**UNIVERSITY OF PRINCE EDWARD ISLAND**

# Into the Quantum-verse

by

Chao Qian

A thesis submitted in partial fulfillment for the
degree of Bachelor of Science

in the
Honours Mathematics
School of Mathematical and Computational Science

April 2019

*Dedicated to*

*My parents Weihong Ren and Hongbo Qian*

*Who supported me this far.*

. . .

# *Acknowledgements*

First and foremost I would like to express my sincere gratitude to my supervisor, Dr. Gordon MacDonald for his patience and support, we met twice a week and discussed the details about the project. He inspired me, taught me this new and promising discipline. In the meantime, Dr. MacDonald encouraged me to participate in the Mathematics problem-solving competition and to give a talk on "2019 Science Atlantic Conference." Under his help, I successfully prepared everything in time, and the presentation went smoothly.

I also would like to thank Dr. Gordon MacDonald, Dr. Kai Liu for leading me into academic research, Dr. Alexander Alvarez and Dr. David Horrocks taught me in Mathematics problem-solving group, providing reference letters and crucial suggestions when I applied for graduate universities. I want to thank all the faculties in the School of Mathematical and Computational Science. I learned a lot of academic knowledge and gained a lot of support from all the professors. You are the crucial part of my kept making progress for the past two years.

My time at UPEI was made enjoyable in large part due to the many friends and groups that became a part of my life. I am grateful for the time spent with many friends, doing homework and attending conferences together. My time at UPEI was also enriched by the friends who already graduated; they set up good examples for me. It is my honour to be a student from the School of Mathematical and Computational Science, and the experience in the past two years will be the most valuable memory that I will cherish for the rest of my life.

Lastly, I want to thank my family for all their love and encouragement. For my parents who raised me with a love of science and supported me in all my pursuits. Thank you all.

# Contents

# Chapter 1

# Introduction to Quantum Computing

We experience the benefits of classical computing every day. Nowadays, computers help and entertain us, connect us with people all over the world, and allow us to process vast amounts of data to solve problems and manage complex systems.

However, there are problems that existed systems will never be able to solve. For challenges above a certain size and complexity, we do not have enough computational power on Earth to tackle them. To stand a chance at solving some of these complex problems, we need a new type of computing: one whose computational power also scales exponentially as the system size grows.

The new computing we are talking about today is called Quantum Computing, which is used in the device called quantum computers. In the later of this thesis, we will examine how quantum computing and quantum algorithms solve specific problems more efficiently and make some exciting comparison between quantum and classical algorithms.

Quantum computing can one day cause a big breakthrough in many disciplines: Dr. Jerry Chow, manager of the experimental quantum research for IBM, applied quantum computing to molecular simulation and this led to new drug discovery; in the case of solving complicated optimization problems, we have quantum version of data fitting method- Quantum Least Square fitting, and quantum semidefinite programming which is used to solve the optimization of a linear function, quantum approximate optimization methods are used to find the approximate solution to optimization questions, which are usually considered to be NP-hard[1], etc.

As the other side of a coin toss, quantum computing will cause a disruption effect on encryption systems. Nowadays we use cryptographic algorithms to encrypt our data; the data can only be decrypted with the correct key. The public and private keys are related to large number

[1] A decision problem H is NP-hard when for every problem L in NP, there is a polynomial-time reduction from L to H.[11]

factorization, on a classical computer, people cannot factorize a large number in polynomial time, so this fact will guarantee our data is safe. But if we run Shor's algorithm on a quantum computer, we can solve the question efficiently.

A cryptographic algorithm which uses symmetrical encryption, take DES as an example, under this encryption, a classical computer needs to search $2^{56}$ possible keys to guarantee to crack DES encryption. However, one can run the Grover search algorithm on their quantum computer, and it will reduce the complexity from $O(N)$, to $O(\sqrt{N})$[1]. Luckily, not all existing cryptographic system is expected to have a quantum attack, since Shor's algorithm can only be effective when factorizing large number, and Grover search algorithm can speed up drastically when searching unordered database. In the future, we should choose the quantum resistant system to protect our privacy.

Quantum computing will not cause disruptive effects to society alone; it will combine with other new technologies. By the fast development of Artificial intelligence, it is suggested that by 2030, intelligent agents and robots will eliminate 30 percent of the world's human labor, and will displace 800 million people's job (McKinsey). If AI equipped with quantum computers, machines can solve questions much faster, run every program easier, those robots will displace most of the jobs in the world, except those jobs need creative thinking.

Can quantum computing change everything in the world? My opinion is "No." I believe quantum computing cannot change the Psychological counselling industry. Because counsellors need to detect the potential symptoms behind the "normal" behaviour of the patient. Psychologists need to understand the feeling behind the words of patients. Powerful computers such as quantum computers cannot have feelings, not even mention using their senses to "feel" others.

## 1.1 Classical and Quantum computing comparison

First thing we should know is that classical information stored in discrete value 0 or 1, and if store one data takes 32 bits, then we need N times 32 bits in total to store N exactly the same information. On the contrary, one big benefit that the quantum computing can bring us is that it processes data in superposition, and due to this, quantum computing has unimaginable power when processing data. Quantum information is stored in quantum bit -"qubit." If one more qubit is involved, the capability to store the data is doubled. For example, 2 qubits will give us 4 different states, i.e., $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, 3 qubits will give us 8 states.

Second, quantum computing cannot "copy" data, which is known as "non-cloning theorem." However, classical computing can.

Third, in the quantum world, if a system is made of multiple subsystems, we need introduce tensor product. What is tensor product? Essentially, tensor product is the bigger Hilbert space (a system) which result from the "product" of smaller sub Hilbert space. The interacting particles live in the space defined by the tensor product.

If the state of two interacting particles cannot be decomposed into a product of two wave functions from different spaces, the state then is "entangled." i.e., if a Hilbert space is not a tensor product, then it cannot be decomposed into two sub Hilbert space. As you can see, tensor product and entanglement appear at the same time.

If you want to find out the tensor product of two vectors $v$ and $w$, the tensor can be constructed using the outer product.

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}, \ \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

$$\mathbf{v} \otimes \mathbf{w} = \begin{bmatrix} v_1 w_1 & v_1 w_2 & \cdots & v_1 w_m \\ v_2 w_1 & v_2 w_2 & \cdots & v_2 w_m \\ \vdots & \vdots & \ddots & \vdots \\ v_n w_1 & v_n w_2 & \cdots & v_n w_m \end{bmatrix}$$

## 1.2 Qubit

As we all know, a bit is the basic unit of information processing in the classical computer. Nothing special to qubit, which is the short cut of a quantum bit, just like the ordinary bit in traditional computers, it is a basic unit of quantum information. Therefore, we may assume that a qubit is very similar to a bit, except it is used in quantum computers. However, we will see there are many fundamental differences between them, and these fundamental differences will let us process information more interestingly.

## 1.3 Quantum Properties

Quantum computation takes place in vector spaces $V$, among which $\mathbb{C}^n$ is especially important. $\mathbb{C}^n$ is a vector space, which satisfies associative, communicative, distributive, identity, and inverse axioms. We can use all the knowledge from linear algebra to do the quantum computation.

How is a quantum bit different from a classical bit? A bit in classical computers can only be in one of the two states - *0* state, or *1* state. While a qubit can be any linear combination of those

two pure states $|0\rangle$ and $|1\rangle$, which we say it exists in a superposition state, and write it as:

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.1}$$

Where $\alpha, \beta$ are complex numbers, we can express them as $z = a + ib, i^2 = -1$. How can we tell which state the qubit is in? we need to make measurement. At the moment we make measurement, the qubit randomly collapses into one of the base states, with the probability $|\ \alpha\ |^2, |\ \beta\ |^2$ respectively.

What constraint should we put on the multiplicative coefficients? We have to realize that the sum of the all possible outcomes should be 1, and hence $\alpha, \beta$ should be constraint by the requirement that

$$|\alpha|^2 + |\beta|^2 = 1 \tag{1.2}$$

Another unusual phenomenon exist in the quantum computing is called "entanglement," when two individual qubits entangled together, they cannot be treated separately. i.e., they can only be treated as a part of the system as a whole. It is also counter-intuitive that no matter how far two entangled qubits away from each other, there is a correlation exists between them: Measuring one qubit will immediately affect the others regardless of the distance between them.

## 1.4 Quantum Gates and Matrices

Compared to very limited logic gates in classical computers, we have a lot of quantum gates in quantum mechanics. What are quantum gates? Quantum gates, also called quantum operators are unitary matrices, they are used to change the direction of the vectors in the complex plane.

### 1.4.1 Some Logic Gates

Here are the true tables of conjunction and disjunction:

| $A$ | $B$ | $A$ AND $B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(1.3)

| $A$ | $B$ | $A$ OR $B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

(1.4)

The output of the exclusive or is true only when two inputs A and B are different, i.e.,

| $A$ | $B$ | $A$ XOR $B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(1.5)

### 1.4.2 Basic Quantum Gates

There are some fundamental quantum gates, such as Pauli gates, Hadamard gate. We can later use these basic and easy gates to build more complicated quantum operators. Pauli - X gate:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{1.6}$$

The Hadamard gate acts on a single qubit and creates a superposition. It maps the base state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$and $|1\rangle$ to$\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, which means that after the measurement, we will have equal probabilities to become 1 or 0. It represents a rotation of $\pi$ about the axis $(\hat{x}+\hat{z})/\sqrt{2}$. Equivalently, it is the combination of two rotations, $\pi$ about the Z-axis followed by $\pi/2$ about the Y-axis. It is represented by the matrix

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.7}$$

### 1.4.3 Reflection Matrix

**Definition 1** (Reflection Matrix)**.** For any vector $\vec{x} \in \mathbb{C}^n$, a *reflection transformation operator* reflects every vector $\vec{x}$ to its symmetric image about some plane $\mathbb{C}^n$.

Reflection operators are very important quantum gates, as we will see its application in the following chapters when we discuss Grover search algorithm. A unitary operator $Ref_a$ reflects unit vector b around a. In other words, the vector after the reflection, $b'$ and original vector b lie on different sides and have the same angle according to the vector a. The mapping from b to $b'$,

created by unitary matrices, results in the unitary of $b'$. Geometrically, the point on the line a is the projection of b onto a and is denoted by $a' = a < a,b >$, and hence,

$$b' = b - 2(b - a < a,b >) = (2P_a - I)b, \tag{1.8}$$

Where $P_a$ is the projection operator which will do the following operation:

$$\forall b, P_a b = a < a,b > . \tag{1.9}$$

**Example 1.1.** The reflection operator for $a$, where $a$ is a unit vector with all N entries $\frac{1}{\sqrt{N}}$, and R is the projector matrix whose entries are all $\frac{1}{N}$ is

$$P = 2R - I = \begin{pmatrix} \frac{2}{N} - 1 & \frac{2}{N} & \dots & \\ \frac{2}{N} & \frac{2}{N} - 1 & \dots & \\ \frac{2}{N} & \frac{2}{N} & \dots & \ddots \end{pmatrix} \tag{1.10}$$

### 1.4.4 Hadamard Matrix

**Definition 2** (Hadamard Matrix)**.** A *Hadamard matrix*, named after the French mathematician Jacques Hadamard, is a square matrix whose entries are either +1 or -1 and whose rows are mutually orthogonal.

Hadamard Matrices are probably the most famous unitary transform in quantum computing, here we assume that the N is always $2^n$ for some n belonging to $\mathbb{Z}$. Hadamard matrices are recursively defined as $H_2 = H$, in order to generate a general formula, we need $n \geq 4$.

$$H_N = H_{N/2} \otimes H = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{N/2} & H_{N/2} \\ H_{N/2} & -H_{N/2} \end{pmatrix} \tag{1.11}$$

It is obvious that $H_1 = (1)$, and $H_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

### 1.4.5 Fourier Matrix

**Definition 3** (Fourier Matrix)**.** An N-point Discrete *Fourier Matrix* (DFT) is expressed as the multiplication $X = Wx$, where $x$ is the original input signal, $W$ is the N-by-N square DFT matrix, and $X$ is the DFT of the signal.

This important matrix is associated with Quantum Fourier Transform, which is crucial to Shor's factorlization algorithm. We use $\omega$ to stand for $e^{2\pi i/N}$, and define the unitary matrix over the

complex Hilbert space $F_N$ as Fourier matrix of order N:

$$\frac{1}{\sqrt{N}}\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{N-2} \\ \vdots & & & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{N-2} & \dots & \omega \end{pmatrix} \tag{1.12}$$

## 1.5 Introduction to Three Games

In this thesis, we are going to discuss three games: Quantum maze, Battleship, Orientable Surfaces game. Grover search algorithm is used to search unstructured N files, and it turns out that this only needs $O(\sqrt{N})$ steps. On the contrary, Classical algorithm checks the files one by one, and we will hit the right one after $N/2$ files are examined with probability $1/2$. We want to apply Grover's algorithm to a maze game, to find out how efficient can the algorithm be to find the exit of the maze.

The second game we are going to discuss is called "Battleship." Different from the version we already familiar with, we borrow the quantum torpedoes from the "Star Trek," which are used to detect the ship (Not destroying it). We list three games' strategies which include both classical algorithm and also quantum algorithms. The goal is to analyse the efficiency between different strategies.

Because of the quantum inference, the quantum walk will exhibit different feature compared with classical random walk. We plan to use the quantum interference to identify different kinds of surface, such as Torus (orientable), Klein Bottle (non-orientable), and projective plane (non-orientable). In the following chapters, we are going to discuss the details of these games, hope you can enjoy!

# Chapter 2

# Quantum Maze

## 2.1 Grover Search Algorithm

Let us quickly go over the process of the Grover search[1]. Lov Grover's algorithm solves the problem- "find a needle in a haystack." Suppose we have a large space size $N$, $N/2$ queries on average are expected to find the target based on the classical search algorithm. In Grover's quantum search algorithm, $N/2$ can be surprisingly improved to $\sqrt{N}$, which achieves polynomial speed up. The steps can be summarized as the following:

Grover search steps:

1. Initialize the system to the state $|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$
2. Perform the following "iteration"
   - First, apply the operator $\mathbf{U}_\omega = \mathbf{I} - 2|\omega\rangle\langle\omega|$
   - Second, apply the operator $\mathbf{U}_s = 2|s\rangle\langle s| - \mathbf{I}$
3. Let $G = \mathbf{U}_S \mathbf{U}_\omega$, and apply $G$ $\frac{\pi\sqrt{N}}{4}$ times.
4. Make the measurement $\Omega$.

One way to think about Grover search is to consider "inverse about the average." The first step of Grover search, we begin by transforming our input into the uniform superposition, as you can observe in figure 2.1, the amplitudes of all qubits strings $\{00\cdots00\}$ are all equal, in other words, the uncertainty of which one is our target is the same.

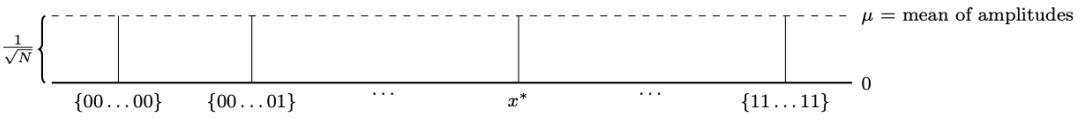



FIGURE 2.1: Step one

The second step is done by the oracle $\mathbf{U}_\omega$, it will flip the amplitudes of our target $x^*$, and leave the rest unchanged.

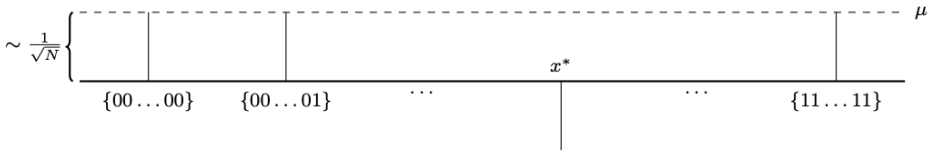



FIGURE 2.2: Step two

In the last step, operator $\mathbf{U}_s$ flips the amplitude of $x^*$ over the mean ($\mu$), then the amplitude of $x^*$ becomes $\frac{3}{\sqrt{N}}$, so the increment of the amplitude is $\frac{2}{\sqrt{N}}$. After we make observation, the probability of finding $x^*$ increases from $\frac{1}{N}$ to $\frac{9}{N}$. However, $\frac{9}{N}$ is still too far away from the

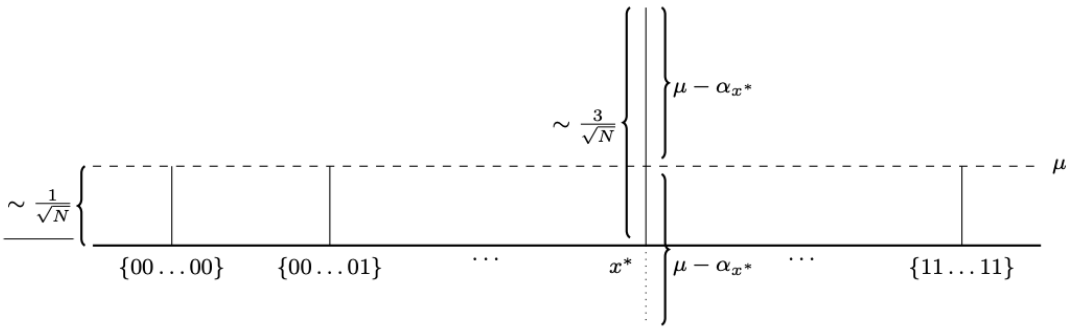



FIGURE 2.3: Step three

result we want. We need the likelihood closes to 1, which means we need to do more Grover iterations to achieve that. One should be careful about the number of iteration he chooses, since it could become a negative process. i.e., the increment is negative. When will it happen? This happens when the amplitude of $x^*$, $\alpha_{x^*}$ is sufficiently large, and we flip it to the negative value by the oracle, it will "drag" the mean to negative as well. Therefore, in the third step, we inverse $\alpha_{x^*}$ over the negative mean; this will result in a decreasing of the amplitude - a negative process. We will talk more details about this in the maze example.

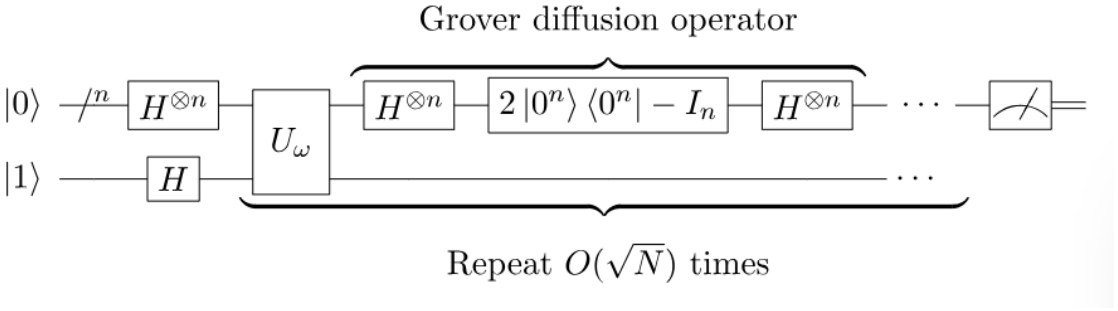
When we want to search multiple targets, There are several ways to handle the case if “k” is unknown. For example, one could run Grover’s algorithm several times, with

$$\frac{\pi}{4}N^{1/2},\ \frac{\pi}{4}\left(\frac{N}{2}\right)^{1/2},\ \frac{\pi}{4}\left(\frac{N}{4}\right)^{1/2},\ \ldots,\ \frac{\pi}{4}\sqrt{\frac{N}{2^k}},\ \ldots \tag{2.1}$$

iterations. For any “k,” one of the iterations will find a matching entry with a sufficiently high probability (almost surely). An upper bound of the number of the iteration is

$$\pi\frac{N^{1/2}}{4}\left(1+\frac{1}{\sqrt{2}}+\frac{1}{2}+\cdots\right)=\pi\frac{\sqrt{N}}{4}\left(2+\sqrt{2}\right) \tag{2.2}$$

which is still $\left(N^{1/2}\right)$. It can be shown that this can be improved. If the number of marked targets is “k,” where “k” is known, there is an algorithm that finds the solution in $\frac{\pi}{4}\sqrt{\frac{N}{k}}$ queries[2]. In this thesis, we will simply use this fact, but will not dig into the details of the Grover search for multiple targets. The quantum circuit for the Grover search is as follows:




Note: Quantum circuit is very popular to express the algorithm, since we do not need to write out all the operators (matrices) any more!

## 2.2 The Maze Game

How can we apply the Grover search to the maze game? Here is the situation:

Phil is trapped in a maze by an Oracle, there are $n$ dead ends in total, at the very end there is a pad attached to it, but only one pad will give Phil a cheese and teleport him out of the maze, the others will disintegrate him. When Phil reaches the end, he can use the pad to communicate with the Oracle, and ask if he is at the correct exit. The Oracle will answer with the truth, but limited times, in other words, there will be no response after some queries, say $m$. The question is, what are the odds that Phil can get out of the maze?

Let's first consider a straightforward maze game. As shown in the following figure 2.4, at every crossroad Phil can make his decision to go left or right, and there are four dead ends in total. We would like to analyze the odds that Phil can get out of the maze.

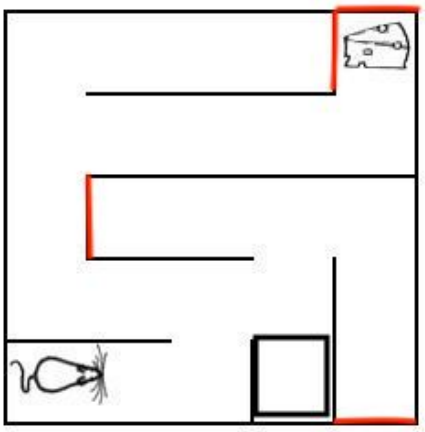


FIGURE 2.4: A simple maze game

The maze can be transferred into a graph as figure 2.5 Before we apply any quantum gates, the odds for each outcomes are equal to $\frac{1}{4}$.

Classical oracle will change the bit which represents the correct location into 1 and leaves the others 0, as shown in table 2.3. However, the quantum oracle will replace the figure with its opposite at that location, but will not tell you which one it is, shown in table 2.4. We need to apply quantum operators and find out on our own. Remember, this oracle corresponds with the second step of the Grover search algorithm.

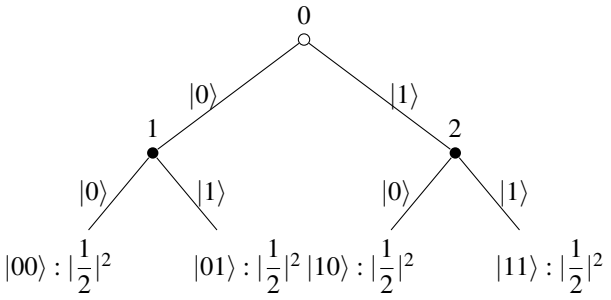



FIGURE 2.5: Example: 2 depth tree

| $x$ | $f(x)$ |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 10 | 0 |
| 11 | 0 |

(2.3)

What are the odds can Phil survive in this classical maze? Let us consider the depth-first search algorithm (please refer to Appendix to see more details). On average, he has 25% chance to survive, and he needs the Oracle to respond three times to guarantee to escape the maze. (If the first three answers obtained from the Oracle are all negative, then the last path is automatically the exit.)

$$U_\omega = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ -\alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} \tag{2.4}$$

In this quantum maze, Oracle matrix $U_\omega$ remains the same through the whole process, with -1 at $|01\rangle$ outcome. We apply $G\ \frac{\pi\sqrt{N}}{4}$ iterations to the initial state $s$, in order to rotate $s$ towards $\omega$, where $G = U_\omega U_s$. $U_s$ is reflection operator:

$$\mathbf{U}_\omega = \mathbf{I} - 2|\omega\rangle\langle\omega| = \begin{array}{c} \\ 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{2.5}$$

$$\mathbf{U}_s = 2|s\rangle\langle s| - \mathbf{I} = \frac{1}{2} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \tag{2.6}$$

As shown in figure 2.6, $U_\omega$ reflects $s$ over $s'$, and $U_s$ reflects the outcome from the last step $U_\omega s$ over $s$. The arrow $U_s U_\omega s$ rotates $\theta$ closer to the target $\omega$ during this process. If we consider the inverse of the average, the amplitude increases by $\frac{2}{\sqrt{N}} = 1$, and the rest three outcomes' amplitudes drop to 0.

Let us see more computation. The amplitude of $|01\rangle$ is 1 means $G_s = U_s U_\omega S = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$. Since

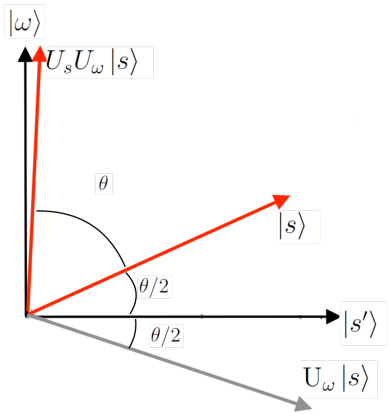



FIGURE 2.6: The first iteration

$\cos\frac{\theta}{2} = \frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} \cdot \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \frac{1}{2}$, it follows that $\frac{\theta}{2} = \frac{\pi}{6}$, using principle rule. Therefore, total angle that has been rotated is $\pi/3$, and hence, $G_s$ is located exactly on $|\omega\rangle$. The probability of finding $|01\rangle$ exit is 1. This means we only need the quantum oracle to reply once to guarantee the survival of Phil.

However, the certainty is not a general case if you increase the depth of the graph in figure 2.6. For example, if the number of outcomes is increasing, and we want to search this string$|01\cdots00\rangle$, then our result would be similar to this: $U^n = [\varepsilon, 0.9, \varepsilon, \cdots, \varepsilon]^T$. The probability of getting the correct location is very close to 1, but still, have some tiny likelihood that the mixed state will collapse to the wrong base state. In other words, the vector cannot lie on the $|\omega\rangle$ exactly.

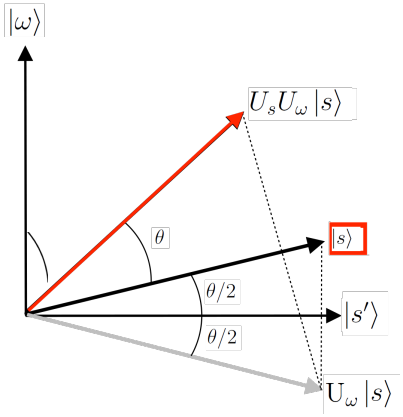



FIGURE 2.7: After many iterations

In this example we can tell there are some advantages of quantum over the classical algorithm, let us make some further comparison when $N$ is bigger. In a classical maze with 64 dead ends,

we can use the depth-first search algorithm to analyze the game. The initial probability of getting out of the maze is $\frac{1}{N}$, and will increase linearly when we make more queries to the oracle. The general formula is the following:

$$Probability = \begin{cases} \frac{1}{N} + \frac{k}{N} & ,0 \leq k \leq N-1 \\ 1 & ,k = N \end{cases} \tag{2.7}$$

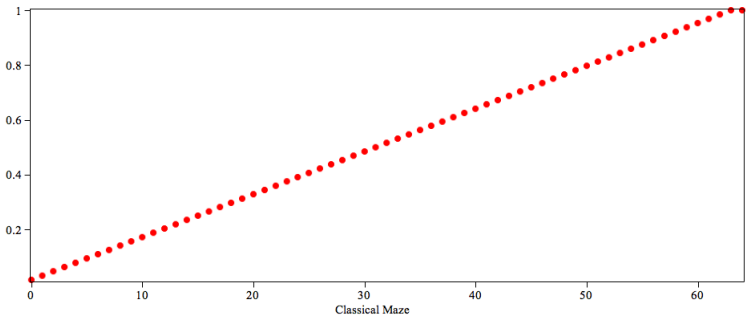



FIGURE 2.8: Classical Algorithm

However, in the quantum case, the odd increases polynomially, reaching almost to 1 after six queries. The speed-up of the quantum algorithm is intuitive.

$$Probability = \begin{cases} |\sin[(2k+1)\cdot arcsin(\frac{1}{\sqrt{N}})]|^2 & ,0 \leq k \leq \pi\frac{\sqrt{N}}{4} \\ \approx 1 & ,k > \pi\frac{\sqrt{N}}{4} \end{cases} \tag{2.8}$$

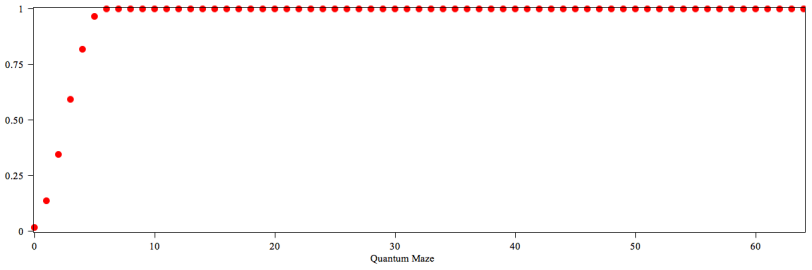



FIGURE 2.9: Quantum Algorithm

When we want to handle more complicated games, they can also be transformed into the graphs and using the same strategy we just used to analyze them. However, if the length of branches of the tree are different, i.e., the paths that Phil can choose are of different length, we need to transform them into the pattern which we already know how to handle: Considering "Virtual path extension" method, adding imaginary paths so that all paths at the same length. For example, figure 2.10 is an uneven graph representation, and figure 2.11 is the Virtual path extension corresponds to it. As you can see, the paths beneath the dashed line "imaginary" are all created by extension, but do not exist. When we make the measurement, the probabilities that we are at state $|100\rangle, |101\rangle, |110\rangle, |111\rangle$ are all zero. Please refer to the following table to see the

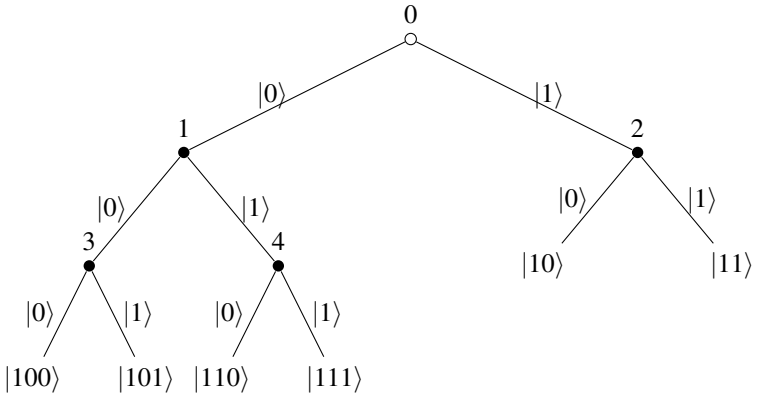



FIGURE 2.10: Example: Uneven path tree

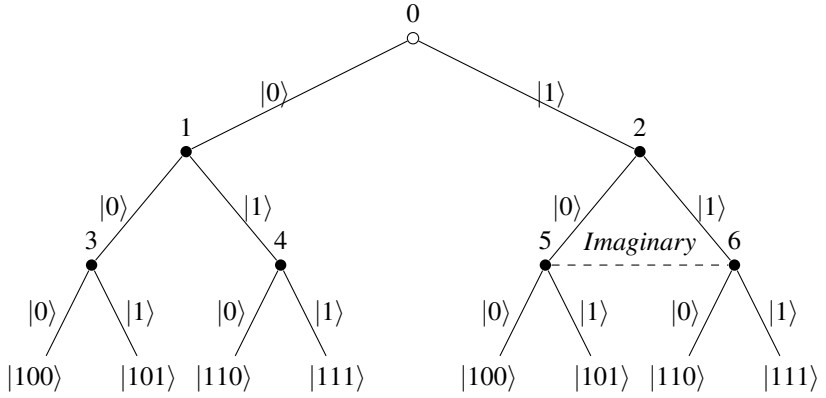



FIGURE 2.11: Example: Virtual path extension

distribution of probability at each state.

| *State* | *Probability* |
|---|---|
| $\|000\rangle$ | $1/8$ |
| $\|001\rangle$ | $1/8$ |
| $\|010\rangle$ | $1/8$ |
| $\|011\rangle$ | $1/8$ |
| $\|10\rangle$ | $1/4$ |
| $\|11\rangle$ | $1/4$ |

Results from the Quantum maze game:

1. There is a speed up when using Grover search algorithm compared with Depth-search algorithm. When $N = 64$, the quantum oracle only needs to reply 6 or 7 times (6.28) to "almost" guarantee Phil getting out of the maze, however, it takes 63 times for classical oracle. The speedup is more significant when $N$ is larger.

2. The probability of getting out of the maze is not 100% in most cases, since $L := \frac{\pi\sqrt{N}}{4}$ cannot give you an integer all the time. Therefore, when you do $\lfloor L \rfloor$, or $\lceil L \rceil$times iterations, the mixed state could have some probability of collapsing into the "wrong exit" when you make the measurement.

# Chapter 3

# Battleship

## 3.1 Introduction to the Battleship

Battleship is a guessing game for two players. It is played on ruled board on which each players fleet of battleships are marked. The locations of the fleets are concealed from the other player. A single query in standard battleship is a classical torpedo. Suppose we allow quantum queries, i.e., at each round, players can take turns either use "quantum torpedoes" to detect the location of the ships, or shooting "classical torpedoes" at the other player's ships to destroy them, and the objective of the game is to destroy the opposing player's fleet. The player who destroys the opponent's fleet first will win. For example, In round 1, player A uses quantum torpedoes to search row 2,4,6. And player B shoots classical torpedoes at A8 at his turn. Player A will get the information (location) collected by quantum torpedoes, and player B will destroy anything in the A8 grid.

We can transfer battleship game board into a grid, and the cost of detecting the ship is equivalent to the cost of searching the grid. And we want to compare several searching methods, in order to find the most effective strategy to win this game.

For example, figure 3.1 is the "Arena" for the two players. The boxes in grey are battleships placed by each player, and the grids which are marked with cross show the square that the opponent has fired upon.

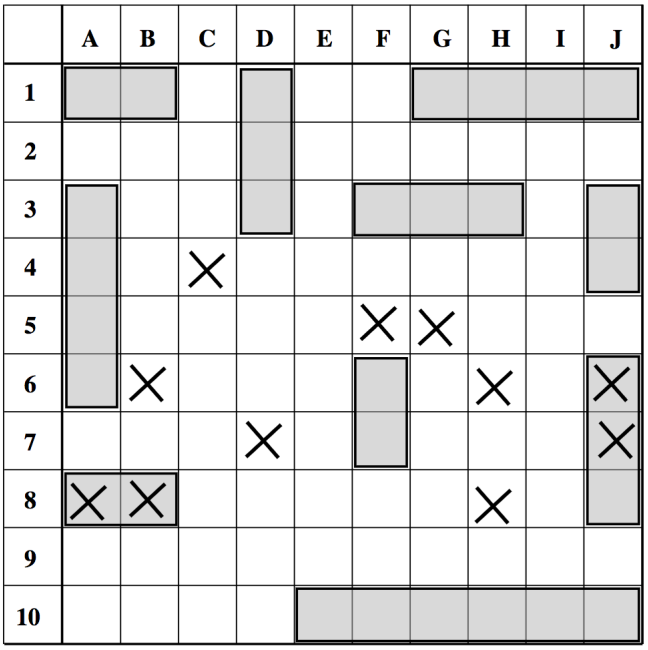



FIGURE 3.1: One player's no moving target battleship $10 \times 10$ grid

## 3.2 Some Game Strategies Based on Searching Grids

### 3.2.1 Grover Search

#### 3.2.1.1 Search One Target

Suppose you have $N^2$ grid squares, and only one ship, of length $L$. We use quantum torpedoes to search for a part of the ship - one random grid of the ship in our case, with the probability of $\frac{\pi\sqrt{N^2}}{4\sqrt{L}}$. After we confirmed the location of that part, we use one real torpedo to sink that part of the ship. The next step, we launch real torpedoes to sink the grids which are adjacent to the one we just detected. The following cases should be considered: Choose one of the two directions, i.e., along X-axis, or Y-axis. If there are one or two parts hit by the torpedoes, we keep shooting torpedoes along this direction until we sink $L$ grids (the length of the ship). If nothing was detected, then change to the other direction. Here is the algorithm:

```
Data{Pseudocode}
Result{Algorithm for Grover search}
T =1;
loc 1,2 = 0;
d= rand(0,1);
determine direction based on d;
launch real torpedoes to the direction and its opposite ;
move forward;
if loc 1 + loc 2 == 0 & T<=L
        change to the orthogonal direction;
elseif loc 1 + loc 2 == 2 & T<=L
```

```
        keep moving forward towards to these direction;
        T+=2;
        if loc 1 + loc 2 == 1 & T<=L
                moving forward to the direction which loc!=0;
                T+=1;
        else
                stop;
        end
else
        moving forward to the direction which loc!=0;
        T+=1;
end
```

#### 3.2.1.2 Search Multiple Targets

Suppose you have $N^2$ grid squares, and a total of $K$ squares are covered by the ships. For simplicity of the following computation, let us assume that all ships are of the same length, and regular shape (no turn over). It follows that $K = B * L$, where B = number of ships and L = length of the ship. One global Grover search takes $\frac{\pi N}{4\sqrt{K}}$ quantum torpedoes, will locate one part of the ship. i.e., using a quantum search algorithm to the whole grid, we can find of 1 part within those $B$ ships. Next, we use the strategy described in searching one target to sink the ship. Keep repeating these two steps until there are no opponent's ships left. Therefore, without doing anything smart, the total quantum torpedoes we use in this strategy is :

$$\begin{aligned}\mathbb{F}(a) &= \frac{\pi N}{4\sqrt{K}} + \frac{\pi\sqrt{(N^2-L)}}{4\sqrt{K-L}} + \frac{\pi\sqrt{(N^2-2L)}}{4\sqrt{K-2L}} + \cdots + \frac{\pi\sqrt{N^2-aL}}{4\sqrt{K-aL}} \\ &= \frac{\pi}{4}\left(\frac{N}{\sqrt{K}} + \frac{\sqrt{N^2-L}}{\sqrt{K-L}} + \frac{\sqrt{N^2-2L}}{\sqrt{K-2L}} + \cdots + \frac{\sqrt{N^2-aL}}{\sqrt{K-aL}}\right)\end{aligned}$$

where a = 1,2, $\cdots, B$. We want to find the upper bound of the range of this function $\mathbf{F}(a)$, and make some comparison. First we differentiate with respect to $a$ to the general term $\frac{\pi\sqrt{N^2-aL}}{4\sqrt{K-aL}}$, we have the following:

$$\begin{aligned}\frac{d\mathbf{F}}{da} &= \left(\frac{\pi\sqrt{(N^2-aL)}}{4\sqrt{K-aL}}\right)' \\ &= \frac{\pi}{4}\frac{\frac{-L\sqrt{(B-a)L}}{2\sqrt{N^2-aL}} + \frac{L(N^2-aL)}{2\sqrt{(B-a)L}}}{(B-a)L}\end{aligned}$$

Next, we take the numerator, ignoring the positive constant $\frac{\pi}{4}$, since it will not change the critical points or monotonicity. Then we solve for $\phi = \frac{-L\sqrt{(B-a)L}}{2\sqrt{N^2-aL}} + \frac{L(N-aL)}{2\sqrt{(B-a)L}} = 0$.

$$\begin{aligned}
\frac{-L\sqrt{(B-a)L}}{2\sqrt{N^2-aL}} &= \frac{L(N-aL)}{2\sqrt{(B-a)L}} \\
\frac{(B-a)L}{N^2-aL} &= \frac{N^2-aL}{(B-a)L} \\
(B-a)L &= N^2-aL \\
K &= N^2.
\end{aligned}$$

Since LB is K, which we assume to be much smaller than N, it follows that the numerator is greater than 0, and hence $\phi$ is increasing. We use the simple trick from calculus: the sum of all $n$ terms, is less or equal to $n\times$ Max$\{$terms$\}$. There are B terms in total, therefore, the total quantum torpedoes needed are strictly less than $\frac{\pi B\sqrt{N^2-(B-1)L}}{4\sqrt{L}}$. Comparing with $N^2/2$,

$$\begin{aligned}
\frac{\pi B\sqrt{N^2-(B-1)L}}{4\sqrt{L}} &= \frac{N^2}{2} \\
\frac{4N^4}{\pi^2B^2} &= \frac{N^2-(B-1)L}{L} \\
4x^2L - x\pi^2B^2 + (B-1)L\pi^2B^2 &= 0
\end{aligned}$$

Case 1: Find the roots of this polynomial if there are any, starting from the $\Delta$:

$$\begin{aligned}
N^2 &= \frac{\pi^2B^2 \pm \sqrt{(\pi^2B^2)^2 - 16L^2(B-1)\pi^2B^2}}{8L} \\
\pi^2B^2 &> 16L^2(B-1) \\
L^2 &< \frac{\pi^2B^2}{B-1}
\end{aligned}$$

Where we use the fact that $K = B\times L$ and $x = N^2$. We use $x_1, x_2$ to represent the roots, $x_1 < x_2$. It follows that the Grover's algorithm is more sufficient when $\{N^2 > x_2 \cup \Delta\} \cup \{N^2 < x_1 \cup \Delta\} = \mathbb{R}$.

Case 2: Suppose there are no roots, we need the minimum value of this convex function greater than 0. Therefore,

$$\frac{16L(B-1)L(\pi^2B^2) + \pi^4B^4}{16L} \geq 0.$$

Which is true for all $L$ and $B$. We can conclude Grover search algorithm is more efficient than the classical one regardless of the $L, B$ we choose. It also verifies why we get $\mathbb{R}$ in the case 1.

### 3.2.2 Crossroad Search

Crossroad search is a classical method, i.e., without using any quantum knowledge to search for the locations of the ships. Intuitively, crossroad search is to search every $k-1$ row in a $N \times N$ grid, and then get the location of the square which has the target, mark them with red. Next, we search the block near the red square(s), locating the rest parts of the ship.

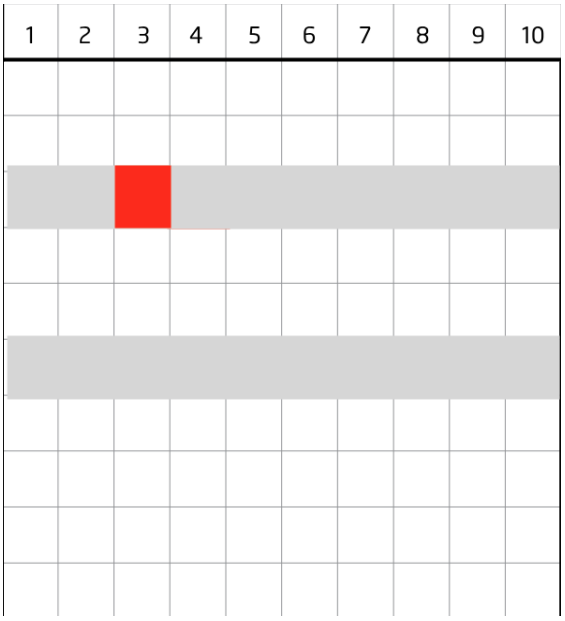



FIGURE 3.2: Crossroad search representation

#### 3.2.2.1 Search One Target

Suppose the size of the target ship is $1 \times k$. We search from the first row and then search every $k-1$ row in the grid. The probability of finding one piece of the ship is $N^2/k$, and then we use the following strategy: start from the location we found, search one grid further to any directions, if found a piece of the ship, keep going towards this direction until there is none in this direction, if the total $T$ is equal to the length of the ship $k$, we are done; if the $T$ is less than the length, we choose the direction which is the opposite to the original, and search for $k-T$ squares. If we did not find a piece of the ship, we choose another direction from the remaining three, and repeat the same process.

#### 3.2.2.2 Search Multiple Targets

The case we search for multiple targets are a little bit complicated. Suppose no two ships interact with each other, i.e., every ship need to keep the distance of one grid away from each other at

least to guarantee the safety. We can repeat the same process used in searching for target, and apply the strategy $B$ times, which is the total number of the ship.

```
Data{Pseudocode}
Result{Algorithm for Diagonal search}
T =1;
loc 1,2 = 0;
d= rand(0,1);
determine direction based on d;
launch real torpedoes to the direction and its opposite ;
move forward;
if loc 1 + loc 2 == 0 & T<=L
        change to the orthogonal direction;
elseif loc 1 + loc 2 == 2 & T<=L
        keep moving forward towards to these direction;
        T+=2;
        if loc 1 + loc 2 == 1 & T<=L
                moving forward to the direction which loc!=0;
                T+=1;
        else
                stop;
        end
else
        moving forward to the direction which loc!=0;
        T+=1;
end
```

### 3.2.3 Diagonal Search

Compared with crossroad search, the advantage of diagonal search is that it can detect the ship regardless of the direction. We apply Grover search to every $k-1$ diagonal of the grid. The ship length is $k$, and therefore have some odds that the ship will be hit twice. If the results fall in this probability space, we can easily launch real torpedoes towards the grids between those two locations. By doing this, we can guarantee to sink the ship, since those two locations detected by quantum torpedoes are exactly the two ends of the ship.

If only one grid is detected by the torpedoes, we will repeat a similar process as the Grover search method described above. This diagonal search method seems a smarter idea since we take advantage of the relationship between the length of the ship(L) and the interval between two searches (L-1).

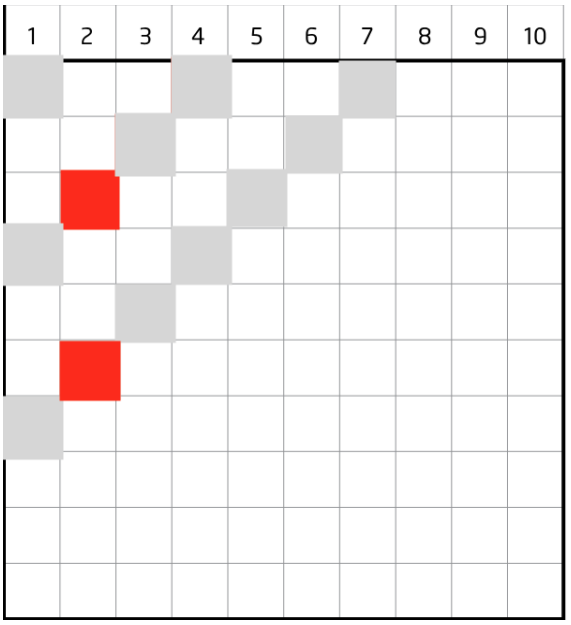



FIGURE 3.3: Diagonal search representation

#### 3.2.3.1 Search Multiple Targets

$$\begin{aligned}\mathbf{F}(a) &= \frac{\pi\sqrt{\frac{N^2}{L}}}{4\sqrt{B}} + \frac{\pi\sqrt{\frac{N^2}{L}-L}}{4\sqrt{B-1}} + \frac{\pi\sqrt{\frac{N^2}{L}-2L}}{4\sqrt{B-2}} + \cdots \\ &= \frac{\pi N}{4\sqrt{K}} + \frac{\pi\sqrt{N^2-L^2}}{4\sqrt{K-L}} + \frac{\pi\sqrt{N^2-2L^2}}{4\sqrt{K-2L}} + \cdots + \frac{\pi\sqrt{N^2-aL^2}}{4\sqrt{K-aL}} \\ &= \frac{\pi}{4}\left(\frac{N}{\sqrt{K}} + \frac{\sqrt{N^2-L^2}}{\sqrt{K-L}} + \frac{\sqrt{N^2-2L^2}}{\sqrt{K-2L}} + \cdots + \frac{\sqrt{N^2-aL^2}}{\sqrt{K-aL}}\right).\end{aligned}$$

There is something that needs to be noticed. When using the diagonal search, we reduce the total number from $N^2$ to $N^2/L$. However, by doing this, we reduce the number of marked squares from $K = BL$ to $B$. After simplification, we get the same representation for the usage of quantum torpedoes as the Grover search which we discussed in the last section. We can conclude that this diagonal search method is not better than the general Grover search.

## 3.3 Conclusion and Comparison

From this simple game strategy comparison, we can easily find out the advantages of a quantum algorithm. Grover search and Diagonal search are better strategies than crossroad search when $N$ is big enough. However, there is no outperformance in Diagonal search compared with Grover search, because when we reduce the sample, we reduce the probability to find the target as well.

Are there any better strategies to win the Battleship? The answer is yes for sure, since we did not consider the ship as a whole in the quantum algorithms, and this is a crucial fact. The ships are consecutive, so we can sink the ship using classical torpedoes after we "detect" one. What if there is a quantum algorithm considers the consecutive property rather than just searching for unordered grids (database)? I believe that will outperform the Diagonal search or Grover search strategy.

When only considering the quantum queries, the results from the Battleship game:

1. The Crossroad search algorithm is the least efficient since it does not use quantum theory.
2. There is some probability to hit the ships twice, by searching every $L-1$ diagonal.
3. The Diagonal and the Grover search algorithms do perform better than the classical Crossroad search algorithm. However, the diagonal search has no outperformance compared with Grover search algorithm, follows by the proof in this chapter.

# Chapter 4

# Orientable Surface Identification

We can exploit quantum properties to identify the Orientable surfaces and Non-orientable surfaces. If we throw a random walk matrix many times on these surfaces, we can not tell if it is a Torus or Klein bottle, since the probability at each location is the same[1]. However, if we use the quantum walk matrices, the probability vector will be different because of quantum interference. i.e., the probability will cancel out at some location, and make the odds to be zero. We could say that the likelihood of the particle at some states is 0 in the long run. Therefore, we can identify which types of a surface it belongs to based on the outcome vectors.

## 4.1 Random and Quantum Walk

The idea of the quantum walk is motivated by classical random walk, and the quantum walk is a part of many quantum algorithms, such as Grover search algorithm. The advantages of the quantum walk are that it can provide polynomial or even exponential speedup over classical algorithms. It is well known that random walk will converge to a Normal distribution by Central Limit Theorem. In contrast, the quantum walk will exhibit different features because of quantum interference. To see the comparison intuitively, we can use the following graph conducted by 50 times coin flips. Similar to the classical random walk, a quantum walk can be classified into two parts: Continuous time and discrete time. In this part of the thesis, we mainly focus on the discrete time of quantum walk, and it can be considered as a "coin flip" repeatedly. Space can be represented as,

$$|\Psi\rangle = |s\rangle \otimes |\psi\rangle \tag{4.1}$$

internal spin space is the following

$$|s\rangle \in \mathbb{H}_C = \{|\uparrow\rangle, |\downarrow\rangle\} \tag{4.2}$$

[1] We learnt this from the Markov chain property.

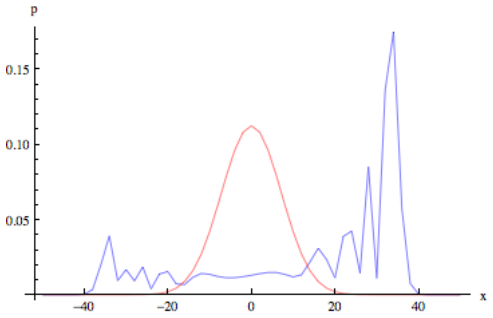



FIGURE 4.1: Quantum and Random Walk comparison

and a position state

$$|\psi\rangle \in \mathbb{H}_P = \left\{ \sum_{x\in\mathbb{Z}} \alpha_x |x\rangle : \sum_{x\in\mathbb{Z}} |\alpha_x|^2 < \infty \right\} \tag{4.3}$$

where $\mathbb{H}_C = \mathbb{C}^2$ is the "coin space" and $\mathbb{H}_P = \ell^2(\mathbb{Z})$ is the space of quantum position states. Let us consider two cases: First, quantum walk on a line; Second, quantum walk on a grid.

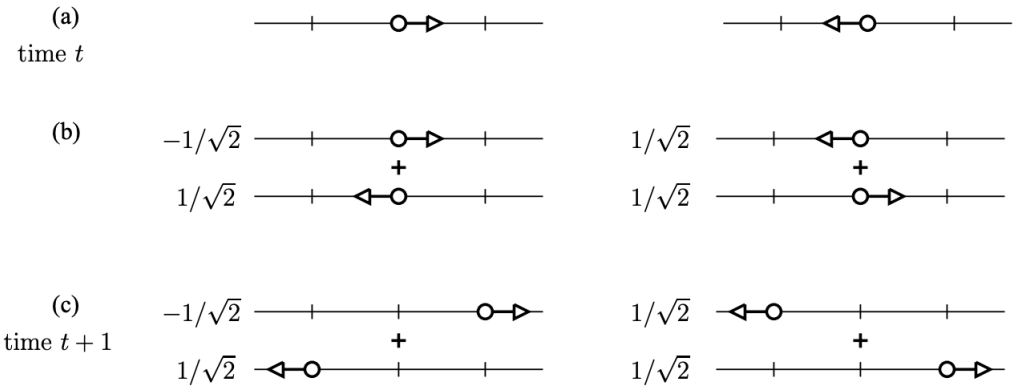



FIGURE 4.2: Quantum Random Walk on a line

As shown in the state (a), we list two possible moving direction of a particle on a line, and then the Hadamard coin will create the superposition, as can be seen in the state (b). When it comes to time t+1, the particle moves to two directions simultaneously with equal probability$(\sqrt{1/2})^2 = \frac{1}{2}$.

Quantum walk on a two-dimensional grid is nothing but just changing two possible moving directions into four. i.e., apply quantum walk on the horizontal line and vertical line simultaneously. At each period (b) corresponds with the state (a), the particle has an equal probability to go to the two directions.


FIGURE 4.3: Quantum Random Walk on a grid

## 4.2 Torus

In this section, we are going to talk about how to use the quantum algorithm to identify the Klein bottle and Torus. Since Torus is orientable, but Klein bottle is not, so we can base on the outcome from quantum walk to find out which kind of surface it belongs, i.e., if the probability at each location is the same, then it is a Non-orientable - Klein bottle, Torus otherwise.

A Torus is a surface or solid formed by rotating a closed curve, especially a circle, around a line that lies in the same plane but does not intersect it (e.g., like a ring-shaped doughnut). Please refer to figure 4.5. Let us see an example of the classical transition matrix, and we will discuss how we come up with this later.

$$\mathbb{T}_c = \frac{1}{4} \cdot \begin{array}{c} \text{state} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array} \tag{4.4}$$

FIGURE 4.4: Classical transition matrix $T_c$ for 9 nodes Torus

Every closed surface can be constructed from the fundamental polygons representation of the surface. One can abstract the Torus and the Klein Bottle as the following. Based on the definition of each surface, we can break the 3-D surface arrows into 2-dimensional square, with directions on each edge. One way to reconstruct the surface is to attach the opposite side and match the arrow to point in the same direction. Take Klein bottle as an example. We first attach the vertical sides, next, twist the horizontal sides and glue them. Figure 4.5 is the polygon representation of the Torus and Klein bottle.

We define the transition matrix T, superposition matrix Q, and initial vector S in each case. Note that T is a coin space and is different every time, since it not only depends on the surface we

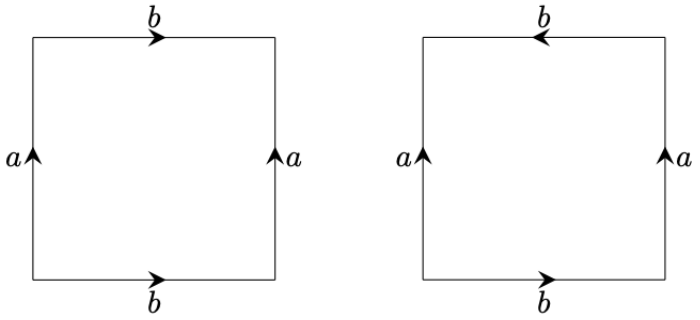


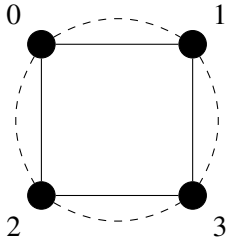

FIGURE 4.5: Torus and Klein bottle

chose but also depends on which algorithm we are using. i.e., the classical transition matrix $T_c$, and quantum transition matrix $T_q$ are not the same, actually, they differentiate a lot. Second, superposition matrix Q is the same for both orientable and non-orientable surfaces. The reason is simple: when we are using quantum algorithms, the superposition matrices are the same, as long as they have the same number of nodes representation (we will introduce later). The identification process is as follows, and we call it the "Standard machine."

Standard machine:

1. Given the surface, label each nodes with numbers.
2. Find $T_q$, Q , $S$.
3. Find out the probability at each state by computing $(T_q \cdot Q)^n \cdot S$.
4. Based on the result, we can tell if the given surface is orientable or not.

Let us start with a simple example - Torus. Suppose we only use four nodes to simulate the structure of the Torus, labelling them from 0 to 3, consecutively from left to right.




The first qubit represents the location (0-4), and the last two qubits decide the direction. i.e., the second qubit represents if it goes West/East, or North/South, the following third qubit represents the specific direction based on what we get from the second qubit. For each location form (0-4), we flip two Hadamard coins consecutively and match all four possible combinations with four

possible directions. Finally, we list them in the following table.

| $x$ | $f(x)$ |
|---|---|
| 00 | *West* |
| 01 | *East* |
| 10 | *North* |
| 11 | *South* |

(4.5)

In quantum computing, qubits move to all possible directions at the same time, and we can use this to build the transition matrix $T_q$ by filling it with 1s and 0s. For example, If you start from the first node, denoted by $|0\rangle$, we get 01 from the coin flip, then the particle will go to the east direction based on the direction table. We will end up with the state $|101\rangle$, so we put 1 at the corresponding entry in the quantum transition matrix $T_q$. If we get 10 from the coin flip, the particle will go North, we will get $|210\rangle$, go west, we have $|100\rangle$, and last, if the particle goes south, it will locate at state $|211\rangle$. Similarly, we repeat this process to find the outcome state from the rest three nodes, and fill the matrix $T_q$ with 1s and 0s, where 1 in the matrix represents that there is some possibility that the qubit in the state corresponds with the column will be in that state corresponds with the row after the superposition happens.

$$
\begin{array}{c}
 \\ |000\rangle \\ |001\rangle \\ |010\rangle \\ |011\rangle \\ |100\rangle \\ |101\rangle \\ |110\rangle \\ |111\rangle \\ |200\rangle \\ |201\rangle \\ |210\rangle \\ |211\rangle \\ |300\rangle \\ |301\rangle \\ |310\rangle \\ |311\rangle
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccccccc} |000\rangle & |001\rangle & |010\rangle & |011\rangle & |100\rangle & |101\rangle & |110\rangle & |111\rangle & |200\rangle & |201\rangle & |210\rangle & |211\rangle & |300\rangle & |301\rangle & |310\rangle & |311\rangle \end{array} \\
\begin{pmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

FIGURE 4.6: Quantum Transition matrix $T_q$ for 4 nodes Torus

*Remark* 4.1. There is exactly one "1" in each row and column in figure 4.6.

As for the matrix $Q$, in general, the matrix $(4n \times 4n)$ is used to create superposition, where $n$ represents the number of nodes on the given surfaces. The partitioned matrix is either 0, or $H \otimes H$, and $H \otimes H$ only appears on the diagonal.

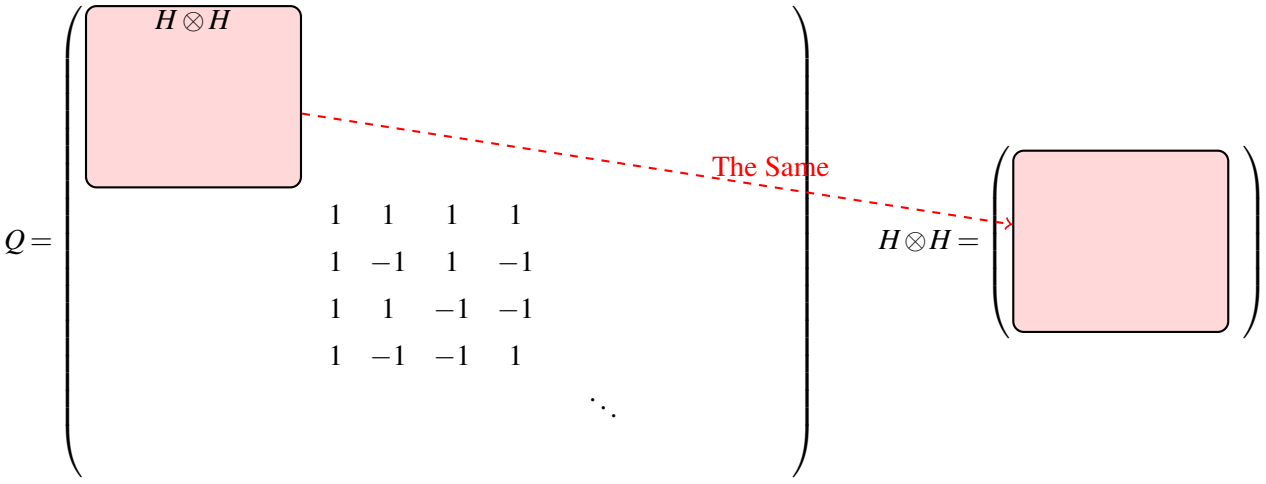



FIGURE 4.7: Superposition matrix $Q$

In this 4 nodes Torus, $Q$ is a 16 by 16 matrix with 4 blocks on the diagonal. Within each block, we have $H \otimes H$.

$$H \otimes H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 \end{pmatrix}$$

However, in the classical case, the transition matrix is different from the quantum one. Actually, the classical transition matrix $T_c$ is a lot easier compared with the $T_q$. To be more specific, the classical bit has no superposition, so it can only travel to one of the possible outcomes rather than all of them. Therefore, the bit has equal probability (1/4) to go to location 0, 1,2, and 3. The matrix representation is the following:

$$\mathbb{T}_c = \frac{1}{4} \cdot \begin{array}{c} \text{state} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \\ 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \end{pmatrix} \end{array} \tag{4.6}$$

FIGURE 4.8: Classical transition matrix $T_c$ for 4 nodes Torus

*Remark* 4.2. The only two possible locations for a particle travelling from node 0 is 1 and 2, because, in 4 nodes Torus, the particle moves West or East will both land on 1; And those going to North or South both lands on 2.

Similarly, going through the same process, we can find out the transition matrix $T_c$ for Torus for 9 nodes. The shape of the surface is shown in figure 4.14.

Lastly, we choose $S$, the initial position of the particle. Usually we fill the column vector $S$ with 1s and 0s, and put 1 at the first entry. For example, $S = [\underbrace{1,0,0,\cdots 0}_{\text{15 0s}}]^T$ is a initial position vector for 4 nodes surfaces, the particle starts at $|000\rangle$. We can also choose other $S$, but it will influence the final result sometime. Both the structure of the surface and the initial location of the particle matter when we look at the long term pattern in the quantum walk. We will make a comparison at the end of this chapter.

$$\mathbb{T}_c = \frac{1}{4}\cdot \begin{array}{c} \text{state} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array} \tag{4.7}$$

FIGURE 4.9: Classical transition matrix $T_c$ for 9 nodes Torus

## 4.3 Klein Bottle

In topology, the Klein bottle is an example of a non-orientable surface; it is a two-dimensional manifold against which a system for determining a normal vector cannot be consistently defined. In other words, it is a one-sided surface which, if travelled upon, could be followed back to the point of origin while flipping the traveller upside down. Other related non-orientable objects include the Möbius strip and the real projective plane, which we will introduce later.

| | $\|000\rangle$ | $\|001\rangle$ | $\|010\rangle$ | $\|011\rangle$ | $\|100\rangle$ | $\|101\rangle$ | $\|110\rangle$ | $\|111\rangle$ | $\|200\rangle$ | $\|201\rangle$ | $\|210\rangle$ | $\|211\rangle$ | $\|300\rangle$ | $\|301\rangle$ | $\|310\rangle$ | $\|311\rangle$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|000\rangle$ | | | | | 1 | | | | | | | | | | | |
| $\|001\rangle$ | | | | | | | | | | | | | | 1 | | |
| $\|010\rangle$ | | | | | | | | | | | 1 | | | | | |
| $\|011\rangle$ | | | | | | | | | | | | 1 | | | | |
| $\|100\rangle$ | | | | | | | | | 1 | | | | | | | |
| $\|101\rangle$ | | 1 | | | | | | | | | | | | | | |
| $\|110\rangle$ | | | | | | | | | | | | | | | 1 | |
| $\|111\rangle$ | | | | | | | | | | | | | | | | 1 |
| $\|200\rangle$ | | | | | | | | | | | | | 1 | | | |
| $\|201\rangle$ | | | | | | 1 | | | | | | | | | | |
| $\|210\rangle$ | | | 1 | | | | | | | | | | | | | |
| $\|211\rangle$ | | | | 1 | | | | | | | | | | | | |
| $\|300\rangle$ | 1 | | | | | | | | | | | | | | | |
| $\|301\rangle$ | | | | | | | | | | 1 | | | | | | |
| $\|310\rangle$ | | | | | | | 1 | | | | | | | | | |
| $\|311\rangle$ | | | | | | | | 1 | | | | | | | | |

FIGURE 4.10: Quantum transition matrix $T_q$ for 4 nodes Klein bottle

The classical transition matrix $T_c$ for 16 nodes Klein bottle is as the following.

$$\mathbb{T}_c = \frac{1}{4} \cdot \begin{array}{c} \text{state} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{4.8}$$

FIGURE 4.11: Classical transition matrix $T_c$ for 16 nodes Klein Bottle

## 4.4 Projection Plane

A projective plane, a twisted sphere is a no boundary surface derived from a usual plane by adding a line at the infinity. For example, a projective plane can be constructed by gluing the opposite edges of the rectangle, and twist each pair of edges. We use the same strategy to build the quantum transition matrix for the projection plane:

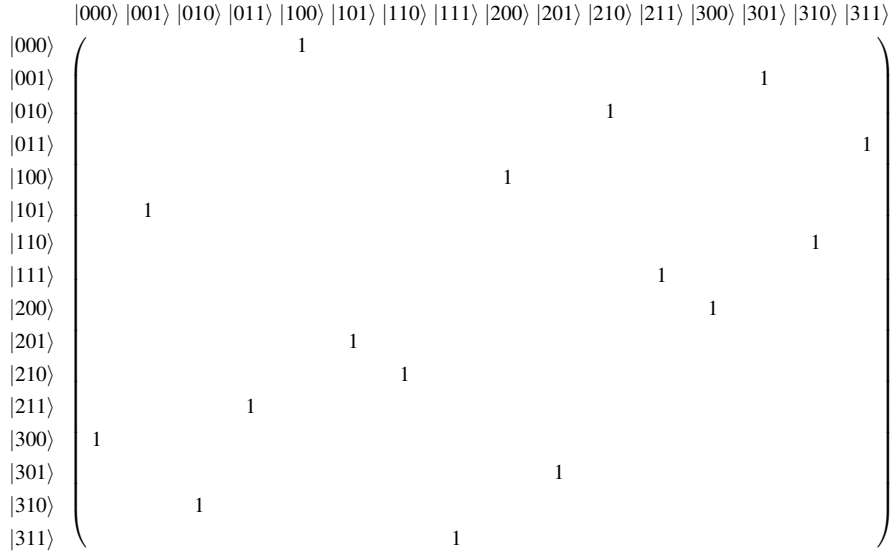



FIGURE 4.12: Quantum transition matrix $T_q$ for 4 nodes Projective plane

In the case of projection plane, the 9 nodes classical random walk transition matrix is :

$$\mathbb{T}_c = \frac{1}{4} \cdot \begin{array}{c} \text{state} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array} \tag{4.9}$$

FIGURE 4.13: Classical transition matrix $T_c$ for 9 nodes Projective plane

Let S be $[1,0,0,\cdots,0]^T$ We apply $(T \cdot P)^n$ to $S$, we will get the sate distribution at each location. The outcome vector from the classical random walk is simply the amount of time the particle spends at each location. However, the probability at each state is the square of the element corresponds with the quantum outcome vector.

The graph will look like figure 4.15 when there are 16 nodes, and we give the examples for the

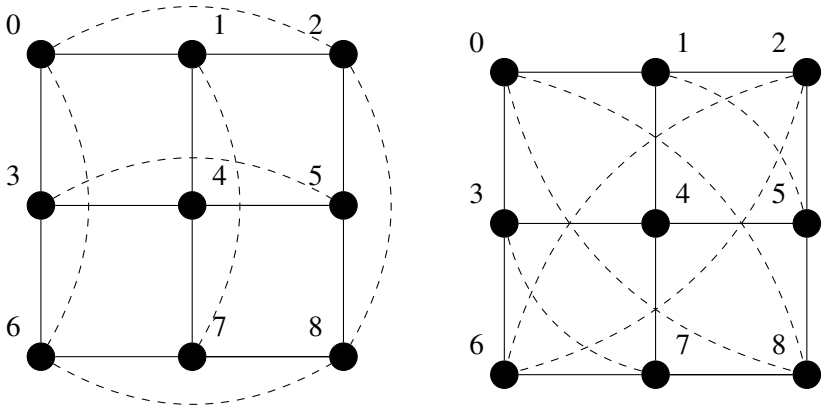



FIGURE 4.14: Torus and Klein Bottle

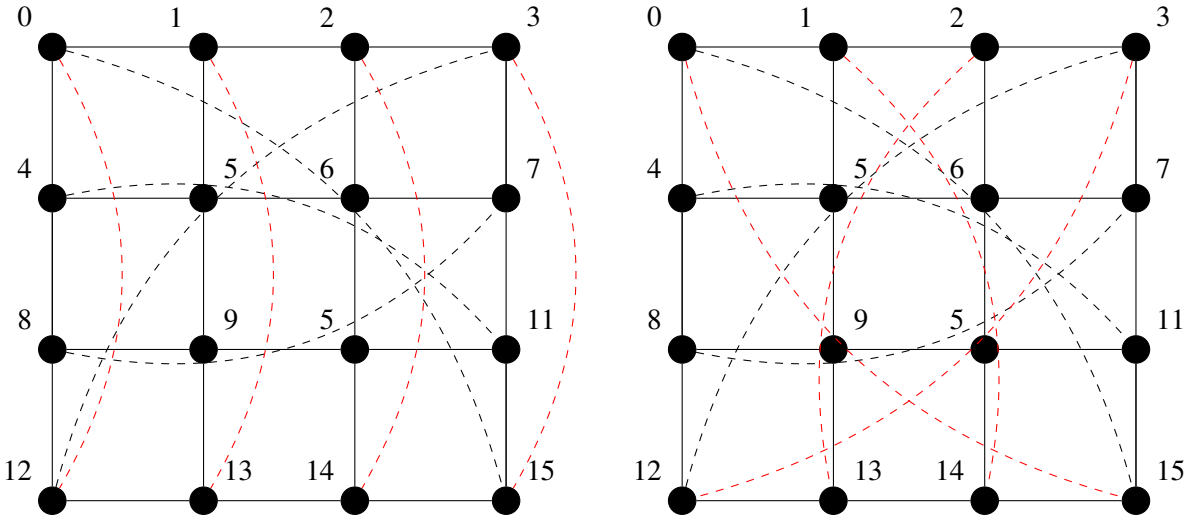



FIGURE 4.15: Klein Bottle and Projective-plane

Klein Bottle and Projective-Plane. Readers can easily construct the case for Torus since it is very similar to the one which has four nodes.

What if the number of nodes approaches to infinity? Or we can say, using as many nodes as possible to simulate the structure of the surface. In this limiting case, the discrete quantum walk will become a continuous quantum walk.

## 4.5 Comparison Between the Surfaces

In order to have a more intuitive feeling of how quantum interference makes a difference, but without being too complicated, we choose four nodes in each case. After many iterations, the probability distribution at each node is exhibited in table 4.1:

The classical transition matrix for four nodes Torus, Klein bottle, and projective plane are 4.10, 4.11, 4.12 respectively.

$$\mathbb{T}_c = \frac{1}{4} \cdot \begin{array}{c} \text{state} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \\ 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \end{pmatrix} \end{array} \tag{4.10}$$

$$\mathbb{T}_c = \frac{1}{4} \cdot \begin{array}{c} \text{state} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix} \end{array} \tag{4.11}$$

$$\mathbb{T}_c = \frac{1}{4} \cdot \begin{array}{c} \text{state} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{pmatrix} \end{array} \tag{4.12}$$

And recall from figure 4.6, 4.10, 4.12, we have the quantum transition $T_q$ for each of them, using either $T_c^n \cdot S$ to find the long term pattern for classical random walk, or using $(T_q \cdot Q)^n \cdot S$, to find out the long term pattern of quantum walk. In these experiments, I chose $n = 500$.

Some people may wonder why there are negative values in the outcome vector. Please notice that the number is an entry in the space of quantum position states $\mathbb{H}_p$ instead of the probability. Let us think about a particle wandering around the surface, because of this negative value appears at some state, which will cancel out some positive value at the same state, and hence make the probability different from the one from the random walk.

Results from the experiment:

1. Klein bottle has the same pattern no matter what the initial position vector we choose.
2. The outcome result for Projective plane will change when initial position vector changes.
   (a) The particle has 0 probability at some other states rather than $|101\rangle, |110\rangle$ as indicated in our experiment in the long run.
   (b) The probability will not converge at location $0, 1, 2, 3$. i.e., the probability will not approach to a fixed number as the number of trials is large.
3. We could identify if the given surface is orientable or not based on the outcome result through the "Standard machine."

| n = 500 | Quantum Walk | Random Walk |
|---|---|---|
| Klein Bottle | $[0.5,0,0,0,0.5,0,0,0,-0.5,0,0,0,0.5,0,0,0]^T$ | $[0.25,0.25,0.25,0.25]^T$ |
| Torus | $[0.0755,0.2434,0.0201,-0.1486,0.1058,0.0747,$ $0.0739,0.4030,0.2644,0.4038,0.2963,0.0755,$ $0.6045,-0.1167,-0.0201,0.1486]^T$ | $[0.25,0.25,0.25,0.25]^T$ |
| Projective plane | $[0.0785,0.3429,-0.0857,-0.2811,0.2572,-0.0000,$ $0.0000,0.2572,0.3832,-0.25720.2572,-0.3503,$ $0.2811,-0.0857,-0.1716,0.3741]^T$ | $[0.25,0.25,0.25,0.25]^T$ |

TABLE 4.1: Result comparison between different surfaces

# Chapter 5

# Conclusion and Enlightment

In this thesis, we first introduced some basic knowledge about Quantum computing, and proposed quantum diagonal search algorithm as a strategy in the Battleship game. Lastly, we used the quantum walk on the grids to identify the orientable surfaces or not. Here are some suggestions for the future research work based on this project: First, develop a quantum algorithm specially for the Battleship game, taking the consecutive property of the ship into consideration, rather than mainly using Grover search algorithm. We could consider using Grover partial search algorithm [5]. Second, we could use the quantum walk to identify among non-oriented surfaces by observing the outcome distribution vector at each state. The target in our Battleship game is all static, what if they can move? i.e., we use quantum algorithms to search for moving target. Suppose we use $n$ quantum queries before the ship moves once, what are the odds of finding the target? The general idea is the following, instead of rotating towards the target $|\omega\rangle$ directly, we rotate the plane towards the target. It is impossible for classical computer if $n = 1$. We believe it is achievable using quantum algorithms. Now, it is your turn to do some experiment and put all the evidence that you found into a coherent story! Good Luck!

# Chapter 6

# Bibliography


[1] GROVER L.K 1996. A fast quantum mechanical algorithm for database search. *Symp. of Theory of Computing*, p212.

[2] P.R. GIRI ,AND V.E. KOREPIN 2016. A Review on Quantum Search Algorithms. *Quantum Inf. Process*, Vol. 16 No. 12 (2017) 1-36.

[3] MIKLOS SANTHA 2008. Quantum walk based search algorithms. *5th Theory and Applications of Models of Computation*, LNCS 4978, 31-46.

[4] V.E. KOREPIN ,AND L.K. GROVER 2006. Simple algorithm for partial quantum search. *Quantum Inf. Process*, 5:5-10.

[5] K. ZHANG ,AND V.E. KOREPIN 2018. Quantum partial search for uneven distribution of multiple target items. *Quantum Information Processing*, 1573-1332.

[6] BS. CHOI ,AND V.E. KOREPIN 2007. Quantum Partial Search of a Database with Several Target Items. *Quantum Inf Process*, 6: 243.

[7] BS. CHOI ,AND T.A. WALKER, 2007. Sure Success Partial Search. *Quantum Inf Process*, 6: 1.

[8] T. BYRNES ,G. FORSTER, AND L. TESSLER 2018. Generalized Grover's Algorithm for Multiple Phase Inversion States. *Physical Review Letters*. 120. 10.1103.

[9] S. IRIYAMA ,AND M. OHYA 2012. Computational complexity and applications of quantum algorithm. *Applied Mathematics and Computation*. 218, 8019-8028.

[10] J.BANG ,J.RYU ,AND C. LEE, ET AL. 2012. A quantum heuristic algorithm for the travelling salesman problem. *Journal of the Korean Physical Society*. 61: 1944.

[11] LEEUWEN ,AND V. JAN, ET AL. 1998. Handbook of Theoretical Computer Science. Vol. A, Algorithms and complexity. Amsterdam: Elsevier.

# Appendix A

# An Appendix

## A.1 Depth-First Search

Depth-first search steps:

1. If you have multiple paths, choose anyone and move forward.
2. Keep choosing a path which you have not seen so far till you exit the maze or reach a dead end.
3. If you exit maze, you are done.
4. If you reach the dead end, so this is the wrong path. Take one step back, and choose a different path. If all paths are seen in this node, take one step back and repeat.