Implementation of Grover's Algorithm

Project Report

Submitted in Fulfillment of the Requirements
for the
Completion of the

ECS417: Quantum Computer Science

by

Rohan Mehra
Indian Institute of Science Education and Research (IISER)
`rohan21@iiserb.ac.in`

Under the supervision of

Dr. Ankur Raina

Indian Institute of Science Education and Research (IISER)

Contents

Grover's Algorithm

Grover's algorithm is a quantum algorithm designed for searching an unsorted database or an unstructured list of items more efficiently than classical algorithms.

- **Classical Approach:** The best a classical algorithm can do is to search through the list sequentially, requiring $O(N)$ queries in the worst case.
- **Grover's Algorithm:** It reduces the number of required queries to $O(\sqrt{N})$, providing a quadratic speedup.

Circuit Diagram

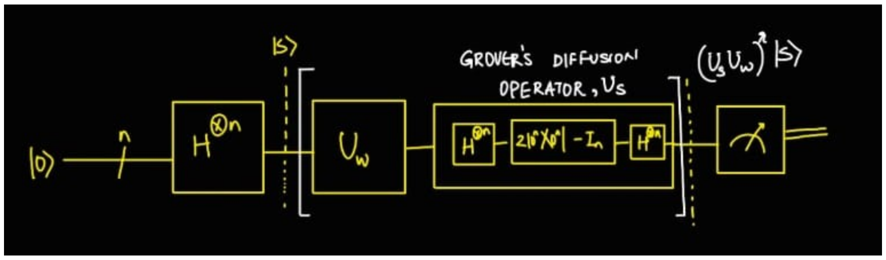

Figure 1: Circuit diagram of Grover's Algorithm

The state $|\omega\rangle$ represents the marked state that Grover's algorithm aims to find. The state $|w^{\perp}\rangle$ all n-bit strings except for a specific marked state $|w\rangle$. It is given by:

$$|w^{\perp}\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq w} |x\rangle,$$

.

1. Initialization State $|s\rangle$

The initialization state $|s\rangle$ is the equal superposition state over all possible n-bit strings. It is defined as:

$$|s\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle,$$

where $H^{\otimes n}$ represents the application of the Hadamard gate to each qubit, and $N = 2^n$ is the total number of possible n-bit states.

$$|s\rangle = \cos\left(\frac{\theta}{2}\right)|w^{\perp}\rangle + \sin\left(\frac{\theta}{2}\right)|w\rangle,$$

or equivalently:

$$|s\rangle = \sqrt{\frac{N-1}{N}}|w^{\perp}\rangle + \sqrt{\frac{1}{N}}|w\rangle,$$

where θ is the angle between the initial state $|s\rangle$ and the marked state $|w\rangle$. An equal superposition state allows a quantum system to represent and explore all possible solutions simultaneously. In classical computing, searching for a specific item among N possible items requires examining each one individually, leading to a worst-case complexity of O(N). In contrast, by starting with an equal superposition, quantum algorithms can leverage quantum parallelism to process all N possibilities at once.

2. Oracle Operator U_w

The Oracle U_w acts as a quantum version of a function that checks whether an input is the desired solution (the "marked" state). Specifically, it flips the sign (or phase) of the amplitude of the quantum state corresponding to the correct solution $|w\rangle$, while leaving the amplitudes of all other states unchanged. .The oracle U_w marks the state $|w\rangle$ by applying a phase shift:

$$U_w|x\rangle = (-1)^{f(x)}|x\rangle,$$

where $f(x)$ is a function such that:

$$U_w|x\rangle = \begin{cases} -|x\rangle & \text{if } f(x) = 1, \text{ i.e., } x = w, \\ |x\rangle & \text{if } f(x) = 0, \text{ i.e., } x \neq w. \end{cases}$$

In matrix form:

$$U_w = I - 2|w\rangle\langle w|.$$

3. Grover's Diffusion Operator U_s

The Grover diffusion operator U_s amplifies the amplitude of the marked state.while decreasing the amplitudes of other states. This process helps to make the correct solution more likely to be measured after several iterations,r . It is defined as:

$$U_s = 2|s\rangle\langle s| - I,$$

where I is the identity matrix.

Expanded, this operator can be written as:

$$U_s = 2\left(\frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} |x\rangle\langle x|\right) - I.$$

Alternatively, using the Hadamard transformation:

$$U_s = H^{\otimes n}\left(2|0\rangle^{\otimes n}\langle 0|^{\otimes n} - I\right)H^{\otimes n}.$$

4. Probability of Measuring the Marked State

The probability $P(\omega, r)$ of measuring the marked state $|\omega\rangle$ after r iterations of Grover's algorithm is given by:

$$P(\omega, r) = |\langle\omega|(U_s U_\omega)^r|s\rangle|^2.$$

This probability can be approximated by:

$$P(\omega, r) \approx \sin^2\left(\theta\left(\frac{1}{2} + r\right)\right),$$

where θ is the angle between $|s\rangle$ and $|\omega\rangle$. For large r, $P(\omega, r)$ approaches 1, indicating a high likelihood of measuring the marked state.

At the start of Grover's algorithm:

$$P(\omega, 0) = |\langle\omega|s\rangle|^2 = \frac{1}{N},$$

which simplifies to:

$$P(\omega, 0) = \sin^2\left(\frac{\theta}{2}\right).$$

For small θ, we use the approximation:

$$\sin\left(\frac{\theta}{2}\right) \approx \frac{\theta}{2}.$$

The inner product between $|s\rangle$ and $|\omega\rangle$ is:

$$\langle\omega|s\rangle = \frac{1}{\sqrt{N}} = \sin\left(\frac{\theta}{2}\right).$$

For small angles:

$$\langle\omega|s\rangle \approx \frac{1}{\sqrt{N}}.$$

5. Optimal Number of Iterations

To maximize $P(\omega, r)$, the argument of the sine function $\theta\left(\frac{1}{2} + r\right)$ must be $\frac{\pi}{2}$. Solving for r:

$$\theta\left(\frac{1}{2} + r\right) = \frac{\pi}{2},$$

$$r = \frac{\pi}{2\theta} - \frac{1}{2}.$$

For large N, $\theta \approx \frac{1}{\sqrt{N}}$, so:

$$r \approx \frac{\pi}{4}\sqrt{N}.$$

Thus, the maximum number of iterations is approximately:

$$r_{\max} \approx \frac{\pi}{4}\sqrt{N}.$$

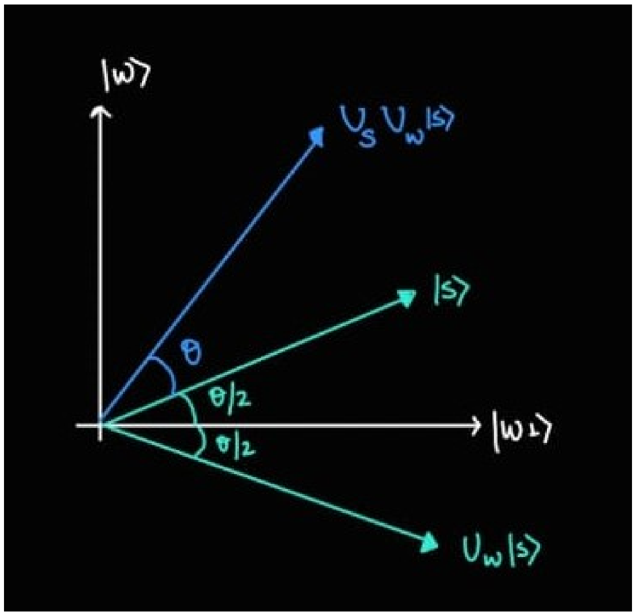

Figure 2: Geometric Representation of Grover's Circuit

Problem Statement

We aim to implement Grover's Algorithm for efficient quantum search within a set of 10-bit strings. Consider a search space $\mathcal{X}$ consisting of all possible 10-bit strings, where the total number of elements is given by:

$$|\mathcal{X}| = 2^{10} = 1024.$$

Our goal is to identify a set of 10 marked strings $\{x_1, x_2, \ldots, x_{10}\} \subset \mathcal{X}$ such that an oracle function $f : \mathcal{X} \to \{0, 1\}$ is defined as:

$$f(x) = \begin{cases} 1 & \text{if } x \text{ is one of the marked strings,} \\ 0 & \text{otherwise.} \end{cases}$$

The problem requires using Grover's Algorithm to find these 10 marked strings with high probability. Specifically, Grover's Algorithm should amplify the amplitude of the marked strings, allowing us to identify them efficiently compared to a classical brute-force search, which would require $O(1024)$ evaluations. We aim to achieve this with fewer oracle calls, leveraging the quadratic speedup provided by Grover's Algorithm.

Objective

The objective is to design and implement a quantum circuit that utilizes Grover's Algorithm to search for the 10 marked strings in the 10-bit search space. The implementation should ensure that the probability of successfully identifying all marked strings is maximized, demonstrating the quantum advantage in search problems.

1 Quantum Search for 10 Marked Strings

In this problem, the search space consists of all 10-bit strings, which leads to a total of $N = 1024$ possible strings. The goal is to find a set of 10 marked strings from this search space. The oracle function $f(x)$ is defined such that it returns 1 if x is one of the marked strings, and 0 otherwise.

We implement Grover's Algorithm using the PennyLane library, which is a framework for quantum machine learning. The quantum circuit is designed to search for the marked strings efficiently by applying Grover's Algorithm iteratively. The algorithm works by repeatedly applying the oracle and diffusion operator.

2 Quantum Circuit Implementation

The quantum circuit is implemented using PennyLane, and the 10 marked strings are represented in the matrix ω. The following Python code demonstrates the implementation of Grover's Algorithm for the search problem: To access the Python code, please follow this link: Colab Notebook.

The states with higher probabilities represent the marked strings. In this case, each of the states shown has the same probability, indicating that Grover's Algorithm is amplifying the correct solutions. With enough iterations, the algorithm will concentrate the probability on the marked states.

3 The 3-SAT Problem

In this project, we focus on a 3-SAT formula with 3 variables. A 3-SAT formula consists of several clauses, each containing three literals. A literal is a variable or its negation.

For the formula:

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$$

the goal is to find values of x_1, x_2, and x_3 that make the formula `True`.

4 Grover's Algorithm for Solving 3-SAT

Grover's algorithm is designed to search through an unsorted list of possibilities to find a solution. The algorithm works by applying an oracle that marks the correct solutions and then applying a diffusion operator that amplifies the probability of these solutions.

4.1 Oracle

In our case, the oracle is a function that flips the sign of the amplitude of a quantum state if the corresponding assignment satisfies the SAT formula. For a 3-SAT problem with n variables, we create an oracle matrix that checks each assignment to see if it satisfies the formula.

The oracle for the given SAT formula is constructed using the following steps:

- Iterate through all possible assignments of x_1, x_2, and x_3.
- For each assignment, check if it satisfies the formula.
- If the assignment satisfies the formula, flip the sign of the corresponding state in the quantum register.

4.2 Diffusion Operator

The diffusion operator is used to amplify the probability amplitude of the correct states. It inverts the average of the amplitudes and is applied to all quantum states after each oracle application.

5 Implementation

The Python code provided implements Grover's algorithm for solving the 3-SAT problem. Below is a summary of its key components:

5.1 SAT Formula Function

The function `sat_formula(x1, x2, x3)` returns the result of applying the SAT formula to a specific assignment of x_1, x_2, and x_3. It returns `True` if the formula is satisfied, otherwise `False`.

5.2 Oracle Matrix Creation

The `create_oracle_matrix` function constructs an oracle matrix based on the SAT formula. The matrix is initialized with 1's, and if an assignment satisfies the formula, the corresponding diagonal entry is set to -1, marking it as a solution.

5.3 Diffusion Operator

The `diffusion_operator` function creates the diffusion operator matrix, which helps amplify the probability of the correct solutions.

5.4 Grover's Algorithm

The function `grover_sat_solver` performs Grover's algorithm. It initializes the quantum register in an equal superposition state using Hadamard gates, applies the oracle matrix and the diffusion operator for a calculated number of steps, and measures the probability distribution of the states.

5.5 Solution Verification

After running the quantum circuit, the function `verify_solution` checks if the measured solution satisfies the original SAT formula.

6 Results

The algorithm successfully finds a solution to the 3-SAT problem. The output includes the found solution, the probability of that solution, and a list of all possible valid solutions for verification.

The following results were observed for the given SAT formula:

- Found solution: `111`
- Probability: 0.667
- Valid solutions: `111, 101, 011`

7 Timing Information

The performance of the algorithm was measured in terms of execution time for the oracle creation, the solution finding process, and the total execution time.

- Oracle creation time: 0.002 seconds
- Solution finding time: 0.036 seconds
- Total execution time: 0.038 seconds

Grover's algorithm successfully solves the 3-SAT problem, providing a quantum solution to a classical problem. The algorithm's efficiency increases with the number of variables and clauses, and it offers significant speedup compared to classical brute force methods.

8 Conclusion

Grover's Algorithm provides a quadratic speedup over classical search algorithms for the unstructured search problem. In this project, we successfully implemented Grover's Algorithm using the PennyLane library to search for 10 marked 10-bit strings from a search space of 1024 possible states. The quantum circuit demonstrated the ability to amplify the probability of the marked states, showcasing the quantum advantage in solving search problems. Further optimization and more iterations can lead to even better results, and the quantum advantage becomes more apparent as the size of the search space increases.

References

[1] Grover's Algorithm-PennyLaneAI. [Online]. Available: https://pennylane.ai/qml/demos/tutorial_grovers_algorithm/

[2] Medium, "Explaining Grover's Algorithm with a Colony of Ants: A Pedagogical Model for Making Quantum Technology Comprehensible," Quantum Algorithms Untangled, [Online]. Available: https://medium.com/quantum-untangled/grovers-algorithm-quantum-algorithms-untangled-bdf15c8ccfab

[3] XanaduAI, "PennyLane Grover's Algorithm Tutorial," GitHub, [Online]. Available: https://github.com/XanaduAI/PennyLane-YouTube-Tutorials/blob/main/grovers_algorithm.ipynb

[4] A fast quantum mechanical algorithm for database search. [Online]. Available: https://www.clausiuspress.com/assets/default/article/2020/12/04/article_1607101207.pdf

[5] Grover's Algorithm: Quantum Database Search. [Online]. Available: https://arxiv.org/pdf/quant-ph/9605043

[6] Polynomial Quantum Speedups for Constraint Satisfaction Problems. [Online]. Available: https://qibo.science/qibo/stable/code-examples/tutorials/grover3sat/README.html

[7] "Exactly 1-3-SAT Problem Grover's Algorithm," LinkedIn, [Online]. Available: https://www.linkedin.com/pulse/exactly-1-3-sat-problem-grovers-algorithm-breaking-rules-porfiris/

[8] "Exact 1-3-SAT Problem Grover's Algorithm," ScienceDirect, [Online]. Available: https://pdf.sciencedirectassets.com/272574/1-s2.0-S0022000006X01782/1-s2.0-S0022000006001012/main.pdf

[9] Qibo, "Grover's Algorithm for 3-SAT," [Online]. Available: https://qibo.science/qibo/stable/code-examples/tutorials/grover3sat/README.html

[10] Classiq, "Grover's Algorithm for 3-SAT," [Online]. Available: https://docs.classiq.io/latest/explore/algorithms/grover/3_sat_grover/3_sat_grover/

[11] Stanford, "Quantum Computation and Information," CS269Q Projects 2019, [Online]. Available: https://cs269q.stanford.edu/projects2019/Dalal_Y.pdf

[12] IEEE Xplore, "Optimized Grover's Search for Constraint Satisfaction," [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9764017

[13] Qiskit, “Quantum Algorithms: Using Grover’s Algorithm,” [Online]. Available: https://notebook.community/antoniomezzacapo/qiskit-tutorial/community/aqua/optimization/grover

[14] K. C. Wang, H. G. L. Reuter, and M. J. L. O’Neill, “Quantum Algorithm Implementations for Beginners,” ACM, [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3517340