Quantum Max-Cut and QAOA

1 Introduction

The **Max-Cut** problem consists of dividing the vertices of a graph $G = (V, E)$ into two non-empty sets S and $\overline{S}$ in order to **maximize the sum of the weights of the edges connecting vertices in different sets**.

Formally, we want:

$$\mathrm{cut}(S, \overline{S}) = \sum_{(i,j)\in E} w_{ij} \cdot \delta(x_i \neq x_j)$$

where:

- w_{ij} is the weight of edge (i, j),
- x_i indicates the set to which vertex i belongs,
- $\delta(x_i \neq x_j)$ is 1 if the vertices belong to different sets and 0 otherwise.

2 Why transform into an adjacency matrix?

To apply classical and quantum algorithms such as **QAOA**, we need to **represent the graph in a direct matrix form**, because:

- It enables efficient calculations,
- It is the basis for constructing the **objective function**,
- It serves as the input for generating the **quantum Hamiltonian**.

However, data often comes in **edge list format**, for example:

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 1 \end{bmatrix}$$

Each row means: **(vertex i, vertex j, weight)**.

3 How to transform mathematically?

If we have n vertices, the adjacency matrix A is an $n \times n$ matrix defined by:

$$A[i][j] = \begin{cases} w_{ij}, & \text{if edge } (i,j) \text{ exists} \\ 0, & \text{otherwise} \end{cases}$$

Example: For $n = 4$ and edges:

- (1, 2) with weight 3
- (2, 3) with weight 4
- (3, 4) with weight 1

The resulting matrix will be:

$$\begin{bmatrix} 0 & 3 & 0 & 0 \\ 3 & 0 & 4 & 0 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Why is this important for the quantum computer? The quantum computer needs to "understand" the structure of the problem. The adjacency matrix is like a relationship map that we show to the quantum algorithm, saying: *"Look, these are the connections that matter!"*

4 What is the Hamiltonian in quantum computing?

The Hamiltonian is an operator H that measures the total energy of the quantum system.

In my algorithm, I used two Hamiltonians:

1. Cost Hamiltonian
2. Mixer Hamiltonian

4.1 Cost Hamiltonian

In quantum computing, the Cost Hamiltonian H_C is a mathematically constructed operator designed to encode the essence of an optimization problem in the language of quantum mechanics. It assigns an energy value to each possible solution, so that the lowest energy state (the ground state) represents the optimal solution we seek.

H_C defined for the MAX-CUT problem:

$$H_C = \frac{1}{2} \sum_{(i,j)\in E} (I - Z_i Z_j)$$

Where:

- E is the set of edges of the graph.
- I is the identity matrix.
- Z_i and Z_j are Pauli-Z operators acting on qubits i and j, respectively.

4.2 Mixer Hamiltonian

The mixer Hamiltonian H_M is the *"quantum push"* that prevents the search from getting stuck in poor solutions and helps explore the solution space intelligently, using superposition and entanglement.

$$H_M = \sum_{i=1}^{N} X_i$$

- N is the number of qubits.
- X_i is the Pauli-X operator acting on qubit i.

5 QAOA Algorithm for Max-Cut

The QAOA is applied to Max-Cut following a structured procedure:

1. **Definition of Input Data**:
   - p: circuit depth (number of layers);
   - $\vec{\gamma} = (\gamma_0, \gamma_1, \ldots, \gamma_{p-1})$: phase parameters;
   - $\vec{\beta} = (\beta_0, \beta_1, \ldots, \beta_{p-1})$: mixing parameters;
   - Graph $G = (V, E)$ represented by an adjacency matrix A, where:

$$A[i][j] = \begin{cases} w_{ij}, & \text{if } (i,j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

2. **Definition of the Cost Hamiltonian**:

$$H_C = \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - Z_i Z_j),$$

3. **Definition of the Mixer Hamiltonian**:

$$H_M = \sum_{i \in V} X_i,$$

4. **Construction of the QAOA Circuit**:
   - Initialize the state $|\psi_0\rangle = |+\rangle^{\otimes n}$ by applying Hadamard gates $H^{\otimes n}$ to $|0\rangle^{\otimes n}$.
   - For each layer $k = 0$ to $p - 1$:
     - Apply the phase operator: $U_C(\gamma_k) = e^{-i\gamma_k H_C}$, implemented via rotations $RZZ(2\gamma_k w_{ij})$.
     - Apply the mixing operator: $U_M(\beta_k) = e^{-i\beta_k H_M}$, implemented via rotations $R_X(2\beta_k)$.
   - The final state is:

$$|\psi(\vec{\beta}, \vec{\gamma})\rangle = \prod_{k=0}^{p-1} U_M(\beta_k) U_C(\gamma_k) |\psi_0\rangle.$$

5. **Computation of the Expected Value**:

$$F(\vec{\beta}, \vec{\gamma}) = \langle \psi(\vec{\beta}, \vec{\gamma}) | H_C | \psi(\vec{\beta}, \vec{\gamma}) \rangle.$$

6. **Parameter Optimization**: A classical optimizer (e.g., COBYLA) is used to maximize $F(\vec{\beta}, \vec{\gamma})$, adjusting the parameters iteratively.

The implementation of the operators $e^{-i\gamma_k H_C}$ and $e^{-i\beta_k H_M}$ can be carried out in Qiskit using decompositions into native quantum gates such as RZZ and R_X.

6 References

1. Farhi, E., Goldstone, J., & Gutmann, S. (2014). *A Quantum Approximate Optimization Algorithm.* arXiv:1411.4028.

2. Qiskit Documentation: Quantum Approximate Optimization Algorithm

3. Lucas, A. (2014). *Ising formulations of many NP problems.* Frontiers in Physics, 2, 5.