Quantum Random Walk Search and Grover's Algorithm - An Introduction and Neutral-Atom Approach

A Senior Project

presented to

the Faculty of the Physics Department

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science

Computer Engineering

by

Anna Maria Houk

Advisor, Dr. Katharina Gillen

June 23, 2020

Contents

1 Introduction

Modern computing has become increasingly powerful, but there are still hard limits on the the time complexity and problems that can be solved. Some of these limits can however, be breached by quantum computers, which use quantum mechanical properties to encode information into quantum particles referred to as qubits and perform computations. In the sub-field of quantum algorithms, physicists and computer scientist take classical computing algorithms and principles and see if there is a more efficient or faster approach implementable on a quantum computer, i.e. a "quantum advantage".

In classical computing, random walks are defined as a sequence of random steps on a mathematical space and are used in computer science and other fields to model natural phenomena. Random walks are a special case of a stochastic model (i.e. a random mathematical process) known as Markov chains, which is a process during which predictions can be made at any state in the process without knowledge of the past states with the same accuracy [1]. There are already many classical applications of random walks in fields such as reinforcement learning, random number generation, and thermodynamics to name a few, so it is a natural question of interest to if there is a similar usefulness for random walks implemented with quantum computers. Exploration of continuous and discrete-time quantum random walks and classical random walks with discrete state-space have shown that there is a quantum advantage and significant differences in the behavior between the classical and quantum random walks [2] [3].

In this paper we will begin by taking classical examples of random walks and move them into the quantum computing paradigm, as well as introduce a popular quantum search algorithm called Grover's search. There are currently many methods being explored for creating quantum computers and qubits, but in Section 3, we will look at using neutral-atom (also known as cold-atom) quantum computing to physically implement the random walks and Grover's algorithm. Lastly, using similar principles to Grover's, we will explore a possible application of quantum random walks as a search algorithm.

2 Background

2.1 Discrete Quantum Random Walks

The classic example of a discrete random walk is a walk along a number-line. Imagine a number-line and a coin, depending on the flip of the coin the position will move one step to the left or right on the

number line. At each step in the walk there is an equal probability that the next position will be either the previous position or the next integer on the number line and transition probabilities are only dependent on the current step. If the probability of ending up at each position is looked at after *t* steps, it will approach a Gaussian distribution at a large number of iterations *T*, centered around zero (the initial position) and a variance of $\delta^2 = T$ i.e. an average distance $\sqrt{T}$ from the initial position [3].

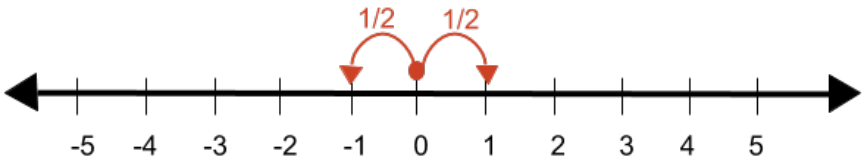


Figure 1: A one-dimensional random walk

Taking the number-line and coin flip random walk model to the quantum paradigm, the number-line can be expressed as a one-dimensional space being traversed by a spin-$\frac{1}{2}$ particle [2]. The position of the particle is the substitute for the position of the number-line and the spin of the particle takes the place of the coin. Study of the mathematics of quantum mechanics shows that the walk of the particle is described by the unitary operator written in such a way to separate the spin and position state spaces:

$$U = e^{-2iS_z \otimes Pl} \tag{1}$$

More information about this operator can be found in much greater depth in [4], but only a surface-level understanding is needed for this paper, specifically the separability of the spin and position state spaces and how they are utilized in a random walk. Above, *P* is the momentum operator and *l* is the constant step length with each iteration to create a discrete state-space. S_z is the operator that represents the z-component of the spin of the particle and is written as:

$$S_z = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \frac{1}{2}(|\uparrow\rangle\langle\uparrow| - |\downarrow\rangle\langle\downarrow|) \tag{2}$$

The operator is applied to the particle described by the wave function in the initial state, $|\psi_0\rangle = (\alpha|\uparrow\rangle + \beta|\downarrow\rangle)|\psi_{x0}\rangle$, where $|\psi_{x0}\rangle$ is the the initial position of the particle and the wave function

is in equal superposition for the spin ($\alpha = \beta = \frac{1}{\sqrt{2}}$). Applying the unitary operator results in:

$$U\left|\psi_0\right\rangle = \alpha\left|\uparrow\right\rangle\left|\psi_{x0-l}\right\rangle + \beta\left|\downarrow\right\rangle)\left|\psi_{x0+l}\right\rangle \tag{3}$$

With each application of U there is a probability $p_{left} = |\alpha|^2 = \frac{1}{\sqrt{2}}$ that the spin state is $|\uparrow\rangle$ and the particle will move l to the left and a probability $p_{right} = |\beta|^2 = \frac{1}{\sqrt{2}}$ that the spin state is $|\downarrow\rangle$ and the particle will move l to the right. Note that there is an equal probability of being in spin state $|\uparrow\rangle$ or $|\downarrow\rangle$, much like flipping a coin has an equal chance of landing heads or tails. In quantum computing the state-space, or registers, of a computer are described as a Hilbert space. Therefore, we will rewrite the quantum walk model in terms of the "registers" of a quantum machine in order to use the quantum random walk computationally.

Take the discrete space travelled by the particle to be of length N. Each position will be a basis state and the last position, $|N-1\rangle$, leads to position $|0\rangle$, so the space is essentially a circle. The Hilbert space that the discrete quantum walk exists in can be again be separated into the spin and state space:

$$H = H_c \otimes H_p \tag{4}$$

Where H_p is the size N positional Hilbert space, $H_p = \{|i\rangle : i = 0, 1, 2, ..., N-1\}$, and H_c is the coin space consisting of the two spin states, $H_c = \{|\uparrow\rangle, |\downarrow\rangle\}$. With these definitions, the operation in Equation (1), which is responsible for the direction of movement of the particle, can be written as a new unitary operator:

$$S = |\uparrow\rangle\langle\uparrow| \otimes \sum_{i=0}^{N-1} |i+1\rangle\langle i| + |\downarrow\rangle\langle\downarrow| \otimes \sum_{i=0}^{N-1} |i-1\rangle\langle i| \tag{5}$$

Now a mechanism is needed to "flip the coin". A common choice is to use a single-qubit gate referred to a Hadamard gate

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}}(|\uparrow\rangle\langle\uparrow| + |\uparrow\rangle\langle\downarrow| + |\downarrow\rangle\langle\uparrow| - |\downarrow\rangle\langle\downarrow|) \tag{6}$$

which will give an equal probability of being in either basis state, given the walk starts in the $|1\rangle$ basis state due to underlying quantum properties of the Hadamard coin [6]. There are a lot of unique characteristics of this coin [5] that are just the tip of the iceberg in the difference between classical and quantum walks. Putting all the pieces together, the model for a discrete quantum walk using a Hadamard gate as

a coin is the unitary

$$U_{walk} = S \cdot (H \otimes I) \tag{7}$$

If U_{walk} iterates T times, as done with the classical model, the behavior differs immensely. The resulting distribution does not come to approximate a Gaussian distribution and instead has a close to uniform spread over the interval $[-\frac{T}{\sqrt{2}}, \frac{T}{\sqrt{2}}]$ and has a variance of $\delta^2 = T^2$ i.e. an average distance T from the initial position. Figure 2 is depicts the probability distribution of a 100-step walk from [2].

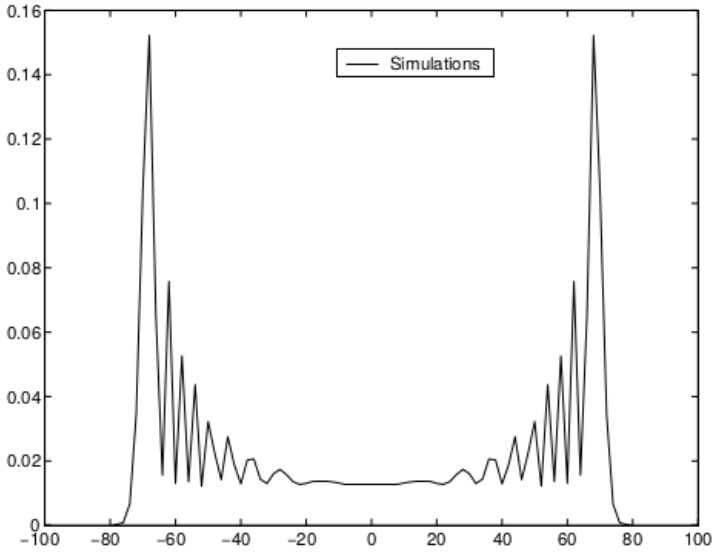


Figure 2: The probability distribution of a quantum walk using a Hadamard gate as the coin

2.2 Continuous Quantum Random Walks

The continuous random walk has a similar idea but does not require the extra dimension of having a coin space. Starting with the classical parallel, the continuous-time Markov chain is described by the discrete state space (as done in the discrete walk case) and a transition rate matrix which gives the "jumping rate" or probability of transition between each state in the state-space. Once the classical continuous walk is understood, we will use these ideas [7] to construct the walk as a decision tree and make a quantum operator to traverse it. Thinking about moving on a number line, the transition matrix created would contain the probability of going from position *i* to position *j*. As it is only possible to move one unit at

each position on the number line, the probability of a transition is $\frac{1}{2}$ for $j = i \pm 1$ and 0 for all other scenarios. The transition matrix M for jumping on a number line (that circles around) of five positions is as follows:

$$M = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{8}$$

Written generally for n-number of positions on a number line:

$$M_{ij} = \begin{cases} \frac{1}{2}, & j = i \pm 1. \\ 0, & j \neq i \pm 1. \end{cases} \tag{9}$$

While the transition matrix stays the same no matter what point in time, the probability of ending up at position *j* from position *i* after T will evolve at each step, but given the independence of each step, the probability at each time-step, p_t, will only depend on the previous move. This means that the probability at each step is

$$p_{t+1} = Mp_t. \tag{10}$$

The evolution of each step and the number of possible ending positions of *t* iterations can be described as a decision tree, shown visually in Figure 3. Each level of the tree is an iteration of the walk and each node the position on the number-line with each branch stemming out of a node being a possible next step from that node. Since we know all possible outcomes that could happen from the starting node after T iterations of the walk, the walk along the number-line can also be thought of as a traversal through its decision tree. Now, it appears that this walk is still operating discretely, so in order to turn this into a continuous walk we assume that a transition could occur at anytime in the walk at a rate of γ, which is a fixed constant rate. The iterations in the walk are still measured in discrete time units, but the jump to the next position on the number line occurs with a probability γ with each passing unit of time. Thus, the entire walk is conducted in the H_p state space and a new continuous transition matrix H can be written:

$$H_{ij} = \begin{cases} -\gamma, & j = i \pm 1 \\ 0, & j \neq i \pm 1 \text{ and } i \neq j \\ \frac{1}{2}\gamma, & i = j \end{cases} \tag{11}$$

As only one position can be moved in each time increment, there is a zero probability of a transition if i and j are further than one space apart, but a non-zero probability that a transition will not occur. The

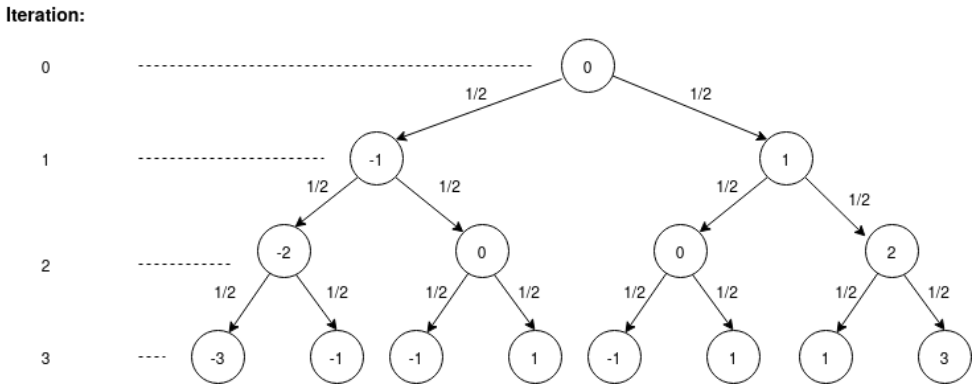


Figure 3: The decision tree to T=4 iterations of walking on a number-line

new continuous probability of being at a a position *i* at some time *t* is described by:

$$\frac{dp_i(t)}{dt} = -\sum_j H_{ij} p_j(t) \tag{12}$$

Solving with an assumption of an initial position of 0, the probability at time *t* becomes:

$$p(t) = e^{-Ht} p(0) \tag{13}$$

To move the continuous walk into the quantum space, a quantum operator can be constructed using the continuous transition matrix H:

$$U_{walk}(t) = e^{-iHt} \tag{14}$$

From [7], at anytime t in the walk the state of the walk will be a superposition of the basis states, i.e. the possible positions on our number-line or the nodes reachable from our place on the decision tree, which all exists in our Hilbert space H_p, as defined in the discrete walk section. The results of the continuous walk show the same quantum advantage as the discrete quantum walk, even though the mechanics of how the walks operate are not the same. Currently there is little known about the relationship between the discrete and continuous quantum walks and is an open research area.

2.3 Grover's Search

Grover's algorithm [8] is the most famous quantum search algorithm and has similar components to the quantum random walk search. As shown in Figure 4, the algorithm can be split into two distinct sections which we will refer to as the oracle circuit and the amplification circuit.

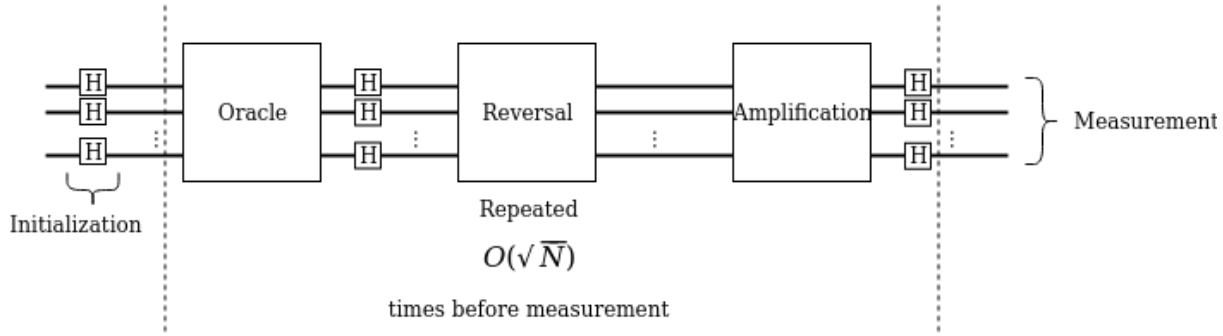
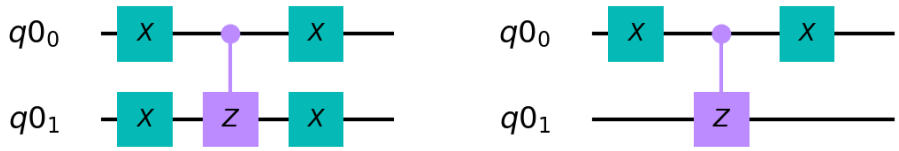


Figure 4: A high-level representation of the key components of Grover's algorithm. The algorithm is run $\sqrt{(N)}$ times on the input, where N is the number of qubits of the input.

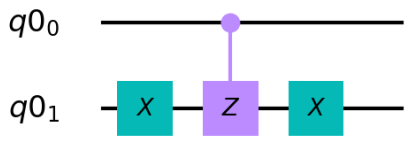
2.3.1 The Oracle Circuit

The oracle portion of the circuit is dependent on the application of the algorithm, so it is left as a "black-box" in its general form. The operation of the oracle is to take the search space (i.e. the superposition of all possible states) as an input and mark the item that is being searched for (i.e. one of the states in the input with a negative sign). Figure 5 shows some examples of oracle circuits for an N=2 circuit.

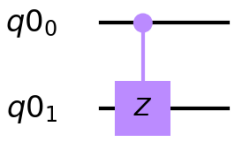


(a) Oracle marks state of $|00\rangle$

(b) Oracle marks state of $|01\rangle$



(c) Oracle marks state of $|10\rangle$



(d) Oracle marks state of $|11\rangle$

Figure 5: Examples of possible two-qubit oracles [10] that flip the sign if input is (a) 00 (b) 01 (c) 10 and (d) 11. Qubits are labelled as follows: q(register number)$_{\text{(qubit position in register)}}$.

The operation of the oracle [9] can be generally described as:

$$|x\rangle\,|q\rangle \rightarrow |x\rangle\,|q \oplus f(x)\rangle \tag{15}$$

Where x is the element being evaluated and q is a single qubit that will flip if f(x) = 1, where f(x) is a function that determines whether x is what we're searching for. q is set to be in superposition so that the oracle has the following behavior:

$$|x\rangle\,(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) \rightarrow \begin{cases} |x\rangle\,(\frac{|1\rangle - |0\rangle}{\sqrt{2}}), & \text{if } f(x) = 1. \\ |x\rangle\,(\frac{|0\rangle - |1\rangle}{\sqrt{2}}), & \text{if } f(x) = 0. \end{cases} \tag{16}$$

This can be rewritten as a sign-flip controlled by the result of f(x) as so:

$$|x\rangle\,(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) \rightarrow |x\rangle\,(-1)^{f(x)}(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) \tag{17}$$

This shows that solutions to the oracle will be marked negatively, which can cleverly be used later in the amplification circuit. Since the value of q does not change other than in sign, the behavior of the oracle can be most simply written as:

$$|x\rangle \rightarrow |x\rangle\,(-1)^{f(x)} \tag{18}$$

2.3.2 Example of the Oracle

An example of applying the oracle from Figure 5 (c) to the input $|x\rangle$ is shown below. First lets define the gates used in the oracle:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} I \otimes X = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{19}$$

The X-gate is only applied to the second qubit, so to get a two-qubit definition we take the tensor product of the Identity with the X-gate, as the Identity will change the values of the first qubit. The CZ, or *controlled-Z*, gate is the definition of the two-qubit gate labeled "Z" in Figure 5 (c). We can now construct:

$$\begin{aligned} \hat{f}(x) &= I \otimes X \cdot CZ \cdot I \otimes X \\ &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{aligned} \tag{20}$$

The input $|x\rangle$ will start in an equal superposition:

$$|x\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} \tag{21}$$

Putting this all together, we can now apply to oracle to the input:

$$\begin{aligned}\hat{f}(x)\,|x\rangle &= \begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1\\1\\-1\\1\end{bmatrix} \\ &= \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle + |11\rangle)\end{aligned} \tag{22}$$

After applying the oracle, the state $|10\rangle$ now has a negative amplitude.

2.3.3 The Amplification Circuit

Taking off from where the oracle has left our qubits, we will have the solution "marked" with a negative amplitude. Simply having a negative amplitude will not help when it comes to taking a measurement, as while the reflection does lower the average amplitude of x, it is not significant enough to detect the solution in a measurement. An additional reflection, however, can be performed to boost the amplitude of our negatively-marked solution(s) [9], moving the state closer to that of the solution state.

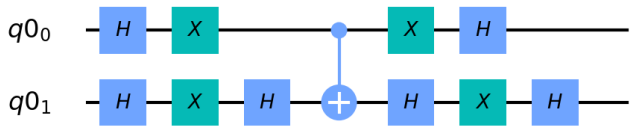


Figure 6: The "amplification" or "rotation" circuit for a 2-qubit system.

A visual representation of the reflections performed on $|\psi\rangle$, the equal superposition of all states, by the oracle and amplification circuits on a plane defined by the superposition of all solution states and the superposition of all non-solution states is shown in Figure 7. We start the amplification circuit by again applying a Hadamard gate to give the equal superposition of states and then applying the conditional

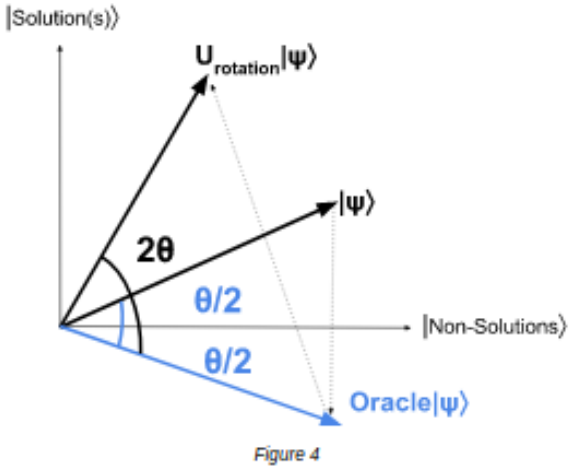


Figure 7: A visual representation of the reflections performed on $|\psi\rangle$, the superposition of all states, by the oracle and amplification circuits on a plane defined by the superposition of all solution states and the superposition of all non-solution states. A θ shift is applied with each iteration of Grover's algorithm.

phase shift:

$$|x\rangle \rightarrow -(-1)^{\delta_x 0} |x\rangle \tag{23}$$

Where $\delta_x 0$ represents a conditional component that applies a negative sign to all states except $|0\rangle$. Lastly, apply another Hadamard gate to x.

A more technical definition of this rotation can be summed up by defining the following operator as the identity subtracted from twice the projector onto $|\psi\rangle$, the equal superposition of all states.

$$U_{rotation} = 2 |\psi\rangle \langle\psi| - I \tag{24}$$

2.3.4 Example of the Amplification Operation

Going back to the state of the input register $|x\rangle$ from the oracle in Equation (22), we will now apply the amplification operation to $|x\rangle$. Since this example is a two-qubit system, the time-complexity is $O(\sqrt{2})$. Only one iteration is needed to find the target state. As the Hadamard and X gates are applied to both qubits, the matrices are:

$$H \otimes H = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} X \otimes X = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{25}$$

The two-qubit gate in Figure 6 is a cNot, or *controlled-not*, gate and will flip the value of the second qubit if the value of the first qubit is in the excited state. However, the second qubit also has a Hadamard applied before and after the cNot gate, which combined becomes:

$$I \otimes H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} cNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{26}$$

$$(I \otimes H) \cdot CNOT \cdot (I \otimes H) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{27}$$

Which is the the CZ gate from Equation (19). Putting all the pieces together to construct the amplification operator:

$$\begin{aligned} U_{rotation} &= (H \otimes H) \cdot (X \otimes X) \cdot cNot \cdot (X \otimes X) \cdot (H \otimes H) \\ &= \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \end{aligned} \tag{28}$$

Applying the operator to the input:

$$\begin{aligned} &U_{rotation} \left|x\right\rangle \\ &= \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \\ &= \frac{1}{8} \begin{bmatrix} 0 \\ 0 \\ 8 \\ 0 \end{bmatrix} \\ &= \left|10\right\rangle \end{aligned}$$

Produces the expected answer.

3 Implementation

3.1 Neutral-Atom Quantum Computing

In neutral atom quantum computing the qubits are represented by contained and addressable atoms. This is accomplished by first cooling and trapping neutral atoms with lasers and magnetic fields inside a magneto-optical trap (MOT). Once they have been trapped in the MOT, the cold atoms are transferred to a light pattern where they can be held as an array of individually addressable atoms. The result of a narrow-band laser excitation of atoms can be found starting from the time-dependent Schrödinger equation representing an atom in a radiation field and then making approximations to constrain ourselves to a Rabi two-level problem [11], a solution of just two states. We refer to these two states as the ground state (i.e. $|0\rangle$), denoted as g below, and excited state (i.e. $|1\rangle$), denoted as e below, the equations for the probability amplitudes of these two states are found to be:

$$\begin{cases} c_g(t) = (cos(\frac{\Omega' t}{2}) - i\frac{\delta}{\Omega'} sin(\frac{\Omega' t}{2}))e^{+i\delta t/2} \\ \\ c_e(t) = -i\frac{\Omega}{\Omega'} sin(\frac{\Omega' t}{2})))e^{-i\delta t/2} \end{cases} \tag{29}$$

Where $|c_g(t)|^2$ and $|c_e(t)|^2$ give the respective probability of finding the atom in that state, with this probability oscillating at an angular frequency of $\Omega' \equiv \sqrt{(}\Omega^2 + \delta^2)$. The angular frequency at which the probability oscillates is dependent on $\Omega \equiv \frac{-eE_o}{\hbar} \langle e| r |g\rangle$, which is the Rabi frequency (i.e. the angular frequency of the cycle of an atom between states e and g) and $\delta \equiv \omega_l - \omega_{eg}$, the laser detuning from ρ_{eg}, the atomic resonance frequency. By manipulating the laser intensity, detuning, and pulse duration, we can take a single-qubit state from anywhere to anywhere on the Bloch sphere, implementing all possible single-qubit gates.

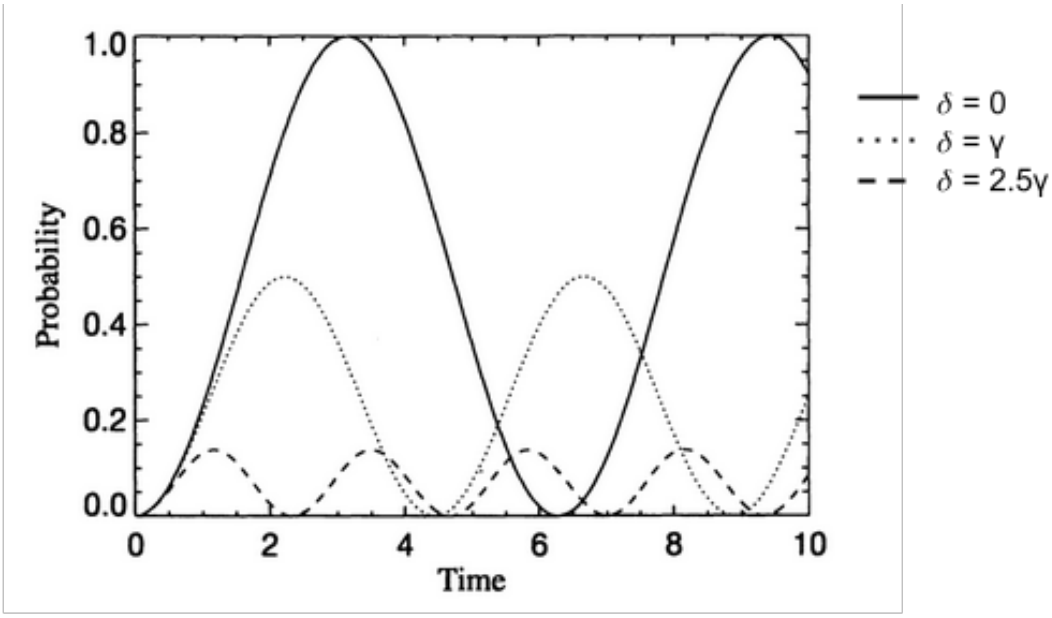
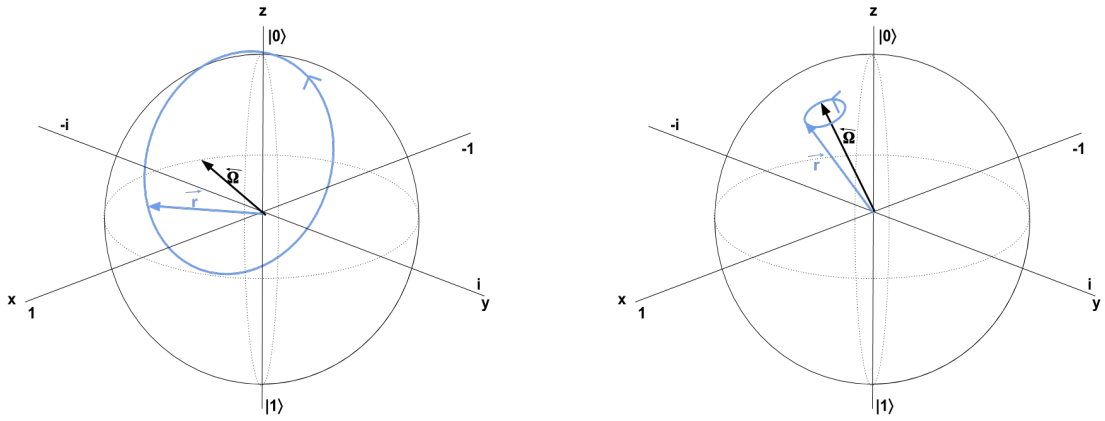


Figure 8: Illustration from [11] of the probability of an atom to be in the excited state for $\Omega = \gamma$ in time units of $\frac{1}{\gamma}$ at different laser detunings.

3.1.1 Building Quantum Gates

The two single-qubit gates used in Grover's algorithm and described in Section 2.3.2 are depicted on the Bloch sphere in Figure 9, the Bloch vector r on the Bloch sphere [11], visualized as a rotating frame depicting the real and imaginary parts of the state. In Figure 9 (a) There is a near zero laser detuning component, so the vector is making large precesssions, giving a close to even probability in being found in either state, exactly what is needed for a Hadamard gate. In Figure 9 (b) there is a large laser detuning component δ. The Bloch vector precesses in a very close circle near one of the axis, giving a very high probability in one state. To create an X-gate, applying a very large detuning in the opposite sense for a half-cycle in duration will flip the direction and precession of r to the other state axis.



(a) Near zero detuning for Hadamard gate

(b) Large detuning for an X-gate

Figure 9: Single qubit gates on the Bloch sphere

For a two-qubit gate, there needs to be an interaction between the control and target qubits. In [12] a neutral-atom approach to a cNOT gate is built using the Rydberg blockade to block the excitation of the target atom if the control is in the excited state. Figure 10 depicts the order and phase *amount* of the Rabi oscillation of applied laser pulses for the cNOT gate. A Hadamard rotation (a $\frac{\pi}{2}$ phase to the wave function) is applied to the target before and after the cPHASE operation(pulses 2, 3, 4). If the control is $|0\rangle$, the target is uninhibited from transitioning to the basis or excited state. If the control is $|1\rangle$, the control is excited into the Rydberg level which prevents pulse 3 from having any effect on the target atom because of the Rydberg blockade.

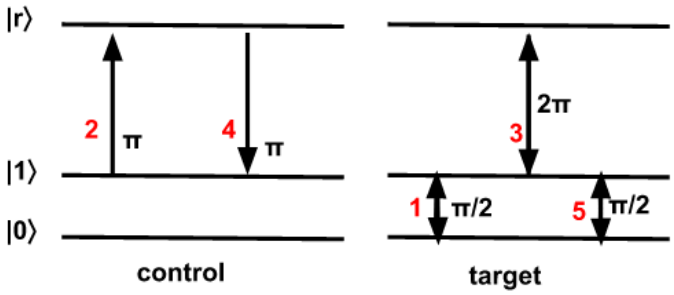


Figure 10: Laser pulses composing a cNOT gate

These gates are the all the components needed to implement Grover's search algorithm. In summary, the series of gates depicted in Section 2.3 that make-up Grover's are an arrangement of a series of laser pulses applied to lattice sites.

3.1.2 Discrete Walk with Neutral Atoms

The discrete quantum walk of section (2.2) using a Hadamard coin has already been physically implemented [13]. In this section we will cover the physical set-up for the n-step walk using the definitions of S and H from equations (5) and (6):

$$|\psi_n\rangle = (SH)^n |\psi_0\rangle \tag{30}$$

The system will be made up of two, one-dimensional lattices, each for trapping a different of the basis states of a neutral atom. The lattices are identical and made up of optical potentials of period *d* that form through counter-propagating harmonic waves with electric fields forming an angle 2θ. The lattices are manipulated by changing the angle θ, which causes the left and right circular polarized components of the standing wave that form the total electric field to shift [13]. Lattice0, which holds atoms in the basis state $|\uparrow\rangle$ and will move with constant velocity $v_0 = -v$ to the left while lattice1 holds atoms in the basis state $|\downarrow\rangle$ and will move with constant velocity $v_0 = v$ to the right.

The walk will start by placing a single neutral atom prepared in state $\frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$ in the lattice0 at the minimum of a potential well which we will refer to as x_0 at time t_0. The Hadamard gate, as described in Section 3.1.1, will then be applied periodically at $t = \frac{nd}{v}$ to all trap locations in the lattice. The atom will move with constant velocity with the lattice it is trapped in determining the direction of travel:

$$x(t) = \begin{cases} x_0 - vt, \text{(i.e. left) if } |\uparrow\rangle \\ x_0 + vt, \text{(i.e. right) if } |\downarrow\rangle \end{cases} \tag{31}$$

At time t_n the the atom has been shifted by n periods and the lattice0 and lattice1 will again be on top of each other, allowing for the atom to switch lattices if the state of the atom was changed by the Hadamard "coin flip" on the state of the atom. After the number of iterations desired have been completed, a measurement can be performed. The position of the particle can be determined by using a florescence measurement, which is done by applying a laser that atoms in the $|\uparrow\rangle$ react to by giving off light, to the system revealing which basis state the atom is in and its location in the lattice based on the light emission

resulting from the aforementioned applied pulse. Running this procedure multiple times will result in the ending location of the particle to match the distribution shown in Figure 2, the expected distribution for a symmetric discrete quantum one-dimensional random walk.

3.1.3 Continuous Walk with Neutral Atoms

An implementation of a continuous quantum random walk in one dimension has been proposed using Rydberg atoms [14] but yet to be done experimentally. Using the blockade mechanisms and strong interactions of Rydberg atoms, [14] fills the sites of an optical lattice with neutral atoms and then promotes one atom in each trap location to higher energy levels to perform the continuous walk.

The energy states of atoms greatly effect the interaction energy between two atoms. Atoms excited to an *np* energy level will have very strong interactions with other atoms in that state. Atoms in an *ns* energy level will have weak interactions with atoms *np* state due to the large difference in energy as well as with other *ns* energy atoms. As a result, Rydberg atoms, which have a very large radii and strong dipole moments, will lead to a blockade mechanism taking place. The blockade means that only one atom can be excited into a Rydberg state due to large shifts in energy levels and other atoms interacting with a Rydberg atom will be prevented from being promoted to a high-energy state. Using this property, [14] studies the diffusion of an excited state among many sites in a lower energy state as a way to perform a continuous walk along an optical lattice.

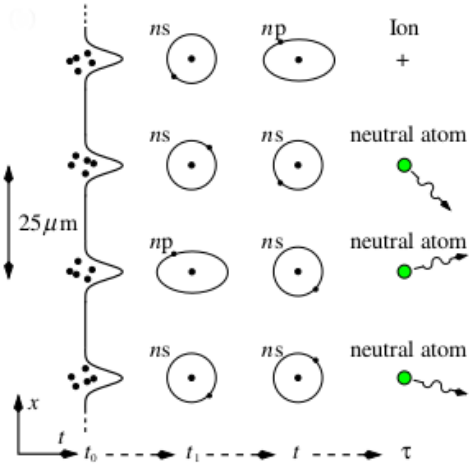


Figure 11: Illustration from [14] of the continuous quantum random walk implementation process

As seen in Figure 11, instead of trapping a single atom as done for the discrete walk in [13], a cluster of atoms are trapped at trap locations about 25μm apart, in order to facilitate easy addressing and detection at each site as well as site separation having an effect on the timescale of how quickly iterations occur [14]. For N-1 of the sites, one atom in each cluster will be promoted to a Rydberg state *ns*. The last remaining site will be promoted to a Rydberg state *np* and will be considered the x_0 location of the walk. The *np* excitation will rapidly transfer from trap site to trap site by a phenomenon called resonant transfer. One step (i.e. one hop along our line) will occur within the time increment that is dependant on the dipole interaction energy between the atoms, $\tau_{hop} \sim \frac{h}{4V_{dipole}}$ with $V_{dipole} \sim \frac{n^4}{R^3}$, n being the principle quantum number and R being the distance of the inter-atomic separations. Using the set-up proposed, the time between hops will be, $\tau_{hop} \sim \frac{h}{4\frac{n^4}{R^3}} = \frac{h}{4\frac{(70)^4}{(25)^3}}$ giving us $\tau_{hop} \sim 170ns$. Waiting for a period of time equal to n iterations passing, the position of the *np* excited state is measured, completing the walk. Performing many repetitions of this the walk procedure will produce a ending-position distribution of the walk matching Figure 2.

4 The Quantum Random Walk Search

Algorithms based on quantum random walks are still in infancy, but a search algorithm has been proposed [15] using a discrete quantum random walk along a hypercube. The algorithmic steps and unitary operator of the quantum random walk search appear quite similar to Grover's search algorithms. The unique properties of the random walk may provide an ease in implementation over other quantum search algorithms, but unfortunately adds complexity to its analysis.

The walk is done on an n-dimensional hypercube, which has $N = 2^n$ nodes, this makes up the H_p part of our Hilbert space. Each node is represented by an n-bit binary number and connected to other nodes based on its *Hamming weight*. The Hamming weight of a binary string is the sum of the number of ones in the string. For example, the string 000 will have a Hamming weight of zero, 001 and 010 both have a Hamming weight of 1, 110 has a Hamming weight of 2, and so forth. A node on the hypercube is connected to all nodes who are a Hamming distance of 1 away from it (i.e. all nodes that are Hamming weight ± 1 of the node's Hamming weight), the connections for an $n = 3$ "hyper"cube is shown in Figure 12. The coin space H_c portion of our Hilbert space is n-dimensional and specifies the direction of the

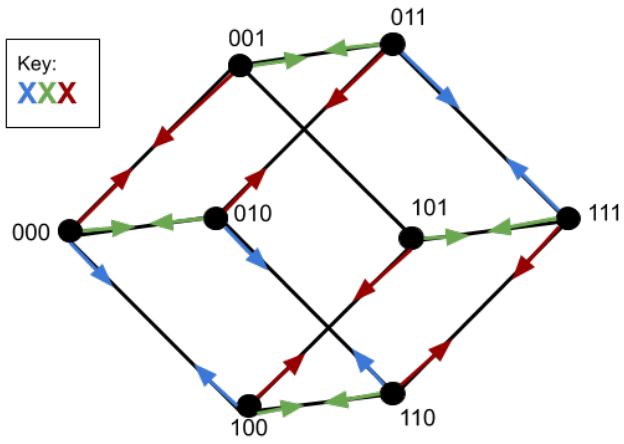


Figure 12: The position space for a walk along a hypercube when n = 3. The graph vertexes are color-coded by direction, i.e. which bit in the binary string is flipped in the transition between nodes

next step, making the total Hilbert space of the search:

$$H = H^n \otimes H^N \tag{32}$$

Like Grover's search, the quantum walk search operator can be broken up into two main components, the shift-operator and the coin-flip. The shift operator creates a superposition of all possible position state transitions and can be described as the following mapping:

$$S = \sum_{d=0}^{n-1} \sum_{x=0}^{N-1} |d, x \otimes e_d\rangle \langle d, x| \tag{33}$$

Where x is the node label in binary, d is the direction , and e_d is the d^{th} coin-space basis vector. Writing out the operator state for an $n = 3$ hyper cube would look like the following:

$$
\begin{aligned}
S &= \sum_{d=0}^{2}\sum_{x=0}^{7} |d, x \otimes e_d\rangle \langle d, x| \\
&= |0, 000 \oplus e_0\rangle \langle 0, 000| + |0, 001 \oplus e_0\rangle \langle 0, 001| + ... + |0, 111 \oplus e_0\rangle \langle 0, 111| \\
&+ |1, 000 \oplus e_1\rangle \langle 0, 000| + |1, 001 \oplus e_1\rangle \langle 1, 001| + ... + |1, 111 \oplus e_1\rangle \langle 1, 111| \\
&+ |2, 000 \oplus e_2\rangle \langle 2, 000| + |2, 001 \oplus e_2\rangle \langle 2, 001| + ... + |2, 111 \oplus e_2\rangle \langle 2, 111| \\
\\
&= |0, 000 \oplus 001\rangle \langle 0, 000| + |0, 001 \oplus 001\rangle \langle 0, 001| + ... + |0, 111 \oplus 001\rangle \langle 0, 111| \\
&+ |1, 000 \oplus 010\rangle \langle 0, 000| + |1, 001 \oplus 010\rangle \langle 1, 001| + ... + |1, 111 \oplus 010\rangle \langle 1, 111| \\
&+ |2, 000 \oplus 100\rangle \langle 2, 000| + |2, 001 \oplus 100\rangle \langle 2, 001| + ... + |2, 111 \oplus 100\rangle \langle 2, 111| \\
\\
&= |0, 001\rangle \langle 0, 000| + |0, 000\rangle \langle 0, 001| + ... + |0, 110\rangle \langle 0, 111| \\
&+ |1, 010\rangle \langle 0, 000| + |1, 011\rangle \langle 1, 001| + ... + |1, 101\rangle \langle 1, 111| \\
&+ |2, 100\rangle \langle 2, 000| + |2, 101\rangle \langle 2, 001| + ... + |2, 011\rangle \langle 2, 111|
\end{aligned}
$$

Now, a coin operator must be chosen to act on H_c. A common coin to use is the rotation operator from Equation (20) that is commonly referred to as "Grover's diffusion operator". We will refer to the operator from (20) as the following to be clear on the space that the operator is applied to:

$$G = U_{rotation} = 2 |\psi_c\rangle \langle \psi_c| - I \tag{34}$$

Where $|\psi_c\rangle$ is the superposition over all n directions of walking on the hypercube. The Grover coin is a useful coin as it is the operator the farthest away from the identity operator, which makes it efficient in *mixing*, which means to reach a stationary distribution, over states regardless of the initial position of the walk.

In order to construct a search algorithm, the operation in the coin will be used like the Oracle in Grover's, to "mark" the node that we are searching for. However, only using the Grover diffusion operator as the coin/oracle in the quantum walk such that the quantum walk operator is defined as:

$$U = S \cdot G \tag{35}$$

However, applying this operator will not change the state of the system if starting from a superposition of the entire Hilbert space. Instead of just applying a negative phase to the node that is being searched for, the oracle/coin operator for the quantum random walk search will apply one coin, C, which [15] chooses to be $C = -I$ for simplicity in analysis, to the the target node and the Grover diffusion operator, G to all of the other nodes, creating the coin operator:

$$C' = G \otimes I + (C - G) \otimes |x_{target}\rangle \langle x_{target}| \tag{36}$$

Using this hybrid coin, the unitary evolution operator U' is:

$$\begin{aligned} U' &= S \cdot C' \\ &= S \cdot (G \otimes I + (C - G) \otimes |x_{target}\rangle \langle x_{target}|) \\ &= U - 2S \cdot (|\psi_c\rangle \langle \psi_c| \otimes |x_{target}\rangle \langle x_{target}|) \end{aligned} \tag{37}$$

Analysis in [15] shows that the perturbation of this operator results in the quantum random walk search algorithm. Now that all parts of the perturbed unitary operator used in this search algorithm have been defined, the steps for its implementation are the following:

1. Apply a Hadamard gate to the all qubits to initialize the entire Hilbert space to be in an equal super position, $|\psi_0\rangle$
2. Apply the quantum walk search unitary operator, U', to $|\psi_0\rangle$
3. Repeat step 2 $O(\sqrt{N})$ times
4. Measure the state of the system

4.1 Similarities to Grover

As mentioned before, there are many similarities between Grover's search algorithm and the quantum random walk search. It is highly recommended to read and thoroughly understand the implementation mechanics of Grover's before moving on to the quantum random walk search. Like Grover's, the algorithm begins in a superposition over all states, have a run-time of $O\sqrt{N}$, consist of a "marking" step, make use of the Grover diffusion operator, and are represented as a rotation in a two-dimensional subspace. The are some key differences however, brought up in [15] that will be discussed in the next few sections:

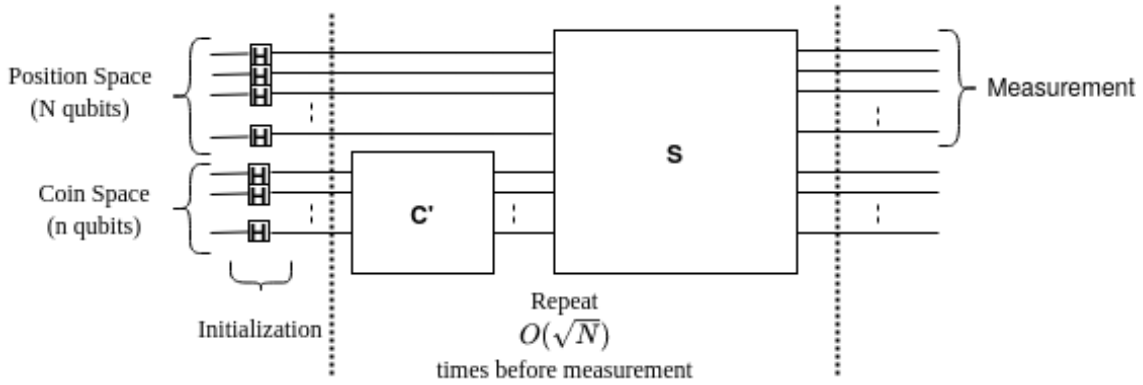


Figure 13: The quantum random walk high-level circuit

1. The random walk search is not an exact mapping onto a two-dimensional subspace

2. The subspace contained by the walk is spanned by (1) a superposition of all states, and (2) a close approximation to the target state, unlike Grover's which is spanned by (1) all non-target states and (2) the target state(s)

3. The final state contains small contributions from its neighbor nodes, unlike Grover's in which the final state is ends up as purely the target state

These differences are yet to be found as advantages or disadvantages, and will be open questions until more work is done experimenting with these algorithms on different quantum hardware.

4.2 The Eigenvectors and Sub-Space

The walk along the hypercube can be reduced down to and described as a weighted walk along a line, as shown in Figure 14. Instead of keeping track of an n-dimensional direction space, the coin flip will correspond to a left (L) or right (R) movement along the line and this dimensionally reduced version of the graph can how be spanned in a two dimensional vector space. Through a series of proofs [15] that we will not get into here, it has been shown that there will be exactly two eigenvalues of unit norm that have a real part greater than $1 - \frac{2}{3n}$. These are the only two eigenvalues of great relevance, and they will make up the two dimensional subspace that is spanned in the walk.

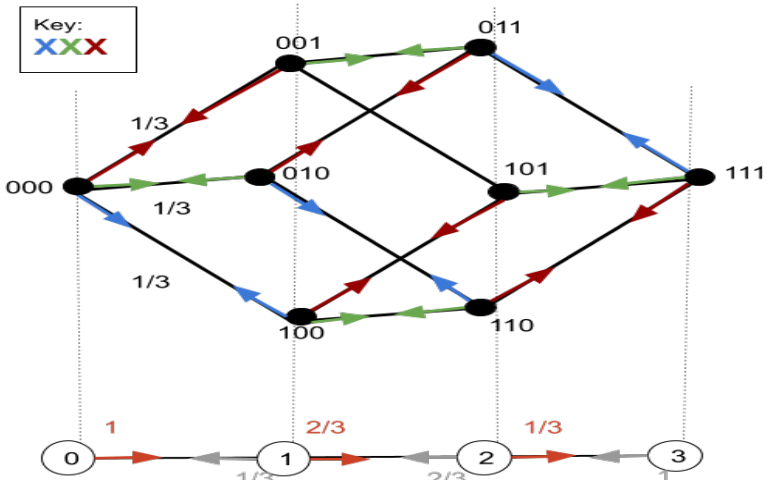


Figure 14: The n=3 walk reduced to an unbalanced walk on a line. The nodes of the walk are now the the possible Hamming weights of an n=3 graph rather than n-bit binary strings.

4.2.1 The Approximate Eigenvectors of U'

We begin with defining two approximate eigenvectors of U', $|\psi_0\rangle$, the superposition over all states and transitions, and $|\psi_1\rangle$, an approximation of the target state. For the examples that follow, we will assume the target state $x_{target} = 0$. The definition of $|\psi_0\rangle$ is as follows[1]:

$$|\psi_0\rangle = \frac{1}{\sqrt{N}}|R,0\rangle + \frac{1}{\sqrt{N}}|L,n\rangle + \sum_{x=1}^{n-1}(\sqrt{\frac{\binom{n-1}{x-1}}{N}}|L,x\rangle + \sqrt{\frac{\binom{n-1}{x}}{N}}|R,x\rangle) \tag{38}$$

Notice that now the superposition over the states is not an equal superposition anymore, with lower probabilities on the farthest left and farthest right node. Expanding ψ_0 for the $n = 3$ walk becomes:

$$\begin{aligned}|\psi_0\rangle &= \frac{1}{\sqrt{8}}|R,0\rangle + \frac{1}{\sqrt{8}}|L,3\rangle + \sum_{x=1}^{2}(\sqrt{\frac{\binom{2}{x-1}}{8}}|L,x\rangle + \sqrt{\frac{\binom{2}{x}}{8}}|R,x\rangle) \\ &= \frac{1}{\sqrt{8}}|R,0\rangle + \frac{1}{\sqrt{8}}|L,3\rangle + \frac{1}{\sqrt{8}}|L,1\rangle + \frac{1}{2}|R,1\rangle + \frac{1}{2}|L,2\rangle + \frac{1}{\sqrt{8}}|R,2\rangle\end{aligned}$$

Note that now x represents the Hamming weight of a position along the line rather than the individual bit sequence of each node due to the collapse of the reduction in dimensions.

[1]In case it is unfamiliar, the mathematical symbol $\binom{n}{x}$ is the binomial function and is defined as $\binom{n}{x} = \frac{n!}{x!(n-x)!}$

Now to find the second approximate eigenvector let ψ_1 be defined as[2]:

$$|\psi_1\rangle = \frac{1}{c}\sum_{x=0}^{\lfloor \frac{n}{2}-1 \rfloor}\left(\sqrt{\frac{1}{\binom{n-1}{x}}}\,|R,x\rangle - \sqrt{\frac{1}{\binom{n-1}{x}}}\,|L,x+1\rangle\right) \tag{39}$$

Where $c = \sqrt{\sum_{x=0}^{\lfloor \frac{n}{2}-1 \rfloor}\left(\frac{1}{\binom{n-1}{x}}\right.}$, is a normalization constant. Expanding ψ_1 for the $n = 3$ walk becomes:

$$\begin{aligned}|\psi_1\rangle &= \sum_{x=0}^{0}\left(\sqrt{\frac{1}{\binom{2}{x}}}\,|R,x\rangle - \sqrt{\frac{1}{\binom{2}{x}}}\,|L,x+1\rangle\right), c = 1\\ &= \frac{1}{\sqrt{2}}\,|R,0\rangle - \frac{1}{\sqrt{2}}\,|L,1\rangle\end{aligned}$$

This vector includes all states in the target position as well as states with a non-zero probability of transitioning to the target state.

4.2.2 The Sub-Space of U'

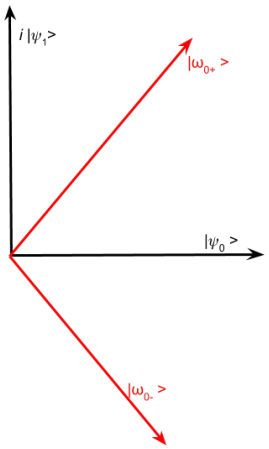


Figure 15: The $|\omega_{0+}\rangle, \omega_{0-}$ plane and initial position of $|\psi_0\rangle$ and complex$|\psi_1\rangle$. All vectors but $|\psi_0\rangle$ remain stationary while $|\psi_0\rangle$ will rotate towards $|\psi_1\rangle$ with each application of U'.

As mentioned previously, the spanned space of the walk is contained within a complex conjugate pair of eigenvectors of U' we will call $|\omega_{0+}\rangle$ and $|\omega_{0-}\rangle$. A spectral analysis on U' (see Appendix A), confirms

[2]In case it is unfamiliar, the mathematical symbol $\lfloor n \rfloor$ is the floor operator and truncates all non-integer numbers to the ones-place, i.e. $\lfloor 3.2 \rfloor = 3$ and $\lfloor 3.7 \rfloor = 3$ as well

that there are only two eigenvalues with a real component greater than $1 - \frac{2}{3n}$. These eigenvectors can be well approximated by $|\psi_0\rangle$ and $|\psi_1\rangle$ (proof in [15]) as the following linear combinations:

$$\begin{aligned} |\omega_{0+}\rangle &\approx \frac{1}{\sqrt{2}}(|\psi_0\rangle + i\,|\psi_1\rangle) \\ |\omega_{0-}\rangle &\approx \frac{1}{\sqrt{2}}(|\psi_0\rangle - i\,|\psi_1\rangle) \end{aligned} \tag{40}$$

And it is possible to implement an approximate search by rotating the initial state$|\psi_0\rangle$ towards the approximate target state $|\psi_1\rangle$ within the $|\omega_{0+}\rangle$, $|\omega_{0-}\rangle$ plane. The initial state $|\psi_0\rangle$ begins at a position perpendicular to $|\psi_1\rangle$, and after applying the operator U' to $|\psi_0\rangle$ an O($\sqrt{N}$) number of times, $|\psi_0\rangle$ will rotate sufficiently close to $|\psi_1\rangle$ to successfully measure $|x_{target}\rangle$. As with all quantum algorithms, a single run will have error, but it can be made arbitrarily small without increasing complexity by repeating the algorithm a fixed number of times.

5 Conclusions

Random walks are a powerful tool in classical computing that can also be used in quantum computing, showing a quadratic speed-up in time complexity. We have introduced and reviewed Grover's quantum search algorithm to build an understanding of quantum gates and common operations, and an intuition on how a quantum searches are implemented. Using the ideas from Grover, we apply a discrete quantum random walk in the quantum random walk search algorithm that is able to perform in $O(\sqrt{N})$ time. Unlike Grover's, which uses of pure states, the quantum random walk search uses many approximations in its search and has residual information encoded based on the structure of the graph. While theoretically there isn't a difference between the two search approaches performance-wise, these differences may come into favor on different quantum hardware, which will be an question until hardware and software become more commonly studied together in quantum computing.

6 Appendix A

The following calculations were done following the results of [15].

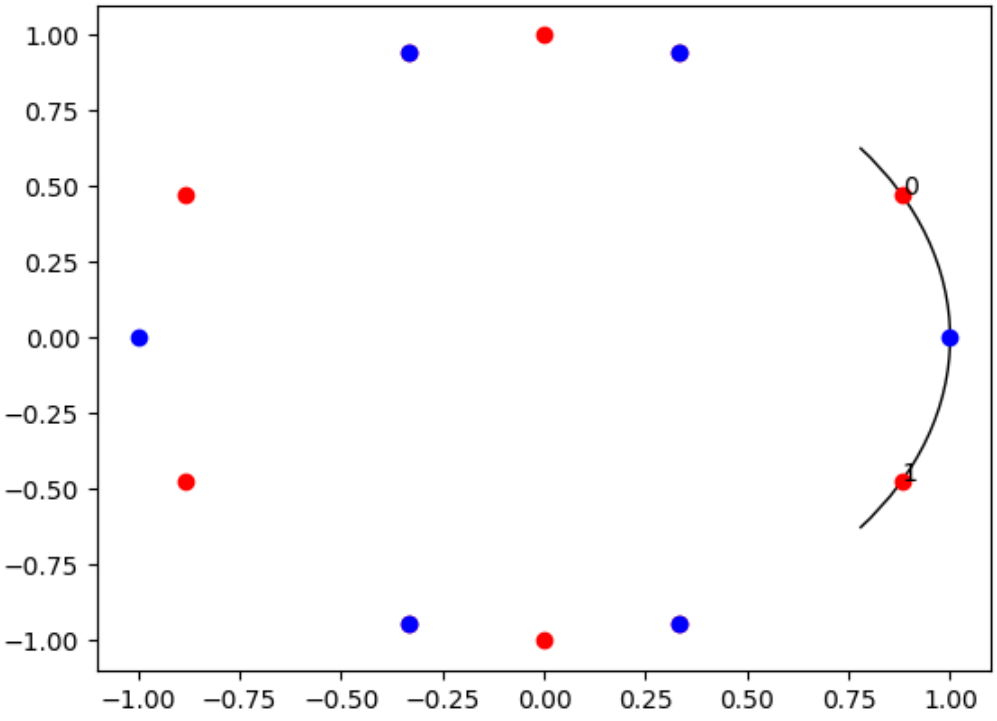


Figure 16: The spectral analysis of an n = 3 search. The red points are the eigenvalues of U' and the blue is are the eigenvalues of U. The arc spans the $1 - \frac{2}{3n}$ area the eigenvalues of $|\omega_{0+}\rangle$ and $|\omega_{0-}\rangle$ could reside

The values of the eigenvalues and corresponding eigenvectors closest to the real axis for the n = 3 walk are:

$$eigenvalue0 = 0.8819171 + 0.47140452i \quad eigenvector1 = 0.8819171 - 0.47140452i$$

$$eigenvector0 = \qquad\qquad eigenvector1 =$$

$$\begin{bmatrix} 2.67261242 \times 10^{-01} & +0.0i \\ -2.35702260 \times 10^{01} & +1.25988158 \times 10^{01}i \\ 1.17851130 \times 10^{01} & +1.25988158 \times 10^{01}i \\ -4.45435403 \times 10^{02} & +1.66666667 \times 10^{01}i \\ 1.17851130 \times 10^{01} & +1.25988158 \times 10^{01}i \\ -4.45435403 \times 10^{02} & +1.66666667 \times 10^{01}i \\ 8.90870806 \times 10^{02} & +1.66666667 \times 10^{01}i \\ 2.15952654 \times 10^{16} & +1.88982237 \times 10^{01}i \\ 2.67261242 \times 10^{01} & -8.34332948 \times 10^{17}i \\ 1.17851130 \times 10^{01} & +1.25988158 \times 10^{01}i \\ -2.35702260 \times 10^{01} & +1.25988158 \times 10^{01}i \\ -4.45435403 \times 10^{02} & +1.66666667 \times 10^{01}i \\ 1.17851130 \times 10^{01} & +1.25988158 \times 10^{01}i \\ 8.90870806 \times 10^{02} & +1.66666667 \times 10^{01}i \\ -4.45435403 \times 10^{02} & +1.66666667 \times 10^{01}i \\ 8.75973249 \times 10^{17} & +1.88982237 \times 10^{-01}i \\ 2.67261242 \times 10^{01} & -6.12696729 \times 10^{-17}i \\ 1.17851130 \times 10^{01} & +1.25988158 \times 10^{-01}i \\ 1.17851130 \times 10^{01} & +1.25988158 \times 10^{-01}i \\ 8.90870806 \times 10^{02} & +1.66666667 \times 10^{-01}i \\ -2.35702260 \times 10^{01} & +1.25988158 \times 10^{-01}i \\ -4.45435403 \times 10^{02} & +1.66666667 \times 10^{-01}i \\ -4.45435403 \times 10^{02} & +1.66666667 \times 10^{-01}i \\ -5.00035475 \times 10^{17} & +1.88982237 \times 10^{-01}i \end{bmatrix} \begin{bmatrix} 2.67261242 \times 10^{01} & -0.0i \\ -2.35702260 \times 10^{01} & -1.25988158 \times 10^{01}i \\ 1.17851130 \times 10^{01} & -1.25988158 \times 10^{01}i \\ -4.45435403 \times 10^{02} & -1.66666667 \times 10^{01}i \\ 1.17851130 \times 10^{01} & -1.25988158 \times 10^{01}i \\ -4.45435403 \times 10^{02} & -1.66666667 \times 10^{01}i \\ 8.90870806 \times 10^{02} & -1.66666667 \times 10^{01}i \\ 2.15952654 \times 10^{16} & -1.88982237 \times 10^{01}i \\ 2.67261242 \times 10^{01} & +8.34332948 \times 10^{17}i \\ 1.17851130 \times 10^{01} & -1.25988158 \times 10^{01}i \\ -2.35702260 \times 10^{01} & -1.25988158 \times 10^{01}i \\ -4.45435403 \times 10^{02} & -1.66666667 \times 10^{01}i \\ 1.17851130 \times 10^{01} & -1.25988158 \times 10^{01}i \\ 8.90870806 \times 10^{02} & -1.66666667 \times 10^{01}i \\ -4.45435403 \times 10^{02} & -1.66666667 \times 10^{01}i \\ 8.75973249 \times 10^{17} & -1.88982237 \times 10^{01}i \\ 2.67261242 \times 10^{01} & +6.12696729 \times 10^{17}i \\ 1.17851130 \times 10^{01} & -1.25988158 \times 10^{01}i \\ 1.17851130 \times 10^{01} & -1.25988158 \times 10^{01}i \\ 8.90870806 \times 10^{02} & -1.66666667 \times 10^{01}i \\ -2.35702260 \times 10^{01} & -1.25988158 \times 10^{01}i \\ -4.45435403 \times 10^{02} & -1.66666667 \times 10^{01}i \\ -4.45435403 \times 10^{02} & -1.66666667 \times 10^{01}i \\ -5.00035475 \times 10^{17} & -1.88982237 \times 10^{0}i \end{bmatrix}$$

The python script used to generate this is provided below and also hosted on `https://github.com/ManyaManya/QRW_Unitary_Analysis`. The number of dimensions of the walk used in the spectral analysis can be changed by changing the value set for n at the top of the script.

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

#set the number of dimensions
n = 3
N = pow(2, n)

np.set_printoptions(linewidth=np.inf)

psi_proj = (1/n)*np.matrix(np.ones((n, n))) #superposition of coin space

Co = 2*psi_proj - np.matrix(np.identity(n))
print("Co:\n", n*Co, "\n")#factor out normalization for ease of reading

C = np.matrix(np.kron(Co, np.identity(N)))
print("C:\n", n*C, "\n")

S = np.matrix(np.zeros((n*N, n*N)))
d = 0
for i in range(0, n):
   for x in range(0, N):
      ed = d + pow(2, i)
      S[x^ed, x+d] = 1
      S[x+d, x^ed] = 1
   d += N

print("S:\n", S, "\n")

U = S*C
print("U:\n", n*U, "\n")

zero_proj = np.matrix(np.zeros((N, N)))
zero_proj[0,0] = 1

#the qrw unitary operator
U_ = U - 2*S * np.kron(psi_proj, zero_proj)
print("U_:\n", n*U_, "\n")

#calc the eigenvalues
val, vect = np.linalg.eig(U_)
val_u, vect_u = np.linalg.eig(U)

#numpy calcs extremely close valued eignvals, round them off
val = np.ndarray.round(val, 8)
val_u = np.ndarray.round(val_u, 8)

#remove duplicate values
val, idx = np.unique(val[val.imag != 0], return_index=True)
```

```
vect = vect[:, idx]
val_u, idx_u = np.unique(val_u, return_index=True)
vect_u = vect_u[:, idx_u]

#print eigvals and vects in arc
idx_s = np.argsort(-val)#rev values to sort in descending order
print("\neigval_0+: ", val[idx_s[0]],"\n omega_0+:\n", vect[:, idx_s[0]])
print("\neigval_0-: ", val[idx_s[1]], "\nomega_0-:\n",vect[:, idx_s[1]])

#plot arc of where eignvalues of omega_0+ and omega_0- lie
x = 1 - (2/(3*n))
y = ((1 - x**2)**0.5)
start = np.degrees(np.arctan(-y/x))
end = np.degrees(np.arctan(y/x))
eigval_range = mpl.patches.Arc((0,0), height=2 , width=2, angle=0,
    theta1=start, theta2=end)
plt.gca().add_patch(eigval_range)

#plot the eign values
plt.plot(val.real, val.imag, 'ro')
plt.plot(val_u.real, val_u.imag, 'bo')
plt.text(val[idx_s[0]].real, val[idx_s[0]].imag, 0)
plt.text(val[idx_s[1]].real, val[idx_s[1]].imag, 1)

plt.show()
```

References

[1] A. Plavnick, *The fundamental theorem of Markov chains*, VIGRE REU at UChicago, (2008).

[2] J. Kempe, *Quantum random walks - an introductory overview*, arXiv:quant-ph/0303081, (2003).

[3] A. Childs, E. Farhi, S. Gutmann, *An example of the difference between quantum and classical random walks*, arXiv:quant-ph/0103020v1, (2002).

[4] C. Cohen-Tannoudji, B. Diu, F. Laloë, *Quantum Mechanics, Volume 1: Basic Concepts, Tools, and Applications*, WILEY-VCH, section 2.1.3, (2019).

[5] E. Bach, M. Goldschen, R. Joynt, J. Watrous, *One-dimensional quantum walks with absorbing boundaries*, arXiv:quant-ph/0207008v3, (2004).

[6] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, J Watrous, *One-Dimensional Quantum Walks*, Conference Proceedings of the Annual ACM Symposium on Theory of Computing, (2001).

[7] E. Farhi, S. Gutmann, *Quantum Computation and Decision Trees*, arXiv:quant-ph/9706062v2, (1998).

[8] L. Grover, *A fast quantum mechanical algorithm for database search*, arXiv:quant-ph/9605043v3, (1996).

[9] M. Nielsen, I. Chuang, *Quantum Information and Quantum Copmutation, 10th ed.*, Cambridge University Press, Section 6,(2010).

[10] A. Asfaw et al., *Learn Quantum Computation Using Qiskit, 1st ed.*, (2020).

[11] H. Metcalf, *Laser Cooling and Trapping, 1st ed. Springer-Verlag New York Inc.*, pp. 1-12, (1999).

[12] L. Isenhower et al., *"Demonstration of a Neutral Atom Controlled-NOT Quantum Gate"*, Physical Review Letters, vol. 104, no. 1, (2010).

[13] W. Dur, R. Raussendorf, V. Kendon, H. Briegel, *Quantum random walks in optical lattices*, arXiv:quant-ph/0207137v1, (2018).

[14] R. Cote, A. Russell, E. Eyler, P. Gould, *Quantum random walk with Rydberg atoms in an optical lattice*, New Journal of Physics, (2006).

[15] N. Shenvi, J. Kempe, K. Whaley, *A Quantum Random Walk Search Algorithm*, arXiv:quant-ph/0210064v1 , (2008).