# A Review of Quantum Random Walks and their Algorithmic Applications

Enis K. Inan[1] and Mohamed Abidalrekab[2]

[1]Portland State University einan@pdx.edu
[2]Portland State University moh29@pdx.edu

June 17, 2017

**Abstract**

In this paper, we introduce the concept of random walks and the motivation behind them from a computer scientist's perspective. Then we dive into quantum random walks, and present the two main theoretical models, the discrete time, and continuous time models, that describe them. We show that there are some theoretically interesting problems where the quantum random walk exhibits different behavior than the classical counterpart. In particular, quantum random walks propagate faster. Finally, we present one practical application of the quantum random walk, quantum walk search problems, and give an overview of three algorithms that fit under this framework: subset finding, group commutativity, and a quantum random walk on a grid. All three algorithms exhibit polynomial time speedups from their classical counterparts.

## 1 Introduction

A random walk (RW) is a process where a probability distribution governs the state transitions of all its objects, resulting in a non-deterministic evolution. To take a simple example, flipping a coin to decide which of two directions to take next (e.g. North and South) is modeled by a RW where each direction has probability $1/2$ of occurring. In computer science, RWs are useful as an algorithmic paradigm. They allow one to explore an exponential state space and stumble upon a solution candidate in a polynomial number of steps with high probability. RW algorithms are monte carlo by nature. Each iteration of the algorithm is independent, producing a correct answer with probability $p$. Thus, multiple iterations of the algorithm will yield a correct result with high probability. Example applications of RWs include calculating the volume of a convex body [8] (1) and the PageRank algorithm used by Google search [3] (2).

Unlike the classical world, the quantum world is inherently random [8]. RWs themselves are random. An interesting, theoretical question, then, is how RWs in the quantum world differ from their classical counterparts (1). From a practical perspective, does the QRW give asymptotic speedups to existing problems (2)? Our paper answers these two questions. Section 2 will present all the necessary background knowledge and terminology that will be used throughout this paper. Sections 3 and 4 presents the discrete time (DTM) and continuous time (CTM) models, and answers Question 1 by showing that there are non-trivial differences between the QRW and CRW. Section 5 will present the quantum random walk on a graph (QWRG) and the general pattern of

the only practical application that's unique to the QRW: quantum walk search (QWS). Sections 6, 8, and 7 will present three algorithms that fit under the QWS framework. Finally, Section 9 will present some open problems regarding QRWs and conclude the paper.

## 2 Preliminaries

Formally, a RW is a process consisting of one or more objects where each object $O$ is in a state $S_O$ at time $t$, with a probability distribution governing its change to state $S'_O$ at time $t'$ [1]. If we let $S$ be the set of possible states for our object and $S_C \subseteq S$ be the candidate "solution" or terminating states, the *hitting time* is defined as the average time taken in a RW to go from a starting state, $s_0 \in S$, to a terminating state $s' \in S_C$ [8]. We can think of it as how many steps we need to walk in a RW before we solve the problem.

A quantum state is represented as the sum of two or more independent quantum states (QSs), i.e. basis vectors. Formally, an $N$-dimensional QS, $|\psi\rangle$, is described as $|\psi\rangle = \sum_i \alpha_i |i\rangle$ where $|i\rangle$ is a basis vector, $\alpha_i \in \mathbb{C}$ and $\sum_i |\alpha_i|^2 = 1$. When $|\psi\rangle$ is measured, it is collapsed into one of the basis vectors $|i\rangle$ with probability $|\alpha_i|^2$. QSs evolve through unitary operations, where an $N \times N$ matrix $U$ is unitary if $UU^\dagger = I$. Specifically, the state $|\phi\rangle$ after applying $U$ on $|\psi\rangle$ is $|\phi\rangle = U|\psi\rangle$.

A special class of random processes are Markov chains (MCs). Here we have our set of possible states $S = \{s_1, s_2, \ldots, s_N\}$ that our object $O$ could be in. At time $t$, $O$'s position is described by the probability distribution $\vec{s}_t = (p_t(s_1), p_t(s_2), \ldots, p_t(s_N))$ where $p_t(s_i)$ is the probability that $O$ is in state $s_i$ at time $t$, and $\sum_i p_t(s_i) = 1$. The transition probabilities are described by the $N \times N$ transition matrix $P$ where entry $P_{ij}$ denotes the probability of going from state $s_i$ to state $s_j$. $O$'s position at time $t+1$ is thus determined by [10]:

$$\vec{s}_{t+1} = \vec{s}_t P \tag{1}$$

where

$$p_{t+1}(s_i) = \sum_j p_t(s_j) P_{ji} \tag{2}$$

An *absorbing* state $s_i$ of a MC is a state that is impossible to leave, i.e. $P_{ii} = 1$ and $P_{ij} = 0$ for $i \neq j$. An *absorbing* MC has at least one absorbing state. A MC is *ergodic* if any state can be reached from any other state in a finite number of steps. Ergodic MCs converge to a unique, stationary distribution $\vec{\pi}$ where [10]

$$\vec{\pi} = \vec{\pi} P$$

This is also called the equilibrium distribution. Finally, an ergodic MC is *reversible* if $\pi_i P_{ij} = \pi_j P_{ji}$ [10].

Intuitively, MC are visualized as a graph $G = (V, E)$ where $V = S$ and $E = \{(s_i, s_j) \mid P_{ij} > 0\}$.

Finally, we finish with terminology that will be useful in the latter parts of the paper. A *d-regular graph* is one where each vertex has out-degree $d$; in other words, there are $d$ edges incident on any given vertex. *Quantizing* some mathematical model $X$ means constructing an intuitively equivalent version of $X$ in the quantum world.

## 3 Discrete Time Models

. There are two models that comprise the DTM: the coin-walker model (CWM) and the quantization of a MC (QMC).

### 3.1 Coin-Walker Model

The CWM consists of a coin and a walker, both represented by a QS. A single step of the QRW in the CWM consists of flipping the coin and then applying a shift operator on the walker. Flipping the coin is captured by the unitary $C$, the coin operator, while the shift operator is governed by the unitary $S$. $C$ depends only on the coin-state; S is conditioned both on the coin- state (result of the flip) and the current position of the walker. At a high-level then, a QRW under the CWM is applying the unitary $U = SC$ to an initial QS $|\psi_0\rangle$for T steps, and then measuring the resulting QS to get the final position of the walker, ignoring the coin-state. So $|\psi^T\rangle = U^T |\psi_0\rangle$ [8]. As a simple example, consider a QRW on a line (QRWL) that is infinite in both the right (+) and left (-) directions. The quantum state is described by the basis $|HT\rangle \otimes |\mathbb{Z}\rangle$ where $|HT\rangle = H, T$ represents the coin state and $\{|\mathbb{Z}\rangle = |i\rangle \, | i \in \mathbb{Z}\}$ represents the walker. The coin unitary is the Hadamard gate

$$C_H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3}$$

while the shift operator is described as follows:

$$S |\psi\rangle = \begin{cases} |H\rangle |i+1\rangle & \text{if } |\psi\rangle = |H\rangle |i\rangle \\ |T\rangle |i-1\rangle & \text{if } |\psi\rangle = |T\rangle |i\rangle \end{cases} \tag{4}$$

so the particle will walk to the right if the coin flip resulted in heads, left otherwise. Let the initial state be $|\psi_0\rangle = |T\rangle |0\rangle$. After the first step of the QRW, the state is:

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|H\rangle |1\rangle - |T\rangle |-1\rangle)$$

If we stop and measure $|\psi_1\rangle$ here, we will get $|1\rangle$ or $|0\rangle$ as the final position of the walker with probability $\frac{1}{2}$ for each – exactly identical to a single step of the CRW on a line (CRWL). In the CRWL, the walker starts at 0 and, at any position $i$, has a $1/2$ chance of going âĂIJleftâĂİ to $i$1 and a $1/2$ chance of going âĂIJrightâĂİ to i + 1. In fact for any discrete-time QRW model, if the process of flipping the coin, shifting, and then measuring the resulting QS is repeated $T$ times using the measured result as input for the next step, then we would end up performing a CRW of that model for $T$ steps [2]. So already, we see that the CRW is a special case of the QRW. If we carry out the QRW for 100 steps, record the final position of the walker, and then repeat the experiment multiple times starting from the same initial state, we would get the following probability distribution [8]: Notice that the values drift towards the left. This contrasts with the normal distribution (centered at 0) resulting from the CRWL [8]. If we changed our initial state to $|H\rangle |0\rangle$ instead, then the distribution is Fig. 1 flipped horizontally on a vertical line centered at 0 (i.e. it would drift to the right) [8]. In both cases, the distribution of the QRWL is asymmetric. The asymmetry arises from $C_H$ treating $|H\rangle$ and $|T\rangle$ differently – it multiplies the phase by -1 for the latter. Because the $|H\rangle$ and $|T\rangle$ components will both have real amplitudes after the application of $C_H$ , they will interfere with each other. The final distribution of the QRWL is thus sensitive to the initial state. We can confirm our latter conclusion by noting that the distribution of Fig. 1 started off with the coin-state at $|T\rangle$. From Eqn. 4, we know that $|T\rangle$ moves the particle to the left. Hence, the initial state induces a left-ward bias on the particle so that the overall distribution will drift towards the left: this is indeed what results. But when we start with $|H\rangle$ as our coin-state, we induce a right-ward bias so that the final distribution drifts to the right. If we change $C_H$ to the following unitary $C_Y$:

$$C_Y = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & -1 \end{pmatrix} \tag{5}$$

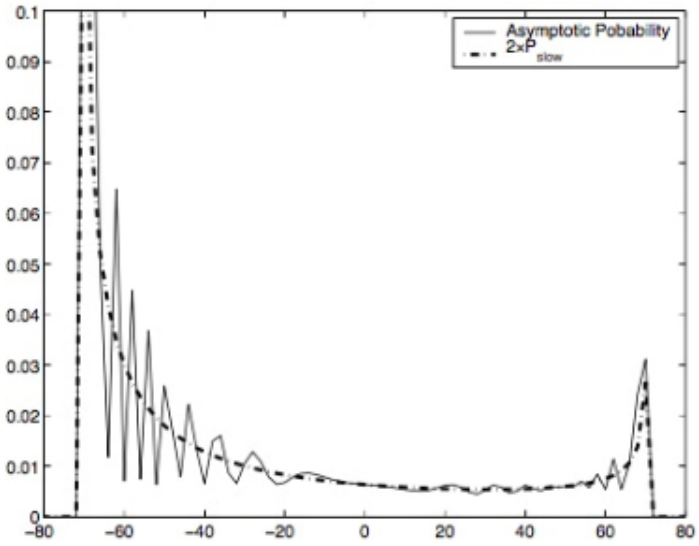


Figure 1: The probability distribution of the QRWL after $T = 100$ steps with the initial condition $|\psi_0\rangle = |T\rangle |0\rangle$ using the coin and shift operators described in Eqns. 3 and 4, respectively. Only even points are plotted – odd points have 0 probability of occurring. Obtained from [8]

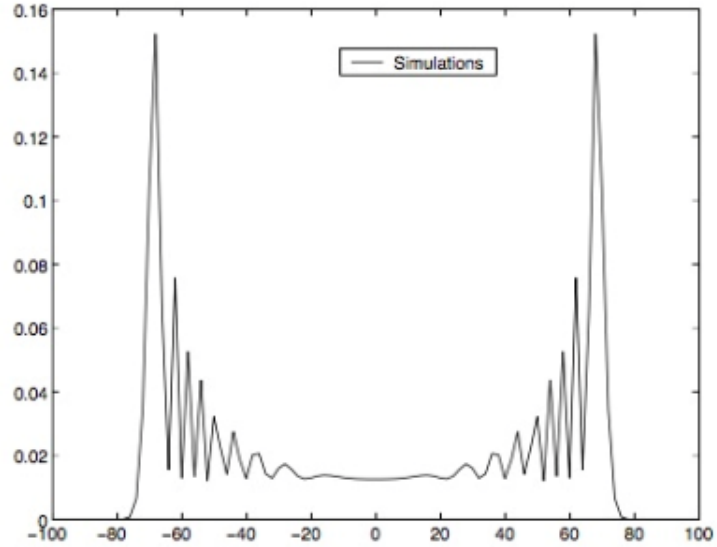


Figure 2: The probability distribution of the QRWL after $T = 100$ steps steps with a symmetric initial condition using the coin and shift operators described in Eqs. 5 and 4, respectively. Only even points are plotted – odd points have 0 probability of occurring. Obtained from [8]

then we get the following distribution for the QRWL using the same experiment [8]: which is symmetric – here the reason is because we have granted $|T\rangle$ the imaginary domain while $|H\rangle$ has the real domain so that the coin-states no longer interfere with each other. However, notice that Fig. 2 still differs from the CRW on a line! Formally, the CRWL after $T$ steps has variance $\sigma^2 = T$ so that the expected distance from the origin is $\sigma = \sqrt{T}$. In the QRWL, we get $\sigma^2 \sim T^2$ so that

$\sigma \sim T$. Thus the QRWL propagates faster than the CRWL and in a uniform manner! The CRWL's distribution, on the other hand, peaks at the origin and then drops off exponentially at several standard deviations away [8].

We conclude this section by defining a QRW on a graph using the CWM (QRWG) as it will be important in the ensuing parts of the paper. In the QRWG, the basis states are $|e\rangle\,|v\rangle$for all $e \in E$ and $v \in V$ where $e$ is incident on $v$ with the coin state being $|e\rangle$ and the position state $|v\rangle$ – this is consistent with what we had for the QRWL. Here, the possible directions to take from a vertex $v$ is exactly all the edges $e$ that are incident to it. A coin-flip corresponds to picking the next outgoing edge from that vertex. For a vertex with degree $m$, it is described by the unitary $C_v = \frac{2}{m}\mathbb{J} - \mathbb{I}$. Intuitively, $C_v$ is structured such that any features provided by the underlying symmetry of the graph are maintained. For example, it lets us reduce a QRWG to a QRWL in a similar manner to the $G_n$ problem that will be outlined in Section 4. The shift operator $S_G$ for a state $|e\rangle\,|v\rangle$ yields $|e\rangle\,|v'\rangle$ and for a state $|e\rangle\,|v'\rangle$ yields $|e\rangle\,|v\rangle$ if $v$ and $v'$ are the endpoints of $e$ [2].

**Note**: One might wonder why the extra coin-state is necessary. The reason is that if we just consider the walker's position, then the valid unitary transformations reduce to trivial cases. For our QRWL, these would be the particle indefinitely moving left, indefinitely moving right, or staying in place [2].

### 3.2 Quantization of a Markov Chain

The material that will be presented in this section is attributed to [11]. One of the limitations of the CWM is that it can only be used to model MCs with uniform transition probabilities and $d$-regular graphs. Here's why. We defined a QWS in the CWM as a tensor product of the coin and walker position states. In the quantum world, systems that are described by tensor products are independent. Hence, the CWM implicitly assumes that every position will have the same set of possible coin-states i.e. directions that it can take in a single step of the walk. For a Markov process, this is not always true – $s_i$ and $s_j$ can have $d_i$ and $d_j$ different directions to go. Using the QRWG presented in Section 3.1 would account for this, but it isn't really an intuitive way to quantize a MC process because it does not use the transition matrix $P$.

It would be desirable, then, if there were a way to quantize discrete MCs in general. We do so using bipartite walks. Let $X$ and $Y$ be two finite sets and $P$ and $Q$ be matrices such that $P_{xy}$ describes the probability of transitioning from an $x \in X$ to a $y \in Y$ and $Q_{yx}$ the probability of transitioning from a $y \in Y$ to an $x \in X$. If we let $G = (V, E)$ where $V = X \cup Y$ and $E = \{(x, y) \mid P_{xy} > 0\} \cup \{(y, x) \mid Q_{yx} > 0\}$, then note that $G$ is a bipartite graph – this is why we use the term "bipartite walk". The walk $(P, Q)$ is a walk over $G$; it is described using basis states $|x\rangle\,|y\rangle$ where $x \in X$ and $y \in Y$. If we define the states

$$\begin{aligned} \phi_x &= \sum_{y \in Y} \sqrt{P_{xy}}\,|x\rangle\,|y\rangle \\ \psi_y &= \sum_{x \in X} \sqrt{Q_{yx}}\,|x\rangle\,|y\rangle \end{aligned} \tag{6}$$

then let $A = \Phi_x$ be the matrix with column vectors $\phi_x$ and $B = \Psi_y$ be the matrix with column vectors $\psi_y$ where $x \in X$ and $y \in Y$. The unitary walk operator $W(P, Q)$ is thus:

$$W(P, Q) = (2\Psi_y\Psi_y^\dagger - I)(2\Phi_x\Phi_x^\dagger - I) \tag{7}$$

We see that a single step of the bipartite walk can, in general, be visualized as zig-zagging from some $x \in X$ to $x' \in X$ through some $y \in Y$, i.e. we go along the edges $(x, y), (y, x')$. Setting

$Q = P$, Eqns. 6 and 7 describe the QMC. Here's why this makes sense. Recall from our note at the end of Section 3.1 that we needed the coin-state in the CWM because the walker's position alone does not result in any interesting unitary transformations. For a MC, the walker's position is the set of states themselves, i.e. the vertices of the underlying graph. Thus, doing a walk on just the vertices of the MC will not yield any interesting behavior. However as we will show later, when we do a walk on the *edges* of the MC, which is exactly what the bipartite walk does, then we do get interesting results.

# 4 Continuous Time Model

In the CTM there is no coin-state. Instead, the QS is a superposition of the basis states where each state is a valid position that the walker can end up at when a measurement is finally made. We can represent the structure traversed by our RW with a graph $G$, where its vertices $V$ are the valid positions of the walker and for $u, v \in V$, $(u, v) \in E$ iff position $v$ is reachable from position $u$ [4]. Thus the CTM is a RW on a graph. The unitary matrix $U$ governing the QS' time-evolution is described by [4]:

$$U = e^{-iHt} \tag{8}$$

where H is the Hamiltonian matrix with the following entries:

$$\langle a| H |b\rangle = \begin{cases} -\gamma & \text{if} \quad a \neq b, (a,b) \in G \\ 0 & \text{if} \quad a \neq b, (a,b) \notin G \\ k\gamma & \text{if} \quad a = b \text{ and } k \text{ is the degree of vortex } a \end{cases} \tag{9}$$

An example application of the CTM is illustrated through a RW on the graph $G_n$. $G_n$ consists of two balanced, n-level binary trees connected together at their leaves. For example, $G_4$ is depicted as in Fig. 3 below [8].

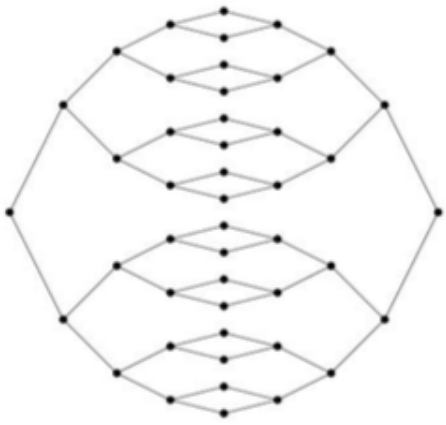

Figure 3: The graph G4. Obtained from [8]

The relevant problem at hand is this. If we start at the left-most root in $G_n$, how many steps on average in a RW do we need to take until we end up at the right-most root? This question is asking for the hitting time if we start at the left-most root, where the right-most root is the only candidate solution state.

It turns out that the symmetry of $G_n$ allows us to reduce it to a RW on a line (RWL) with $2n+1$ positions [8]. The reduction consists of clumping the vertices at each level together into a single representative point for that level. For our example, the reduced $G_4$ is depicted below:

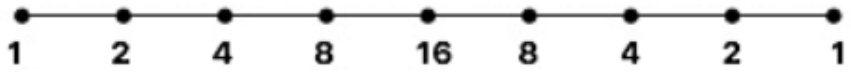


Figure 4: The resulting reduction of $G_4$ to a RWL. The $i^{th}$ point from the left,$0 \leq i \leq 8$, represents the nodes at level i for the left binary tree for $0 \leq i \leq 4$ and for $0 \leq i \leq 8$, the nodes at level $8-i$ for the right binary tree. Each point is labeled with the total number of nodes that it represents from Fig.3. Obtained from [8]

Performing a CRWL on the reduced $G_n$, the expected hitting time is $O(2^n)$ [8]. To do a QRWL on $G_n$, we first describe each point $i$ on the line by the following basis vector:

$$|i\rangle = \frac{1}{\sqrt{N_i}} \sum_{a \notin column_i} |a\rangle$$

where $N_i$ is the number of nodes represented by point $i$ and $column_i$ consists of the nodes at level i for the left binary tree when $0 \leq i \leq n$ and for $0 \leq i \leq 2n$, the nodes at level $2n-1$ for the right binary tree [8]. Note that $|i\rangle$ is a uniform superposition of all the nodes at column i. Thus, the matrix elements of H become [8]:

$$\begin{aligned} \langle j| H |j \pm 1\rangle &= -\sqrt{2}\gamma \\ \langle j| H |j\rangle &= \begin{cases} 2\gamma & j = 0, n, 2n \\ 3\gamma & otherwise \end{cases} \end{aligned} \tag{10}$$

where to gain some intuition on H, the diagonal entries come from the fact that the degree of the left-most root, middle nodes, and the right-most root is 2 and all other nodes is 3 (1 for the parent, 2 for the children) âĂŞ see $G-4$ in Fig 3 for an example. With this reduction, the hitting time for the QRW on $G_n$ is $O(n^2)$ [2]. This is an exponential speed-up over the equivalent, classical counterpart, although the problem is highly theoretical.

## 5 Quantum Walk Search

The problems that we will present in Sections 6, 7, and 8 fall under the general category of a QWSP. Here's what a QWSP is. We have a database $D$ of $N$ entries with $M \subseteq N$ marked entries. Our problem asks us to search through $D$ for a marked entry $m \in M$. If we model the database as a graph with $N$ vertices and we associate with each vertex $v$ its corresponding database entry $D(v)$, then the problem reduces to a RW on a graph where we stop upon finding a marked vertex. There are five relevant costs to any algorithm that falls under the QWSP framework [9], [5].

1. **Set-up cost** S. This is the cost of setting up a single database entry $D(v)$ for a given vertex $v$.
2. **Update cost** U. This is the cost associated with transitioning from a vertex $u$ to a vertex $v$ and then updating $d(u)$ to $d(v)$.

3. **Checking cost** C. This is the cost of checking if $u$ is a marked vertex – it is done via. amplitude amplification as described in Grover's algorithm [6].

4. **Initialization time** I. This is the time complexity of setting up the initial superposition.

5. **Transition time** T. This is the time complexity of a single transition in the QMC

The overall cost of the algorithm is some summation of these five costs, depending on which model is used to carry out the RW over the graph. There is no general formula yielding this cost for algorithms that use a QRWG under the CWM described in Section 3.1 – this is what Section 6 uses. However for a QMC, which is what Sections 7 and 8 use, there are some helpful theorems.

**Theorem 1.** *Existence of a Marked Element under the QMC*

*Let $\delta$ be the eigenvalue gap of a MC's transition matrix $P$ with $V$ being the set of states and $M \subseteq V$ the set of marked states. Let $\frac{|M|}{|V|} \geq \epsilon > 0$ when $M$ is non-empty. Then there exists a quantum algorithm that determines if $M$ is non-empty with cost $S + O((U+C)/\sqrt{\delta\epsilon})$, and an additional time complexity of $I + O(T/\sqrt{\delta\epsilon})$.*

where the eigenvalue gap of a matrix $A$ is the difference between its largest and second largest eigenvalues. Theorem 1 was obtained from [9] - it essentially establishes an upper bound on determining the existence of a marked element.

**Theorem 2.** *Estimate of the Hitting Time for Detection*

*Given a reversible, ergodic MC $P$ with marked elements $M \subset V$, there exists a quantum algorithm that, with probability 2/3, outputs an estimate $h' \in O(HT_{eff}(P, M))$ of the number of steps we can apply the absorbing walk $P'$ starting from the initial, stationary distribution $\pi$ to stumble upon a marked element with probability 3/4. Its cost is $O(S + \sqrt{h}(U + C))$.*

where $HT_eff(P, M)$ is the effective hitting time, i.e. the number of steps of $P'$ required to reach a marked vertex with probability 2/3, starting from a random unmarked vertex that's selected based on $\pi$. Theorem 2 is found in [7].

# 6 Subset Finding

## 6.1 Problem Statement

We are given a black-box function $f : D \to R$ and a property $\mathcal{P} \subset (D \times R)^L$ that must be satisfied. The domain $D$ and range $R$ are finite sets, and $|D| = N$ is the problem size. The question asks us to find an L-subset $x_1, .....x_L \subset D$ such that

$$((x_1, f(x_1)), (x_2, f(x_2)), .....(x_L, f(x_L))) \in \mathcal{P}$$

or output REJECT if none exists. An example instance of this general problem is the L-element distinctness problem. Given a list $M = [X_1, x_2, .....x_N]$, the L-element distinctness problem asks if there exist L copies of the same number. For our problem, this is the same as finding an L-subset with $D = [1, ....N]$ be the set of indices, $R = M$ be the elements, $f(i) \to x_i$, and $\mathcal{P} = ((i_1, y), ..., (i_L, y))$.

For L = 1, this problem reduces to unstructured search that can be solved in $O(\sqrt{N})$ queries. For L = 2, we have a lower bound of $\Omega(N^{\frac{2}{3}})$. This comes from the fact that we can reduce the 2- element distinctness problem, which has a lower bound of $\Omega(N^{\frac{2}{3}})$[??] to an instance of this problem.

### 6.2 Algorithm

At a high-level, the algorithm boils down to a QRW of a bipartite graph (*not* to be confused with the one presented in Section 3.2). It consists of the following steps:

1. Repeat $t_2$ times starting with an initial state $|\psi_0\rangle$
   - Perform the phase flip operation P.
   - Walk for $t_1$ steps
2. Measure the final state $|\psi_F\rangle$to get an outcome $P_c$ . If $P_c \in \mathcal{P}$ then output $P_c$ . Otherwise, REJECT

with $t_1 = \lfloor \frac{\pi}{2}\sqrt{\frac{M}{L}} \rceil$ and $t_2 = \frac{\pi}{2}(\frac{N}{M})^{\frac{L}{2}}$, where $\lfloor \rceil$ denotes rounding to the nearest integer. The vertices V of the graph G are all subsets of D that are size M or M+1. Mis chosen to be $\Theta(N^q)$ for 0 < q < 1 since |D| = N. Let $A, B \in V$such that $|A| = M$ and $|B| = M + 1$. Vertices A, B are connected iff $|A \cap B| = M$ so that $B = A \cup K$ for $k \in D - A$ - i.e., A and B are sets who differ by one element where B is a superset of A. We have M associated function values $f(A) \in R^M$ for A and M + 1 associated function values $f(B) \in R^{M+1}$for B – these values are stored in the algorithm along with the subset. Thus, the Walker's position state is $|A, f(A)\rangle$ – think of f(A) as being part of the vertex A. The coin-state for A is $|k\rangle$ where $k \in D - A$, and the coin-state for B is $|j\rangle$ where $j \in B$. The coin operators $c_1$ and $C_2$ for $k \notin A$ and $k \in B$, respectively, are the Grover diffusion operators defined as follows:

$$\begin{aligned} C_1 |A, f(A), k\rangle &= |A, f(A), k\rangle - \frac{2}{N-M} \sum_{k' \notin A} \left| A, f(A), k' \right\rangle \\ C_1 |B, f(B), k\rangle &= |B, f(B), k\rangle \end{aligned} \tag{11}$$

$$\begin{aligned} C_2 |B, f(B), k\rangle &= |B, f(B), k\rangle \\ C_2 |B, f(B), k\rangle &= |B, f(B), k\rangle - \frac{2}{M+1} \sum_{k' \in B} \left| B, f(B), k' \right\rangle \end{aligned} \tag{12}$$

and the shift operation S is as follows:

$$\begin{aligned} S |A, f(A), k\rangle &= |A \cup k, f(A \cup k), k\rangle \\ S |B, f(B), k\rangle &= |B\ k, f(A\ k), k\rangle \end{aligned} \tag{13}$$

And a single step of the walk corresponds to the unitary $W = SC_2SC_1$. For simplicity, assume that there is exactly one L-subset $S = x_1, ....., x_L \subset D$ such that $((x_1, f(x_1)), (x_2, f(x_2)), .....(x_L, f(x_L))) \in \mathcal{P}$. The phase flip operation P is then defined as follows for some subset C:

$$P |C, f(C)\rangle = \begin{cases} -|C, f(C)\rangle & S \subset C \\ |C, f(c)\rangle & S \not\subset C \end{cases} \tag{14}$$

The initial state for the algorithm is then a uniform superposition on subsets of size M:

$$|\psi_0\rangle = \frac{1}{\sqrt{c}} \sum_{|A|=M} |A, f(A)\rangle \sum_{k' \notin A} |k\rangle \tag{15}$$

where $c = \binom{N}{M}(N-M)$. Then, the algorithm step is $(W^{t_1}P)^{t_2}$ . Intuitively, it first magnifies the candidates C where $S \subset C$ through P in a similar manner to how Grover's algorithm magnifies the marked elements in an unstructured search problem. From these candidates, it attempts to construct S through a random walk by adding and removing elements until, at the end of the algorithm, it outputs either S, or REJECT if S does not exist, with a high probability of a correct answer. The overall complexity of the algorithm is $O(N^{\frac{L}{L+1}})$. Note: The authors state that the algorithm can be modified using standard sampling techniques to handle multiple candidate L-subsets.

# 7 Group Commutativity

We are given a black-box group $G$ with generators $g_1, \ldots, g_k$ where $g_1$ is the identity element. Here, the non-commutative elements in $G$ are the marked items. At a high-level, the algorithm quantizes a MC and then invokes Theorem 1 to show that a RW under this QMC will determine whether a non-commutative element in $G$ exists. Here's how it works. Let $S_l$ denote the set of all distinct $l$-tuples formed by the indices $\{1, \ldots, k\}$ where $l = o(k)$. For a tuple $u = (u_1, \ldots, u_l) \in S_l$, $g_u$ denotes the group element $g_{u_1} \ldots g_{u_l}$ where the group operation is implicit in the multiplication. For each $u$, we store a balanced binary tree $t_u$ with $l$ leaves; these leaves, from left-to-right, are the elements $g_{u_i}$ for $1 \leq i \leq n$. If $l$ is not a power of 2, then put the deepest leaves to the left. The internal nodes correspond to the group product of its children so that at the root, we have $g_{u_1}, \ldots, g_{u_l}$.

The random walk will be on $S_l^2$ (i.e. pairs of $l$-tuples of group elements). For a pair $(u, v)$ of $l$-tuples, the walk $S_l^2$ is two independent walks on $S_l$ – one on $u$ and one on $v$. The walk $S_l$ is described as follows. Assume without loss of generality that the current state is $u \in S_l$. Then we flip a fair-coin and do one of the following:

1. Stay at $u$
2. Pick some position $i \in \{1, \ldots, l\}$ and an index $j \in \{1, \ldots, k\}$. If $j = u_m$ for some $m$, then swap $u_i$ and $u_m$. Otherwise set $u_i = j$. Update the tree $t_u$ using $O(\log l)$ group operations. The update step is like updating a heap – we start at the modified leaf, traverse up to its parent and re-compute the internal node's group element, traverse up to its parent and do the same thing, etc., until we are at the root.

The initial state is the uniform superposition

$$\sum_{u \in S_l} \sum_{v \in S_l} |u, t_u\rangle \, |v, t_v\rangle \tag{16}$$

Using $S_l^2$ as our walk, the overall complexity of this algorithm is $O(k^{2/3} \log k)$ queries and $O(k^{2/3} \log^2 k)$ time. Intuitively, $S_l$ randomly decides to either leave $u$ alone (Case 1), or choose to modify it (Case 2). The modification randomly tries to either change $u$ to a non-commuting group element by swapping two of its existing elements with each other, or it "gives" up on the current $u$ by substituting a randomly chosen, different element for one of its existing elements so that it can try to see if it later finds a non-commuting group element using the resulting, new $l$-tuple $u'$.

## 8 Quantum Random Walk on a Torus

Here we have our database represented by an $N \times N$ torus graph, which is like a 2-D grid except that the boundary vertices connect to vertices on the opposite side of the boundary. For example, the vertex in the top-left corner connects to both the vertex at the bottom-left corner, and the vertex at the top-right corner. The problem is defined as follows. Given an $N \times N$ torus $G_{torus}$ with some subset $M \subset V$ marked vertices, either find a marked vertex or output "unsuccessful search". The algorithm is comprised of seven steps. First, let $H_{unique} = HT_{eff}(G_{torus}, \{m\}) \in \Theta(N \log N)$ be the effective hitting time for the torus when there is a unique marked element.

1. Compute the estimate $h'$ outlined in Theorem 2. If the algorithm does not halt after $\sqrt{H_{unique}}$ steps, halt it and use $H_{unique}$ to estimate $h'$
2. Set $d = 2\left\lceil 4\sqrt{h'} \right\rceil$. If $d > N$, set $d = N$.
3. Divide the $N \times N$ torus into $\Theta(\left(\frac{N}{d}\right)^2)$disjoint sub-grids so that each sub-grid has length between $D$ and $D+1$ for some $d \leq D < 2d$.
4. Create the initial state $|\pi\rangle = \sum_{v \in G_{torus}} \sqrt{\pi_v} |v\rangle$ over all the vertices in the torus corresponding to its stationary distribution.
5. For each vertex $v \in G_{torus}$, keep an ancilla register that indicates the name of the sub-grid $v$ belongs to, i.e. the superposn. becomes $\sum_{v \in G_{torus}} \sqrt{\pi_v} |v\rangle |subgrid(v)\rangle$.
6. Set $\epsilon' = 1/2^k$ where $k$ is uniformly picked at random s.t. $1 \leq 2^k < N$
7. Run an absorbed quantum walk on each sub-grid for $\Theta(D\sqrt{\log D})$ steps with estimate $\epsilon'$ in superposition over all sub-grids. The walk is conditioned on the name of the sub-grid in the ancilla register.
8. Measure the final state, which is a vertex $v$ of the torus. If $v$ is marked, then output $v$. Otherwise, output "unsuccessful search".

The algorithm was copied pretty much the same as it's written in [7]. Here's the intuition. The bulk of the algorithm is in Step 7, which does an absorbed RW on each of the subgrids constructed in Step 5. We establish the dimensions of each sub-grid using our estimate obtained from Step 1 – the value $d$ in Step 2 was chosen because [7] writes that RWs in a torus tend to be localized. Specifically, the walker is at a distance $O(\sqrt{T})$ after $T$ steps of the RW in all directions of the torus. In essence, if we do a walk on the composite torus, we are being inefficient because $O(\sqrt{T})$ is not encompassing the entire surface area of the torus. Instead, what the authors proposed is to first figure out how far we would have to go to find a marked element in the torus alone – this is $h'$. Then they proceed to partition the torus into sub-grids of $O(h')$ dimensions – we take advantage of the locality because we know that *inside these grids*, we have a much higher chance of hitting *a* marked element. The resulting complexity of the algorithm is $S + \min\{\sqrt{H \log H}, \sqrt{N \log N}\}(U + C)$ where $H$ is the effective hitting time.

## 9 Conclusion & Open Problems

We have given a theoretical and practical presentation of QRWs. Specifically, we have presented two models – the DTM and the CTM – and showed that within both models, the quantum counterparts of certain problems results in interesting theoretical differences between the quantum and

classical worlds. In particular, the QRW propagates faster than the CRW. Finally, we have presented a specific application of QRWs – QWSPs – and discussed three algorithms, subset finding, group commutativity, and a QRW on a torus, that fit under that framework. All three algorithms exhibit polynomial time speed-ups from their classical counterparts.

Some interesting open problems in the field are the relationship between the DTM and the CTM – it is currently not clear how to transform an instance of one into the other. Finally, to the authors' knowledge, QWS is the main practical application that is unique to the QRW. Finding more natural problems that get a speedup from applying the QRW is also another open problem.

## References


[1] Random walks. `http://www.mit.edu/~kardar/teaching/projects/chemotaxis%28AndreaSchmidt%29/random.htm`, note = Accessed: 06-16-2017.

[2] AMBAINIS, A. Quantum walks and their algorithmic applications. *Journal of Quantum Information* (2003), 507–518.

[3] CHUNG, F., AND WENBO ZHAO, J. Pagerank and random walks on graphs. *Fete of Combinatorics and Computer Science Bolyai Society Mathematical Studies* (2010), 43–62.

[4] ELIAS, S., AND VENEGAS-ANDRACA. Quantum walks: a comprehensive review. *Quantum Information Processing 11*, 5 (August 2012), 1015–1106.

[5] FREDERIC MAGNIEZ, ASHWIN NAYAK, J. R., AND SANTHA, M. Search via quantum walk. *SIAM Journal on Computing 40*, 1 (2011), 142–164.

[6] GROVER, L. K. A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC 96Âă* (1996).

[7] HOYER, P., AND KOMEILI, M. Efficient quantum walk on the grid with multiple marked elements. *Algorithmica* (December 2016).

[8] KEMPE, J. Quantum random walks: an introductory overview. *Contemporary Physics 50*, 1 (2009), 339–359.

[9] MAGNIEZ, F., AND NAYAK, A. Quantum complexity of testing group commutativity. *Algorithmica 48*, 3 (July 2007), 221–232.

[10] SNELL, J., AND GRINSTEAD, C. M. *The introduction to probability*. McGraw-Hill, 2006.

[11] SZEGEDY, M. Quantum speed-up of markov chain based algorithms. *45th Annual IEEE Symposium on Foundations of Computer Science* (April 2004).