Project Report for Quantum Information Science and Technology (Spring 2023)

Grover and The Search Problem

Abdul Rafay Salim[1]

PHY-417, Department of Physics, SBASSE, LUMS.

E-mail: 24100096r@lums.edu.pk

Abstract: The Grover search algorithm, a quantum-inspired search algorithm rooted in amplitude amplification, has garnered considerable attention for its potential to significantly accelerate search tasks compared to classical algorithms. In this report, we conduct an in-depth analysis of the Grover search algorithm, elucidating its theoretical foundations, key components, and practical applications. We delve into the algorithm's inner workings, including the oracle and diffusion operator, and elucidate how they synergistically amplify the amplitude of the target state, leading to a quadratic speedup in the search process. We further explore the algorithm's performance in diverse scenarios, such as unstructured search, element distinctness, and approximate counting, and discuss its advantages and limitations. Moreover, we review the current status of experimental implementations of Grover search in quantum computers, outlining the progress made and the challenges encountered. Additionally, we highlight potential future directions and applications of Grover search, such as quantum machine learning, optimization, and cryptography. Overall, this report aims to thus present a comprehensive and insightful overview of the Grover search algorithm, encompassing its theoretical foundations, experimental advancements, and potential applications, underscoring its significance as a promising quantum-inspired search tool with far-reaching implications for quantum computing and information processing.

Keywords: Quantum Algorithms, Grover Search

Contents

1 Introduction

We begin our dive into the workings of the Grover Search Algorithm by asking a simple a question: if we were given a range of possibilities and had to determine which, if any, of the possibilities satisfy a required outcome ; how long would it take?

In its essence, the algorithm seeks to solve the search problem, a problem that was widely prevalent in the world of f computational complexity theory, computability theory, and decision theory well before the algorithm was even conceptualised.

1.1 The Search Problem

Before we go into the details how Grover aimed to solve the search problem, it would serve us well to first properly understand the essence of the problem itself.

Lets try to frame the problem in a more practical manner. Imagine that we have a large book containing an 'N' number of names and of those 'N' names we want to determine whether a specific name that we seek, say for example 'Alice', exists within this book of names. Now if we were to look at this problem in the classical sense, we could go through each name one by one and see if it matches up with 'Alice'. This means that at most we would have to do this N times, each time examining a name and comparing it to 'Alice', in order to find our desired output. Therefore we would say that this process would be accomplished in a maximum of O(N) steps.

Now in a nutshell, the search problem can be simply be summarised as follows:

If $f(x)$ is a function that takes as its inputs items from a data set and works such that $f(x) = 1$ if x is our required search time or solution and $f(x) = 0$ if it isn't then our problem would be to find the particular x_o from the data set such that $f(x_o) = 1$ [Theory of Grover]

Note that all search problems often include:

- A set of states: our database that we must search through
- A start state
- A goal state: this is our function that evaluates whether the state given is the one that we require
- A successor function

Note that with our solutions to this problem, we need to solve the identification problem without making any assumptions about the specific structure of the problem itself. In other words, our solution must be easily applied to any search problem that is defined by our characteristics above.

Classically, there exist three types of classical algorithms that aim to solve the problem. These are:

- Linear Search
- Binary Search
- Hashing

Linear Search The linear search is perhaps the most basic and simple one of all the algorithms. It follows the same rules as we did in our earlier example of the names within the book. The search examines each element of our data-set one by one until it detects a match or finishes searching through the set. This type of search operates in linear time and will have a maximum of N iterations. In practical applications, linear search is rarely ever preferred due to other more efficient classical algorithms such as the binary search and hash tables.

Binary Search A more efficient classical algorithm to solve the search problem is that of the binary search. Like before, the algorithm aims to find and output our goal state within a data-set. It works by taking a given state and contrasting it with the middle element of the data-set. If the given state is not equal to our goal state then the algorithm does away with the half of the array in which the goal state cannot lie and then searches through the other half. This process is repeated until our goal state is found or if the half of the array we're left with is empty, in which case our goal state is not present within the data-set. One caveat of this algorithm however is that it only works for sorted lists and cannot be employed for unsorted data-sets. Unlike the linear search, which works in linear time, the binary search operates in logarithmic time and required at most $\log n$ iterations thus making it much more efficient than the linear search. A more efficient version of this is the exponential search, which can be utilised for unbounded or infinite lists.

1.2 Grover's Idea

Having gotten an idea of the history of search problems in computational theory, we can now start to properly appreciate the ingenuity of the Grover search. In 1996, Lov Kumar Grover, an Indian-American computer scientist proposed the idea of the Grover database search algorithm. A quantum mechanical algorithm that utilised effects only present at the quantum level to speed up the search problem, making it much more efficient than any classical approaches. In quantum circuits, the state of a system can be in a superposition allowing multiple computations to be performed simultaneously, such successful computations reinforce each other while others randomly interfere. All of this allows the Grover search to provide us with a quadratic speedup with a maximum of $\sqrt{N}$ iterations required to find our goal state. [1 - Grover's Paper].

Thus if we had 1 million names,that have been listed randomly, to search from within our book, then the best classical algorithm would take at most 1 million iterations to arrive to the conclusion whereas by utilising Grover search one would only have to go through a maximum of 1000 iterations.

1.3 Complexity Classes

After getting a better idea of the concept behind search problems and Grover's approach to solving them, we can now briefly go over the concept of decision problems and complexity classes.

A decision problem is simply any problem that takes a set of input values and outputs a yes or a no. Hence our search problem will always have an associated decision problem

as well. A **P(polynomial)** problem is one that can be solved in polynomial time by any deterministic machine. Solutions to such problems are quite easy and can be applied both theoretically and practically. An **NP(Nondeterministic Polynomial)** problem is any decision problem such that its solution can be guessed by a non-deterministic algorithm in polynomial time.What this implies is that while it would be hard to find the solution to an NP problem, we can easily verify solutions to the problem in polynomial time. Note that the non-deterministic part implies that the algorithm follows no particular rule in making its guess. As for **NP-hard** problems, these are simply problems such that every NP problem can in polynomial time be reduced to the NP-hard problem. Finally, if a problem is **NP-complete** then it means that all other NP-problems can be reduced to it in polynomial time.

It is important to note however that the Grover search is not able to give us any solutions for NP-Complete problems in polynomial time as it is only a quadratic speedup and the square root of an exponential function would still be exponential. Yet, the Grover search can still be applied to other algorithms that seek to solve NP-Complete problems, as a crucial part of such algorithms involves conducting an intensive search which can be sped up by Grover search. One very famous example of an NP-Complete problem is the satisfiability problem (SAT). It was the first problem to be discovered to be NP-Complete and it will be of great importance to us as we will try to employ the workings of the Grover Search to try and solve it in a later section.

2 Mathematical Formulation

We can now finally begin to understand the workings of the Grover Search from the mathematics that underlays the process. We will then try to understand this process from a geometric perspective as well after which we will briefly go over the ways in which the algorithm can be used to solve many famous problems in the field of mathematics

2.1 Background

The paper assumes that the reader has a well-established understanding of the way Dirac notation is used to represent the evolution of quantum states and any operators that act upon it. In any case, we shall first go over some important guidelines and other pieces of information that are important for us to understand first.

As always, a quantum state is represented, using dirac notation, as $|x\rangle$ where the x is our qubit which could be either 1 or 0 thus each qubit can be represented as:

$$|x\rangle = |0\rangle \text{ or } |1\rangle \tag{2.1}$$

We will also by writing states together in terms of their tensor product thus a state that is say represented as: $|01\rangle$ is simply:

$$|01\rangle = |0\rangle \otimes |1\rangle \tag{2.2}$$

Furthermore, each state can also be represented in its vector form as:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.3}$$

Note that we the total number of states as $2^n = N$ where n is number of qubits and N is the total number of possible states.

We also need to know about some quantum gates that are utilised in the algorithm:

Hadamard : Firstly we will look at the hadamard. This gate works in such a manner that it takes an input state outputs a type superposition of the basis states depending on the input state it acts on:

$$\hat{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{2.4}$$

while the action of the gate on our two states is represented as:

$$\begin{aligned} \hat{H}|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ \hat{H}|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned} \tag{2.5}$$

Not Gate : The Not Gate also known as the Pauli-X gate acts in a manner such that it takes its input quantum state and flips its value to the other basis state. Similarly to how a not gate flips 1 to a 0 or vice versa with normal logic gates.

In matrix form it is represented as:

$$\hat{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.6}$$

while the action of the gate on our two states is represented as:

$$\begin{aligned} \hat{X}|0\rangle &= |1\rangle \\ \hat{X}|1\rangle &= |0\rangle \end{aligned} \tag{2.7}$$

The Controlled Not and Toffli Gate : The Controlled Not Gate is similar to our not gate except that it takes two qubits as its input with one qubit acting as the control and the other as the target. If the control qubit is in the $|1\rangle$ state that then it flips the target qubit otherwise it leaves the target qubit untouched. The Toffli gate works in the same manner except that it takes three qubits as its input and has two control qubits, both which must be in state of $|1\rangle$ in order for the third target qubit to be flipped:

In Matrix representation they are:

$$\hat{C_N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.8}$$

$$\hat{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.9}$$

While we can define their actions on the states as:

$$\begin{aligned} \hat{C_N}|10\rangle &= |11\rangle \\ \hat{C_N}|00\rangle &= |00\rangle \end{aligned} \tag{2.10}$$

$$\begin{aligned} \hat{T}|110\rangle &= |111\rangle \\ \hat{T}|100\rangle &= |100\rangle \end{aligned} \tag{2.11}$$

The Oracle : Finally, there's also the concept of the oracle in quantum circuits that we must also understand. The quantum oracle is a type of black box that takes as its input a set of qubits, performs linear transformations on them and outputs a set of qubits. The oracle is built in such a way that it will automatically run a particular operation based on the input value, acting sort of like a switch statement. This means that the particular switch statement in mind must already by known and programmed beforehand. We can, in some sense, simply view the oracle as a collection of quantum gates that act on a set of input qubits.

We are now finally in a position to behind understanding the logic and mechanism of the Grover Search. Before we deal with a specific example, we begin by stating the simple formula that the algorithm applies on all of its input states. If we have an n string of qubits: $|\mathbf{O}\rangle$ of which $|x\rangle$ is the solution then:

$$\begin{aligned} |\mathbf{O}\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) &\xrightarrow{\hat{T}} \text{Phase difference applied on } |x\rangle \\ &\xrightarrow{\hat{D}} \text{Probability Amplitude of } \langle x\rangle \text{ amplified} \end{aligned} \tag{2.12}$$

Hence, the Grover algorithm takes as its input a string on n bits, applies a Toffli gate on them to imbue a phase difference on our target and then by taking the advantage of this phase difference, it amplifies the probability amplitude of our target state using the operator $\hat{D}$.

The $\hat{D}$ in this case is what forms the crux of our quantum algorithm. Indeed, the Grover Search is sometimes referred to as Amplitude Amplification as well. The gate that applies this is known as the Diffusion Operator. It works by essentially inverting the probability amplitude of a particular state about its mean. When we imbue a phase difference using the Toffli gate on our desired state, the diffusion operator then inverts the

amplitudes of the states about their mean which results in our target state having a higher amplitude then the rest.

The Toffli Gate alongside the Diffusion Operator together form what is known as the Grover Operator $\hat{G}^k$. The oracle in this case is simply our Toffli gate.

The Diffusion operator is defined as follows:

$$\begin{aligned} D_{ij} &= \frac{2}{N},\ i \neq j \\ D_{ii} &= -1 + \frac{2}{N},\ \text{otherwise} \end{aligned} \tag{2.13}$$

The operator can also be written as

$$\begin{aligned} \hat{D} &= 2\hat{P} - \mathbb{1} \\ \hat{D} &= (2|\psi\rangle\langle\psi| - \mathbb{1}) \end{aligned} \tag{2.14}$$

The P mentioned above is simply a projection operator such that:

$$\hat{P_{ij}} = \frac{1}{N},\ \text{for all i,j} \tag{2.15}$$

The action of it is such that P acting on any vector gives a vector each of whose components is equal to the average of all components.

Hence if we have a state $|\phi\rangle$ such that:

$$|\phi\rangle = \sum_{i=1}^{N} a_i|i\rangle \tag{2.16}$$

then the action of the diffusion operator on such a state would be:

$$\begin{aligned} (2|\psi\rangle\langle\psi| - \mathbb{1})|\phi\rangle &= \left[2\left(\sum_{i=1}^{N}\frac{1}{\sqrt{N}}|i\rangle\right)\left(\sum_{j=1}^{N}\frac{1}{\sqrt{N}}\langle j|\right) - \mathbb{1}\right]\left[\sum_{i=1}^{N} a_i|i\rangle\right] \\ &= 2\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\frac{1}{N}a_k|i\rangle\langle j|k\rangle - \sum_{k=1}^{N} a_k|k\rangle \\ &= 2\sum_{i=1}^{N}\sum_{k=1}^{N}\frac{1}{N}a_k|i\rangle - \sum_{k=1}^{N} a_k|k\rangle \\ &= 2\langle a\rangle\sum_{i=1}^{N}|i\rangle - \sum_{i=1}^{N} a_i|i\rangle \\ &= \sum_i \big[2\langle a\rangle - a_i\big]|i\rangle \end{aligned} \tag{2.17}$$

Hence equation (2.17) gives us the operation of the diffusion operator on any state $|\phi\rangle$. It multiplies the average probability amplitude by 2 and then from it subtracts each individual probability amplitude, creating a new set of states. This will become more clearer when we do an example.

2.2 The 2-Qubit Case

Lets now consider an example and see how this algorithm specifically manipulates our quantum states. Suppose we have a two qubit state such that our n = 2 and we have a total of $2^n = 2^2 = 4$ states. In order to perform the algorithm on these two qubits we also need another qubit that will act as our anciliary and allow us to 'kick' up a phase to our desired state. In this case we will choose our desired state to be the $|11\rangle$ state.

We begin by initialising all three of our qubits in the $|0\rangle$ state:

$$|0\rangle \otimes |0\rangle \otimes |0\rangle \tag{2.18}$$

We then apply a not gate to our third qubit:

$$|0\rangle \otimes |0\rangle \otimes |0\rangle \xrightarrow{\hat{X}_3} |0\rangle \otimes |0\rangle \otimes |1\rangle \tag{2.19}$$

After that we apply a hadamard gate on each of the three qubits:

$$|0\rangle \otimes |0\rangle \otimes |1\rangle \xrightarrow{\hat{H}_{1,2,3}} \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big) \otimes \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big) \otimes \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) \tag{2.20}$$

Thus after the first few gates we define the state as:

$$|\psi_1\rangle = \frac{1}{2}\big(|00\rangle + |01\rangle + |10\rangle + |11\rangle\big) \otimes \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) \tag{2.21}$$

We then enter our oracle which carries out the operation of $\hat{G}^k$. Firstly we apply a Toffli gate with our first two qubits acting as a control and our third qubit acting as the target:

$$\begin{aligned}
\hat{T}|\psi_1\rangle &= |\psi_2\rangle \\
&= \left[\frac{1}{2}\big(|00\rangle + |01\rangle + |10\rangle\big) \otimes \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)\right] - \left[\frac{1}{2}\big(|11\rangle\big) \otimes \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)\right] \\
&= \frac{1}{2}\big(|00\rangle + |01\rangle + |10\rangle - |11\rangle\big) \otimes \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)
\end{aligned} \tag{2.22}$$

Hence due to the function of the Toffli gate, the $|11\rangle$ state caused the third qubit to flip its states thereby kicking up a phase to the $|11\langle$ making it so that of the four possible states of the first two qubits, the $|11\langle$ state now has a phase difference.

We then take the state $|\psi_2\rangle$ output by the Toffli gate and have the Diffusion operator act upon the fist two qubits only. Lets see how the diffusion operator would work on these particular states. We start by using the equation (2.17), $\sum_i \big[2\langle a\rangle - a_i\big]|i\rangle$. The four states have probability amplitudes $\frac{1}{2}$, $\frac{1}{2}$, $\frac{1}{2}$, -$\frac{1}{2}$ respectively. The average of these amplitudes, $\langle a\rangle$, is:

$$\begin{aligned}
\langle a\rangle &= \frac{\frac{1}{2} + \frac{1}{2} + \frac{1}{2} - \frac{1}{2}}{4} \\
&= \frac{1}{4}
\end{aligned} \tag{2.23}$$

Thus for each of the four states:

$$\begin{aligned}
\left[2(\frac{1}{4}) - \frac{1}{2}\right]|00\rangle &= 0|00\rangle \\
\left[2(\frac{1}{4}) - \frac{1}{2}\right]|01\rangle &= 0|01\rangle \\
\left[2(\frac{1}{4}) - \frac{1}{2}\right]|10\rangle &= 0|10\rangle \\
\left[2(\frac{1}{4}) + \frac{1}{2}\right]|11\rangle &= 1|11\rangle
\end{aligned} \tag{2.24}$$

After the application of the oracle we then define our state as:

$$|\psi_3\rangle = 0|00\rangle + 0|01\rangle + 0|10\rangle + 1|11\rangle \tag{2.25}$$

In our final $|\psi_3\rangle$ state, we have the first three states with a probability of 0 whereas our desired state, the $|11\rangle$ state has a probability of 1. Thus if we were to carry out a measurement, we would only measure $|11\rangle$.

To summarise, we started with a two qubit state. We added an anciliary qubit, bringing the total qubits we have to 3. We applied a Not gate on the third qubit so that after applying the Hadamard gate its $|1|rangle$ state has a negative phase so that when the Not gate is applied to it that phase is switched; this allows us to kick that switch in the phase to the $|11\rangle$ state that caused it. This gives us the first two qubits in a superposition of four states of which the $|11\rangle$ state has a negative phase. We then applied the diffusion operation, which took advantage of the -ve phase on the fourth state to amplify its amplitude and suppress the amplitude of the other states. Note that for this two qubit case we only had to apply the Grover Operator once to get our state with a probability of 1. Greater number of qubits would require multiple applications of the operator to increase the probability of obtaining our desired state. No matter the number of qubits, the algorithm has to be applied a maximum of $\sqrt{N}$ times in order for us to arrive at our desired state. This is why it is said that the Grover Search algorithm is $O(\sqrt{N})$

We could carry out our algorithm for different desired states as well. If our desired state is $|00\rangle$ then we would apply two Not gates on the first two qubits so that the first state is now $|11\rangle$. We would then apply the Toffli gate which would imbue a phase difference on our new $|11\rangle$ state. After which we can again apply two not gates on the first two qubits so that our state with a phase difference is now the $|00\rangle$ state. The Diffusion operator can then be applied just like before.

2.3 Geometrical Interpretation

There is also another approach to understanding the way the Grover Search which is known as its geometrical interpretation. Using the earlier example of the two qubit case, we can divide the four states into two, $|x\rangle$ and $|y\rangle$, where:

$$|y\rangle = \frac{1}{\sqrt{N-1}} \sum_{i=1}^{N-1} |a_i\rangle = \frac{1}{\sqrt{3}}\big(|00\rangle + |01\rangle + |10\rangle\big) \tag{2.26}$$

These are the basis states that we are not interested in

Whereas:

$$|x\rangle = |11\rangle \tag{2.27}$$

is our desired goal state.

Hence we define a $|\psi_o\rangle$ such that:

$$|\psi_o\rangle = \frac{1}{\sqrt{N}}|x\rangle + \frac{\sqrt{N-1}}{\sqrt{N}}|y\rangle \tag{2.28}$$

Equation (2.28) forms our space that we work in where the $|y\rangle$ is orthogonal to $|x\rangle$. Note that both $|y\rangle$ and $|x\rangle$ are orthogonal to each other. We will see that in this new space that we have defined, the action of the Grover Operation $\hat{G}^k$ is a series of successive reflections and rotations within this space.

By applying the Toffli gate, we cause a phase inversion of our desired state thus in our solution space it is a reflection about the $|y\rangle$ axis. We then look at the angle between our newly formed reflected state and our original state. By applying the diffusion operator, this newly formed state is then rotated clockwise bringing it closer to our required $|x\rangle$ axis.

Thus in our solution space, this looks like:

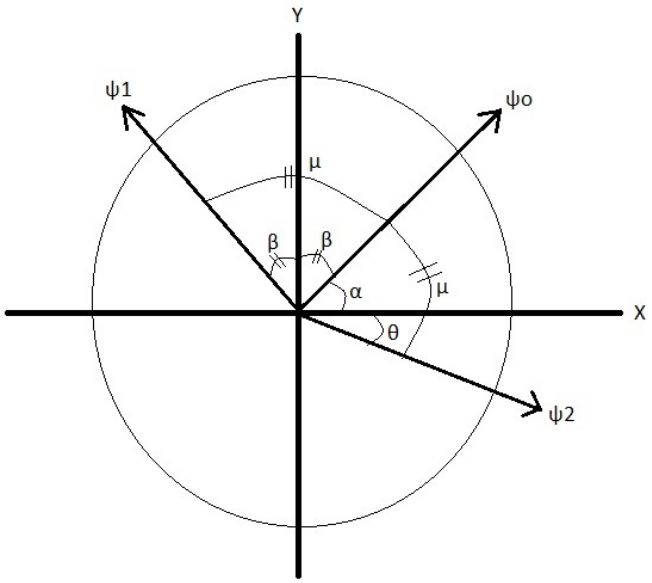

Figure 1. The Solution Space

The angle α here is equal to $cos^{-1}\left(\frac{1}{\sqrt{N}}\right)$ which means that β is equal to $cos^{-1}\left(\frac{\sqrt{N-1}}{\sqrt{N}}\right)$ and because $2\beta = \mu$ then μ is equal to $2cos^{-1}\left(\frac{\sqrt{N-1}}{\sqrt{N}}\right)$. Hence our angles are:

$$\alpha = cos^{-1}\left(\frac{1}{\sqrt{N}}\right),\ \beta = cos^{-1}\left(\frac{\sqrt{N-1}}{\sqrt{N}}\right),\ \mu = 2cos^{-1}\left(\frac{\sqrt{N-1}}{\sqrt{N}}\right) \tag{2.29}$$

We began in our ψ_o state, it makes an angle α with the solution axis thus making an angle β with the undesired axis. The Toffoli gate reflects the state about the $|y\rangle$ axis which means that now the new ψ_1 state makes an angle $2\beta = \mu$ with the old state. The diffusion operator then causes the ψ_1 state to be rotated, clockwise, by an angle μ and then a further angle μ bringing us to the ψ_2 state. This count as one iteration of our $\hat{G}$ operator.

After one iteration, we can check whether we have arrived at our goal state by looking at the angle θ. This is the angle that our ψ_2 state makes with the solution axis. If θ is 0 then we have arrived at our goal state. Otherwise more applications are required. Using this information we therefore calculate the probability of obtaining our goal state after k applications of the Grover operator by:

$$P(search) = cos^2\left(\frac{\pi}{2} - cos^{-1}\left(\frac{\sqrt{N-1}}{\sqrt{N}}\right) - 2kcos^{-1}\left(\frac{\sqrt{N-1}}{\sqrt{N}}\right)\right)$$
$$where, \tag{2.30}$$
$$\theta = \frac{\pi}{2} - cos^{-1}\left(\frac{\sqrt{N-1}}{\sqrt{N}}\right) - 2kcos^{-1}\left(\frac{\sqrt{N-1}}{\sqrt{N}}\right)$$

2.4 Some Applications to Mathematical Problems

In our introduction, we briefly mentioned how the Grover Search finds great use in being applied to solve several other mathematical problems, whether that be on its own or as a subroutine of a larger algorithm. We shall go over two such problems and hopefully by the end we have a greater appreciation for our quantum algorithm and its applicability in the modern world.

2.4.1 The Satisfiability Problem

The first of these problems is the Satisfiability problem. It is an NP-Complete problem (see section 1.2) and one of the most exhuastively studied problems in the field of computational complexity theory. We will be looking at a specialised version of the problem known as 3-SAT but our algorithm can be applied to more general k-SAT problems as well, which is where the Grover Search really outshines any classical algorithms.

The problem is as follows. We have a Boolean Formula consisiting of a number of Boolean variables. The problem asks if, given the Boolean formula, we can consistently replace the boolean variables by true or false such that the Boolean formula outputs a true. Thus the problem consists of finding the values of the boolean problems such that the boolean formula is true hence satisfying the satisfiability conditions.

When looking at the 3-SAT version of this problem we have the Boolean formula:

$$f(v_1, v_2, v_3) = (\neg v_1 V \neg v_2 V \neg v_3) \wedge (v_1 V \neg v_2 V v_3) \wedge (v_1 V v_2 V \neg v_3) \wedge (v_1 V \neg v_2 V \neg v_3) \wedge (\neg v_1 V v_2 V v_3) \tag{2.31}$$

One approach to finding the correct values of v_1, v_2, v_3 would be to try and evaluate every single possible combination of the three. As we can see, this is simply nothing more than our Grover Algorithm but for 3 qubits. Where we have certain 3 qubit states that

are our solution and satisfy this problem. Thus we can utilise the Grover Search to find the solution in a runtime of $O(\sqrt{N})$.

For this particular problem, the combination of variables that solve it are: [T, F, T], [F, F, F] and [T, T, F].

2.4.2 The Collision Problem

The second problem we'll be looking at is the Collision Problem. In the problem we are given a function f such that:

$$f : (1, ..., N) \rightarrow (1, ..., N) \tag{2.32}$$

We are told that N is even and f is two-to-one. f being two-to-one implies that there are two input values to f that give the same output. Thus our problem is to find an x and y such that $x \neq y$ and $f(x) = f(y)$. The efficiency of the solution depends on how long it will take us before we encounter a collision. A classical randomised algorithm can actually be used to solve the problem with $O(\sqrt{N})$ queries, this follows the same principle of the famous "Birthday Attack". Similarly we could also use the Grover search, where we calculate $f(1)$ and do a search for another value of x such that $x \neq 1$ and $f(x) = f(1)$. This method also solves the problem with $O(\sqrt{N})$ queries.

Then in 1997, Brassard, Hoyer and Tapp posited a method that combined both approaches in order to give more efficient algorithm. Their approach works as follows: Firstly, $\sqrt[3]{N}$ inputs are chosen at random and queried by f. The results of the query are then sorted. Then the Grover Search runs on the random inputs that weren't involved in the first time$(N^{2/3})$. Each one of these inputs are then given a mark depending on whether $f(x) = f(y)$ where y are the inputs of the first step. Note that we still only make N pairwise comparisons.

Using this method the efficiency of the algorithm comes out to be:

$$N^{1/3} + \sqrt{N^{2/3}} = O(N^{1/3}) \tag{2.33}$$

Therefore we combined a classical algorithm and the Grover Search to form a new algorithm that allowed us to evaluate the problem much faster than either individual algorithm could on its own.

A Bibliography

References

[1] Grover, Lov K. "A Fast Quantum Mechanical Algorithm for Database Search." Symposium on the Theory of Computing, July 1996, https://doi.org/10.1145/237814.237866.

[2]Grover's Search Algorithm and Amplitude Amplification — Grove 1.7.0 Documentation. grove-docs.readthedocs.io/en/latest/grover.html

[3]Team, Qiskit. Solving Satisfiability Problems Using Grover'S Algorithm. 6 Apr. 2023, qiskit.org/textbook/ch-applications/satisfiability-grover.html

[4] Brassard, Gilles, et al. "Quantum Algorithm for the Collision Problem." Springer eBooks, Springer Nature, May 1997, pp. 1662–64. https://doi.org/10.1007/978-1-4939-2864-4304.

[5] Nielsen, Michael A., and Isaac L. Chuang. Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge UP, 2010.

[6] GeeksforGeeks. "Types of Complexity Classes P NP CoNP NP Hard and NP Complete." GeeksforGeeks, 2023, www.geeksforgeeks.org/types-of-complexity-classes-p-np-conp-np-hard-and-np-complete.

[7] Knuth, Donald (1998). Sorting and Searching. The Art of Computer Programming. Vol. 3 (2nd ed.)