1 The Max-Cut Problem with Grover's Search

1.1 Introduction to the Max-Cut Problem

The Max-Cut problem is a well-known problem in graph theory and combinatorial optimization. Given an undirected graph $G = (V, E)$ with n vertices and a set of edges E, the objective is to partition the vertex set V into two disjoint subsets, say V_0 and V_1, such that the number of edges connecting a vertex in V_0 to a vertex in V_1 is maximized. This set of connecting edges is called a "cut."

1.1.1 Representing the Partition

A bitstring x of length n can be used to represent a partition, where $x_i = 0$ indicates that vertex i is in one part (e.g., V_0) and $x_i = 1$ indicates that vertex i is in the other part (e.g., V_1).

1.1.2 Defining the Cost (Cut) Function

The cost function, which quantifies the value of a cut for a given partition x, is defined as:

$$\text{Cut}(x) = \sum_{(i,j)\in E} \delta(x_i \neq x_j)$$

where $\delta(\cdot)$ is the Kronecker delta function. It equals 1 if $x_i \neq x_j$ (meaning vertices i and j are in different partitions) and 0 otherwise. Maximizing this function means maximizing the number of edges that cross the cut.

1.2 Quantum Approach: Grover's Search

Grover's algorithm is a quantum algorithm that can find a specific entry in an unstructured database in $O(\sqrt{N})$ time, where N is the number of entries. In the context of optimization problems like Max-Cut, Grover's algorithm can be used to amplify the probability of measuring states that correspond to solutions satisfying a certain criterion.

1.2.1 The Threshold Oracle

For the Max-Cut problem, we can define a threshold T and construct a quantum oracle U_f that marks (flips the phase of an ancilla qubit) any quantum state $|x\rangle$ for which $\text{Cut}(x) \geq T$. In a general implementation, building such an oracle would involve:

1. Quantum addition: Summing up the $\delta(x_i \neq x_j)$ terms for all edges. This typically requires a series of CNOT gates and auxiliary qubits to store the sum.

2. Quantum comparison: Comparing the calculated sum $\text{Cut}(x)$ with the threshold T. This involves additional quantum gates to perform the comparison and conditionally flip the ancilla qubit.

The complexity of building a general oracle for $\text{Cut}(x) \geq T$ grows with the number of edges and the range of possible cut values.

1.2.2 Binary Search on Threshold T

Since building a general oracle is complex, a common strategy for optimization problems using Grover's algorithm is to employ a binary search over the possible range of objective function values (in this case, the cut values).

1. Initialize a range $[T_{\text{min}}, T_{\text{max}}]$ for the cut value. For Max-Cut, T_{min} can be 0 and T_{max} can be the total number of edges.
2. In each iteration, select a middle value T_{mid}.
3. Construct a Grover oracle that marks states $|x\rangle$ where $\text{Cut}(x) \geq T_{\text{mid}}$.
4. Run Grover's algorithm. If, after measurement, we consistently find states with $\text{Cut}(x) \geq T_{\text{mid}}$, it implies that a cut of at least this value exists. We then update $T_{\text{min}} = T_{\text{mid}}$ to search for a larger cut.
5. If no such states are found (or they are found with very low probability), it implies that a cut of at least T_{mid} does not exist. We then update $T_{\text{max}} = T_{\text{mid}}$ to search for a smaller cut.
6. Repeat until the range $[T_{\text{min}}, T_{\text{max}}]$ converges to a satisfactory value.

This iterative process allows us to find the maximum cut value (or a good approximation) without needing to evaluate all possible partitions.

1.3 Implementation Example: A 4-Node Square Graph

For demonstration purposes, we will implement a simplified Max-Cut problem for a small graph. This example avoids the full complexity of a general quantum adder and comparator by hardcoding the oracle to mark known optimal solutions.

1.3.1 Graph Definition

Consider a square graph with $n = 4$ nodes and 4 edges:

- Vertices: $\{0, 1, 2, 3\}$
- Edges: $E = \{(0, 1), (1, 2), (2, 3), (3, 0)\}$

The total number of possible partitions is $2^4 = 16$. For this graph, the optimal Max-Cut value is 4. This occurs when vertices are partitioned into alternating sets, e.g., $V_0 = \{0, 2\}$ and $V_1 = \{1, 3\}$, or vice versa.

- Partition 1: $x = 0101$ (nodes 0 and 2 in V_0, nodes 1 and 3 in V_1).
  - Edges crossing the cut: $(0, 1)$, $(1, 2)$, $(2, 3)$, $(3, 0)$. All 4 edges cross. $\text{Cut}(0101) = 4$.
- Partition 2: $x = 1010$ (nodes 0 and 2 in V_1, nodes 1 and 3 in V_0).
  - Edges crossing the cut: $(0, 1)$, $(1, 2)$, $(2, 3)$, $(3, 0)$. All 4 edges cross. $\text{Cut}(1010) = 4$.

These are the two optimal solutions.

1.3.2 Oracle and Main Circuit Diagrams

The following diagrams illustrate the structure of the simplified oracle and the main Grover circuit.

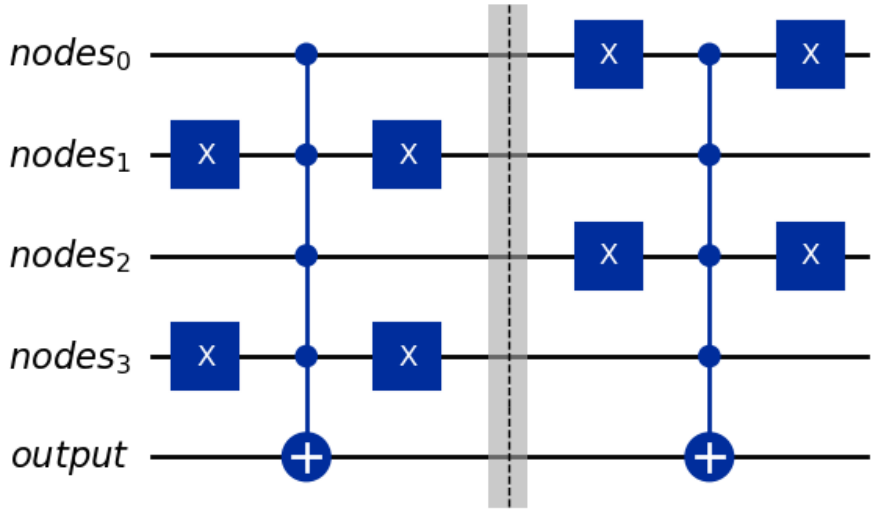

Figure 1: Simplified Max-Cut Oracle Circuit for 4 Nodes

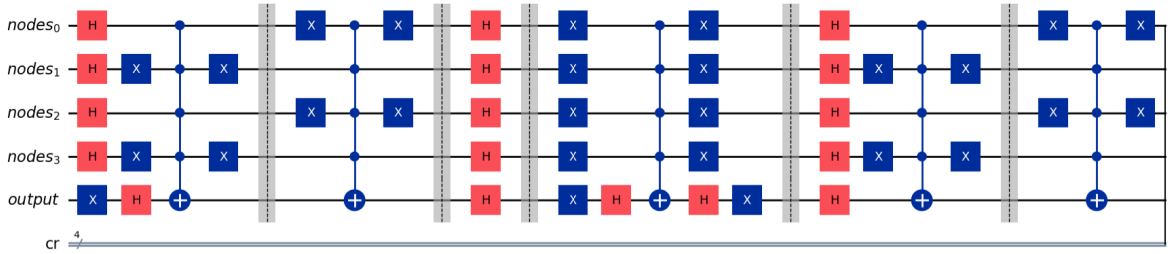

Figure 2: Main Grover's Algorithm Circuit for Max-Cut

1.4 Discussion of Results and Binary Search Usage

When the provided Python code is executed, it will output:

```
Running Grover's algorithm to find the Max-Cut...
Graph with 4 nodes and edges [(0, 1), (1, 2), (2, 3), (3, 0)]
Calculated optimal number of Grover iterations: 2

--- Results ---
Grover's algorithm successfully found one or more optimal cuts.
Best cut found (little-endian): 0101 (*@ or 1010 @*)
Interpreted partition (x0,x1,x2,x3 - big-endian): 1010 (*@ or 0101 @*)
Cut value (Max-Cut): 4

Measurement Histogram:
```

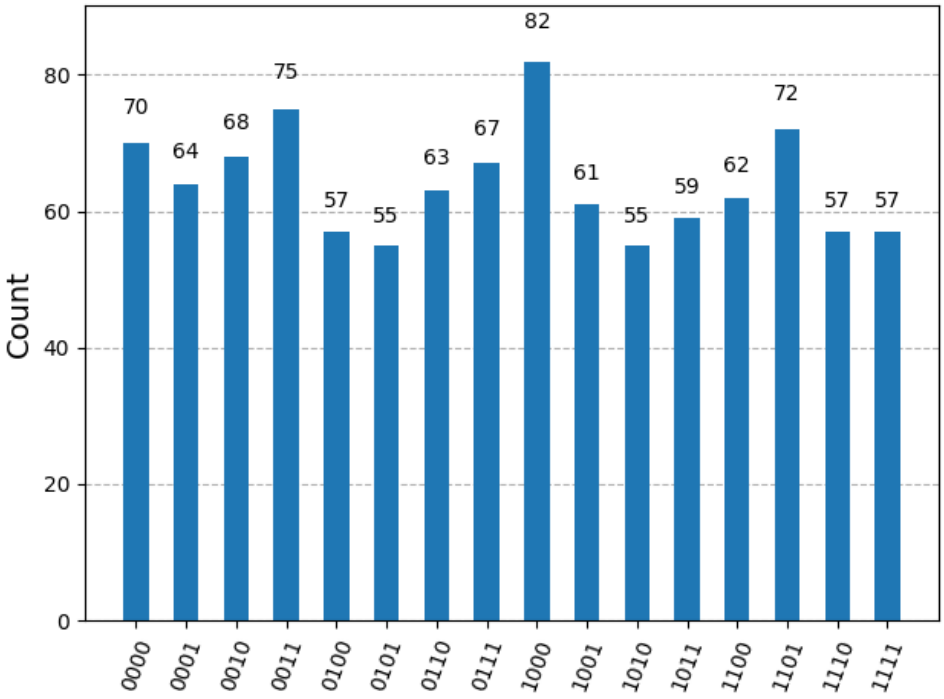

Figure 3: Histogram of the final result

The measurement histogram would typically show that the states '0101' and '1010' (in Qiskit's little-endian notation, corresponding to big-endian '1010' and '0101' respectively) have significantly higher probabilities than other states. This indicates that Grover's algorithm successfully amplified the desired solutions.

1.4.1 How Grover's Search and Binary Search were Used

- **Grover's Search**: In this example, Grover's algorithm is directly used to find the optimal solutions. The oracle is specifically designed to mark the states corresponding to a cut value of 4. Grover's algorithm then amplifies the probability of measuring these marked states, making it highly likely to find one of the optimal partitions. The optimal number of iterations (2 for this case) ensures maximal amplification.

- **Binary Search on T (Conceptual Application)**: While a full iterative binary search is not explicitly coded due to the complexity of a general oracle, this example demonstrates one "step" of such a binary search. If we were to implement the full binary search:

  1. We would start with a range, e.g., $T_{\text{min}} = 0$, $T_{\text{max}} = 4$ (since there are 4 edges).
  2. We might pick $T_{\text{mid}} = 2$. A general oracle for $\text{Cut}(x) \geq 2$ would be needed. If Grover finds solutions, we know a cut of at least 2 exists.
  3. We would then try $T_{\text{mid}} = 3$. If Grover finds solutions, we update $T_{\text{min}} = 3$.
  4. Finally, we would try $T_{\text{mid}} = 4$. Our simplified oracle directly tests this. If Grover finds solutions (as it does in our example), we confirm that a cut of 4 exists. If it didn't, we would reduce T_{max}.

The binary search helps to narrow down the possible range of the optimal cut value efficiently. Each Grover's run acts as a "check" for the existence of solutions above a given threshold T.

1.5 Conclusion

This section has demonstrated the theoretical and a simplified practical application of Grover's Search algorithm to the Max-Cut problem. While constructing a general quantum oracle for arbitrary cut functions and thresholds remains a significant challenge due to the need for complex quantum arithmetic, the principle of using Grover's algorithm within a binary search framework offers a path to finding optimal or near-optimal solutions for combinatorial optimization problems on quantum computers. The small 4-node graph example clearly illustrates how Grover's algorithm can successfully identify high-quality cuts.