

Universidade da Beira Interior

Departamento de Informática



**Departamento de
Informática**

Nº 16 - 2019: *Quantum Neural Networks*

Elaborado por:

António José Marques Abreu

Orientador:

Professor Doutor Luís Filipe Barbosa de Almeida Alexandre

22 de Junho de 2019

Agradecimentos

Este projeto, é o resultado de muitas horas de trabalho e não teria sido possível sem o apoio e incentivo de algumas pessoas, a quem quero exprimir os meus sinceros agradecimentos.

Ao meu orientador, Professor Doutor Luís Filipe Barbosa de Almeida Alexandre, pela sua orientação e apoio, pelo saber que me transmitiu, pelas opiniões críticas e total disposição na resolução de problemas e dúvidas que foram surgindo ao longo da realização deste trabalho. Agradecer também, o do espaço e equipamento do *Soft Computing and Image Analysis Lab* (SOCIA Lab), que por ele me foi cedido.

A todos os docentes da Universidade da Beira Interior (UBI), que de uma forma ou outra, ao longo destes últimos três anos, me transmitiram muito do conhecimento necessário para a realização deste projeto.

Aos meus colegas e membros do SOCIA Lab, pela amizade e pela disponibilidade e interesse em esclarecerem as minhas duvidas.

A todos os meus amigos, sem referenciar nomes para não correr o risco de me esquecer de algum, por todo o apoio e amizade, durante todo o meu percurso académico.

Os últimos são os primeiros, e tendo consciência que sem eles a minha vida académica não teria sido possível, quero agradecer à minha família, em especial aos meus pais e avós, que sempre foram os meus maiores exemplos de sucesso, coragem e determinação ao longo da vida. Agradecer todo o incentivo, amizade, apoio e todos os inúmeros sacrifícios que fizeram para que eu pudesse chegar até aqui. A eles, dedico este trabalho.

Conteúdo

Conteúdo	iii
Lista de Figuras	v
1 Introdução	1
1.1 Enquadramento	1
1.2 Motivação	1
1.3 Objetivos	1
1.4 Organização do Documento	2
2 Introdução à Computação Quântica	5
2.1 Introdução	5
2.2 Computadores Quânticos e Qubits	6
2.3 Correção Quântica de Erros e Computação Quântica Tolerante a Falhas	8
2.4 Escalonamento dos Sistemas Quânticos	9
3 IBMQ - Qubits e Gates	11
3.1 Introdução	11
3.2 <i>Quantum Composer</i>	11
3.3 Representação em Histogramas (Gráfico de Barras)	13
3.4 Visualização e Representação de Qubits	14
3.5 <i>Single-Qubit Gates</i>	15
3.5.1 <i>X-Gate</i>	16
3.5.2 <i>H-Gate</i> e Criação de Sobreposição	17
3.5.3 Sobreposição Negativa	18
3.5.4 <i>Qubit Phase</i>	19
3.6 Decoerência	20
3.6.1 Relaxamento de Energia e T_1	21
3.6.2 <i>Dephasing</i> e T_2	21
3.6.3 Processo de Decoerência com Qubits Super-condutores	22

3.7	Sistemas de N Qubits ($N > 1$)	22
3.7.1	Reversibilidade	24
3.8	Entrelaçamento e Testes de Bell	24
3.9	Conclusões	25
4	O Algoritmo de Shor	27
4.1	Introdução	27
4.2	Algoritmo de Shor	28
4.3	Implementação do Algoritmo de Shor	31
4.3.1	Resultados	31
4.4	Conclusões	33
5	Redes Neurais Quânticas	35
5.1	Introdução	35
5.2	Como Implementar uma Rede Neuronal Quântica	36
5.3	Redes Neurais Clássicas Vs Quânticas	37
5.4	Conclusões	38
6	Redes Bayesianas Quânticas	39
6.1	Introdução	39
6.2	Rede Bayesiana Clássica	39
6.3	Rede Bayesiana Quântica	41
6.4	Comparação dos Resultados	42
6.5	Conclusões	42
7	Conclusões e Trabalho Futuro	45
7.1	Conclusões Principais	45
7.2	Trabalho Futuro	46
A	Algoritmos	47
A.1	Algoritmo de Shor	47
A.2	Rede Bayesiana Clássica	50
A.3	Rede Bayesiana Quântica	52
	Bibliografia	55

Lista de Figuras

3.1	<i>Score</i> [16]	12
3.2	Medição [16]	12
3.3	Histograma [17]	13
3.4	Histograma 2 [17]	14
3.5	' <i>Bloch Sphere</i> ' [22]	15
3.6	<i>X Gate</i> [21]	16
3.7	Resultados da aplicação do <i>T-Gate</i> [18]	20
3.8	Evolução dos tempos de coerência ao longo dos anos [15]	22
3.9	Representação gráfica de um <i>CNOT</i> [19]	23
4.1	Algoritmos de fatorização clássicos <i>Vs</i> algoritmo de Shor [20]	31
4.2	Resultados do algoritmo de Shor, para $a = 2$	32
4.3	Resultados do algoritmo de Shor, para $a = 7$	32
4.4	Resultados do algoritmo de Shor, para $a = 11$	33
4.5	Circuito gerado por Qiskit, do algoritmo de Shor [20]	33
5.1	Neurónio [12]	36
6.1	Output WetGrass.py	40
6.2	Output QuWetGrass.py	41

Acrónimos

e.g.	por exemplo
IBM	<i>"International Buisness Machines"</i>
IBMQ	<i>IBM Quantum</i>
NISQ	<i>Noisy Intermediate-scale Quantum</i>
UC	Unidade Curricular
UBI	Universidade da Beira Interior
RN	Rede Neuronal
RNs	Redes Neurais
RNQ	Rede Neuronal quântica
RNQs	Redes Neurais quânticas
RSA	<i>Rivest-Shamir-Adleman</i>
SOCIA Lab	<i>Soft Computing and Image Analysis Lab</i>

Capítulo 1

Introdução

1.1 Enquadramento

Este projeto, será realizado no âmbito da Unidade Curricular (UC) Projeto (11572) do curso de Engenharia Informática, da Universidade da Beira Interior (UBI).

1.2 Motivação

O presente projeto, vai de encontro a duas áreas de interesse pessoal, que são elas, Inteligência Artificial e Computação quântica. A oportunidade de estudar um tema, que junta ambas as áreas e poder alargar o meu conhecimento sobre as mesmas, despertou-me um grande interesse em realizar este projeto. Sendo também, a computação quântica uma área recente, poder começar a estudar os conceitos base da mesma numa fase inicial da sua existência e estender esse mesmo conhecimento à medida que a área se vai desenvolvendo, certamente irá levar a um melhor entendimento da mesma no futuro.

1.3 Objetivos

A área da Computação quântica, tem visto um grande crescimento na última década, devido ao seu futuro promissor na área da Computação e da Ciência. Um dos maiores desafios que a área da Computação tem em mãos, é a criação de Redes Neurais quânticas, na esperança de criar uma verdadeira inteligência artificial.

Assim, o objetivo do presente projeto, é a criação de uma pequena Rede Neu-

ronal quântica e comparação da mesma com Redes Neurais já existentes. A Rede Neuronal será algo simples e com utilidade meramente acadêmica, mas que servirá como um marco fulcral para a realização de trabalho futuro, não só na área da Computação quântica, bem como na da Inteligência Artificial.

1.4 Organização do Documento

Para atingir o objetivo do presente projeto, foram necessários pontos intermédios, desde a aprendizagem das noções base da Física quântica até aos algoritmos quânticos já existentes. Como tal, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.
2. O segundo capítulo – **Introdução Computação Quântica** – contém uma breve introdução, sobre o que é a Computação quântica e algumas diferenças desta, para com a Computação tradicional. São introduzidos alguns conceitos chave, relacionados com a Física quântica, que são aprofundados nos capítulos posteriores. Explica, a forma como a informação é representada e processada num sistema quântico e como estes sistemas fazem uso das propriedades da Mecânica quântica. São, ainda, referidos os conceitos de erro, tolerância e escalonamento em sistemas quânticos.
3. O terceiro capítulo – **IBMQ - Qubits e Gates** – começa com uma breve explicação de como funcionam os computadores quânticos da IBM e a forma como podemos interagir com os mesmos. Este, é um capítulo mais teórico, onde são referidos e explicados os vários tipos de *gates* (portas lógicas quânticas) e a forma estes manipulam os qubits, de forma a extraírem as propriedades quânticas dos mesmos. Apresenta, também, algumas justificações de cariz matemático para alguns dos conceitos mais relevantes.
4. O quarto capítulo – **O Algoritmo de Shor** – contém uma breve introdução sobre o algoritmo de Shor (um dos mais relevantes na área da computação quântica), bem como os fundamentos matemáticos por detrás do mesmo. Posteriormente, é apresentada uma implementação do mesmo e interpretados os resultados obtidos.
5. O quinto capítulo – **Redes Neurais Quânticas** – reflete o objetivo central do presente projeto. Inicia-se, com uma breve introdução às Redes Neuro-

nais tradicionais e os principais motivos que estão a levar à tentativa de criação de Redes Neurais quânticas. Explica os pontos teóricos para a criação de Redes Neurais quânticas (RNQs) e os principais entraves à criação das mesmas. Termina, com uma breve comparação entre as Redes Neurais tradicionais e quânticas.

6. O sexto capítulo – **Redes Bayesianas Quânticas** – debruça-se sobre o conceito das Redes Bayesianas, fazendo uma comparação direta, entre uma implementação tradicional e uma quântica. Mostra que de facto, os resultados obtidos por ambas, se encontram dentro de uma margem de erro aceitável.
7. O sétimo capítulo – **Conclusões e Trabalho Futuro** – discute as principais conclusões retidas ao longo do projeto, e enuncia o que não foi conseguido e o que se pode vir a fazer no futuro, para melhorar e complementar o trabalho feito até aqui.

Capítulo 2

Introdução à Computação Quântica

2.1 Introdução

Todos os dias beneficiamos das capacidades do mundo digital. Desde o entretenimento à investigação, os computadores estão presentes constantemente nas nossas vidas. Atualmente, já são capazes de processar enormes quantidades de dados e de resolver em tempo útil problemas e cálculos, de tal maneira complexos que levariam anos a serem resolvidos por nós Humanos. Contudo, existem problemas cuja complexidade é de tal forma exponencial, que levariam anos a serem concretizados por um computador "tradicional". Posto isto, surgiu a necessidade de criar um computador, com capacidade de processamento igualmente exponencial à complexidade dos problemas que foi concebido para resolver. Por outras palavras, um computador que logaritmicamente reduz as complexidades exponenciais, resolvendo em tempo útil este tipo de problemas.

Os computadores tradicionais, baseiam-se na sua capacidade de armazenar e processar informação, ao manipular individualmente bits (0 e 1), através de portas lógicas como as *NOT*, *AND*, *OR*. Os milhões de bits num sistema, e a capacidade de manipular o estado individual de cada um deles, conferem ao computador tradicional as capacidades que todos conhecemos. Por outro lado, os computadores quânticos aproveitam-se de fenómenos Físicos, conhecidos como quânticos (sobreposição, entrelaçamento e interferência), para manipular informação. Neste caso, os bits passam a chamar-se qubits (junção das palavras quântico e bits).

O fenómeno de sobreposição, refere-se à combinação de estados, que normalmente descrevemos como independentes. Usando um exemplo simplista, se tocarmos duas notas musicais, por exemplo num piano, o que iríamos ouvir, seria uma sobreposição de ambas as notas.

O entrelaçamento, é um conceito quântico extremamente contra-intuitivo, e que

descreve um comportamento, que nunca será visível no mundo que observamos no dia-a-dia. Partículas que se dizem entrelaçadas, têm um comportamento conjunto que não consegue ser descrito pela Física clássica. Einstein, chamou a este fenómeno "*spooky action at a distance*". De forma sucinta, observar um qubit, causa que este se comporte de forma aleatória, contudo, indica ao observador o estado exato em que se encontraria o par deste, caso fosse observado. Entrelaçamento, envolve a correlação entre dois qubits. Se o primeiro qubit do par for observado no estado 1, então o segundo qubit vai encontrar-se no estado 0 e vice-versa. Devido, no entanto, ao rigor científico, só podemos afirmar a existência de correlação, depois de observarmos os estados dos dois qubits, impedindo de certa forma o uso direto desta propriedade. Contudo a existência desta propriedade num sistema quântico, é uma das principais responsáveis pela capacidade de processamento superior, dos computadores quânticos, quando comparados com sistemas tradicionais.

Por último, mas não menos importante, temos o fenómeno da interferência, que dita que cada estado quântico tem uma fase, e como tal, pode ser alvo de interferência. Este conceito, pode ser comparado ao comportamento das ondas sonoras. Nas ondas sonoras, existem dois tipos de interferência: a interferência construtiva e a destrutiva. Se ambas as ondas estão "em fase", as suas amplitudes somam (interferência construtiva), se as ondas estiverem "fora de fase", as amplitudes cancelam-se (interferência destrutiva), tal como acontece, por exemplo, nos auscultadores com tecnologia de cancelamento de ruído. Estas duas propriedades de interferência são também observadas no mundo quântico.

2.2 Computadores Quânticos e Qubits

Um computador tradicional usa bits tradicionais, em que cada um pode assumir o valor 0 (zero) ou 1 (um), mas nunca ambos os valores ao mesmo tempo. Isto permite que um computador clássico, com N bits, num determinado instante de tempo, se encontre em um de 2^N estados possíveis. Já os computadores quânticos, usam bits quânticos (qubits), que podem encontrar-se no valor 1 (um) e 0 (zero) ao mesmo tempo. Os estados base de um sistema quântico, são convencionalmente escritos como $|1\rangle$ e $|0\rangle$ (*ket 1* e *ket 0* respetivamente), e o estado de um qubit é a sobreposição linear de ambos os estados. O que significa, que cada qubit pode ser representado como uma combinação linear de $|1\rangle$ e $|0\rangle$, segundo a equação (equação 2.1):

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad (2.1)$$

em que α e β , elevados ao quadrado, representam a probabilidade do qubit se encontrar no estado 1 (um) ou 0 (zero), respetivamente ($|\alpha|^2 + |\beta|^2 = 1$).

Esta característica conhecida como sobreposição, é o que confere uma maior capacidade de processamento de dados aos computadores quânticos em comparação aos tradicionais. A sobreposição, permite a um sistema quântico aumentar exponencialmente o número de estados em que se encontra. Um computador quântico, pode encontrar-se em todos os estados de $|00\dots 0\rangle$ até $|11\dots 1\rangle$, ao mesmo tempo. Atualmente, usam-se fótons ou elétrons como qubits, e apesar de características distintas dos elementos, o conceito é exatamente mesmo. O estado de menor energia é considerado 0 (zero) e o de maior energia é considerado 1 (um) (*spin up and down*). Como se tratam de “objetos” com propriedades quânticas, podem encontrar-se em sobreposição de estados (em ambos os estados ao mesmo tempo) (*up and down*). No entanto quando o ‘*spin*’ é medido, o estado de sobreposição colapsa, e o qubit, encontra-se *up* ou *down*, segundo as probabilidades de α e β da equação 2.1.

Podemos então concluir que um sistema com 2 qubits, se estiver no estado de sobreposição, antes de ser medido, encontra-se simultaneamente nos estados ($|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$). Caso fosse um sistema de 3 qubits ($|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$, $|111\rangle$). Logo, em N qubits, existem 2^N bits de informação (por exemplo (e.g.) num sistema com 300 qubits, temos 2^{300} bits de informação). Veremos nos capítulos seguintes, que a informação dos qubits se vai degradando com o tempo devido a ruído e decorrência dos sistemas.

Apesar dos qubit se poderem encontrar numa combinação de estados, assim que são medidos, têm obrigatoriamente de se inserir num dos estados base (segundo as probabilidades dos mesmos), no caso do sistema de 2 *spins*, após medidos, têm de se inserir num dos 4 estados base. E toda a outra informação existente em relação ao estado antes da medição, é perdida. Posto isto, por norma, não queremos ter como resultado final da computação quântica, algo que seja uma sobreposição de estados, pois é impossível medir uma sobreposição. Apenas é possível medir 1 dos N estados base do sistema. O que se pretende, é então a criação das operações lógicas que são necessárias para chegar ao resultado computacional final, de modo a que o resultado final, seja, de facto, algo que possa ser medido, ou seja, um estado único.

Apesar desta escalabilidade da computação quântica, que permite resolver problemas com complexidades computacionais exponenciais, em tempo logarítmico, em relação ao um computador tradicional, os computadores quânticos não podem nem devem ser vistos como um substituto ao computador tradicional, sobretudo no que diz respeito a casos de uso comerciais. Um computador quântico não é universalmente mais rápido que um computador tradicional. Um computador

quântico é apenas mais rápido, em certos e determinados tipos de cálculos computacionais, em que é possível tirar partido das propriedades da Física quântica, em especial da sobreposição de estados para efetuar uma espécie de paralelismo computacional. Para casos mais básicos como acesso à Internet e consumo de conteúdo, os computadores tradicionais são mais rápidos e eficientes. Isto, porque num computador quântico, as operações de processamento não são mais rápidas do que na computação tradicional, alias, as operações individuais num computador quântico podem até ser mais lentas. É apenas nos casos, em que são necessárias várias operações computacionais para chegar ao resultado final, que um computador quântico consegue reduzir o número de operações necessárias de forma exponencial. O ganho, não está na velocidade de cada operação individual, mas sim no número de operações necessárias fazer para se chegar ao resultado final, que é exponencialmente mais pequeno num computador quântico [23].

2.3 Correção Quântica de Erros e Computação Quântica Tolerante a Falhas

Algoritmos tolerantes a falhas [6], foram dos primeiros a serem desenvolvidos na área da computação quântica, nomeadamente o algoritmo de fatorização de Shor e o algoritmo de pesquisa não estruturada de Grover. Estes algoritmos, foram concebidos para correrem em computadores quânticos tolerantes a erros, que são atualmente o sonho e objetivo supremo da área da computação quântica.

Apesar de já existirem alguns computadores quânticos atualmente, onde já estão a ser corridos alguns algoritmos quânticos mais primitivos, estes primeiros ainda têm um número muito reduzido de qubits, o que limita o número de operações dos segundos. Esta limitação, tanto do número de qubits como do número de operações realizadas, permite que os erros gerados, sejam compensados ao executar os algoritmos várias vezes, e no final, calcular o valor médio dos resultados. Os algoritmos tolerantes a erros, exigem extensas sequências de portas lógicas (chamadas '*gates*') e um número elevado de qubits, para conseguirem correr da melhor forma possível, retornando valores o mais exatos possível. No entanto, como já referido, este tipo de computadores quânticos, apesar de já estarem a ser estudados, não irão existir num futuro próximo. Mas ainda assim, os algoritmos e *hardware* já existente, já permitem demonstrar, que em certos casos específicos (em que é necessário processar uma quantidade de dados elevada), este tipo de computação, supera largamente a computação tradicional.

Os estados quânticos são extremamente delicados, como tal, para criar um

computador quântico estável é necessário criar mecanismos de correção de erros. É necessário evitar que ruído interfira com o estado quântico do sistema.

Os computadores tradicionais não usam sistemas de correção de erros. Dado o elevado número de transístores, mesmo que haja erro num deles, não é grave para o funcionamento geral do sistema. Já num computador quântico, um qubit é representado por apenas uma, ou um conjunto muito pequeno de partículas com propriedades quânticas. Daí a necessidade de garantir que este permanece num estado que seja previsível e estável. Ainda comparando os dois tipos de sistemas, o computador tradicional, é digital, permitindo apenas 2 valores (discretos), 0 ou 1. Ao fim de cada iteração o valor é corrigido pelo valor mais próximo (mais próximo de 0 ou de 1). Já na vertente quântica, este pode assumir um conjunto contínuo de valores, o que impossibilita a correção por aproximação ao fim de cada ciclo. Se tentássemos proceder do mesmo modo na vertente quântica, ao fim de cada pequena correção haveria o risco do estado ficar com um valor ligeiramente diferente do estado que se pretende, devido a sobre-rotações nos spins. Estes pequenos desvios de valor, iriam aumentar gradualmente ao fim de cada iteração, até se tornar num erro considerável.

No entanto, é de facto necessário criar sistemas quânticos tolerantes a erros, de modo a que um simples erro não ponha em causa todo o sistema. Cada vez que um qubit é “usado”, o mesmo, irá inevitavelmente ser ligeiramente alterado, devido a ruído ou decoerência. O objetivo é criar protocolos, que continuem a produzir resultados corretos, mesmo que componentes individuais do sistema falhem, tal como acontece na computação tradicional. Contudo, os métodos usados na computação tradicional, não podem ser adaptados diretamente ao mundo quântico. Uma das formas mais populares de correção de erros é através da redundância. Em que, ao fim de cada ciclo, o estado atual é comparado ao que está guardado, e caso, os estados não sejam coincidentes, procede-se á correção do erro. Devido ao teorema de ‘non-cloning’, que determina a impossibilidade de copiar informação quântica, torna de todo impossível esta estratégia [23].

2.4 Escalonamento dos Sistemas Quânticos

De forma a aumentar o poder computacional dos computadores quânticos, terão de ser efetuadas melhorias em dois sentidos distintos. O primeiro, terá de se aumentar o número de qubits presentes nos computadores quânticos. Quantos mais qubits existirem num computador, maior é o número de estados que podem ser manipulados e armazenados, permitindo assim, aumentar a complexidade dos algoritmos que usamos. Em segundo, é necessário reduzir a taxa de erros. Temos de ser capazes de manipular os estados dos vários qubits, de forma precisa, sem

criar ruído no processo, o que vai permitir resultados finais mais precisos, reduzindo, assim, a necessidade de correr o algoritmo várias vezes e calcular o valor médio dos resultados obtidos.

Combinando estes dois conceitos/objetivos, podemos criar uma medida para quantificar a capacidade de processamento de um computador quântico. Esta medida é designada por '*quantum volume*', ou em português, volume quântico, e mede a relação entre a quantidade e a qualidade dos qubits, complexidade de circuitos e taxa de erros. Quanto "maior" for o sistema e com uma correspondente baixa taxa de erros, maior vai ser o proveito que podemos tirar das propriedades quânticas deste tipo de sistemas.

É de certa forma, equivalente a relacionar o número de *cores* de um processador tradicional e a respetiva velocidade em GHz de cada um, para determinar a sua qualidade e capacidade de processamento [23].

Capítulo 3

IBMQ - Qubits e *Gates*

3.1 Introdução

De forma a alcançar o objetivo deste projeto, é preciso primeiro perceber como funciona, não só um sistema quântico, mas também as leis da Física que lhes estão adjacentes. Tratando-se de uma área recente, existem ainda poucas plataformas bem implementadas, para programar este tipo de sistemas. Durante a elaboração do projeto, a da "*International Buisness Machines*" (IBM) foi a que pareceu ser mais adequada. Para além de possuírem uma vasta e intuitiva documentação, disponibilizam, via *cloud* a interação com um verdadeiro sistema quântico. Desenvolveram também, uma biblioteca (*Qiskit*) [24] compatível com a linguagem *Python*, que permite simular o comportamento de um sistema quântico real, numa máquina tradicional. Assim, foi necessário, perceber como funciona os sistemas quânticos no geral, bem como as ferramentas disponibilizadas pela IBM.

Como tal, este capítulo, reflete precisamente essa aprendizagem prévia, expondo os conceitos mais relevantes.

3.2 *Quantum Composer*

O '*Quantum Composer*' da IBM, consiste numa interface gráfica que permite programar, de forma simples, um computador quântico da IBM que se encontra disponível através de um serviço *cloud*, ou então, correr uma simulação de um computador quântico, num sistema tradicional. O *composer*, permite a criação de um *score*, que representa graficamente o sistema que queremos simular, figura 3.1. Cada linha do *score* representa um qubit, e a sua progressão ao longo do tempo. No início da execução do algoritmo, os estados quânticos são "preparados" em estados bem definidos, $|0\rangle$ por *default*, e a execução segue o fluxo da esquerda para

a direita.



Figura 3.1: *Score* [16]

Os *gates* quânticos, são representados por quadrados e emitem frequências durante um determinado tempo e com uma certa amplitude. A frequência, tempo e amplitude variam de *gate* para *gate*. Todos os circuitos, têm de terminar com uma de medição.



Figura 3.2: Medição [16]

No exemplo da figura 3.2, foi criado apenas um qubit, com apenas um bit clássico no registo 'c' (Este registo serve para guardar os valores dos estados em que se encontravam os qubits quando foram medidos. O registo pode ser comparado a um vetor, onde a informação dos estados é guardada de 0 (zero) a $N - 1$). Na medição, o qubit encontrava-se no estado 0 (zero), e esse valor foi guardado na primeira posição do registo 'c'..

Após a medição, a informação do qubit, passa a ser apenas um bit, o que significa que perdeu as propriedades quânticas de sobreposição e entrelaçamento. Cada qubit, após a sua medição, assume os valores 0 (zero), se foi medido no estado $|0\rangle$, ou 1 (um) se foi medido no estado $|1\rangle$. Por vezes, o nosso qubit tem uma probabilidade de 50/50 de se encontrar no estado $|1\rangle$ ou $|0\rangle$, como é o exemplo de um estado equilibrado de sobreposição. Neste caso, ao repetir o circuito várias vezes no sistema quântico verdadeiro, constatamos que metade das medições assumem o valor 0 (zero) e a outra metade, o valor 1 (um).

3.3 Representação em Histogramas (Gráfico de Barras)

Nos histogramas [17], a sequência de zeros e uns, na base de cada barra, representam os estados medidos em cada qubit. Os valores estão ordenados da direita para a esquerda (o valor mais à direita representa o primeiro qubit e o mais à esquerda, representa o último qubit). A altura das barras, representa a frequência (de zero a um), em que a combinação de estados foi visualizada.

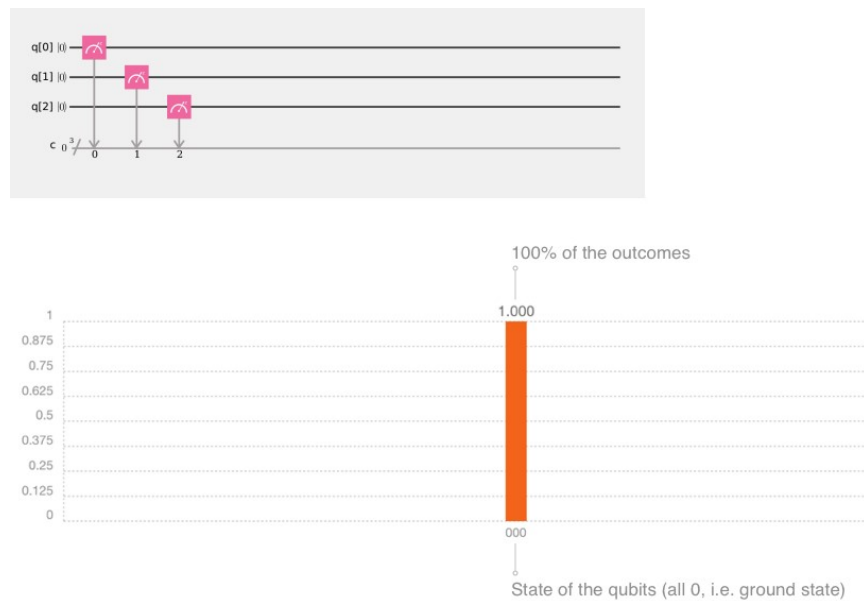


Figura 3.3: Histograma [17]

No exemplo da figura 3.3, o histograma apresenta apenas uma barra, com a legenda '000', com frequência de valor 1. Isto, significa, que apenas foi observada a combinação de estados '000'. Se adicionarmos um *H gate* (que será explicado mais à frente) antes de cada medição, o histograma apresentado já é completamente diferente do exemplo anterior, como se pode observar na figura 3.4. Neste caso, já foram observadas mais combinações de estados.

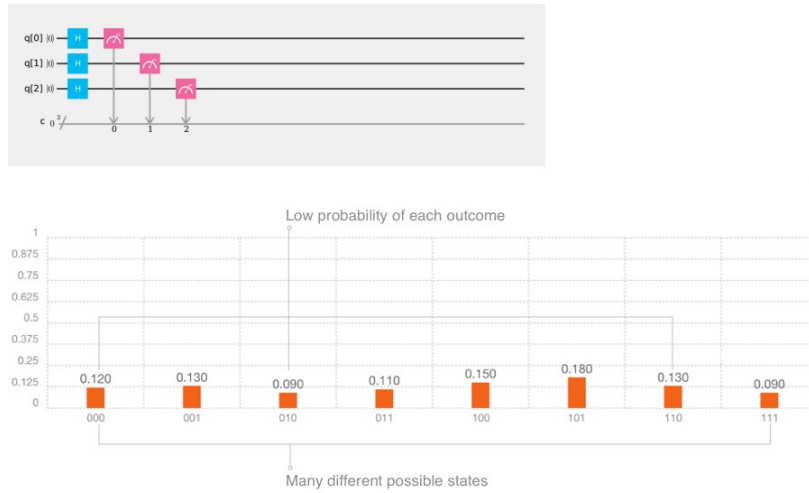


Figura 3.4: Histograma 2 [17]

3.4 Visualização e Representação de Qubits

Como já foi referido, um qubit, é objeto quântico, com dois níveis de energia $|1\rangle$ e $|0\rangle$. Combinados, $|1\rangle$ e $|0\rangle$, formam um vetor base, e tal como todos os outros vetores, estes possuem uma direção e uma magnitude. É possível definir vetores, segundo a Álgebra Linear e uma vez definidos, é possível construir outros vetores, a partir da combinação linear de vetores já existentes. Adicionalmente, os qubits também têm uma Fase, resultante das propriedades da sobreposição dos mesmos. Tirando partido da equação 2.1, podemos escrever $|1\rangle$ e $|0\rangle$ como uma combinação linear, onde a proporção de cada coeficiente depende de α e β . Estes coeficientes α e β , podem ser valores positivos, negativos ou até mesmo complexos.

Os estados base $|1\rangle$ e $|0\rangle$ e a sua respetiva combinação linear, que pode ser descrita pela equação 2.1, descrevem o estado de um qubit. Para ajudar a visualização de toda esta informação relativa ao estado de um qubit, é usada a '*Bloch Sphere*' [22]. A '*Bloch Sphere*', é uma esfera de raio 1 (um), cujos pontos da superfície representam o estado de um qubit. Para descrever o estado de um qubit, são usados ângulos, funcionando como a longitude e latitude no Globo. Isto, permite que o estado de qualquer qubit, seja representado como um ponto na superfície da Esfera, podendo mesmo, serem visualizados vários qubits por Esfera (com mais do que um qubit, a esfera passa a chamar-se '*QSphere*').

Fazendo uso das imagens da *IBM Quantum* (IBMQ) que explicam este conceito, na imagem 3.5, podemos ver a laranja o estado de um qubit x . Os valo-

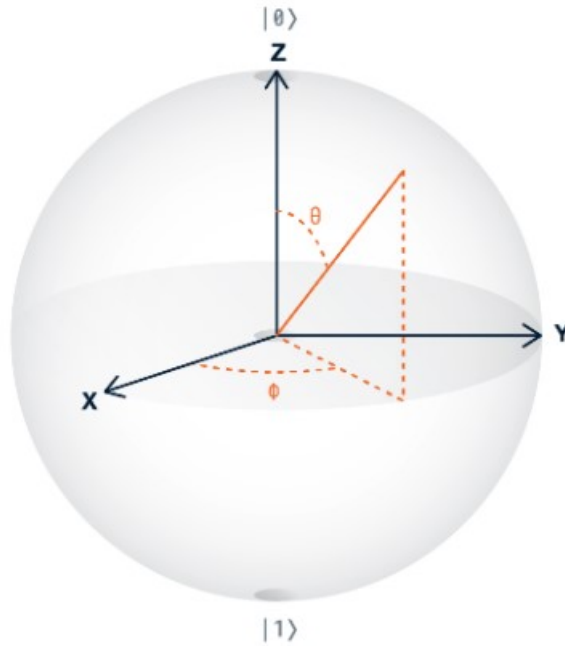


Figura 3.5: 'Bloch Sphere' [22]

res positivos de Z representam o estado $|0\rangle$ e os valores negativos, o estado $|1\rangle$. Quando um qubit se encontra num estado de sobreposição entre $|1\rangle$ e $|0\rangle$, o vetor vai apontar algures no meio da Esfera (com θ no intervalo $]0, \pi[$).

Esta esfera, permite ainda outro grau de visualização sobre a informação do estado de um qubit. Rotações em torno do ângulo ϕ , descrevem se o qubit mudou ou não de Fase. Se ϕ for diferente de zero, significa que o qubit sofreu uma alteração na sua Fase.

3.5 Single-Qubit Gates

Tal como computadores tradicionais usam portas lógicas para efetuarem operações sobre bits, também na vertente quântica, existem portas lógicas para manipular qubits. Neste capítulo, veremos algumas propriedades fundamentais dos qubits e como manipular qubits singulares com recurso a portas lógicas quânticas [21].

3.5.1 *X-Gate*

O *X Gate* [21] é conhecido como um "*bit flip*". Ou seja, transforma qubits do estado 0 (zero) para 1 (um) e vice-versa. Comporta-se de certa forma como uma porta *NOT* tradicional.

É essencialmente uma rotação sobre o eixo *X*. Se inicialmente o qubit se encontrava no estado $|0\rangle$, na parte superior da superfície da Esfera (segundo o eixo *Z*), a atuação do *X Gate* faz com que o estado do qubit passe para $|1\rangle$, ou seja, para a metade inferior da superfície da Esfera. Este comportamento, pode ser visualizado na figura 3.6.

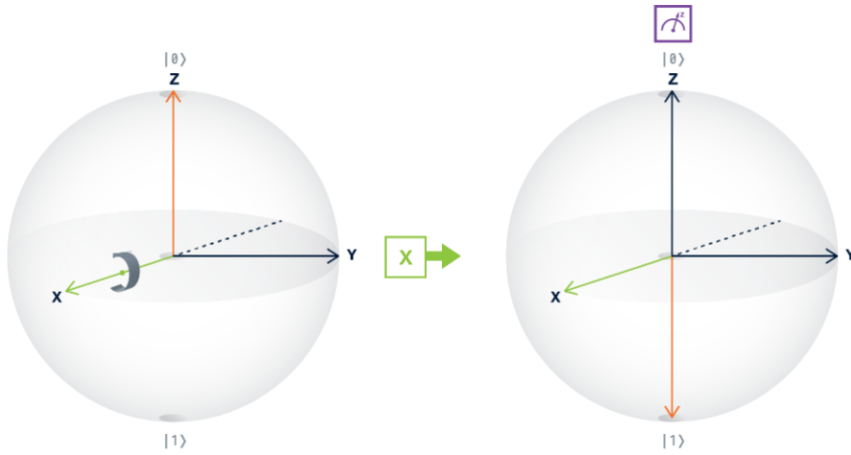


Figura 3.6: *X Gate* [21]

Podemos escrever matematicamente os estados base de um qubit, como sendo uma matriz coluna. O estado $|0\rangle$, é representado pela matriz 3.1 e o estado $|1\rangle$ pela matriz 3.2.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad (3.1)$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (3.2)$$

Matematicamente, o *X-Gate* é representado pela matriz 3.3. Se multiplicarmos a matriz X pela matriz do estado $|0\rangle$, obtemos a matriz do estado $|1\rangle$. Similarmente, se multiplicarmos a matriz X pela matriz do estado $|1\rangle$, obtemos a matriz do estado $|0\rangle$.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (3.3)$$

3.5.2 *H-Gate* e Criação de Sobreposição

Para criar sobreposição [14] nos sistemas da IBM, é usado o *H-Gate*. Isto vai fazer com que, ao corrermos a simulação várias vezes, em metade dos casos a medição do qubit terá o valor 1 (um) e na outra o valor 0 (zero). Antes da medição forçar o qubit a escolher um dos dois casos, este, não se encontra nem no estado $|1\rangle$ nem no estado $|0\rangle$, mas sim, numa sobreposição dos dois estados. A equação 3.4 representa a aplicação do *H-Gate*, ao estado $|0\rangle$. Isto é o mesmo que dizer que os coeficientes α e β da equação 2.1 têm o mesmo valor ($\frac{1}{\sqrt{2}}$, pois este valor ao quadrado representa a probabilidade de cada um dos estados ser observado, e que numa situação de sobreposição é igual a $\frac{1}{2}$). O estado $|+\rangle$, têm $\frac{1}{2}$ de probabilidade de ser medido como 0 (zero) e $\frac{1}{2}$ de probabilidade de ser medido como 1 (um).

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle). \quad (3.4)$$

Se no simulador da IBM, criarmos um *score* com apenas um qubit e lhe aplicarmos um *H-Gate* antes de o medirmos, iríamos verificar que o resultado seria de aproximadamente $\frac{1}{2}$ para o caso 0 (zero) e $\frac{1}{2}$ para o caso 1 (um). Mas se em vez do simulador, usarmos uma máquina quântica real, veríamos que os resultados seriam bem diferentes devido ao ruído, que como já foi referido é um dos maiores desafios da computação quântica. No entanto, se usarmos o método das execuções repetidas e seguidamente calculássemos o valor médio, também chegaríamos a um valor de aproximadamente 50/50.

A aleatoriedade quântica, é muito distinta da aleatoriedade "tradicional". Para testarmos este fenómeno, em vez de apenas um *H-Gate*, usamos dois (um a seguir ao outro no mesmo qubit) antes de medirmos o estado, sobre o eixo X . Antes de verificarmos os resultados, estaríamos à espera de uma distribuição de resultados de aproximadamente 50/50, contudo, mesmo numa simulação sem ruído, não seria este o caso. O resultado da medição é na verdade 1 (100%) para o estado $|0\rangle$.

Este é um resultado completamente inesperado, e complexo de explicar. No primeiro exemplo, o qubit passa de um estado base para um estado de sobreposição ao entrar no *H-Gate*. Este estado de sobreposição é rompido com a medição, que força o qubit a voltar a um dos estados base. Neste caso, diz-se que a medição foi feita segundo a base computacional. No segundo exemplo, o segundo *Gate* pode ser visto como parte da medição final e que força a medição final numa base de sobreposição. Esta conclusão é provada pelo resultado da medição final (sempre $|0\rangle$), que mostra que o sistema se encontra num estado de sobreposição positiva $|+\rangle$. A explicação matemática será dada de seguida".

3.5.3 Sobreposição Negativa

A sobreposição negativa [14] é representada pela equação 3.5.

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \quad (3.5)$$

Se aplicarmos os dois exemplos anteriores a um qubit, com estado inicial $|1\rangle$, verificamos que os resultados do primeiro são iguais em ambos os casos. No segundo exemplo, o resultado já não é de 1 para o estado $|0\rangle$, mas sim de 1 para o estado $|1\rangle$. Estamos, neste caso, perante uma sobreposição negativa.

Conclui-se que, a aplicação do *H-Gate* no estado inicial $|0\rangle$ é representado pela equação 3.4 e é chamada de sobreposição positiva. A aplicação do *H-Gate* ao estado inicial $|1\rangle$ é representado pela equação 3.5 e é chamada de sobreposição negativa. Significa isto, que o *H-Gate*, transforma a base computacional numa nova base, chamada de base de sobreposição, definida pelo conjunto $\{|+\rangle, |-\rangle\}$.

Os *H-Gates* também podem ser representados por matrizes, o que nos permite demonstrar matematicamente as sobreposições negativa (equação 3.6) e positiva (equação 3.7).

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = |+\rangle \quad (3.6)$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = |-\rangle \quad (3.7)$$

3.5.4 Qubit Phase

Nas secções anteriores sobre *Single-Qubit Gates*, vimos como usar as portas lógicas X e H , que nos permitem manipular um qubit individualmente e criar sobreposição. Agora, vamos ver como podemos manipular a Fase [18] de uma sobreposição. Para manipular as sobreposições, é usado o T -Gate. Graficamente, T -Gate é uma rotação em torno do eixo Z . As rotações são feitas segundo um ângulo θ , com amplitudes em múltiplos de $\frac{\pi}{4}$. Veremos, mais à frente o que acontece com diferentes amplitudes de θ .

Dada a grandeza da complexidade da sobreposição, e como já seria de esperar, manipular este fenómeno quântico tem associado uma panóplia de demonstrações matemáticas, cujas complexidades excedem o âmbito deste projeto. A maioria das demonstrações é de cariz meramente teórico e como tal apenas serão referidas apenas, as equações e demonstrações, que de facto contribuam para atingir os objetivos finais do presente projeto.

Vimos já, como podemos representar o estado de um qubit (equação 2.1). Além disto, a Fase global de um qubit é "não-detetável", assim, podemos escrever $|\psi\rangle$ como $e^{i\gamma}|\psi\rangle$. Juntando estas duas condições, podemos reescrever a equação 2.1, para a equação 3.8, onde $0 \leq p \leq 1$ é a probabilidade do qubit estar no estado 0 (zero) e $0 \leq \theta \leq 2\pi$ é a Fase quântica.

$$|\psi\rangle = \sqrt{p}|0\rangle + e^{i\theta}\sqrt{1-p}|1\rangle \quad (3.8)$$

O conjunto de portas, geradas por $\{T, H\}$, permite chegar a todos os valores de p e θ . A representação matricial de T pode ser vista a seguir (equação 3.9), onde é claramente visível a aplicação de Fases de $\frac{\pi}{4}$ por parte de T ao qubit.

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \quad (3.9)$$

Se o sistema se encontrar no estado inicial $|+\rangle$ e aplicarmos entre 0 a múltiplas vezes (até 7 neste caso), o T -Gate sobre o qubit e medirmos o mesmo sobre o eixo X , teremos a seguinte tabela de resultados (figura 3.7):

Nota: Para obter estes resultados, os algoritmos foram corridos 1024 vezes e posteriormente calculado o valor médio.

A partir da porta T , podemos deduzir o conjunto de portas $\{Z, Y, S, S^t, T^t\}$:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} := T^2 \quad (3.10)$$

Experiment	Phase angle	Gates	Prop 0	Prob 1	X length
0	0		1	0	1
1	$\pi/4$	T	0.8535533	0.1464466	0.7071067
2	$\pi/2$	T^2	0.5	0.5	0
3	$3\pi/4$	T^3	0.1464466	0.8535533	-0.707106
4	π	T^4	0	1	-1
5	$5\pi/4$	T^5	0.8535533	0.1464466	0.7071067
6	$3\pi/2$	T^6	0.5	0.5	0
7	$7\pi/4$	T^7	0.1464466	0.8535533	-0.707106

Figura 3.7: Resultados da aplicação do T -Gate [18]

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} := T^4 \quad (3.11)$$

$$S^t = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} := T^6 \quad (3.12)$$

$$T^t = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\frac{\pi}{4}} \end{pmatrix} := T^7 \quad (3.13)$$

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} := XZ \quad (3.14)$$

Estas novas portas, nada mais são que generalizações de T , evitando o uso repetitivo da mesma para fazer elevados números de rotações.

3.6 Decoerência

Os computadores quânticos reais, têm de lidar com decoerência [15], que nada mais é do que perda de informação devido a ruído. Coerência é o tempo que a informação quântica de um qubit consegue permanecer num estado puro (sem ruído). Se o tempo de coerência for de $20ms$, significa que no intervalo de tempo

$[0; 20]ms$, a informação do qubit é coerente. Esta métrica é importante, pois determina o tempo que temos para efetuar operações sobre um qubit, antes deste começar a perder a sua informação quântica. A limitação temporal, limita consequentemente o número de operações que podemos efetuar sobre um qubit.

A Esfera de Bloch é um ótimo recurso para compreender e visualizar este fenómeno. Os estados base ou puros que foram vistos até agora, têm comprimento 1 (um) na Esfera, tocando assim na superfície da mesma. A decoerência, leva a que os estados puros passem a estados mistos. Visualmente, isto faz com que, o comprimento do vetor na Esfera seja menor que 1 (um).

3.6.1 Relaxamento de Energia e T_1

Um importante processo de decoerência é o relaxamento de energia [15], no qual, o estado excitado $|1\rangle$ vai degradando até ao estado não excitado $|0\rangle$. A constante tempo deste processo, T_1 , é uma figura extremamente importante para qualquer tipo de implementação de sistemas quânticos, tendo-nos levado ao tipo de protótipos de sistemas quânticos existentes.

Uma das formas de observarmos este fenómeno, é através da utilização de portas *Idle* (ou portas identidade), que não têm efeitos sobre os qubits (apenas se limitam a esperar que o qubit "passe" por elas). Verificamos nas medições, que apesar destas portas não terem qualquer tipo de efeito sobre os qubits, mesmo assim, o qubit tende gradualmente para o estado $|0\rangle$. Esta observação, confirma que de facto, existe degradação do estado excitado do qubit devido a ruído.

3.6.2 Dephasing e T_2

Desfasamento [15] é outro processo de decoerência, que contrariamente ao relaxamento, apenas afeta a sobreposição de estados. Este conceito, apenas pode ser compreendido num contexto quântico, visto que, não existe uma analogia clássica para o mesmo.

A constante tempo T_2 , inclui o efeito de desfasamento, bem como o de relaxamento. Também esta, é uma figura muito importante na atualidade da computação quântica, uma vez que o seu objetivo é a redução do ruído, que como já vimos é um grande obstáculo para a computação quântica. T_1 e T_2 são usadas pela IBM e outros centros de investigação quânticos, permitindo a criação de sistemas mais estáveis e com baixos níveis de ruído.

3.6.3 Processo de Decoerência com Qubits Super-condutores

Com o decorrer dos anos e do esforço científico da área não só da Computação, como também da Física, foi sendo possível aumentar os intervalos de tempo em que um sistema quântico se mantém coerente. Manter um sistema coerente e com baixos níveis de ruído tem uma importância extrema para o futuro da área. Na Física, um sistema diz-se coerente, se duas fontes, tiverem uma diferença de Fase constante, a mesma frequência e o mesmo formato de onda.

Assim, a coerência descreve todas as propriedades de correlação entre quantidades físicas de uma onda ou entre várias ondas. Apesar do progresso parecer lento, estes pequenos aumentos no intervalo de tempo, estão a permitir progressos extraordinários.

Na figura 3.8, é possível ver a relação entre T_2 *versus* tempo.

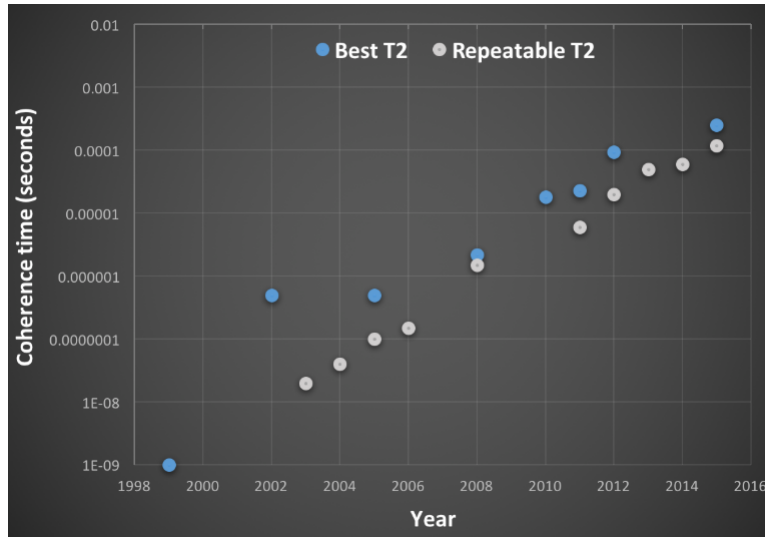


Figura 3.8: Evolução dos tempos de coerência ao longo dos anos [15]

3.7 Sistemas de N Qubits ($N > 1$)

Até agora, apenas vimos sistemas com um qubit. Vamos agora considerar sistemas com mais do que um qubit [19].

O espaço vetorial de um sistema com n qubits, é igual a 2^n e a base, é o conjunto de *strings* binárias, tais que: $k \in \{0, 2^n - 1\}$. Por exemplo, a base para dois qubits é:

$$\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\};$$

Para três,

$$\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\},$$

e assim sucessivamente.

Com o aumentar do valor n , aumenta também o número de termos. Este aumento ocorre de forma exponencial e como já foi referido no início deste capítulo, é esta exponencialidade que confere à computação quântica o seu poder computacional. Daí, ser impossível correr algoritmos quânticos em sistemas convencionais. Um computador convencional com n bits, tem 2^n configurações possíveis. Em qualquer instante t , o estado do computador, é uma e uma só das várias configurações. Por exemplo, um computador convencional pega num número de n bits (e.g. 00000) e efetua operações sobre os bits do mesmo, mapeando o input num estado intermédio de n bits (e.g. 00001) que posteriormente é passado como *output* como um outro número de n bits (e.g. 10101). De forma similar, um computador quântico, também pega num número de n bits e devolve um número de n bits. Contudo, devido à sobreposição e entrelaçamento, o estado intermédio deste, é bastante diferente do de um sistema convencional. Para descrever este valor intermédio, são necessários 2^n números complexos, que permitem um maior grau de liberdade.

Para fazer uso das várias configurações do "mundo quântico", precisamos de portas lógicas que façam operações entre qubit, ao invés de fazerem operações apenas sobre qubits individuais. O *Gate* 'condicional' criado pela IBM para o efeito é o CNOT (ou *Controller-NOT*) e pode ser visualmente representado da seguinte forma (figura 3.9):

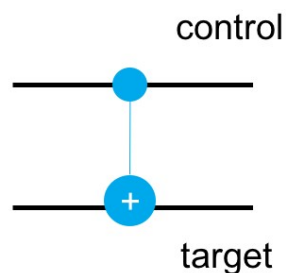


Figura 3.9: Representação gráfica de um CNOT [19]

Numa base clássica, a CNOT faz um *flip* ao estado do qubit alvo (*target*), caso o qubit de controlo (*control*) se encontre no estado $|1\rangle$, caso contrário, não faz nada. A CNOT atua como se fosse um XOR, mas contrariamente ao XOR, esta é

uma porta com dois outputs em vez de um, de forma a ser reversível (como todas as portas lógicas quânticas têm de ser (*ver secção 3.7.1*)). A *CNOT* pode ser representada pela matriz seguinte (matriz 3.15):

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.15)$$

3.7.1 Reversibilidade

Na Física clássica, o tempo é reversível [9]. Isto é, se gravarmos para um ficheiro um vídeo das moléculas de ar (a nível microscópico) a moverem-se no espaço, se invertêssemos o vídeo para ser visto de trás para a frente, ele continuaria a fazer sentido. As equações que descrevem o movimento estão corretas tanto para t como para $-t$.

Na Física quântica, existe o conceito de "*Unitary*" (ou Unitário) [29], que define que o ponto futuro é único e o ponto passado é único. Se não houver perda de informação durante a transição de uma configuração para a outra, então essa transição é única. Se existir uma lei que determina como se vai do ponto A para o B , então é possível deduzir uma lei que determine como se vai de B para A . Esta é uma restrição na evolução de sistemas quânticos, que garante que a soma de todos os resultados possíveis de um sistema, é sempre igual a 1 (um).

Para reproduzir as propriedades referidas, num sistema quântico, as portas lógicas do mesmo, têm de ser obrigatoriamente revertíveis [27].

Os computadores clássicos, também têm reversibilidade, contudo, usam um método diferente. Em vez de terem portas lógicas revertíveis, estes descartam a informação que já não é necessária. Deitar fora "lixo" durante operações quânticas é uma opção. Mas como a informação descartada conta como uma medição e como as medições tendem a quebrar algoritmos quânticos, este conceito nunca foi implementado nos sistemas quânticos.

3.8 Entrelaçamento e Testes de Bell

Uma das mais famosas e contra-intuitivas noções da Mecânica quântica é o entrelaçamento. A teoria do entrelaçamento, dita que dois sistemas que estejam entrelaçados, se influenciam um ao outro, mesmo que estejam anos luz de distância.

Uma das experiências que prova esta teoria é o teste de Bell [1]. Para provar a existência ou não de entrelaçamento e de um "Mundo Quântico", Bell começou por definir três pressupostos da Física clássica:

- **Realismo:** Determina que os objetos possuem características físicas, que mantêm independentemente de serem observadas ou não;
- **Localidade:** Afirma que nada pode influenciar algo que esteja suficientemente distante, de tal forma que o sinal entre os dois tivesse de exceder a velocidade da luz;
- **Liberdade de Escolha:** Físicos podem fazer as medições e observações que quiserem, sem que isso altere o estado do observado.

Se os testes de Bell obedecerem a estes três pressupostos, então viemos num mundo onde apenas a Física clássica existe. Contudo, já foi mostrado várias vezes que de facto os testes de Bell não obedecem ao pressupostos, significando que de facto, vivemos num mundo que no seu nível subatômico não consegue ser explicado apenas pela Física clássica. Está então provada a existência e veracidade da Física quântica, bem como do entrelaçamento.

3.9 Conclusões

Findo este capítulo, possuímos um campo de conhecimento, que apesar de ser básico para aquilo que é o campo da computação quântica, nos irá permitir dar o passo seguinte deste projeto. Os conceitos aqui expostos, irão revelar-se fundamentais nos capítulos seguintes, onde o presente documento, passa a um estilo de cariz mais prático, através da implementação e análise de um dos algoritmos quânticos mais relevantes.

Capítulo 4

O Algoritmo de Shor

4.1 Introdução

Vimos até agora como funciona um computador quântico, desde as teorias da Física quântica em que se baseia, até à forma como manipulamos qubits. Restamos agora, perceber como pegar na informação adquirida, para criar algoritmos [10] mais complexos.

Sabemos que a informação quântica é armazenada sob a forma de qubits e que esses qubit gozam das propriedades de sobreposição, entrelaçamento e interferência. Combinando os primeiros dois, podemos criar uma sobreposição de estados para todas as combinações possíveis de um problema que pretendemos resolver, em vez de termos de calcular as configurações possíveis uma a uma. Ao codificarmos um problema para um sistema quântico, estamos a aplicar uma Fase a cada um dos estados (a Fase de um qubit é uma rotação segundo o eixo Z da Esfera de Bloch, que é medida na amplitude de um ângulo α). Ao codificarmos uma Fase em cada um dos estados dos qubits, estes passam a ter o comportamento de uma onda. É aqui que entra o conceito de interferência. Quando as ondas estão "em fase" ou sincronizadas, as amplitudes somam-se e quando estão "fora de fase" cancelam-se. Podemos assim, usar a interferência para amplificar as respostas desejadas e anular as indesejadas, até chegarmos à solução.

Assim, quantos mais qubits tivermos, maior é o número de estados que podemos ter. Mas, um outro fator importante, é o ruído. Temos de ser capazes de controlar os qubits de forma a efetuarmos operações sobre os mesmos. Se a taxa de erro for muito elevada, as operações não correm como esperado e a solução atingida é inútil. Para lidar com esta relação entre o número de qubits e a taxa de ruído, foi criada uma métrica chamada de 'Volume Quântico', que dita, que quantos mais qubits tiver um sistema, maior é o seu poder computacional. No entanto, não podemos simplesmente aumentar o número de qubits num sistema, se a taxa

de erro for demasiado elevada.

Revistos estes conceitos fundamentais dos computadores quânticos, podemos começar a estudar os algoritmos quânticos e como implementá-los, usando neste caso, a ferramenta *Qiskit*.

4.2 Algoritmo de Shor

Um dos algoritmos mais relevantes, teorizados até hoje no campo da computação quântica, é o algoritmo de Shor. Este algoritmo, é capaz de acompanhar a exponencialidade [28] de um dos problemas mais complexos da atualidade: a fatorização de números primos (usada, por exemplo, na encriptação *Rivest-Shamir-Adleman* (RSA)).

Durante esta secção, veremos como funciona e como implementar este algoritmo.

Dado um número N , queremos saber, quais os dois números primos, tais que: $N = p \star q$. A existência de tais pares é garantida (pelo teorema fundamental da aritmética), o problema é que quanto maior for N , maior é a complexidade em descobrir p e q .

A criptografia RSA, usa atualmente a fatorização de primos para garantir a segurança de dados. Para um sistema convencional, calcular dois grandes números primos e multiplicá-los é fácil, no entanto, descobrir os dois fatores primos de um número de vários dígitos, requer muito tempo e capacidade computacional. O sucesso da implementação do algoritmo de Shor, de forma a fatorizar em tempo útil, determinará uma profunda reforma na forma como a criptografia funciona hoje em dia, visto que o método usado pela RSA se tornaria obsoleto. A única coisa que impede a implementação deste algoritmo, é o fraco poder computacional dos sistemas quânticos existentes.

A primeira propriedade em que Shor se baseou para o seu algoritmo, foi a Aritmética Modular de Euler, que define o seguinte (equação 4.1):

$$a \equiv x \bmod N \quad \text{se} \quad \frac{a}{N} \rightarrow \text{tem resto} = x \quad (4.1)$$

Consideremos os seguintes exemplos:

k	$3^k \bmod 10$	$2^k \bmod 7$		
1	3	2	$3^k \bmod 10$	$2^k \bmod 7$
2	9	4	\Downarrow	\Downarrow
3	7	1	período = 4	período = 3
4	1	2		
5	3	4		
...		

Verificamos que existe um padrão que se repete. O comprimento desse padrão é chamado de período (r). Se x e N forem coprimos (não têm fatores primos em comum), então, r é o número mais pequeno possível, tal que: $x^r \equiv 1 \bmod N$. Assim, podemos dividir o algoritmo de Shor em quatro etapas:

1. Escolher um número a menor que N ;

Para tal, temos de garantir que a e N são coprimos. Calculamos o maior múltiplo comum entre eles através do algoritmo Euclidiano. Se forem coprimos, então: $\gcd(a, N) = 1$ (\gcd : *greatest common divisor*).

2. Calcular o período de $a \bmod N$;

O período r , tem de ser obrigatoriamente par e a seguinte desigualdade tem de se verificar:

$$a^{\frac{r}{2}} + 1 \not\equiv 0 \bmod N$$

Se alguma das condições falhar, tem de ser escolhido outro valor a (a , tem pelo menos 50% de probabilidade de resultar num r , que verifique ambas as condições).

3. Fazer algumas transformações algébricas aos valores até aqui calculados;

$$\begin{aligned} a^r &\equiv 1 \bmod N \\ a^r - 1 &\equiv 0 \bmod N \quad (a^r - 1 \text{ é múltiplo de } N) \\ a^r - 1 &= k * N \end{aligned}$$

Dado que r é par:

$$\begin{aligned}(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) &= k * N \quad (\text{como } N = p * q) \\ (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) &= k * p * q\end{aligned}$$

Considerando, e.g. $N = 35$, $a = 8$ e $r = 4$, temos:

$$\begin{aligned}8^4 &\equiv 1 \mod 35 \\ 8^4 - 1 &\equiv 0 \mod 35 \\ (8^{\frac{4}{2}} - 1)(8^{\frac{4}{2}} + 1) &= k * p * q\end{aligned}$$

4. Resolver em ordem a p e q .

$$\begin{aligned}p &= \gcd(a^{\frac{r}{2}} - 1, N) \\ q &= \gcd(a^{\frac{r}{2}} + 1, N)\end{aligned}$$

No exemplo de $N = 35$, temos:

$$\begin{aligned}p &= \gcd(63, 35) = 7 \\ q &= \gcd(65, 35) = 5\end{aligned}$$

A grande complexidade deste algoritmo, reside no passo 2 (calcular o período). No entanto, problemas complexos como calcular o período de $x \mod N$, é precisamente o tipo de problemas nos quais os computadores quânticos são bons. Através da utilização da Transformada Quântica de Fourier (esta transformação é uma implementação quântica da Transformada de Fourier), conseguimos manipular as ondas das fases dos qubits, ampliando a resposta correta e cancelando as incorretas até chegarmos ao resultado. Esta transformação é conseguida através da manipulação dos qubits, por um conjunto de portas H (Hamiltoniano) e R_θ (operador \mod). Com esta transformação, conseguimos descobrir o r , exponencialmente mais rápido do que com algoritmos tradicionais, tirando proveito do conceito de sobreposição.

4.3 Implementação do Algoritmo de Shor

Agora que sabemos como se processa o algoritmo de Shor, podemos começar a sua implementação, usado *Python* e *Qiskit*. De notar, que a implementação deste algoritmo é apenas experimental e conta com bastantes otimizações derivadas de conhecimento *apriori*. Regra geral, são precisos $2 + \frac{3}{2} \log_2 N$ qubits, para fatorizar um número N , o que significa que seriam necessários 1154 qubits para calcular RSA – 768 (o maior número fatorizado até hoje).

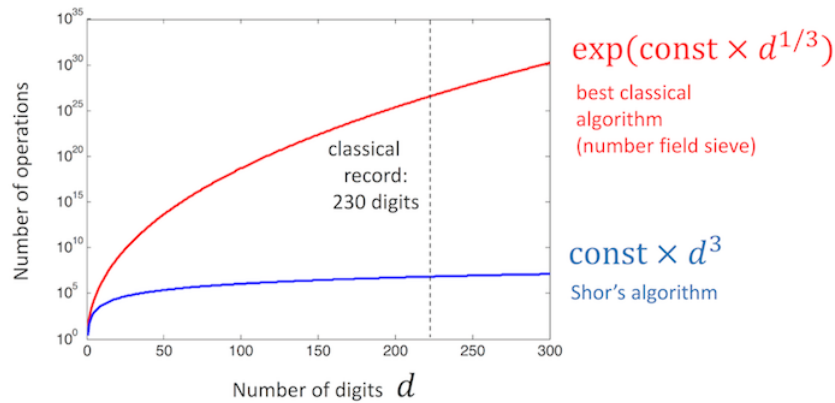


Figura 4.1: Algoritmos de fatorização clássicos Vs algoritmo de Shor [20]

A implementação do algoritmo de Shor, presente a seguir, é da autoria de Miguel Sozinho Ramalho. Mais detalhes sobre a implementação do algoritmo e a justificação matemática por detrás das otimizações, podem ser vistas na referência [11] da bibliografia.

A implementação do algoritmo de Shor, pode ser vista no apêndice A.1.

4.3.1 Resultados

Para os valores $a = \{2, 7, 8, 13\}$, temos $r = 4$ e para $a = 11$, temos $r = 2$.

Para determinar estes valores, temos de calcular o múltiplo comum dos resultados obtidos (figuras 4.2, 4.3 e 4.4 (2 para $a = \{2, 7, 8, 13\}$ e 4 para $a = 11$)). Temos para $a = \{2, 7, 8, 13\}$, que $\frac{M}{r} = 2$ e para $a = 11$, $\frac{M}{r} = 4$. $M = 2^k$ (k é o número de qubits usados na Transformada de Fourier), como o número de qubits é 3, então, $M = 2^3 = 8$. Posto isto, para $a = \{2, 7, 8, 13\}$, temos $\frac{8}{r} = 2 \rightarrow r = 4$ e para $a = 11$, temos $\frac{8}{r} = 4 \rightarrow r = 2$.

Agora que temos os valores $N(15)$, a (e.g. 7) e $r(4)$, podemos calcular p e q . $p = \gcd(a^{\frac{r}{2}} - 1, N) = \gcd(48, 15) = 3$ e $q = \gcd(a^{\frac{r}{2}} + 1, N) = \gcd(50, 15) = 5$.

$p * q = 3 * 5 = 15 = N$, estão assim, determinados os fatores de $N = 15$.

As figuras seguintes, apresentam os gráficos gerados pelo algoritmo de Shor.

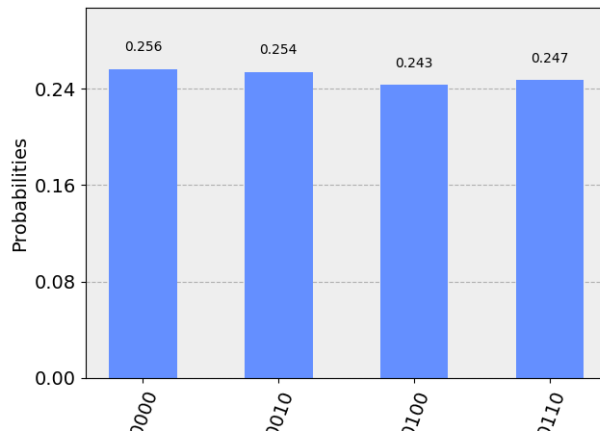


Figura 4.2: Resultados do algoritmo de Shor, para $a = 2$

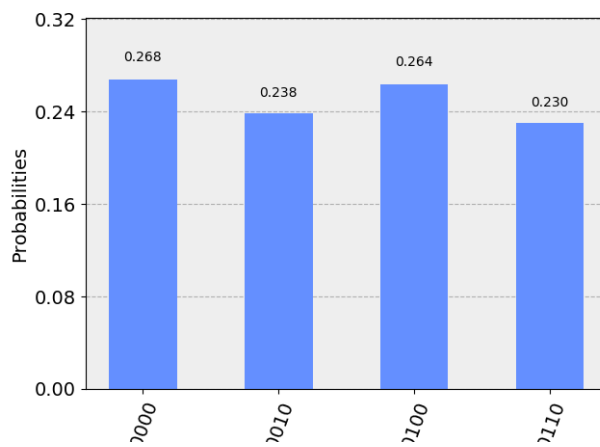


Figura 4.3: Resultados do algoritmo de Shor, para $a = 7$

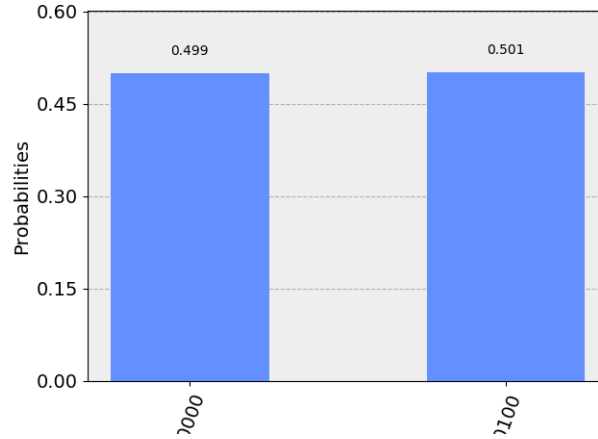


Figura 4.4: Resultados do algoritmo de Shor, para $a = 11$

O circuito gerado, pela implementação do algoritmo acima, é o circuito presente na figura 4.5.

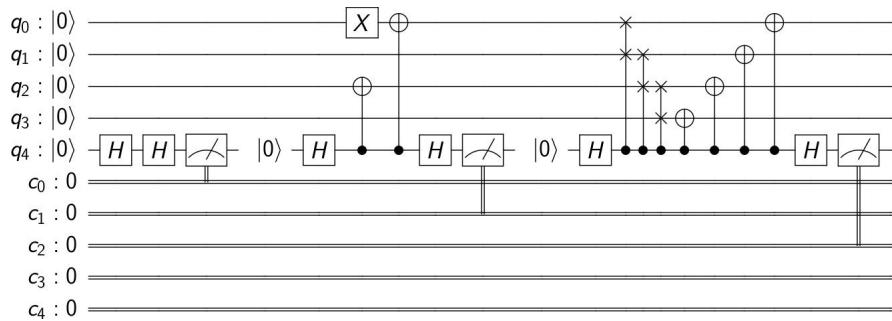


Figura 4.5: Circuito gerado por Qiskit, do algoritmo de Shor [20]

4.4 Conclusões

Como já foi referido anteriormente, na secção 4.3, esta implementação conta com otimizações de conhecimento *a priori* dos resultados. Como tal, esta é uma implementação com relevância estritamente académica. Para o caso em questão, seria mais rápido utilizar um algoritmo tradicional do que o de Shor. Contudo, demonstra que, é de facto possível implementar o algoritmo de Shor num computador quântico e que com o evoluir desta tecnologia, iremos, de facto, conseguir fatorizar números que até então eram impossíveis.

O sucesso da implementação deste e de outros pequenos algoritmos, abrem a porta para um futuro de algoritmos quânticos, demonstrando que de facto, é possível tirarmos partido dos fenómenos quânticos no campo da computação.

"The future is quantum"
IBMQ

Capítulo 5

Redes Neuronais Quânticas

5.1 Introdução

As Redes Neuronais [12], são a peça central dos algoritmos de Inteligência Artificial. Inspiradas pelo sistema neural humano, as Redes Neuronais (RNs), são a evolução do perceptron, proposto por Rosenblatt em 1957. Ao seu nível mais básico, um neurónio [4] de uma Rede Neuronal (RN), aceita um conjunto de entradas x_i e devolve uma saída y . A função de ativação, recebe a soma pesada das entradas e consoante o valor dessa soma, devolve uma saída. Quando os neurónios são agrupados, ligando a saída de uma camada de neurónios, às entradas da camada seguinte, obtemos uma Rede Neuronal. Quanto mais camadas tiver uma rede, maior é a capacidade que esta tem de processar dados. Através do ajuste dos pesos, é possível adaptar a rede a diversos problemas que seja necessário resolver. Esta flexibilidade das Redes Neuronais, é o que as torna tão apropriadas para o uso em Inteligência Artificial [5] [26].

A quantidade de dados que têm de ser processados hoje em dia é enorme, graças à facilidade de uso da Internet, exigindo RNs cada vez mais complexas. Os algoritmos utilizados atualmente são complexos e exigem demasiado dos CPU's e GPU's que dispomos atualmente. Os computadores quânticos, têm vindo a mostrar-se como o tipo de sistema mais adequado para correr Redes Neuronais [13], devido ao poder computacional que é teorizado para os sistemas quânticos mais avançados do futuro. É expectável, que haja grandes melhorias dos sistemas quânticos em relação aos tradicionais, sobretudo na fase de aprendizagem dos algoritmos, visto que graças ao entrelaçamento quântico, é possível ajustar os pesos de forma paralela. De facto, as capacidades intrínsecas da mecânica quântica, para representar grandes conjuntos de dados sobre a forma de vetores e matrizes e a capacidade de efetuar operações lineares sobre essas matrizes e vetores, resulta num poder computacional que consegue fazer par com a quantidade

de poder computacional requerido pelas Redes Neurais mais complexas. Existe assim, uma crescente vontade e ambição em criar RNQs [8] [3]. Neste capítulo, iremos discutir, o que é necessário para implementar uma Rede Neuronal quântica (RNQ) e quais as vantagens e desvantagens em relação a implementações de RNs em sistemas tradicionais.

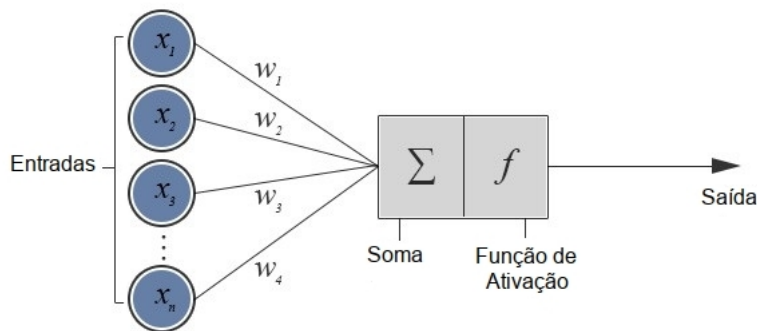


Figura 5.1: Neurónio [12]

5.2 Como Implementar uma Rede Neuronal Quântica

Apesar dos esforços já realizados, tanto por grandes empresas, como por investigadores individuais [30], ainda não existe uma forma geral para criar Redes Neurais em sistemas quânticos. Como já foi referido, apesar das vantagens dos sistemas quânticos, estes também tem as suas desvantagens, nomeadamente o ruído e os curtos intervalos de coerência. Visto que as RNs, lidam com grandes quantidades de dados, perder informação sobre esses dados, resulta inevitavelmente em resultados finais que não correspondem ao que se pretende, levando a uma enorme perda de tempo computacional durante o processo. Por outro lado, ainda são poucos os computadores quânticos existentes, diminuindo assim a quantidade de teorias e algoritmos que se podem testar. Este último ponto, infelizmente impede a implementação de uma RNQ com os recursos disponíveis durante a elaboração deste projeto. Com apenas cinco qubits, é de todo impossível criar de raiz uma Rede Neuronal nos computadores disponibilizados pela IBM, sem proceder a otimizações baseadas em conhecimento *a priori*, como aconteceu com o exemplo do Algoritmo de Shor no capítulo 4.

Segundo a *Noisy Intermediate-scale Quantum* (NISQ) [25] (uma biblioteca criada com o intuito de facilitar a investigação de Inteligência Artificial em sistemas quânticos), para implementar uma RN, para dados clássicos num computador quântico, é necessário seguir os seguintes cinco passos:

- Codificação dos dados: Codificar os dados "clássicos", em tipos de dados adequados a um circuito quântico;
- Preparação de estado: Implementar um circuito quântico, para preparar o sistema para o conjunto de dados que vai receber;
- Evolução Unitária: Implementar um conjunto de *gates* parametrizados, para transformar os dados;
- Medição: Medir o estado dos dados quânticos, após a evolução unitária reverter os dados à sua forma "clássica"; neste ponto pode efetuar-se processamento clássico, para introduzir não-linearidade;
- Treino: Definir a função de custo e otimizar os parâmetros do unitário, de forma a minimizar os custos.

Pode-se concluir, que o maior desafio está em combinar a não-linearidade das Redes Neurais, com a linearidade e a dinâmica unitária dos sistemas quânticos. Este, tem sido o maior entrave para chegar a um procedimento que permita juntar os benefícios de ambas as partes.

Neste projeto, um dos grandes entraves a chegar ao objetivo, foi o de não saber como se codificam os dados de forma a serem interpretados por um sistema quântico. Como ainda não existe um consenso e/ou um método comum, cada implementação é diferente da seguinte, impedindo a aquisição de conhecimento suficiente a este ponto, para criar um algoritmo de codificação.

5.3 Redes Neurais Clássicas Vs Quânticas

Como já foi mencionado várias vezes no decorrer deste documento, a computação quântica abre as portas a uma nova realidade sem precedentes na história da computação. As capacidades destes, estendem-se muito além do que os sistemas clássicos nos permitem. Tendo como o exemplo o algoritmo de Shor, este, permite uma solução polinomial para o problema da fatorização, algo impensável antes da introdução dos computadores quânticos. É fácil compreender os motivos pelos quais as RNQs nos vão trazer vantagens sobre as implementações em sistemas de Von Neumann. Espera-se que com a evolução na área, se verifiquem algumas das seguintes vantagens [2]:

- Capacidade de memória exponencial;
- Maior performance para números menores de camadas ocultas em RNs;
- Aprendizagem mais rápida;
- Maior velocidade de processamento;
- Maior estabilidade e confiabilidade dos resultados.

Estas são algumas das potenciais vantagens no futuro, e certamente que o futuro irá revelar ainda mais.

5.4 Conclusões

Já muito trabalho e investigação foi realizado no âmbito das Redes Neurais em sistemas quânticos. Os próximos anos, irão certamente ser repletos de novas descobertas das potencialidades destes sistemas, com características tão peculiares. A ser provado o que se tem vindo a teorizar, os computadores quânticos irão marcar uma nova etapa na história da computação e da forma como nós resolvermos problemas e redescobrimos o que nos rodeia.

Apesar de ambicioso, o trabalho que é possível realizar no âmbito deste projeto, fica aquém do objetivo, não tendo sido possível criar um algoritmo para um neurónio quântico. No entanto, e apesar de não ser esse o objetivo último, foi possível a implementação de uma Rede Bayesiana quântica simples, à qual se dedica o capítulo 6.

Capítulo 6

Redes Bayesianas Quânticas

6.1 Introdução

Uma Rede Bayesiana é um modelo probabilístico, que usa inferência bayesiana para computação de probabilidades. Fortemente baseadas em dependência condicional, são um ótimo recurso para calcular probabilidades de algo que pode acontecer, com base no que já aconteceu. Um dos exemplos mais usados, no âmbito das RNs, sobretudo no âmbito acadêmico, é o "*Wet Grass*", que calcula a probabilidade de a relva estar molhada, com base em três variáveis (Enublado, Chuva, Aspersores). Neste capítulo, iremos ver a implementação deste exemplo simples, como problema quântico e clássico, comparando os resultados. Ambas as implementações seguintes, são adaptações dos algoritmos implementados por Henning Dekant, Henry Tregillus, Robert Tucci e Tao Yin. Mais detalhes sobre a implementação do algoritmo, podem ser vistas na referência [7] da bibliografia. Foi usado o mesmo conjunto de dados em ambos os exemplos (uma lista de 2000 entradas, com as variáveis: enublado, chuva, aspersor, relva molhada, em que a cada variável era atribuído o valor 1 (um) caso se verificasse, e 0 (zero) caso não se verificasse).

6.2 Rede Bayesiana Clássica

A implementação da Rede Bayesiana clássica, pode ser vista no apêndice A.2. Os resultados obtidos ao correr esse código foram:

```
antonio@LinuxPc:~/Desktop/wetgrass
$ python3 WetGrass.py
Cloudy
brute engine: ['Cloudy']
[0.58488387 0.41511613]

Rain
brute engine: ['Rain']
[0.30129603 0.69870397]

Sprinkler
brute engine: ['Sprinkler']
[0.30129603 0.69870397]

WetGrass
brute engine: ['WetGrass']
[0. 1.]
```

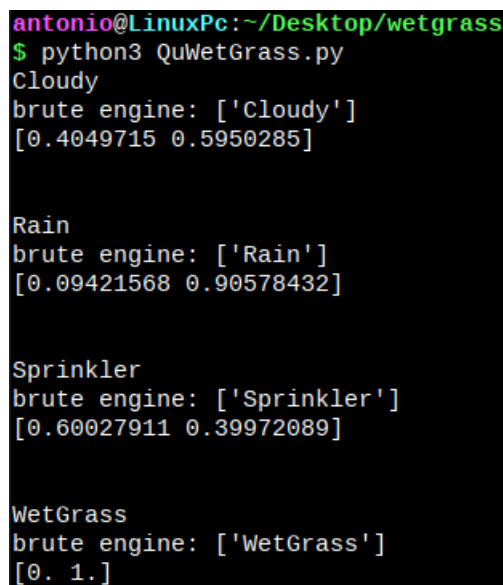
Figura 6.1: Output WetGrass.py

```
1 network unknown {
2
3 variable Rain {
4 type discrete [ 2 ] { state0 , state1 };
5 }
6 variable Cloudy {
7 type discrete [ 2 ] { state0 , state1 };
8 }
9 variable Sprinkler {
10 type discrete [ 2 ] { state0 , state1 };
11 }
12 variable WetGrass {
13 type discrete [ 2 ] { state0 , state1 };
14 }
15 probability ( Rain | Cloudy ) {
16 (state0) 0.4, 0.6;
17 (state1) 0.5, 0.5;
18 }
19 probability ( Cloudy ) {
20 table 0.5, 0.5;
21 }
22 probability ( Sprinkler | Cloudy ) {
23 (state0) 0.2, 0.8;
24 (state1) 0.7, 0.3;
25 }
26 probability ( WetGrass | Sprinkler , Rain ) {
```

```
27 (state0 , state0) 0.99, 0.01;  
28 (state1 , state0) 0.01, 0.99;  
29 (state0 , state1) 0.01, 0.99;  
30 (state1 , state1) 0.01, 0.99;  
31 }  
32 }
```

6.3 Rede Bayesiana Quântica

A implementação da Rede Bayesiana Quântica, pode ser vista no apêndice A.3. Os resultados obtidos ao correr esse código foram:



```
antonio@LinuxPc:~/Desktop/wetgrass  
$ python3 QuWetGrass.py  
Cloudy  
brute engine: ['Cloudy']  
[0.4049715 0.5950285]  
  
Rain  
brute engine: ['Rain']  
[0.09421568 0.90578432]  
  
Sprinkler  
brute engine: ['Sprinkler']  
[0.60027911 0.39972089]  
  
WetGrass  
brute engine: ['WetGrass']  
[0. 1.]
```

Figura 6.2: Output QuWetGrass.py

```
1 network unknown {  
2  
3 variable Cloudy {  
4 type discrete [ 2 ] { state0 , state1 };  
5 }  
6 variable WetGrass {  
7 type discrete [ 2 ] { state0 , state1 };  
8 }  
9 variable Rain {  
10 type discrete [ 2 ] { state0 , state1 };  
11 }  
12 variable Sprinkler {
```

```

13 type discrete [ 2 ] {state0 , state1 };
14 }
15 probability ( Cloudy ) {
16     table 0.70014+0.140028j , 0.70014+0.j ;
17 }
18 probability ( WetGrass | Sprinkler , Rain ) {
19     (state0 , state0) 0.999898+0.j , 0.010100-0.0101j ;
20     (state1 , state0) 0.0039936+0.j , 0.3953616+0.9185169j ;
21     (state0 , state1) 0.0019619-0.9809543j , 0.1942289+0.j ;
22     (state1 , state1) 0.0071063+0.7106332j , 0.7035268+0.j ;
23 }
24 probability ( Rain | Cloudy ) {
25     (state0) 0.3980149+0.j , 0.5970223-0.696526j ;
26     (state1) 0.6454972-0.1290994j , 0.6454972+0.3872983j ;
27 }
28 probability ( Sprinkler | Cloudy ) {
29     (state0) 0.2073903+0.5184758j , 0.8295614+0.j ;
30     (state1) 0.2152064+0.9223132j , 0.0922313-0.3074377j ;
31 }
32 }

```

6.4 Comparação dos Resultados

Os resultados impressos para o terminal são as probabilidades simples, com base nos dados recebidos. Os resultados escritos para os ficheiros, são os que de facto importam neste caso, pois contêm as probabilidades calculadas a partir de inferência bayesiana. Em ambos os casos, quântico e clássico, os resultados obtidos pelas redes foram semelhantes. As pequenas discrepâncias entre os resultados de ambas as implementações, devem-se à simulação dos erros naturais dos sistemas quânticos e a erros de arredondamento.

Como é óbvio, a dimensão da rede e dos dados é muito reduzida, impossibilitando assim, ver as diferenças nos tempos de processamento entre ambas as implementações. Apenas nos permite ver, que, de facto, a implementação quântica está correta, pois produz os mesmos resultados que implementações clássicas já existentes.

6.5 Conclusões

Tal como a implementação do algoritmo de Shor no capítulo 4, ambas as implementações deste capítulo, não têm utilidade em contexto real. Simplesmente, servem para demonstrar que de facto é possível implementar algoritmos quânti-

cos que obtêm os mesmos resultados que os algoritmos já existentes, no entanto, fazem-no de forma mais rápida e eficiente.

Capítulo 7

Conclusões e Trabalho Futuro

7.1 Conclusões Principais

Com pouco mais de um século de existência, a Física quântica, rompe com as noções clássicas que tínhamos sobre o universo. De facto, o seu impacto, foi e continua a ser tal, que levou à criação de vários novos ramos científicos. Um deles, é o ramo da Computação quântica, que ambiciona a criação de verdadeiros computadores quânticos, capazes de processar informação a velocidades sem precedentes. Ao fazerem uso de conceitos da Mecânica quântica, estes sistemas, permitem fazer coisas que até aqui eram impensáveis. Algoritmos que hoje demoram anos, poderão vir a ser resolvidos em segundos, abrindo as portas a avanços tecnológicos e científicos, que são impossíveis com os sistemas que usamos hoje em dia.

O fundamento da Computação quântica é simples. Se os sistemas que temos hoje em dia, não nos permitem resolver problemas que passem de uma certa complexidade, temos de criar algo, que consiga fazer par à complexidade desses problemas. Contudo, a Computação quântica, não irá ditar o fim da Computação clássica. Em tarefas do dia-a-dia, os computadores atuais são melhores que os quânticos. Estes últimos, apenas servem para resolver problemas que vão além das capacidades dos primeiros. Em instruções individuais, a computação atual, mantém a liderança. É apenas em casos em que seja necessário fazer muitos cálculos e comparações entre casos, que os sistemas quânticos brilham, visto que têm uma enorme facilidade em processar informação em paralelo, em vez de sequencial, como acontecia até aqui.

Este é o começo de uma nova Era para a Computação e para a ciência. Certamente, que o futuro trará novas descobertas, que nos irão permitir usar ao máximo as capacidades destes sistemas, e que sabe, responder às perguntas mais complexas com quem nos deparamos atualmente.

7.2 Trabalho Futuro

Chegado ao fim este projeto, fica por implementar uma Rede Neuronal quântica, ou até mesmo um simples neurónio da mesma. A complexidade, tanto da Física como da Computação quântica, estende-se para além do que é possível aprender e fazer em apenas um semestre. Sendo, também uma área muito recente, ainda não existe um consenso geral de como proceder com estes sistemas. Estamos ainda numa fase embrionária, marcada por muita investigação, de pessoas e empresas que estão a tentar dar os primeiros passos nesta área.

Com o passar do tempo e com o emergir de convenções mais generalistas e ferramentas bem otimizadas, certamente que a programação de sistemas quânticos, irá ser tão vulgar como programar sistemas tradicionais em *Python* ou *Java*.

Como trabalho futuro, fica a continuidade do estudo daquilo que é a Computação quântica, acompanhando os progressos da área, até me ser possível implementar uma RNQ.

Apêndice A

Algoritmos

A.1 Algoritmo de Shor

```
1 from qiskit import Aer
2 from qiskit import QuantumCircuit, ClassicalRegister,
  QuantumRegister
3 from qiskit import execute, compile
4 from qiskit.tools.visualization import plot_histogram,
  circuit_drawer
5
6 # qc = quantum circuit, qr = quantum register, cr = classical
  register, a = 2, 7, 8, 11 or 13
7 def circuit_amod15(qc, qr, cr, a):
8     if a == 2:
9         qc.cswap(qr[4], qr[3], qr[2])
10        qc.cswap(qr[4], qr[2], qr[1])
11        qc.cswap(qr[4], qr[1], qr[0])
12    elif a == 7:
13        qc.cswap(qr[4], qr[1], qr[0])
14        qc.cswap(qr[4], qr[2], qr[1])
15        qc.cswap(qr[4], qr[3], qr[2])
16        qc.cx(qr[4], qr[3])
17        qc.cx(qr[4], qr[2])
18        qc.cx(qr[4], qr[1])
19        qc.cx(qr[4], qr[0])
20    elif a == 8:
21        qc.cswap(qr[4], qr[1], qr[0])
22        qc.cswap(qr[4], qr[2], qr[1])
23        qc.cswap(qr[4], qr[3], qr[2])
24    elif a == 11: # this is included for completeness
25        qc.cswap(qr[4], qr[2], qr[0])
26        qc.cswap(qr[4], qr[3], qr[1])
27        qc.cx(qr[4], qr[3])
```

```

28         qc.cx(qr[4], qr[2])
29         qc.cx(qr[4], qr[1])
30         qc.cx(qr[4], qr[0])
31     elif a == 13:
32         qc.cswap(qr[4], qr[3], qr[2])
33         qc.cswap(qr[4], qr[2], qr[1])
34         qc.cswap(qr[4], qr[1], qr[0])
35         qc.cx(qr[4], qr[3])
36         qc.cx(qr[4], qr[2])
37         qc.cx(qr[4], qr[1])
38         qc.cx(qr[4], qr[0])
39
40 # qc = quantum circuit, qr = quantum register, cr = classical
    register, a = 2, 7, 8, 11 or 13
41 def circuit_aperiod15(qc, qr, cr, a):
42     if a == 11:
43         circuit_11period15(qc, qr, cr)
44         return
45
46     # Initialize q[0] to |1>
47     qc.x(qr[0])
48
49     # Apply a**4 mod 15
50     qc.h(qr[4])
51     # controlled identity on the remaining 4 qubits, which is
    equivalent to doing nothing
52     qc.h(qr[4])
53     # measure
54     qc.measure(qr[4], cr[0])
55     # reinitialise q[4] to |0>
56     qc.reset(qr[4])
57
58     # Apply a**2 mod 15
59     qc.h(qr[4])
60     # controlled unitary
61     qc.cx(qr[4], qr[2])
62     qc.cx(qr[4], qr[0])
63     # feed forward
64     if cr[0] == 1:
65         qc.u1(math.pi/2., qr[4])
66     qc.h(qr[4])
67     # measure
68     qc.measure(qr[4], cr[1])
69     # reinitialise q[4] to |0>
70     qc.reset(qr[4])
71
72     # Apply a mod 15
73     qc.h(qr[4])
74     # controlled unitary.

```

```

75     circuit_amod15(qc,qr,cr,a)
76     # feed forward
77     if cr[1] == 1:
78         qc.u1(math.pi/2.,qr[4])
79     if cr[0] == 1:
80         qc.u1(math.pi/4.,qr[4])
81     qc.h(qr[4])
82     # measure
83     qc.measure(qr[4],cr[2])
84
85
86 def circuit_11period15(qc,qr,cr):
87     # Initialize q[0] to |1>
88     qc.x(qr[0])
89
90     # Apply a**4 mod 15
91     qc.h(qr[4])
92     # controlled identity on the remaining 4 qubits, which is
93     # equivalent to doing nothing
94     qc.h(qr[4])
95     # measure
96     qc.measure(qr[4],cr[0])
97     # reinitialise q[4] to |0>
98     qc.reset(qr[4])
99
100    # Apply a**2 mod 15
101    qc.h(qr[4])
102    # controlled identity on the remaining 4 qubits, which is
103    # equivalent to doing nothing
104    # feed forward
105    if cr[0] == 1:
106        qc.u1(math.pi/2.,qr[4])
107    qc.h(qr[4])
108    # measure
109    qc.measure(qr[4],cr[1])
110    # reinitialise q[4] to |0>
111    qc.reset(qr[4])
112
113    # Apply 11 mod 15
114    qc.h(qr[4])
115    # controlled unitary.
116    qc.cx(qr[4],qr[3])
117    qc.cx(qr[4],qr[1])
118    # feed forward
119    if cr[1] == 1:
120        qc.u1(math.pi/2.,qr[4])
121    if cr[0] == 1:
122        qc.u1(math.pi/4.,qr[4])
123    qc.h(qr[4])

```

```

122     # measure
123     qc.measure(qr[4], cr[2])
124
125
126 q = QuantumRegister(5, 'q')
127 c = ClassicalRegister(5, 'c')
128
129 shor = QuantumCircuit(q, c)
130 circuit_aperiod15(shor, q, c, a) #substituir a por 2, 7, 8, 11 ou
    13
131
132 backend = Aer.get_backend('qasm_simulator')
133 sim_job = execute([shor], backend)
134 sim_result = sim_job.result()
135 sim_data = sim_result.get_counts(shor)
136 print(sim_data)
137 plot_histogram(sim_data).savefig("out.png")
138
139 im = circuit_drawer(shor)
140 im.save("circuitShor.pdf", "PDF")

```

A.2 Rede Bayesiana Clássica

```

1 import numpy as np
2
3 from nodes.BayesNode import *
4 from graphs.BayesNet import *
5 from potentials.DiscreteUniPot import *
6 from potentials.DiscreteCondPot import *
7 from inference.EnumerationEngine import *
8 from inference.MCMC_Engine import *
9 from inference.JoinTreeEngine import *
10
11
12 class WetGrass:
13
14     @staticmethod
15     def build_bnet():
16
17         cl = BayesNode(0, name="Cloudy")
18         sp = BayesNode(1, name="Sprinkler")
19         ra = BayesNode(2, name="Rain")
20         we = BayesNode(3, name="WetGrass")
21
22         we.add_parent(sp)
23         we.add_parent(ra)
24         sp.add_parent(cl)
25         ra.add_parent(cl)

```

```

26
27     nodes = {cl, ra, sp, we}
28
29     cl.potential = DiscreteUniPot(False, cl) # P(a)
30     sp.potential = DiscreteCondPot(False, [cl, sp]) # P(b|
a)
31     ra.potential = DiscreteCondPot(False, [cl, ra])
32     we.potential = DiscreteCondPot(False, [sp, ra, we])
33
34     # in general
35     # DiscreteCondPot(False, [y1, y2, y3, x]) refers to P(x|
y1, y2, y3)
36     # off = 0
37     # on = 1
38
39     cl.potential.pot_arr[:, :] = [.5, .5]
40
41     ra.potential.pot_arr[1, :] = [.5, .5]
42     ra.potential.pot_arr[0, :] = [.4, .6]
43
44     sp.potential.pot_arr[1, :] = [.7, .3]
45     sp.potential.pot_arr[0, :] = [.2, .8]
46
47     we.potential.pot_arr[1, 1, :] = [.01, .99]
48     we.potential.pot_arr[1, 0, :] = [.01, .99]
49     we.potential.pot_arr[0, 1, :] = [.01, .99]
50     we.potential.pot_arr[0, 0, :] = [.99, .01]
51
52     return BayesNet(nodes)
53
54
55 if __name__ == "__main__":
56     def main():
57         bnet = WetGrass.build_bnet()
58         brute_eng = EnumerationEngine(bnet)
59         # introduce some evidence
60         bnet.get_node_named("WetGrass").active_states = [1]
61         node_list = brute_eng.bnet_ord_nodes
62         brute_pot_list = brute_eng.get_unipot_list(node_list)
63
64
65         for k in range(len(node_list)):
66             print(node_list[k].name)
67             print("brute engine:", brute_pot_list[k])
68             print("\n")
69
70         bnet.write_bif('/home/antonio/Desktop/wetgrass /
examples_cbnets/WetGrass.bif', False)
71         bnet.write_dot('/home/antonio/Desktop/wetgrass /

```

```

72     examples_cbnets / WetGrass . dot ' )
       main ()

```

A.3 Rede Bayesiana Quântica

```

1  import numpy as np
2
3  from nodes . BayesNode import *
4  from graphs . BayesNet import *
5  from potentials . DiscreteUniPot import *
6  from potentials . DiscreteCondPot import *
7  from inference . EnumerationEngine import *
8  from inference . MCMC_Engine import *
9  from inference . JoinTreeEngine import *
10
11
12  class QuWetGrass :
13
14      @staticmethod
15      def build_bnet () :
16
17          cl = BayesNode ( 0 , name = " Cloudy " )
18          sp = BayesNode ( 1 , name = " Sprinkler " )
19          ra = BayesNode ( 2 , name = " Rain " )
20          we = BayesNode ( 3 , name = " WetGrass " )
21
22          we . add_parent ( sp )
23          we . add_parent ( ra )
24          sp . add_parent ( cl )
25          ra . add_parent ( cl )
26
27          nodes = { cl , ra , sp , we }
28
29          cl . potential = DiscreteUniPot ( True , cl ) # P ( a )
30          sp . potential = DiscreteCondPot ( True , [ cl , sp ] ) # P ( b | a
31      )
32          ra . potential = DiscreteCondPot ( True , [ cl , ra ] )
33          we . potential = DiscreteCondPot ( True , [ sp , ra , we ] )
34
35          # in general
36          # DiscreteCondPot ( True , [ y1 , y2 , y3 , x ] ) refers to A ( x |
37          y1 , y2 , y3 )
38          # off = 0
39          # on = 1
40
41          cl . potential . pot_arr [ : ] = [ .5 + .1 j , .5 ]
42
43          ra . potential . pot_arr [ 1 , : ] = [ .5 - .1 j , .5 + .3 j ]

```

```

42     ra.potential.pot_arr[0, :] = [.4, .6 - .7j]
43
44     sp.potential.pot_arr[1, :] = [.7 + 3.j, .3 - 1.j]
45     sp.potential.pot_arr[0, :] = [.2 + .5j, .8]
46
47     we.potential.pot_arr[1, 1, :] = [.01 + 1j, .99]
48     we.potential.pot_arr[1, 0, :] = [.01 - 5.j, .99]
49     we.potential.pot_arr[0, 1, :] = [.01, .99 + 2.3j]
50     we.potential.pot_arr[0, 0, :] = [.99, .01 - .01j]
51
52     cl.potential.normalize_self()
53     ra.potential.normalize_self()
54     sp.potential.normalize_self()
55     we.potential.normalize_self()
56
57     return BayesNet(nodes)
58
59 if __name__ == "__main__":
60     def main():
61         bnet = QuWetGrass.build_bnet()
62         brute_eng = EnumerationEngine(bnet, is_quantum=True)
63         # introduce some evidence
64         bnet.get_node_named("WetGrass").active_states = [1]
65         node_list = brute_eng.bnet_ord_nodes
66         brute_pot_list = brute_eng.get_unipot_list(node_list)
67
68
69         for k in range(len(node_list)):
70             print(node_list[k].name)
71             print("brute engine:", brute_pot_list[k])
72             print("\n")
73     main()

```


Bibliografia

- [1] Laura Dattaro. The quest to test quantum entanglement, 2018. <https://www.symmetrymagazine.org/article/the-quest-to-test-quantum-entanglement>. Último acesso a 8 de Abril de 2019.
- [2] Alexandr A. Ezhov e Dan Ventura. Quantum neural networks, 2018. <http://axon.cs.byu.edu/papers/ezhov.fdisis00.pdf>. Último acesso a 1 de Junho de 2019.
- [3] Xanadu e Strawberryfields. Continuous-variable quantum neural networks, 2018. <https://github.com/arturml/quantum-neural-network>. Último acesso a 5 de Maio de 2019.
- [4] Dario Gerace e Daniele Bajoni Francesco Tacchino, Chiara Macchiavello. An artificial neuron implemented on an actual quantum processor, 2019. <https://www.nature.com/articles/s41534-019-0140-4>. Último acesso a 3 de Junho de 2019.
- [5] Cristian Romero García. Quantum machine learning, 2016. <https://www.qutisgroup.com/wp-content/uploads/2014/10/TFG-Cristian-Romero.pdf>. Último acesso a 29 de Maio de 2019.
- [6] Daniel Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation, 2009. <https://arxiv.org/pdf/0904.2557.pdf>. Último acesso a 14 de Fevereiro de 2019.
- [7] Robert Tucci e Tao Yin Henning Dekant, Henry Tregillus. Python tools for analyzing both classical and quantum bayesian networks, 2019. <https://medium.com/technologymadeeasy/for-dummies-the-introduction-to-neural-networks-we-all-need-c50f6012d5eb>. Último acesso a 20 de Maio de 2019.
- [8] Juan Miguel Arrazola Maria Schuld Nicolás Quesada e Seth Lloyd Nathan Killoran, Thomas R. Bromley. Continuous-variable quantum

- neural networks, 2019. <https://github.com/XanaduAI/quantum-neural-networks?files=1>. Último acesso a 5 de Maio de 2019.
- [9] Nadeem Rojenko e Bruce Thomson Patrick Hochstenbach. What is (in simple terms) time reversibility in quantum mechanics, 2015. <https://www.quora.com/What-is-in-simple-terms-time-reversibility-in-quantum-mechanics>. Último acesso a 2 de Abril de 2019.
- [10] Scott Pakin Adetokunbo Adedoyin John Ambrosiano Petr Anisimov William Casper-Gopinath Chennupati Carleton Coffrin Hristo Djidjev David Gunter Satish Karra Nathan Lemons Shizeng Lin Andrey Lokhov Alexander Malyzhenkov David Mascarenas Susan Mniszewski Balu Nadiga Dan O'Malley Diane Oyen Lakshman Prasad Randy Roberts Phil Romero Nandakishore Santhi Nikolai Sinitsyn Pieter Swart Marc Vuffray Jim Wendelberger Boram Yoon Richard Zamora e Wei Zhu Patrick J. Coles, Stephan Eidenbenz. Quantum algorithm implementations for beginners, 2018. <https://arxiv.org/pdf/1804.03719.pdf>. Último acesso a 12 de Abril de 2019.
- [11] Anna Phan. Shor's algorithm for integer factorization, 2018. https://nbviewer.jupyter.org/github/Qiskit/qiskit-tutorial/blob/master/community/awards/teach_me_quantum_2018/TeachMeQ/Week_7-Quantum_Factorization/exercises/w7_02.ipynb. Último acesso a 15 de Abril de 2019.
- [12] Harsh Pokharna. For dummies -the introduction to neural networks we all need! (part 1), 2016. <https://medium.com/technologymadeeasy/for-dummies-the-introduction-to-neural-networks-we-all-need-c50f6012d5eb>. Último acesso a 29 de Maio de 2019.
- [13] IBM quantum experience. Implementation of a 2d quantum perceptron, 2018. <https://github.com/fvalle1/quantumPerceptron>. Último acesso a 5 de Maio de 2019.
- [14] IBM Research and the IBM QX team. Creating superpositions, 2017. https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=full-user-guide&page=002-The_Weird_and_Wonderful_World_of_the_Qubit~2F002-Creating_superpositions. Último acesso a 8 de Março de 2019.
- [15] IBM Research and the IBM QX team. Decoherence, 2017. https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=full-user-guide&page=002-The_Weird_and_Wonderful_World_of_the_Qubit~2F006-Decoherence. Último acesso a 22 de Março de 2019.

- [16] IBM Research and the IBM QX team. Getting started, 2017. https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=beginners-guide&page=003-Getting_Started~2F001-Getting_Started. Último acesso a 18 de Fevereiro de 2019.
- [17] IBM Research and the IBM QX team. Histogram representation (bar graph), 2017. [https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=beginners-guide&page=003-Getting_Started~2F002-Histogram_representation_\(Bar_graph\)](https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=beginners-guide&page=003-Getting_Started~2F002-Histogram_representation_(Bar_graph)). Último acesso a 21 de Fevereiro de 2019.
- [18] IBM Research and the IBM QX team. Introducing qubit phase, 2017. https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=full-user-guide&page=002-The_Weird_and_Wonderful_World_of_the_Qubit~2F003-Introducing_qubit_phase. Último acesso a 16 de Março de 2019.
- [19] IBM Research and the IBM QX team. Multi-qubit gates, 2017. https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=beginners-guide&page=006-Multi-Qubit_Gates~2F001-Multi-Qubit_Gates. Último acesso a 23 de Março de 2019.
- [20] IBM Research and the IBM QX team. Shor's algorithm, 2017. https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=full-user-guide&page=004-Quantum_Algorithms~2F110-Shor%27s_algorithm. Último acesso a 13 de Abril de 2019.
- [21] IBM Research and the IBM QX team. Single-qubit gates, 2017. https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=beginners-guide&page=005-Single-Qubit_Gates~2F001-Single-Qubit_Gates. Último acesso a 3 de Março de 2019.
- [22] IBM Research and the IBM QX team. The weird and wonderful world of the qubit, 2017. https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=beginners-guide&page=004-The_Weird_and_Wonderful_World_of_the_Qubit~2F001-The_Weird_and_Wonderful_World_of_the_Qubit. Último acesso a 1 de Março de 2019.
- [23] IBM Research and the IBM QX team. What is quantum computing?, 2017. <https://www.research.ibm.com/ibm-q/learn/what-is-quantum-computing/>. Último acesso a 15 de Fevereiro de 2019.

-
- [24] IBM Research and the IBM QX team. Qiskit, 2019. <https://qiskit.org/>. Último acesso a 27 de Fevereiro de 2019.
- [25] Nic Ezzell Joe Iosue e Arkin Tikku Ryan LaRose, Yousif Almulla. Nisqai: One-qubit quantum classifier, 2019. <https://www.nature.com/articles/s41534-019-0140-4>. Último acesso a 3 de Junho de 2019.
- [26] Krishna Kumar Sekar. Awesome quantum machine learning, 2017. <https://github.com/krishnakumarsekar/awesome-quantum-machine-learning>. Último acesso a 9 de Maio de 2019.
- [27] Barak Shoshany. Why must quantum gates be reversible?, 2014. <https://www.quora.com/Why-must-quantum-gates-be-reversible>. Último acesso a 29 de Março de 2019.
- [28] Esteban A. Martinez¹ Matthias F. Brandl¹ Philipp Schindler¹ Richard Rines² Shannon X. Wang² Isaac L. Chuang² Rainer Blatt Thomas Monz¹, Daniel Nigg¹. Realization of a scalable shor algorithm, 2016. <https://arxiv.org/pdf/1804.03719.pdf>. Último acesso a 12 de Abril de 2019.
- [29] Wikipedia. Unitarity (physics), 2019. [https://en.wikipedia.org/wiki/Unitarity_\(physics\)](https://en.wikipedia.org/wiki/Unitarity_(physics)). Último acesso a 23 de Março de 2019.
- [30] Rui Zhai Yu-Chun Wu Guang-Can Guo e Guo-Ping Guo Zhih-Ahn Jia, Biao Yi. Quantum neural network states: A brief review of methods and applications, 2019. <https://arxiv.org/pdf/1808.10601.pdf>. Último acesso a 4 de Junho de 2019.