Ciência de Dados Quântica 2021/22

QAOA:

Quantum **A**pproximate **O**ptimization **A**lgorithm

LUÍS PAULO SANTOS

Material de Consulta

- [Schuld2021] – Sec. 3.6.5.2; Chap. 5
- Qiskit Textbook:
  Solving combinatorial optimization problems using QAOA
  https://qiskit.org/textbook/ch-applications/qaoa.html
- Quantum Approximate Optimization Algorithm explained
  https://www.mustythoughts.com/quantum-approximate-optimization-algorithm-explained
- Quantum TSP tutorial
  https://github.com/mstechly/quantum_tsp_tutorials

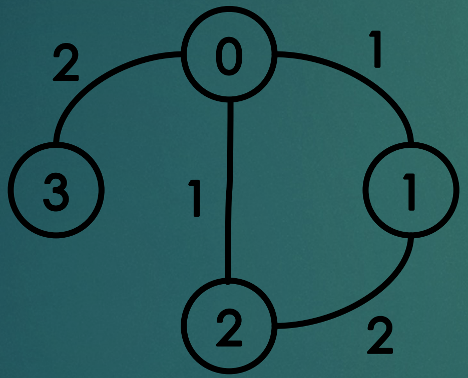
Combinatorial Optimization

- Combinatorial optimization problems:
  finding an optimal object out of a finite set of objects.

- Our formulation:
  finding optimal bit strings, $z = \{0,1\}^{\otimes n}$, out of a set of finite bitstrings

Combinatorial Optimization: MaxCut

- Given a graph G=(V,E) find the cut with maximum cost
  A cut is a partitioning of the graph's nodes into 2 sets



$$A = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

$z = z_0 z_1 z_2 z_3$	Cut
0000	0
0001	2
0010	3
0011	5
0100	3
0101	5
0110	2
0111	4
1000	4
1001	2
1010	5
1011	3
1100	5
1101	3
1110	2
1111	0

Combinatorial Optimization: MaxSAT

- Given Boolean statements on the bits z_i identify the bit strings which satisfy more statements

$z = z_0 z_1 z_2$ $\qquad C_1(z) = z_0 z_1$ $\qquad C_2(z) = z_0 \bar{z}_2$

$$C(z) = \sum_{z=1}^{K} C_k(z)$$

$z=z_0z_1z_2$	$C_1(z)$	$C_2(z)$	$C(z)$
000	0	0	0
001	0	0	0
010	0	0	0
011	0	0	0
100	0	1	1
101	0	0	0
110	1	1	2
111	1	0	1

Time Evolution

- Let H be the Hamiltonian of a quantum system. Then the system time evolution, according to Schrodinger equation is:

$$i\hbar \frac{d}{dt}|\psi(t)\rangle = H\ |\psi(t)\rangle$$

$$|\psi(t)\rangle = e^{\frac{-iHt}{\hbar}}|\psi(0)\rangle$$

- The Hamiltonian is Hermitian $(H = H^{\dagger})$,

therefore $U_H(t) = e^{\frac{-iHt}{\hbar}}$ is unitary $(U_H^{-1}(t) = U_H^{\dagger}(t))$

$$|\psi(t)\rangle = e^{\frac{-iHt}{\hbar}}|\psi(0)\rangle = U_H(t)|\psi(0)\rangle$$

Adiabatic Quantum Computing

"A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough ..." [1]

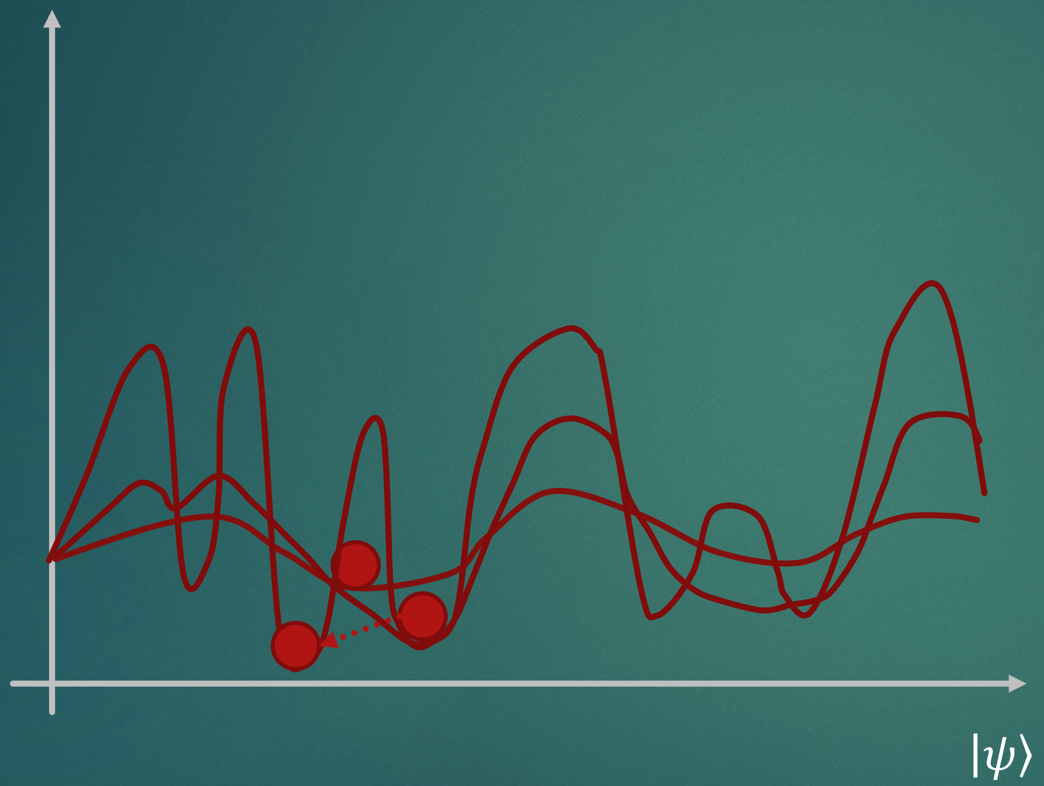
- Start from the **ground state** $|s\rangle$ of a quantum system with an **easily realisable Hamiltonian** $\boldsymbol{H_0}$, which corresponds to a **smooth landscape**
- Aim towards the **ground state of the Hamiltonian** $\boldsymbol{H_p}$, which describes the quantum system of interest
- Then the **Hamiltonian** used by the **adiabatic approach** is:

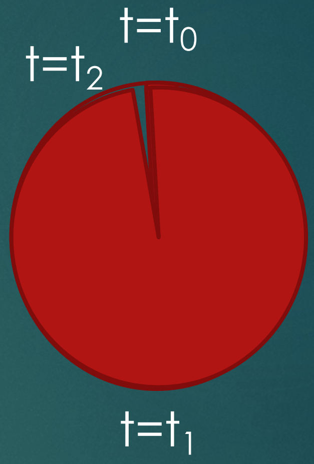
$$H(t) = (1 - t)\, H_0 + t\, H_p$$

---

[1] - *M. Born and V. A. Fock (1928). "Beweis des Adiabatensatzes". Zeitschrift für Physik A.* ***51*** *(3–4): 165–180. doi:10.1007/BF01343193*

Adiabatic Quantum Computing





Slow time evolution from the ground state of H_0 to the ground state of H_p

QAOA operator

- Up to now:
  - $|\boldsymbol{\psi(t)}\rangle = \boldsymbol{e}^{\frac{-iHt}{\hbar}}|\boldsymbol{\psi(0)}\rangle$
  - $\boldsymbol{H(t) = (1-t)\,H_0 + t\,H_p}$
- The circuit based QAOA substitutes $(1-t)$ and t by parameters β and α, which are variationally optimized:
  - $|\boldsymbol{\psi(t)}\rangle = \boldsymbol{e}^{\frac{-i\beta H_0}{\hbar}+\frac{-i\alpha H_p}{\hbar}}|\boldsymbol{\psi(0)}\rangle$

QAOA operator: trotterization

- if $\boldsymbol{H_0}$ and $\boldsymbol{H_p}$ do not commute (and they can't commute for QAOA to converge[2]), then:

$$\boldsymbol{e^{(-i\beta H_0)+(-i\propto H_p)} \neq e^{-i\beta H_0} * e^{-i\propto H_p}}$$

- Lie product formula states that: $e^{A+B} = \lim_{p\to\infty}\left(e^{A/p} * e^{B/p}\right)^p$

- The Trotter Suzuki formula allows truncation (trotterization) of the above product:

$$e^{A+B} = \left(e^{A/p} * e^{B/p}\right)^p + \mathcal{O}(p^{-1}), \text{ for finite } p \in \mathbb{N}$$

---

[2] – for an explanation of the implications of commutation see: https://physics.stackexchange.com/questions/9194/what-is-the-physical-meaning-of-commutation-of-two-operators

QAOA operator: trotterization

$$\boldsymbol{e}^{(-\boldsymbol{i\beta H_0})+(-\boldsymbol{iH_p})} = \left(e^{-\boldsymbol{i\beta H_0}/p} * e^{-i\propto \boldsymbol{H_p}/p}\right)^p + \mathcal{O}(p^{-1}),\ \text{for finite } p \in \mathbb{N}$$

- Let $U(\beta, \boldsymbol{H_0}) = \boldsymbol{e}^{-\boldsymbol{i\beta H_0}}$ and $U(\alpha, \boldsymbol{H_p}) = \boldsymbol{e}^{-i\propto \boldsymbol{H_p}}$, where $\frac{1}{p}$ has been absorbed into the parameters α and β

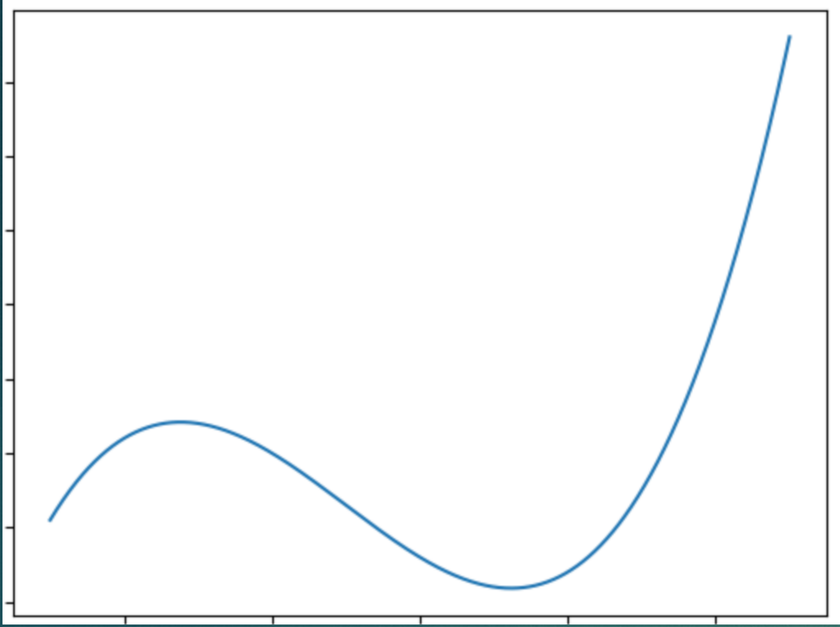
- Furthermore, allow α and β to be different for each term of the product:

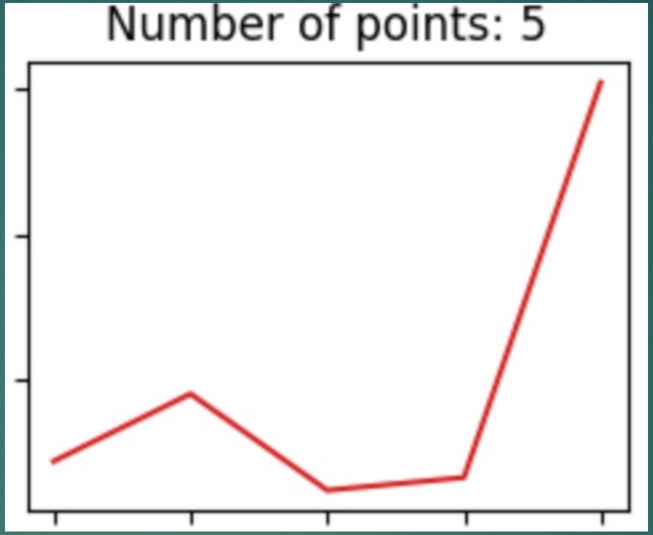
$$\boldsymbol{e}^{(-\boldsymbol{i\beta H_0})+(-\boldsymbol{i}\propto \boldsymbol{H_p})} \approx \prod_{\boldsymbol{j=1}}^{\boldsymbol{p}} \left(U(\beta_j, \boldsymbol{H_0})U(\alpha_j, \boldsymbol{H_p})\right)$$

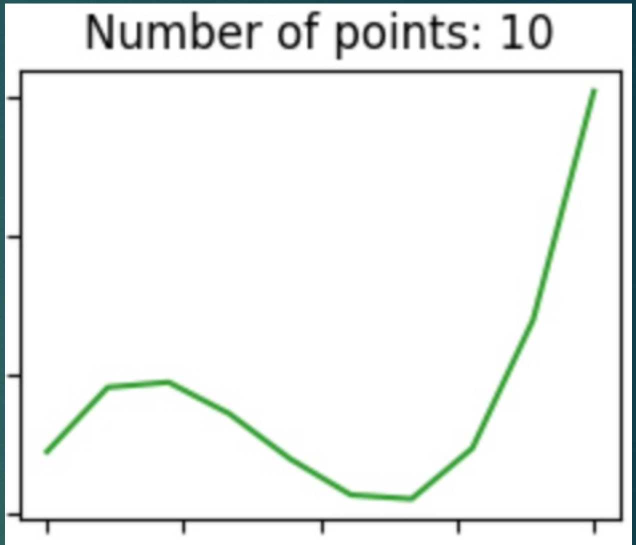
- Finally we get:

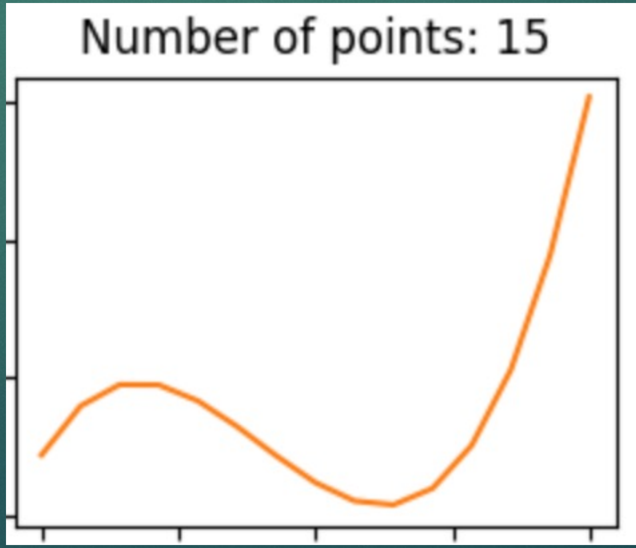
$$|\psi(\alpha,\beta)\rangle = U(\beta_p, \boldsymbol{H_0})U(\alpha_p, \boldsymbol{H_p}) \cdots U(\beta_1, \boldsymbol{H_0})U(\alpha_1, \boldsymbol{H_p})|\psi(0)\rangle$$

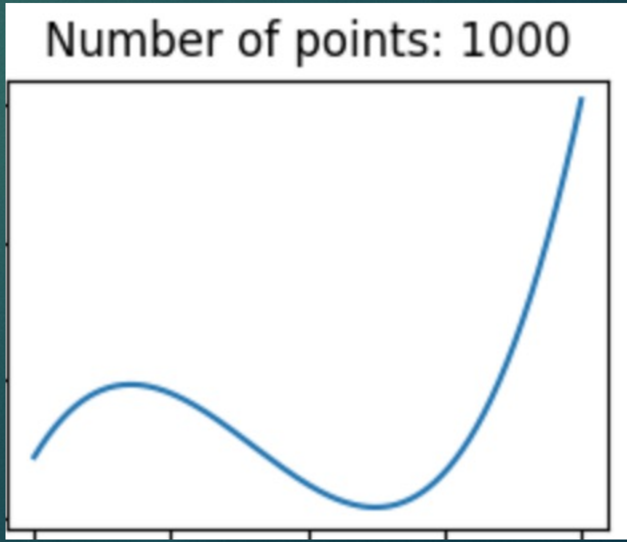
QAOA operator: trotterization









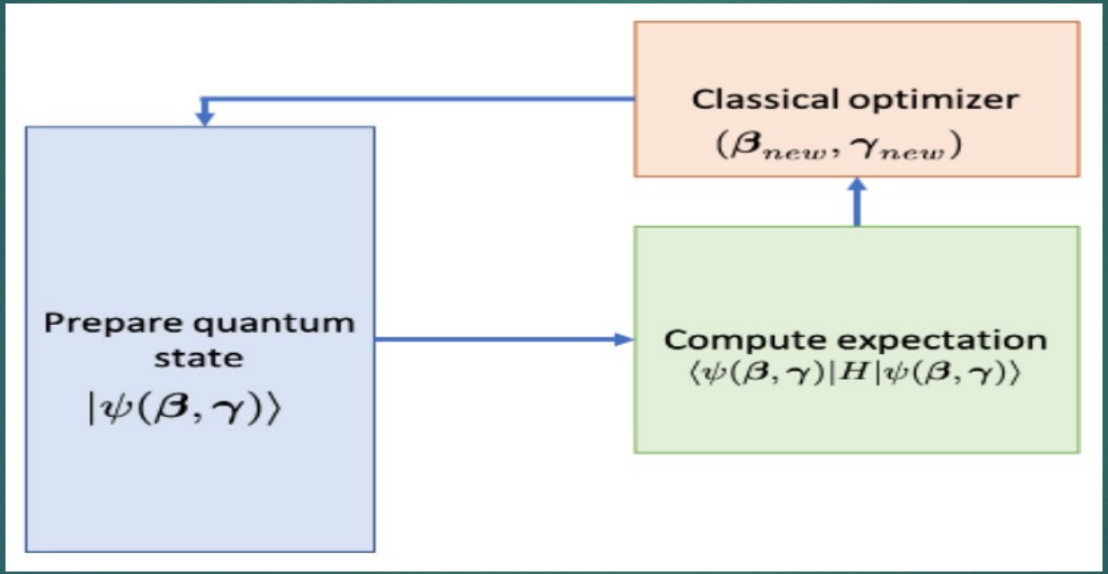


trotterization:
discretization of the time evolution

QAOA: overall

- find (α^*, β^*) such that the expectation of H_p is minimized:

$$|\psi(\alpha^*, \beta^*)\rangle = \underset{\alpha,\beta}{\operatorname{argmin}} \langle|\psi(\alpha, \beta)|H_p\, |\psi(\alpha, \beta)\rangle$$



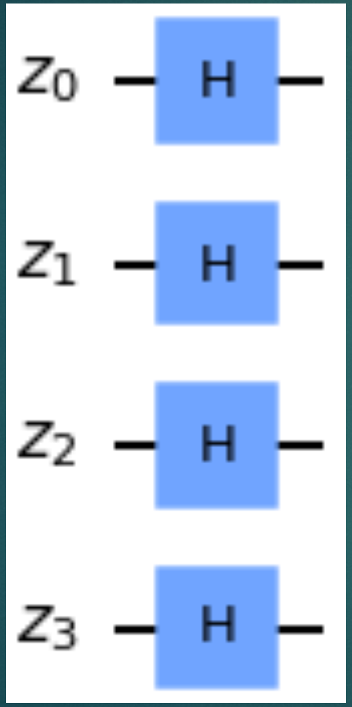
- sample basis states $|z\rangle$ from $|\psi(\alpha^*, \beta^*)\rangle$ to find a solution

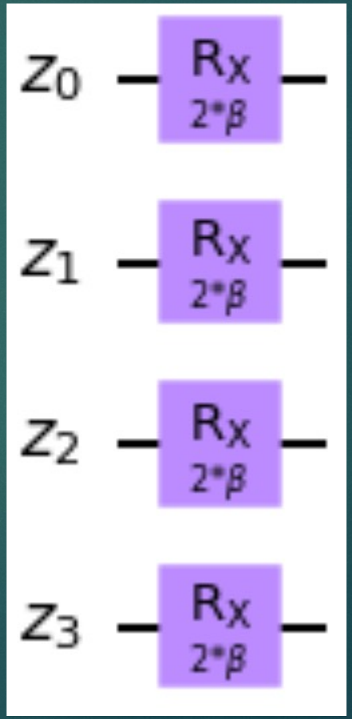
The mixer Hamiltonian: H_0

- H_0 must:
  - have a smooth landscape
  - be easy to implement
  - have a ground state easy to prepare
  - not commute with H_p
- A good option is using a Pauli-X (σ_j^x) based Hamiltonian:
  - $H_0 = \sum_j^n \sigma_j^x$; $U(\beta, \boldsymbol{H_0}) = \boldsymbol{e^{-i\beta H_0}} = \boldsymbol{R_X^{\otimes n}}(\beta)$, i.e, a X-rotation for all n qubits
  - The ground state of H_0 is $|+\rangle^{\otimes \boldsymbol{n}} = H^{\otimes \boldsymbol{n}}|0\rangle$, where H is an Hadarmard gate
  - H_p will be based on Pauli-Z operators (σ^z), which do not commute with σ^x

The mixer Hamiltonian: H_0

$|\psi(0)\rangle = |s\rangle = |+\rangle^{\otimes \boldsymbol{n}} = H^{\otimes \boldsymbol{n}}|0\rangle$; $H_0 = \sum_j^n \sigma_j^x$; $U(\beta, \boldsymbol{H_0}) = \boldsymbol{e}^{-i\beta \boldsymbol{H_0}} = \boldsymbol{R_X^{\otimes n}}(\beta)$;





The mixer Hamiltonian: H_0

- What is the role of $\boldsymbol{H_0}$?
  - Suppose there is no $\boldsymbol{H_0}$. Then the circuit would repetitively apply $U(\alpha_j, \boldsymbol{H_p})$
  - Once an eigenstate of $\boldsymbol{H_p}$ is reached the state won't evolve any further. Applying an operator to its eigenvector can change its length, but not direction.

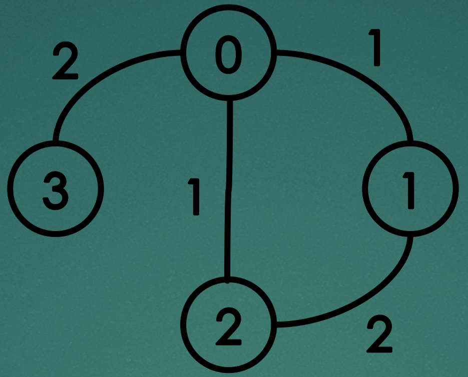
    The same would apply if $\boldsymbol{H_0}$ would commute with $\boldsymbol{H_p}$.
  - $\boldsymbol{H_0}$ allows the quantum system to escape from local minima and search for the ground state;
- Imagine you're in a forest full of traps and you want to find a treasure. $\boldsymbol{H_p}$ allows you to feel the treasure and move towards it, while $\boldsymbol{H_0}$ provides you with a set of rules on how to move to avoid the traps.
  Without $\boldsymbol{H_p}$ you have no idea which direction to go and without $\boldsymbol{H_0}$ you cannot make any moves.
  [https://www.mustythoughts.com/quantum-approximate-optimization-algorithm-explained]

The problem Hamiltonian: H_p

- H_p is built from the problem cost function $C(z)$, where $z \in \{0,1\}^n$ is a bit string
- Let $z = z_0 z_1 \cdots z_n$, where $z_i = \{0,1\}$ refers to the ith bit in the string
- Using Pauli-Z (σ^z) operators to develop H_p (σ^z does not commute with σ^x) we can map:
  - $z_i \rightarrow \frac{1-\sigma_i^z}{2}$, $\sigma_i^z = 1 \Longrightarrow z_i = 0$; $\sigma_i^z = -1 \Longrightarrow z_i = 1$

H_p: MaxCUT

G=(V,E)



$$A = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

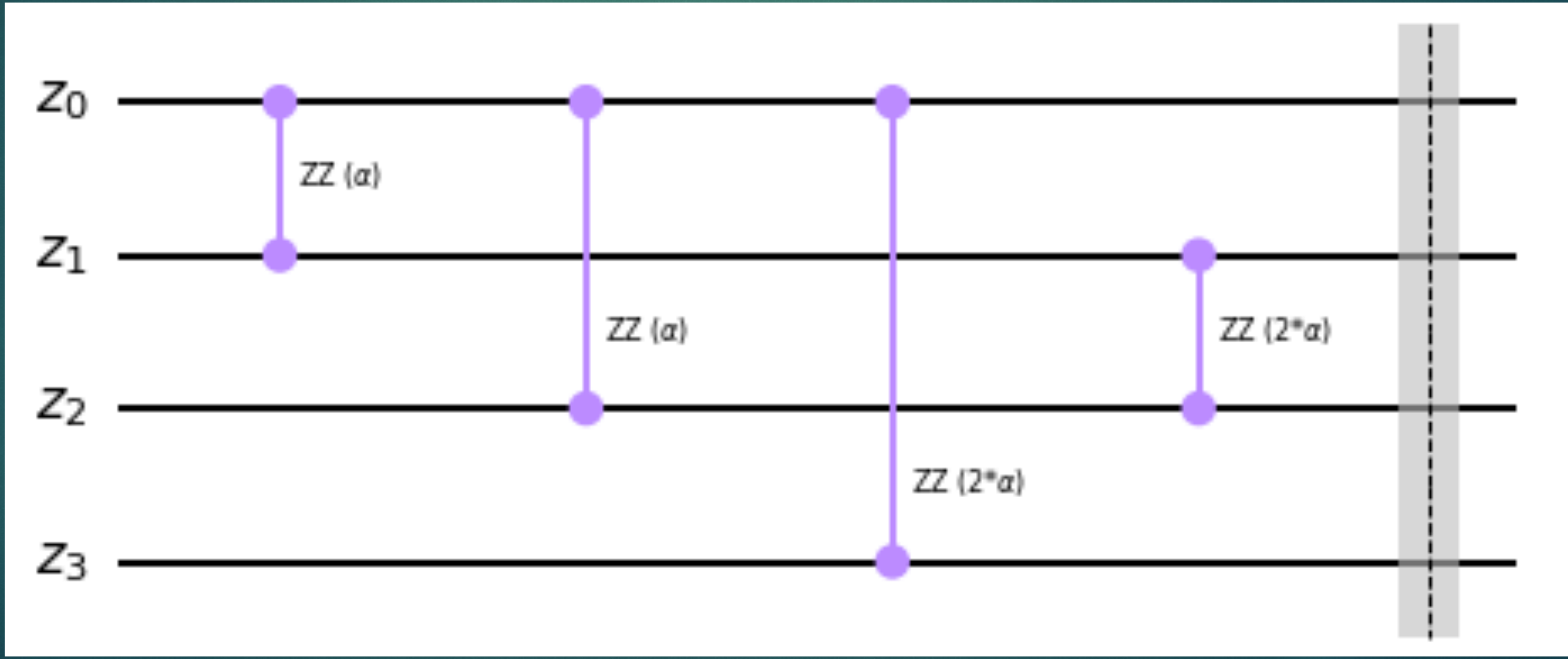
- $C(z) = \sum_{(i,j)\epsilon E}\left[w_{i,j}\left(z_i(1 - z_j) + z_j(1 - z_i)\right)\right]$ $\qquad z_i \rightarrow \frac{1-\sigma_i^Z}{2}$
- $H_p = \frac{1}{2}\sum_{(i,j)\epsilon E}\left[w_{i,j}\left(1 - \sigma_i^Z\sigma_j^Z\right)\right]$
- $H_p = \frac{1}{2}\left[(1 - \sigma_0^Z\sigma_1^Z) + (1 - \sigma_0^Z\sigma_2^Z) + 2(1 - \sigma_1^Z\sigma_2^Z) + 2(1 - \sigma_0^Z\sigma_3^Z)\right]$
- $H_p = (\sigma_0^Z \otimes \sigma_1^Z \otimes \mathbb{I}_2 \otimes \mathbb{I}_3) + (\sigma_0^Z \otimes \mathbb{I}_1 \otimes \sigma_2^Z \otimes \mathbb{I}_3) + 2(\mathbb{I}_0 \otimes \sigma_1^Z \otimes \sigma_2^Z \otimes \mathbb{I}_3) + 2(\sigma_0^Z \otimes \mathbb{I}_1 \otimes \mathbb{I}_2 \otimes \sigma_3^Z)$
- $U(\alpha, H_p) = e^{-i\alpha(\sigma_0^Z \otimes \sigma_1^Z \otimes \mathbb{I}_2 \otimes \mathbb{I}_3)} e^{-i\alpha(\sigma_0^Z \otimes \mathbb{I}_1 \otimes \sigma_2^Z \otimes \mathbb{I}_3)} e^{-i2\alpha(\mathbb{I}_0 \otimes \sigma_1^Z \otimes \sigma_2^Z \otimes \mathbb{I}_3)} e^{-i2\alpha(\sigma_0^Z \otimes \mathbb{I}_1 \otimes \mathbb{I}_2 \otimes \sigma_3^Z)}$

See section 4 of https://arxiv.org/pdf/1001.3855.pdf for details on how to develop the Hamiltonian and the respective circuit

H_p: MaxCUT

$$U(\alpha, H_p) = e^{-i\alpha(\sigma_0^z \otimes \sigma_1^z \otimes \mathbb{I}_2 \otimes \mathbb{I}_3)} e^{-i\alpha(\sigma_0^z \otimes \mathbb{I}_1 \otimes \sigma_2^z \otimes \mathbb{I}_3)} e^{-i2\alpha(\mathbb{I}_0 \otimes \sigma_1^z \otimes \sigma_2^z \otimes \mathbb{I}_3)} e^{-i2\alpha(\sigma_0^z \otimes \mathbb{I}_1 \otimes \mathbb{I}_2 \otimes \sigma_3^z)}$$

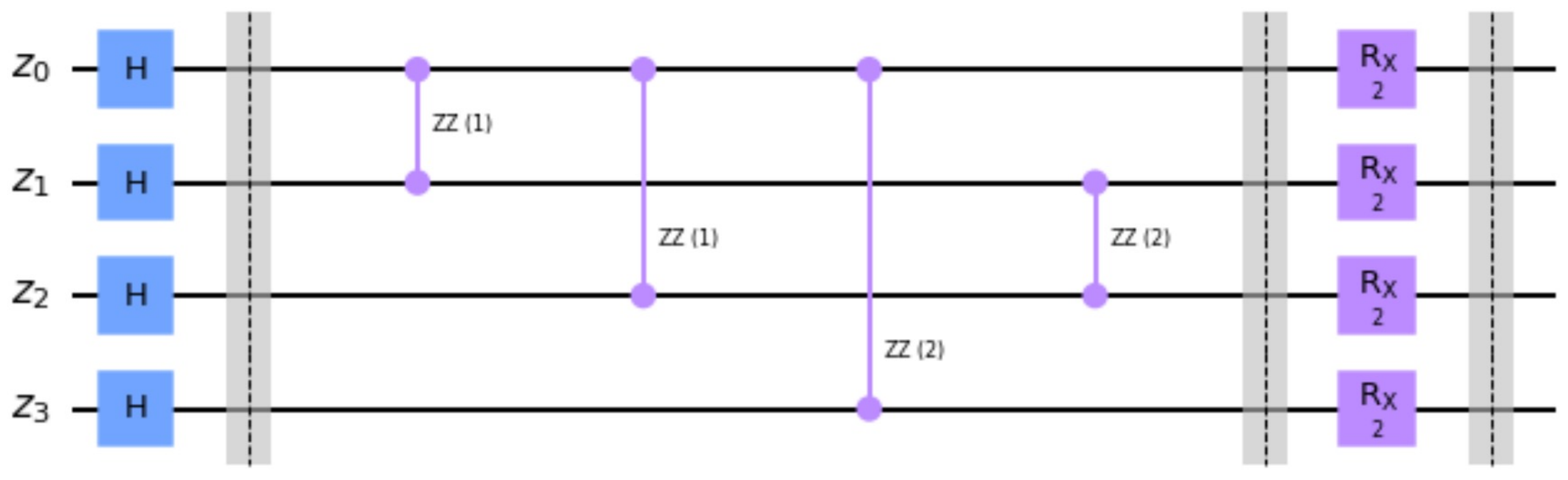
$$U(\alpha, H_p) = R_{0,1}^z(2\alpha) R_{0,2}^z(2\alpha) R_{1,2}^z(4\alpha) R_{0,3}^z(4\alpha)$$



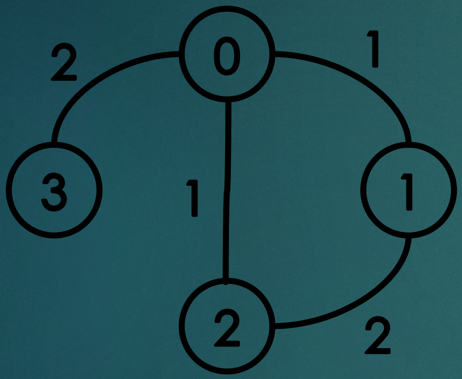
MaxCUT: cost and loss functions

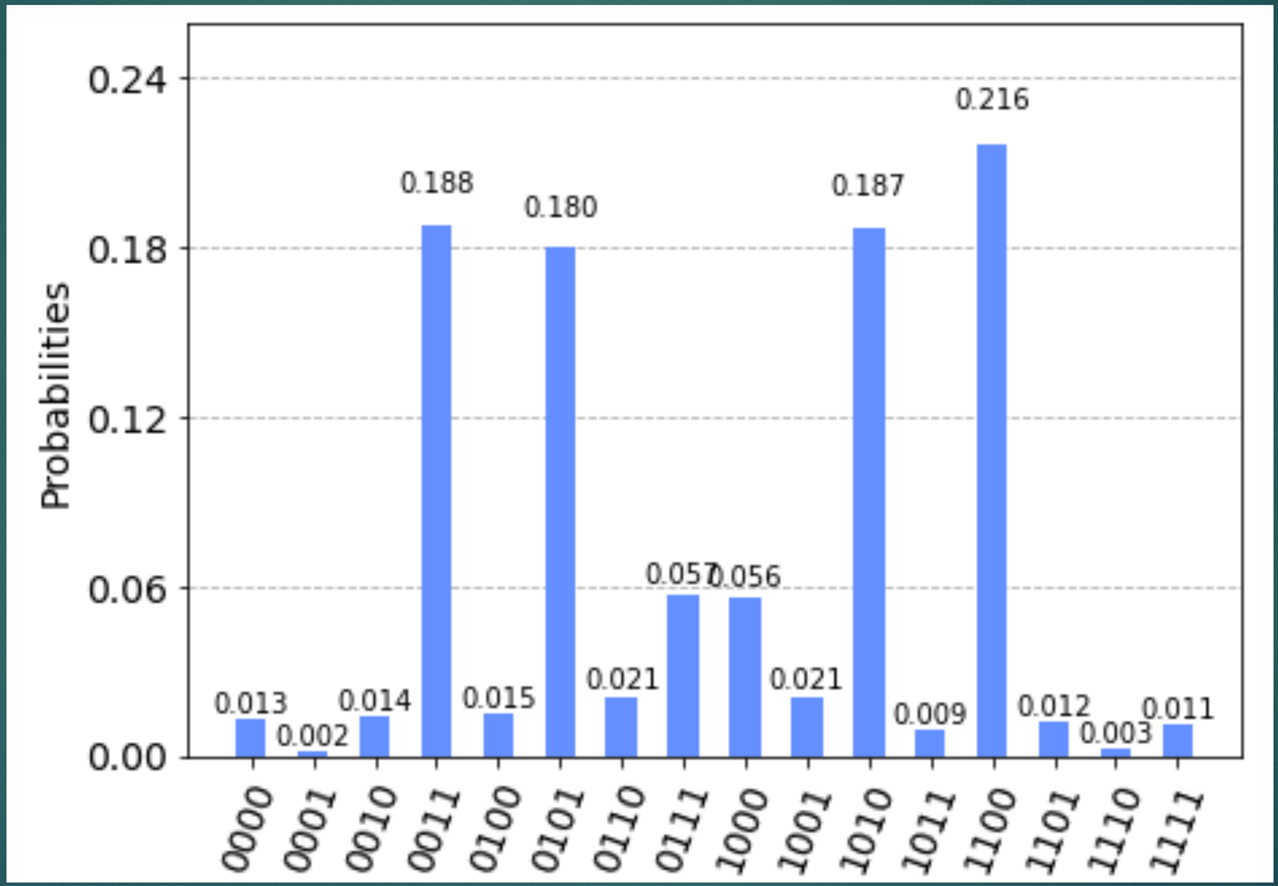

```
def loss_function(z,counts):
    loss=0
    for i in range(n_qubits):
        for j in range(i+1,n_qubits):
            if z[i] != z[j] and A[i,j]:
                loss -= A[i,j]
    loss *= counts
    return loss

def cost_function():
    counts  = execute_circuit(full_qaoa_circuit, shots=2048)
    cost = 0
    for (z,c) in counts.items():
        cost+=loss_function(z, c)
    cost /= shots
    return cost
```

MaxCUT: p=1



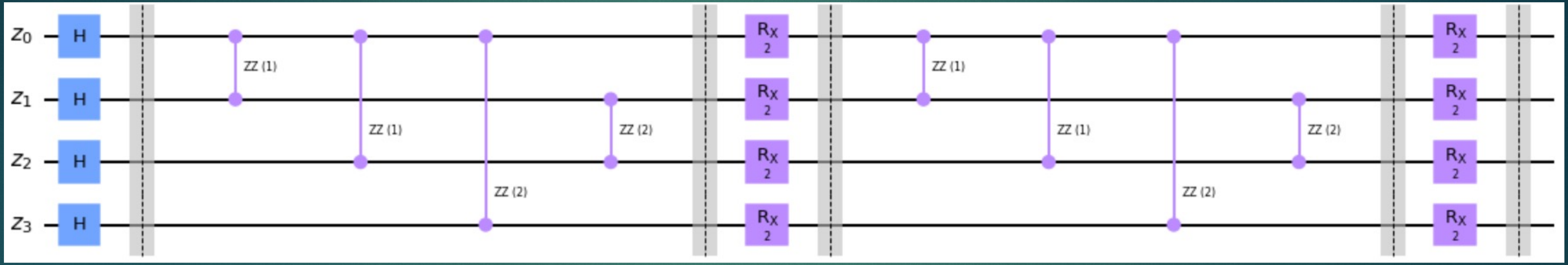
MaxCUT: p=1



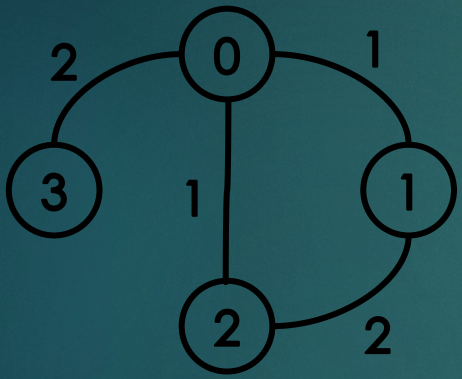


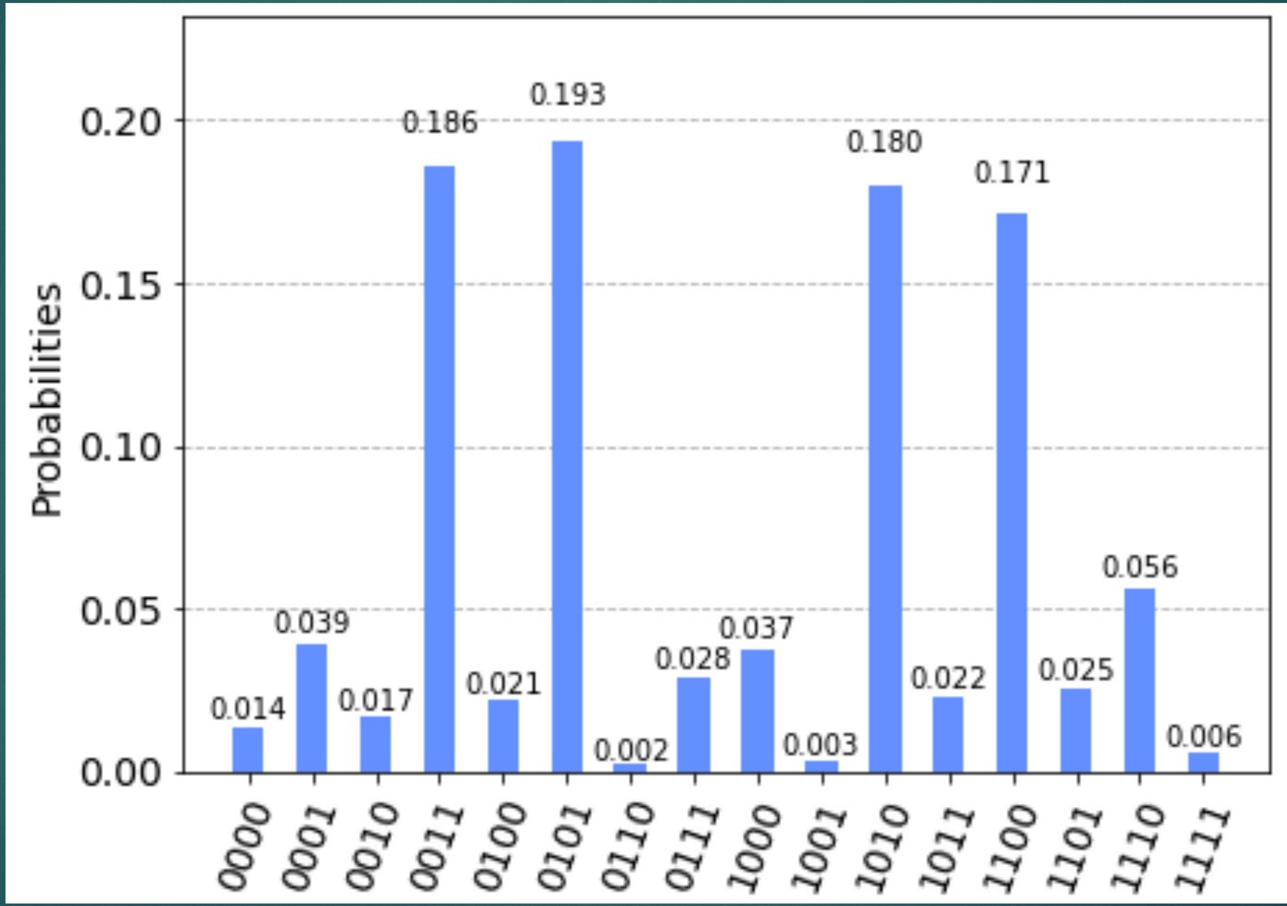
$z=z_0z_1z_2z_3$	Cut
0000	0
0001	2
0010	3
0011	5
0100	3
0101	5
0110	2
0111	4
1000	4
1001	2
1010	5
1011	3
1100	5
1101	3
1110	2
1111	0

MaxCUT: p=2


z_0
z_1
z_2
z_3
H
H
H
H
ZZ (1)
ZZ (1)
ZZ (2)
ZZ (2)
Rx 2
Rx 2
Rx 2
Rx 2
ZZ (1)
ZZ (1)
ZZ (2)
ZZ (2)
Rx 2
Rx 2
Rx 2
Rx 2

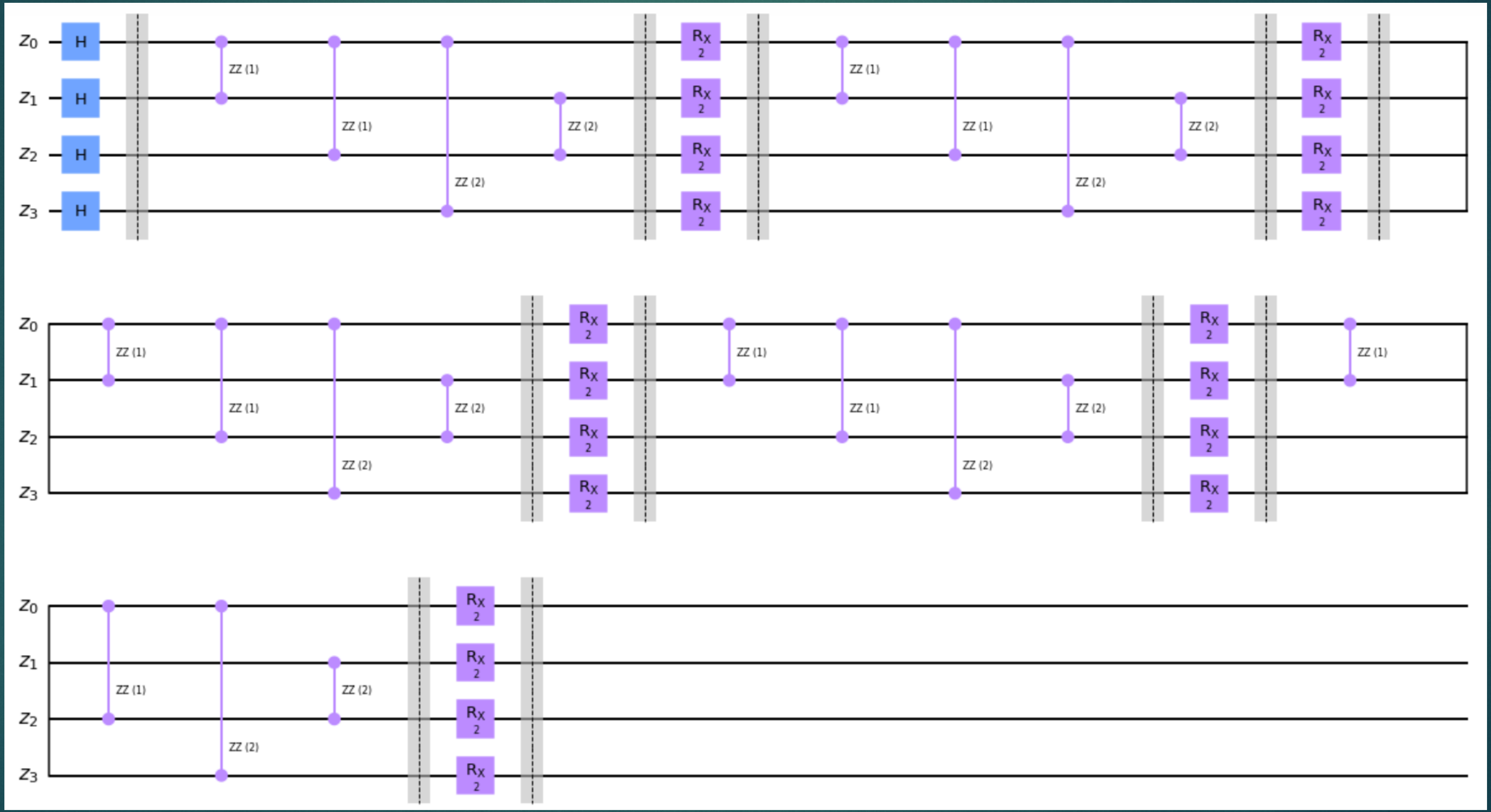
MaxCUT: p=2



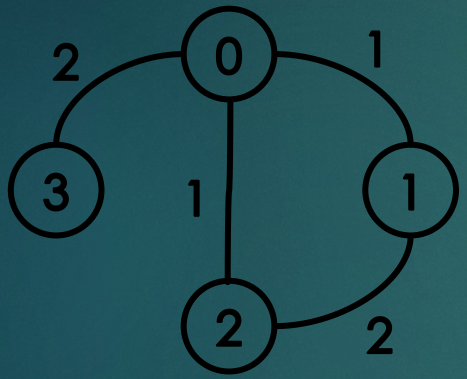


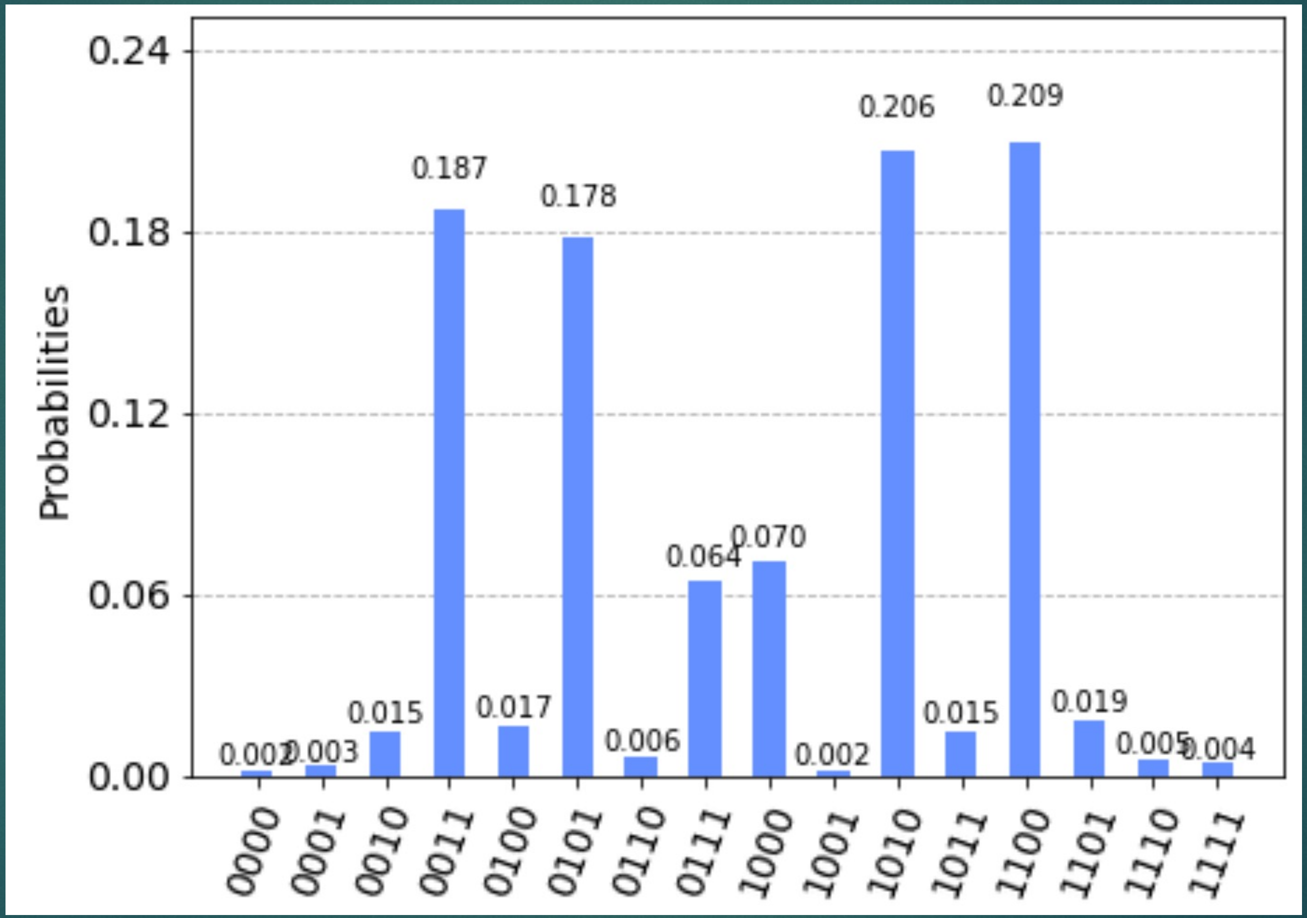
$z=z_0z_1z_2z_3$	Cut
0000	0
0001	2
0010	3
0011	5
0100	3
0101	5
0110	2
0111	4
1000	4
1001	2
1010	5
1011	3
1100	5
1101	3
1110	2
1111	0

MaxCUT: p=5



MaxCUT: p=5





$z=z_0z_1z_2z_3$	Cut
0000	0
0001	2
0010	3
0011	5
0100	3
0101	5
0110	2
0111	4
1000	4
1001	2
1010	5
1011	3
1100	5
1101	3
1110	2
1111	0