MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA

Igor Cesar Torres de Oliveira

Análise de Complexidade de Algoritmos Quânticos

Rio de Janeiro
2015

Instituto Militar de Engenharia

Igor Cesar Torres de Oliveira

Análise de Complexidade de Algoritmos Quânticos

Relatório Final do Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação do CNPq / Instituto Militar de Engenharia.
Orientador: Prof. Alex Garcia - D.C.

Rio de Janeiro
2015

Sumário

Lista de ilustrações

Lista de tabelas

Lista de códigos

Resumo

Usando a computação clássica, baseada em registradores binários e lógica determinística, uma série de problemas computacionais não possuem solução eficiente. Tais problemas são de grande interesse para a humanidade, pois estão relacionados com criptografia, inteligência artificial e problemas NP de modo geral. A computação quântica surge como uma possibilidade para a resolução de tais problemas, principalmente pelos algoritmos quânticos utilizarem processamento simultâneo ao invés de sequencial. Nesse trabalho foram estudados os primeiros algoritmos quânticos de Deutsch, Deutsch-Jozsa, Simon, Grover e Shor, bem como a fundamentação matemática para tal, e implementadas simulações dos algoritmos de Deutsch e Deutsh-Jozsa.

Palavras Chave: algoritmos, computação quântica, algoritmo quântico

1 Introdução

Na computação clássica o computador é baseado na arquitetura de Von Neumann que faz uma distinção clara entre elementos de processamento e armazenamento de dados, isto é, possui processador e memória destacados por um barramento de comunicação, sendo seu processamento sequencial.

Entretanto os computadores atuais possuem limitações, como por exemplo, na área de Criptografia e Segurança de Dados, onde não existem computadores com potência ou velocidade de processamento suficiente para suportar algoritmos de alta complexidade. Dessa forma surgiu a necessidade da criação de um computador diferente dos usuais que resolvesse problemas como a fatoração de números primos muito grandes, logaritmos discretos e simulação de problemas da Física Quântica.

Na computação quântica a unidade de informação básica é o bit quântico, também chamado de q-bit ou qubit. Um dos diferenciais da computação baseada em mecânica quântica é que ao invés de assumir valores 0 ou 1 de forma determinística como na computação clássica, o qubit pode assumir uma superposição dos estados $|0\rangle$ e $|1\rangle$, colapsando probabilísticamente numa observação.

Esta propriedade da superposição de estados que motivou os estudos em computação quântica. Se na computação clássica o processamento é sequencial, na computação quântica o processamento é simultâneo. Através de operadores chamados portas lógicas quânticas, é possivel agir sobre o sistema quântico e com uma única observação obter informações do sistema como um todo.

1.1 Justificativa

A computação quântica é um assunto que está começando a ser estudado na esfera global, embora o computador quântico seja incipiente, já existem alguns algoritmos quânticos, como por exemplo, o algoritmo de Shor para fatoração e algoritmo de Grover para acelerar o processo de procura em banco de dados.

Os algoritmos quânticos abrem portas para complexidades inferiores aos já existentes, o que futuramente terá impacto direto em criptografia, inteligência artificial e outras áreas com problemas computacionais difíceis, pois com algoritmos de processamento simultâneo será possível tratar os problemas NP-completos (para os quais não se conhece solução polinomial).

1.2 Objetivos

Este projeto tem como objetivo a análise de caráter geral de algoritmos computacionais quânticos, bem como a comparação com algoritmos da computação tradicional, para resolução de problemas da literatura convencional.

1.3 Metodologia

Foi utilizado o livro Quantum Computing for Computer Scientists para o aprendizado dos capítulos 1, 2, 3, 4, 5 e 6, os quais tratam da base matemática e os algoritmos quânticos introduzidos (capítulo 6).

Utilizando o software ipython notebook e as bibliotecas de uso livre *qutip* e *numpy*, foi implementado um simulador de sistemas quânticos, um simulador da Experiência de Fendas Múltiplas Quântica e implementações dos algoritmos de Deutsch e Deutsch-Jozsa com exemplos resolvidos.

Ao longo do ano foram feitas apresentações após o estudo de cada capítulo do livro, culminando nos simuladores 4.1.1 e 4.5.1 do capítulo 4 do livro, implementados na linguagem de programação python. Findo o estudo do capítulo 6, foi feita a implementação dos algoritmos apresentados.

1.4 Cronograma

O desenvolvimento da pesquisa tem o seguinte planejamento:

1. Pesquisa bibliográfica do princípio de funcionamento de um computador quântico.
2. Pesquisa sobre as diferenças na programação de algoritmos quânticos em relação à programação binária tradicional.
3. Análise dos algoritmos quânticos específicos.
4. Estudo comparativo dos algoritmos quânticos com os respectivos algoritmos do paradigma tradicional.

Com base nessas atividades, o seguinte cronograma é previsto para o projeto:

Atividade	1º Trimestre	2º Trimestre	3º Trimestre	4º Trimestre
1	X			
2	X	X		
3		X	X	
4			X	X

Tabela 1 – Distribuição de atividades

2 Desenvolvimento

2.1 Simulador de sistemas quânticos

Os sistemas quânticos são modelados e controlados através de operadores lineares, modelos algébricos das portas lógicas e das dinâmicas dos sistemas. O simulador a seguir implementa um sistema quântico simples, exemplificando como a álgebra matricial determina a distribuição de probabilidades de observação numa medida.

```
1  from numpy import linalg as LA
2  import numpy as NP
3
4  #Matriz de Entrada
5  M = NP.matrix([[2**(-0.5),2**(-0.5),0],
6                 [-1j*(2**(-0.5)),1j*(2**(-0.5)),0],
7                 [0 , 0 , 1j]])
8
9  #Estado inicial
10 X = NP.matrix([[3**(-0.5)],
11                [2j*(15**(-0.5))],
12                [(2./5)**(0.5)]])
13
14 #Numero de clicks
15 clicks = 1
16
17 #Estado apos n clicks
18 Y = NP.matrix(X)
19 for i in range (0,clicks):
20     Y = M*Y
21 print Y
```

Código 2.1 – Simulador quântico básico

2.2 Simulador de Fendas Múltiplas

O exemplo a seguir simula o experimento de múltiplas fendas, exemplificando o fenômeno de interferência observado em sistemas quânticos.

```
1  from numpy import linalg as LA
2  import numpy as NP
3
4  ###################ENTRADAS##################
5  #Numero de fendas
6  slits = 2
7
```

```
col = [0]

#Dimensao da matriz
n = 1 + slits + 1 + 2*slits

#Probabilidades de ir para cada slit
p = [2**(-0.5),2**(-0.5)]

############CONSTRUCAO DA MATRIZ############
#Primeira coluna
col = col + p

for k in range (slits+1,n):
    col.append(0)
col = [col]

#Proximas 'slit' colunas
for i in range (0,slits):
    cols = []
    for j in range (0, slits+1+2*i):
        cols.append(0)
    cols.append((-1+1j)/(6**(0.5)))
    cols.append((-1-1j)/(6**(0.5)))
    cols.append((1-1j)/(6**(0.5)))
    k = j+3
    for k in range (k+1,n):
        cols.append(0)
    col = col + [cols]

#Demais colunas
for i in range (slits+1,n):
    cols = []
    for j in range (0, i):
        cols.append(0)
    cols.append(1)
    for k in range (j+2, n):
        cols.append(0)
    col = col + [cols]

#Matriz B
B = NP.matrix(col)
B = B.T

#Entrada inicial X
a = [1]
for i in range (1,n):
    a.append(0)
X = NP.matrix(a)
```

```
56 X = X.T
57
58 #Resposta final
59 Y = NP.matrix((B*B)*X)
60 print Y
61
62 #Interferences
63 for i in range (slits+1,n):
64     if Y[i]==0:
65         print "There is an interference in the target " + str(i-slits)
```

Código 2.2 – Simulador do experimento de múltiplas fendas

2.3 Algoritmo Deutsch

O algoritmo de Deutsch é o primeiro exemplo de como o uso de portas quânticas pode reduzir a complexidade de execução de algoritmos. O problema resolvido pelo algoritmo é determinar se uma função $f : \{0,1\}^2 \to \{0,1\}$ é balanceada (isto é, $|\{x|f(x) = 1\}| = |\{x|f(x) = 0\}|$).

Um algoritmo tradicional precisaria fazer duas consultas para observar ambos os valores de $f(x)$, enquanto o algoritmo de Deutsch consegue determinar a resposta para o problema com uma única observação. A estratégia serviu de modelo base para os outros algoritmos quânticos, que operam uma mudança de base no sistema para tirar proveito da superposição de estados. Essa operação é definida na porta lógica H, o operador de Hadamard, definido por:

$$\begin{aligned} H(|0\rangle) &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H(|1\rangle) &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned} \tag{2.1}$$

Após a mudança de base, o sistema é sofre a ação do operador Uf. Este operador representa a função f numa porta lógica quântica e é assumido como entrada do problema. Seu comportamento é definido por:

$$Uf(|x\rangle\,|y\rangle) = |x\rangle\,|x \oplus f(y)\rangle \tag{2.2}$$

Após a ação de Uf, a mudança de base é desfeita e é feita então a observação dos estados dos qubits. Como a observação do primeiro qubit é o suficiente para a solução do problema, o segundo qubit pode ser ignorado após a ação de Uf.

Iniciando o sistema no estado $|01\rangle$, O resultado final do sistema será da forma

$$\pm\frac{1}{\sqrt{2}}\,|f(0) \oplus f(1)\rangle\,(|0\rangle - |1\rangle)$$

portanto se a função f for balanceada, observaremos com probabilidade 1 o primeiro qubit no estado $|1\rangle$, caso contrário observaremos $|0\rangle$.

Na implementação abaixo, instanciamos um circuito lógico (variável *composition*) com o operador identidade e os registradores de qubit no estado inicial $|01\rangle$ (variável *qreg*). Adicionamos então operadores de Hadamard em ambos os qubits através do produto tensorial, criamos operador Uf para uma função f exemplo e adicionamos ao circuito e finalmente adicionamos outro operador Hadamard antes da saída.

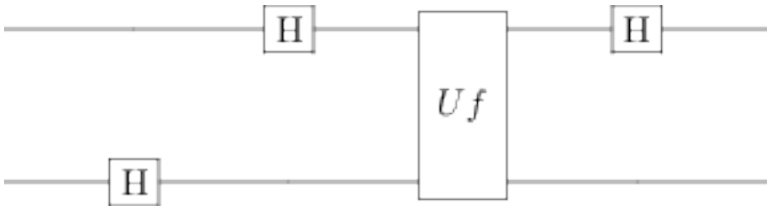

Figura 1 – Diagrama de portas lógicas do algoritmo de Deutsch

```
N = 2 # Numero de fios de entrada
qreg = basis(4,1) # vetor que representa o estado quantico de entrada
circuit = QubitCircuit(N) # instancia um circuito de N fios paralelos
composition = cphase(0) # instancia o circuito com a matriz identidade

H = snot() # instancia uma porta de Hadamard
composition = composition * tensor(H,H) # coloca uma porta de hadamard em cada fio

def U(f): # cria a porta quantica caixa-preta que representa a funcao f
    M = [[(f(0)==0),(f(0)==1),0,0],
         [(f(0)==1),(f(0)==0),0,0],
         [0,0,(f(1)==0),(f(1)==1)],
         [0,0,(f(1)==1),(f(1)==0)]]
    return Qobj(M, dims=[[2,2],[2,2]])

def f(x): # uma funcao exemplo a determinar se e balanceada
    return 1-x

Uf = U(f) # cria a porta quantica da funcao exemplo

composition = Uf * composition # adiciona a porta ao circuito

# adiciona outra porta de hadamard ao circuito para ser lida
composition = tensor(H, rx(0)) * composition

# saida do algoritmo
print qreg.transform(composition)
```

Código 2.3 – Implementação do algoritmo de Deutsch

No exemplo acima a função $f(x) = 1 - x$ leva o estado final a

```
Quantum object: dims = [[4], [1]], shape = [4, 1], type = ket
Qobj data =
[[ 0.        ]
 [ 0.        ]
 [-0.70710678]
 [ 0.70710678]]
```

que corresponde ao estado $\frac{1}{\sqrt{2}}|0\rangle(|0\rangle - |1\rangle)$. Portanto a observação do primeiro qubit indica função balanceada.

Ao trocarmos a função exemplo para $f(x) = 1$, o estado final será

```
Quantum object: dims = [[4], [1]], shape = [4, 1], type = ket
Qobj data =
[[-0.70710678]
 [ 0.70710678]
 [ 0.        ]
 [ 0.        ]]
```

que corresponde ao estado $\frac{1}{\sqrt{2}}|1\rangle(|0\rangle - |1\rangle)$. Portanto a observação do primeiro qubit indica função balanceada.

2.4 Algoritmo Deutsch-Jozsa

O algoritmo de Deutsch-Jozsa é uma extensão do algoritmo de Deutsch para funções $f : \{0,1\}^n \to \{0,1\}$, determinando se uma função f (sabido que ela é balanceada ou constante) é balanceada.

O algoritmo demonstra que a diferença no custo de observação do caso base descoberto por Deutsch anteiormente leva a um crescimento exponencial no caso de algoritmos clássicos ao contrário de complexidade constante no caso quântico. Ele serviu de base para os algoritmos de Shor e Grover, que se utilizam de esquemas semelhantes respectivamente para fatorar números e inverter funções.

Enquanto algoritmos clássicos precisam aumentar sua complexidade computacional, algoritmos quânticos aumentam sua complexidade de hardware por se tratar de computação simultânea. O algoritmo de Deutsch-Jozsa aumenta a complexidade de hardware pois precisa de um novo registrador para cada dimensão no espaço de estados quânticos.

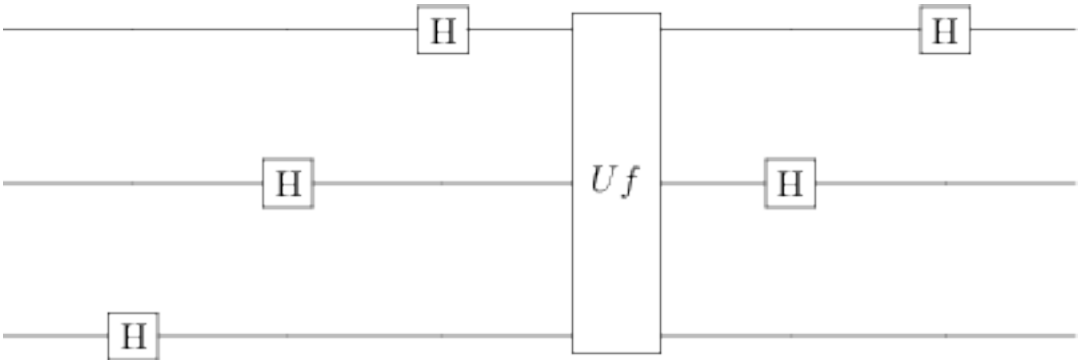

Figura 2 – Diagrama de portas lógicas do algoritmo de Deutsch-Jozsa

De forma semelhante ao algoritmo de Deutsch, no sistema final os n primeiros qubits de saída tem uma probabilidade de serem observados no estado $|0\rangle^{\otimes n}$ dada por:

$$\left| \frac{1}{2^n} \sum_{k=0}^{2^n-1} (-1)^{f(k)} \right|^2$$

que resulta em 0 se e somente se f é balanceada e em 1 se e somente se f é constante.

A seguir apresentamos uma implementação para o algoritmo com $n = 2$, muito semelhante ao algoritmo de Deutsche (caso $n = 1$).

```
N = 3
qreg = basis(2**(N),1)
circuit = QubitCircuit(N)
composition = cphase(0,3)

H = snot()
composition = composition * tensor(H,H,H)

def U(f):
    M = [[f(0,0)==0,f(0,0)==1,0,0,0,0,0,0],
         [f(0,0)==1,f(0,0)==0,0,0,0,0,0,0],
         [0,0,f(0,1)==0,f(0,1)==1,0,0,0,0],
         [0,0,f(0,1)==1,f(0,1)==0,0,0,0,0],
         [0,0,0,0,f(1,0)==0,f(1,0)==1,0,0],
         [0,0,0,0,f(1,0)==1,f(1,0)==0,0,0],
         [0,0,0,0,0,0,f(1,1)==0,f(1,1)==1],
         [0,0,0,0,0,0,f(1,1)==1,f(1,1)==0]]
    return Qobj(M, dims=[[2,2,2],[2,2,2]])

def f(x,y):
    return x+y%2

Uf = U(f)

composition = Uf * composition

composition = tensor(H, H, rx(0)) * composition

print qreg.transform(composition)
```

Código 2.4 – Implementação do algoritmo de Deutsch-Jozsa

O estado final com $f(x, y) = x + y \mod 2$, por exemplo, é

```
Quantum object: dims = [[8], [1]], shape = [8, 1], type = ket
Qobj data =
[[ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.70710678]
 [-0.70710678]]
```

que representa superposição de estados $|110\rangle$ e $|111\rangle$, logo probabilidade 0 de observar $|00\rangle$ nos dois primeiros qubits.

Alternativamente para $f(x, y) = 1$, temos:

```
1
2 Quantum object: dims = [[8], [1]], shape = [8, 1], type = ket
3 Qobj data =
4 [[-0.70710678]
5  [ 0.70710678]
6  [ 0.        ]
7  [ 0.        ]
8  [ 0.        ]
9  [ 0.        ]
10 [ 0.        ]
11 [ 0.        ]]
```

que representa superposição de estados $|000\rangle$ e $|001\rangle$, logo probabilidade 1 de observar $|00\rangle$ nos dois primeiros qubits.

Podemos ver que a implementação de tais algoritmos usando a computação clássica torna-se muito custosa para casos grandes, pois exige processamento de muitos casos sequencialmente. Porém, se o processamento fosse simultâneo, como aconteceria com um hardware quântico, a complexidade seria na ordem de 1.

3 Conclusão

Com a implementação dos algoritmos de Deutsch e Deutsch-Jozsa pode-se perceber que tais algoritmos quânticos podem resolver problemas com complexidade muito inferior aos clássicos.

Entretanto todos os casos vistos, a simulação de um computador quântico é mais complexa que o problema original resolvido pelos algoritmos quânticos.

Portanto as simulações são úteis em caráter de prova de conceito e demonstração, porém é necessário a implementação em hardware específico para um real ganho computacional.

Referências

YANOFSKY, N. S.; MANNUCCI, M. A. *Quantum Computing for Computer Scientists*. 1. ed. New York: Cambridge University Press, 2008.