GROVER'S ALGORITHM FOR MULTI-TARGET SEARCHING AND CONSTRAINT SATISFACTION PROBLEMS

Yunlun Li
yunlun.li@rutgers.edu

Shuohao Ping
sp1831@scarletmail.rutgers.edu

December 23, 2022

ABSTRACT

Grover's Algorithm is a well-known Quantum Searching Algorithm. In this project, we discussed the situation of searching multiple target states using Grover's Algorithm and its application in solving the Constrained Satisfaction Problem by comparing the result on real Quantum computers and Simulators. Our project program takes input as a DIMACS CNF file and outputs a possible solution of the input logic expression.
source code: `https://github.com/ping-shuo-hao/n_qubits_grover_algorithm.git`

***K*eywords** Quantum Computing · Grover's Algorithm · Constraint Satisfaction Problem

1 INTRODUCTION

1.1 Motivation/Significance/Background

Grover's algorithm is a quantum searching algorithm that provides quadratic speedup for searching unstructured data. For any classical algorithm, it would take $\frac{n}{2}$ searches on average to search for an element from a list of unstructured data with size n. In contrast, Grove's algorithm has a time complexity of $O(\sqrt{n})$ according to [7]. Although this speedup is very moderate compared to Shor's algorithm, it has more applications. For example, in the article[7], the author mentioned that Grover's algorithm could be used as the subroutine for any classical algorithm with some search component, such as finding shortest paths and minimum spanning trees, various other graph algorithms, etc. Also, it can speed up some NP-complete problems, like the satisfiability problem. However, the speedup is only quadratic rather than exponential, so we still can't solve the NP problems in polynomial time. To achieve the quadratic speedup, Grover's algorithm uses amplitude amplification to increase the probability amplitude of the target state, which increases the probability of measuring the target state. In the case of searching one target state, Grover's algorithm will need $O(\sqrt{n})$ amplifications before measuring the final state. Firstly, we will set all qubits to equal superposition with probability amplitude $\frac{1}{n}$ for each state. Then, we can use amplitude amplification. There are two steps for amplification: inverse the phase of the target state and rotate all states around the mean. We use an oracle to invert the phase of the target state from $\frac{1}{n}$ to $-\frac{1}{n}$. Then, we apply a diffuser to rotate all states around the mean. Since we assume n is large, the mean will remain close to $\frac{1}{n}$ if we only inverse the phase of a single state. And rotation will change the phase of the target state to $\frac{2}{n}$. Repeating this process at most $O(\sqrt{n})$ times, we can measure the final state and obtain the target state as the measurement result with high probability.

1.2 Key Idea/Problem Addressed/Limitation of Prior Approaches

In Grover's paper[3], the author stated that the problem is to find an input x of a function f such that $f(x) = 1$. It is assumed that function $f(x) \in 0, 1$ for all $0 \leq x \leq N - 1$ where $N = 2^n$ for some arbitrary natural number n has exactly one input x such that $f(x) = 1$. This assumption limits the possible choice of f. In general, there can be multiple inputs that have an output 1 for f. Therefore, our first project idea is to explore a more general case for Grover's algorithm, which is to study the result of running Grover's algorithm on functions with multiple solutions. Specifically,

we want to run Grover's algorithm on function like $f(x) = \begin{cases} 1 & \text{if } x = x_1, x_2, ... \\ 0 & \text{otherwise} \end{cases}$. In Figgatt et al.'s article[2], Figgatt built the quantum circuits for both one-solution and two-solution cases and ran the program on a programmable quantum computer. Therefore, it is possible to design quantum circuits for functions with multiple solutions, but this paper only demonstrates this for the two-solution case. In this project, we want to extend the two-solution case into more general cases. In particular, we want to examine the output of Grover's algorithm when the number of target states is less than, equal to, or greater than $\frac{N}{2}$, and we want to run the algorithm for the 4-qubit case on a programmable quantum computer and compared the results with our simulation results. As we discussed in section 1.1, one important assumption in Grover's paper is that the mean doesn't change much if we only invert the phase of a single state. However, if we invert too many states, we might change the mean significantly, and the diffuser will start to fail as we increase the number of target states. In addition, we have another project idea that explores using Grover's algorithm to solve Boolean Satisfiability Problem(SAT). Since Qiskit already implements this in their library, we will study their algorithm for generating the oracle for Grover's algorithm given a Boolean expression.

1.3 Team member roles

Shuohao Ping:
1. Designing quantum oracles for multiple solution cases.
2. Implementing Grover's algorithm for searching multiple target states.
3. Preparing presentation slides and final report(Part I).

Yunlun Li:
1. Designing quantum oracles for Boolean expressions.
2. Implementing Grover's algorithm for solving constraint satisfaction problems.
3. Preparing presentation slides and final report(Part II).

2 PART I: GROVER'S ALGORITHM FOR MULTIPLE TARGETS

2.1 Technical Implementation

For part one of this project, we examine how Grover's algorithm performs when searching for multiple target states. We implement Grover's algorithm using IBM Qiskit. The reason for choosing Qiskit is that it not only allows us to do simulations, but also gives us a chance to run Grover's algorithm on a real quantum computer using IBM Quantum. Please refer to file `general_n_qubits_grover_algorithm.py` for our implementation.

2.1.1 Prior Work

In the Qiskit textbook[6], there are two examples of implementing Grover's algorithm. One implementation is for two qubits, and another implementation is for three qubits. However, in both examples, they only show the circuit for searching a specific state without an explanation of a general rule for designing the oracle, and their design can't extend to a more general case when we have more than three qubits in the circuit. Also, their implementation only supports searching a single state, while I need to search for more than one target state. Therefore, our implementation is not based on the two examples in the Qiskit textbook. Rather, it is based on the observation of circuit design in Figgatt et al.'s paper[2].

2.1.2 My Work

Our implementation is a generalization of previous works. The goal of our implementation is to generate a quantum circuit given the number of qubits and a list of target states. The key challenge is to design the oracle for phase inversion because most of the resources about Grover's algorithm only provide examples for searching a single state. Their way of designing oracles is to decompose the matrix into gates. However, this approach is hard to generalize because as the number of qubits increases, the size of the matrix grows exponentially, and it will be hard to decompose the matrix due to its large size. We need to find a general rule for designing the oracle. To address this problem, we first try to find a general pattern for oracles inverting a single state. In Figgatt et al.'s paper[2], the author list all possible oracles for inverting a single state of 3 qubits. By observing these oracles, I found the following rule: the oracles are always symmetrical, and they always have a Controlled-Z gate or a multi-Controlled-Z gate between X gates. We will put two X gates to the qubit whose target state value is 0. For example, when we want to search the state |0010>, we will put two X gates on the first, the third, and the fourth qubit, and put a controlled-controlled-controlled-Z gate between two

sets of X gates. Here, the first digit represents the fourth qubit in the circuit, and the last digit represents the first qubit. The following graph is the oracle for searching |0010>:

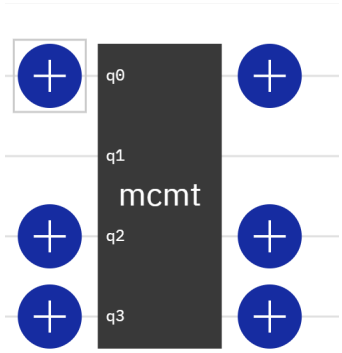

Fig 1. This is the oracle for inverting the phase of the state |0010>. The blue circle with a plus sign represents an X gate, and the MCMT gate is a controlled-controlled-controlled-Z gate.

This rule can easily extend to a more general case when we have more than three qubits. And we will build our oracle for multiple target states by combining all oracles for a single target state. For example, when we want to invert the phase of |0000> and |0010>, we just put the oracle for |0000> and |0010> together to build a new oracle for inverting the phase of these two states. The following graph is the oracle circuit for inverting the phase of |0000> and |0010>:

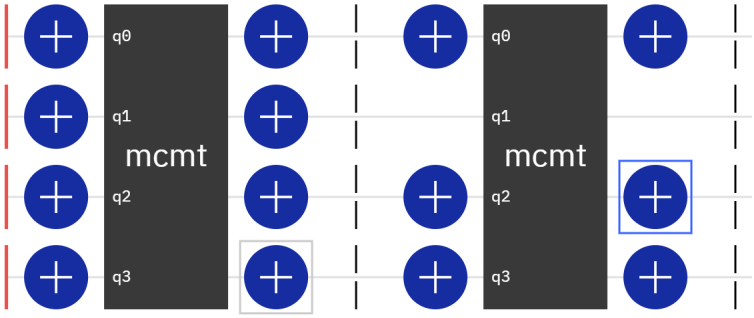

Fig 2. This is the oracle for inverting the phase of the state |0000> and |0010>. The gates before the center dashed line are the oracle for inverting the phase of |0000>, and the gates after the center dashed line are the oracle for inverting the phase of the state |0010>.

We can do this because the matrix for inverting the phase of multiple target states is the product of all matrices inverting a single target state. However, there is a limitation to this method. The problem is that we have redundant gates in our circuit. In the above example, the two X gates for qubits 0,2,3 between the two MCMT gates are redundant because two consecutive X gates have no effect on the qubit. This redundancy won't affect the simulation result. Still, it will affect the result on a real quantum computer since the gates are not perfect in reality, and we should use as few gates as possible to reduce the effect of quantum noise.

2.2 Evaluation

We will conduct two experiments. The first experiment is discovering how Grover's algorithm performs when searching for multiple target states in an ideal setting. We will use quantum simulation to exclude the effect of quantum noise. The second experiment is to run Grover's algorithm on a real quantum computer to examine how quantum noise affects the measurement result in reality.

2.2.1 Experiment Setup

For the first experiment, we will use four qubits in our quantum circuit, and we will run the simulation for searching from 3 target states to searching 15 target states. We will ignore the case for searching zero and sixteen target states because these searches will always output uniform distribution. We also ignore the case of searching for one target state because we already know Grover's algorithm will work for searching a single state. For each search, we will run the amplification only once and sample 1000 times to get the distribution.

For the second experiment, we will also use four qubits in our circuit. We will search four target states: |0101>,|0111>,|1101>, and |1111>. The reason for us choosing these four states is that when we search one-fourth of all possible states, we can obtain exact results with zero probability measuring non-target states with only one search, and the oracle for searching these four states is the simplest, which only requires a controlled-Z gate on qubits 0 and 2. We want to control the number of gates in our circuit to the minimum so that the quantum computer won't just return a uniform distribution due to quantum noise, and we can still see a similar trend between simulation results and results from the quantum computer. We will run this experiment on IBM Quantum using both simulation and the quantum computer. For each search, we will use only one amplification and sample 1024 times to get the distribution of the measurement result.

2.2.2 Results and Conclusion

Firstly, we will examine the simulation result of the first experiment. The following six plots are the simulation results from searching two target states to searching seven target states. The X-axis of all plots is the 16 possible states for four qubits, and the Y-axis is the number of times measuring that state from simulation.

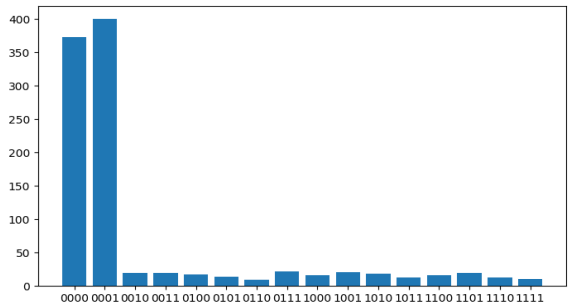

Fig 3. This is the distribution of searching the first two states: |0000> and |0001>.

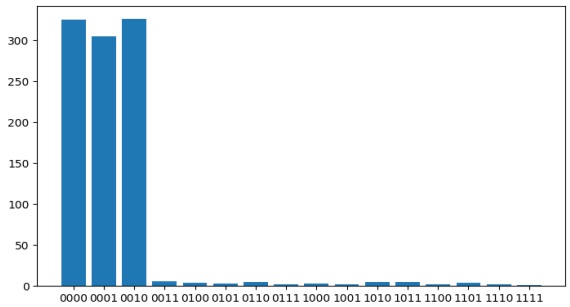

Fig 4. This is the distribution of searching the first three states.

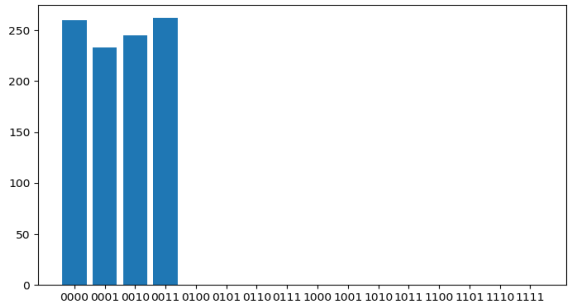

Fig 5. This is the distribution of searching the first four states.

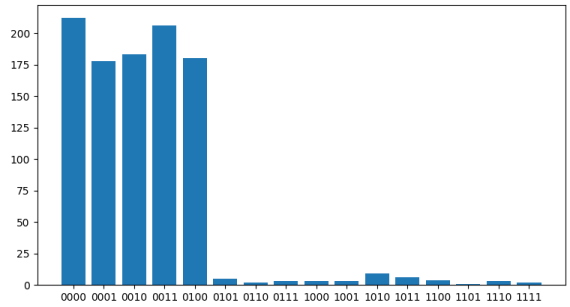

Fig 6. This is the distribution of searching the first five states.

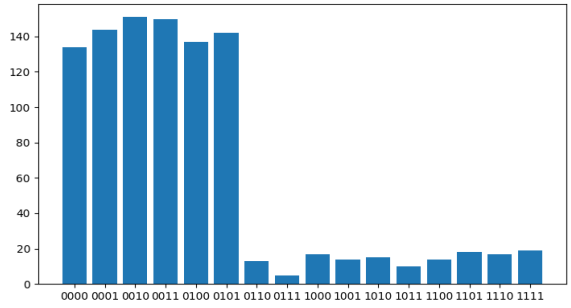

Fig 7. This is the distribution of searching the first six states.

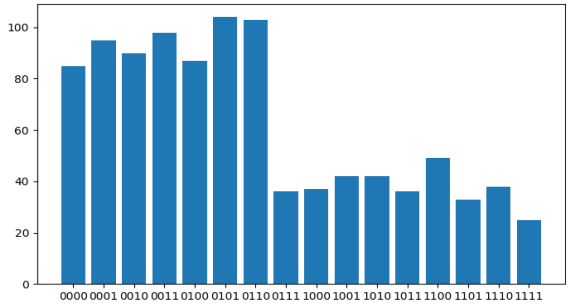

Fig 8. This is the distribution of searching the first seven states.

From the above six plots, we can see that Grover's algorithm still works properly and amplify the probability of all target states. Also, there is a special case when we search four target states. In this case, the probability of measuring non-target states is 0, and we are guaranteed to measure one of the target states. One way to explain this is when we invert the phase of 4 states, the phase of all target states changes to $-\frac{1}{4}$, while the phase of all non-target states remains $\frac{1}{4}$. Thus, the mean becomes $(-\frac{1}{4} * 4 + \frac{1}{4} * 12)/16 = \frac{1}{8}$. When we rotate the phase of non-target states around the mean, the phase of non-target states changes to 0. Actually, based on the paper[1], it is proved that this is always the case when searching one-fourth of the total possible states for any number of qubits. In general, Grover's algorithm still works when the number of target states is strictly less than half of all states. This is because the mean remains positive after we invert the phase of target states.

Now, we will see what happens when we search exactly half of all states. The following plot is the distribution of searching the first eight states using simulation. The plot's X-axis is the 16 possible states for four qubits, and the Y-axis is the number of times measuring that state from the simulation.

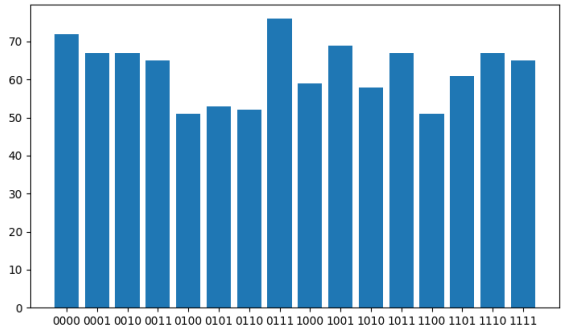

Fig 9. This is the distribution of searching the first eight states.

The above plot looks like a random distribution. Grover's algorithm fails to amplify the probability amplitude of the first eight states(from |0000> to |0111>). This is because when we invert the phase of half of all states, the mean will change to 0. When the diffuser rotates the phase about the mean, the phase of target states changes from $-\frac{1}{4}$ to $\frac{1}{4}$, while the phase of non-target states changes from $\frac{1}{4}$ to $-\frac{1}{4}$. Since the probability of measuring a state is the square of its probability amplitude, we will measure all states with equal probability $\frac{1}{16}$. $((-\frac{1}{4})^2 = (\frac{1}{4})^2 = \frac{1}{16})$

Finally, when we start to amplify more than eight states, Grover's algorithm starts to amplify the probability amplitude of non-target states. The following seven plots are the simulation results from searching nine target states to searching fifteen target states. The X-axis of all plots is the 16 possible states for four qubits, and the Y-axis is the number of times measuring that state from simulation.

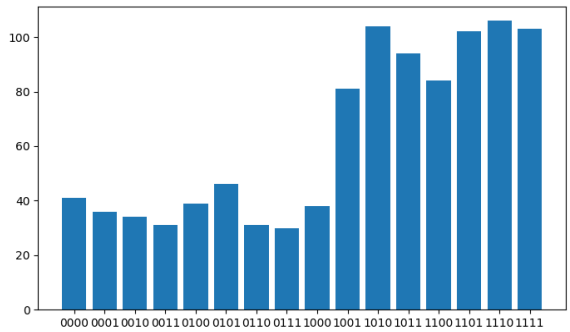

Fig 9. This is the distribution of searching the first nine states.

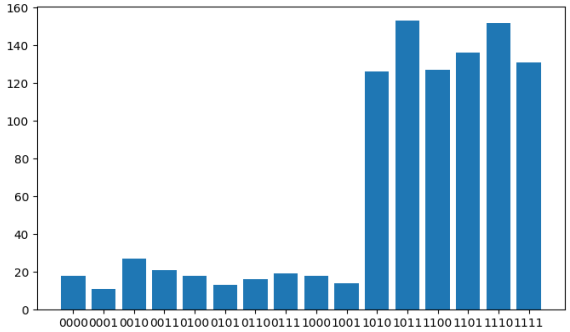

Fig 10. This is the distribution of searching the first ten states.

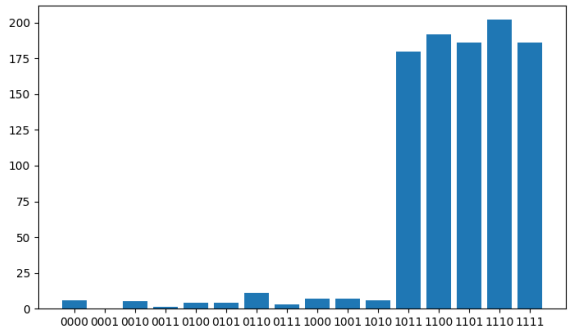

Fig 11. This is the distribution of searching the first eleven states.

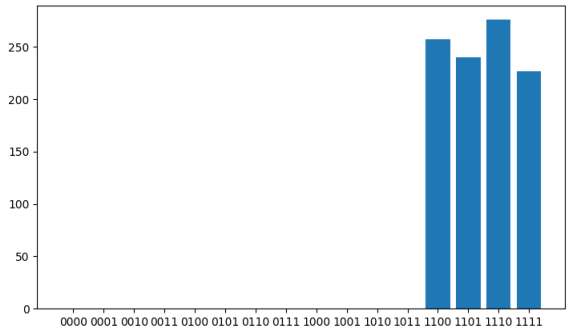

Fig 12. This is the distribution of searching the first twelve states.

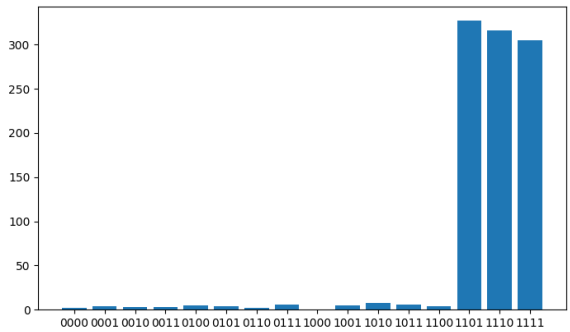

Fig 13. This is the distribution of searching the first thirteen states.

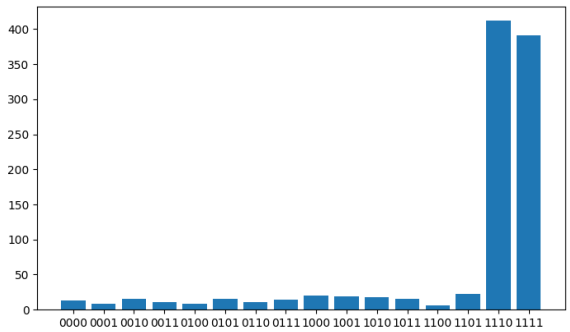

Fig 14. This is the distribution of searching the first fourteen states.

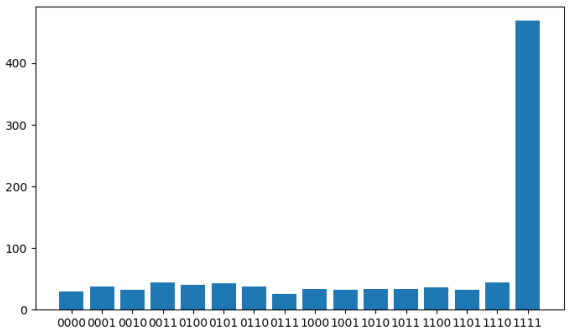

Fig 15. This is the distribution of searching the first fifteen states.

From the above seven plots, we can see that Grover's algorithm starts to do the opposite of what we want to do. It amplifies the probability of measuring non-target states. For instance, in Fig. 15, we try to amplify the probability of measuring all states other than |1111>. However, based on the distribution of measurement results, we observe that Grover's algorithm amplifies the probability amplitude of the only non-target state |1111>. The reason for this to happen is that when we invert more than half of all states, the mean will change to a negative value. When the diffuser rotates the phase about the mean, the absolute value for the phase of non-target states will increase because their phase is a positive value. Therefore, we have a higher probability of measuring the non-target states.

All in all, the conclusion of our first experiment is that Grover's algorithm only works when searching for strictly less than half of all possible states. Suppose the number of target states is equal to or greater than half of all states. In that case, Grover's algorithm might give a random result with equal probability or even return a non-target state with high probability.

Our second experiment is to compare the measurement results of the quantum simulation to the quantum computer. We use the following circuit for both the simulation and running on the quantum computer:

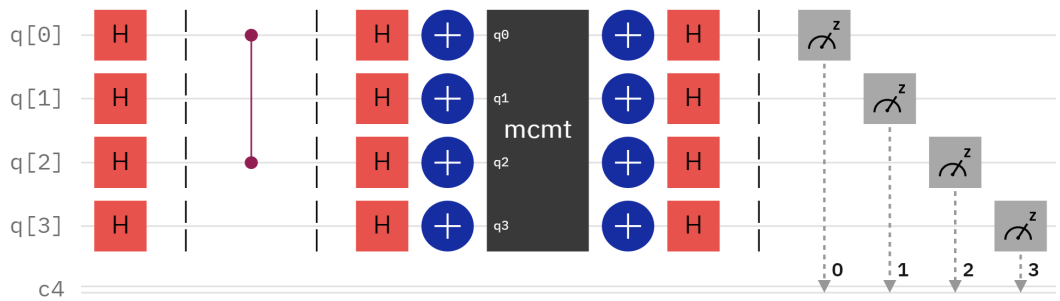

Fig 16. This is the quantum circuit for the second experiment. This circuit only has one amplification with target states: |0101>,|0111>,|1101>, and |1111>.

And we obtain the following results:

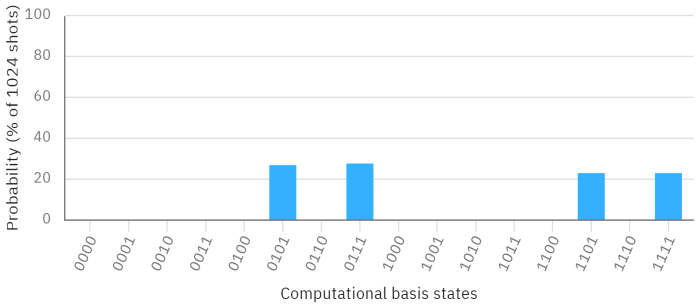

Fig 17. This is the distribution of running the circuit in Fig.16 using simulation. The X-axis is the state of 4 qubits, and the Y-axis is the probability of measuring that state in 1024 samplings.

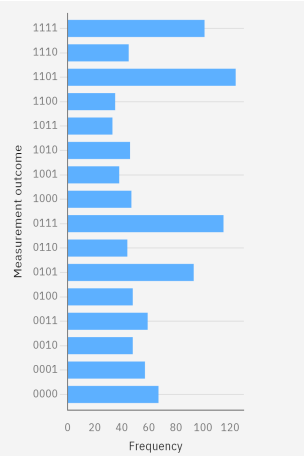

Fig 18. This is the distribution of running the circuit in Fig.16 using a quantum computer. The Y-axis is the states of 4 qubits, and the X-axis is the number of times measuring that state in 1024 samplings.

From Fig.17, we can see that in an ideal setting without quantum noise, the probability of measuring each target state is close to 25%. And the probability of measuring non-target states is 0%. However, in Fig.18, we observe that we will measure non-target states on a real quantum computer, though the 4 target states still have a slightly higher probability of being measured. This means quantum noise has a significant impact on Grover's algorithm, and it needs to run on a false-tolerant and error-corrected quantum computer.

3 PART II: CONSTRAINT SATISFACTION PROBLEMS SOLVER

The Constraint Satisfaction Problem is a mathematical problem that finds a solution for a constrained equation. This project focuses on Boolean Satisfiability Problem (SAT Problem). SAT Problem is widely used in the field of circuit design. However, as an NP-complete problem, there is no polynomial time algorithm for 3SAT problems. Therefore, we developed a program that inputs the DIMACS CNF file[5] and solves the Conjunctive Normal Form(CNF) expression.

3.1 Oracle Builder

The Oracle for a CNF logic expression requires simulating classical OR and AND gates. Therefore, compared to our previous work, we use a boolean oracle installed of phase oracle in this section. Unfortunately, this approach increased the number of qubits and the complexity of the circuit. However, phase kickback allows the circuit to implement AND gate between the CNF clusters using a multi-controlled Toffoli Gate.

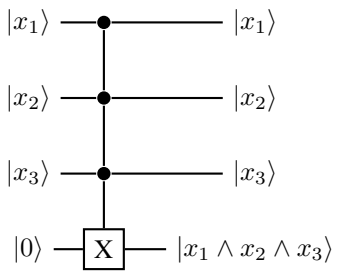

Figure 1: An example of 3 inputs AND Gate

Similarly, the OR Gate can be formed through one Toffoli gate and two CNOT gates with one ancillary qubit. To prove this gate is correct, we construct the truth table.

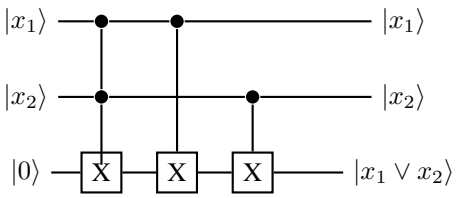

Figure 2: An example of 2 inputs OR Gate

x1	x2	x3	x3'
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	1

Using these OR gates, we can construct the circuit of each cluster and output the result to an ancillary qubit. Connecting these ancillary qubits through a AND gate, we can assemble the oracle for the expression. Note that we are using the phase kickback, so the gates need to be applied again. Since every OR gate requires an ancillary qubit, an AND gate and phase kickback process require an additional qubit. The circuit requires n qubits, where n = number of variables + number of OR gates + 1.

3.2 EXPERIMENT

From part 1, we discovered that when the number of the searching targets is equal to $\frac{1}{4}$ of the total possible outcome, Grover's Algorithm can find results without any error using only one iteration. Therefore, to simplify the problem. We use the logical expression $f(x_1, x_2) = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$ as our testing expression. This expression has only one solution: $x_1 = 1$ and $x_2 = 2$. With two variables and three OR gates, the circuit requires six qubits. The circuit is shown below.

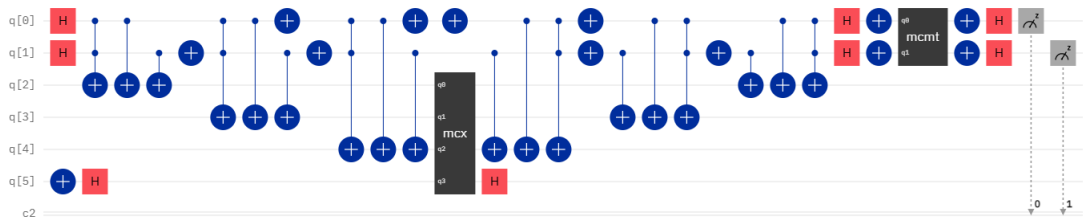

Fig 19. The circuit for the example CNF.

On the simulator, the circuit successfully returns the expected result.

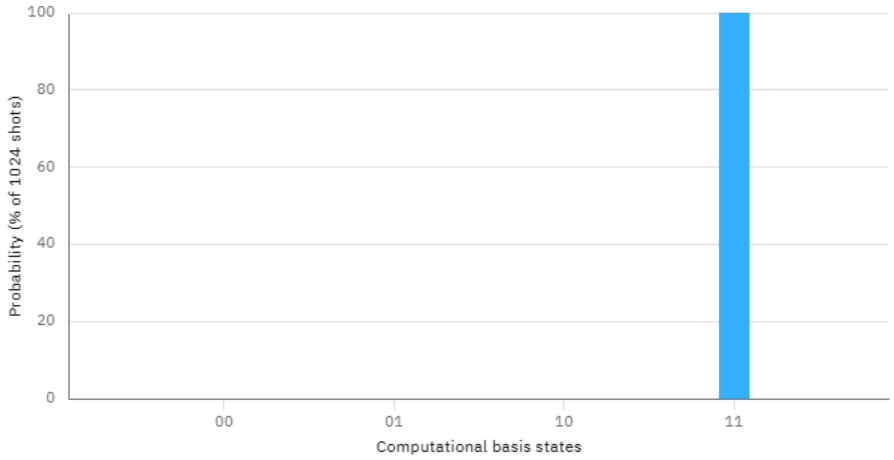

Fig 20. Simulator Result for the example CNF. the y-axis represents the probability of each state, and the x-axis represents the possible outcome

However, when we run the same circuit on a real Quantum computer, the result is not ideal. The four outcomes are equally distributed. We believe this is due to the large number of gates in the circuit which lead to a high error rate.

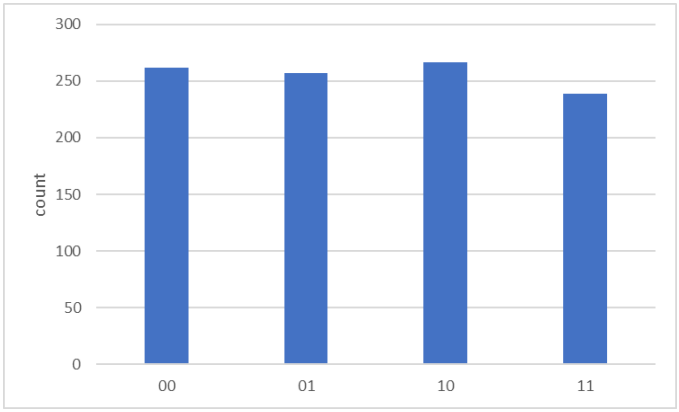

Fig 21. Real Quantum Computer Result for the example CNF. the y-axis represents the count of each state in a total of 1024 trails, and the x-axis represents the possible outcome

Furthermore, when we increased the CNF complexity, we found that the model won't be able to find the correct result for every CNF expression. We believe this is because the iteration time is incorrect. However, it is impossible o determine the correct iteration time since it requires knowing the number of target outcomes. We tested the algorithm on three variable cases. Our algorithm successfully finds the solution with 84% accuracy on the first equation $f(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$. This equation has three targets that are close to $\frac{1}{4}$ of the total outcome. However, when we test on equation $f(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$. The algorithm failed to return the correct result.

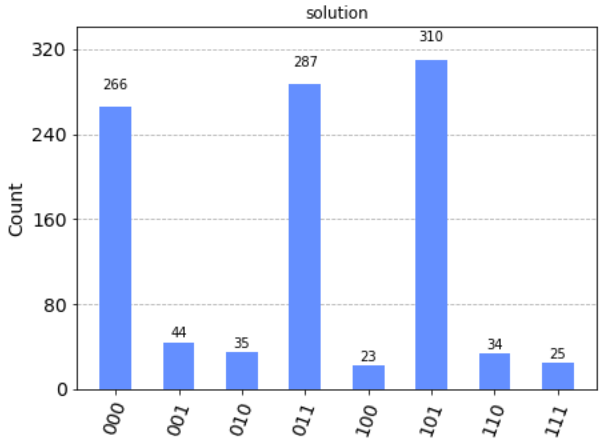

Fig 22. Result for expression $f(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

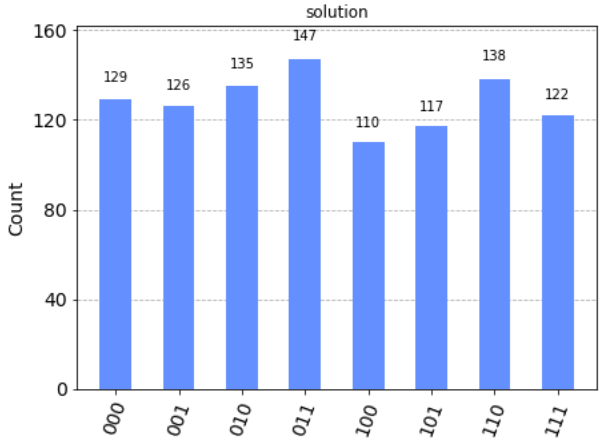

Fig 23. Result for expression $f(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$

3.3 CONCLUSION

The experiment shows our circuit built is correct for simple problems. However, when the target outcome is not $\frac{1}{4}$ of the total outcome, Grover's Algorithm won't be able to return the correct result in 1 iteration without error. Therefore, using Grover's Algorithm to solve SAT problem has limitations. Future work should be focused on generating the algorithm for every CNF expression. There are some related works, such as improving Grover's Algorithm by replacing phase rotation steps.[4] Overall, Grovers's Algorithm shows the potential to solve complex problems; in the best cases, it can solve 3SAT problems in linear time. Thus, Quantum Computing provides a new thought to determine whether P is equal to NP.

References

[1] Zijian Diao. "Exactness of the original Grover search algorithm". In: *Physical Review A* 82.4 (2010). DOI: 10.1103/physreva.82.044301. URL: https://doi.org/10.1103%2Fphysreva.82.044301.

[2] C. Figgatt et al. "Complete 3-Qubit Grover search on a programmable quantum computer". In: *Nature Communications* 8.1 (2017). DOI: 10.1038/s41467-017-01904-7. URL: https://doi.org/10.1038%2Fs41467-017-01904-7.

[3] Lov K Grover. "From Schrödinger's equation to the quantum search algorithm". In: *Pramana* 56.2-3 (2001), pp. 333–348. DOI: 10.1007/s12043-001-0128-3. URL: https://doi.org/10.1007%2Fs12043-001-0128-3.

[4] G. L. Long. "Grover algorithm with zero theoretical failure rate". In: *Physical Review A* 64.2 (2001). DOI: 10.1103/physreva.64.022307. URL: https://doi.org/10.1103%2Fphysreva.64.022307.

[5] *SAT competition 2009: Benchmark submission guidelines*. 2008. URL: http://www.satcompetition.org/2009/format-benchmarks2009.html.

[6] The Qiskit Team. *Grover's algorithm*. 2022. URL: https://qiskit.org/textbook/ch-algorithms/grover.html.

[7] Ronald de Wolf. *Quantum Computing: Lecture Notes*. 2019. DOI: 10.48550/ARXIV.1907.09415. URL: https://arxiv.org/abs/1907.09415.