IBM Q

Qiskit-hackathon-Taiwan

Quantum simulation of one-dimensional system on IBMQ device

Members

李泰岳

Anandu Kalleri Madhu

張育慈

楊英正

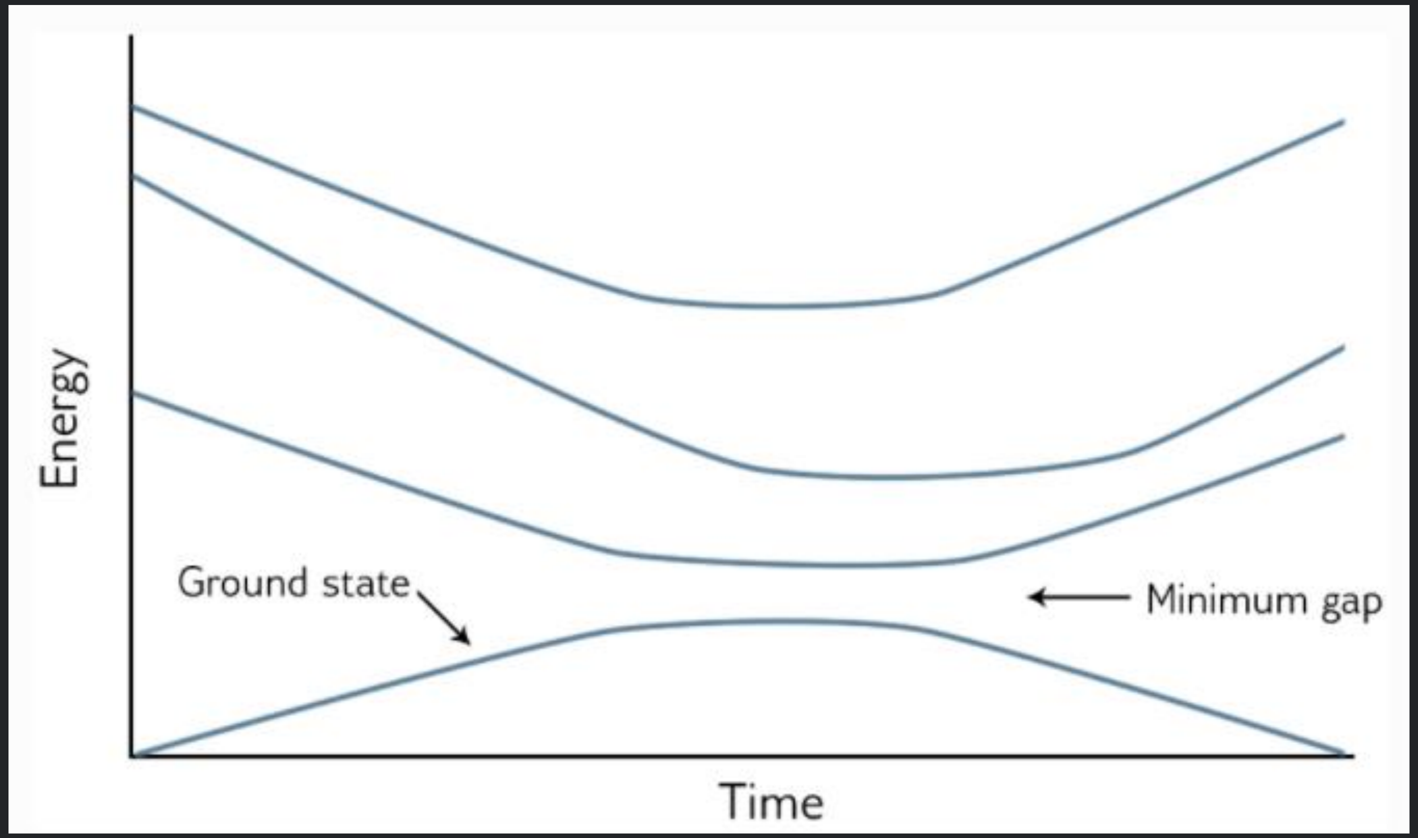
Coach

卓建宏

How to connect the problems and Quantum computer?

The Hamiltonian of Ising model:

$$H = \frac{-A(s)}{2}\left(\sum_i \sigma_x^{(i)}\right) + \frac{1-A(s)}{2}\left(\sum_i h_i \sigma_z^{(i)} + \sum_{i>j} J_{i,j}\sigma_z^{(i)}\sigma_z^{(j)}\right)$$



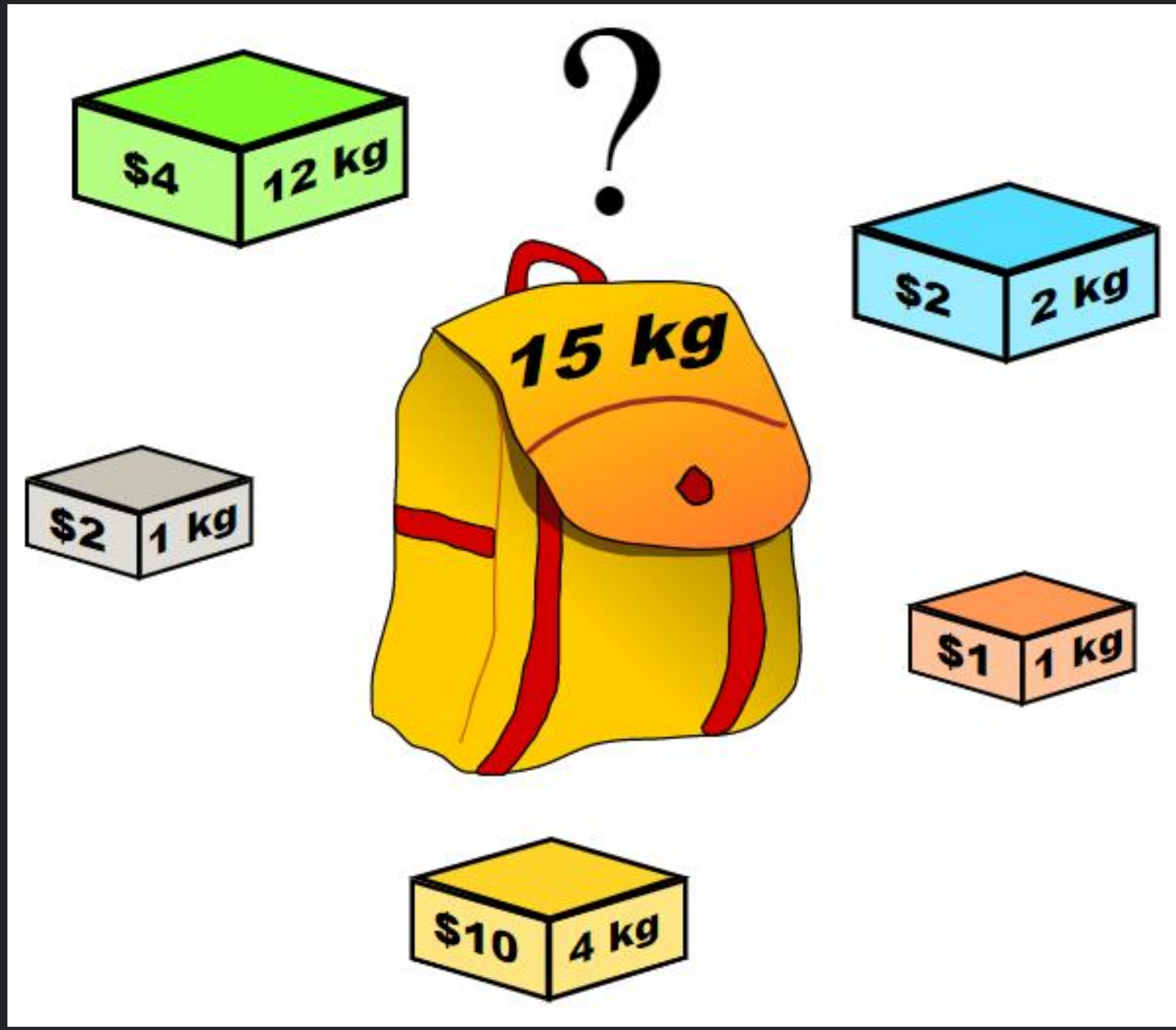
Pauli matrices operating on a qubit x_i

h_i **and interacting parameter** $J_{i,j}$ **are the qubit biases and coupling strengths**

https://docs.dwavesys.com/docs/latest/c_gs_2.html?fbclid=IwAR1zq5W45Lr0SHCGS5KrKcNZBoSBCKcTpBqrd2-aGaNUW-D_LhZ56NQDLPc

There's an interesting topic!

Knapsack problem



Item	1	2	3	4	5
Value	10	6	11	4	5
Weight	3	2	4	2	3

Parameters:
- **Values**
- **Weights**
- **maximum weight**

maximize (Knapsack problem) $\sum_{i=1} V_i x_i$ ⟷ **subject to** $\sum_{i=1} W_i x_i \leq C$ $\quad x_i$ is 0 or 1, $i = 1,2,3,\cdots$

https://en.wikipedia.org/wiki/Knapsack_problem

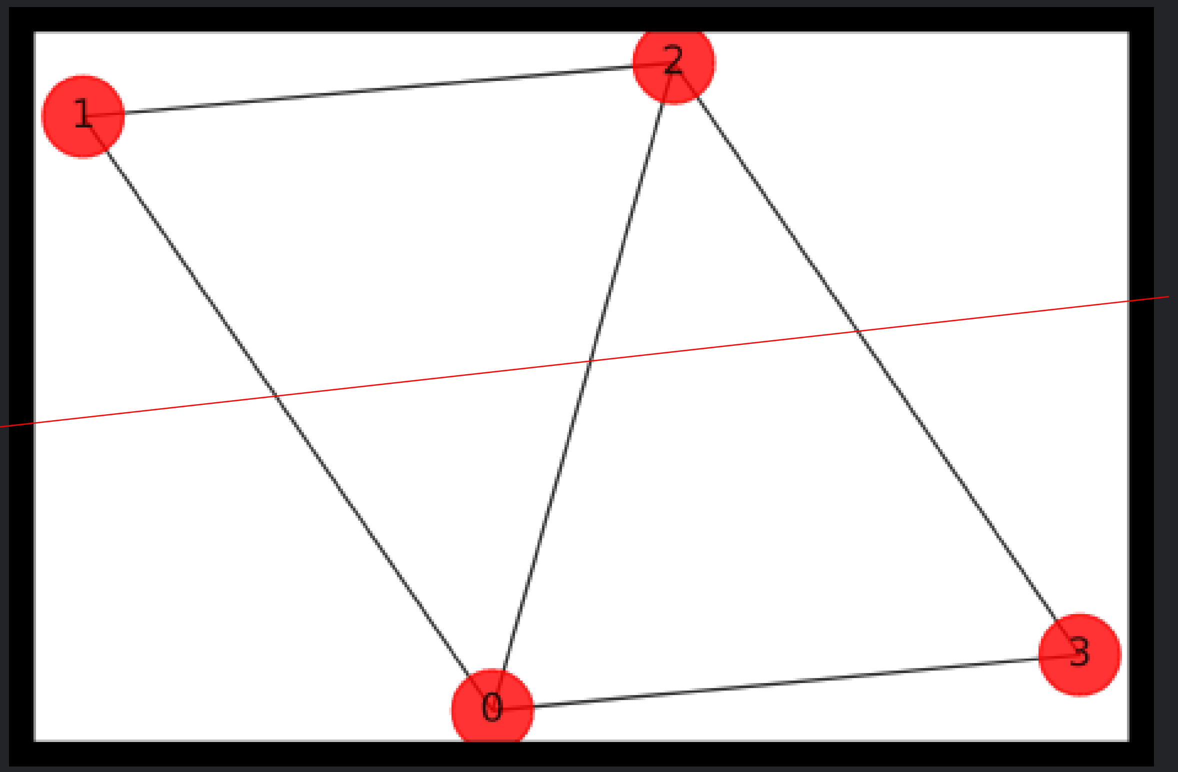
Max-Cut problem

Model :

$$C(x)= \sum_{i,j} w_{ij} x_i \left(1 - x_j\right) + \sum_i w_i x_i$$

$x_i \to (1 - Z_i)/2$, **where Z_i is the Pauli Z operator,** $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

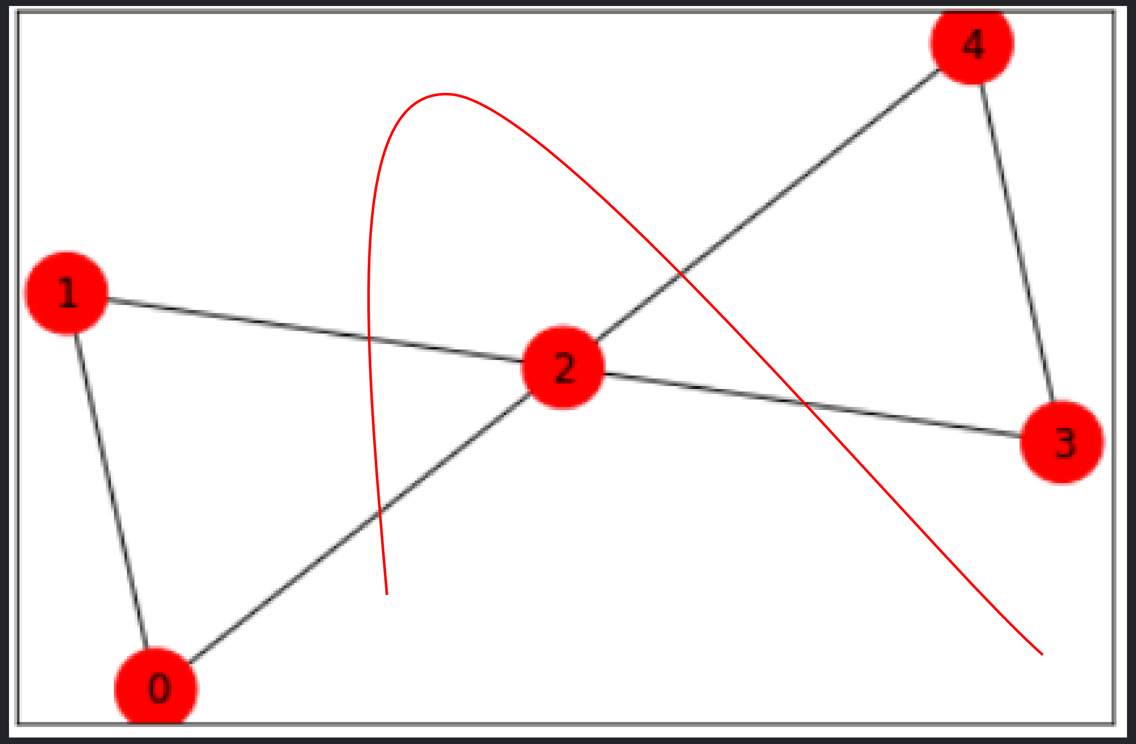
Cut line



But we can solve it only if we mapping this model to the Ising model :

$$H = \sum_i w_i Z_i + \sum_{i<j} w_{ij} Z_i Z_j$$

After mapping, we use quantum adiabatic algorithm to do optimization.



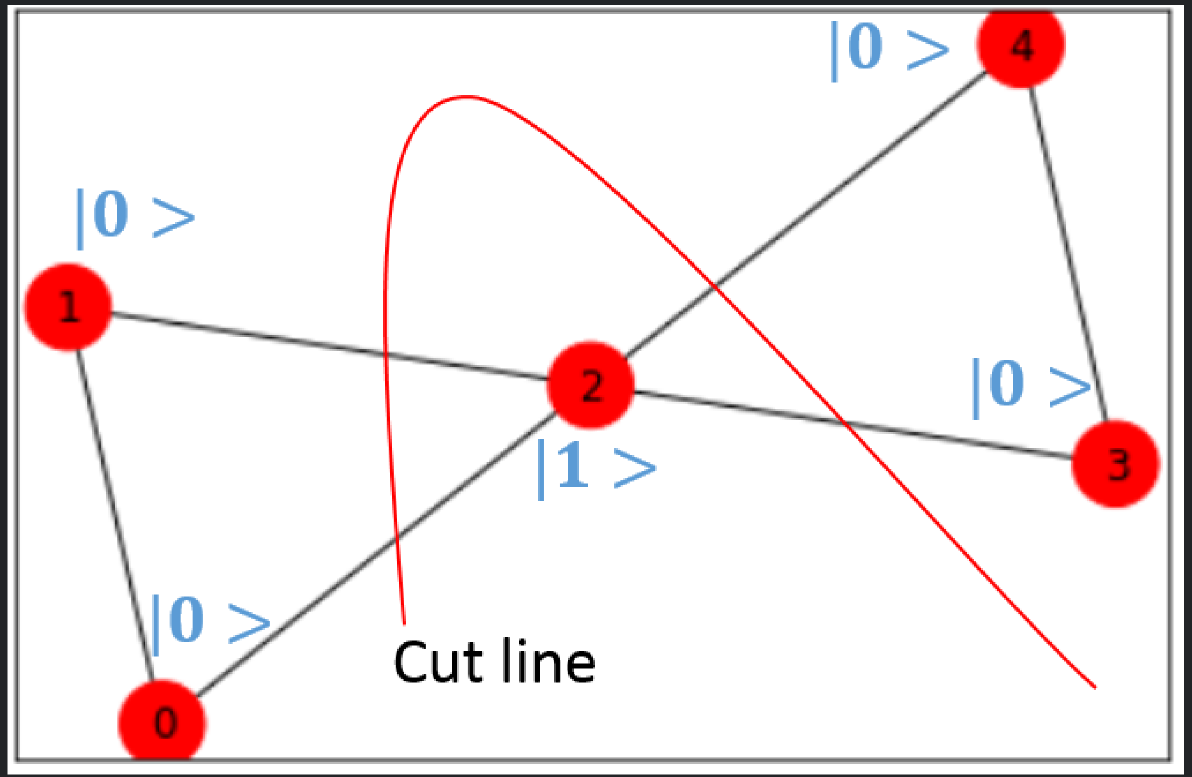
What we want to solve, with n=5.

9 Process

01

Coding process: Set up qubit dots connection

```
w=np.zeros([n,n])
for i in range(n):
    for j in range(n):
        temp=G.get_edge_data(i,j,default=0)
        if temp != 0:
            w[i,j]=temp['weight']
print(w)
```




```
[[0. 1. 1. 0. 0.]
 [1. 0. 1. 0. 0.]
 [1. 1. 0. 1. 1.]
 [0. 0. 1. 0. 1.]
 [0. 0. 1. 1. 0.]]
```

9 Process

02

Coding process: Transform into Ising model

```python
from qiskit.optimization.applications.ising import max_cut
qubitOp, offset = max_cut.get_operator(w)
print('Offset:', offset)
print('Ising Hamiltonian:')
print(qubitOp.print_details())
```

$$H = \sum_i w_i Z_i + \sum_{i<j} w_{ij} Z_i Z_j.$$

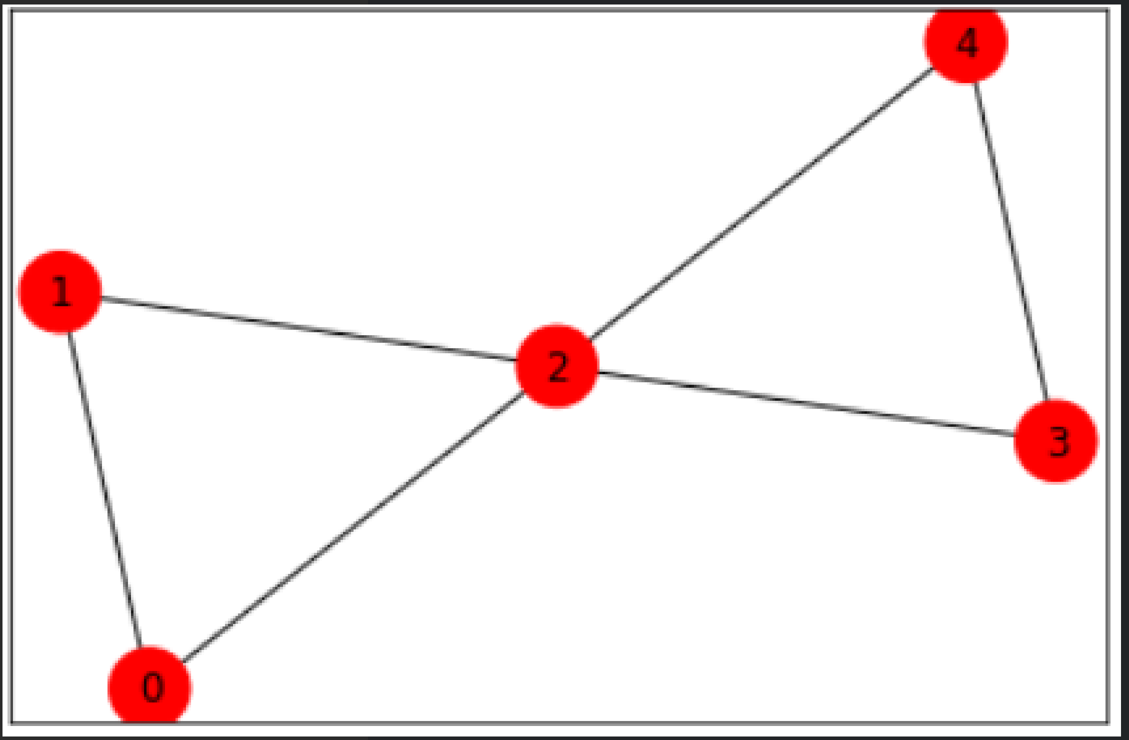
```
Ising Hamiltonian:
IIIZZ    (0.5+0j)
IIZIZ    (0.5+0j)
IIZZI    (0.5+0j)
IZZII    (0.5+0j)
ZIZII    (0.5+0j)
ZZIII    (0.5+0j)
```

9 Process

03

Coding process: Set up 5-qubit dots

```
# Generating the butterfly graph with 5 nodes
n      = 5
V      = np.arange(0,n,1)
E      =[(0,1,1.0),(0,2,1.0),(1,2,1.0),(3,2,1.0),(3,4,1.0),(4,2,1.0)]

G      = nx.Graph()
G.add_nodes_from(V)
G.add_weighted_edges_from(E)

# Generate plot of the Graph
colors       = ['r' for node in G.nodes()]
default_axes = plt.axes(frameon=True)
pos          = nx.spring_layout(G)

nx.draw_networkx(G, node_color=colors, node_size=600, alpha=1, ax=default_axes, pos=pos)
```



9 Process

Hamiltonian in Quantum Annealing

$$H = \underbrace{\boxed{\frac{-A(s)}{2}\left(\sum_i \sigma_x^{(i)}\right)}}_{\text{H0, initial state}} + \underbrace{\boxed{\frac{1-A(s)}{2}\left(\sum_i h_i \sigma_z^{(i)} + \sum_{i>j} J_{i,j}\sigma_z^{(i)}\sigma_z^{(j)}\right)}}_{\text{Hp, final state}}$$

- Use 5 qubits with initial state evolving to final state
- Final state is from the Max-cut problems

9 Process

05

Coding process: HamiltonianGate → Set up Hamiltonian

```
H0 = -A(s)/2*(x.tensor(i).tensor(i).tensor(i).tensor(i)+
            i.tensor(x).tensor(i).tensor(i).tensor(i)+
            i.tensor(i).tensor(x).tensor(i).tensor(i)+
            i.tensor(i).tensor(i).tensor(x).tensor(i)+
            i.tensor(i).tensor(i).tensor(i).tensor(x))
H_p = (1-A(s))/2*(qubitOp.to_opflow())
circ.append(HamiltonianGate(H0, 1), qr)
circ.append(HamiltonianGate(H_p, 1), qr)
```

9 Process

06

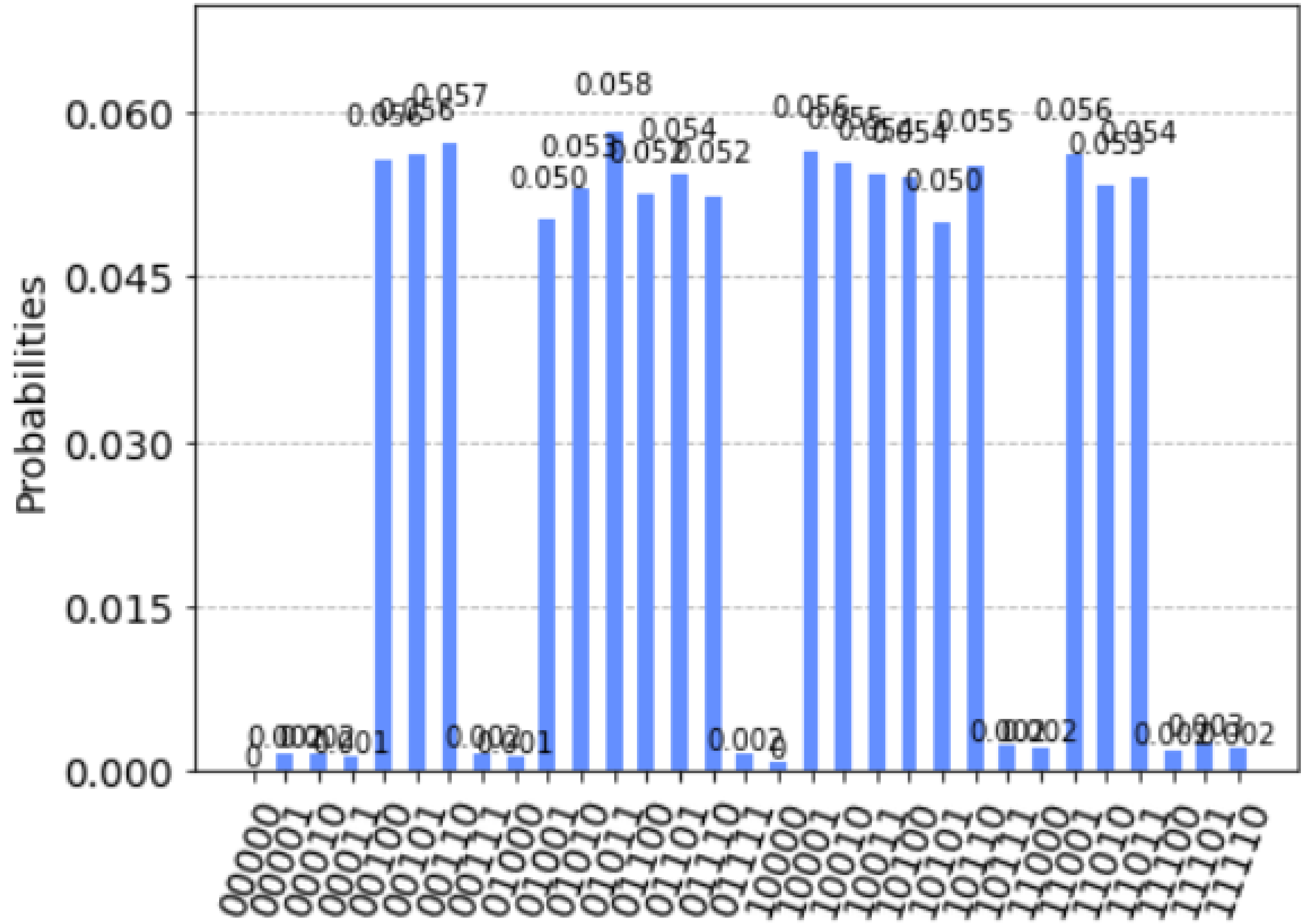
Coding process: Measuring backend with simulator

```
sim_backend = provider.get_backend('ibmq_qasm_simulator')
```

```
s = round((s-0.01),6)
circ.measure(qr,cr)
job = execute(circ,backend = sim_backend, shots = 8192 )
output = job.result()
answer = output.get_counts(circ)
```

Result

{'00000': 1, '00001': 14, '10000': 7, '10001': 462, '10010': 453, '10011': 446, '10100': 444, '10101': 410, '10110': 451, '10111': 19, '11000': 18, '11001': 459, '11010': 436, '11011': 444, '11100': 16, '11101': 22, '11110': 17, '00010': 14, '00011': 11, '00100': 455, '00101': 459, '00110': 469, '00111': 14, '01000': 11, '01001': 412, '01010': 434, '01011': 477, '01100': 430, '01101': 445, '01110': 429, '01111': 13}



9 Process

07

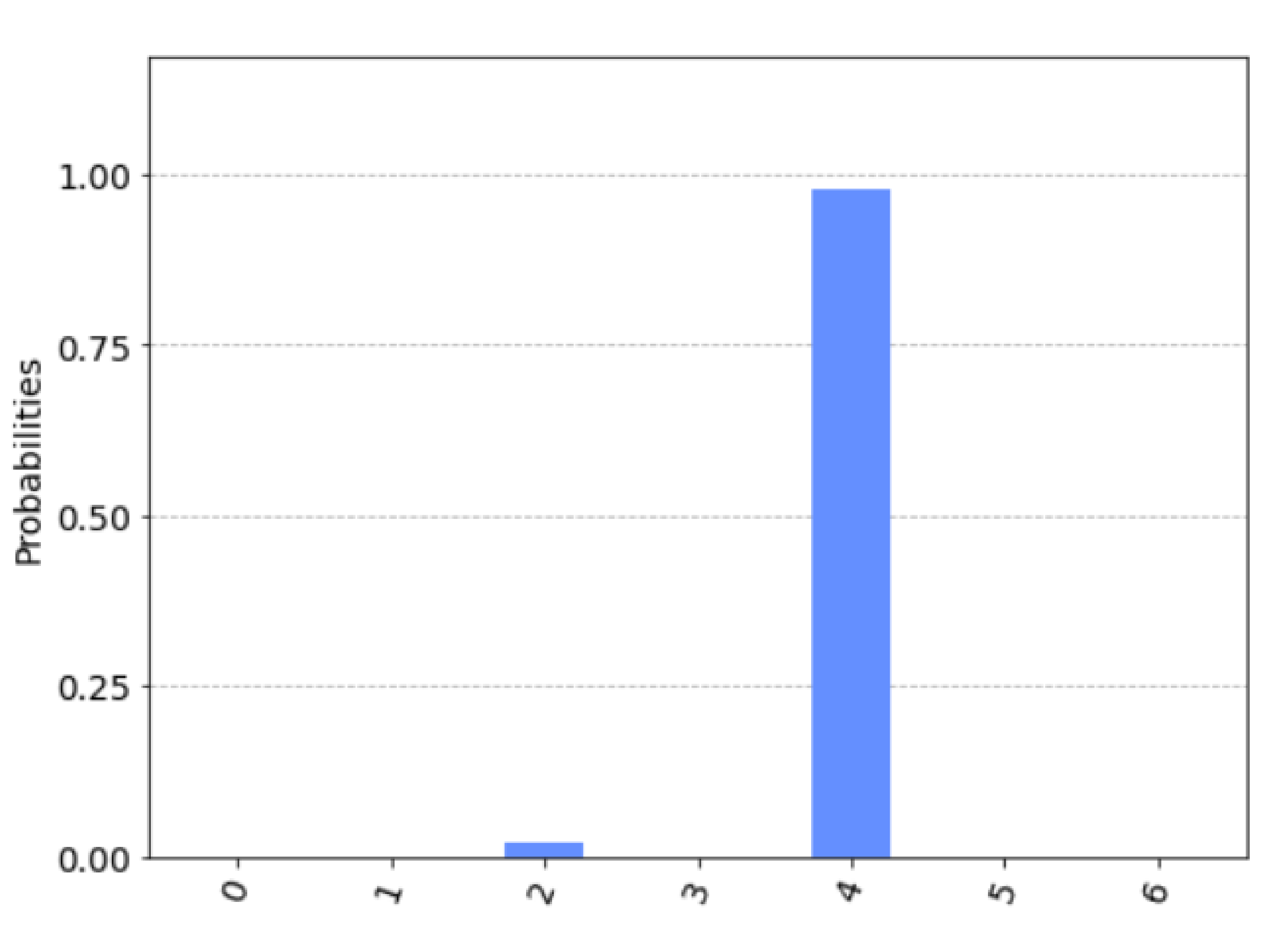
Coding process: Counts for eigenvalue which means ‘cut’

```
p3=[]
n=5
best_cost_brute = 0
for b in range(2**n):
    x = [int(t) for t in reversed(list(bin(b)[2:].zfill(n)))]
    cost = 0
    for i in range(n):
        for j in range(n):
            cost = cost + w[i,j]*x[i]*(1-x[j])

    if best_cost_brute < cost:
        best_cost_brute = cost
        xbest_brute = x
    p3.append(best_cost_brute)
    print('case = ' + str(x)+ ' cost = ' + str(cost))

colors = ['r' if xbest_brute[i] == 0 else 'b' for i in range(n)]
nx.draw_networkx(G, node_color=colors, node_size=600, alpha=.8, pos=pos)
print('\nBest solution = ' + str(xbest_brute) + ' cost = ' + str(best_cost_brute))
```

9 Process

08

Coding process: Calculate the eigenvalue value vary with steps



9 Process

09

Coding process: Calculate the expectation value vary with steps

```
[111.11111111111111, 125.0, 142.85714285714286, 166.66666666666666, 200.0, 250.0, 333.3333333333333, 500.
0, 1000.0] [3.68701171875, 3.689697265625, 3.685546875, 3.6865234375, 3.704345703125, 3.686767578125, 3.69
0673828125, 3.692138671875, 3.685302734375]
```

The expectation value is near the maximum value 4 !

Reference

- Lucas, Andrew. "Ising formulations of many NP problems." *Frontiers in Physics* 2 (2014): 5.
- Coffey, Mark W. "Adiabatic quantum computing solution of the knapsack problem." *arXiv preprint arXiv:1701.05584* (2017).
- https://qiskit.org/documentation/stubs/qiskit.optimization.applications.ising.knapsack.html?fbclid=IwAR2GWY01Qjtzl-ai9NhudxMN42q_Zzq8d9510a9IRBgBy0Zb80gxFo_uL8M
- Murawski, Carsten, and Peter Bossaerts. "How humans solve complex problems: The case of the Knapsack problem." *Scientific reports* 6 (2016): 34851.

THANK YOU FOR LISTENING!

ANY QUESTIONS?