



Minicurso- Introdução à Computação Quântica

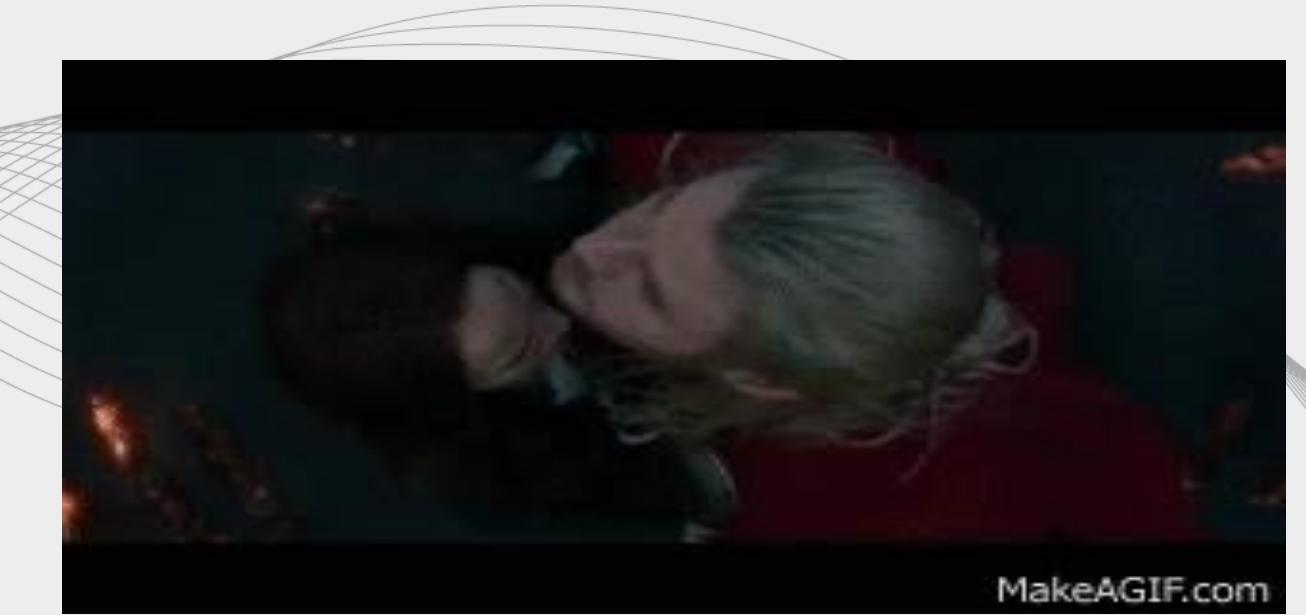
Anderson Buarque, PhD –
anderson.correia@fieb.org.br

Henrique Ghizoni, PhD –
henrique.ghizoni@fbter.org.br

Yan Chagas – yan.chagas@fieb.org.br



Teleporte Quântico



Teleporte Quântico

- Técnica para mover estados quânticos, mesmo na ausência de um canal de comunicação quântica ligando o remetente do estado quântico ao destinatário.
- A informação que é transferida, as partículas continuam “sem se mover”.
- É possível graças ao emaranhamento

Teleporte Quântico

- O termo teletransporte quântico foi dado pelo físico Charles Bennett que, com sua equipe em 1993, mostrou ser possível realizar a implementação do teletransporte em um canal clássico utilizando pares de partículas de estados emaranhados.



Teleporte Quântico



Localidade e não-localidade

Paradoxo EPR

Desigualdades de Bell

Localidade e Não-Localidade

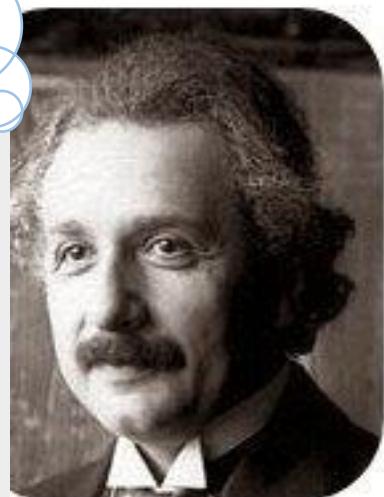
Localidade	Não-Localidade
Uma ação só afeta o seu entorno imediato	Ações à distância afetam instantaneamente
Nenhuma informação pode viajar mais rápido do que a luz	Não há limite de distância
Sistemas precisam estar próximos para interagirem	Não há analogias clássicas para o emaranhamento
Base da Física Clássica	Base da Mecânica Quântica

Sistemas correlacionados (sistemas compostos) são representados por estados não separáveis e, consequentemente, a observação de um dos sistemas garante informações sobre o outro.

Exercício FPR



mental que questiona a completude da mecânica quântica. Ele afirma que se duas partículas estiverem emaranhadas, o estado da outra parte se altera instantaneamente quando uma delas é medida, mesmo que esteja separada por grandes distâncias. Se essa correlação fosse usada para enviar mensagens, seria possível transmitir informação, pareceria violar a relatividade, que estabelece que a velocidade da luz é constante e nenhuma informação pode viajar mais rápido que a luz.



A. Einstein



B. Podolsky



N. Rosen

Paradoxo EPR

Para o trio de cientistas, uma teoria completa deve descrever todos os elementos da realidade física.

A mecânica quântica estaria incompleta, ou toda a física clássica estaria errada.

As tentativas de explicar o paradoxo EPR, mesmo buscando refutar a mecânica quântica, impulsionaram a criação de um novo campo da física: a informação quântica.

John Bell propôs um teste para a hipótese de Einstein

Desigualdades de Bell

- Proposto por John Stewart Bell em 1964, estabelece distinção entre mecânica quântica e mecânica clássica.
- Não existe um regime de variáveis ocultas locais que possam reproduzir todos os resultados da MQ.
- Bell mostrou que a hipótese do realismo local, ou seja...
 - **que uma partícula possui valores definitivos que não dependem do processo de observação e**
 - **que a velocidade de propagação dos efeitos físicos é finita**

não é compatível com a mecânica quântica.

Desigualdades de Bell

- provou que a existência de variáveis ocultas locais,
- As desigualdades de Bell descrevem a fronteira do conjunto de correlações clássicas locais,
- Foram testadas em laboratório
- Como resultado de violação dessas desigualdades, há um conjunto de estados quânticos maximamente emaranhados, envolvendo dois qubits que exibem a correlação máxima permitida pela mecânica quântica.

Estados de Bell

Estados de Bell

- Fundamentais para a Informação Quântica.
- quando uma partícula em um estado de Bell é medida, o resultado apresenta imediatamente informações sobre a outra, independentemente da distância entre elas.
- permanecem emaranhados em operações que afetam apenas um qubit, exceto medições. (**Invariância sob medições locais**)
- Aplicações em protocolos de criptografia baseado em emaranhamento (ex: E91)

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle)$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}} (|00\rangle - |11\rangle)$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}} (|01\rangle + |10\rangle)$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}} (|01\rangle - |10\rangle)$$

Teleporte Quântico

Alice e Bob estão separados a uma distância física.

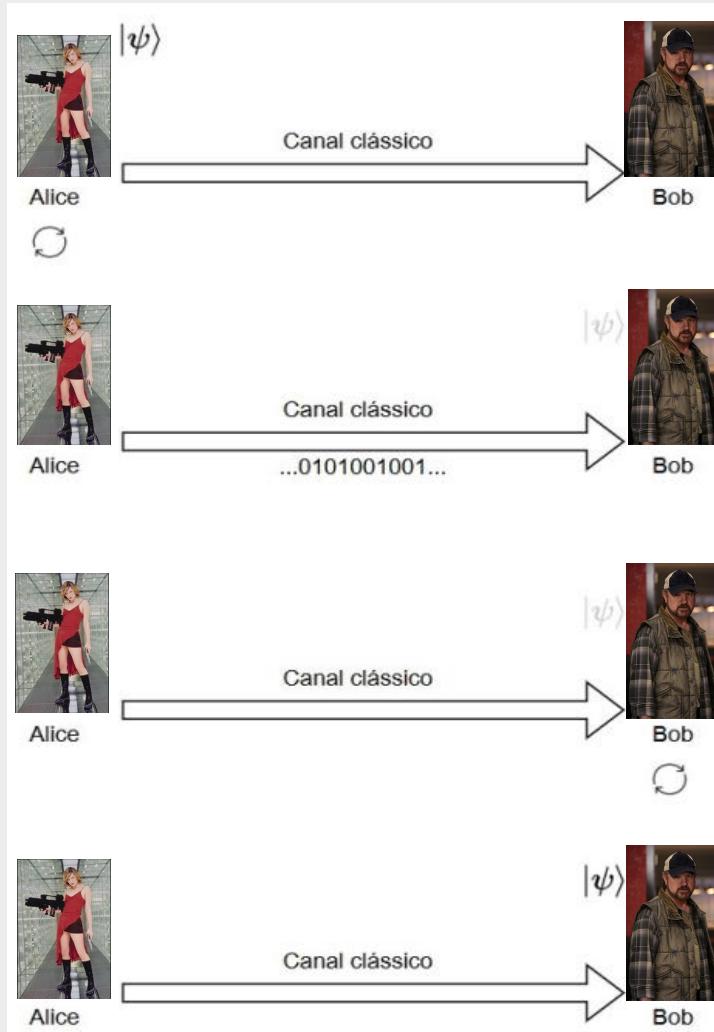
Alice tem um par de qubits emaranhandos, porém separados fisicamente.

Alice possui um estado quântico ψ , e pode se comunicar com Bob através de um canal clássico

Eles não conseguem enviar o estado quântico por canais clássicos

Como fazer para mandar esse estado para Bob ?

Teleporte Quântico



Objetivo: Transferir $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

Alice e Bob devem dividir um par de qubits emaranhados.

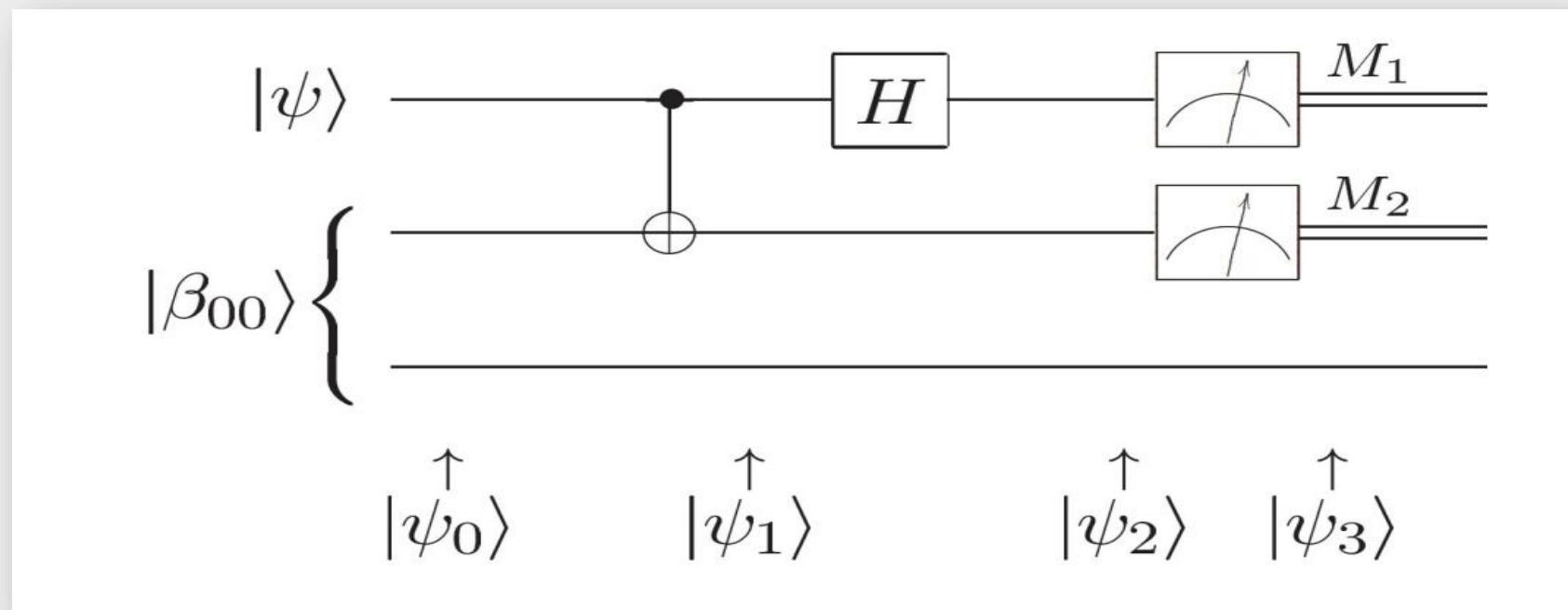
$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

Teleporte Quântico

Considere o estado inicial

$$|\psi_0\rangle = |\psi\rangle|\beta_{00}\rangle, \text{ onde } |\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Representando em forma de circuito, temos:



Teleporte Quântico

Estado inicial

$$|\psi_0\rangle = (\alpha|0\rangle + \beta|1\rangle)\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right) = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle)$$

Alice aplica um CNOT nos seus qubits

$$|\psi\rangle = \frac{1}{\sqrt{2}}[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|10\rangle + |01\rangle)]$$

Teleporte Quântico

Alice aplica a porta Hadamard no primeiro qubit

$$|\psi_2\rangle = \frac{1}{2} [\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle)(|10\rangle + |01\rangle)]$$

Reescrevendo $|\psi_2\rangle$, temos

$$\frac{1}{2} [|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)]$$

Teleporte Quântico

Assim $|\psi_3\rangle$ assume 4 estados possíveis dependendo do valor medido por Alice.

M1	M2	$ \psi_3\rangle$
0	0	$\alpha 0\rangle + \beta 1\rangle$
0	1	$\alpha 1\rangle + \beta 0\rangle$
1	0	$\alpha 0\rangle - \beta 1\rangle$
1	1	$\alpha 1\rangle - \beta 0\rangle$

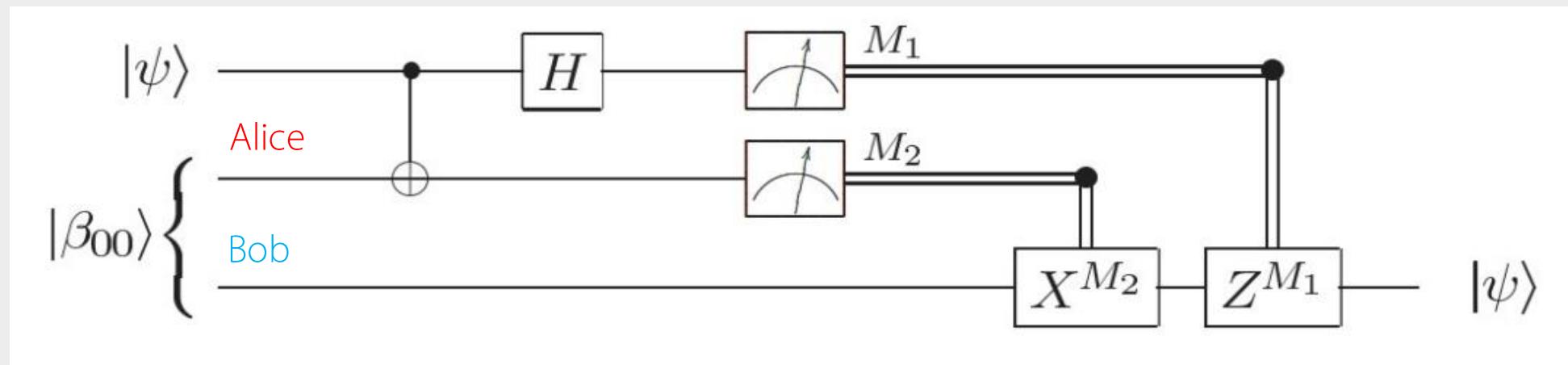
Se $M2 = 1$, precisamos inverter os estados $|0\rangle$ e $|1\rangle$

Se $M1 = 0$, é necessário corrigir os valores de β

Teleporte Quântico

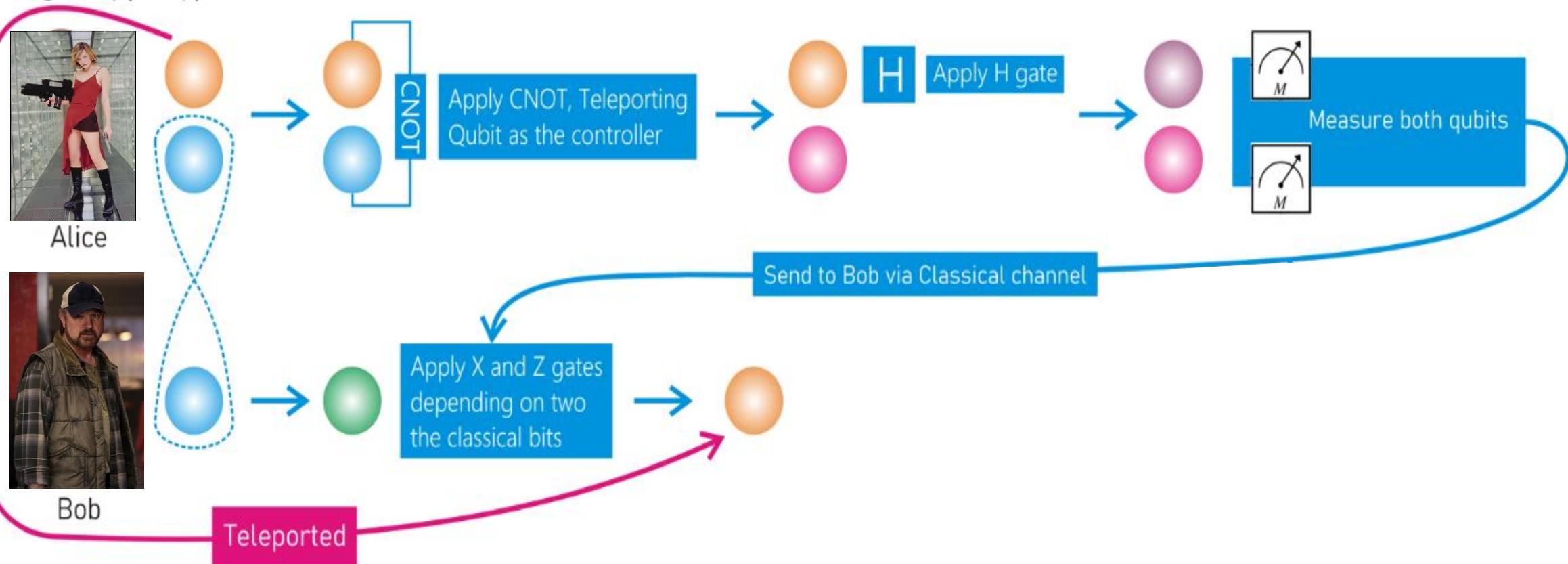
Para garantir que o que chegou em Bob é realmente o estado $|\psi\rangle$, ele precisa complementar o circuito com portas condicionadas aos valores clássicos que foram medidos.

Alice avisa a Bob o que foi medido e ele adiciona as portas no circuito, ficando da seguinte forma.



Teleporte Quântico

Message = $a|0\rangle + b|1\rangle$



O qubit foi clonado?



Antes de Medir

Bob tem uma superposição de informações

Depois de Medir

O qubit de Alice é colapsado, e a informação está com Bob.



O qubit **não** foi clonado

Teorema da não clonagem

Não é possível clonar um estado quântico, apenas transportar a informação dos qubits.

Todas as amplitudes do estado são preservadas.

Sempre será possível recuperar o estado ψ

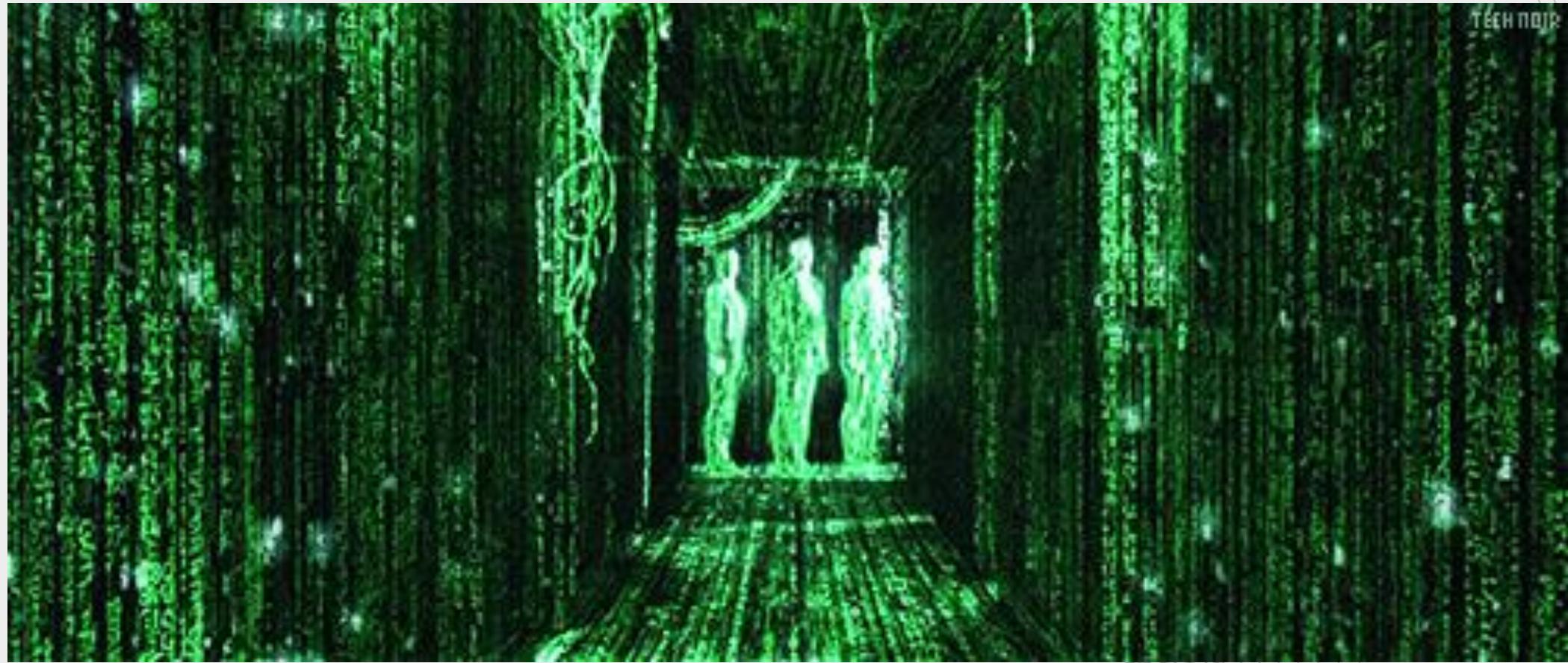
Teorema da não clonagem

Bob tendo a tabela abaixo em mãos, consegue recuperar o estado por qualquer medição de Alice.

$$\begin{aligned} 00 &\rightarrow |\psi_3(00)\rangle \equiv [\alpha|0\rangle + \beta|1\rangle] \\ 01 &\rightarrow |\psi_3(01)\rangle \equiv [\alpha|1\rangle + \beta|0\rangle] \\ 10 &\rightarrow |\psi_3(10)\rangle \equiv [\alpha|0\rangle - \beta|1\rangle] \\ 11 &\rightarrow |\psi_3(11)\rangle \equiv [\alpha|1\rangle - \beta|0\rangle] \end{aligned}$$

Em um circuito quântico, basta aplicar algumas portas para recuperar o estado.

Implementando Estados de Bell



Os estados a seguir representam os pares de Bell maximamente emaranhados necessários para que Alice e Bob possam se comunicar, vamos implementar cada um deles:

$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$$|\beta_{01}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$

$$|\beta_{10}\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

$$|\beta_{11}\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

Instalando as bibliotecas necessárias

```
[ ] !pip install qiskit
→ Mostrar saída oculta

[ ] pip install qiskit-aer
→ Mostrar saída oculta

[ ] pip install matplotlib
→ Mostrar saída oculta

[ ] pip install pylatexenc
→ Mostrar saída oculta
```

$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) :$$

```
[ ] from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit_aer import AerSimulator

# Criando um registrador quântico com 2 qubits
q = QuantumRegister(2, 'q')
# Criando registradores clássicos para medições
c = ClassicalRegister(2, 'c')

# Criando o circuito quântico
bell_circuit = QuantumCircuit(q, c)

# Aplicando Hadamard no primeiro qubit
bell_circuit.h(q[0])
# Aplicando CNOT com o primeiro qubit como controle e o segundo como alvo
bell_circuit.cx(q[0], q[1])

# Medindo os qubits
bell_circuit.measure(q, c)

# Configurando o simulador
simulator = AerSimulator()

# Exibir o circuito
bell_circuit.draw(output='mpl', scale=0.5)
```

$$|\beta_{01}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) :$$

```
# Criando o circuito quântico
bell_circuit = QuantumCircuit(q, c)

# Inicializando o segundo qubit no estado |1>
bell_circuit.x(q[1])

# Aplicando Hadamard no primeiro qubit
bell_circuit.h(q[0])
# Aplicando CNOT com o primeiro qubit como controle e o segundo como alvo
bell_circuit.cx(q[0], q[1])

# Medindo os qubits
bell_circuit.measure(q, c)

# Configurando o simulador
simulator = AerSimulator()

# Exibir o circuito
bell_circuit.draw(output='mpl', scale=0.5)
```

$$|\beta_{10}\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) :$$

```
# Criando o circuito quântico
bell_circuit = QuantumCircuit(q, c)

# Inicializando o primeiro qubit no estado |1>
bell_circuit.x(q[0])

# Aplicando Hadamard no primeiro qubit
bell_circuit.h(q[0])
# Aplicando CNOT com o primeiro qubit como controle e o segundo como alvo
bell_circuit.cx(q[0], q[1])

# Medindo os qubits
bell_circuit.measure(q, c)

# Configurando o simulador
simulator = AerSimulator()

# Exibir o circuito
bell_circuit.draw(output='mpl', scale=0.5)
```

$$|\beta_{11}\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) :$$

```
# Criando o circuito quântico
bell_circuit = QuantumCircuit(q, c)

# Inicializando ambos os qubits no estado |1>
bell_circuit.x(q[0])
bell_circuit.x(q[1])

# Aplicando Hadamard no primeiro qubit
bell_circuit.h(q[0])
# Aplicando CNOT com o primeiro qubit como controle e o segundo como alvo
bell_circuit.cx(q[0], q[1])

# Medindo os qubits
bell_circuit.measure(q, c)

# Configurando o simulador
simulator = AerSimulator()

# Exibir o circuito
bell_circuit.draw(output='mpl', scale=0.5)
```

Implementando Circuito do Teleporte



```
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt

# Mapeamento de correções baseado no resultado da medição de Alice
corrections = {
    '00': [],
    '01': ['X'],
    '10': ['Z'],
    '11': ['Z', 'X']
}

# Simulador
simulator = AerSimulator()
```

```
# Loop sobre os 4 possíveis resultados da medição de Alice
for meas_result in ['00', '01', '10', '11']:
    # Criar circuito
    qc = QuantumCircuit(3, 3)

    # Criar par de Bell entre q1 e q2
    qc.h(1)
    qc.cx(1, 2)

    # Preparar o estado de entrada: |+> = H|0>
    qc.h(0)

    # Teletransporte: Alice
    qc.cx(0, 1)
    qc.h(0)
    qc.barrier()

    # Medições de Alice
    qc.measure(0, 0)
    qc.measure(1, 1)
    qc.barrier()
```

```
# Correções de Bob baseadas em meas_result
if meas_result == '01':
    qc.x(2)
elif meas_result == '10':
    qc.z(2)
elif meas_result == '11':
    qc.z(2)
    qc.x(2)

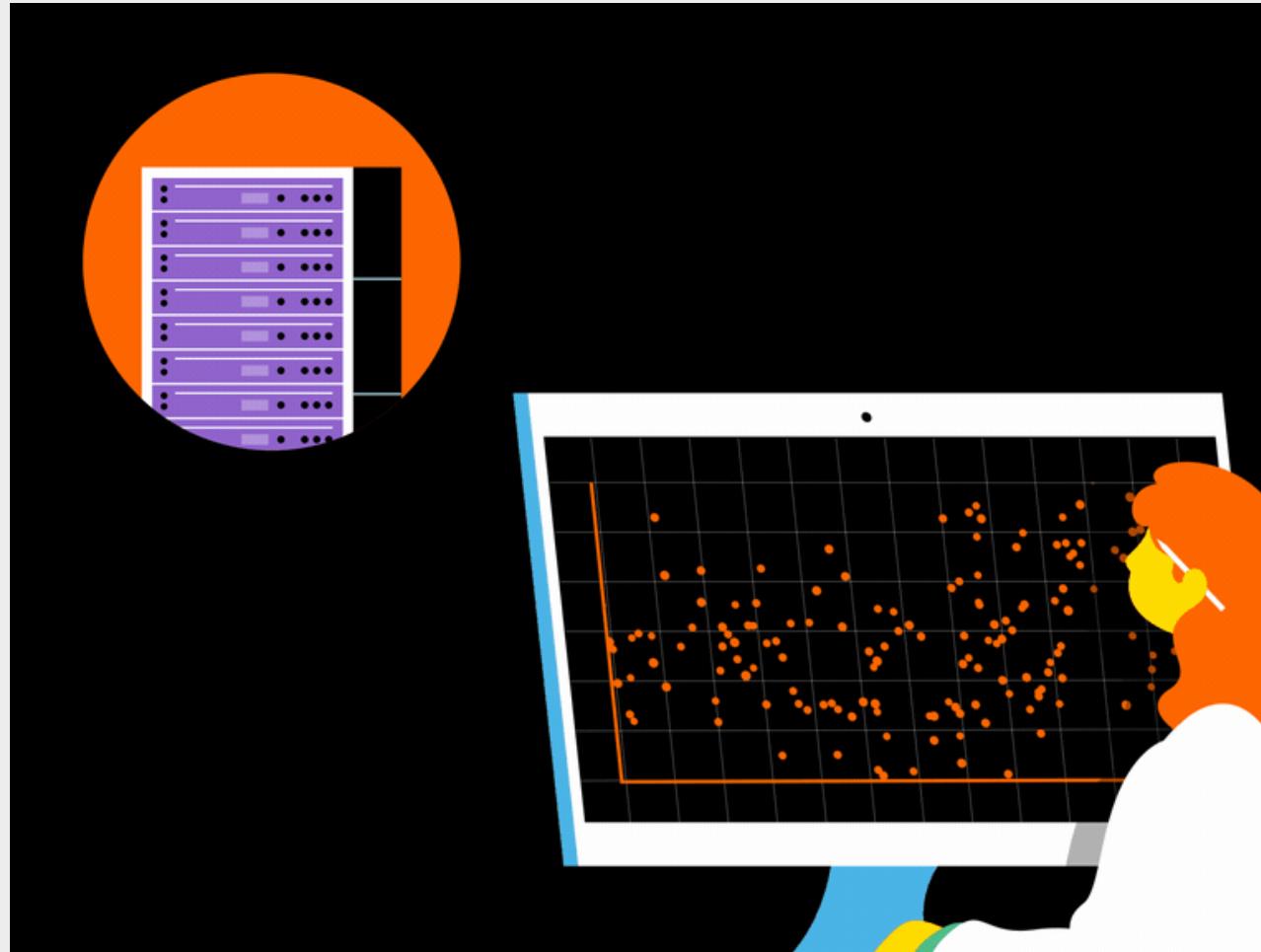
# Medição final de Bob
qc.measure(2, 2)
```

```
# Transpilar e simular
compiled = transpile(qc, simulator)
job = simulator.run(compiled, shots=1, memory=True) # apenas 1 shot para simplificar
result = job.result()
memory = result.get_memory()[0] # resultado no formato 'c2c1c0'

# Quebrar resultado em bits
bob_meas = memory[0] # c2
alice_meas = memory[2] + memory[1] # c0c1 (como string, mesma ordem usada em nosso loop)

# Exibir informações
print("=*40)
print(f"Resultado simulado da medição de Alice: {meas_result}")
print("Correções aplicadas por Bob:", 'Nenhuma' if not corrections[meas_result] else '.join(corrections[meas_result]))')
print(f"Bob mediu o qubit final como: |{bob_meas}>")
print("\nCircuito:")
display(qc.draw('mpl', scale=0.8))
```

Algoritmos Quânticos

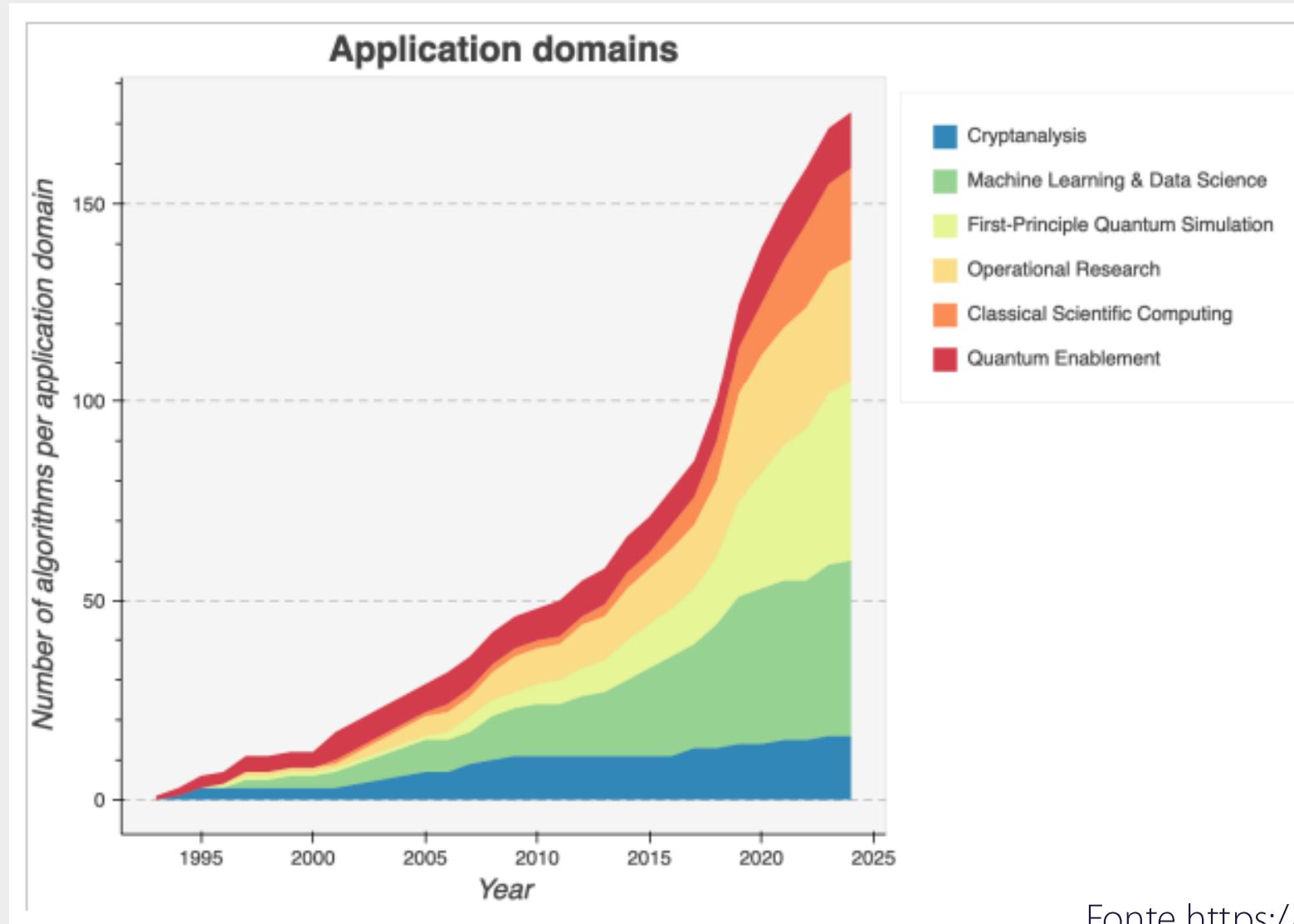


Algoritmos quânticos

Utiliza-se de circuitos quânticos para manipular qubits de forma que obtenha-se vantagem dos fenômenos quânticos para realizar cálculos de forma mais eficiente do que computadores clássicos.

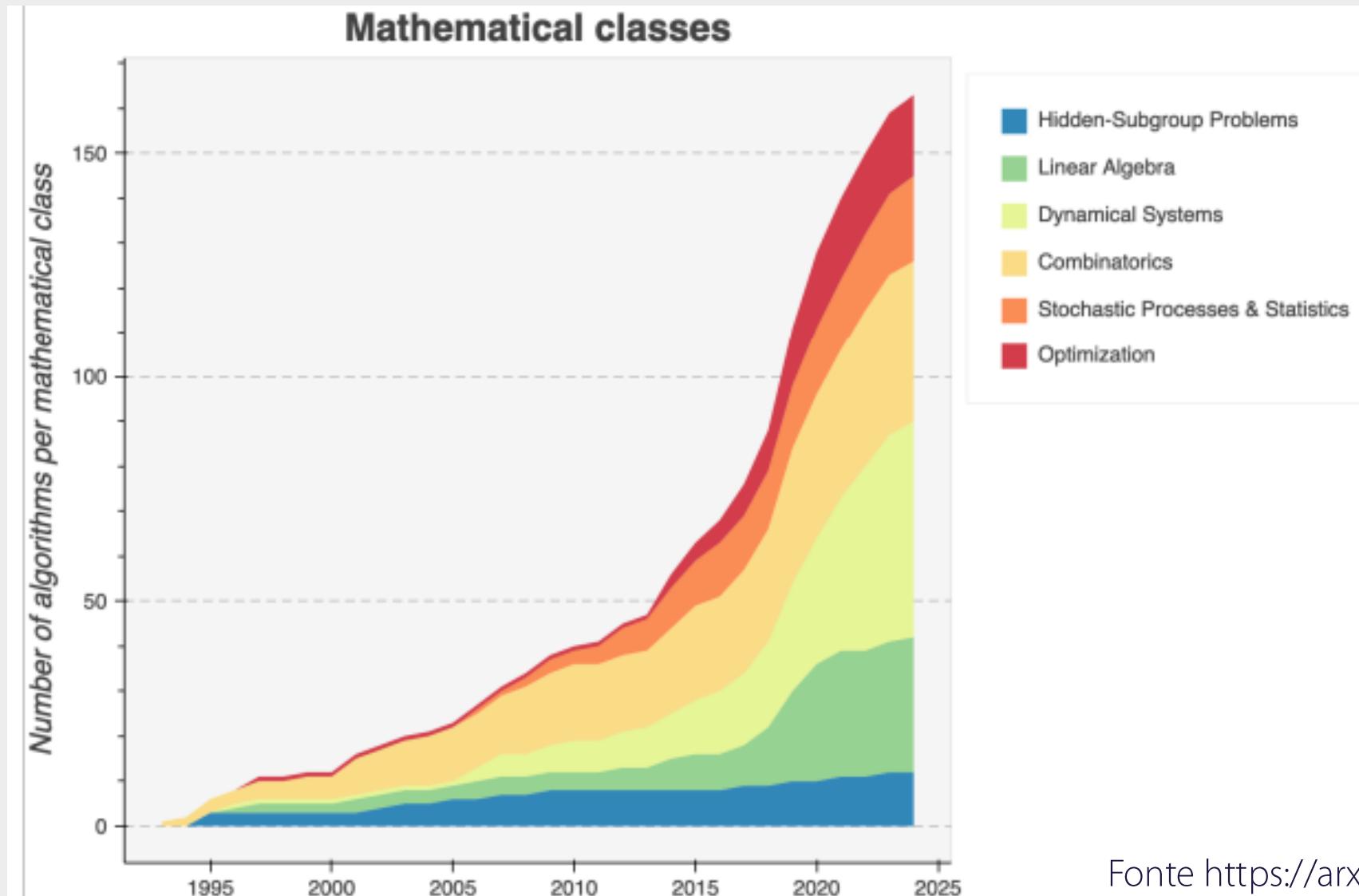
Geralmente são desenvolvidos de maneira abstrata, focando na matemática, o que independe do tipo de arquitetura que será implementada.

Algoritmos Quânticos



Fonte <https://arxiv.org/pdf/2407.05178v1>

Algoritmos Quânticos



Fonte <https://arxiv.org/pdf/2407.05178v1>

Algoritmo de Deutsch

- Primeiro algoritmo quântico a ser desenvolvido
- Proposto por David Deutsch em 1985
- Objetivo de definir se uma função é constante ou balanceada de forma mais eficiente do que sua contrapartida clássica.
- Corroborou a ideia de Feynman do computador quântico.

Algoritmo de Deutsch

Funções Constantes:

Todos os valores da imagem são iguais.

Funções Balanceadas:

Metade dos valores será 0 e a outra metade será 1 (para o caso binário)

OBS: Há classes de funções que não se encaixam nem como constantes, nem como balanceadas.

Solução Clássica

- Realizar metade das medições mais uma, ou seja
- Medir $2^{n-1} + 1$
- Quanto maior o problema



...i se tornando o

Algoritmo de Deutsch

Trabalharemos com dois qubits

Nosso espaço de função é dado por: $f: \{0,1\}^2 \rightarrow \{0,1\}$

	f_1	f_2	f_3	f_4
0	0	0	1	1
1	0	1	0	1

Se $f = 0 \rightarrow$ Constante ; Se $f = 1 \rightarrow$ Balanceada

Algoritmo de Deutsch

Há uma U_f que é aplicada no algoritmo, que atua da seguinte forma:

$$U_f |x\rangle |j\rangle = |x\rangle |j \oplus f(x)\rangle$$

Onde \oplus é a soma direta de módulo 2, sendo assim

$$0 \oplus 0 = 0$$

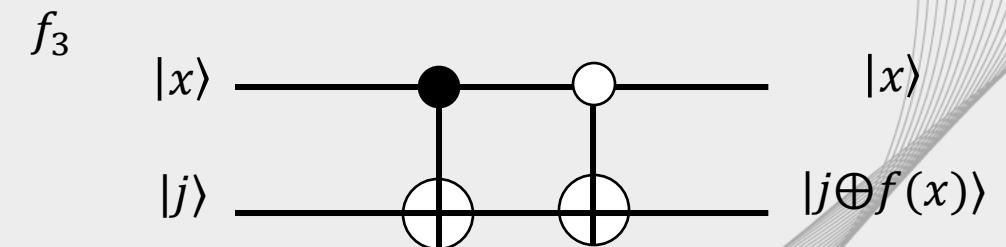
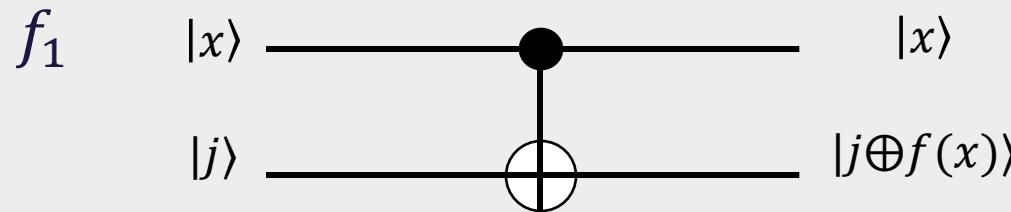
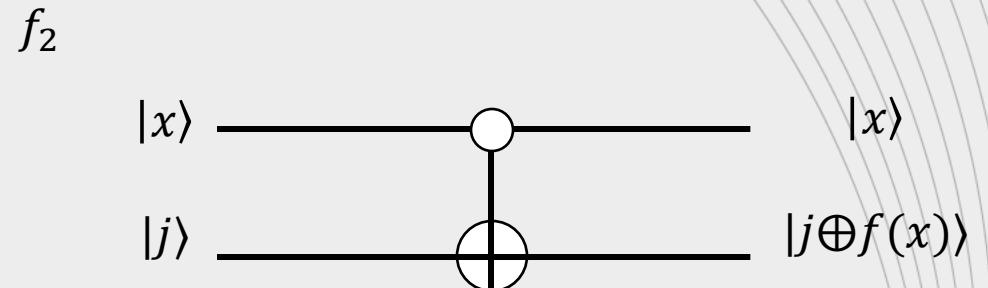
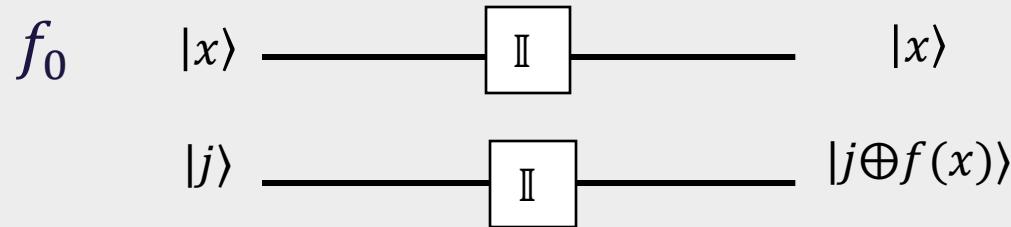
$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 1$$

$$1 \oplus 1 = 0$$

Caso quiséssemos implementar essa U_f , como fazer?

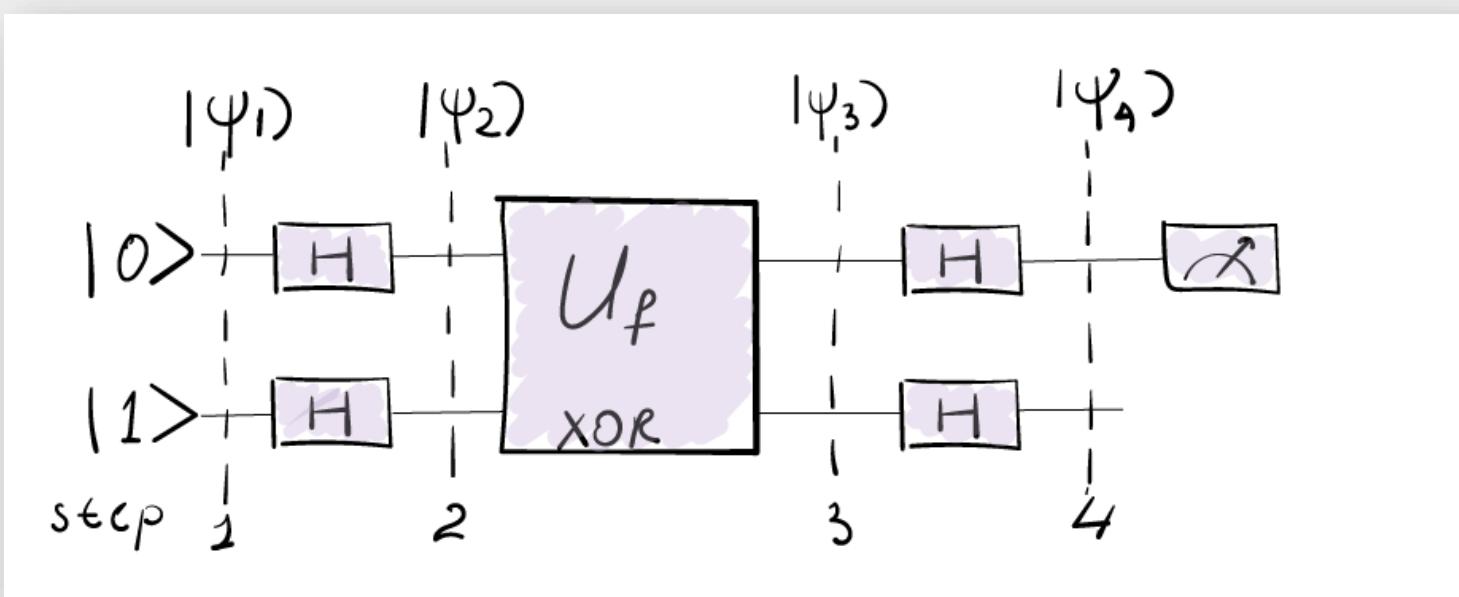
Algoritmo de Deutsch



O objetivo do algoritmo de Deutsch é definir se uma função é constante ou balanceada

Pseudo Código do Algoritmo

1. Preparar o estado $|0\rangle|1\rangle$
2. Aplicar $H \otimes H$
3. Aplicar U_f (oráculo)
4. Aplicar $H \otimes H$ novamente
5. Medir o primeiro qubit na base computacional



Analizando os Estados

$$|\psi_0\rangle = |0\rangle|1\rangle - \text{É entrada}$$

$$\begin{aligned} |\psi_1\rangle &= (H \otimes H)|0\rangle|1\rangle = (H|0\rangle) \otimes (H|1\rangle) = |+\rangle|-\rangle \\ &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) |-\rangle = \frac{|0\rangle|-\rangle + |1\rangle|-\rangle}{\sqrt{2}} \end{aligned}$$

$$|\psi_2\rangle = U_f \left(\frac{|0\rangle|-\rangle + |1\rangle|-\rangle}{\sqrt{2}} \right) = \frac{U_f|0\rangle|-\rangle + U_f|1\rangle|-\rangle}{\sqrt{2}}$$

Proposição: $U_f(|x\rangle|-\rangle) = (-1)^{f(x)}|x\rangle|-\rangle$

Analisando os Estados

$$|\psi_2\rangle = \begin{cases} \pm |+\rangle|-\rangle & \text{se } f(0) = f(1) \\ \pm |-\rangle|-\rangle & \text{se } f(0) \neq f(1) \end{cases}$$

$$|\psi_3\rangle = \begin{cases} \pm H|+\rangle H|-\rangle & \text{se } f(0) = f(1) \\ \pm H|-\rangle H|-\rangle & \text{se } f(0) \neq f(1) \end{cases}$$

$$|\psi_3\rangle = \begin{cases} \pm |0\rangle|1\rangle \\ \pm |1\rangle|1\rangle \end{cases}$$

Resultados

O algoritmo aplica a U_f na superposição, e tem complexidade 1 (1 consulta no oráculo)

Na computação clássica, a complexidade é igual ao número de entradas.

No caso específico, seriam 2 consultas.

No caso geral, seria $\frac{N}{2} + 1$ consultas no caso clássico (Algoritmo de Deutsch-Jozsa).

Vantagem computacional com apenas superposição

Como ele não usa emaranhamento, é possível implementar esse circuito apenas com 1 qubit (círculo econômico de Deutsch)

Implementando o algoritmo de *Deutsch*



Implementando o Circuito de Deutsch

```
from qiskit import QuantumCircuit  
from qiskit.providers.basic_provider import BasicProvider  
from qiskit.visualization import plot_histogram  
from qiskit.visualization import circuit_drawer  
import pylatexenc
```

Implementando o Circuito de Deutsch

```
def deutsch_function(case: int):
    # Essa função gera um circuito quântico para um dos quatro casos possíveis.
    if case not in [1, 2, 3, 4]:
        raise ValueError(`case` must be 1, 2, 3, or 4.)

    f = QuantumCircuit(2)
    if case in [2, 3]:
        f.cx(0, 1)
    if case in [3, 4]:
        f.x(1)

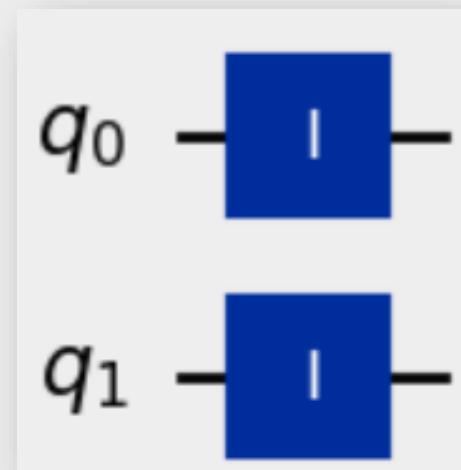
    return f
```

Implementando o Circuito de Deutsch

```
for i in range(1, 5):
    qc = deutsch_function(i)
    print(f"Circuito para o caso {i}:")
    display(qc.draw(output="mpl", style={"backgroundcolor": "#EEEEEE"}))
```

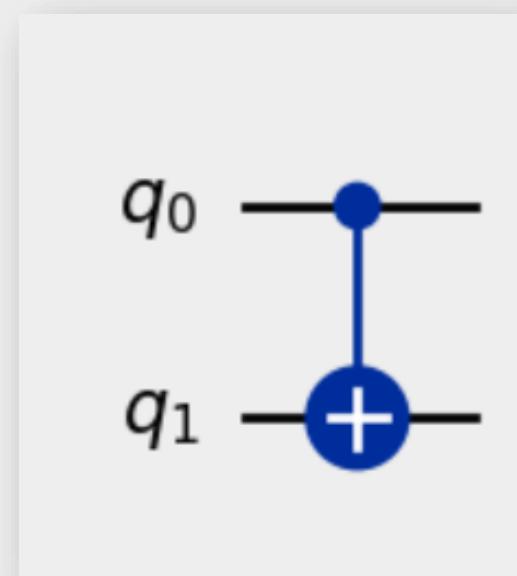
Possibilidades

f_0	f_1	f_2	f_3
$0 \rightarrow 0$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow 1$
$1 \rightarrow 0$	$1 \rightarrow 0$	$1 \rightarrow 0$	$1 \rightarrow 1$



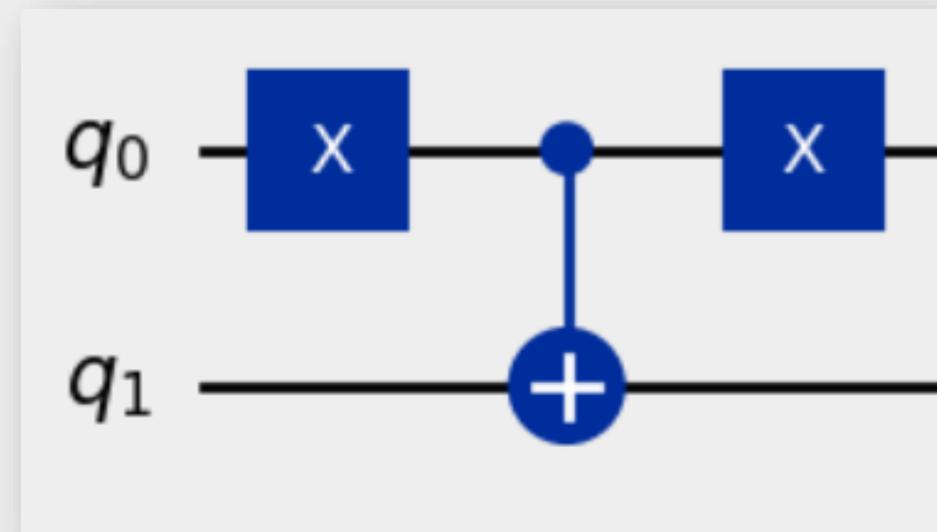
Possibilidades

f_0	f_1	f_2	f_3
$0 \rightarrow 0$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow 1$
$1 \rightarrow 0$	$1 \rightarrow 0$	$1 \rightarrow 0$	$1 \rightarrow 1$



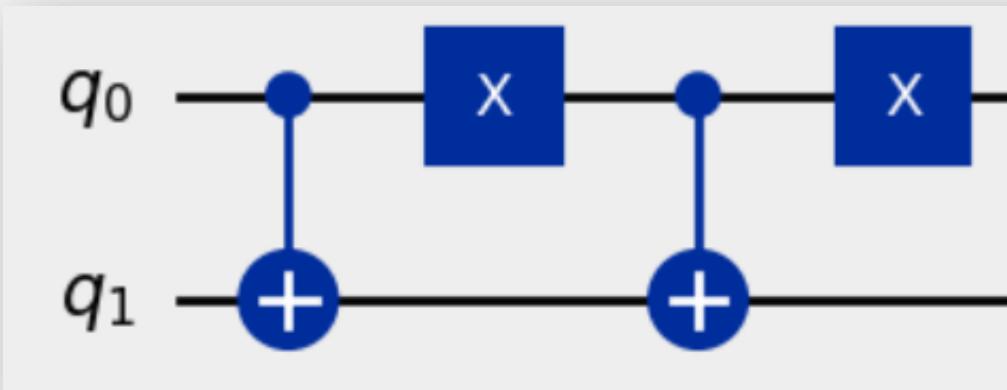
Possibilidades

f_0	f_1	f_2	f_3
$0 \rightarrow 0$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow 1$
$1 \rightarrow 0$	$1 \rightarrow 0$	$1 \rightarrow 0$	$1 \rightarrow 1$



Possibilidades

f_0	f_1	f_2	f_3
$0 \rightarrow 0$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow 1$
$1 \rightarrow 0$	$1 \rightarrow 0$	$1 \rightarrow 0$	$1 \rightarrow 1$



Implementando o Circuito de Deutsch

```
def compile_circuit(function: QuantumCircuit):
    # Compilando o circuito do algoritmo de Deustch.

    n = function.num_qubits - 1
    qc = QuantumCircuit(n + 1, n)

    qc.x(n)
    qc.h(range(n + 1))

    qc.barrier()
    qc.compose(function, inplace=True)
    qc.barrier()

    qc.h(range(n))
    qc.measure(range(n), range(n))

return qc
```

Implementando o Circuito de Deutsch

```
display(compile_circuit(deutsch_function(3)).draw(output="mpl", style={"backgroundcolor": "#EEEEEE"}))
```

Implementando o Circuito de Deutsch

1.

```
def deutsch_algorithm(function: QuantumCircuit):
    # Determina se a função é constante ou balanceada.

    qc = compile_circuit(function)

    result = AerSimulator().run(qc, shots=1, memory=True).result()
    measurements = result.get_memory()
    if measurements[0] == "0":
        return "constante"
    return "balanceada"
```

Implementando o Circuito de Deutsch

```
for i in range(1,5):
    print(f"Caso {i}:")
    f = deutsch_function(i)
    display(deutsch_algorithm(f))
```

Algoritmo de Deutsch-Jozsa

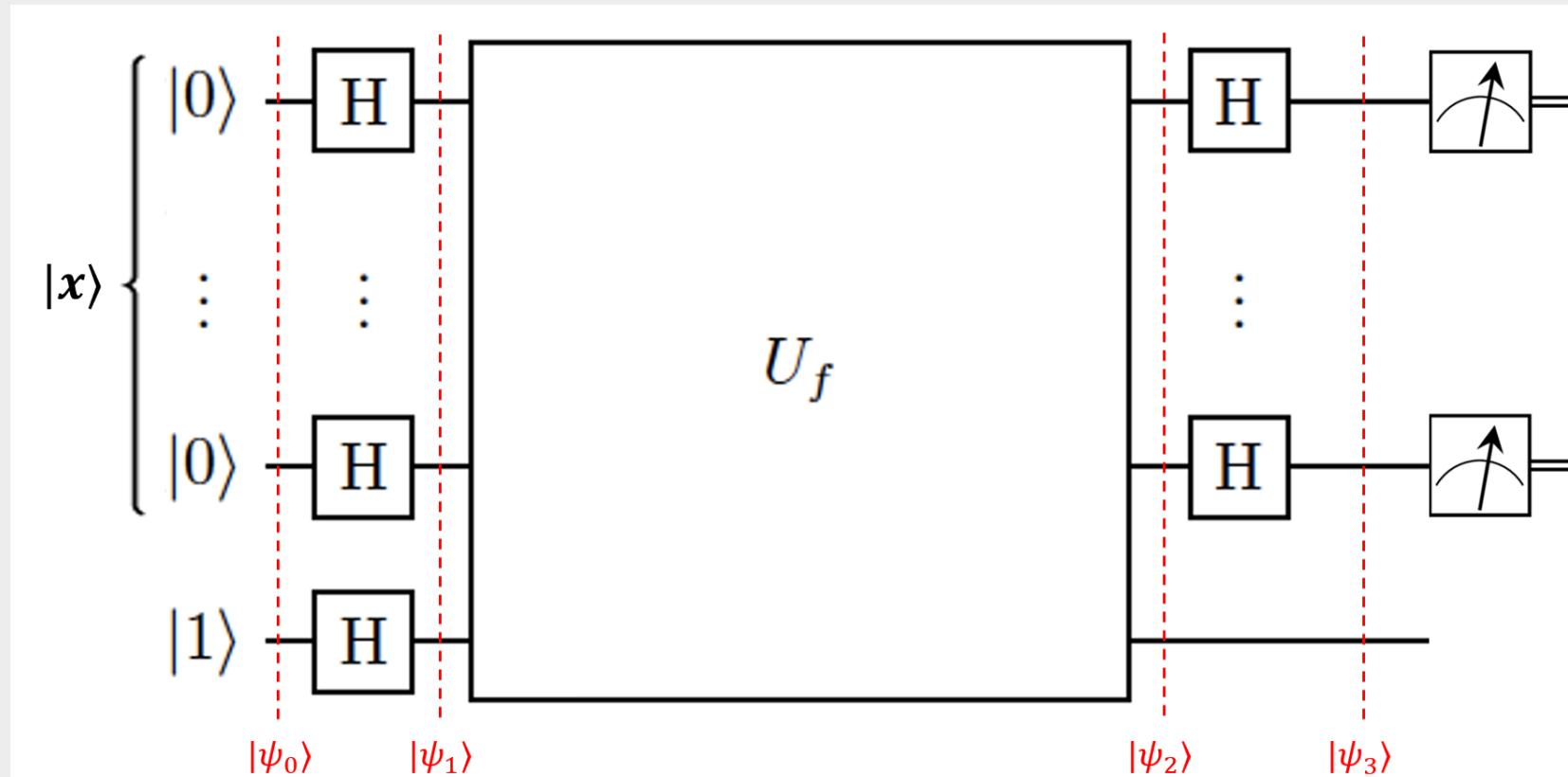
Caso geral do Algoritmo de Deutsch

Proposto por David Deutsch e Richard Jozsa em 1992

Determinar se a função é constante ou balanceada, para N entradas

Assim como o algoritmo anterior, está limitada por esses tipos de funções

Algoritmo de Deutsch-Jozsa



Entrada de $f: \{0,1\}^n \rightarrow \{0,1\}$

Saída: $\begin{cases} 0 & \text{se } f \text{ é constante} \\ \text{caso contrário } f \text{ é balanceada} \end{cases}$

Definição

Seja $f: \{0,1\}^N \rightarrow \{0,1\}$ uma função booleana de N entradas. Consideramos ainda que a função respeita uma das seguintes propriedades:

- $f(x) = 0 \forall x$ ou $f(x) = 1 \forall x$ - Constante
- 2^{N-1} valores são $f(x) = 0$ e 2^{N-1} valores são $f(x) = 1$ – Balanceada

Minimizando o número de consultas à f , como descobrir se é constante ou balanceada.

Para o caso clássico

Pior caso, só temos conclusão após medir $\frac{N}{2} + 1$ vezes

Se optarmos por concluir que f é constante após k avaliações, a probabilidade de acerto é dada por:

$$P_c(k) = 1 - \frac{1}{2^{k-1}}$$

Algorithm 1 Algoritmo clássico

```
1: for  $x = 1$  to  $N/2 = 2^{n-1}$  do
2:   if  $f(x)! = f(0)$  then
3:     return Balanceado
4:   end if
5: end for
6: return Constante
```

Solução Quântica

Codificamos a f da mesma forma onde $U_f|x\rangle|j\rangle = |x\rangle|j\oplus f(x)\rangle$

Porém nesse caso nosso primeiro registrador $|x\rangle$ possui N qubits, enquanto o segundo, possui apenas um.

U_f precisa ser um operador unitário. Ou seja, $U_f^\dagger U_f = I$.

Isto sempre é verdade?

Como descobrir se U_f é unitária

Para mostrar que uma matriz é unitária precisamos que ela satisfaça o seguinte:

$$\langle j | A | i \rangle = A_{ij} = \delta_{ij} = I$$

Porém estamos trabalhando com $N + 1$ qubits

Onde $x \in \{0, \dots, 2^N - 1\}$ e $j \in \{0, 1\}$

Como descobrir se U_f é unitária

Produto de Kronecker
 $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$

Então

$$\langle x' | x \rangle \langle j' + f(x') | j + f(x) \rangle$$

$$\langle i | j \rangle = \delta_{ij}$$

Se $x \neq x' \Rightarrow \delta x' x = 0$
Portanto
 $x' = x$

$$\langle x' | \langle j' | U^\dagger U | x \rangle | j \rangle$$

$$(\langle x' | \langle j' | U^\dagger)(U | x \rangle | j \rangle)$$

$$((U | x' \rangle | j' \rangle)^\dagger (U | x \rangle | j \rangle))$$

$$(|x'\rangle |j' + f(x')\rangle) 1^\dagger (|x\rangle |j + f(x)\rangle)$$

$$(\langle x' | \langle j + f(x') |) (|x\rangle |j + f(x)\rangle)$$

O algoritmo

Algoritmo de Deutsch-Jozsa	
Entrada	$f: \{0,1\}^n \rightarrow \{0,1\}$
Passo 1	Preparar os estados $ 0\rangle^{\otimes n}$ e $ 1\rangle$
Passo 2	Gerar superposição aplicando $H^{\otimes n+1}$ nos qubits
Passo 3	Aplicar U_f
Passo 4	Aplicar $H^{\otimes n+1}$
Passo 5	Medir o primeiro registrador na base computacional
Saída	$\begin{cases} 0 & \text{se } f \text{ é constante} \\ \text{caso contrário} & f \text{ é balanceada} \end{cases}$

Analizando os estados

A entrada é dada por

$$|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$$

onde $|0\rangle^{\otimes n}$ equivale a $|0\rangle \otimes |0\rangle \otimes \dots \otimes |0\rangle$

Aplicando H em todos os qubits temos:

$$|\psi_1\rangle = (H|0\rangle)^{\otimes n} \otimes H|1\rangle$$

Podendo ser reescrito como:

$$|\psi_1\rangle = |+\rangle|+\rangle\dots|+\rangle|-\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \dots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |-\rangle = \frac{1}{\sqrt{2^n}} [(|0\rangle + |1\rangle) \dots (|0\rangle + |1\rangle) |-\rangle]$$

Proposição 1

Seja $f: \{0,1\}^n \rightarrow \{0,1\}$ uma função Booleana de n -bits, defina U_f como

$$U_f = \sum_{x \in \{0,1\}^n} \sum_{j=0}^1 |x,j \oplus f(x)\rangle \langle x,j|.$$

Então,

$$U_f(|x\rangle \otimes |- \rangle) = (-1)^{f(x)} |x\rangle \otimes |- \rangle.$$

Analizando os estados

Como o estado anterior permite obter todas as combinações possíveis, podemos dizer que

$$|\psi_1\rangle = \sum_{x=0}^{2^n-1} |x\rangle |-\rangle.$$

Passando por uma U_f temos

$$|\psi_2\rangle = U_f |\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle |-\rangle$$

Analisando os estados

Aplicando H novamente temos

$$|\psi_3\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} (H^{\otimes n}|x\rangle)(H|-\rangle)$$

Proposição 2:

Seja $x \in \{0,1\}^n$ um bit de tamanho $n, (x_0, x_1, \dots, x_{n-1})$. Então,

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{x \cdot y}|y\rangle,$$

onde $x \cdot y = x_0y_0 + \dots + x_{n-1}y_{n-1}$, a soma nesse caso é mod 2.

Analizando os estados

$$|\psi_3\rangle = \frac{1}{2^n} \sum_{y=0}^{2^n-1} \left(\sum_{x=0}^{2^n-1} (-1)^{x \cdot y + f(x)} \right) |y\rangle |1\rangle$$

Chamando $g(y) = \left(\frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{x \cdot y + f(x)} \right)$, temos

$$|\psi_3\rangle = \sum_y g(y) |y\rangle |1\rangle$$

Diferentemente do algoritmo anterior, onde tínhamos um único estado ao final do algoritmo (antes da medição), aqui nós temos uma superposição de vários termos, onde $g(y)$ é a amplitude de probabilidade para os possíveis resultados.

Analizando os estados

Se $y = 0$, a amplitude será

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{x \cdot 0 + f(x)} = \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)}$$

Mas $f(x)$ pode ser constante ou balanceada, caso seja constante, $f(0) = f(y)$:

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} \pm 1 = \pm \frac{\sum_{x=0}^{2^n-1} 1}{2^n} = \frac{2^n}{2^n} = \pm 1$$

Amplitude de probabilidade = 100%

Analisando os estados

Por outro lado, se f for balanceada, pela definição, temos que para cada 1 que tivermos no somatório, teremos um -1 , resultando em uma amplitude que pode ser reescrita como uma soma de termos que se anulam.

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} = \frac{1}{2^n} \left(\sum_x (-1)^0 + \sum_x (-1)^1 \right) = 0.$$

Sendo assim

$$\begin{cases} 0 & \text{se } f \text{ é constante} \\ \text{caso contrário} & f \text{ é balanceada} \end{cases}$$



Obrigado !

Colocar um QR
code aqui com
as
oportunidades
do Quiin

