Solving MAX-Cut Problem with QAOA

Armin Ahmadkhaniha
ahmadkha@mcmaster.ca

June 23, 2025

Abstract

The Max-Cut problem is a well-known combinatorial optimization problem that seeks to partition the vertices of an undirected graph into two subsets, maximizing the number of edges between them. It has broad applications in fields such as network design and statistical physics. In this report, we explore the Max-Cut problem and its connection to the Quadratic Unconstrained Binary Optimization (QUBO) formulation, where the goal is to find a binary vector that maximizes a quadratic cost function. We demonstrate how the Max-Cut problem can be expressed as a QUBO problem by defining the appropriate weights and cost function. Furthermore, we introduce the Quantum Approximate Optimization Algorithm (QAOA), a quantum algorithm that provides an approximate solution to combinatorial optimization problems like Max-Cut.

Keywords: Combinatorial Optimization, Max-Cut, QAOA, QUBO

1 Introduction

The Max-Cut problem is a well-known optimization problem in graph theory. Given an undirected graph, the task is to partition the vertices into two subsets such that the number of edges between these subsets is maximized. This problem is of great importance in areas like approximation algorithms and theoretical computer science.
Definition 1.1 (Max-Cut): Max-Cut is the problem of [2], given an undirected graph $G = (V, E)$ with vertices V, to find a partition V_1, V_2 of V such that the number of edges $\{u, v\}$ where $\{u, v\} \cap V_1 \neq \emptyset$ and $\{u, v\} \cap V_2 \neq \emptyset$ is maximized.
Definition 1.2 (Max-Ek-Lin-2): Max-Ek-Lin-2 is the problem of, given a system L of linear equations over $\mathbb{Z}_2$, with exactly k variables in each equation, to find x that maximizes $N(L, x)$, where $N(L, x)$ denotes the number of satisfied equations under the assignment x [2].
Definition 1.3 (a-gadget): The a-gadget is a construction used in reductions, particularly in the Max-E3-Lin-2 to Max-Cut reduction [3]. It encodes the possible assignments of the variables and creates edges in the graph such that satisfying the equation corresponds to a favorable cut in the Max-Cut solution. The gadget's structure ensures that a correct solution to the Max-Cut problem also corresponds to a valid assignment in the Max-E3-Lin-2 problem.
LEMMA 1.1 Suppose there is an R-gadget reducing Max-E3-Lin-2 to an optimization problem O. Then, unless NP = P, for any $\epsilon > 0$, O cannot be approximated within

$\frac{2R}{2R-1} - \epsilon$ in polynomial time.
Proof: using LEMMA 2.8 in [3], we only sketch the proof. We use the gadget to construct an instance of O. If the total weight of the Max-E3-Lin-2 instance is 1, then for any solution that satisfies equations of total weight w, the corresponding solution of the transformed problem satisfies constraints of total weight

$$w\alpha + (1 - w)(\alpha - 1).$$

Since it is NP-hard to distinguish the two cases when $w = 1 - \delta$ and $w = \frac{1}{2} + \delta$, if we could determine the optimum of the transformed problem to a better accuracy than

$$\frac{(1-\delta)\alpha + \delta(\alpha - 1)}{(1/2 + \delta)\alpha + (1/2 - \delta)(\alpha - 1)},$$

we would solve an NP-hard problem. Since δ was arbitrary, the *lemma* follows. □
THEOREM 1.1 For any $\epsilon > 0$, it is NP-hard to approximate undirected Max-Cut within a factor of $\frac{17}{16} - \epsilon$.
Proof: Use the 8-gadget for $abc = 1$ and the 9-gadget for $abc = 1$. If there are more equations of the second type, we complement all the variables. The result follows from a minor extension of **LEMMA 1.1**. □

1.1 Quadratic Optimization Problem

A Quadratic Optimization problem involves optimizing a quadratic objective function, which typically has the form:

$$\text{maximize} \quad f(x) = \sum_{i,j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n} b_i x_i$$

where $x_i \in \{0, 1\}$ are binary decision variables, and a_{ij} and b_i are coefficients. This type of problem is widely encountered in areas like combinatorial optimization and can be formulated in various ways depending on the specific constraints and objectives.

The Quadratic Unconstrained Binary Optimization (QUBO) problem is a specific case of quadratic optimization problems where the goal is to minimize or maximize a quadratic cost function subject to no constraints. It is typically written as:

$$C(x) = \sum_{i,j=1}^{n} x_i Q_{ij} x_j + \sum_{i=1}^{n} c_i x_i$$

where Q_{ij} is a symmetric matrix, and c_i is a vector of linear coefficients. A QUBO problem can be solved by classical algorithms or, more recently, by quantum algorithms.

Using the relationship between the problem's weights W_{ij} and the elements of the matrix Q_{ij}, the QUBO formulation of a problem like Max-Cut can be expressed as:

$$c_i = \sum_{j=1}^{n} W_{ij}, \quad Q_{ij} = -W_{ij}$$

The Max-Cut problem can be formulated as a QUBO problem by defining binary variables $x_i \in \{0, 1\}$, which correspond to the vertices in the graph. The objective is to maximize the number of edges that are "cut" between the two sets. The QUBO formulation for Max-Cut is given by:

$$C(x) = \sum_{i,j=1}^{n} x_i Q_{ij} x_j + \sum_{i=1}^{n} c_i x_i$$

$$= x^T Q x + c^T x$$

where $Q_{ij} = -W_{ij}$ and c_i are defined as above. In this case, the weight matrix W represents the graph's edge weights, and the goal is to find the binary vector x that maximizes the cut.

1.2 Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA) is a quantum algorithm designed to solve combinatorial optimization problems like Max-Cut. QAOA uses a parameterized quantum circuit to approximate the solution of a given problem. The algorithm consists of alternating layers of problem-specific operations and mixing operations, where the problem is encoded in the Hamiltonian of the quantum system. The parameters of the quantum circuit are optimized iteratively to minimize the QUBO cost function, providing an approximation to the optimal solution. QAOA offers a promising approach for solving problems such as Max-Cut [1], particularly for large instances where classical algorithms struggle. By leveraging quantum superposition and interference, QAOA can potentially outperform classical heuristics, especially for large, complex problems.

2 QAOA Algorithm

The QAOA algorithm works by applying the unitary operators corresponding to both the cost and mixer Hamiltonians iteratively, with two parameters γ and β that control the evolution. The quantum state at any point in time is given by:

$$|\psi(\gamma, \beta)\rangle = e^{-i\beta H_M} e^{-i\gamma H_C} |\psi(0)\rangle$$

where $|\psi(0)\rangle$ is the initial quantum state. The parameters γ and β are optimized using classical optimization techniques to maximize the expectation value of the cost Hamiltonian H_C.

3 Solving Max-Cut Using the Adiabatic Process

The adiabatic process in quantum mechanics involves evolving a quantum system from an initial Hamiltonian H_M (mixer Hamiltonian) to a final Hamiltonian H_C (cost Hamiltonian), such that the ground state of H_C encodes the solution to the optimization problem. The evolution is governed by the time-dependent Hamiltonian:

$$H(t) = (1 - s(t))H_M + s(t)H_C,$$

where $s(t)$ is a schedule function with $s(0) = 0$ and $s(T) = 1$, and T is the total evolution time. The adiabatic theorem guarantees that if T is sufficiently large, the system will remain in the instantaneous ground state of $H(t)$, thereby producing the ground state of H_C at the end of the evolution.

The cost Hamiltonian H_C for the Max-Cut problem is defined as (derived in Appendix A):

$$H_C = \sum_{i,j=1}^{n} \frac{1}{4} Q_{ij} Z_i Z_j - \sum_{i=1}^{n} \frac{1}{2} \left(c_i + \sum_{j=1}^{n} Q_{ij} \right) Z_i + \left(\sum_{i,j=1}^{n} \frac{Q_{ij}}{4} + \sum_{i=1}^{n} \frac{c_i}{2} \right),$$

where Q_{ij} are the weights derived from the QUBO formulation, and Z_i represents the Pauli-Z operator for the i-th qubit.

The mixer Hamiltonian H_M is defined as:

$$H_M = \sum_{i=1}^{n} X_i,$$

where X_i is the Pauli-X operator for the i-th qubit. Explicitly:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

To implement the adiabatic evolution, the Hamiltonians are exponentiated to apply unitary operations during the process. The matrix exponential of the mixer Hamiltonian is:

$$e^{-i\beta H_M} = \prod_{i=1}^{n} R_X(2\beta),$$

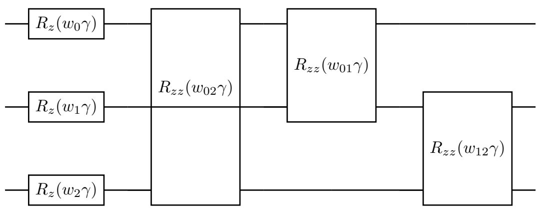
where $R_X(\theta)$ represents a rotation around the X-axis by an angle θ.

The matrix exponential of the cost Hamiltonian is:

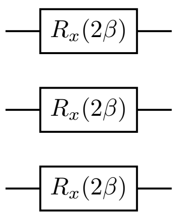
$$e^{-i\gamma H_C} = \prod_{i,j=1}^{n} R_{Z_i Z_j} \left(\frac{1}{2} Q_{ij} \gamma \right) \prod_{i=1}^{n} R_{Z_i} \left(\left(c_i + \sum_{j=1}^{n} Q_{ij} \right) \gamma \right),$$

where $R_{Z_i}(\theta)$ and $R_{Z_i Z_j}(\theta)$ are single- and two-qubit rotations around the Z-axis, respectively.

The quantum system evolves from the ground state of H_M, which is an equal superposition state, to the ground state of H_C, which encodes the solution to the Max-Cut problem. The success of the adiabatic process depends on the smoothness of $s(t)$ and the minimum energy gap between the ground state and the first excited state of $H(t)$ during the evolution.



(a) Cost Hamiltonian for a 3-node graph.



(b) Mixer Hamiltonian for a 3-node graph.

Figure 1: Quantum circuits representing the cost (a) and mixer (b) Hamiltonians for a 3-node graph. These demonstrate the encoding of the Max-Cut problem in QAOA.

4 Conclusion

In this work, we explored the Max-Cut problem, a cornerstone of combinatorial optimization that seeks to maximize the number of edges crossing between two subsets of a graph. We highlighted its NP-hard nature, demonstrating the computational challenges inherent in approximating this problem beyond certain factors. Through a detailed reduction from Max-E3-Lin-2, we established the theoretical hardness of achieving optimal solutions.

To tackle such challenges, we delved into the Quadratic Unconstrained Binary Optimization (QUBO) formulation of Max-Cut, bridging classical optimization with quantum approaches. The QUBO formulation lays the foundation for solving Max-Cut using quantum algorithms, notably the Quantum Approximate Optimization Algorithm (QAOA). We presented QAOA's structure, leveraging parameterized quantum circuits to provide approximate solutions to the Max-Cut problem.

Additionally, we explored the adiabatic process, detailing the cost and mixer Hamiltonians required for encoding Max-Cut onto a quantum system. Using precise mathematical representations, we described the time-evolution operators and matrix exponentiation central to the adiabatic framework. This quantum perspective offers a promising avenue for solving Max-Cut, particularly as quantum hardware continues to advance.

In summary, this report bridges the gap between classical NP-hardness theory and quantum computational approaches, demonstrating how Max-Cut can be approached from both theoretical and practical quantum perspectives, paving the way for efficient solutions to large-scale optimization problems.

5 Codes

For Python Codes sample:
https://github.com/ArminAhmadkhaniha/QAOA-and-maxcut/blob/main/qaoamax.ipynb

References

[1] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *ArXiv:1411.4028 [Quant-Ph]*, 2014.

[2] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48:798, 2001.

[3] L. Trevisan, G. Sorkin, M. Sudan, and D. Williamson. Gadgets, approximation and linear programming. *SIAM J. Comput.*, 29:2074–2097, 2000.

Appendix

A Hamiltonian Derivation for Cost Function

The Max-Cut problem can be expressed with the following cost function:

$$C(x) = x^T Q x + c^T x = \sum_{i,j=1}^{n} Q_{ij} x_i x_j + \sum_{i=1}^{n} c_i x_i.$$

We want to find the Hamiltonian H_C such that:

$$H_C |x\rangle = C(x) |x\rangle .$$

The Pauli-Z operator acts as:

$$Z_i |x\rangle = (-1)^{x_i} |x\rangle = (1 - 2x_i) |x\rangle .$$

Thus, we can express x_i in terms of Z_i as:

$$x_i = \frac{1 - Z_i}{2}.$$

Substituting into the cost function, we have:

$$H_C = \sum_{i,j=1}^{n} Q_{ij} x_i x_j + \sum_{i=1}^{n} c_i x_i,$$

$$H_C = \sum_{i,j=1}^{n} Q_{ij} \frac{1 - Z_i}{2} \frac{1 - Z_j}{2} + \sum_{i=1}^{n} c_i \frac{1 - Z_i}{2}.$$

Simplify each term:

$$H_C = \sum_{i,j=1}^{n} \frac{Q_{ij}}{4}(1 - Z_i - Z_j + Z_i Z_j) + \sum_{i=1}^{n} \frac{c_i}{2}(1 - Z_i).$$

Expanding and grouping terms:

$$H_C = \sum_{i,j=1}^{n} \frac{Q_{ij}}{4} + \sum_{i=1}^{n} \frac{c_i}{2} - \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \frac{Q_{ij}}{4} \right) Z_i - \sum_{j=1}^{n} \left(\sum_{i=1}^{n} \frac{Q_{ij}}{4} \right) Z_j - \sum_{i=1}^{n} \frac{c_i}{2} Z_i + \sum_{i,j=1}^{n} \frac{Q_{ij}}{4} Z_i Z_j.$$

After simplifying further:

$$H_C = \sum_{i,j=1}^{n} \frac{Q_{ij}}{4} Z_i Z_j - \sum_{i=1}^{n} \left(c_i + \sum_{j=1}^{n} Q_{ij} \right) \frac{Z_i}{2} + \sum_{i,j=1}^{n} \frac{Q_{ij}}{4} + \sum_{i=1}^{n} \frac{c_i}{2}.$$