COMPUTACIÓN CUÁNTICA EN LA ERA DE LA IA: UNA PERSPECTIVA DESDE LA FÍSICA

NOELIA SÁNCHEZ GÓMEZ

II CONGRESO DE INTELIGENCIA ARTIFICIAL DE ANDALUCÍA

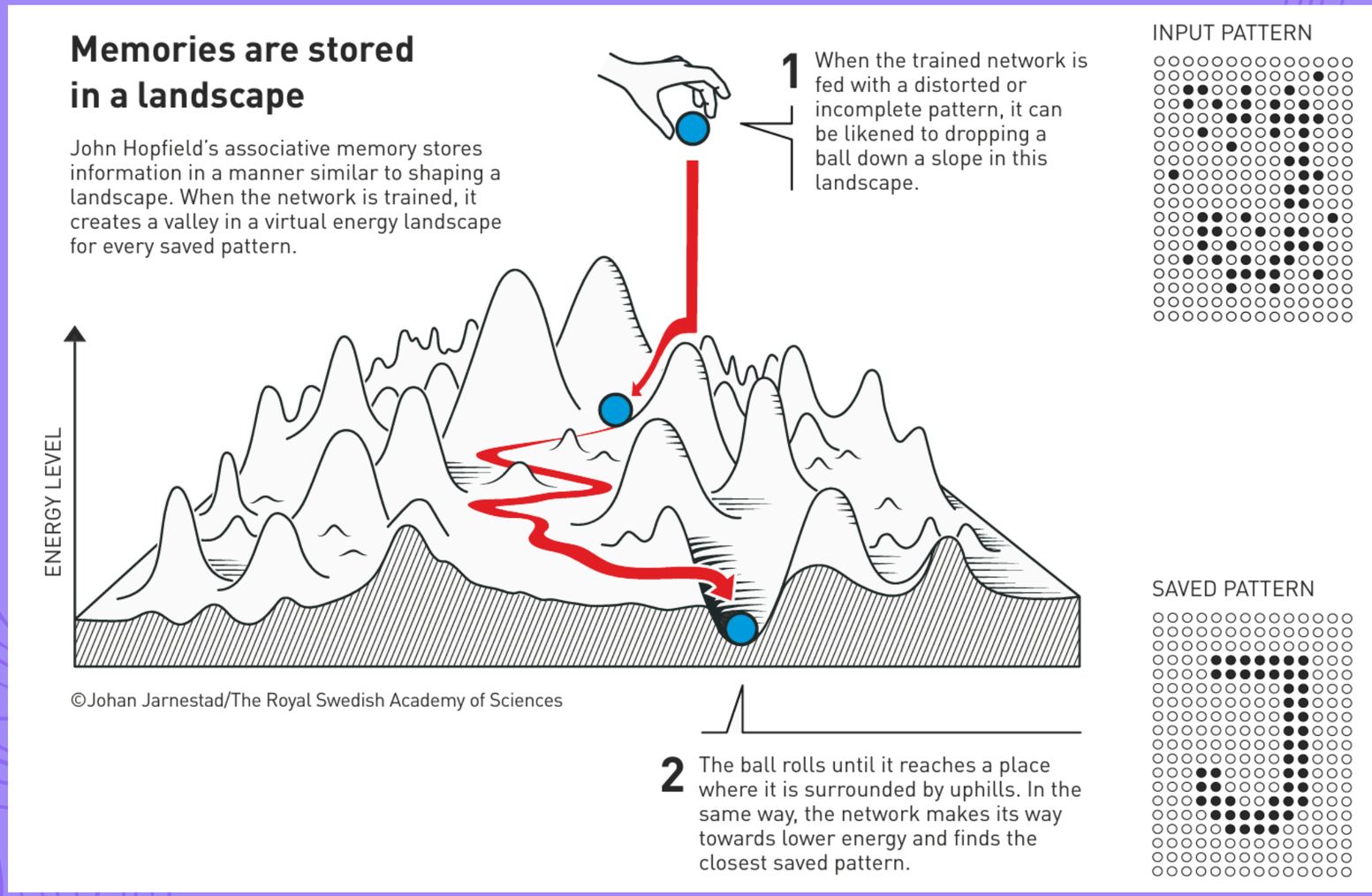
8 October 2024

The Royal Swedish Academy of Sciences has decided to award the Nobel Prize in Physics 2024 to

John J. Hopfield
Princeton University, NJ, USA

Geoffrey Hinton
University of Toronto, Canada

"for foundational discoveries and inventions that enable machine learning with artificial neural networks"

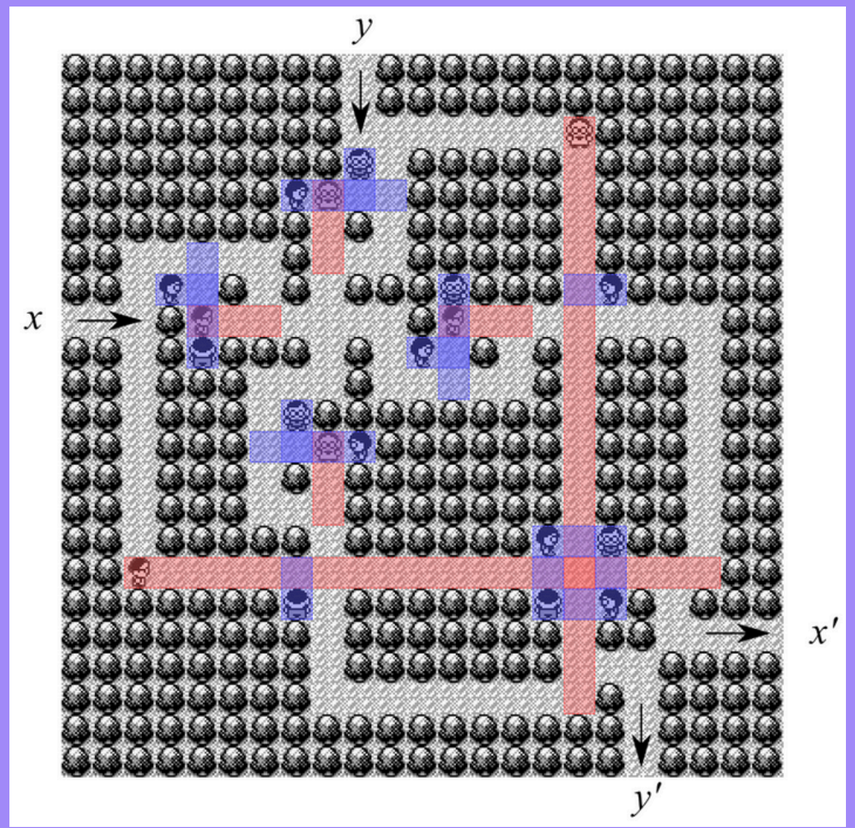


COMPUTACIÓN CUÁNTICA EN LA ERA DE LA IA: UNA PERSPECTIVA DESDE LA FÍSICA

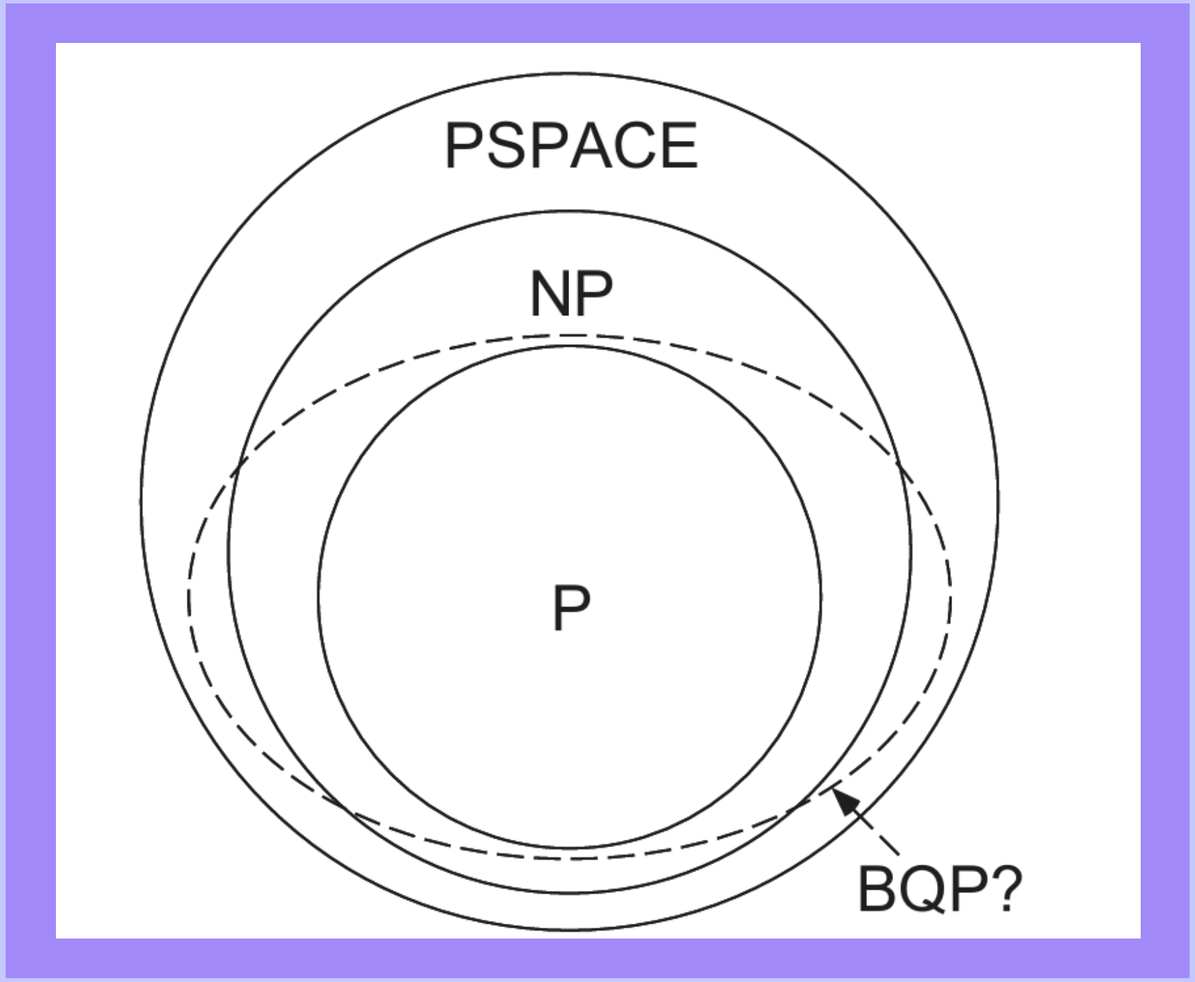
Classic Nintendo Games are (Computationally) Hard

Greg Aloupis* Erik D. Demaine† Alan Guo†‡ Giovanni Viglietta§

February 10, 2015

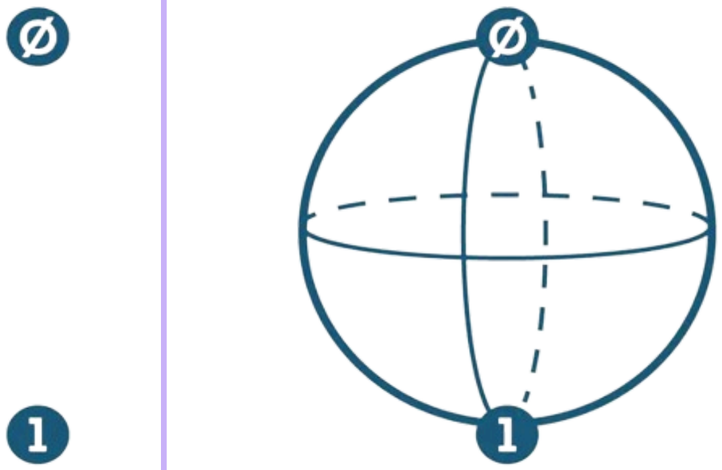


CLASES DE COMPLEJIDAD



BIT

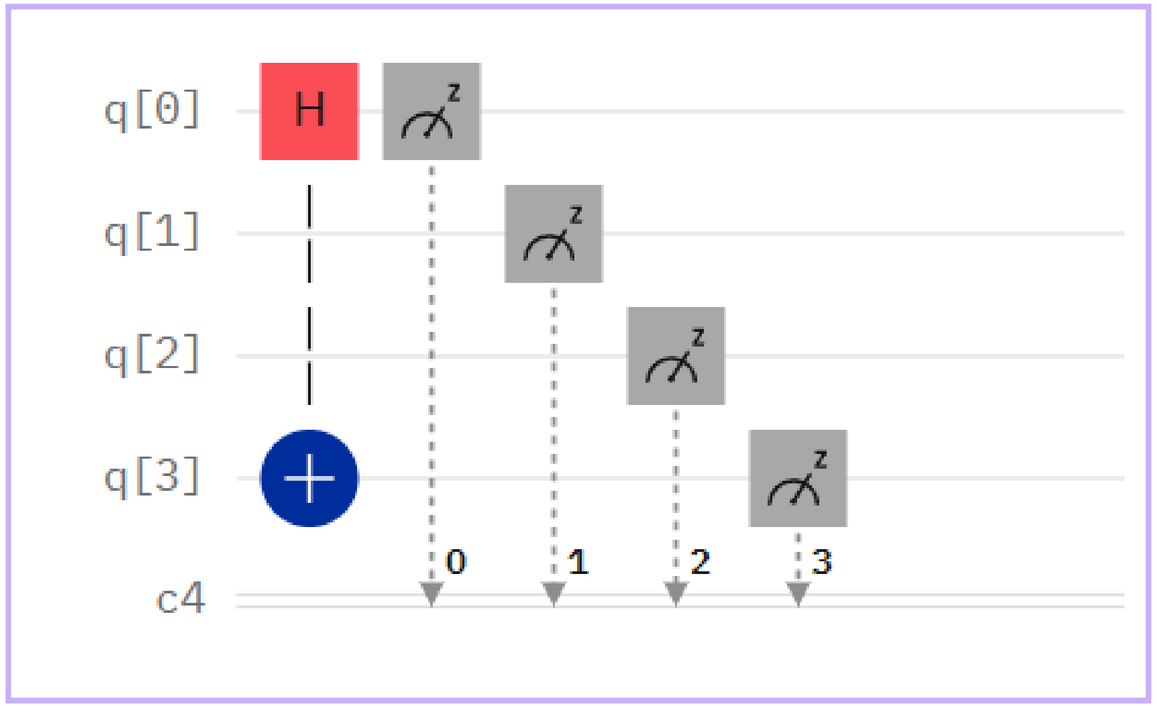
QUBIT



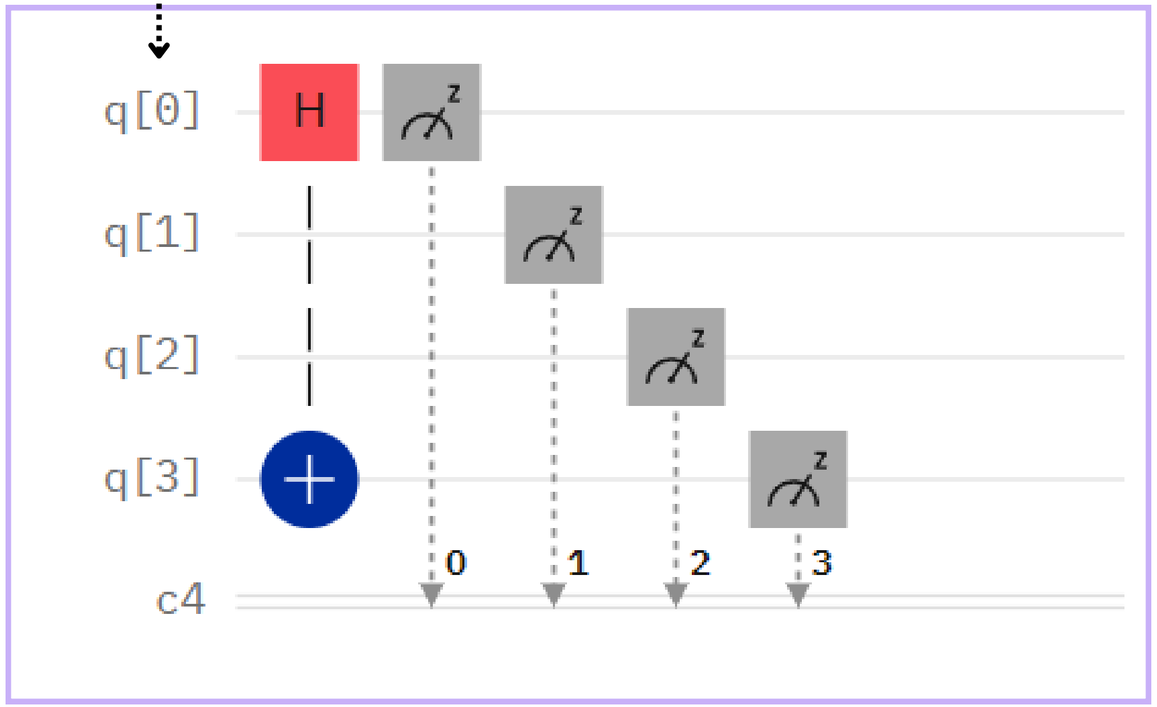
$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

- Estados posibles (antes de medir):
  - $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$
- Resultados posibles (después de medir):
  - Valor 0, con probabilidad $|\alpha|^2$.
  - Valor 1, con probabilidad $|\beta|^2$.

Circuito cuántico

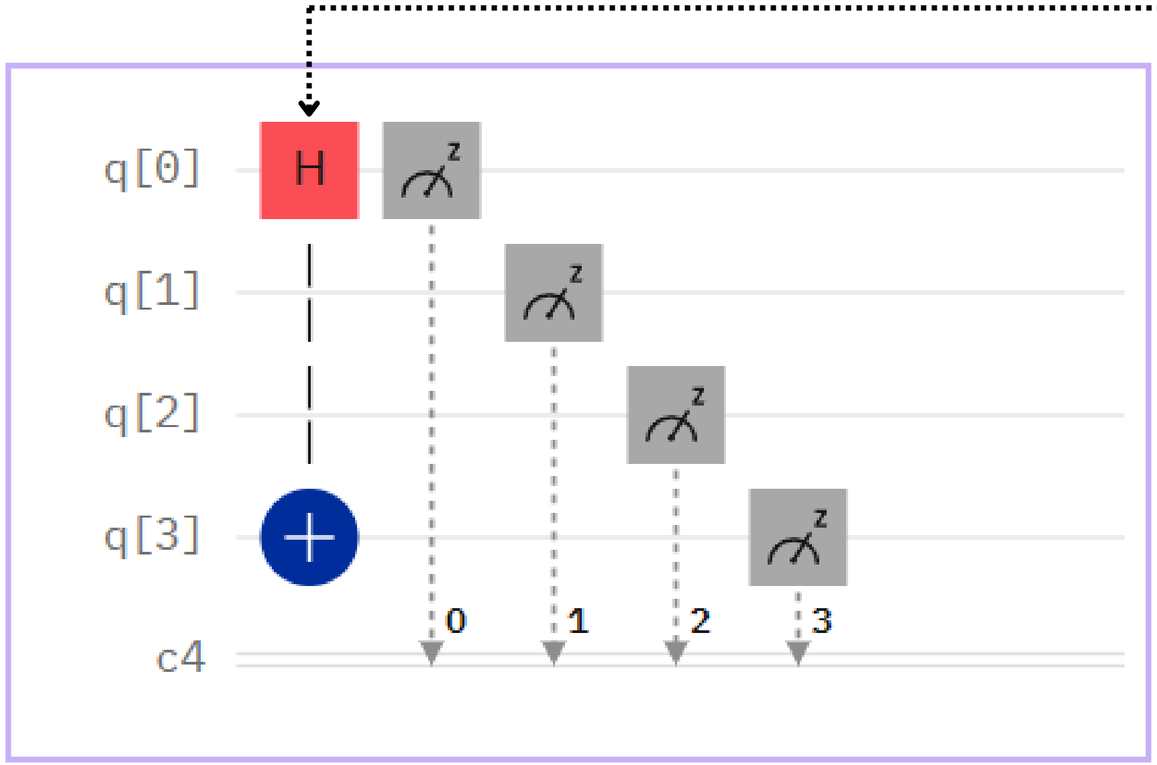


Circuito cuántico



Circuito de 4 qubits. Estado inicial compuesto por todo $|0\rangle$s.

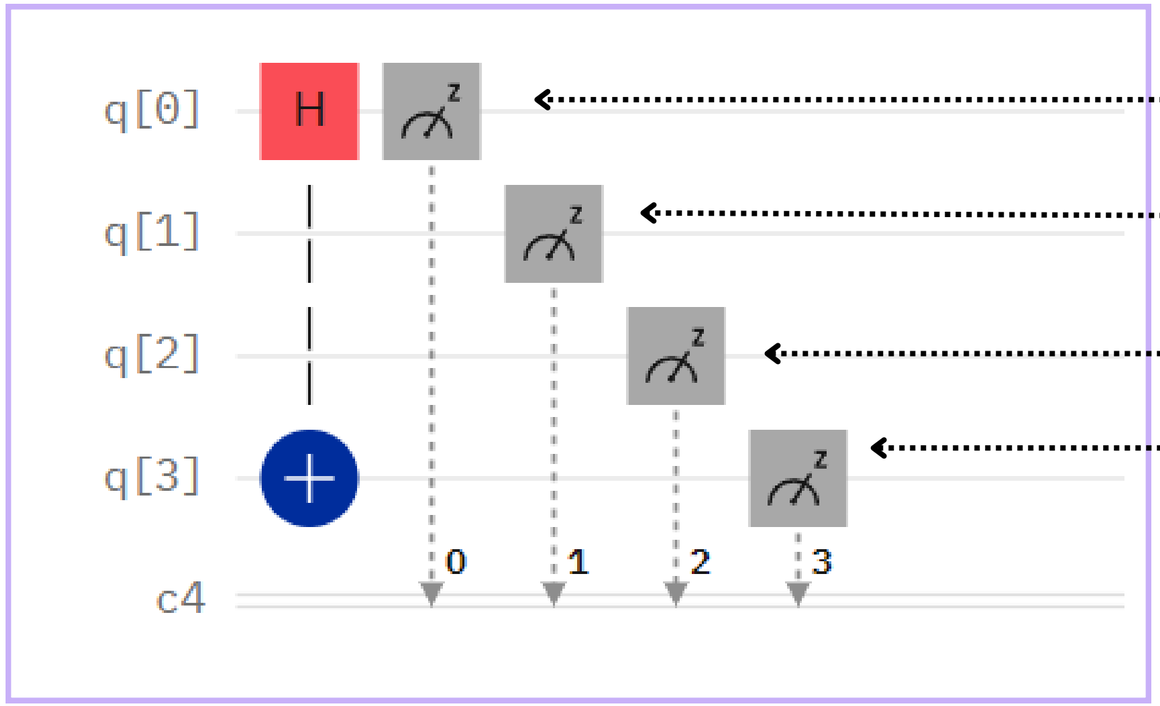
Circuito cuántico



Circuito de 4 qubits. Estado inicial compuesto por todo $|0\rangle$s.

Puertas cuánticas: operan sobre el estado.

Circuito cuántico

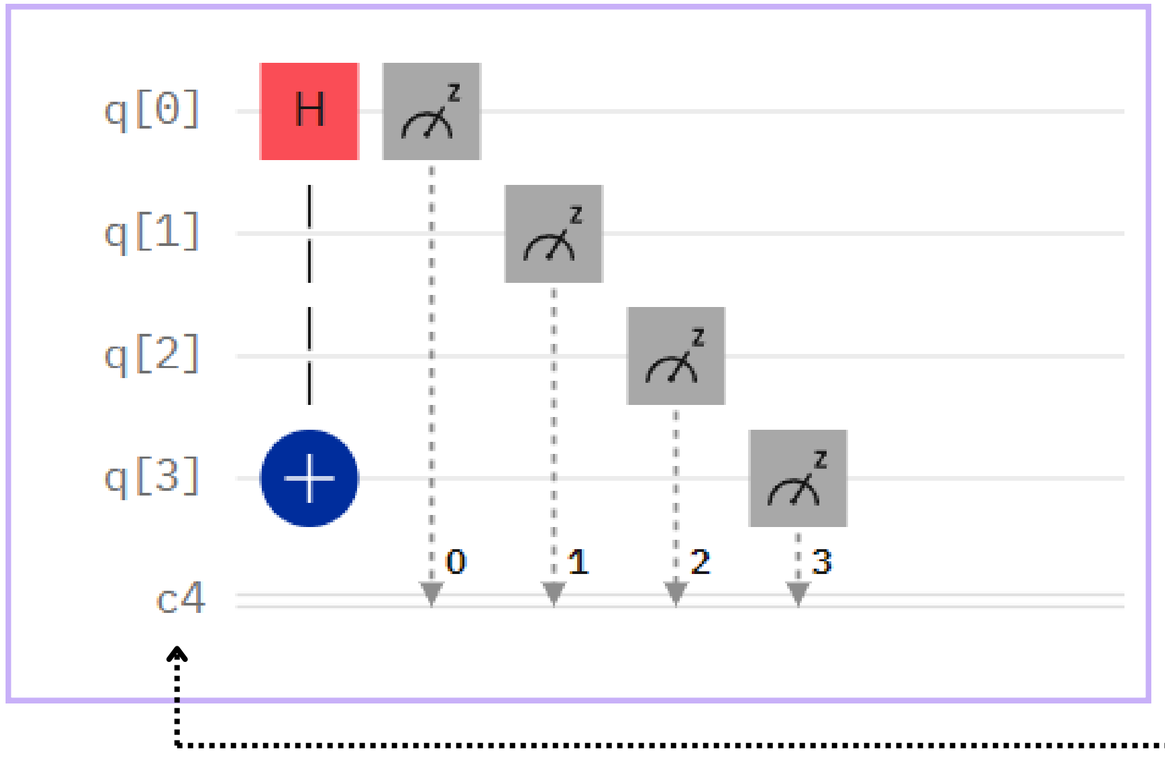


Circuito de 4 qubits. Estado inicial compuesto por todo $|0\rangle$s.

Puertas cuánticas: operan sobre el estado.

Proceso de medida, generalmente irreversible: la función de onda colapsa a un estado.

Circuito cuántico



Circuito de 4 qubits. Estado inicial compuesto por todo $|0\rangle$s.

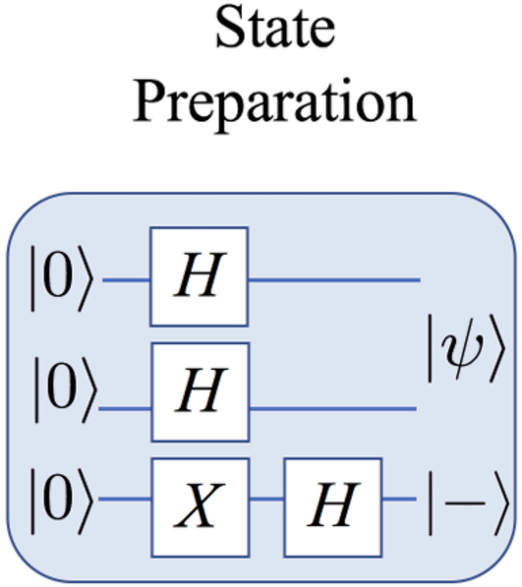
Puertas cuánticas: operan sobre el estado.

Proceso de medida, generalmente irreversible: la función de onda colapsa a un estado.

Bits donde se registra la salida.

Algoritmo de Grover

Algoritmo de Grover

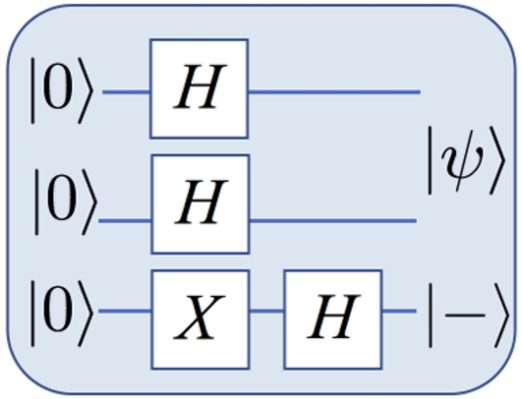


Quantum Algorithm Implementations for Beginners

ABHIJITH J.*, ADETOKUNBO ADEDOYIN, JOHN AMBROSIANO, PETR ANISIMOV, WILLIAM CASPER, GOPINATH CHENNUPATI, CARLETON COFFRIN, HRISTO DJIDJEV, DAVID GUNTER, SATISH KARRA, NATHAN LEMONS, SHIZENG LIN, ALEXANDER MALYZHENKOV, DAVID MASCARENAS, SUSAN MNISZEWSKI, BALU NADIGA, DANIEL O'MALLEY, DIANE OYEN, SCOTT PAKIN, LAKSHMAN PRASAD, RANDY ROBERTS, PHILLIP ROMERO, NANDAKISHORE SANTHI, NIKOLAI SINITSYN, PIETER J. SWART, JAMES G. WENDELBERGER, BORAM YOON, RICHARD ZAMORA, WEI ZHU, STEPHAN EIDENBENZ*, ANDREAS BÄRTSCHI*, PATRICK J. COLES*, MARC VUFFRAY*, and ANDREY Y. LOKHOV*,

Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA

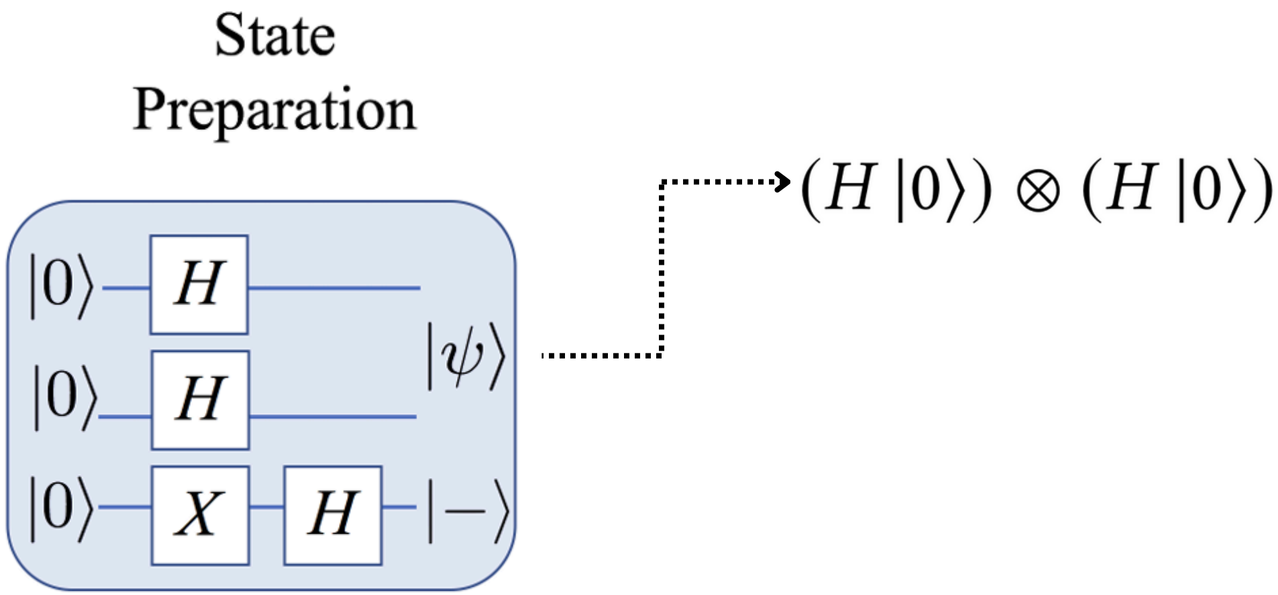
Algoritmo de Grover



Superposición uniforme de todos los elementos de la base: todos los elementos que conforman nuestra base de datos.

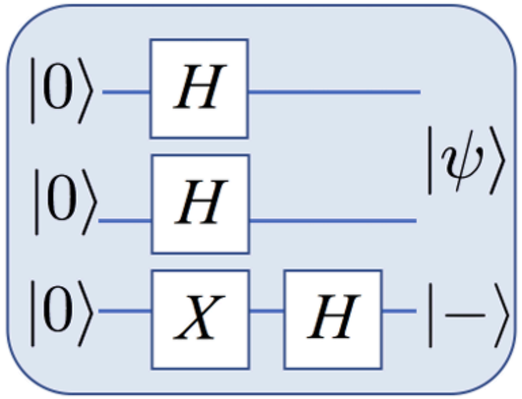
Qubit ancilla o auxiliar.

Algoritmo de Grover



Algoritmo de Grover

State Preparation

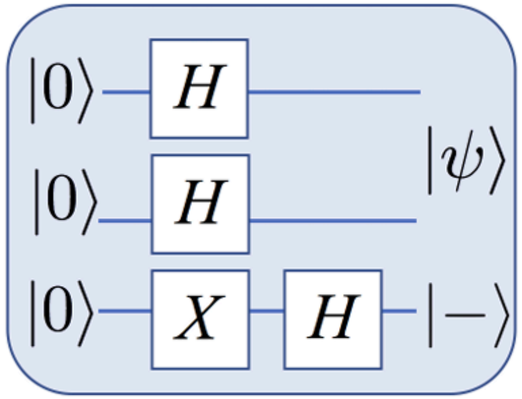


$$(H\,|0\rangle) \otimes (H\,|0\rangle)$$

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

Algoritmo de Grover

State Preparation

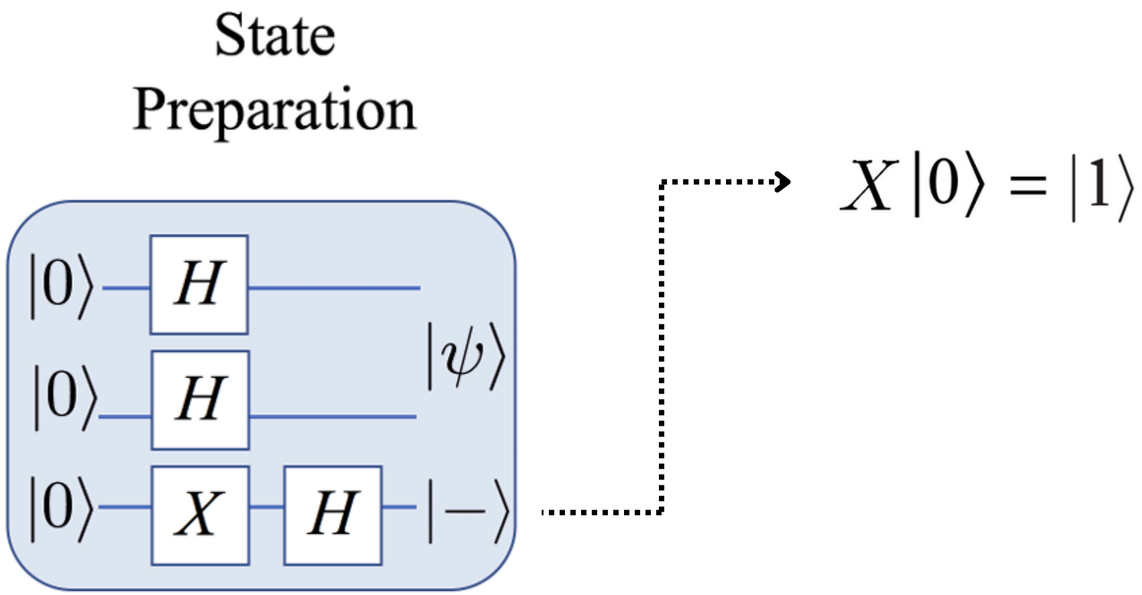


$$(H|0\rangle) \otimes (H|0\rangle)$$

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

$$\frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}$$

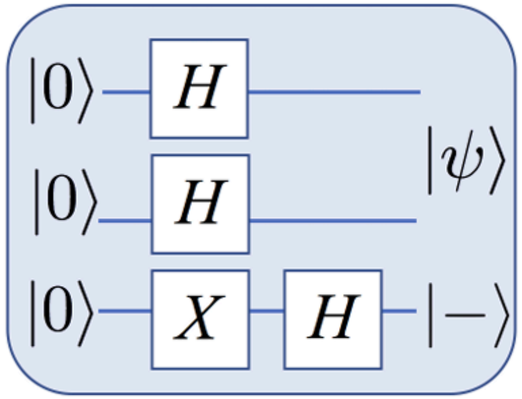
Algoritmo de Grover



$$X|0\rangle = |1\rangle$$

Algoritmo de Grover

State Preparation



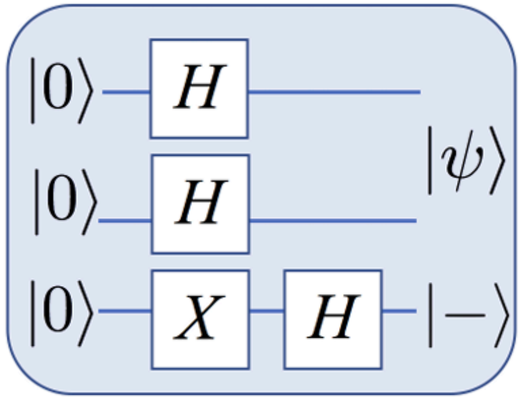
$$X|0\rangle = |1\rangle$$

$$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Algoritmo de Grover

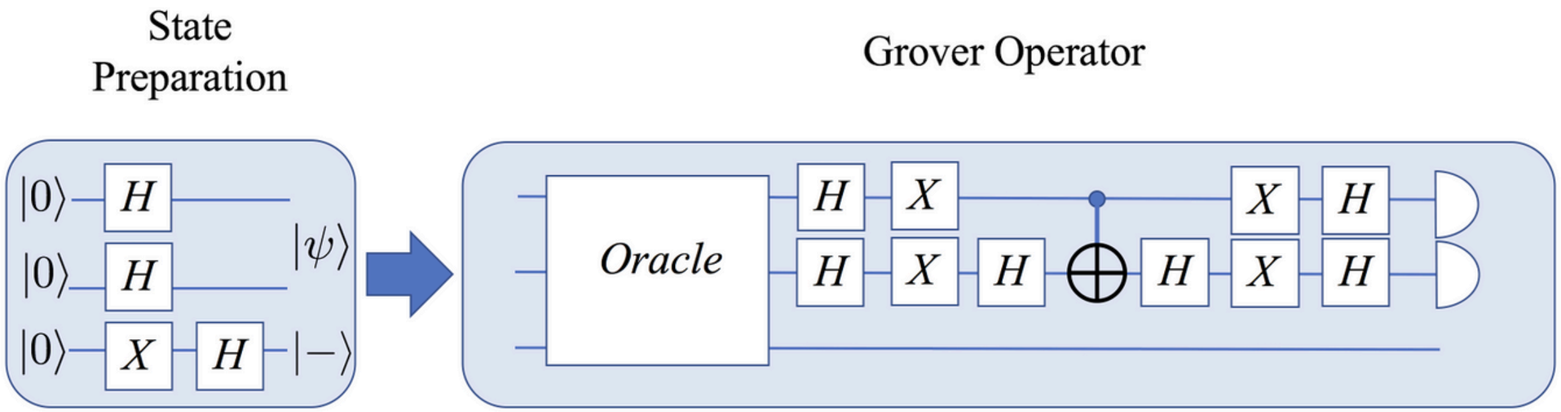
Qiskit
Kit de desarrollo de software

State Preparation

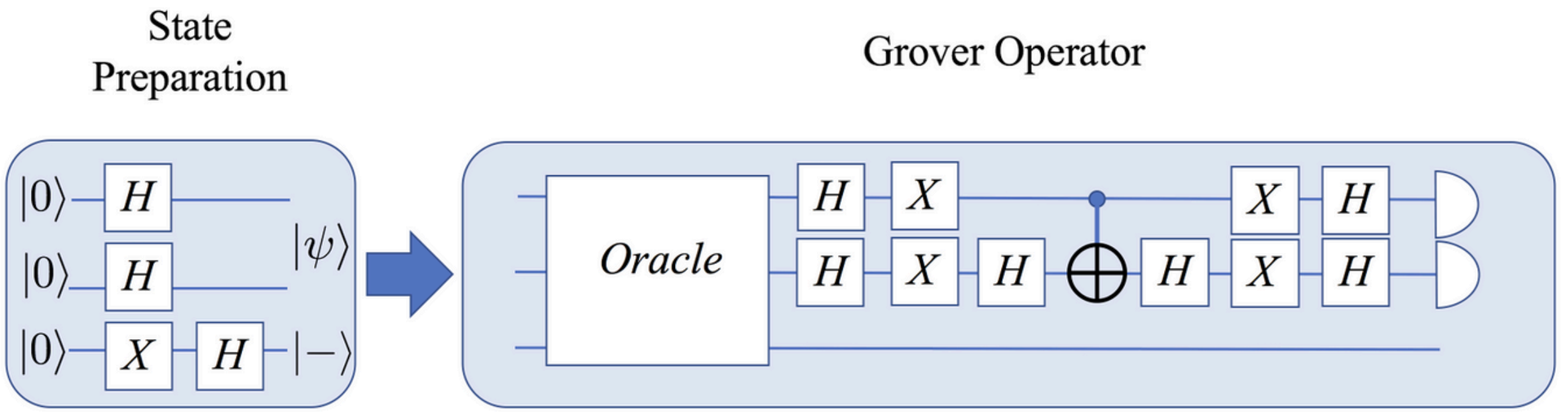



```python
from qiskit import QuantumCircuit

n=3
qc=QuantumCircuit(n,n-1)

#inicialización del estado
qc.x(2)
qc.h(0)
qc.h(1)
qc.h(2)
```

Algoritmo de Grover

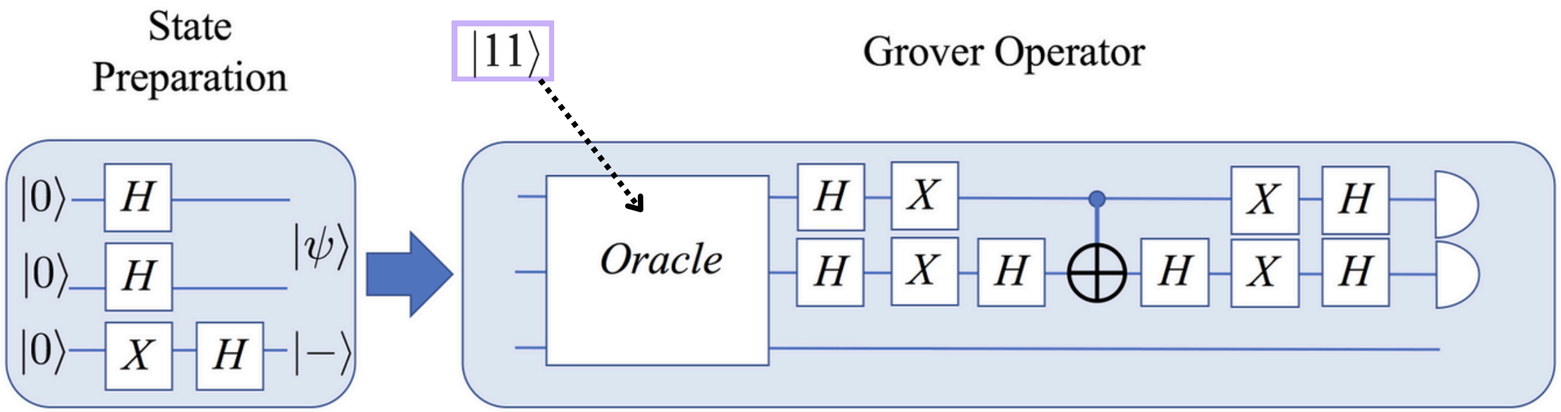


Algoritmo de Grover



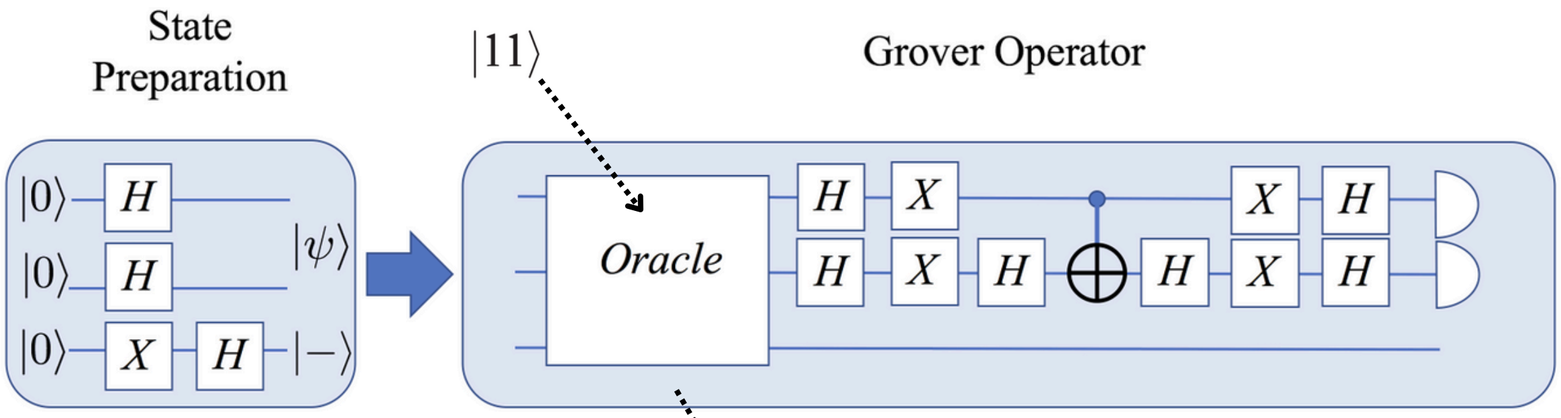
$$\frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

Algoritmo de Grover



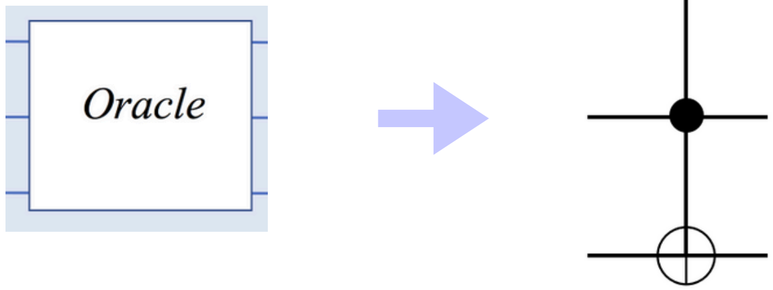
$$\frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

Algoritmo de Grover



$$\frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = \frac{|00\rangle + |01\rangle + |10\rangle - |11\rangle}{2} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

Algoritmo de Grover



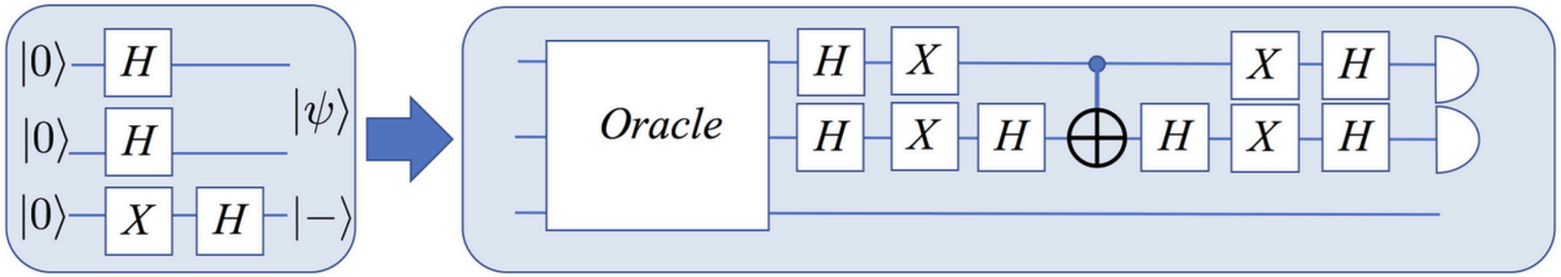

```
qc.ccx(0, 1, 2)
```

INPUT			OUTPUT		
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Algoritmo de Grover

State Preparation

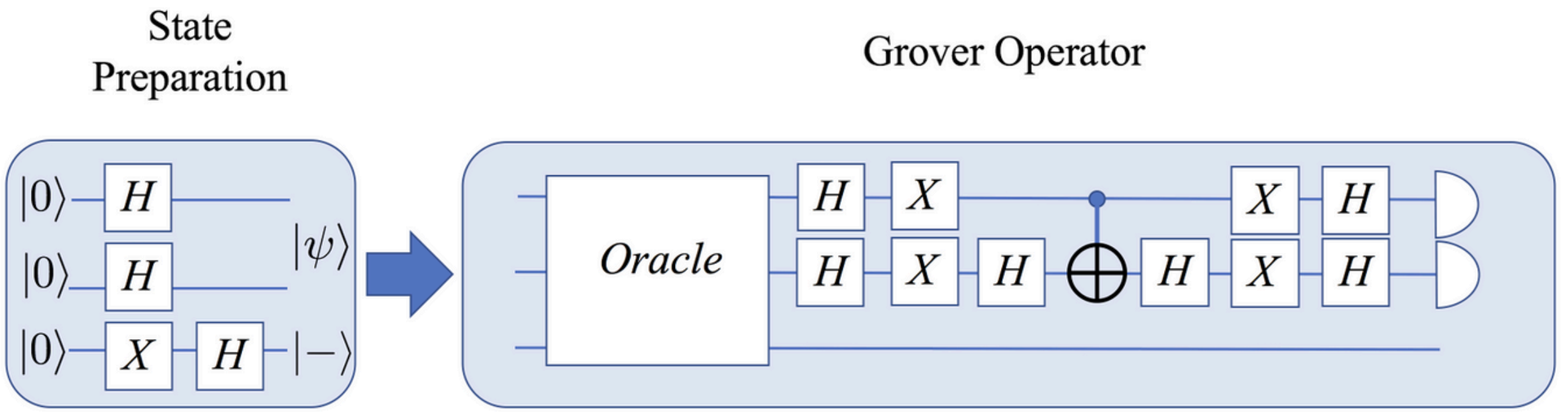
Grover Operator $G = (2|\psi\rangle\langle\psi| - I)$



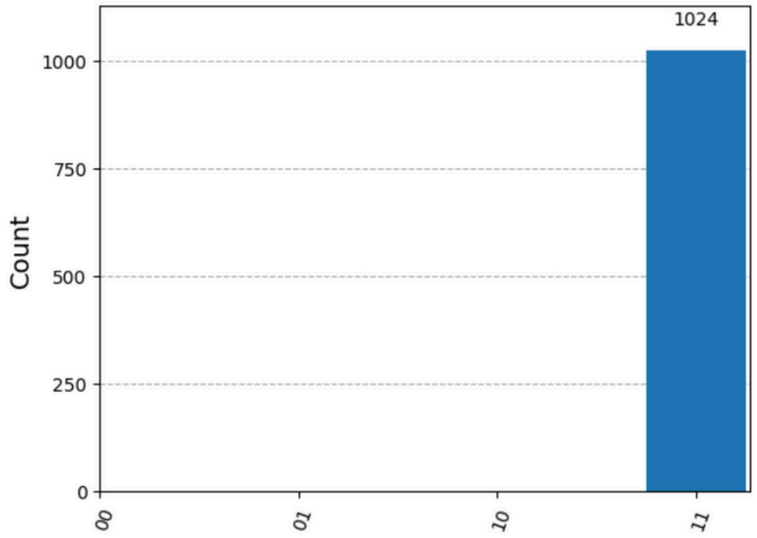
$$\left\lceil \frac{\pi\sqrt{N}}{4} \right\rceil$$

$$\frac{|00\rangle + |01\rangle + |10\rangle - |11\rangle}{2} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

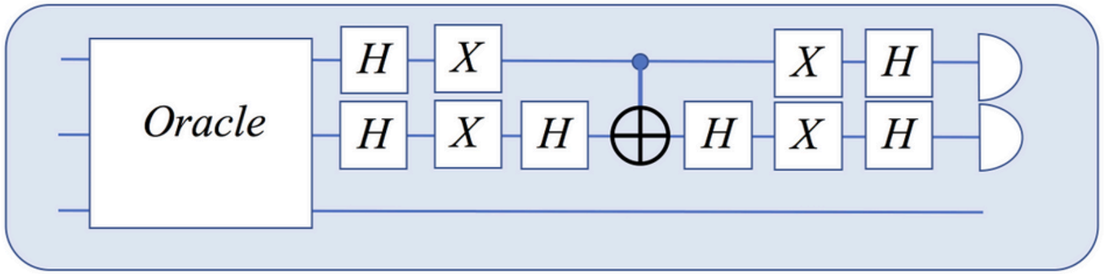
Algoritmo de Grover



$$\frac{|00\rangle + |01\rangle + |10\rangle - |11\rangle}{2} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$



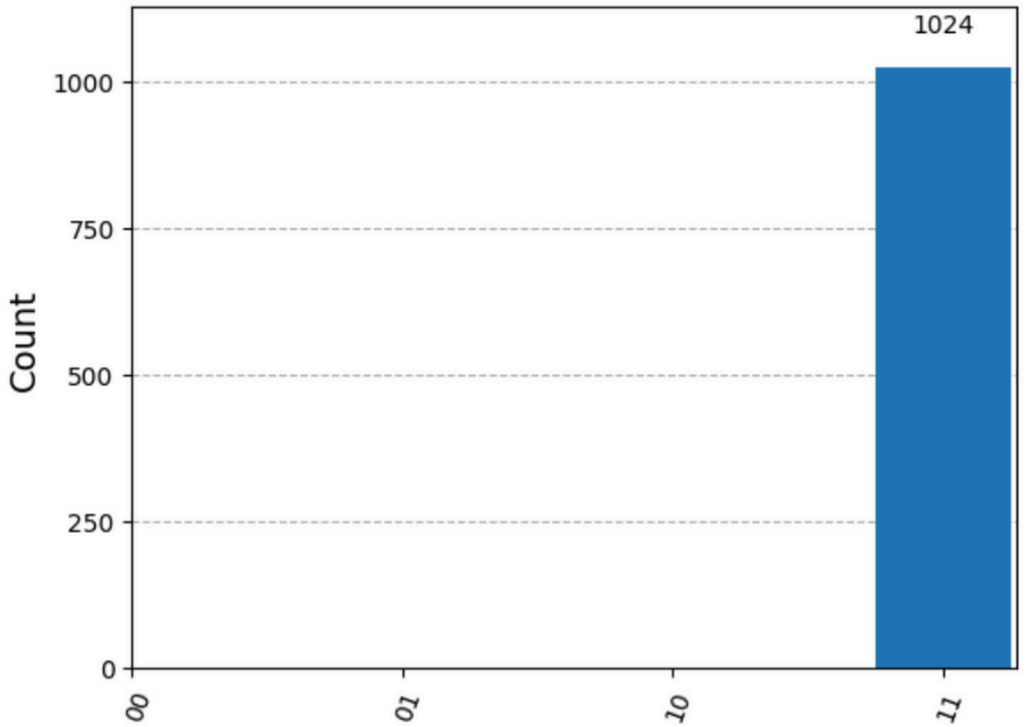
Algoritmo de Grover




```
grover(qc, 0, 1, 2)

for i in range(0,n-1):
    qc.measure(i, i)
```

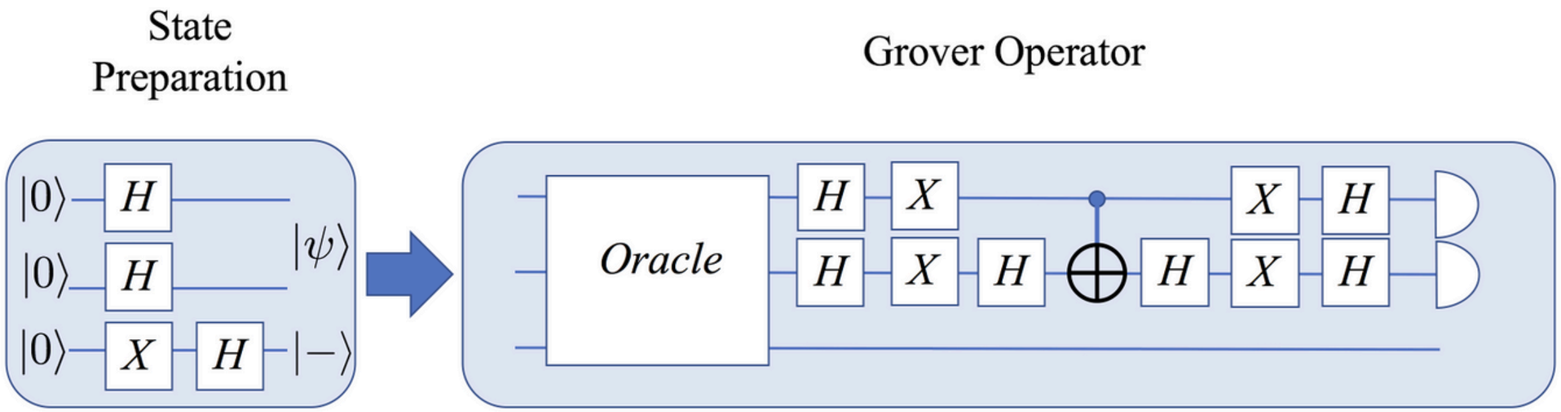
```
def grover(qc, x0, x1, q):

    qc.ccx(0, 1, 2)

    qc.h(x0)
    qc.h(x1)
    qc.x(x0)
    qc.x(x1)
    qc.h(x1)
    qc.cx(x0, x1)
    qc.h(x1)
    qc.x(x0)
    qc.x(x1)
    qc.h(x0)
    qc.h(x1)
```

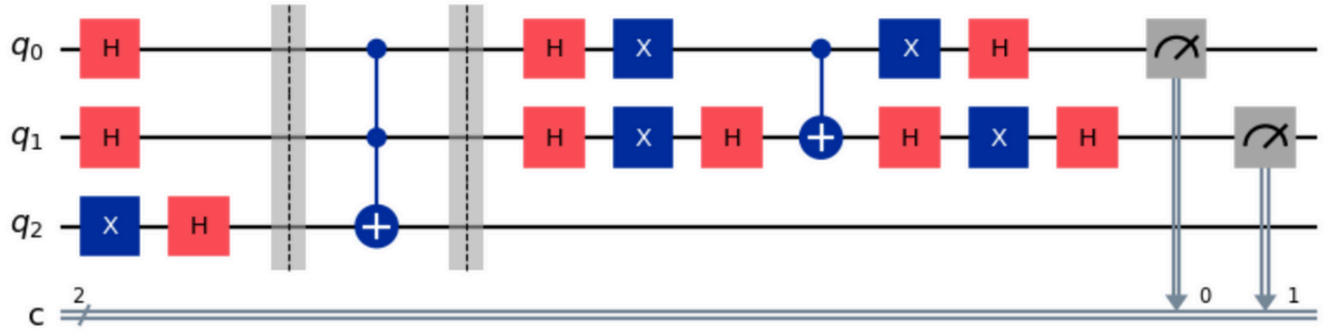
Algoritmo de Grover




```qiskit_aer import Aer
from qiskit.visualization import plot_histogram

sim=Aer.get_backend('aer_simulator')
result=sim.run(qc).result()
counts=result.get_counts()
plot_histogram(counts)
```

Algoritmo de Grover





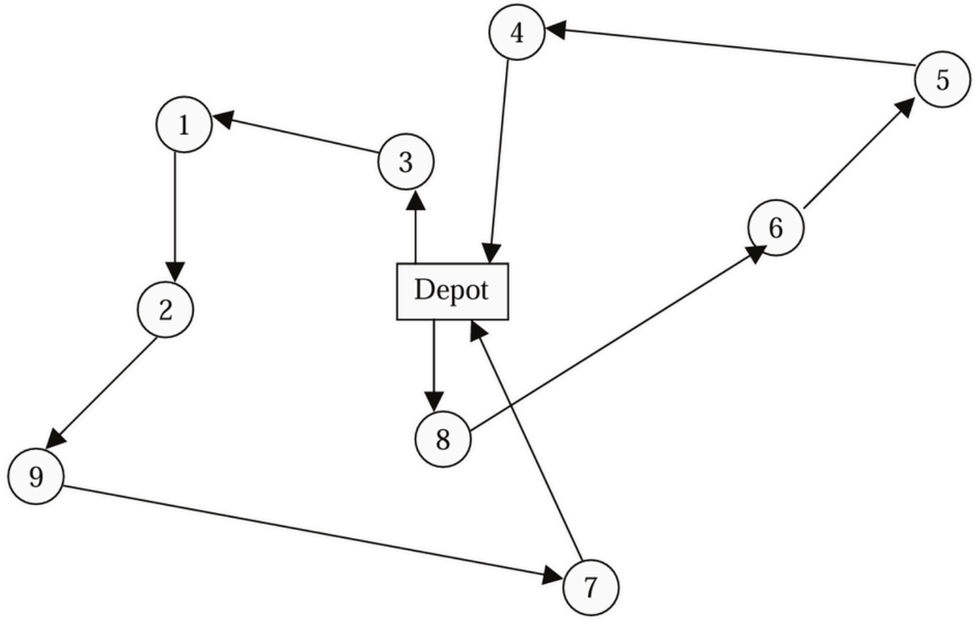

```
qc.draw(output='mpl', scale=0.8, fold=20)
```

QAOA

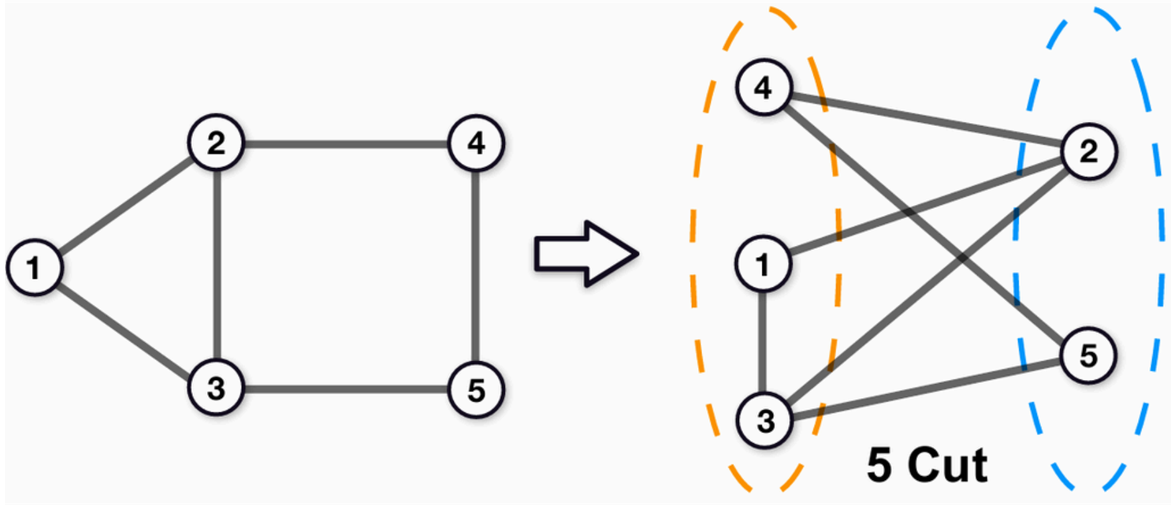
Quantum Approximate Optimization Algorithm

QAOA

- Problema de Enrutamiento de Vehículos. (Vehicle Routing Problem (VRP))



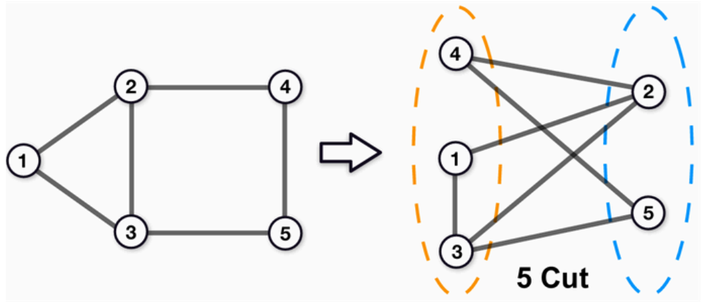
- Problema de Corte Máximo (Max-Cut).
  - Enrutamiento en redes de comunicación.
  - Distribución de recursos en redes de computadoras.



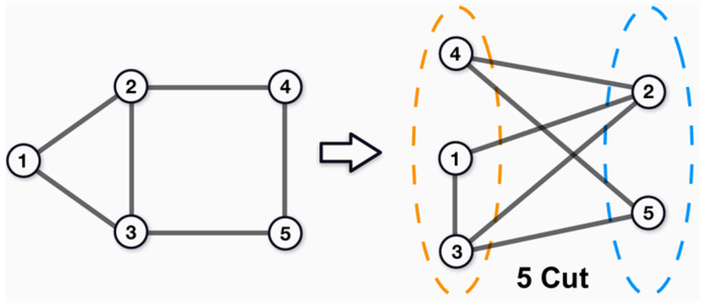
QAOA

- Buscamos maximizar:

$$\sum_{\{u,v\}\in E} \mathrm{XOR}(x_u, x_v) \quad = \quad \sum_{\{u,v\}\in E} x_u + x_v - 2x_u x_v$$



QAOA



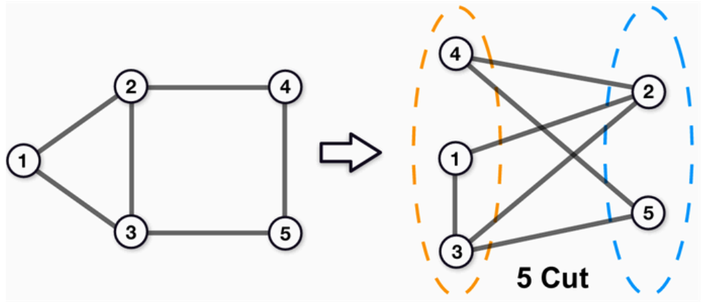
- Buscamos maximizar:

$$\sum_{\{u,v\}\in E} \mathrm{XOR}(x_u, x_v) \quad = \quad \sum_{\{u,v\}\in E} x_u + x_v - 2x_u x_v$$

- Interpretación física:

$$x_i \in \{0,1\} \xrightarrow[x_i = (z_i+1)/2]{} z_i \in \{-1,+1\}$$

QAOA



- Buscamos maximizar:

$$\sum_{\{u,v\}\in E} \mathrm{XOR}(x_u, x_v) \quad = \quad \sum_{\{u,v\}\in E} x_u + x_v - 2x_u x_v$$

- Interpretación física:

$$x_i \in \{0,1\} \xrightarrow[x_i = (z_i+1)/2]{} z_i \in \{-1,+1\}$$

- Nueva expresión a maximizar:

$$\sum_{\{u,v\}\in E} \frac{1}{2}(1 - z_u z_v)$$

QAOA

- Nueva expresión a maximizar:

$$\sum_{\{u,v\}\in E} \frac{1}{2}(1 - z_u z_v)$$

QAOA

- Nueva expresión a maximizar: $\sum_{\{u,v\}\in E} \frac{1}{2}(1 - z_u z_v)$

- Formulación de Hamiltoniano:

$$1 \longrightarrow Id = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$z_i \longrightarrow Z_i \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

QAOA

- Nueva expresión a maximizar: $\sum_{\{u,v\}\in E} \frac{1}{2}(1 - z_u z_v)$

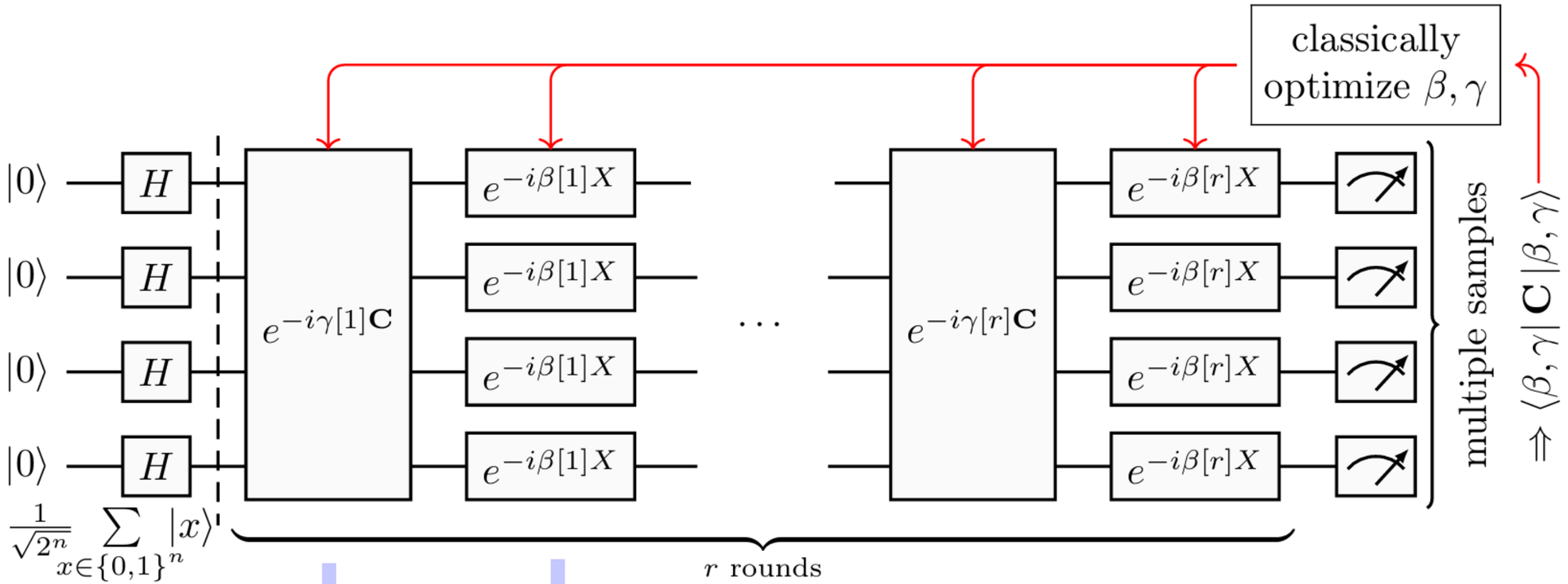
- Formulación de Hamiltoniano:

$$1 \longrightarrow Id = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

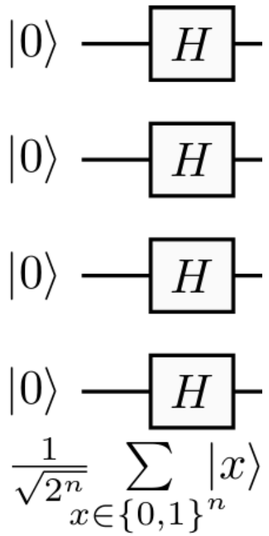
$$z_i \longrightarrow Z_i \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Hamiltoniano del problema

QAOA

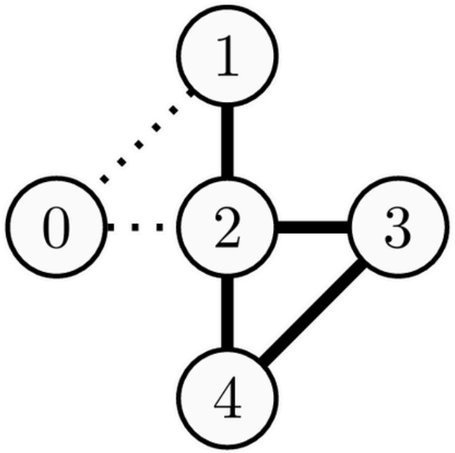

classically optimize β, γ
$|0\rangle$
H
$e^{-i\gamma[1]\mathbf{C}}$
$e^{-i\beta[1]X}$
$\cdots$
$e^{-i\gamma[r]\mathbf{C}}$
$e^{-i\beta[r]X}$
multiple samples
$\Rightarrow \langle\beta,\gamma|\,\mathbf{C}\,|\beta,\gamma\rangle$
$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$
r rounds
Hamiltoniano de mezcla
Hamiltoniano del problema

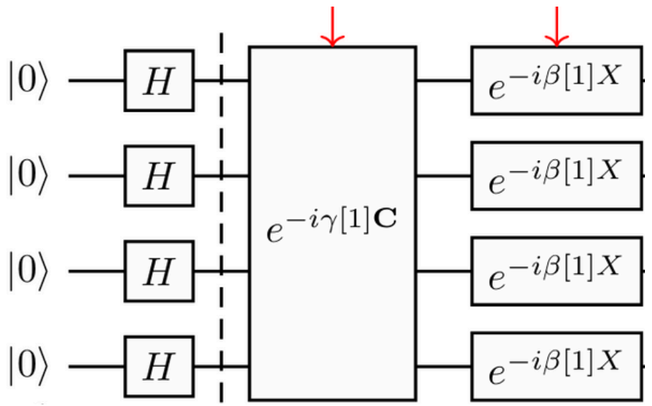
QAOA



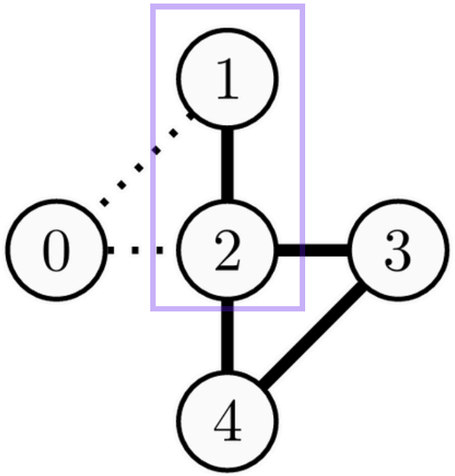

```python
import networkx
from qiskit import QuantumCircuit

def initH(circ, G):
    for q in G.nodes():
        circ.h(q)
```

QAOA

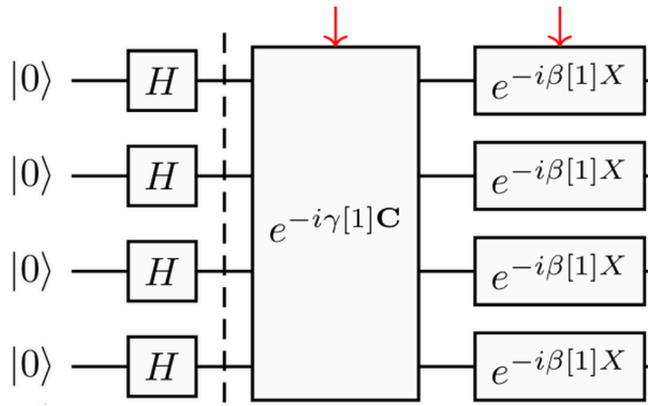




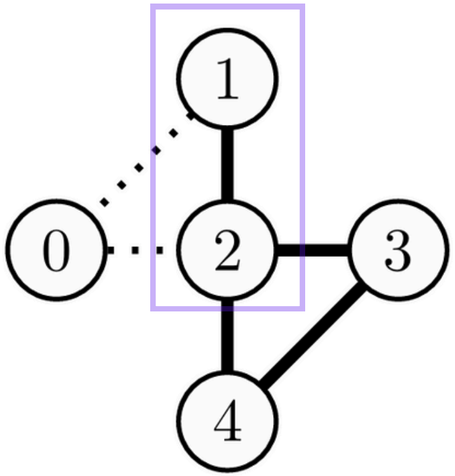
QAOA



$$e^{-i\frac{\gamma}{2}(Id-Z_1Z_2)} = e^{-i\gamma\begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix}}$$

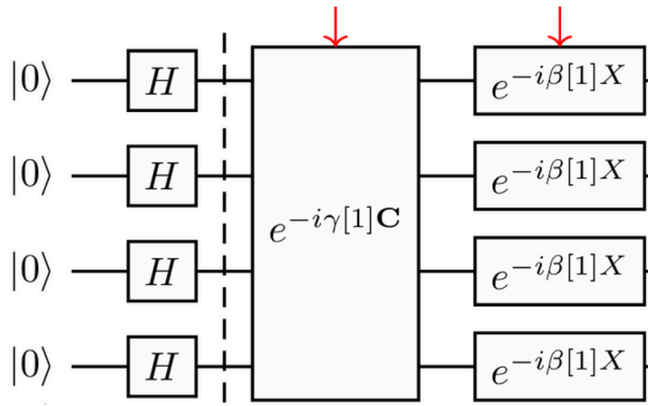


QAOA

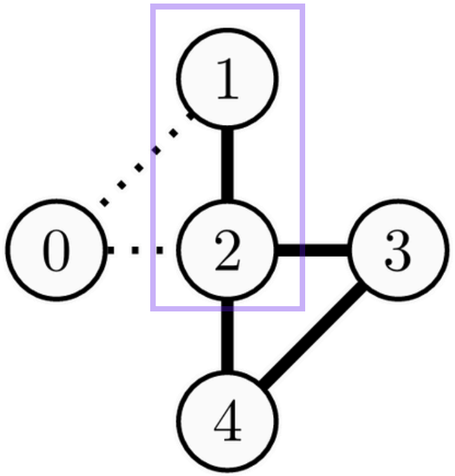


$$e^{-i\frac{\gamma}{2}(Id-Z_1Z_2)} = e^{-i\gamma\begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix}}$$

$$= \begin{pmatrix}1&0&0&0\\0&e^{-i\gamma}&0&0\\0&0&e^{-i\gamma}&0\\0&0&0&1\end{pmatrix}$$



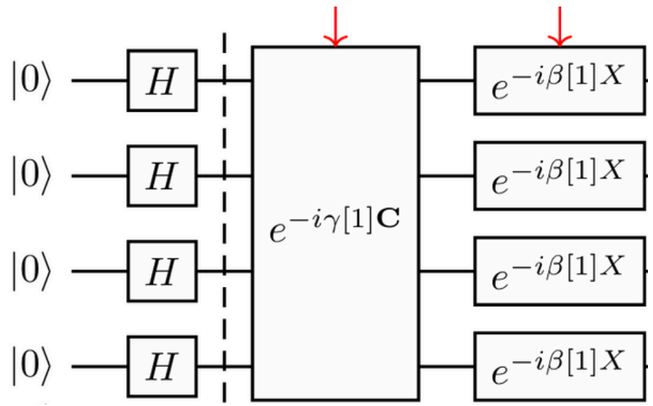
QAOA



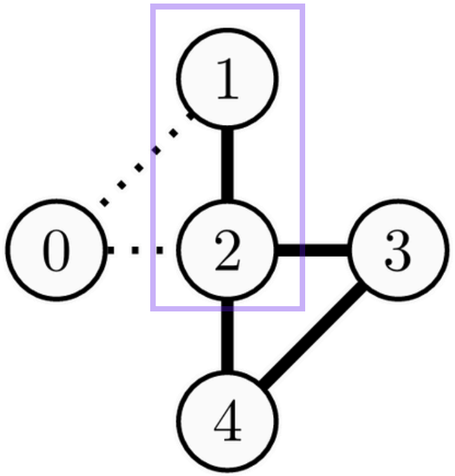
$$e^{-i\frac{\gamma}{2}(Id-Z_1Z_2)} = e^{-i\gamma\begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix}}$$

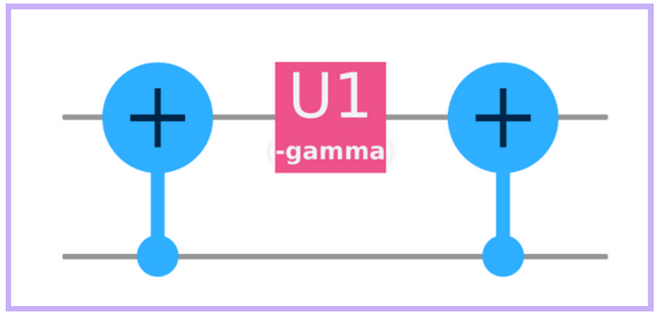
$$= \begin{pmatrix}1&0&0&0\\0&e^{-i\gamma}&0&0\\0&0&e^{-i\gamma}&0\\0&0&0&1\end{pmatrix}$$

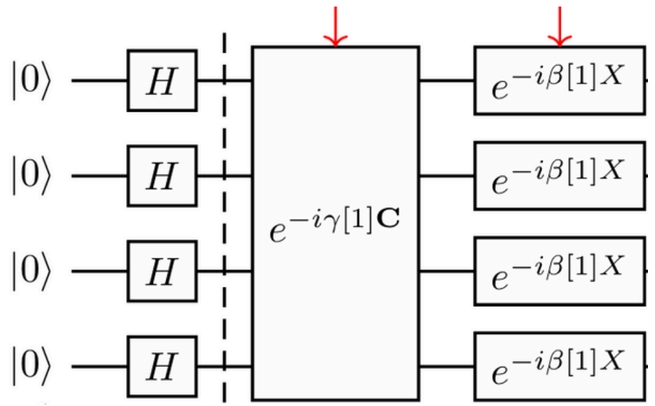
$$= \underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}}_{\mathrm{CNOT}_{12}} \cdot \underbrace{\begin{pmatrix}1&0&0&0\\0&e^{-i\gamma}&0&0\\0&0&1&0\\0&0&0&e^{-i\gamma}\end{pmatrix}}_{Id\otimes U_1(-\gamma)} \cdot \underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}}_{\mathrm{CNOT}_{12}}$$



QAOA

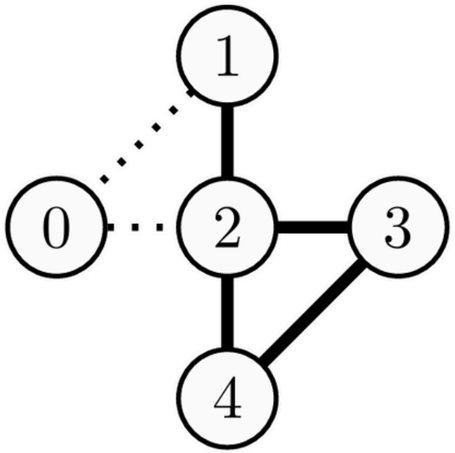


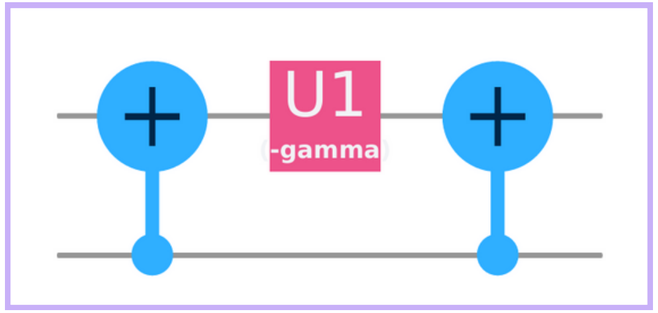




$$= \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathrm{CNOT}_{12}} \cdot \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-i\gamma} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-i\gamma} \end{pmatrix}}_{Id \otimes U_1(-\gamma)} \cdot \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathrm{CNOT}_{12}}$$

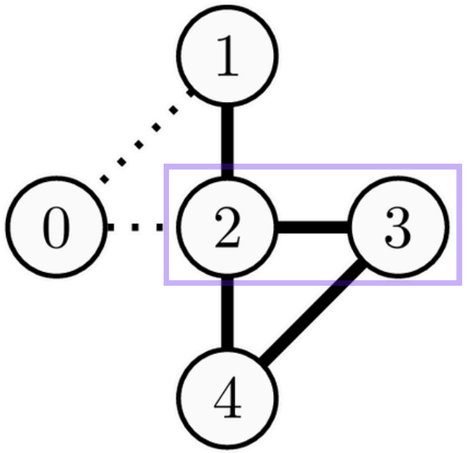
QAOA

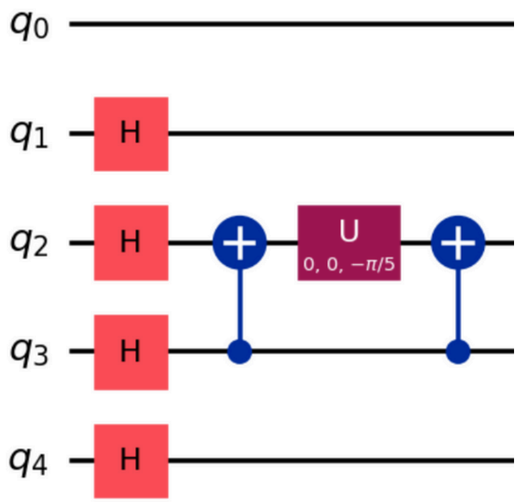





```
def phaseSep(circ, G, gamma):
    for q0, q1 in G.edges():
        circ.cx(q1, q0)
        circ.u(0,0,-gamma, q0)
        circ.cx(q1,q0)
```

QAOA

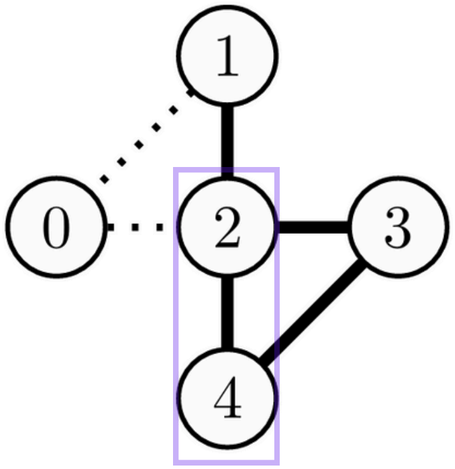


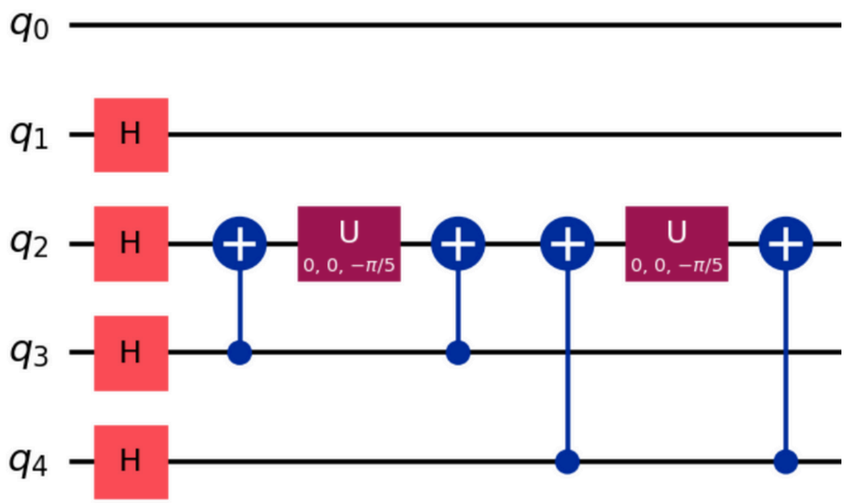



```python
def phaseSep(circ, G, gamma):
    for q0, q1 in G.edges():
        circ.cx(q1, q0)
        circ.u(0,0,-gamma, q0)
        circ.cx(q1,q0)
```

QAOA

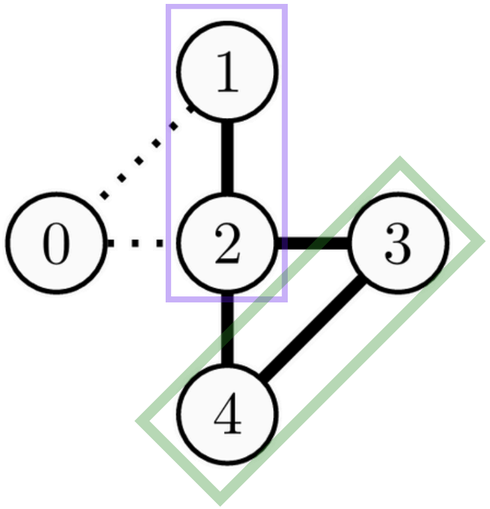
```
def phaseSep(circ, G, gamma):
    for q0, q1 in G.edges():
        circ.cx(q1, q0)
        circ.u(0,0,-gamma, q0)
        circ.cx(q1,q0)
```

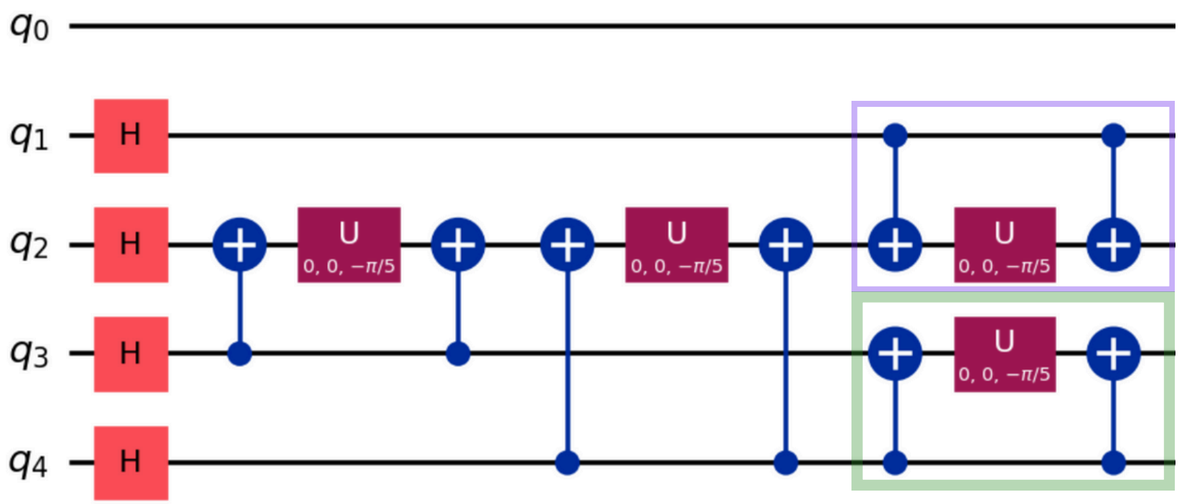




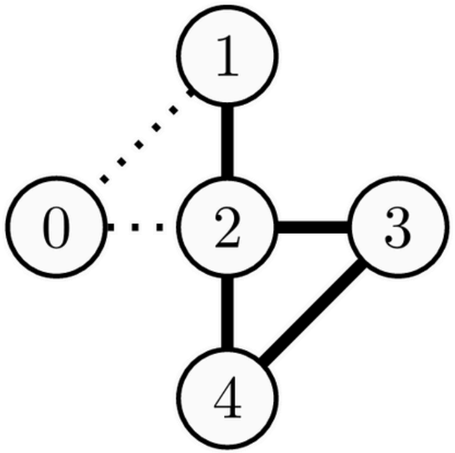
QAOA

```
def phaseSep(circ, G, gamma):
    for q0, q1 in G.edges():
        circ.cx(q1, q0)
        circ.u(0,0,-gamma, q0)
        circ.cx(q1,q0)
```

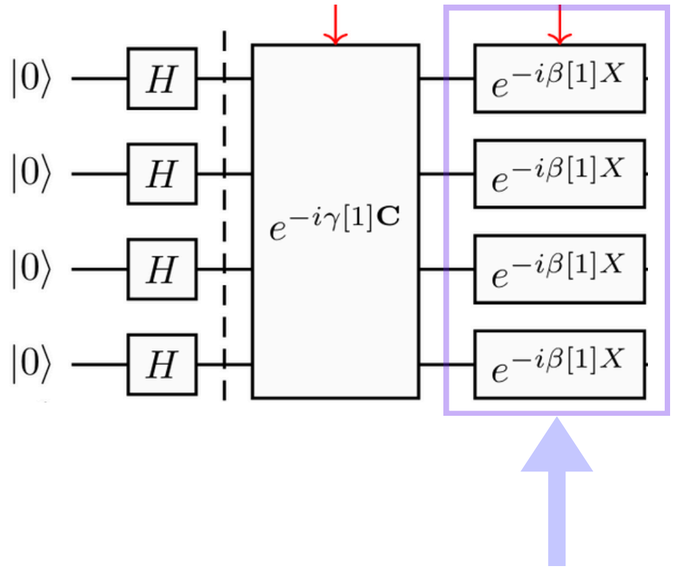




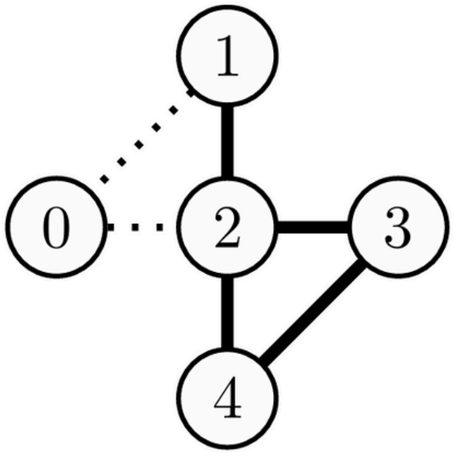
QAOA

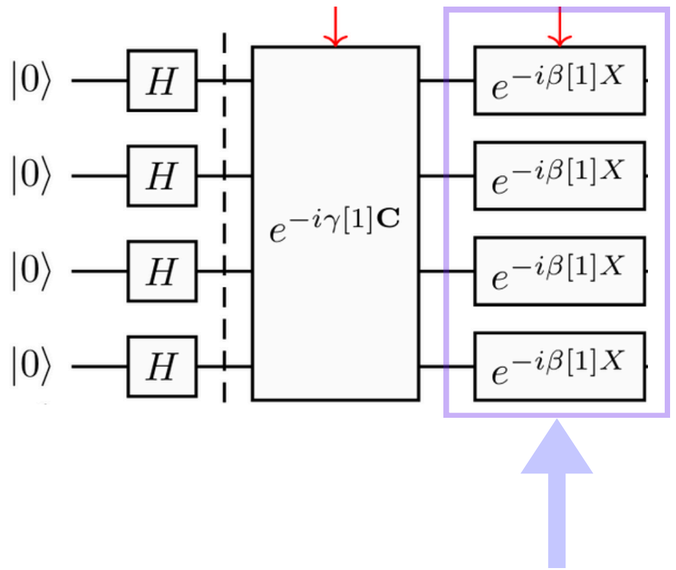


$$e^{-i\beta X} = e^{-i\beta \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}}$$



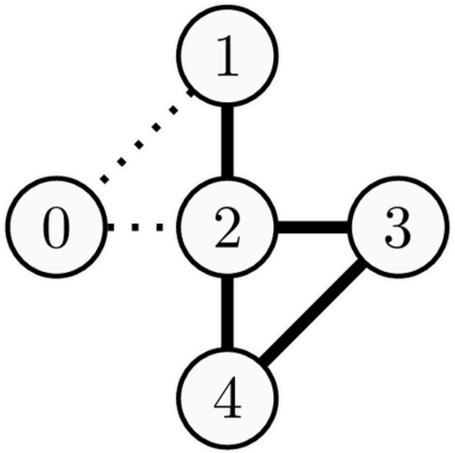
QAOA



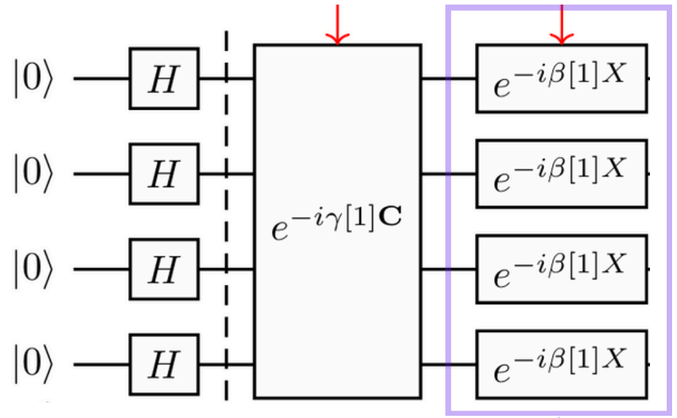


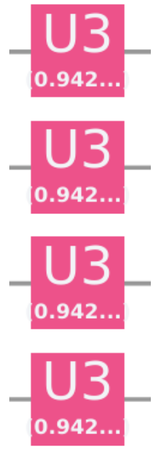
$$e^{-i\beta X} = e^{-i\beta\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}} = \begin{pmatrix}\cos(\beta) & -i\sin(\beta)\\ -i\sin(\beta) & \cos(\beta)\end{pmatrix}$$

QAOA

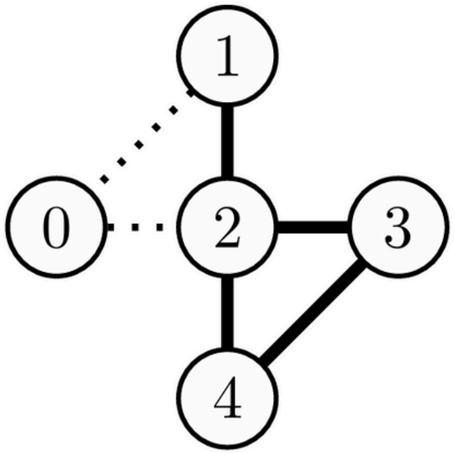


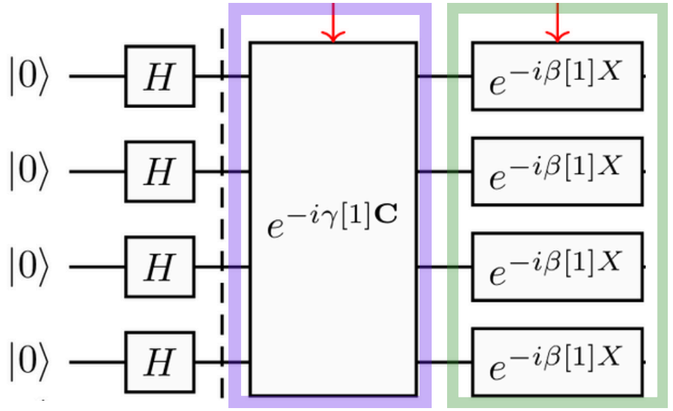

```
def mixer(circ, G, beta):
    for q in G.nodes():
        circ.u(2.0*beta, -numpy.pi/2.0, numpy.pi/2.0, q)
```

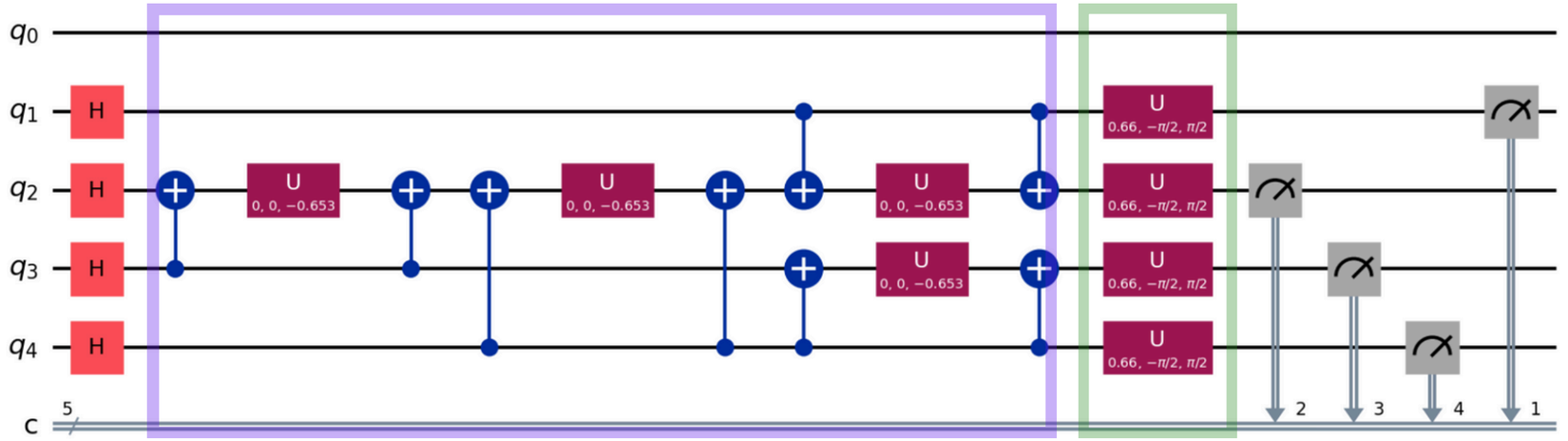




QAOA

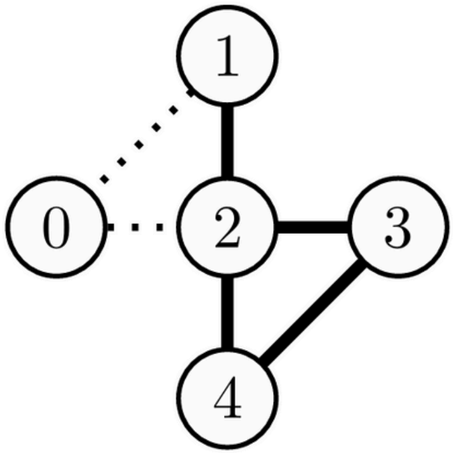






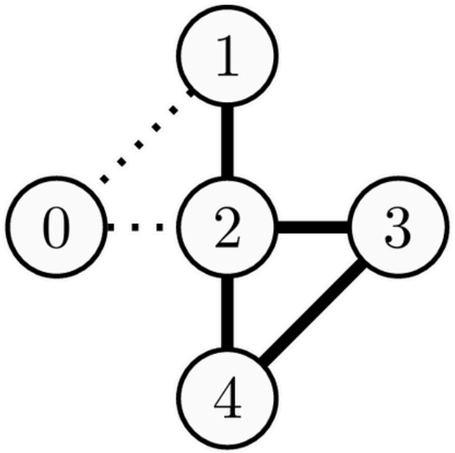

```
def measure(circ, G):
    for q in G.nodes():
        circ.measure(q, q)
```

QAOA

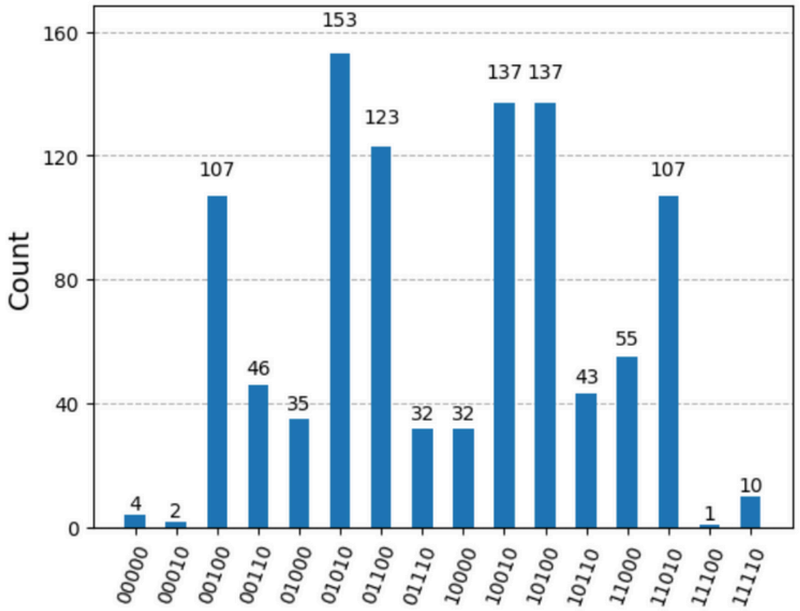



```python
import numpy
import networkx

## Grafos
Edge = networkx.Graph([(1,2)])
Triangle = networkx.Graph([(2,3),(2,4),(3,4)])
TplusE = networkx.compose_all([Triangle,Edge])
```

QAOA

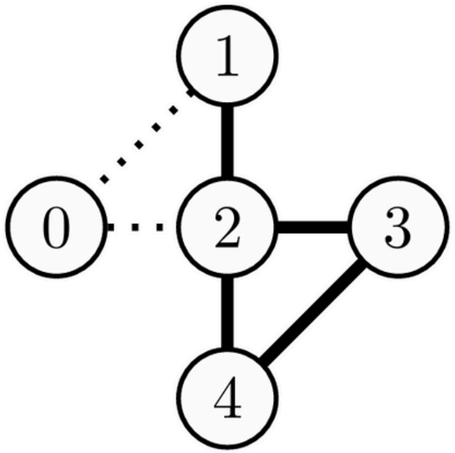



```
from qiskit_aer import Aer
from qiskit.visualization import plot_histogram

sim=Aer.get_backend('aer_simulator')
result=sim.run(circ_tpluse).result()
counts=result.get_counts()

plot_histogram(counts)
```

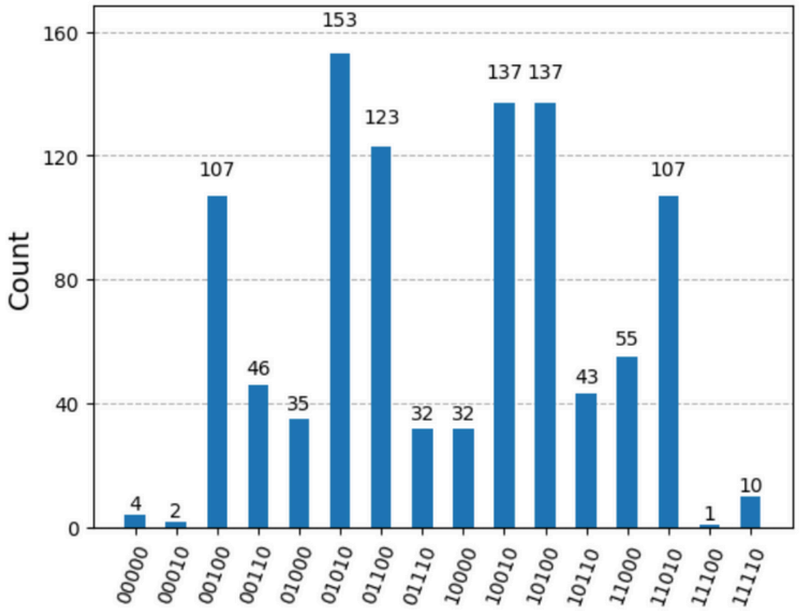


Una vuelta

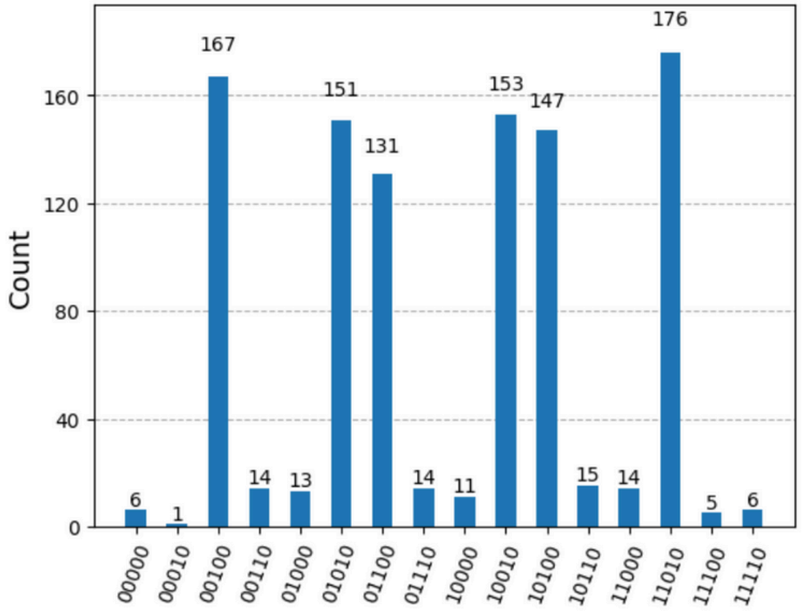
QAOA




```
from qiskit_aer import Aer
from qiskit.visualization import plot_histogram

sim=Aer.get_backend('aer_simulator')
result=sim.run(circ_tpluse).result()
counts=result.get_counts()

plot_histogram(counts)
```

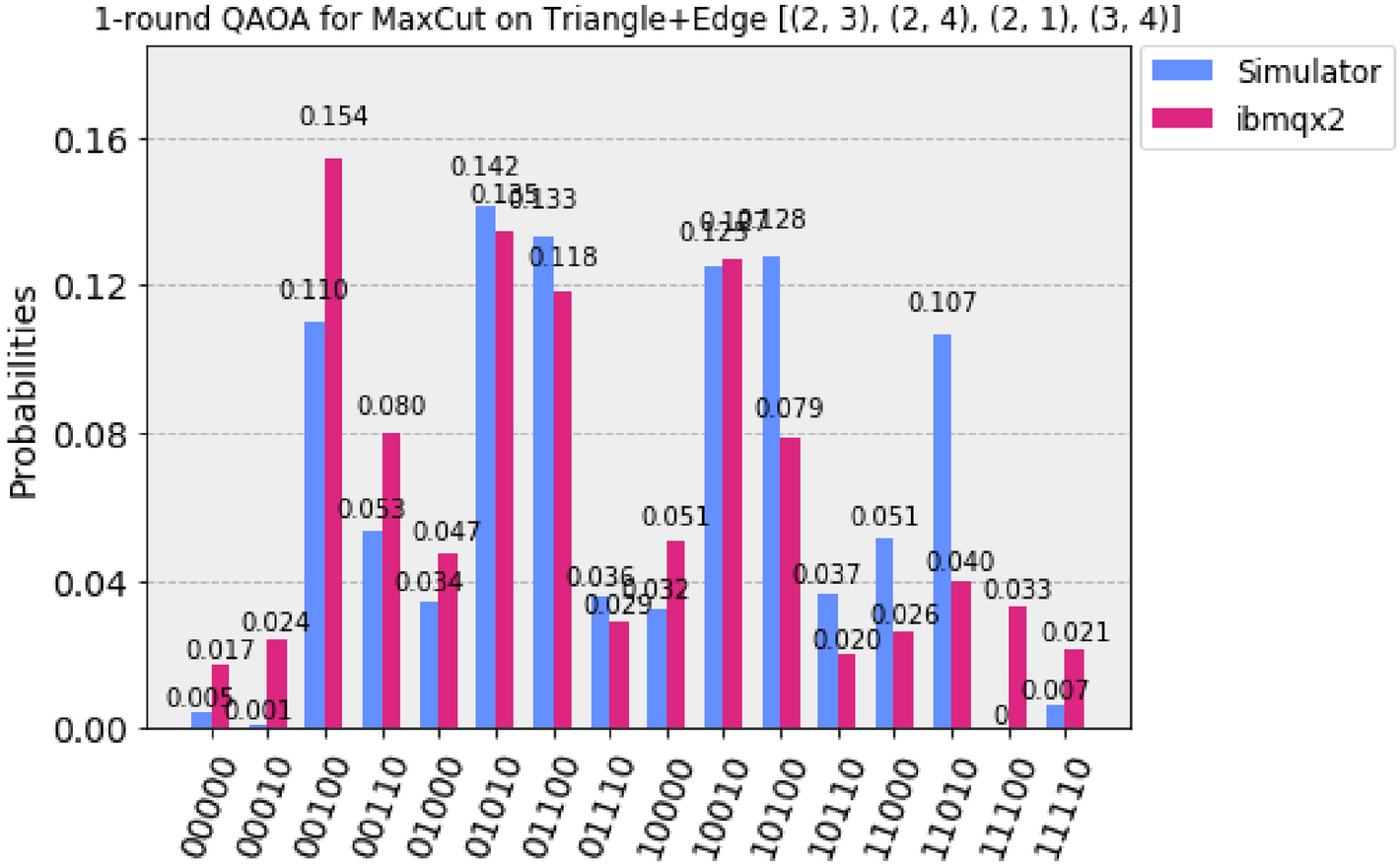


Una vuelta

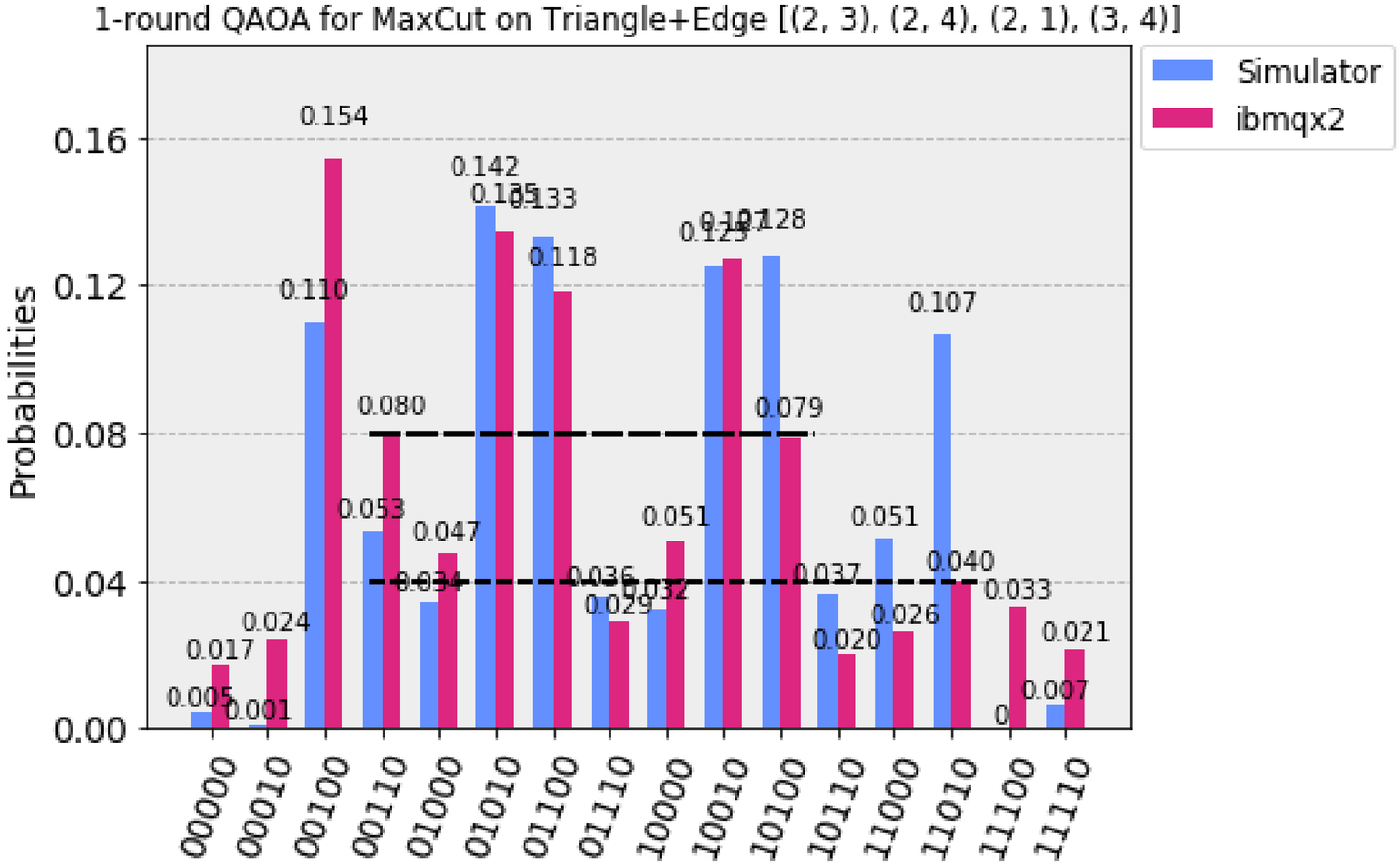


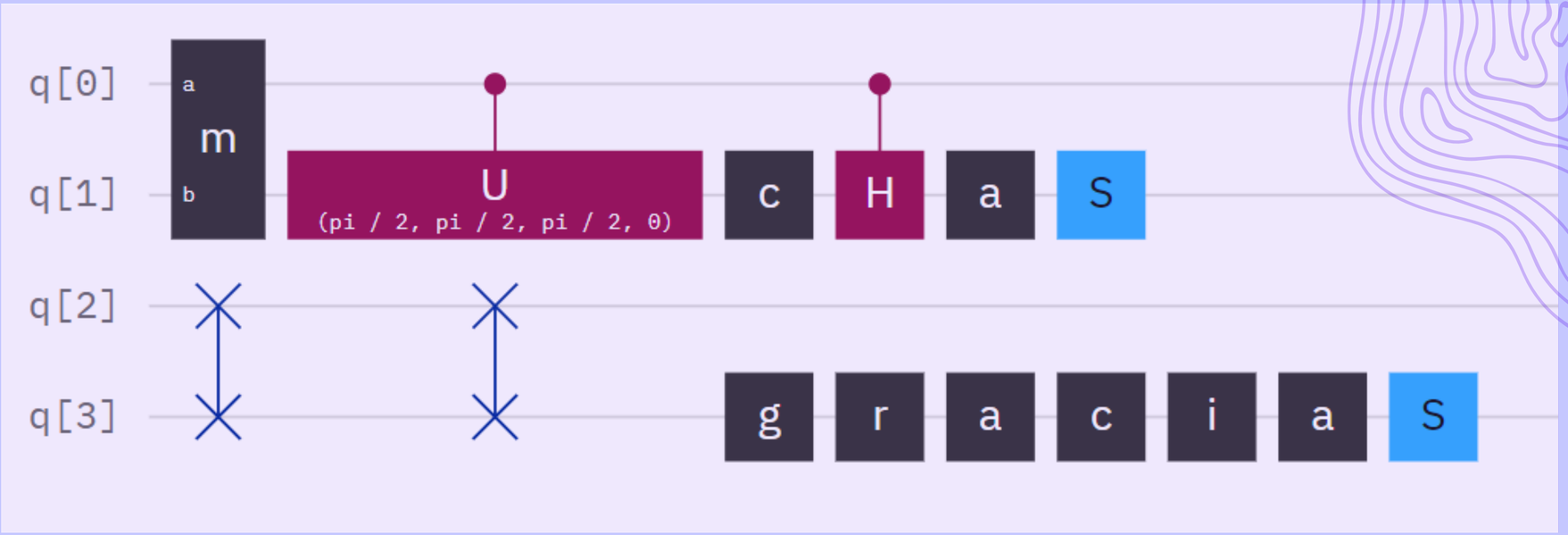
Dos vueltas

QAOA


1-round QAOA for MaxCut on Triangle+Edge [(2, 3), (2, 4), (2, 1), (3, 4)]
Simulator
ibmqx2
Probabilities
0.00
0.04
0.08
0.12
0.16
00000
00010
00100
00110
01000
01010
01100
01110
10000
10010
10100
10110
11000
11010
11100
11110

QAOA


1-round QAOA for MaxCut on Triangle+Edge [(2, 3), (2, 4), (2, 1), (3, 4)]
Simulator
ibmqx2
Probabilities
0.00
0.04
0.08
0.12
0.16
00000
00010
00100
00110
01000
01010
01100
01110
10000
10010
10100
10110
11000
11010
11100
11110



REFERENCIAS PARA PROFUNDIZAR: