1 The Max-Cut Problem Using QAOA

The Max-Cut problem is a well-known combinatorial optimization problem in graph theory. Given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, the objective is to partition the vertices into two disjoint sets, say S_0 and S_1, such that the number of edges connecting vertices in different sets is maximized. This number is known as the ”cut size”.
For a given partition, the cut size can be expressed as:

$$\text{Cut}(S_0, S_1) = |\{(u, v) \in E \mid u \in S_0, v \in S_1\}|$$

Finding the exact solution to Max-Cut is an NP-hard problem, meaning that for large graphs, classical computers cannot solve it in a reasonable amount of time. The Quantum Approximate Optimization Algorithm (QAOA) provides a hybrid quantum-classical approach to find good approximate solutions.

1.1 Problem Instance

This report addresses the Max-Cut problem for a specific 6-vertex graph defined by the following edges:

$$E = \{(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 0), (0, 3), (1, 4), (2, 5)\}$$

The goal is to use QAOA to find a partition that yields a cut size close to the maximum possible value.

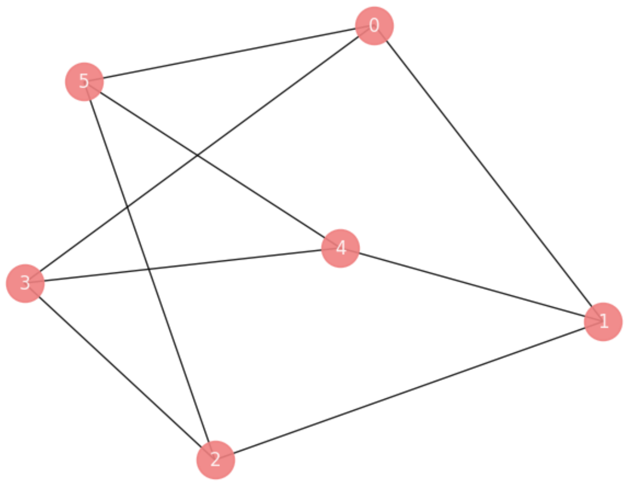

Figure 1: The 6-vertex graph for the Max-Cut problem.

1.2 Algorithm Implementation

The implementation follows these key steps:

1.2.1 Mapping to a Cost Hamiltonian

The Max-Cut problem is first mapped to a quantum mechanical equivalent, the **Cost Hamiltonian** (H_C). Each vertex is represented by a qubit. A partition can be represented by a bitstring $z = z_1 z_2 ... z_n$, where $z_i = 0$ if vertex i is in set S_0 and $z_i = 1$ if it is in set S_1.

The cost function for an edge (u, v) is defined to contribute $+1$ to the cut if the vertices are in different sets ($z_u \neq z_v$) and 0 otherwise. This can be expressed using Pauli-Z operators:

$$C_{uv} = \frac{1}{2}(1 - Z_u Z_v)$$

The total Cost Hamiltonian is the sum over all edges:

$$H_C = \sum_{(u,v)\in E} C_{uv} = \sum_{(u,v)\in E} \frac{1}{2}(I - Z_u Z_v)$$

Maximizing the cut size is equivalent to maximizing the expectation value $\langle\psi|H_C|\psi\rangle$.

1.2.2 QAOA Circuit (Ansatz)

The QAOA circuit, or ansatz, is constructed from p alternating layers of two unitary operators:

1. **Cost Unitary** $U_C(\gamma) = e^{-i\gamma H_C}$: This unitary encodes the problem's cost function. It is parameterized by an angle γ. For the Max-Cut Hamiltonian, this is implemented using 'RZZ' gates for each edge.

2. **Mixer Unitary** $U_M(\beta) = e^{-i\beta H_M}$: This unitary, parameterized by an angle β, allows the quantum state to explore different possible solutions. A common choice for the mixer Hamiltonian is $H_M = \sum_{i\in V} X_i$, where X_i is the Pauli-X operator on qubit i. This is implemented using 'RX' gates on all qubits.

The complete ansatz for a given set of parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta}) = (\gamma_1, ..., \gamma_p, \beta_1, ..., \beta_p)$ is:

$$|\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = U_M(\beta_p)U_C(\gamma_p)\cdots U_M(\beta_1)U_C(\gamma_1)|+\rangle^{\otimes n}$$

where $|+\rangle^{\otimes n}$ is the uniform superposition state created by applying Hadamard gates to all qubits initialized in the $|0\rangle$ state.

1.2.3 Classical Optimization

The core of QAOA is a classical optimization loop:

1. Choose an initial set of parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$.

2. Construct and run the QAOA circuit with these parameters on a quantum device or simulator.

3. Measure the qubits to obtain a distribution of bitstrings.

4. Calculate the expectation value of the Cost Hamiltonian, which is the expected cut size.
5. Feed this value to a classical optimizer (e.g., COBYLA), which suggests a new set of parameters.
6. Repeat until the expectation value is maximized.

The objective function for the optimizer is the negative of the expected cut size, as optimizers typically perform minimization.

1.3 Results and Analysis

1.3.1 Optimization and Solution

The QAOA algorithm was run with $p = 3$ layers. The classical optimizer successfully found a set of optimal parameters that maximize the expected cut size. The final state produced by the QAOA circuit with the optimal parameters shows a high probability of measuring the bitstring `010101`, which corresponds to a perfect cut.

Table 1: QAOA Optimization Results for $p = 3$

Metric	**Value**
Maximum Expected Cut	7.5248
Most Likely Solution (bitstring)	`010101`
Cut Size of Most Likely Solution	9

1.3.2 Classical Comparison and Verification

To verify the quality of the QAOA solution, a classical brute-force solver was implemented. This solver checks all $2^6 = 64$ possible partitions to find the exact maximum cut. The QAOA algorithm found a solution (`010101`) with a cut size of 9, which matches the exact maximum cut found by the brute-force method. This gives an **approximation ratio** of:

Table 2: Classical Brute-Force Verification

Metric	**Value**
Exact Max-Cut Size	9
Optimal Partitions (bitstrings)	`'101010'`, `'010101'`

$$\text{Approximation Ratio} = \frac{\text{QAOA Cut}}{\text{Max Cut}} = \frac{9}{9} = 1.0000$$

An approximation ratio of 1.0 indicates that the QAOA algorithm successfully found an optimal solution for this problem instance.

1.4 Conclusion

The Quantum Approximate Optimization Algorithm was successfully implemented using Qiskit to solve the Max-Cut problem on a 6-vertex graph. With $p = 3$ layers, the algorithm was able to find one of the exact optimal solutions, achieving a perfect approximation ratio of 1.0. This demonstrates the potential of QAOA as a powerful tool for tackling combinatorial optimization problems, even for small-scale instances where it can find exact solutions. The hybrid quantum-classical nature of the algorithm makes it a promising candidate for execution on near-term, noisy intermediate-scale quantum (NISQ) devices.