Universidade Federal Fluminense

Instituto de Computação

Simulando Algoritmos Quânticos em um Computador Clássico

Igor Valente Blackman

NITERÓI - RJ

2017

Universidade Federal Fluminense
Instituto de Computação
Departamento de Ciência da Computação

Igor Valente Blackman

Simulando Algoritmos Quânticos em um Computador Clássico

NITERÓI - RJ
2017

IGOR VALENTE BLACKMAN

SIMULANDO ALGORITMOS QUÂNTICOS EM UM COMPUTADOR CLÁSSICO

Trabalho submetido ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Karina Mochetti de Magalhães

Niterói - RJ

2017

Dedicatória

Agradecimentos

Agradecimento 1.

Agradecimento 2.

...

Agradecimento N.

Resumo

A construção de um computador quântico terá impactos profundos em várias áreas, tais como Física, Matémática e, especialmente, a Ciência da Computação. As dificuldades práticas para se construir um computador quântico são muitas, logo não é possível prever quando ou mesmo se um computador quântico será construído. Apesar disso, a teoria na área da computação quântica vem evoluindo fortemente e para auxiliar na análise desses algoritmos pode ser essencial a criação de um simulador quântico de fácil acesso. Esse simulador utilizaria um computador clássico para realizar os trabalhos de um computador quântico, simulando o entrelaçamento e a sobreposição em bits clássicos e realizando uma estimativa de tempo gasto num computador quântico. O objetivo desse trabalho é realizar a implementação de um simulador para algoritmos quânticos.

Palavras-chave: PALAVRA1, PALAVRA2, PALAVRA3.

Abstract

The construction of a quantum computer will have deep impacts in several áreas, such as Physics, Mathematics, and especially Computer Science. There are several practical difficulties in building a quantum computer, so it is not possible to predict when or even if a quantum computer will be built. Nevertheless, the theory in the Quantum Computing field is evolving strongly and to assist in the analysis of these algorithms can be essential to creaté a quantum simulator. This simulator would be implemented in a classic computer, performing the work of a quantum computer, simulating the superposition and entanglement in clássical bits with an estimaté of efficiency. Therefore, the aim of this work is implement a simulator for quantum algorithms.

Keywords: WORD1, WORD2, WORD3.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A computação quântica aproveita a possibilidade de que as particulas subatômicas podem ter mais de um estado a qualquer momento [1]. Em um computador normal, que chamaremos de clássico, um bit é um simples pedaço de informação que possui apenas dois estados, 0 ou 1. Para representar tal informação um computador deve conter algum tipo de sistema físico com dois estados distintos para associar aos valores como, por exemplo um switch que pode estar fechado (1) ou aberto (0) [9].

Computadores quânticos utilizam bits quânticos ou "qubits", sistemas quânticos que também possuem dois estados, entretanto, diferente do bit clássico qubits podem armazenar muito mais informação já que eles podem existir em qualquer sobreposição desses valores, 0 ou 1, ou ambos ao mesmo tempo [1].

No mundo da mecânica quântica os eventos são governados por probabilidades. Um átomo radioativo, por exemplo, pode enfraquecer emitindo um eletron, ou não. É possível preparar um experimento de tal forma que exista uma chance exata de 50% de que um dos átomos de um pedaço de matérial radioativo caia em um determinado tempo [8].

Pode-se tomar como exemplo o experimento teórico do "Gato de Schrödinger". São colocados um gato, um frasco de veneno e uma fonte de radiação em uma caixa selada. Se um monitor interno (ex. Contador Geiger) detecta radiação, o frasco é quebrado soltando o veneno, matando o gato. No mundo clássico existe uma chance de 50% de o gato estar morto, e sem olhar dentro da caixa podemos dizer que o gato está vivo ou morto. Segundo a teoria, nenhuma das duas possibilidades tem qualquer realidade a menos que seja observada [8]. Isso se equivale para o qubit, ele pode ser 0 ou 1 ao mesmo tempo mas quando ele é observado ele deixa de ser um bit quântico e passa a ser um bit clássico, ou

seja, só pode ser 0 ou 1.

Outra visão quanto a um qubit pode-se pensar em uma esfera imaginaria onde um bit clássico pode estar apenas em um dos polos da esfera e o qubit, bit quântico, pode estar em qualquer ponto da esfera. Assim qubits podem armazenar uma quantidade muito maior de informação que um bit clássico.

1.1 Motivação e Objetivo

A computação quântica é uma área de pesquisa muito relevante, pois utiliza elementos de três áreas importantes: como a Física, Matémática e Engenharia. Apesar de ainda não ser possível prever quando ou mesmo se um computador quântico será construído, caso isto ocorra serão gerados grandes impactos em todas essas áreas. Em um computador quântico, os sistemas não devem ter interações físicas que não sejam sob o controle total e completo do programa. Diferente de como ocorre em um computador comum, ou os chamados computadores clássicos, interações físicas não controladas introduziriam interrupções potencialmente catastróficas na operação de um computador quântico, o que resulta numa dificuldade na construção do mesmo.

Entretanto, os benefícios que podem ser gerados com a construção de um computador quântico vão muito além do que os pesquisadores podem imaginar. Para os cientistas da computação, o mais impressionante na computação quântica e a eficiência alcançada por seus algoritmos, algo não possível na teoria clássica da complexidade computacional. O tempo de execução de algumas tarefas em um computador quântico cresce de forma muito mais lenta com o tamanho da entrada do que em um computador clássico.

Neste trabalho apresentamos a implementação de um simulador de circuitos quânticos que utiliza um computador clássico para executar os algoritmos em um computador quântico. Assim, características quânticas como o entrelaçamento e a sobreposição serão realizadas em bits clássicos o que resultará num gasto de tempo maior. Por isso, o simulador também gera uma estimativa de tempo gasto a partir das portas quânticas, emulando o que seria obtido por um computador quântico caso ele seja construído. As portas e o tempo gasto em cada uma e dado pelo usuário. O simulador é uma ferramenta online e dispõe de arraste das portas quânticas para formação dos circuitos com o intuito de facilitar a interação com o usuário.

1.2 Revisão Bibliográfica

1.2.1 Quantum - Davy Wybiral

Existe um simulador online desenvolvido por Davy Wybiral de Austin Texas, Quantum ou como está no projeto dele do Git Quantum Circuit Simulator [21]. O projeto foi disponibilizado no Git em Janeiro de 2017 [20]. Davy Wybiral utiliza o Canvas para gerar toda a parte gráfica dos circuitos e menus, e javascript para toda a parte de cálculo. O simulador não possui nenhum tipo de documentação e tem uma interface bem simples com um menu superior onde existem algumas opções e uma área com as portas quânticas que podem ser adicionadas nos circuitos e ter a quantidade dos mesmos alterada.

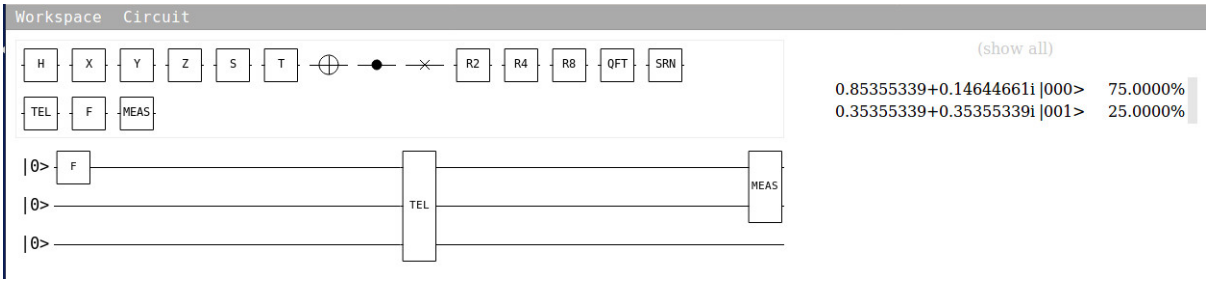


Figura 1.1: Simulador quântico de Davy Wybiral

A montagem do circuito funciona com a seleção de uma das portas e após isso basta clicar nos circuitos para adicionar a porta selecionada. Pode-se deletar uma porta clicando com o botão direito. Nos casos de portas de 2-bits é necessário que haja uma ação de clique e arraste onde o clique seleciona a posição do bit de controle e ao largar fica a porta. Existe também a possibilidade de arrastar com uma porta de 1-bit selecionada o que adiciona a mesma em diversos circuitos.

Para efetuar o cálculo do circuito basta pressionar a tecla Enter ou selecionar no menu a opção Evaluaté. O resultado é mostrado ao lado direito com todas as possibilidades possíveis e suas respectivas probabilidades.

1.2.2 Quirk - Craig Gidney

O projeto teve seu início em Março de 2014 mas não teve commits após o mes de criação. Em Novembro do mesmo ano, 2014, voltou a ter commits e entre Março de 2016 e Novembro de 2016 foi quando se teve a maior concentração de commits, 647 dos 1006

totais. O projeto ainda vem sendo atualizado sendo o commit mais recente do dia 29 de Abril de 2017 [5]. Craig Gidney, desenvolvedor do Quirk, explica que decidiu criar o simulador porque ficou interessado em computação quântica e pelo fato de ter lido ”Media for Thinking the Unthinkable”de Bret Victor onde ele menciona benefícios do feedback imediato em relação ao entendimento e produtividade. Craig diz não ter achado nenhum simulador de circuitos quânticos que passasse essa experiência de ”manipulação direta” [6]. Foi desenvolvido utilizando Webgl.

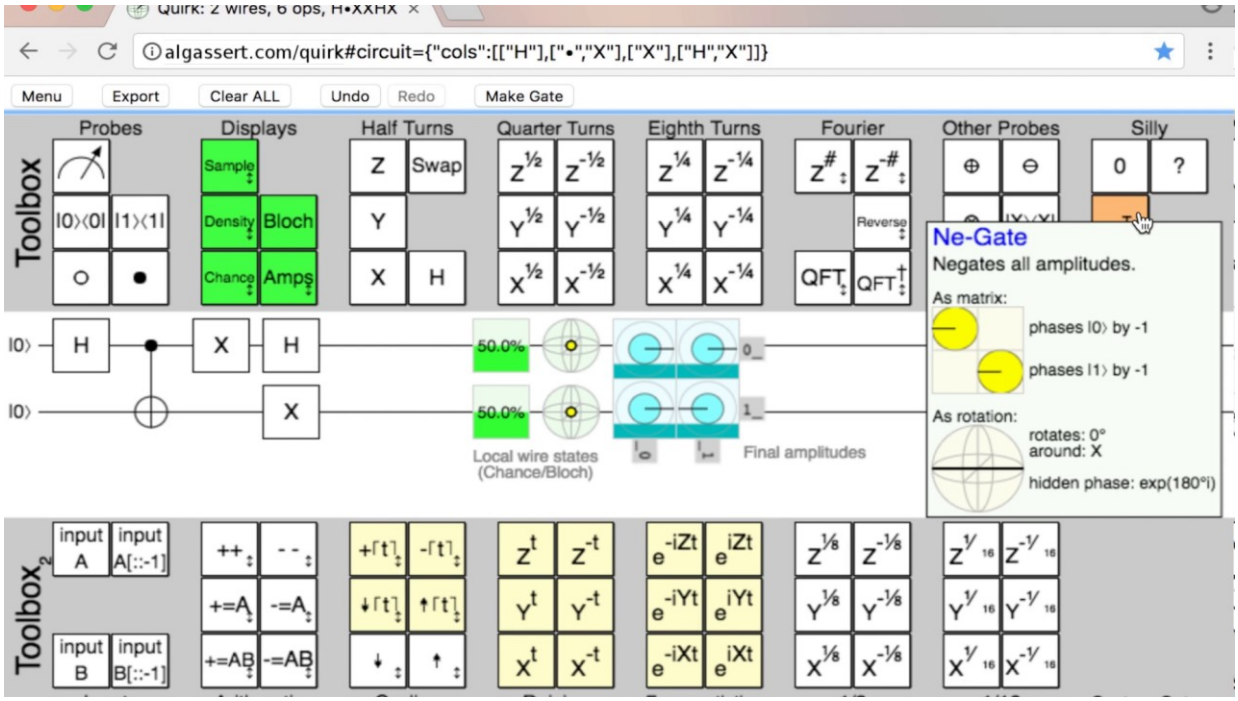


Figura 1.2: Simulador quântico Quirk de Craig Gidney

Craig faz comparações com alguns simuladores como por exemplo o já mencionado de Davy Wybiral e o IBM Quantum Experience. Ele também menciona alguns outros como Quantum Computing Playground e Microsoft's LIQ Ui|>.

O simulador Quirk é composto por um menu que possui opções de carregar alguns exemplos de circuitos já prontos, exportar o circuito montado, desfazer ou refazer alguma alteração e inclusive criar porta. Ele possui uma área com as opções de portas a serem utilizadas no circuito, formas de mostrar o resultado e mais diversas opções. Algumas vantagens desse simulador são: as portas são colocadas da forma ”arrastar e soltar”, o resultado é mostrado a todo momento não havendo a necessidade de se apertar um botão para isso, e é possível adicionar mais bits ao circuito dinâmicamente com o arrastar de uma das portas para além da linha do último bit.

1.3 Organização do Texto

Este trabalho está divido da seguinte forma: a Seção 2 é apresentada a motivação; na Seção 3 é realizada uma revisão bibliográfica, expondo a teoria e os frameworks existentes atualmente; a Seção 4 apresenta a solução implementada; a Seção 5 traz os resultados obtidos com o trabalho; e por final na Seção 6 são apresentadas as conclusões e trabalhos futuros.

Capítulo 2

Circuito Quântico

Primeiramento vamos relembrar um pouco de circuitos lógicos. Os circuitos são formados por portas como AND, OR e NOT que recebem uma sequência de valores binários 0 e 1. Ao passar pelas portas é gerada uma saída que é a entrada transformada pelo circuito.

2.1 Qubit

Já um bit quântico, ou seja um qubit, é um vetor de dois números complexos que chamaremos de a e b.

$$|\Psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix}. \tag{2.1}$$

Ao ser medido os elementos $|a|^2$ e $|b|^2$ são, respectivamente, a probabilidade de que o qubit seja $|0\rangle$ e $|1\rangle$.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.2}$$

Sendo assim os valores a e b devem seguir a regra de normalização

$$|a|^2 + |b|^2 = 1, \tag{2.3}$$

$$|\Psi\rangle = a\,|0\rangle + b\,|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}. \tag{2.4}$$

O estado $|\Psi\rangle$ é dito de estar em superposição dos estados $|0\rangle$ e $|1\rangle$ com amplitudes a e b. Se a ou b for 0 e o outro for 1, por exemplo, é o caso especial onde o estado do qubit é igual a um dos estados classicos $|0\rangle$ ou $|1\rangle$. (pagina 17)

2.2 Entrelaçamento

Assim como o estado geral de um único qubit é qualquer superposição normalizada (2.4) dos dois possíveis estados clássicos, o estado geral de $|\Psi\rangle$ nos possibilita associar com dois qubits em qualquer superposição dos quatro estados clássicos,

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.5}$$

$$|\Psi\rangle_2 = a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \tag{2.6}$$

Se tivermos dois qubits $|\Psi\rangle = a\,|0\rangle + b\,|1\rangle$ e $|\Phi\rangle = c\,|0\rangle + d\,|1\rangle$, então o estado $|\Psi\rangle_2$ do par é o produto tensor deles,

$$|\Psi\rangle_2 = |\Psi\rangle \otimes |\Phi\rangle = \begin{pmatrix} a \cdot c \\ a \cdot d \\ b \cdot c \\ b \cdot d \end{pmatrix} \tag{2.7}$$

Assim os valores das quatro amplitudes devem seguir a regra de normalização, onde

$$|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1 \tag{2.8}$$

Com (2.7) temos

$$|a \cdot c|^2 + |a \cdot d|^2 + |b \cdot c|^2 + |b \cdot d|^2 \tag{2.9}$$

$$|a|^2 \cdot |c|^2 + |a|^2 \cdot |d|^2 + |b|^2 \cdot |c|^2 + |b|^2 \cdot |d|^2 \tag{2.10}$$

$$|a|^2 \cdot (|c|^2 + d^2) + |b|^2 \cdot (|c|^2 + |d|^2) \tag{2.11}$$

$$|a|^2 \cdot 1 + |b|^2 \cdot 1 \tag{2.12}$$

$$|a|^2 + |b|^2 = 1 \tag{2.13}$$

Essa relação não se mantém, e o estado geral de 2-qubit é diferente do estado geral de 2 bits clássicos, ele não é um produto 2.7 de dois estados de 1-qubit. Esse estado de dois ou mais qubits é chamado de entrelaçamento.

2.3 Portas quânticas

Portas quânticas são operações básicas que podemos realizar em qubits. Essas portas podem ser representadas graficamente através de circuitos ou matemáticamente por matrizes.

$$|\Psi'\rangle = U\,|\Psi\rangle \qquad \text{-}\boxed{\text{H}}\text{-} \tag{2.14}$$

Uma porta quântica deve sempre ser representada por uma matriz unitária porque ao multiplicar o vetor pela matriz a norma do vetor é mantida.

2.3.1 NOT

A operação NOT quântica faz com que as amplitudes do qubits se invertam.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \text{-}\boxed{\text{X}}\text{-} \tag{2.15}$$

$$X\,|\Psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 0 \cdot a + 1 \cdot b \\ 1 \cdot a + 0 \cdot b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix} = |\Psi'\rangle \tag{2.16}$$

2.3.2 Y e Z

Análogas a operação NOT temos as operações Y e Z.

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \text{-}\boxed{\text{Y}}\text{-} \tag{2.17}$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad \text{-}\boxed{\text{Z}}\text{-} \tag{2.18}$$

2.3.3 Identidade

A operação Identidade quântica mantém os valores iniciais do qubit.

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \text{———} \tag{2.19}$$

$$I\,|\Psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1 \cdot a + 0 \cdot b \\ 0 \cdot a + 1 \cdot b \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} = |\Psi'\rangle \tag{2.20}$$

2.3.4 Hadamard

A porta Hadamard é uma operação fundamental na computação quântica e não possui uma operação clássica equivalente. Ela transforma os qubits bases $|0\rangle$ e $|1\rangle$ para um estado de sobreposição onde $|0\rangle$ e $|1\rangle$ possuem a mesma probabilidade.

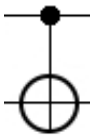
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \text{H} \tag{2.21}$$

$$H\,|\Psi\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \cdot a + 1 \cdot b \\ 1 \cdot a - 1 \cdot b \end{pmatrix} = \begin{pmatrix} \frac{a+b}{\sqrt{2}} \\ \frac{a-b}{\sqrt{2}} \end{pmatrix} = |\Psi'\rangle \tag{2.22}$$

2.3.5 CNOT

A operação CNOT possui um controle. Dados dois qubits, x e y, a matriz C mantém o valor de y caso x seja igual a 0 e caso x seja igual a 1 ela inverte o valor de y.

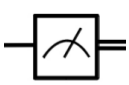
$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.23}$$



$$C\,|\Psi\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 1 \cdot a + 0 \cdot b + 0 \cdot c + 0 \cdot d \\ 0 \cdot a + 1 \cdot b + 0 \cdot c + 0 \cdot d \\ 0 \cdot a + 0 \cdot b + 0 \cdot c + 1 \cdot d \\ 0 \cdot a + 0 \cdot b + 1 \cdot c + 0 \cdot d \end{pmatrix} = \begin{pmatrix} a \\ b \\ d \\ c \end{pmatrix} = |\Psi'\rangle \tag{2.24}$$

2.3.6 Medição

Quando um qubit passa pela operação de medição ele se torna um bit clássico, 0 ou 1, e por esse motivo essa operação é muito importante na computação quântica. Para fazer essa diferenciação entre qubit e bit clássico em um circuito lógico quântico, um traço representa um qubit e um traço duplo um bit clássico.



(2.25)

Capítulo 3

Escolha das ferramentas

Neste capítulo iremos falar sobre as opções e escolhas de linguagem, biblioteca e frameworks para desenvolvimento do proejto.

3.1 Linguagem

Python foi a escolha por possuir todos os requisitos necessários para o desenvolvimento do projeto, frameworks de desenvolvimento web e bibliotecas de circuitos quânticos. Além disso, ela é vastamente utilizada em diversas aplicações e empresas como YouTube, Google, Industrial Light e Magic [4] o que adiciona mais valor ao seu aprendizdo.

3.2 Biblioteca

Tendo em vista que escolhemos python como linguagem agora é a vez de escolhermos alguma biblioteca para nos ajudar com os cálculos de cicuito quântico. A seguir falamos sobre algumas opções encontradas e qual foi a escolhida.

3.2.1 PyQu

PyQu é um módulo de extensão para Python 3 com o objetivo principal de providenciar um completo conjunto de tipos de dados e funções para simular computação quântica com uma sintaxe pura. Entretanto o último commit feito foi há 7 anos o que nos fez entender que o projeto foi descontinuado, fazendo com que discartassemos essa opção [15].

3.2.2 Qitensor

Qitensor outro módulo que essencialmente é um pacote para Numpy [12], utiliza semânticas mais úteis para mecânica quântica de dimensões finitas de muitas partículas. Infelizmente essa biblioteca também não recebia atualizações há mais de ano [17].

3.2.3 Qubiter

Qubiter tem como intuíto ajudar no designer e simulação de circuitos quânticos em computadores clássicos. É um projeto gêmeo do Quantum Fog e eles esperam que um dia o Quantum Fog chame o Qubiter para realizar algumas tarefas, como compilação e simulação quântica [18].

3.2.4 QuTiP

QuTiP teve sua primeira versão em Julho de 2011 [10]. É um software open-source para simular as dinâmicas de um sistema quântico aberto que utiliza os pacotes numéricos Numpy, Scipy e Cython. Teve sua primeira versão em Julho de 2011. Possui também um output gráfico provido pela Matplotlib. QuTiP tem como objetivo providenciar simulações numéricas eficientes e amigáveis para o usuário de um variado numero de Hamiltonianas comumente achadas em uma vasta quantidade de aplicações físicas. Está disponível sem custos para Linux, Mac OSX e Windows, sendo esta última com algumas restrições [11]. [**?**]

3.2.5 Escolhida

QuTip foi a escolhida por possuir pacotes numéricos e também output gráfico, mesmo este último não tendo sido utilizado no projeto mas com ele existe a possibilidade de utilização em trabalhos futuros. Além de possuir a melhor documentação dentre as bibliotecas pesquisadas, com exemplos e tutoriais.

No site eles pedem para citar da seguinte forma. J. R. Johansson, P. D. Nation, and F. Nori: "QuTiP: An open-source Python framework for the dynamics of open quantum systems.", Comp. Phys. Comm. 183, 1760–1772 (2012) [DOI: 10.1016/j.cpc.2012.02.021].

3.3 Framework Web

Frameworks web são projetados para dar suporte no desenvolvimento de sites dinâmicos, aplicações web e web services, auxiliando no trabalho de tarefas comuns como, controle de sessão, validação de dados, acesso ao banco de dados, templates, tratamento de requisições e respostas HTTP, etc. Dentre os frameworks web existentes para python avaliamos neste trabalho Flask, Pyramid e Django.

3.3.1 Flask

Flask é o framework mais recente encontrado, foi criado em meados de 2010. Evoluiu muito a partir da análise de framworks anteriores e tem se destacado em pequenos projetos. A sua comunidade é pequena se comparada a do Django mas é ativa em listas de email e no IRC [2].

3.3.2 Pyramid

O Pyramid foi criado a partir do projeto Pylons [14] e recebeu o nome de Pyramid somente no ano de 2010, mesmo tendo sua primeira realease em 2005. É um framework tão maduro quanto o Django e disponibiliza mais flexibilidade para desenvolvedores que possuem projetos onde o caso de uso não se encaixe muito bem nos padrões do mapeamento objeto-relacional, que nada mais é do que a representação das tabelas do banco de dados através de classes onde cada resgistro é representado como uma instancia da classe [16]. Pode-se dizer que Pyramid é o framework mais flexível dentre os estudados. Apesar disso, pareceu muito mais complexo do que noso projeto necessitava. Além disso ele possui a comunidade mais inativa dentre os frameworks pesquisados.

3.3.3 Django

Django foi lançado em 2005. É um framework com foco em grandes aplicações que inclui dezenas de extras para lidar com as tarefas comuns do desenvolvimento web, como por exemplo autenticação e administração de conteúdo, além de auxiliar também com questões de segurança como SQL injection com seus querysets, evitar ataques Cross Site Scripting e fornecer proteção a Clickjacking [3]. Sem dúvida o Django é o framework mais popular e a lista de sites que o utilizam é impressionante, tendo mais de 5000 páginas [19]. Para

sites que possuem requisitos comuns, Django segue padrões bem sensatos tornando-o uma escolha bem popular entre aplicações web de porte médio ou grande.

3.3.4 Comunidade

Django possui uma quantidade enorme de perguntas no StackOverflow se comparado com os outros dois Frameworks, Django 144.000, Flask 16.600, Pyramid 1.900 (Maio 2017) [13]. Já no Github Flask e Django têm números semelhantes e Pyramid 10 vezes menos, Flask possui um pouco mais de 27.200 estrelas, Django pouco mais de 25.900 e Pyramid 2.300 [7]. Esses números evidenciam a popularidade e o suporte oferecido por cada framework.

3.3.5 Escolhido

O Flask seria uma boa opção entretanto escolhemos o Django por:

- possuir uma grande comunidade ativa;
- diversas ferramentas já inclusas, evitando o tempo de reprogramação de funções comuns;
- ser facilmente escalável, pensando em uma futura continuação deste Proejto Final
- ser uma ferramenta muito utilizada no mercado, trazendo mais benefícios com o aprendizado.

Capítulo 4

Desenvolvimento

4.1 Drag and Drop

Como o simulador deveria ser amigável para o usuário decidimos em utilizar Drag & Drop (D & D) para a criação/edição do circuito. Resolvemos que o simulador deveria ter uma área com as portas quânticas disponíveis para montar o circuito quântico e a paritr dessa area o usuário poderia arrastar as portas para formar o circuito. Entretanto os D & D convencionais apenas possibilitavam o arrastar de um elemento para um outro container, ou seja, o elemento deixava de existir no container original e era adicionado no final. O que queriamos era que o elemento inicial continuasse existindo onde estava e que uma cópia do mesmo fosse criada e adicionada ao container alvo, ou final, por isso decidimos criar um.

Tive varios problemas com a implementação do Drag and Drop, o que achei na internet apenas movia um elemento de um lugar para o outro mas nao era bem isso que eu queria.

No lugar inicial sempre tinha que ter a imagem e eu poderia arrasta-la para um dos pontos do circuito para assim montar o mesmo.

Entao vi que, ao iniciar o drag, nao deveria mover o objeto e sim criar uma copia dele e mover a copia.

Mas ao fazer isso acabei gerando novos problemas:

- eu conseguir colocar diferentes imagens na mesma div, o que meio que gerava uma pilha, e a ideia nao era essa, em cada div do circuito deveria haver apenas 1 imagem.

Entao tive que adicionar o tratamento do Drop para que a imagem já existente fosse deletada e ai sim adicionada a que foi arrastada até la.

- As imagens se moviam de uma div do circuito para outra do circuito, o que resolvi fazendo com q a div apenas possuisse o evento de drop

Do matérial que achei na internet o JavaScript utilizado usava o ev.target o que gerava alguns problemas no tratamento do drop, pois esse target era o elemento sobre o qual o drop ocorreu, independente se o elemete possuia o evento de drop ou nao, o que me gerou confusao, mas acabei resolvendo o problema trocando target por currentTarget

4.2 Iframe ou Div

Objetivo inicial: Facilitar a visualização e utilização do simulador para montar os circuitos, com a existencia de scrolls latérais.

-Primeira solução foi englobar a div com uma div onde seriam aplicados as propriedades overflow e fixar a width e height. Mas com isso os novos circuitos nao aparecem, entao proximo passo seria modificar a altura da div quando um novo circuito e criado.

-foram criadas 2 funcoes para alterar a altura da div de acordo com a adicao ou exclusao de um circuito. Acabei utilizando a funcao outerHeight() do Jquery para pegar a altura total do circuito, quando estava usando apenas a funcao height() após acrescentar alguns circuitos o aumento nao era proprocional pois nao pegava os valores como padding, border, etc.

-foi criado tbm a funcao de ajustar width, para alterar a largura da div que contem os circuitos dependendo do numero de portas que os circuitos podem ter

4.3 Ajax

Escolhi fazer o cálculo do circuito por ajax para apenas alterar o resultado sem haver a necessidade de recarregar a pagina inteira. (1) Precisei fazer o tratamento do circuito para um objeto Json que seria passado para o ajax. (2) O sistema recebendo esse objeto utiliza a lib de python e calcula o circuito retornando o resultado. (3) Mostra o resultado na tela

(1)Tive varios problemas para entender como funcionava a parte de request do Django, entender o objeto QueryDict que e como o Django recebe o JSon enviado pelo Ajax. Tive que fazer tratamento para leitura, pois ele le de uma forma estranha.

https://docs.djangoproject.com/en/1.11/ref/request-response/

(2)Tive muita dificuldade para fazer a importação da lib Qutip no projeto, resolvendo as dependencias ta no OBS. Resolvi fazer o tratamento e criar uma funcao para realizar a transformação dependendo da porta (string) passada.

(3) O cálculo do tempo esta sendo feito pelo javascript pois já possuo todos os dados necessários na tela, nao existe a necessidade de enviar para o sistema. O sistema no passo anterior (2) processou os dados e retorna o resultado que e mostrado na tela.

OBS: Tive muito problema para arrumar as dependencias da lib Qutip http://qutip.org/docs/3

4.4 Usando QuTip

4.5 Hospedagem - Heroku

-criei um novo virtualenv para usar apenas as dependências necessárias -após o projeto estar rodando localmente gerei o arquivo requirements.txt igual ao pio freeze daquele env. -adicionei um arquivo runtime.txt para especificar a versão do python a ser utilizada, python-2.7.13 (heroku só aceita algumas versões, ele tem isso no site) -criei o Procfile com apenas "web: gunicorn quantum-circuit-simulator.wsgi –log-file -"comando a ser executado qnd a aplicação começar a rodar -logei e criei uma aplicação no heroku, "heroku creaté demo-quantum -especificquei o build pack, "heroku buildpacks:set heroku/python-addicionei e commitei as alterações feitas e realizei o deploy, "git push heroku master-Tirando os erros no processo, esses foram basicamente os passos PS: -Para instalar todas as dependências tive que fazer o deploy duas vezes alterando o requirements.txt porque não consegui achar uma forma de fazer a instalação do qutip por último e por isso acabava dando erro na instalação dele falando q faltavam dependência mas apenas prq na ordem de instalação elas não estavam sendo instaladas antes, primeiro deploy com a qutip comentado e segundo com ela, onde apenas ela foi instalada -aprendi a escrever requirements hahaha (fiquei MT tempo sem conseguir fazer o deploy prq eu havia escrito requirements errado e não consegui reparar nisso

Capítulo 5

Resultados

Texto...

Capítulo 6

Conclusão e Trabalhos Futuros

Texto...

Referências Bibliográficas

[1] Abigail Beall. Quantum computing: what is it and how does it differ to classical computing | wired uk. http://www.wired.co.uk/article/quantum-computing-explained, May 2017.

[2] Ryan Brown. Django vs flask vs pyramid: Choosing a python web framework. https://www.airpair.com/python/posts/django-flask-pyramid, May 2017.

[3] Django. The web framework for perfectionists with deadlines | django. https://www.djangoproject.com/, June 2017.

[4] Python Software Foundation. Quotes about python | python.org. https://www.python.org/about/quotes/, May 2017.

[5] Craig Gidney. Git project quirk. https://github.com/Strilanc/Quirk, April 2017.

[6] Craig Gidney. My quantum circuit simulator: Quirk. http://algassert.com/2016/05/22/quirk.html, April 2017.

[7] GitHub. Github. https://github.com/, May 2017.

[8] John Gribbin. *In Search of Schrodinger's Cat: Quantum Physics And Reality.* Random House Publishing Group, May 2011.

[9] N. David Mermin. *Quantum Computer Science, An Introduction.* Cambridge University Press, Cornell University, New York, USA, September 2007.

[10] P.D. Nation and J.R. Johansson. Qutip - change log. https://github.com/qutip/qutip-doc/blob/master/changelog.rst, May 2017.

[11] P.D. Nation and J.R. Johansson. Qutip - quantum toolbox in python. http://http://qutip.org/, April 2017.

[12] Numpy. Numpy. http://www.numpy.org/, June 2017.

[13] Stack Overflow. Tags - stack overflow. https://stackoverflow.com/tags, May 2017.

[14] Pylons Project. Pylons framework (deprecated) | pylons project. http://pylonsproject.org/about-pylons-framework.html, May 2017.

[15] Pyqu. pyqu - python project - developer fusion. www.developerfusion.com/project/42536/pyqu/, June 2017.

[16] Pyramid. Welcome to pyramid, a python web framework. https://trypyramid.com/, May 2017.

[17] Qitensor. qitensor for quantum information in python. http://www.stahlke.org/dan/qitensor, June 2017.

[18] Qubiter. Python tools for reading, writing, compiling, simulating quantum computer circuits. https://github.com/artiste-qb-net/qubiter, June 2017.

[19] Django sites. Latest additions :: Djangosites.org - powered by django. https://www.djangosites.org/, June 2017.

[20] Davy Wybiral. Git project quantum circuit simulator. https://github.com/wybiral/quantum, April 2017.

[21] Davy Wybiral. Quantum circuit simulator. http://www.davyw.com/quantum/, April 2017.