Quantum Walks and Resetting

Dhruva Sambrani

A dissertation submitted for the partial fulfilment of BS-MS dual degree in Science

Indian Institute of Science Education and Research, Mohali

2023-04-02

Certificate of Examination

This is to certify that the dissertation titled **Quantum Walks and Resetting** submitted by **Dhruva Sambrani** (Reg. No. MS18163) for the partial fulfillment of BS- MS Dual Degree programme of the institute, has been examined by the thesis committee duly appointed by the institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Dr. Manabendra Nath Bera Dr. Abhishek Chaudhary Dr. Sandeep Goyal

Dr. Manabendra Nath Bera
(Supervisor)

Dated: 2023-04-02

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Dr. Manabendra Nath Bera at the Indian Institute of Science Education and Research, Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Dhruva Sambrani
(Candidate)
Dated: 2023-04-02

In my capacity as the supervisor of the candidates project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Dr. Manabendra Nath Bera
(Supervisor)
Dated: 2023-04-02

Acknowledgements

TODO: Acknowledgements

Abstract

TODO: Abstract

List of Figures

List of Tables

Table of contents

1 Introduction

Random walks are a commonly used tool in the arsenal of algorithms on Classical Computers to solve a variety of problems which do not have a known easy solution. The power of such methods can generally be attributed to the fact that while the space of possible solutions is vast, we generally only need to sample few solutions to come to a close solution. This power has been routinely exploited in the past to sample from Markov Chains using the MCMC algorithm which can be found in any introductory textbook[1] of markov processes, and they have recently been used in the fields of Financial engineering[2], fluid mechanics[3], fitting blackhole images[4] and most famously in the Los Alamos project[5].

With the advent of quantum algorithms, quantum walks have arisen as an obvious extension of classical walks in the quantum domain. The applications of quantum walks are just as many: ANN training[6], Random Number Generation[7], List coloring (Grover)[8], collision finding[9], link prediction[10]. There has even been a recent foray into classifying and using a quantum-classical walk to speed up certain classical algorithms, namely the Google PageRank Algorithm[11]. The increased interest in quantum walks can be attributed to the quadratic speed up which it grants the solution a-la the Grover algorithm, which itself can be considered as a Quantum Walk[12]. While there is also interest in studying the physical implementation of the walks, we do not concern ourselves with these questions, since experimentalists are much better equipped to ask them.

In this master thesis, we will define both the classical (Chapter 2) and quantum (Chapter 3) walks along with certain properties that are, while interesting in their own regard, problematic for search algorithms (Chapter 4). Then, we look at previous attempts at solving them by resetting (Chapter 5), and discuss the shortcomings of the solution. Finally, we introduce a new mechanism (Chapter 6) which we believe is more general and may show interesting phenomena with particular advantage in the search problem.

Along with the theory, certain aspects of the implementation of the walks computationally are also added as necessary. This is done inline instead of in an appendix, which is the norm, since the author believes that such a presentation solidifies the readers' understanding of both, the model and the implementation.

2 Classical Walks

Classical random walks are defined on a graph, with the walker being on some initial node i_0 with a probability λ_i, and "hopping" from node i to j in each time step with a probability given by p_{ij}. We can thus define a transition matrix $P := P_{ij} = p_{ij}$ which is row stochastic. Thus the probability mass function of the walker at timestep t is given by λP^t. Of course, the exact structure of the graph and the probabilities will decide the properties of the walk and there exists a vast amount of literature devoted to this analysis.

We however will restrict our discussion to the symmetric walk on the 1D chain. This is often called the 1D-Simple Symetric Random Walk, and defined in the following way.

2.1 Definition

Consider an infinite 1 D chain, with nodes marked by $\mathbb{Z}$.

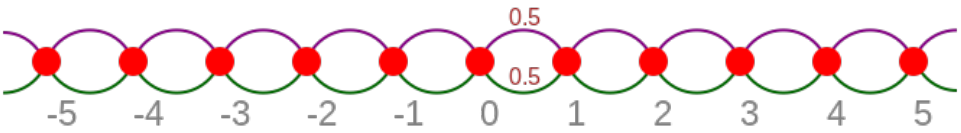


Figure 2.1: 1 D Chain

Define the probability of hopping from node i to node j

$$p_{ij} = \begin{cases} 1/2 & |i-j| = 1 \\ 0 & \text{otherwise} \end{cases}$$

And the initial state as

$$\lambda_{ij} = \begin{cases} 1 & i = 0 \\ 0 & \text{otherwise} \end{cases}$$

Thus, the hopper can only move to it nearest neighbors, starting from node 0.

Note that $P(X_t = i|$ HISTORY $) = P(X_t = i|X_{t-1})$, which means that the walk is a markov chain.

2.2 Multiple Walkers

The SSRW is clearly a stochastic process, and each run of this process will lead to different paths being chosen by the walker, and we are my interested in what the walker does on an average rather than what happens in a particular instance. Thus, we can observe multiple walks, and plot their paths to visualize how they would spread.

In order to simulate a walk, we sample $X_i \in \{-1, 1\}$ with equal probability, and add it to the previous position S_{i-1} to get S_i. Note $S_0 = 0$. Thus, this is equivalent to sampling from `[1, -1]` uniformly `t` times and performing a cumulative sum. For `n` walkers, we can sample `(t, n)` such random numbers, and do a cumulative sum along the first dimension.

```
bm = cumsum(rand([1, -1], (200, 15)), dims=1);
```

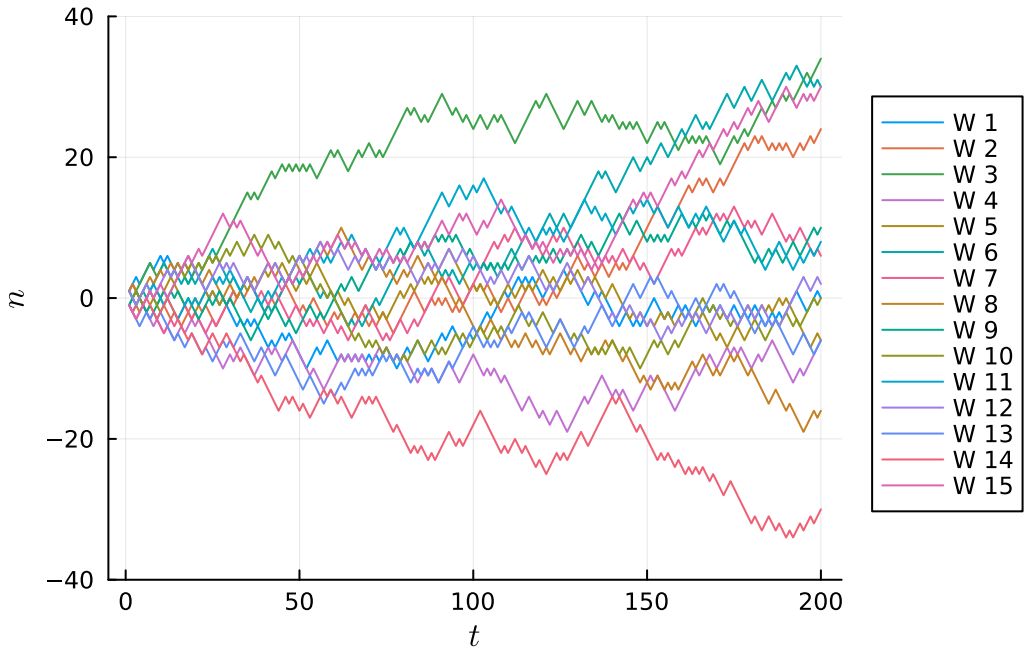


Figure 2.2: An ensemble of classical walkers

2.3 Probability mass function

Another common way to visualize the walk is to plot the probability that the walker is on node i at time t. For this, we simply define the transition matrix and λ appropriately and find λP^t. Note however that the transition matrix for the SSRW is

Tridiagonal with the Upper and Lower diagonals as 0.5 and the diagonal as 0, and there exist efficient storage and multiplication routines. Also, since we can only store a finite matrix, we limit the walk to some size. At the boundaries, we simply allow open conditions, because this is the easiest to implement.

```
cps, cps_t = let
    t = 21
    U = SymTridiagonal(fill(0., 31), fill(0.5, 30))
    λ = fill(0., 31)
    λ[16] = 1
    ps = accumulate(1:t, init=λ) do old, _
        U * old
    end[t÷3:t÷3:t], t÷3
end;
```

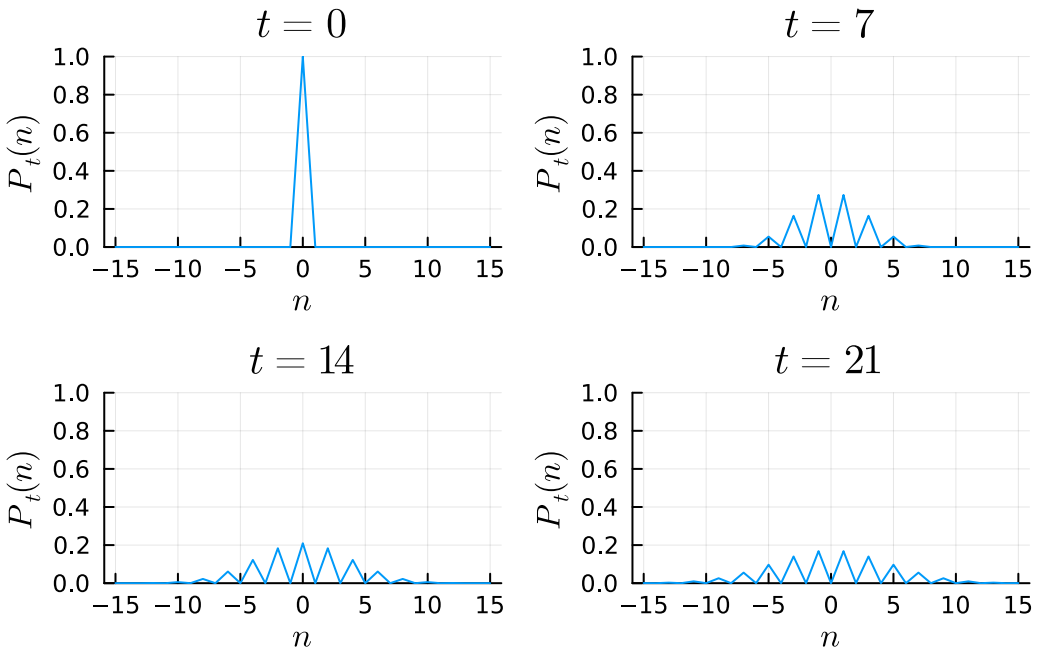


Figure 2.3: Probability distribution of a classical walker in time

2.4 Properties of the walk

There are certain properties of the walk that are interesting for the problem we pose in the subsequent sections. Primarily, we are interested in the mean, standard deviation and recurrence of the graph.

The probability that a walker is on node n at time t is given by the expression

$$p(n,t) = \binom{t}{\frac{t+n}{2}} \frac{1}{2^t}$$

This equation is valid only if $t + n$ is even and $n \leq t$. If $t + n$ is odd or $n > t$, the probability is zero

It should be obvious from the even symmetry of $p(n, t)$ that the mean $\mu(t) = \sum_n np(n, t) = 0$. This comes from the fact that the walk is symmetric.

The standard deviation of the walker position $\sigma(t) = \sqrt{\langle n^2 \rangle - \langle n \rangle^2} = \sqrt{\sum_n n^2 p(n, t)} = \sqrt{t}$[13].

3 Quantum Walks

3.1 Introduction

Quantum walks are defined analogous to the classical walks.

First, our walker is quantum mechanical, and the position of the walker can be a superposition of the nodes. Thus, we define the state of the walker to be a superposition of the node states $|i\rangle$.

$$|\psi\rangle = \sum_i c_i |i\rangle$$

Let this Hilbert space be denoted as $\mathcal{H}_W$. The projection of the state on some node i given by $|\langle i|\psi\rangle|^2 = |c_i|^2$ is understood as the probability that the walker will collapse to the state $|i\rangle$ on measurement.

To define the evolution of the node, we look back at the transition matrix P_{ij}. Thus we would define an operation in the following manner -

$$O|i\rangle = \sum_j \sqrt{p_{ij}}|j\rangle$$

and the walk proceeds by repeated application of O.

While this expression is enough to define the walk, it is not immediately clear how one would explicitly realize such an operation. There are multiple formalisms to define such walks, but we shall use the coined quantum walk formalism which is very natural for regular graphs, as is the case for the 1D chain. Note that for the symmetric walks on nD lattices, like the 1D chain, the tight-binding model is already a well understood continous time quantum walk. However, we prefer a discrete time formalism.

3.2 Coined Quantum Walk for 1D Chain

For the 1D chain, given a specific node, there are only 2 other nodes connected to it. Thus, we can add a 2 level system, which "decides" which node the walker jumps to. More formally, we attach a 2 level qubit system to the walker whose bases are denoted by $|0\rangle$ and $|1\rangle$. Let us denote this hilbert space as $\mathcal{H}_C$

Thus, we can define the shift operation as

$$S|0\rangle|i\rangle = |0\rangle|i-1\rangle; S|1\rangle|i\rangle = |1\rangle|i+1\rangle$$

How would we define such an operation explicitly?

$$S = |0\rangle\langle 0| \otimes S_L + |1\rangle\langle 1| \otimes S_R$$

Where S_L and S_R are defined as

$$S_L|i\rangle = |i-1\rangle,\ S_R|i\rangle = |i+1\rangle$$

Note that S operates on $\mathcal{H}_C \otimes \mathcal{H}_W$, whereas S_L and S_R operate on $\mathcal{H}_W$ only.

The superposition in the two choices at each step is recovered by putting the coin into a superposition of its basis states. This is achieved via a coin operator, which is commonly defined as $H \otimes I$, where H is the single qubit hadamard operator.

Let us explicitly write down two steps of the walk

$$H \otimes I(|0\rangle|0\rangle) = \frac{|0\rangle|0\rangle + |1\rangle|0\rangle}{\sqrt{2}}$$

$$S\left(\frac{|0\rangle|0\rangle + |1\rangle|0\rangle}{\sqrt{2}}\right) = \frac{|0\rangle|-1\rangle + |1\rangle|1\rangle}{\sqrt{2}}$$

$$H \otimes I\frac{|0\rangle|-1\rangle + |1\rangle|1\rangle}{\sqrt{2}} = \frac{|0\rangle|-1\rangle + |1\rangle|-1\rangle + |0\rangle|1\rangle - |1\rangle|1\rangle}{2}$$

$$S\frac{|0\rangle|-1\rangle + |1\rangle|-1\rangle + |0\rangle|1\rangle - |1\rangle|1\rangle}{2} = \frac{|0\rangle|-2\rangle + (|1\rangle + |0\rangle)|0\rangle - |1\rangle|2\rangle}{2}$$

> **Repeated application of Coin Operator**
>
> Note specifically the repeated application of the coin operator. If the coin operator is not applied in step 3, our state will end up in $\frac{1}{\sqrt{2}}(|0\rangle|-2\rangle + |1\rangle|2\rangle)$ which is not what we wanted. This is because the application of the controlled shift operation entangles the coin and the walker systems, and thus there is no superposition in each term of the system

3.3 Computational Implementation

There are 2 ways to implement the Quantum walks, which are both interesting in their own ways. But the foremost thing to tackle would be how to store an infinite vector and an infinite dimensional operator. Since we cannot do either of these simply, we instead limit our walk to a node space of $2n$, and use periodic boundary conditions.

> ⚠ Other Boundary Conditions
>
> One could pick other boundary conditions too, such as the open or the absorbing boundary conditions, but both of these BCs lead to nonunitary operations, which complicate the definition and application of the gate.

3.3.1 Matrix formalism

The first and more immediate method is to simply write down the matrix equivalent of the above operations. For the visualization of the implementations, we will assume that the walk occurs in a 1 D chain of size 4.

The coin is prefered to be in the $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ when we start so that the walk proceeds symmetrically[13].

```
init_coin = 1/√2 * (ket(1,2) - 1im * ket(2,2));
```

The coin operator is trivial to write,

```
H = KrausOperators([sparse(hadamard(2)⊗I(4))]);
```

The left and right shift operators can be defined in the following manner.

```
R = collect(Tridiagonal(fill(1., 3), zeros(4), zeros(3)))
R[1, end] = 1
L = collect(Tridiagonal(zeros(3), zeros(4), fill(1., 3)))
L[end, 1] = 1
```

Thus the shift operator is defined as

```
S = KrausOperators([sparse(proj(ket(1, 2)) ⊗ L + proj(ket(2, 2)) ⊗ R)]);
```

Thus, we can repeatedly apply $S \circ H$ to the initial state and accumulate the results.

```
init_state = proj(init_coin ⊗ ket(2,4))
ψ = [[init_state]; accumulate(1:40, init=init_state) do old, _
    H(S(old))
end];
```

Now we can plot the readout statistics by partially tracing out the coin, and plotting the diagonal. The following is a walk on 61 nodes.

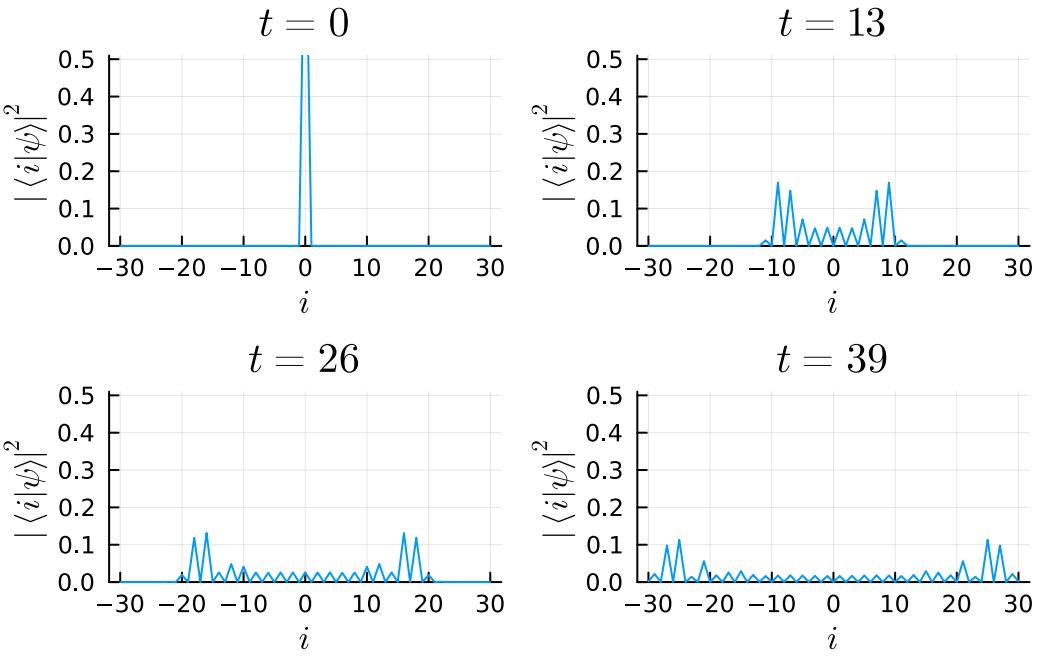


Figure 3.1: Probability distribution of a Quantum Walker in time

> **Quantum Walks are not random**
>
> Note, this is a single walker, not an ensemble of walkers as is the case in classical random walks. Quantum walks (without measurement), are completely deterministic.

3.3.2 Circuit Formalism

While the matrix definition of the Quantum Walks are easy to formalize and understand, finally these need to be simulated on some sort of device which may require us to reformulate it. Further, we try to ensure that the reformulation allows for easy additions of other dynamics which we may want to study.

The advantages of using a Quantum Computer over a classical computer for a quantum walk should be obvious. The problem however is that the quantum walk is over a high dimensional space, and we rarely have access to such high dimensional systems which we can control easily. Instead we need to simulate such a system using the accessible 2 level systems available in quantum computers.

Let us define the basic components of our walk.

3.3.2.1 Nodes

- Each numbered node is then converted into its 2-ary n length representation denoted by (x_i). n is chosen such that $2^n > N$

- These bitstrings are encoded into an n qubit computer, where each basis state in the computational basis corresponds to the node with the same bitstring.
- The amplitude of a particular basis corresponds to the amplitude of the walker in the corresponding node

3.3.2.2 Coin

- The Walk Coin is a two level system, as usual

3.3.2.3 Edges and Shifts

- Since shifts are only to adjacent nodes, the left (right) shift is equivalent of subtracting (adding) 1 from the bitstring of the state.
- From the Quantum adder circuit, we can set one input to be $(0)_{n-1}1$ and reduce the circuit to get the Quantum AddOne circuit. Shown below is the circuit for $n=4 $ ($N = 16$)
- We can similarly construct the SubOne circuit, but that is simplified by noting that the subone circuit is simply the inverse of the addone circuit, and this corresponds to just inverting the circuit (all gates are unitary).” “ ”

```
rightshift(n) = chain(
    n,
    map(n:-1:2) do i
        control(1:i-1, i⇒X) end...,
    put(1⇒X)
)
leftshift(n) = rightshift(n)';
```

```
YaoPlots.plot(rightshift(4))
```

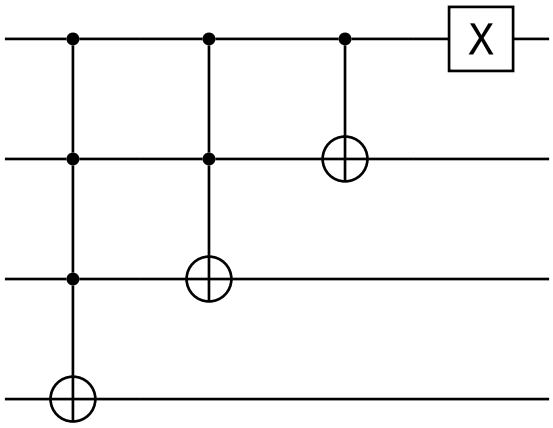


Figure 3.2: Right shift circuit

```
YaoPlots.plot(leftshift(4))
```

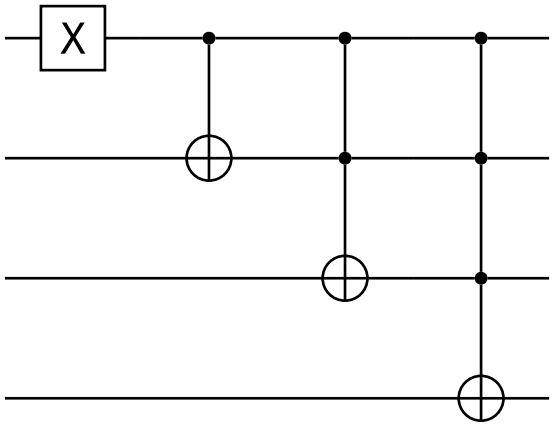


Figure 3.3: Left shift circuit

The controlled shift operation is encoded as

```
shift(n) = chain(n+1,
    control(1, 2:n+1 ⇒ rightshift(n)),
    put(1 ⇒ X),
    control(1, 2:n+1 ⇒ leftshift(n)),
    put(1 ⇒ X),
)

YaoPlots.plot(shift(4))
```

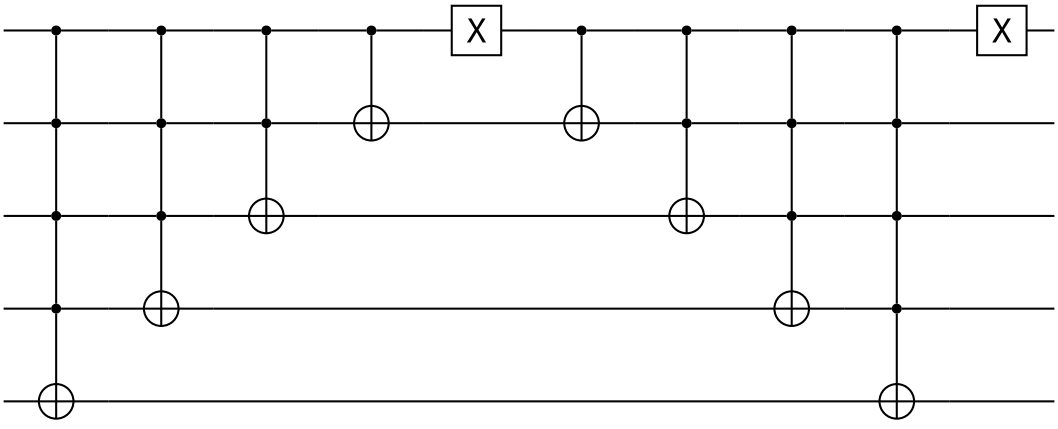


Figure 3.4: Controlled shift circuit

3.3.2.4 Coin operator

We can add the coin operator as usual on the first rail.

```
coin(n, c=Yao.H) = chain(n+1, put(1 => c))
YaoPlots.plot(coin(4))
```

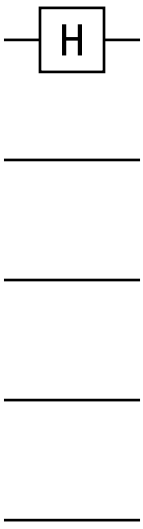


Figure 3.5: Coin Operator circuit

3.3.2.5 Evolve Circuit

Putting these together, we get the operation for a single step of the evolution as below. Note that the top qubit rail is that of the coin, and the rest are those of the simulation of the system.

This circuit can be repeated to acheive any number of steps.

```
evolve(n) = shift(n) * coin(n)
YaoPlots.plot(evolve(4))
```

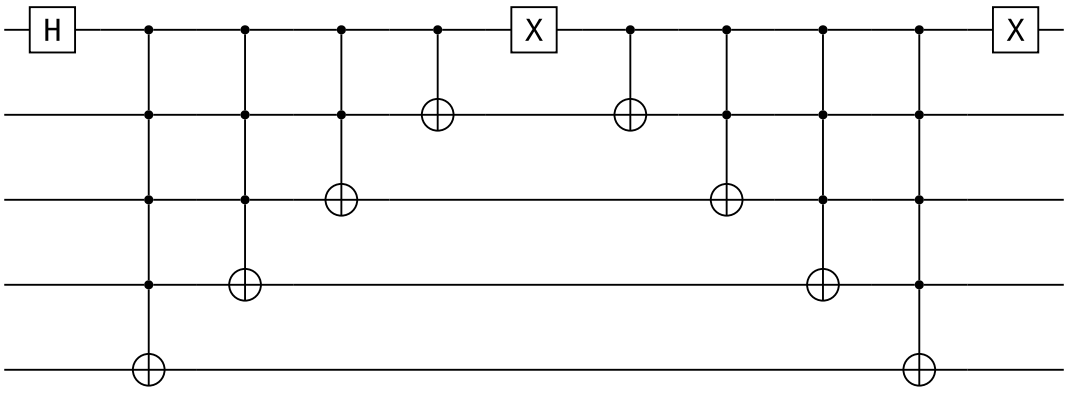


Figure 3.6: Quantum walk evolve circuit

3.3.2.6 Prepare circuit

While we can already simulate the walk, an very useful helper function that we can define is the prepare circuit.

Quantum registers are often initialized to the 0 state, and it is also easy to restart the walk from the 0 state. However, we often like to start the walk in the center of the chain instead of at node 0. Also, the coin is prefered to be in the $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ when we start so that the walk proceeds symmetrically[13].

Hence, to perform these steps, we define the following `prepare` subroutine.

```
prepare(n) = chain(
    n+1,
    put(1⇒Yao.H),
    put(1⇒Yao.shift(-π/2)),
    put(n+1⇒X)
)

YaoPlots.plot(prepare(3))
```

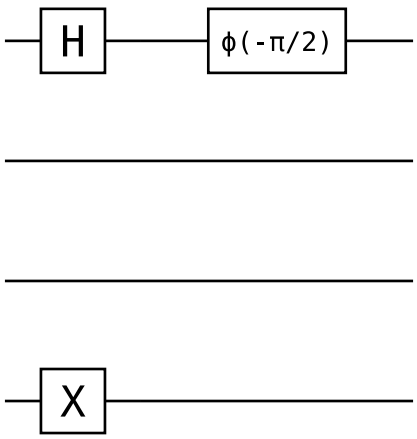


Figure 3.7: Initial state preparation circuit

Similarly plotting as before,

Thus we can see that the two formalisms are the same.

3.4 Properties of the walk

Similar to the classical random walk, the mean $\mu(t) = 0$.

However, a more interesting feature is that the standard deviation $\sigma(t) = 0.54t$[13]. Compare this with the classical random walk, the quantum walk has a quadratic speed up. This the reason for the quadratic speedup commonly seen in the Grover search and other Monte Carlo problems.

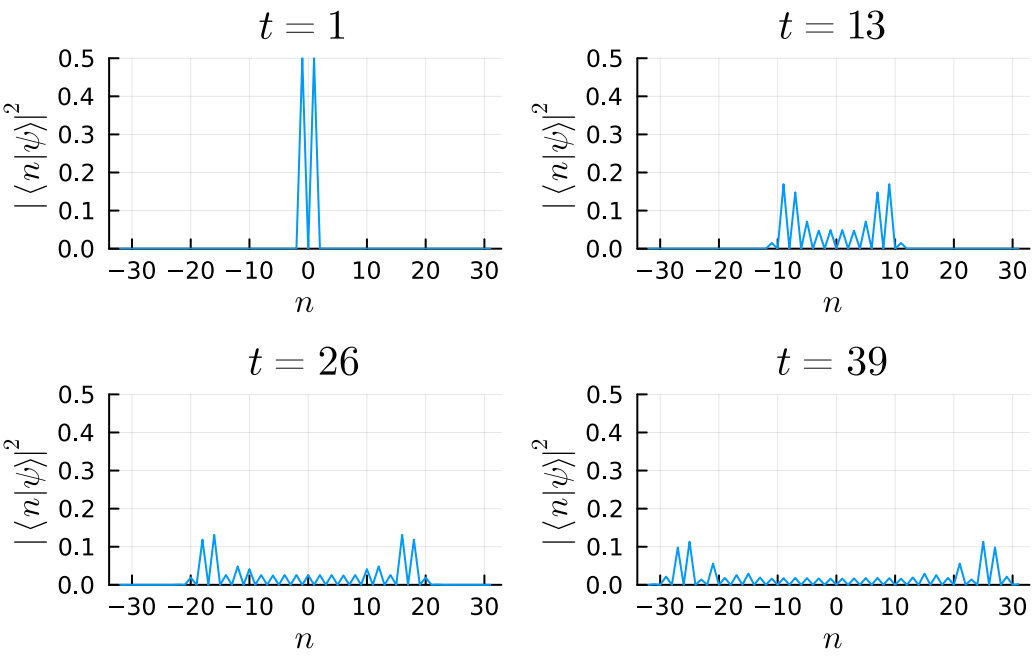


Figure 3.8: Quantum walk using circuit formalism

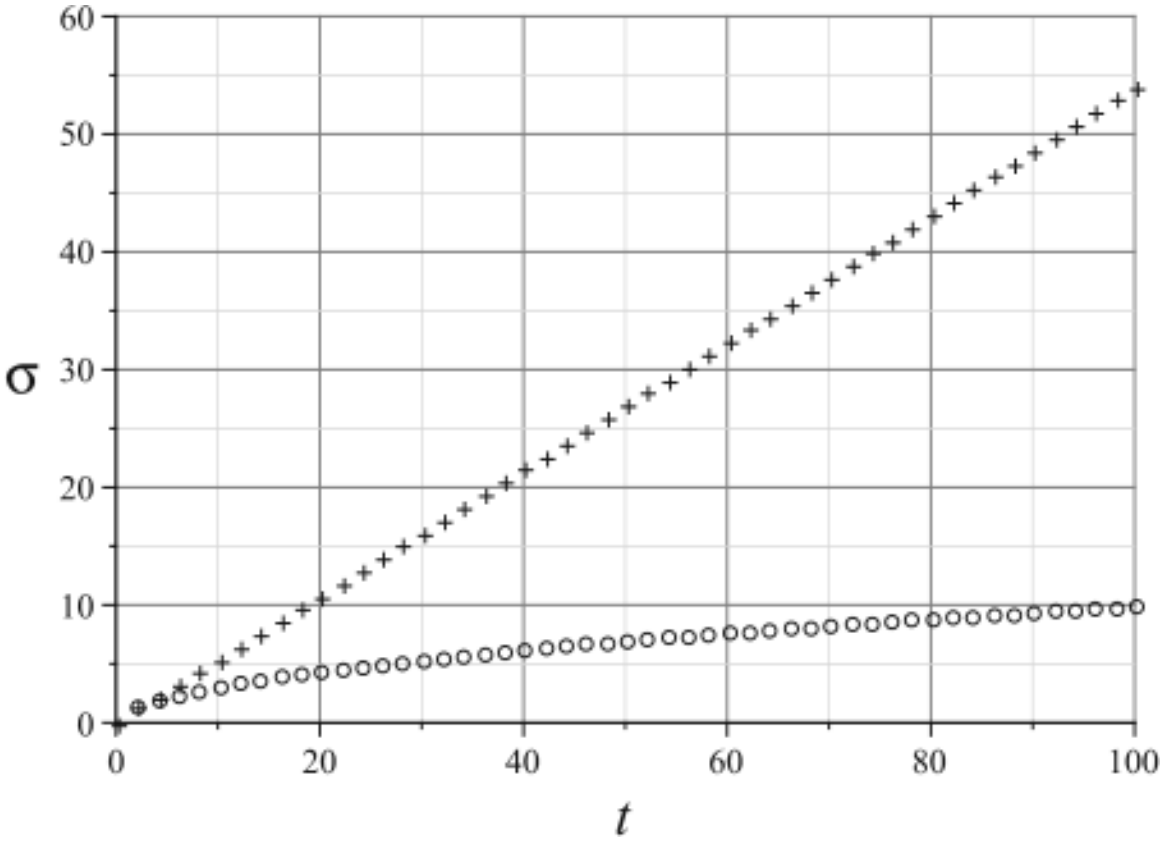


Figure 3.9: Standard Deviation with time for the quantum(*crosses*) and classical random(*circles*) walks

4 Search Algorithms

A common application of walks is in search problems. In search problems, we generally have a black box function

$$f(x) = \begin{cases} 0 & x \in G \\ 1 & x \in G^C \end{cases}$$

where $G \cup G^C = S$ which is the search space. We are interested in developing an algorithm to output some $x \in G$ by querying f.

Naively, one can run a stochastic process over the domain $G \cup G^C$ and observe the system until we see an element in G. The average time to succeed in this protocol is called the hitting time of G, and denoted as T^G. Naturally, we not only want high success rates, but we also want low mean hit times.

In the particular case of Markov chains, hitting times are a kind of stopping time [TODO: cite appendix], and are well studied in their own regard.

4.1 Formalism

Define the following

- Denote the readout at the n^{th} measurement (at $t = n\tau$) as X_n.
- Select a target node δ
- Probability of first hit in n steps $F_n = P(X_n = \delta | X_i \neq \delta \forall i \in [0, n-1])$
- Mean hit time $\langle t_S \rangle = \sum_{i=0}^{\infty} iF_i$
- Success probability in n steps $S_n = \sum_{i=1}^{n} F_i = P(\exists i \in [0, n] | X_i = \delta)$
- Survival probability = Failure probability $= \mathcal{S}_n = 1 - S_n$
- Asymptomatic versions of these terms are given by taking $n \to \infty$

4.1.1 Readout in Walks

To identify whether a walker has hit the target node, we need to track the location of the walker as it evolves in time.

For the classical case, this poses no problem, as measurement does not disturb the system. In the quantum case however, we need to be a bit more careful.

A constantly measured walker will freeze the dynamics of a quantum walker. This is known as the Quantum Zeno effect. The solution for this is to measure after every τ steps.

> **Reduction to classical random walk with $\tau = 1$**
>
> The quantum $\tau = 1$ case reduces the discrete time quantum walk to a classical random walk with $\tau = 1$. Effectively, we apply a dephasing operator on the density matrix, dropping all off diagonal terms. Thus, we lose all effects of superposition, causing the classical random walk.

Let us plot the readout trajectories of walkers with different paramters

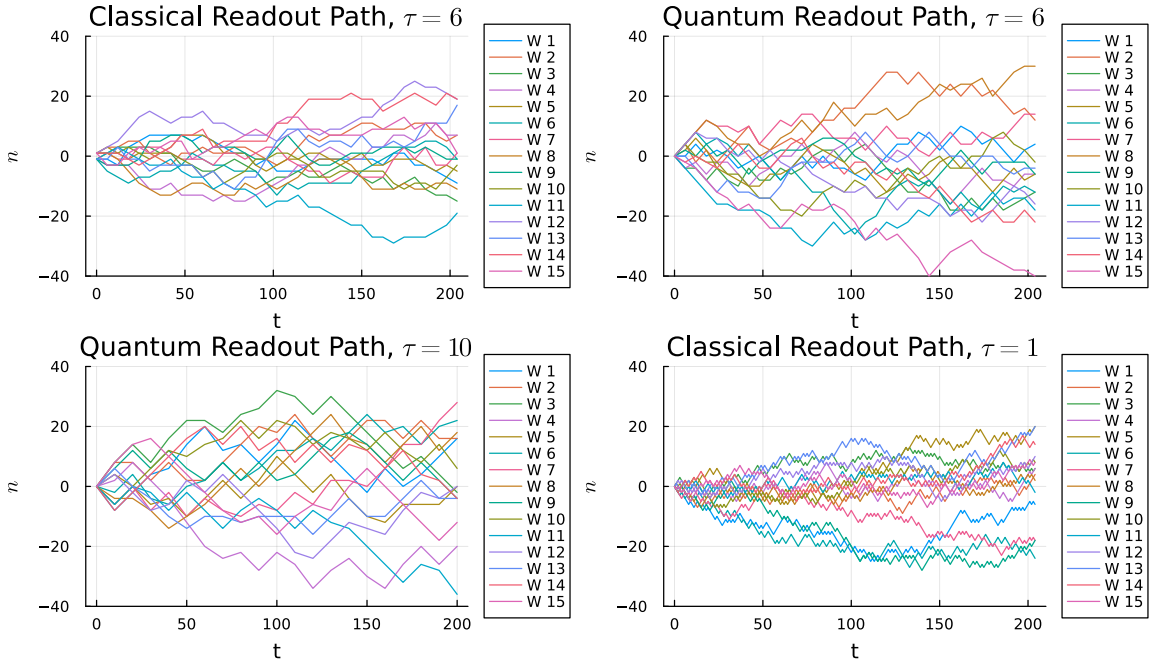


Figure 4.1: Trajectories of readouts in quantum and classical walks

Note the very clear difference in the spread between the classical readout and the quantum readout for same τ. Similarly note the spread between the quantum readouts for different τs.

4.2 Markov Chains and Walks

In the classical case, it is clear that the 1D SSRW is a markov process. In [TODO: Cite Appendix 1], we show that the quantum walk with measurement is also a markov process.

Consequently, we can use certain results from the theory of Markov processes to compare the two walks.

4.2.1 Irreducibilty and Recurrence

Under the usual definition of irreducibility[1],

> **Definition: Irreducibility**
>
> If $\forall i, j \in S, \exists n, m | P_{ij}^n > 0, P_{ji}^m > 0$, then the chain is called irreducible.

It is trivial that in the 1D chain, $n = m = |i - j|\tau$ satisfies the condition.

Following[1],

> **Definition: Recurrence**
>
> If $i \in S, \sum_{n=1}^{\infty} P_{ii}^n \to \text{inf}$, then the state is called recurrent. Equivalently, $\sum_n F_n \to 1$ for recurrent nodes. If a chain is irreducible and one of its states is recurrent, all its states are recurrent and thus the chain is called recurrent.

It is a well known fact that the 1D SSRW is recurrent. It is just as well known a fact that the 3D SSRW is transient[1]. In the quantum case, it is not as clear whether any of the available definitions of the quantum markov process is more or less better than the others. Thus, the transience of quantum walks depends on the exact definition we are working under[14]. In our definition however, it can be shown using symbolic computation[15] that the walk is indeed transient. This poses an interesting problem.

4.3 Survival probability

We can thus measure and plot the $S_n - n$ curve for the quantum and classical walks to compare the efficiency of these walks in the first hit problem.

In previous work[16], the analytical solution of the success rate as a function of time is found. An analytical result for discrete time walks is harder due to the nature of the walk. However we reproduce the analytical results for the discrete time case computationally. Figure 4.2a[16] shows the success rate for the continuous time quantum and classical walks, and Figure 4.2b shows our results for the discrete time case.

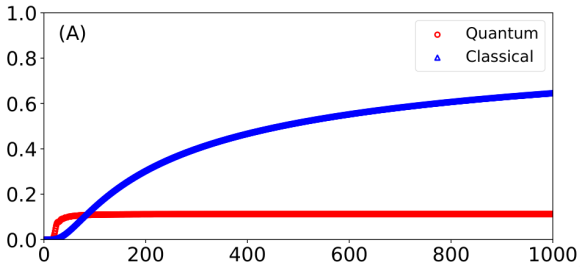
4.3.1 Observations

In the *continuous time* case:

- The quantum walk has a fast rise in the initial phase but saturates at ~0.1
- The classical walk has a slow rise, but eventually reaches 1

In the *discrete time* case, solved computationally:

- Both walks do eventually reach 1
- The quantum walk shows a saturation for a while before suddenly rising again



(a) Success probability vs Time for both walks[16]

(b) TODO

Figure 4.2: Success Probability

- The classical walk shows a slow rise in the beginning phases, in contrast to the sharper rise of the quantum walk.

The observations in the continuous time case can be explained by the recurrence of the walk. While the quantum walker is faster (See Chapter 3), the transient nature (See Chapter 4.2.1) of the walk leads to a non zero asymptomatic failure rate. Clearly this is a problem. What this means practically, is that for the first hit problem, if the quantum walk hits the target node, it does so faster than the classical walk, but a majority of the times, it doesn't hit the target node at all.

There seems to be an apparent difference between the discrete and the continuous time walks, but this is only an artifact of the finite size of the walk space. It is well known that any irreducible finite chain is recurrent[17]. This claim can be verified by simply increasing the size of the state space, and noting that the saturation phase in the quantum walk lasts longer.

5 Stochastic Resetting

The crux of the matter is this. The quantum walk is fast in the initial phase, but eventually saturates asymptotically to a success rate of < 1. This means that some of the walkers hit the target, and others do not. If that is the case, is it possible to restart the walk when it starts saturating? That is, can we restart the walk for the walkers that have not yet hit the target after $r\tau$ time?

For such dynamics, we will need to reformulate the walk slightly.

5.1 Formalism

A reset of the walker implies that the walker returns to its initial state, and the walk dynamics continue from there.

However, we can vary when we reset the walker by considering multiple reset processes. A common reset process is the Poissonian reset, where the reset times are modelled as a poissonian process with some parameter r. This is particularly convenient in the case of continuous time walks.

For the discrete walks, the geometric distribution is more natural.

$$t \sim \text{Geom}(\gamma)$$

that is, at each step, there is a γ probability of reset.

This results in different dynamics, which can be seen in subsequent subsections, but it has an equally drastic effect on the recurrence of the walk.

5.1.1 Recurrence and Resetting

In the geometric resetting case, it is clear that $P_{00}^n \geq \gamma$, and hence $\sum_n P_{00}^n \geq \sum_n \gamma$ which diverges as $n \to \infty$. Thus regardless of the initial walk, the final walk will definitely be recurrent. Thus the motivation for resetting the quantum walk which is transient should be immediately clear.

5.1.1.1 Stochastic Reset Classical Walk

In the classical case, the transition probabilities change to

$$p_{ij} = \begin{cases} (1-\gamma)\cdot 1/2 & |i-j| = 1 \\ \gamma & j = r_0 \\ 0 & \text{otherwise} \end{cases}$$

Thus, the new transition matrix is modelled as

```
γ = 0.2
U1 = sparse(SymTridiagonal(fill(0., 31), fill(0.5, 30)))
R = fill(0., (31, 31))
R[21,:] .= 1
U = sparse((1-γ) * U1 + γ * R);
```

where `U1` is the unchanged walk matrix and `R` is the reset matrix.

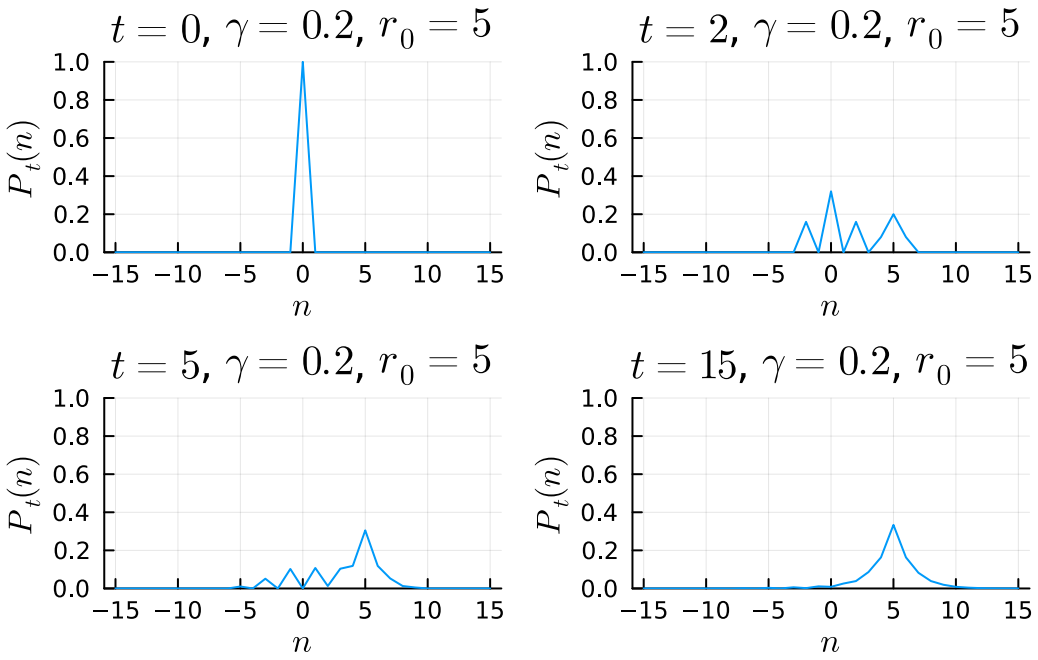


Figure 5.1: Stochastic Reset Classical Walk

5.1.1.2 Stochastic Reset Quantum Walk

In the quantum case, the evolution changes from unitary dynamics to a nonunitary CPTP map of the following form.

$$O_{SR}(\rho) = (1-\gamma)\left(U\rho U^\dagger\right) + \gamma|r_0\rangle\langle r_0|$$

where U is the walk unitary.

```
function swqw(n, γ)
    R = collect(Tridiagonal(fill(1., n), zeros(n+1), zeros(n)))
    R[1, end] = 1
    L = collect(Tridiagonal(zeros(n), zeros(n+1), fill(1., n)))
    L[end, 1] = 1
    U = KrausOperators(
        [sparse(proj(ket(1, 2)) ⊗ L + proj(ket(2, 2)) ⊗ R)]
    )
    init_coin = 1/√2 * (ket(1,2) - 1im * ket(2,2))
    H = KrausOperators([sparse(hadamard(2)⊗I(n+1))])
    init_state = proj(init_coin ⊗ ket(n÷2 + 1, n+1))
    ψ = [[init_state]; accumulate(1:40, init=init_state) do old, _
        (1-γ)*H(U(old)) + γ*proj(init_coin ⊗ ket(n÷2+6, n+1))
    end]
    map(enumerate(real.(diag.(ptrace.(
        ψ, Ref([2, n+1]), Ref([1])
    ))))) do (t, ps)
        Plots.plot(
            -n÷2:n÷2, ps,
            ylims=(0, 1), xlabel="\$i\$",
            ylabel="\$\\langle i|\\psi\\rangle\$",
            title="\$t=$(t), \\gamma=$(γ), r_0 = |5\\rangle \$",
            legend=false
        )
    end
end;
```

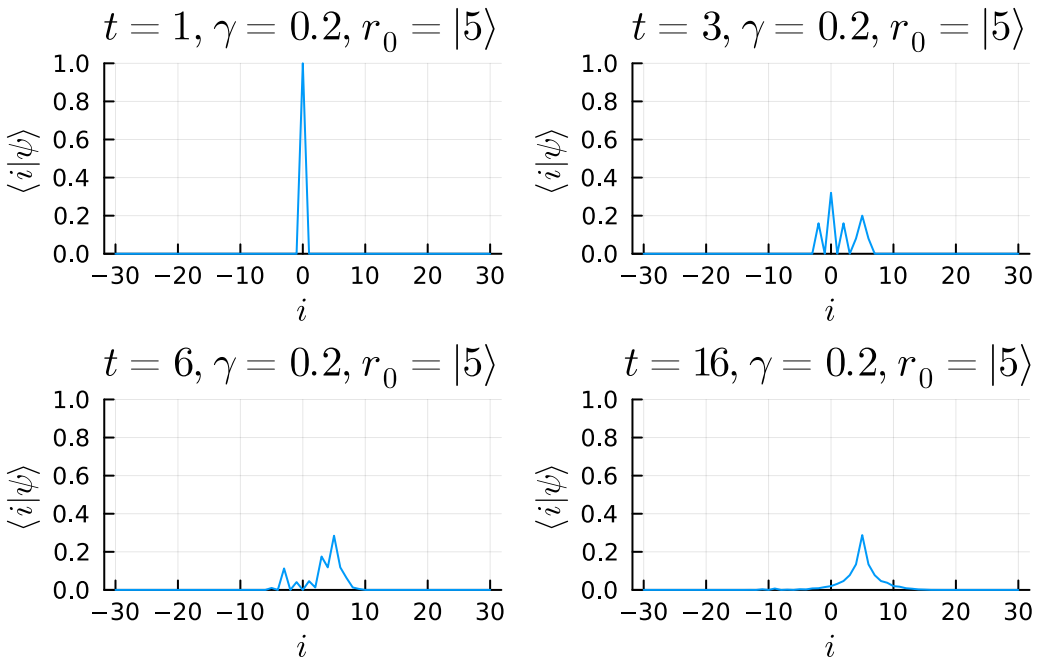


Figure 5.2: Stochastic Reset Quantum Walk

5.2 Effect of Resetting on First Hit problem

For this analysis, we will consider the simpler "sharp reset" formalism[16], where the reset time is sampled from a distribution

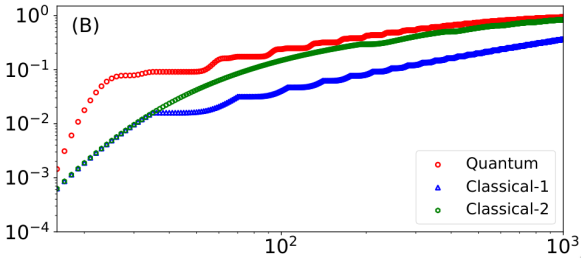
$$t_r \sim \delta(t - r\tau)$$

Thus we restart after r measurement events (at $t = r\tau$). Once we understand the effect of this kind of reset, gaining intuition for reset times sampled differently is easier.

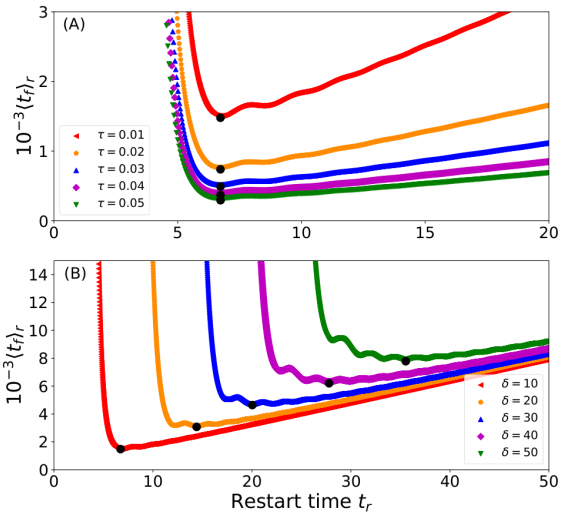
The success probability vs time can now be plotted (Figure 5.3a)[16] for the reset case.

Now we see that the success probability is drastically increased for both cases, but due to the ballistic nature of the quantum walk, we see that the reset quantum walk performs much better than the classical walk.

For a better understanding of the performance of the reset quantum walk with reference to changing reset rates (r) and measurement times (τ), we can plot (Figure 5.3b)[16] the mean first hitting time vs these parameters.



(a) Reset Success probability vs Time[16]



(b) Effect of r and τ on mean hitting time[16]

Figure 5.3: Sharp Reset Quantum Walk

5.2.1 Observations

- Deterministic restart leads to zero asymptomatic failure rate
- Eager restarting leads to walker never reaching δ, reducing success rates drastically
- Cautious restart reduces the effect of restart, reducing success rates.
- There exists an optimal r, but this needs to be optimized, which is non trivial for general τ and graph structures.

Can stochastic restarting be better than sharp reset?

No, because even if $\langle r \rangle = r_{\text{optimal}}$, $\langle t_f \rangle > \langle t_f \rangle_{\text{optimal}}$ due to the non-monotonic nature of the curve

5.3 Sharp Reset for Discrete time walks

For discrete time walks, the sharp reset walk is given by

$$|\psi_t\rangle = U^{t-r_l\tau}(|c_{\text{init}}\rangle \otimes |0\rangle)$$

where $r_l\tau$ was the time of last reset.

Equivalently, in the circuit model, the reset circuit is considered to be a measure followed by postprocess where we apply the appropriate unitary rotation to rotate to $|\psi_0\rangle$.

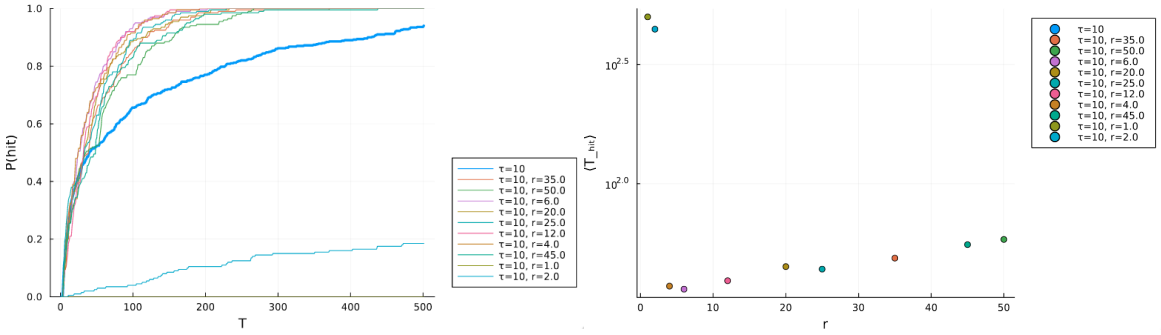


Figure 5.4: P(hit) vs Time

Figure 5.5: $\langle T_{hit} \rangle$ vs γ

Figure 5.6: Sharp Reset Quantum Walk

Our results are plotted in Figure 5.6. Note the similarity between the curves. However, also note the difference between the non reset curve, where in the discrete case, success probability still reaches 1, this can be attributed, as before, to the finiteness of the walk space.

5.4 The Problem in the Solution

As discussed before, reset rates which are very high or low can end up being detrimental to the success times of the walk. Secondly, there is no clear path as to how to optimize the reset parameter for arbitrary graph structures.

Just as we harnessed the power of quantum superposition to speed up the walk, can we similarly have a superposition between the reset and evolution?

6 Quantum Resetting by Superposition

From the previous discussion, it seems fruitful to consider a quantum reset of quantum walks. The concept is similar to that of the quantum walk vs classical walk: let the resetting occur on probability amplitudes rather than probabilities, allowing for useful interference.

For this, we suggest the following formalism

6.1 Formalism

On a finite 1D chain of length $2N+1$, define the following - - Reset operation - $\mathcal{R}$ by the Kraus operators $\{\mathcal{R}_i = |r_0\rangle\langle i|\}_{i\in[-N,N]}$ on $\mathcal{H}_W$ - Evolve operation - $\mathcal{U}$ by the unitary operation $S \circ H$ on $\mathcal{H}_C \otimes \mathcal{H}_W$ - Attach another two level coin - states denoted by $|0\rangle$ and $|1\rangle$. Resulting state lies in $\mathcal{H}_R \otimes \mathcal{H}_C \otimes \mathcal{H}_W$ - Controlled reset operation - $\mathcal{E}$ by the Kraus operators $\{\mathcal{E}_i = |0\rangle\langle 0| \otimes I_2 \otimes \mathcal{R}_i + \frac{1}{\sqrt{N}}|1\rangle\langle 1| \otimes \mathcal{U}\}_{i\in[-N,N]}$. See TODO:appendix for proof that this set of Krauss operators represents a CPTP map. - A reset coin operator $\Gamma(\gamma) = \begin{bmatrix} \sqrt{1-\gamma} & \sqrt{\gamma} \\ \sqrt{\gamma} & -\sqrt{1-\gamma} \end{bmatrix}$ on $\mathcal{H}_R$ - One step of the resulting walk is defined as $\mathcal{E} \circ (\Gamma \otimes I_2 \otimes I_{2N+1})$

The initial motivation for such a formalism comes from Anubhav's[18] master thesis, where a similar formalism was applied to a qubit system. The dominant motivation is the faster convergence in the quantum case, which was confirmed in the qubit case. However, the current problem we are considering has drastically changed the methods of exploration and the property we want to optimize.

6.1.1 Interpretation of γ and dependence on initial condition of the coin

In the stochastic resetting γ is understood as the "rate" of resetting. However, this is no longer accurate for the quantum case. Consider the case for $\gamma = 0$. Then, the Gammamard operator $\Gamma = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. This does not automatically imply that the reset never occurs; we also require that the initial state of the reset coin is $|0\rangle$. If the initial state of the reset coin is $|1\rangle$, this corresponds to the always reset case. Thus, γ is better understood as the probability that given the last step was a reset, what is the probabilty this step is an evolve, and vice versa. Thus, it is not immediately

obvious how we can compare the reset mechanisms for a given γ. Nor is it obvious how the initial state of the reset coin finally affects the success probability and such, and needs to be numerically checked.

6.1.2 Resetting of the walker coin

In our specific formalism (Section 6.1), we have only applied the reset operation on the walker $\mathcal{H}_W$. One could also reset the walker coin $\mathcal{H}_C$ and see what happens. In our current formalism, there is no entanglement broken between the two coins, but there is no apriori understanding of how this may (or may not) affect the walk.

6.2 Computational Implementation

Implementations of $\mathcal{U}, U, H$ follow as before. The $\mathcal{E}$ operation is implemented as

```
function €(n, r₀)
    ks = map(1:n) do i
        (1/√n * [1 0; 0 0] ⊗ (S(n)*H(n))) +
        [0 0; 0 1] ⊗ I(2) ⊗ real(ket(r₀,n)*bra(i,n))
    end
    return KrausOperators(sparse.(ks))
end;
```

The gammamard operation Γ is defined as

```
Γ(γ) = [
    √(1-γ)  √γ;
    √γ      -√(1-γ)
];
```

We take the initial reset coin to be $(S(-\pi/2) \circ \Gamma(\gamma))|0\rangle$ for the same reason as in the quantum walk

```
reset_init(γ) = [1 0; 0 -1im]*gammamard(γ)*ket(1, 2);
```

6.3 Results

6.3.1 What the walk looks like

For the specific choice of initial reset coin as $1/\sqrt{2}\,(|0\rangle - i|1\rangle)$ and $\gamma = 0.2$, the quantum reset walk looks like Figure 6.1.

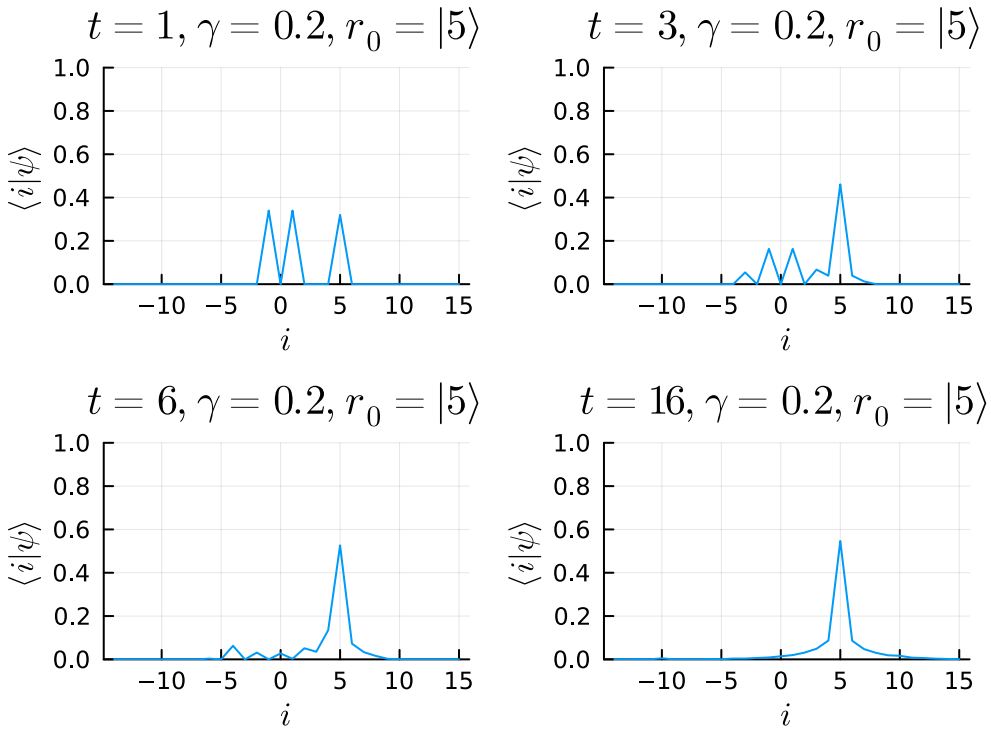


Figure 6.1: Quantum Reset Walk

6.4

7 Summary

In summary, this book has no content whatsoever.

References

[1] J.R. Norris, *Markov Chains*, 1st pbk. ed (Cambridge University Press, Cambridge, UK ; New York, 1998).
[2] P. Glasserman, *Monte Carlo Methods in Financial Engineering* (Springer, New York, 2004).
[3] K.P. Griffin, S.J. Suresh, T.J. Flint, and W.H.R. Chan, (2019).
[4] D. Psaltis, F. Özel, L. Medeiros, P. Christian, J. Kim, C. Chan, L.J. Conway, C.A. Raithel, D. Marrone, and T.R. Lauer, ApJ **928**, 55 (2022).
[5] N. Metropolis, (n.d.).
[6] L.S. de Souza, J.H.A. de Carvalho, and T.A.E. Ferreira, in *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)* (IEEE, Salvador, Brazil, 2019), pp. 836–841.
[7] M. Bae and W.O. Krawec, (2021).
[8] S. Mukherjee, IEEE Trans. Quantum Eng. **3**, 1 (2022).
[9] X. Bonnetain, A. Chailloux, A. Schrottenloher, and Y. Shen, (2022).
[10] M. Goldsmith, G. García-Pérez, J. Malmi, M.A.C. Rossi, H. Saarinen, and S. Maniscalco, (2022).
[11] S.A. Ortega and M.A. Martin-Delgado, (2022).
[12] Qiskit, Quantum Walk Search Algorithm (n.d.).
[13] R. Portugal, *Quantum Walks and Search Algorithms*, 2nd ed. 2018 (Springer International Publishing : Imprint: Springer, Cham, 2018).
[14] M. Štefaňák, I. Jex, and T. Kiss, Physical Review Letters **100**, 020501 (2008).
[15] H. Friedman, D.A. Kessler, and E. Barkai, Phys. Rev. E **95**, 032141 (2017).
[16] R. Yin and E. Barkai, (2022).
[17] user940, (n.d.).
[18] A. Srivastava, Resetting Quantum Systems Through Superposition of Evolution, PhD thesis, IISER Mohali, 2021.