geração é criada a partir da geração anterior a cada ciclo. A nova geração substitui a antiga geração e o ciclo continua. Na PG *steady-state*, não há intervalos fixos de gerações, mas sim um fluxo contínuo de indivíduos encontrando-se, cruzando-se e produzindo descendentes. Os descendentes substituem indivíduos existentes na mesma população.

Os ciclos de execução dos diferentes tipos de PG são similares e independem da representação e dos operadores. O algoritmo básico de PG segue os seguintes passos:

1. Gera-se uma população inicial, composta por programas criados aleatoriamente, ou seja, cujos tamanhos e conteúdos são definidos aleatoriamente, com distribuição uniforme.

2. Avaliam-se individualmente os programas e determina-se a aptidão para cada indivíduo.

3. Copiam-se os melhores indivíduos para a próxima geração ou ciclo e efetua-se cruzamento e mutação em alguns deles.

4. Repete-se este processo, iniciado no passo 2, até que um critério de finalização seja satisfeito.

5. Apresenta-se o melhor indivíduo da população e sua aptidão como a saída do algoritmo, ou seja, a solução do algoritmo para o problema.

2.4.2 Avaliação e Seleção

A PG não é uma forma de busca do tipo "escalada" (*hill climbing*), que busca apenas por um caminho através do espaço de busca, tampouco conduz uma busca exaustiva pelo espaço de todos os possíveis programas de computador. Ao invés disto, a PG é um tipo de "busca em feixe" (*beam search*). A população é o feixe, ou seja, o conjunto de pontos no espaço de busca a partir do qual as buscas são conduzidas [8].

A PG deve selecionar os membros da população que serão submetidos aos operadores genéticos. Ao fazer esta seleção, a PG implementa uma das mais importantes partes do seu modelo de aprendizado evolutivo orgânico: a seleção baseada em aptidão, que afeta tanto a ordenação dos indivíduos no feixe quanto seu conteúdo.

A métrica de avaliação da PG é chamada de função de avaliação da aptidão, e a maneira pela qual esta função afeta a seleção dos indivíduos para os operadores genéticos é considerado como sendo o algoritmo de seleção da PG. Há inúmeros tipos de algoritmos de seleção usados em PG [8].

A aptidão é a medida usada pela PG, durante a evolução simulada, do quão bem um programa aprendeu a predizer as saídas a partir das entradas, ou seja, as características do domínio de aprendizagem. O objetivo da avaliação da aptidão é, portanto, fornecer um retorno ao algoritmo de aprendizado sobre quais indivíduos têm uma maior probabilidade de se multiplicarem e se reproduzirem, e quais indivíduos têm a maior probabilidade de serem removidos da população. A função de avaliação é calculada com base em um conjunto de treinamento e deve ser modelada de forma a fornecer um retorno contínuo e gradual do quão bem um programa é executado com este conjunto de treinamento.

2.4.3 População

Em algoritmos evolutivos em geral, a população de soluções individuais pode ser dividida em múltiplas subpopulações. A migração de indivíduos entre as subpopulações promove a evolução da população como um todo. Na biologia, este modelo foi inicialmente descrito por Wright [45] como "modelo de ilha" (*island model*), argumentando que em populações semi-isoladas, denominadas *demes*, a evolução progride mais rapidamente que em uma única população de mesmo tamanho. Esta aceleração inerente à evolução por *demes* foi confirmada para algoritmos evolutivos [46] e, em particular, para PG [47, 48]. Uma razão para esta aceleração pode residir no fato da diversidade genética ser melhor preservada em múltiplos *demes* com uma migração restrita de indivíduos. Por sua vez, a diversidade influencia na probabilidade da busca evolutiva encontrar um mínimo local, uma vez que um mínimo local em um *deme* pode ser transposto por outros *demes* que contenham uma direção de busca melhor [6].

Um tipo especial do modelo de ilha, o modelo de "alpondras" (*stepping-stone*), assume que a migração de indivíduos é apenas possível entre certos *demes* adjacentes, que estejam organizados como grafos com arestas fixas de interconexão. Os indivíduos podem alcançar populações remotas somente depois de passarem através destes vizinhos. Dessa forma, a possibilidade de haver uma troca de indivíduos entre dois *demes* depende da distância entre os mesmos na topologia do grafo. Topologias comuns são estruturas em anel ou em matriz [6].

2.5 Programação Genética Linear

A evolução de programas expressos por uma ou mais estruturas em árvore, os quais são avaliados por um interpretador apropriado, é o tipo de PG original e mais difundido. Entretanto, existem outros tipos de PG, onde programas são representados de diferentes formas, tais como, por exemplo, programas lineares [6] e programas (paralelos) em forma de grafos [49, 50].

Na PG linear, cada programa é uma sequência de instruções, tal como o mostrado na figura 2.3. O número de instruções pode ser fixo, significando que todos os programas da população têm o mesmo comprimento, ou variável, o que significa que diferentes indivíduos podem ter diferentes tamanhos.

Instrução 1	Instrução 2	...	Instrução N

Figura 2.3: Representação típica de PG linear para programas.

2.5.1 Motivações

Há dois diferentes motivos para se considerar a PG Linear. Primeiramente, a maioria das arquiteturas de computador representa seus programas por estruturas lineares (listas de instruções), com instruções adjacentes sendo normalmente executadas em instantes de tempo consecutivos, embora estruturas de controle, saltos e laços possam alterar a ordem de execução. Portanto, a ideia de se evoluir programas lineares levou Banzhaf [51], Perkins [52], assim como Openshaw e Turton [53], a investigar a PG Linear.

Em segundo lugar, computadores não executam programas em forma de árvore naturalmente, então interpretadores ou compiladores têm que ser usados como parte da PG baseada em estruturas em árvores. Por sua vez, ao evoluir os padrões de *bits* realmente obedecidos pelo computador, a PG Linear pode evitar o uso deste mecanismo computacionalmente dispendioso e a PG pode ser executada mais rapidamente, em várias ordens de magnitude. Esta busca por velocidade foi o que motivou Nordin *et al.* [54, 55], Crepeau [56] e Eklund [57] em seus respectivos trabalhos, assim como o autor desta tese [7, 58, 59].

2.5.2 Representações

É possível usar uma representação linear em PG com árvores. Entretanto, nesta abordagem, as estruturas lineares são simplesmente representações niveladas das árvores. Sendo assim, ainda se pode identificar o nó raiz na estrutura linear, seu

filho, e o resto da estrutura da árvore. Em tal sistema, as instruções se comunicam tipicamente através de seus argumentos.

Todavia, a semântica da PG Linear é completamente diferente. Na PG Linear, as instruções tipicamente lêem sua(s) entrada(s) de um ou mais registradores e/ou endereços de memória, e armazenam os resultados de seus cálculos em um registrador. Estas instruções podem, por exemplo, assumir a forma mostrada na figura 2.4, onde R_0 a R_7 referem-se a registradores da unidade central de processamento (*Central Processing Unit* – CPU). Isto significa que todas as instruções na PG Linear têm papéis equivalentes e se comunicam apenas via registradores ou memória. Em PG Linear não há equivalente para a distinção entre funções e terminais que é inerente à PG com árvores. Além disso, na ausência de laços e desvios, a posição das instruções determina a ordem das suas execuções. Já no caso da PG com árvores, os nós são normalmente visitados em profundidade pré-fixada, mas não executados. As primitivas são executadas apenas quando seus argumentos tiverem sido avaliados. Portanto, o nó raiz é o primeiro visitado, mas o último executado.

Saída	Argumento 1	Código da operação	Argumento 2
$R_0 \ldots R_7$	$R_0 \ldots R_7$	$+ - \times \div$	$0 \ldots 127$ ou $R_0 \ldots R_7$

Figura 2.4: Exemplo de um possível formato de uma instrução em PG linear.

As instruções na PG Linear podem ou não representar código de máquina executável. Ou seja, há essencialmente dois tipos de PG Linear: PG Linear de código de máquina, onde cada instrução é diretamente executável pela CPU, e a PG Linear interpretada, onde cada instrução é executável por uma máquina virtual de alto nível, que normalmente é escrita em uma linguagem eficiente em termos de tempo de execução do código compilado, tal como C ou C++. Quando as instruções são realmente código de máquina, a ordem dos elementos da representação mostrada na figura 2.4 é então determinada pela arquitetura particular do computador utilizado, e os dados correspondentes devem ser armazenados em campos de *bits* de tamanhos apropriados. Entretanto, quando se usa instruções de máquina virtual dedicada, o custo total de armazenamento e recuperação dos dados pode ser evitado, uma vez que o projetista de um sistema de PG tem total liberdade sobre como a máquina virtual interpretará suas instruções.

Se o objetivo for velocidade de execução, então o código evoluído deve ser código de máquina para o computador real, ao invés de código para uma linguagem de nível mais alto ou para uma máquina virtual. Este é o motivo pelo qual Nordin

começou a evoluir código de máquina para computadores Sun [54], através do seu modelo AIMGP, e Crepeau visou ao processador Z80 [56].

A plataforma Sun SPARC possui uma arquitetura RISC simples de 32 bits, o que facilita a criação de operadores genéticos que manipulem seu código de máquina. Em [60], cada indivíduo da PG de código de máquina é encapsulado dentro de uma função C. Cada uma das entradas do programa da PG é copiada de um dos argumentos da função C para um dos registradores de máquina. Assim como os registradores usados como entradas, um número menor de outros registradores (como, por exemplo, de 2 a 4) é usado como memória auxiliar para armazenar resultados de cálculos intermediários. Finalmente, a PG simplesmente deixa sua resposta em um dos registradores, cujo valor é utilizado como o retorno da função C.

Desde que o Unix foi desenvolvido para plataformas baseadas na linha de microprocessadores Intel x86 [61], o conjunto de instruções complexas da Intel, que já era padrão para computadores baseados em Windows™, vem predominando quase que completamente no mercado. Percebendo isto, Nordin migrou seu sistema de PG Linear da arquitetura RISC da Sun para a arquitetura CISC da Intel. Foram feitas diversas mudanças nas operações genéticas, as quais garantem que os programas aleatórios iniciais sejam feitos apenas de código de máquina Intel válido e que as operações de mutação, que atuam dentro da palavra de 32 *bits* do x86, respeitem os complexos sub-campos do x86. Uma vez que o x86 possui instruções de diferentes comprimentos, deve-se tomar um cuidado especial ao alterá-las. Tipicamente, várias instruções curtas são agrupadas em palavras de 4 *bytes*. Se houver *bytes* incompletos, estes são preenchidos com código de máquina não efetivo (*no-operation* – NOP). Desta forma, faz-se o melhor uso do espaço disponível, sem que as instruções ultrapassem o limite de 32 *bits*. O trabalho de Nordin culminou no sistema comercial Discipulus™ [62], o qual tem sido utilizado em aplicações que variam desde Bioinformática a Robótica e desativação de armamentos explosivos, conforme citadas em [12].

É importante notar que a velocidade de execução não é a única motivação para se utilizar PG Linear. Apesar de programas lineares interpretados serem mais lentos que programas em código de máquina, um sistema de PG Linear interpretada pode ser mais eficiente que um sistema interpretado baseado em estruturas em árvores [12]. Além disto, uma estrutura linear simples favorece sua rápida análise. Em [63] mostrou-se que um programa linear pode ser trivialmente esquadrinhado e que qualquer código não-efetivo nele contido pode ser removido. Sob alguns aspectos, o espaço de busca da PG Linear é também mais fácil de se analisar que o da PG com árvores [64]. Por exemplo, em [65, 66] foram utilizadas (por simulação) duas arquiteturas simples para diversos experimentos de grande escala e para a análise

matemática de PG Turing-completa. Por tais razões, faz sentido se considerar a PG linear de código de máquina virtual mesmo ao se utilizar linguagens como Java que são tipicamente executadas em máquinas virtuais. De fato, pode-se usar uma máquina virtual para se interpretar o *byte code* evoluído como, por exemplo, em [67].

2.5.3 Operadores

Os operadores típicos de cruzamento e mutação para PG Linear ignoram os detalhes do código de máquina do computador utilizado. Por exemplo, o cruzamento pode escolher aleatoriamente dois pontos quaisquer em cada genitor e trocar o código entre eles para gerar os descendentes. Uma vez que os fragmentos trocados são normalmente de comprimentos diferentes, tal cruzamento pode alterar o comprimento dos programas, conforme mostrado na figura 2.5. Sendo o código de máquina de computador organizado em palavras de 32 ou 64 *bits*, os pontos de cruzamento ocorrem apenas nos limites entre estas palavras. Portanto, números inteiros de palavras, contendo números inteiros de instruções, são trocados entre si. De forma similar, as operações de mutação normalmente respeitam os limites das palavras e geram código de máquina válido.

Entretanto, a PG Linear pode ser submetida a uma variedade de outros operadores genéticos [64]. Por exemplo, no cruzamento homólogo [55, 62], os dois pontos de cruzamento são os mesmos em ambos os genitores. Como consequência, o código extraído não muda sua posição relativa ao início do programa e os programas descendentes têm os mesmos comprimentos dos seus respectivos genitores. O cruzamento homólogo é normalmente combinado com uma pequena quantidade de cruzamento normal de dois pontos (figura 2.5) para introduzir mudanças de comprimento nos indivíduos da população.

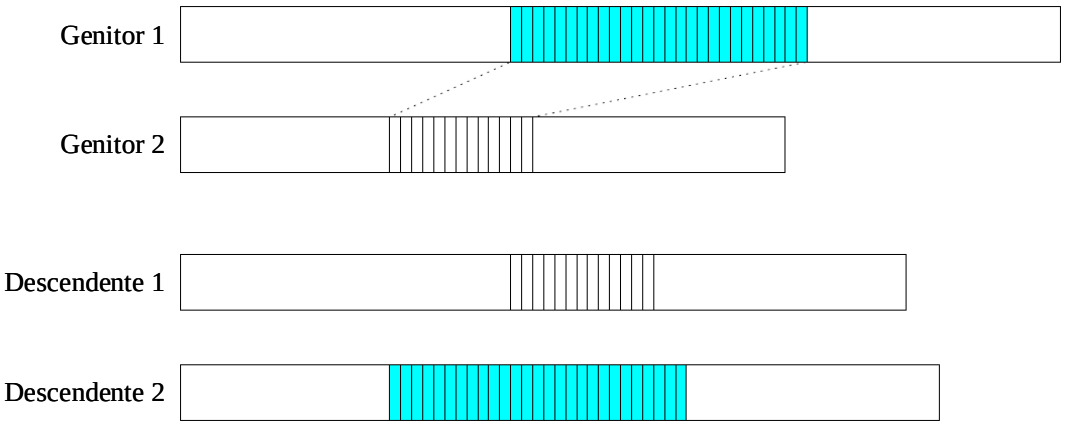

Figura 2.5: Cruzamento de dois pontos entre dois indivíduos em PG Linear.

Em um sistema de PG que evolui diretamente código de máquina [68], o operador de mutação também atua diretamente nas instruções em código de máquina. Portanto, este operador é limitado de forma a garantir que apenas instruções do conjunto de funções sejam geradas e que os valores dos registradores e das constantes estejam dentro das faixas permitidas, conforme predefinido pela configuração do experimento. Foi reportado em [68] que, em alguns problemas de classificação, o desempenho foi melhor quando se usou cruzamento e mutação em iguais proporções. Também foi sugerido que isto se devia à criação de *introns* (blocos de código que não afetam a aptidão) pela população, em resposta ao operador de cruzamento, e que estes *introns* eram subsequentemente transformados em material genético útil pela mutação.