QUANTUM WALKS

Daniel Reitzner

Department of Mathematics, Technische Universität München, 85748 Garching, Germany

Daniel Nagaj, Vladimír Bužek

Research Center for Quantum Information, Institute of Physics, Slovak Academy of Sciences, Dúbravská cesta 9, 845 11 Bratislava

This tutorial article showcases the many varieties and uses of *quantum walks*. Discrete time quantum walks are introduced as counterparts of classical random walks. The emphasis is on the connections and differences between the two types of processes (with rather different underlying dynamics) for producing random distributions. We discuss algorithmic applications for graph-searching and compare the two approaches. Next, we look at quantization of Markov chains and show how it can lead to speedups for sampling schemes. Finally, we turn to continuous time quantum walks and their applications, which provide interesting (even exponential) speedups over classical approaches.

> *. . . I walk the line.*
> Johnny Cash

Contents

1   Introduction

For a physicist, a *quantum walk* means the dynamics of an excitation in a quantum-mechanical spin system described by the tight-binding (or other similar model), combined with a measurement in a position basis. This procedure produces a random location, with a prescription for computing the probability of finding the excitation at a given place given by the rules of quantum mechanics.

For a computer scientist or mathematician, a *quantum walk* is an analogy of a classical random walk, where instead of transforming probabilities by a stochastic matrix, we transform probability amplitudes by unitary transformations, which brings interesting interference effects into play.

However, there's more to quantum walks, and the physics and computer science worlds have been both enriched by them. Motivated by classical random walks and algorithms based on them, we are compelled to look for quantum algorithms based not on classical Markov Chains but on quantum walks instead. Sometimes, this "quantization" is straightforward, but more often we can utilize the additional features of unitary transformations of amplitudes compared to the transfer-matrix-like evolution of probabilities for algorithmic purposes. This approach is amazingly fruitful, as it continues to produce successful quantum algorithms since its invention. On the other hand, quantum walks have motivated some interesting results in physics and brought forth several interesting experiments involving the dynamics of excitations or the behavior of photons in waveguides.

Our article starts with a review of classical random-walk based methods. The second step is the search for an analogy in the quantum world, with unitary steps transforming amplitudes, arriving at a new model of discrete-time quantum walks. These return back to classical random processes when noise is added to the quantum dynamics, resulting in a loss of coherence. On the other hand, when superpositions come into play, discrete-time quantum walks have strong applications in graph searching and other problems. Next, we will look at how general Markov chains can be quantized, and utilized in physically motivated sampling algorithms. Finally, we will investigate our original motivation – analyzing the dynamics of an excitation in a spin system, and realize that this is also a quantum walk, in continuous-time. We will investigate its basic properties, natural and powerful algorithms (e.g. graph search and traversal), and computational power (universality for quantum computation). Finally, we will conclude with a view of the analogy and relationship to discrete-time quantum walks.

The paper is writen as a tutorial with the aim of clarifying basic notions and methods for newcomers to the topic. We also include a rich variety of references suitable for more enthusiastic readers and experts in the field. We know that such a reference list can never be truly complete, just as this work can not contain all the things we had on our minds. We had to choose to stop writing, or the work would have remained unpublished forever. As it stands, we believe it contains all the basic information for the start of your journey with quantum walks.

Let's walk!

2 Classical Random Walks

To amuse students, or to catch their attention, it is quite usual to start statistical mechanics university courses with the *drunkard's walk* example. We are told the story of a drunken sailor who gets so drunk that the probability for him to move forward on his way home is equal to the probability to move backward to the tavern where he likes to spend most of his time when he is not at sea. Looking at him at some point between the tavern and his home and letting him make a number of drunken steps, we ask where will we find him with the highest probability?

This seemingly trivial problem is very important. In fact, it can be retold in many various ways. For example the *Galton's board* [1] device, also known as the *quincunx* (see Fig. 2.1), has the same properties as the drunkard's walk. Balls are dropped from the top orifice and as they move downward, they bounce and scatter off the pins. Finally, they gather and rest in the boxes on the bottom. The number of balls in a box some distance from the horizontal position of the orifice can be mapped (is proportional) to the probability for the drunken sailor to be found at a particular distance from the tavern (after a specific number of steps corresponding to the vertical dimension of the quincunx). This is due to the fact, that the ball on its way downward scatters from the pins with approximately the same probability to the left as to the right — it moves in the same manner as the sailor on his way from the tavern. Another retelling of the sailor's story is the magnetization of an ideal paramagnetic gas[1] of spin-1/2 particles. Tossing a coin is yet another example of drunkard's walk.
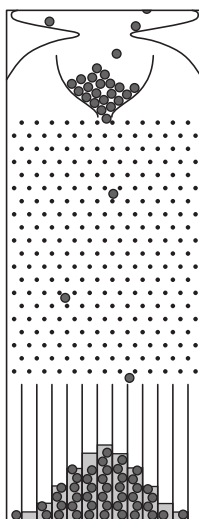
Fig. 2.1. A possible realization of the drunkard's walk – the quincunx (Galton's board). The number of marbles in a bin at the bottom correspond to the probability of the sailor to be at that position after taking 19 random steps (the number of pins a falling marble encounters on its way down).

---

[1]It is an ensamble of magnetic particles (in this case without any external field), that do not interact — they do not feel each other.

This simple *walk on a line* is not just an example of some simple problems. It serves as a toy model and starting point for more involved variations. These modifications of the drunkard's walk spread across many branches of science and their position is justified by the results they provide. The roots of random walks, however, lie in the areas of mathematics, economy and physics. Probably the earliest (1900) attempt to describe random walks originates in the doctoral thesis of Bachelier [2], who considered the evolution of market prices as a succession of random changes. His thesis, however, was not very rigorous and random walks had to wait to be defined properly from a mathematical viewpoint. This was done by Wiener [3] in 1923. In physics it was Einstein [4], who in his theoretical paper on Brownian motion from 1905 put the first theoretical description of a random walk in natural sciences with explanation based on the kinetic theory, where gases (and liquids) are considered to be composed of many small colliding particles — it was aimed to explain thermodynamical phenomena as statistical consequences of such movement of particles. Einstein speculated, that if (as he believed) the kinetic theory was right, then the seemingly random Brownian motion [5] of large particles would be the result of the action of myriads of small solvent particles. Based on kinetic theory, he was able to describe the properties of the motion of the solute particles — i.e. connect the osmotic pressure of these particles with their density — and show the connection with the diffusive process. Assuming the densities of solvent and solution are known, an experiment based on the barometric formula stating how the pressure changes with altitude provides us with the kinetic energy of solute particles at a given temperature. Hence, the Avogadro number can be estimated. This, indeed, has been done [6] and strengthened the position of the kinetic theory at that time.

2.1   Markov Chains

Random walks have been formulated in many different ways. Generally, we say that a *random walk* is a succession of moves, generated in a stochastic (random) manner, inside a given state space Ω that generally has a more complex structure than a previously mentioned liear chain (drunkard's walk). If the stochastic moves are correlated in time, we talk about non-Markovian walks (walks with memory), however, for our purposes we will always assume the walks to be Markovian – the random moves of the walker are uncorrelated in time. Note, though, that the stochastic generator may be position dependent. The sequences of such moves result in the so-called *Markov Chains*.

Taking only one instance (path) of a random walk is, naturally, not enough to devise the general properties of the phenomenon of random walks. That is why we consider the evolution of distributions on state space Ω. These distributions are the averages over many instances of a random walk of a single walker (or an ensemble of walkers). In such case, for a Markovian walk in a countable space[2] Ω, we define a distribution

$$p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \end{bmatrix}$$

---

[2]The topic of uncountable spaces is not important for the scope of this work.

over indexed states, with p_k corresponding to the probability of finding the walker at the position with index k. For p to define a probability, it is necessary that

$$\sum_k p_k = 1.$$

In this framework, the stochastic generator that describes the single-step evolution of a distribution function is given as a stochastic matrix[3] M, giving the one-step evolution of a distribution p into p' as

$$p' = Mp.$$

Having the matrix M and an initial state $p(0)$, the distribution after m steps is described by the formula

$$p(m) = M^m p(0). \tag{2.1}$$

The allowed transitions (steps) of the random walker from position i to position j are reflected within M by the condition $M_{ji} \neq 0$. For now, we also assume (although it is not necessary) a weak form of symmetry in the transitions by requiring that M_{ij} is also not zero. Forbidden transitions correspond to the condition $M_{ji} = M_{ij} = 0$. The allowed transitions reflected in the non-zero elements of M induce a graph structure $\mathcal{G} = (V, E)$ on Ω. Here $V = \Omega$ is a set of vertices (corresponding to states) and $E = \{(j, i) : j, i \in \Omega, M_{ji} \neq 0\}$ is a set of ordered pairs of vertices (oriented edges, corresponding to transitions) – the allowed connections between vertices. Conversely, we may say that the graph structure defines allowed transitions. For simplicity, from now on we will use the notation ij instead of (i, j) for the oriented edges.

The graph structure is not the only thing reflected in M. The coefficients of M also define the transitional probabilities to different states and in this manner reproduce the behaviour of the stochastic generator, usually called a *coin*. For an *unbiased coin,* which treats all directions equally, the coefficients of M are defined as

$$M_{ji} = \begin{cases} \frac{1}{d(i)} & \text{for } (i, j) \in E, \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

with $d(i)$ being the degree of the vertex i. If this is not true, we say that the coin is *biased*, preferring some directions above others.

2.2   Properties of Random Walks

When studying properties of random walks we search for specific properties that (potentially) help us solve various posed problems. Sometimes we want to know where the walker is after some time, in other cases we may want to know how much time does the walker take to arrive at some position or what is the probability of the walker to get there in given time. Another question asks how much time we need to give the walker so that finding him in any position has

---

[3]A *stochastic matrix* is a real matrix with columns summing to 1, which preserves the probability measure, i.e. all its elements are positive and smaller than one.

approximately the same probability. All these questions are not only interesting, but also useful for the construction (and understanding) of randomized algorithms.

If we are concerned with an unbiased random walk in one dimension (the so called drunkard's walk introduced in the previous Chapter), according to the central limit theorem, the distribution very quickly converges to a Gaussian one. The standard deviation of the position of the walker becomes proportional to $\sqrt{m}$, where m is the number of steps taken. Such an evolution describes a diffusive process.

All the routes to get from position 0 to position x (positive or negative) after performing m steps are composed of $n_+ = (m + x)/2$ steps upward and $n_- = (m - x)/2$ steps downward. This is under supposition of what we shall call *modularity* – after an even (odd) number of steps, the probability to find the walker on odd (even) positions is always zero, constraining $m + x$ to be even (this is consequence of bipartite character of the graph). The number of these paths characterized by specific $n_\pm$ is

$$N(x, m) = \binom{m}{n_+} = \frac{m!}{n_+! n_-!}.$$

The number of all possible paths is $N(m) = 2^m$ and, hence, the probability to find the walker at position x is $P_{\text{rw}}(x, m) = N(x, m)/N(m)$. Using Stirling's approximation

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n,$$

we can express the probability of finding the walker at x after m steps as

$$
\begin{aligned}
P_{\text{rw}}(x, m) &= \frac{m!}{2^m \, n_+! n_-!} \\
&\simeq \sqrt{\frac{2}{m\pi}} \left[\left(1 + \frac{x}{m}\right)\left(1 - \frac{x}{m}\right)\right]^{-\frac{m}{2}} \left(1 + \frac{x}{m}\right)^{-\frac{x-1}{2}} \left(1 - \frac{x}{m}\right)^{\frac{x-1}{2}} \\
&\simeq \sqrt{\frac{2}{m\pi}} \left[1 - \frac{x^2}{m^2}\right]^{-\frac{m}{2}},
\end{aligned}
$$

where we assumed $m \gg x$, making the last two terms in the second line approximately one. In the limit of large m and small x, this approximation can be further transformed into the formula[4]

$$P_{\text{rw}}(x, m) \simeq \frac{2}{\sqrt{2\pi m}} \exp\left(-\frac{x^2}{2m}\right).$$

The function P_{rw} is not yet a probability distribution as it is normalized to 2, but we have to recall the modularity of the walk telling us, that after even number of steps, the walker cannot be on odd position and after odd number of steps, the walker cannot be on even position; this leads us to a probability distribution having form

$$p_{\text{rw}}(x, m) = \frac{1 + (-1)^{m-x}}{\sqrt{2\pi m}} \exp\left(-\frac{x^2}{2m}\right). \tag{2.3}$$

---

[4]Employ $\lim_{x \to \infty} \left(1 - \frac{1}{x}\right)^x = e$.

Finally, we observe

$$\langle x^2 \rangle \equiv \sum_{x=-\infty}^{\infty} x^2 p_{\mathrm{rw}}(x,m) \simeq \frac{1}{2} \int_{-\infty}^{\infty} x^2 P_{\mathrm{rw}}(x,m)\mathrm{d}x = m, \qquad \sqrt{\langle x^2 \rangle} = \sqrt{m}, \quad (2.4)$$

where we approximated the modular probability distribution by a smooth one, $P_{\mathrm{rw}}/2$.

The standard deviation of a walker is thus proportional to $\sqrt{m}$, and this fact may be deduced even without the knowledge of the limiting distribution (see e.g. Ref. [7]). The position of the walker x is in fact a sum of m independent variables (steps up and down) $y_1, \ldots, y_m$:

$$x = \sum_{j=1}^{m} y_j.$$

The square of the standard deviation then reads

$$\sigma^2(m) = \langle |x - \langle x \rangle|^2 \rangle = \langle x^2 \rangle - \langle x \rangle^2 = \left\langle \sum_{j=1}^{m} y_j \sum_{l=1}^{m} y_l \right\rangle - \left\langle \sum_{j=1}^{m} y_j \right\rangle^2.$$

As the y_j's are independent, the sums and averages can be exchanged, giving us

$$\sigma^2(m) = \sum_{j=1}^{m} \sum_{l=1}^{m} \left(\langle y_j y_l \rangle - \langle y_j \rangle \langle y_l \rangle \right).$$

For $j \neq l$, the independence of the random variables y_j is reflected in $\langle y_j y_l \rangle = \langle y_j \rangle \langle y_l \rangle$, so the square of the standard deviation simplifies to

$$\sigma^2(m) = \sum_{j=1}^{m} \left(\langle y_j^2 \rangle - \langle y_j \rangle^2 \right) = \sum_{j=1}^{m} \sigma_j^2 = m\sigma^2,$$

where σ_j are standard deviations of the random variables y_j. In this case, they are are all the same and equal to $\sigma_j \equiv \sigma = 1$. Thus we finally arrive at $\sigma(m) = \sqrt{m}$ by a different route.

There are also other simple observations we can make. Let us list them here, with the aim of later comparing them to the properties of the distributions arising from quantum walks.

Reaching an absorbing boundary. Let us look at the probability with which the walker returns into its starting position. After leaving this position, the walker makes a move forward. Now there is probability p_{10} for him to get back to the original position by taking all the possible routes into consideration. Under closer inspection, we find that this probability consists of the probability for him to make one step backward, which is $1/2$ and the probability for him to get back to the initial position only after moving further away from it first. This latter probability equals $\frac{1}{2}p_{10}^2$, since he moves forward with probability $1/2$ and then has to move two steps back with equal probabilities p_{10} (we suppose, that no matter how far from the initial position the walker is, he always has equal probabilities to move forward and backward). To sum up, we see that

$$p_{10} = \frac{1}{2} + \frac{1}{2}p_{10}^2.$$

The only solution to this quadratic equation is $p_{10} = 1$. Let us interpret this result: the probability for the walker to return to his initial position is equal to one, i.e. he always gets back. In other words, if we employ an *absorbing boundary* at position 0 and start at position 1, the walker will eventually hit the boundary.

   This result can be extended to the statement that the probability of reaching vertex i from any other vertex j is one. As the walk is translationally symmetric, the probability to get from any x to $x \pm 1$ at any time is always the same, $p_{10} = 1$. Thus, the probability to get from j to i is

$$p_{ji} = p_{10}^{|j-i|} = 1. \tag{2.5}$$

This is quite different from quantum walks, where the probability to hit some boundary is not one, as shown in Sec. 3.5.3.

Exercise 1 What is the probability p_{10} in general case, when the probability to move right is p and the probability to move left is $1 - p$ in each step?

Limiting distribution.   Now let us step away from the example of a walk on a linear chain and look at the larger picture, considering general graphs. For these, two quantities describing the properties of random walks are widely used in the literature — the *hitting time* and the *mixing time*. But first, let us have a closer look at walks with unbiased coins on finite graphs. It is interesting to find, that all such graphs (if connected and non-bipartite) converge to its stationary distribution $\pi = (d(1), d(2), \ldots, d(N))/2|E|$, where $d(j)$ stands for the degree of vertex j. It is easy to see that this is a stationary distribution. For any vertex j we have

$$(M\pi)(j) = \sum_{i:\ ji \in E} M_{ji} \pi(i) = \sum_{i:\ ji \in E} \frac{1}{d(i)} \frac{d(i)}{2|E|} = \frac{d(j)}{2m} = \pi(j).$$

This distribution is clearly uniform for regular graphs. In Appendix A we also provide a proof that for connected and non-bipartite graphs, this distribution is unique and also is the limiting distribution. With this notion we are ready to proceed to define the quantities of hitting and mixing time.

Hitting time.   It is the average time (number of steps) that the walker needs to reach a given position j, when it starts from a particular vertex (c stands for *classical*)

$$h^c(j) = \sum_{m=0}^{\infty} m\, p(j, m). \tag{2.6}$$

This quantity also makes sense for infinite graphs. We will later see (in Sec. 3.5.2) that for quantum walks this quantity is defined differently, due to the fact that measurement destroys the quantum characteristics of the walk.

Example 1 We have seen, that when the walker starts at position 1 she will eventually reach position 0. Let us evaluate now the hitting time between these two positions. As the number of

paths of length $2k$ which do not get to position 0 is determined by the Catalan number C_k (see Appendix C), the probability for the walker to hit vertex 0 after $2k+1$ steps is

$$p_{2k+1} \geq \frac{C_k}{2^{2k+1}},$$

where the inequality sign comes from the fact that we counted all 2^{2k} paths, even those crossing and/or hitting 0. When we employ Eq. (2.6) and Eq. (C.5) we find, that

$$h^c(1 \to 0) \geq \sum_{k=0}^{\infty} \frac{(2k+1)C_k}{2^{2k+1}} \geq \frac{\sqrt{\pi}}{e^2} \sum_{k=0}^{\infty} \frac{1}{\sqrt{2k+1}},$$

which diverges and so the hitting time is infinite, although the probability of eventually reaching vertex 0 is

$$p_{10} = \frac{1}{2} \sum_{n=0}^{\infty} \frac{C_n}{2^{2n}} = \frac{1}{2} c(1/4) = 1,$$

where $c(x)$ is the generating function for Catalan numbers.

Exercise 2 Consider now, that the probability to jump from position 1 to 0 is $p > 0$. Starting at position 1 we wait for one step and look whether the walker hit the boundary. If not we again set the walker to position 1 and repeat procedure again and again until we hit the boundary. Show that in such experiment the hitting time is

$$h^c(1 \to 0) = \frac{1}{p}. \tag{2.7}$$

In this case the hitting time is finite. We could as well let the walker go for some time T in which case the probability of hitting the boundary within this time would be some other p but the hitting time would be expressed in the same way as in Eq. (2.7). This definition leads to the emergence of tuples (T, p) with the hitting time given by Eq. (2.6) being just a special case basically when $p < 1$ for $T \to \infty$. Such definition seems to coincide more with the notion of absorbing boundary — the probability of absorption is the smallest p for which (∞, p) describes correct analouge of the hitting time given by Eq. (2.7). See also definition of hitting time in the quantum case in Sec. 3.5.2.

Mixing time. The second important quantity is the classical *mixing time* $\mathcal{M}_\epsilon^c$. As we have seen each unbiased random walk (on connected non-bipartite graphs) reaches a stationary distribution, which we denote π. The mixing time is then the time (number of steps) after which the probability distribution remains ϵ-close to the stationary distribution π, i.e.

$$\mathcal{M}_\epsilon^c = \min\{T : \ \forall t \geq T, |p(\cdot, t) - \pi(\cdot)|_{tvd} \leq \epsilon\}, \tag{2.8}$$

where $p(\cdot, t)$ is the distribution in time t and $|\cdot|_{tvd}$ is (total variational) distance of distributions,

$$|p(\cdot, t) - \pi(\cdot)|_{tvd} \equiv \sum_j |p(j, t) - \pi(j)|. \tag{2.9}$$

Again, we will later see in Sec. 3.5.1 that the mixing time is defined differently for quantum walks, as they are unitary and converge to a stationary distribution only in a time-averaged sense.

In classical case mixing time depends on the gap between the first two eigenvalues $\lambda_1 = 1$ (for the stationary distribution π) and λ_2. The use is illustrated by the next theorem.

Theorem 1 (Spectral gap and mixing time)

$$\frac{\lambda_2}{(1 - \lambda_2)\log 2\epsilon} \leq \mathcal{M}_\epsilon^c.$$

For closer details see e.g. Ref. [8].

2.3   Classical Random Walk Algorithms

Random walks are are a powerful computational tool, used in solving optimization problems (e.g. finding spanning trees, shortest paths or minimal-cuts through graphs), geometrical tasks (e.g. finding convex hulls of a set of points) or approximate counting (e.g. sampling-based volume estimation) [9]. In the previous Sections we had an opportunity to get acquainted with several interesting properties of random walks. These are often exploited when constructing new algorithms. For example, small mixing times can lead to more efficient and accurate sampling, while short hitting times can lead to fast search algorithms.

At present, there is a wide range of algorithms that use these properties of random walks to their advantage. These random-walk based algorithms range from searches on graphs, through solving specific mathematical problems such as satisfiability, to physically motivated simulated annealing that searches for "optimal" states (ground states, or states that represent minima of complex fitness functions). A large group of algorithms, that (not historically, though) can be considered to be based on random walks are Markov Chain Monte Carlo methods for sampling from low-temperature probability distributions and using them for approximate counting or optimization.

A huge research effort is devoted today to this vast area of expertise. We will briefly introduce a few of these interesting algorithms in the following pages. Even though the connection to random walks is often not emphasized in the literature, all these algorithms can be viewed from the vantage point of random walks.

2.3.1   Graph Searching

Random walks can be used to search for a marked item (or M items) on a graph of size N. Knowing the structure of the graph can sometimes allow us to find efficient deterministic solutions. However, sometimes a random walk approach can become useful. One such example is a search on a complete graph with loops. This task is just a rephrasing of a *blind search* without structure when any element might be the targeted one. One can easily show that on average (even if one remembers vertices that are not targets) one needs $O(N/M)$ queries to the oracle[5]. Later we will see that in the quantum case, a clever utilization of an oracle in quantum walks can produce a quadratic speedup due to faster mixing.

---

[5]In this case the oracle just gives the answer whether the vertex we picked is marked. We will deal with oracles a bit later in Sec. 4.1.1.
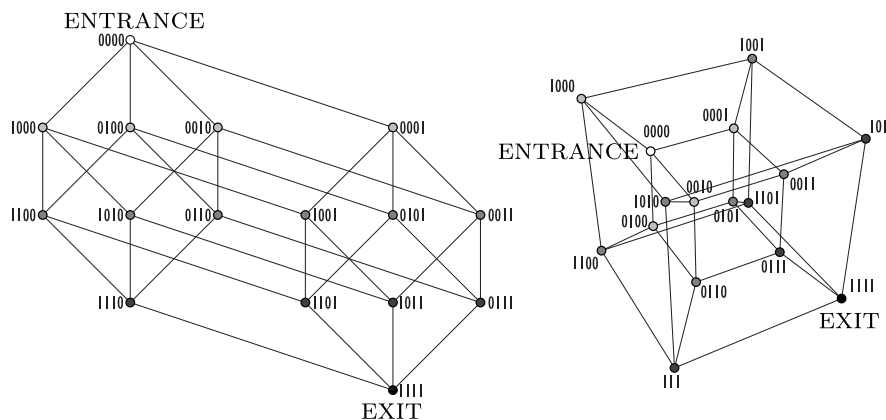
Fig. 2.2. Two graphical representations of a 4-dimensional hypercube with vertices labeled by 4-bit strings, connected when these strings. Connections exist only between vertices whose binary labels differ by a single digit. Vertices with the same number of zeros in their label create a layer of the hypercube. There are $n + 1 = 5$ layers in the 4-cube — layers 0 and 4 are denoted by black circles, layer 1 by red circles, layer 2 by green circles and layer 3 by blue circles. A quantum walker can traverse from the ENTRANCE vertex to the EXIT vertex in $O(n)$ steps (scaling linearly with the number of layers). A classical random walk is exponentially slower. Note though, that a different efficient classical algorithm (using memory to help the traversal) exists for this problem.

Two interesting examples of graph-search are hypercube traversal (Fig. 2.2) and glued trees traversal (Fig. 2.3), in both of which the goal is to find a vertex directly "opposite" to the starting one. We are given a description of the graph (with the promised structure) as an oracle that tells us the "names" of the neighbors of a given vertex. The goal is to find the "name" of the desired opposite vertex. In these examples, employing a quantum walk results in tantalizing exponentially faster hitting times than when using classical random walks. Yet, as we will see in the following examples, this does not mean that there are no deterministic algorithms that can do it as fast as the quantum walk one. To see an actual exponential speedup by a quantum walk over any possible classical algorithm, we will have to wait until Section 6.1.2.

Example 2 (Traversing a hypercube) An n-dimensional *hypercube* is a graph with vertices that are binary strings of length n (see Fig. 2.2). For every pair of vertices a and b we can define *Hamming distance,* which is the number of positions in which the two binary representations of a and b differ. Clearly this is the smallest number of steps one has to make in order to get from the vertex a of the hypercube to the vertex b as two vertices are connected by an edge if and only if the two strings corresponding to the vertices differ only by a single bit (thus have Hamming distance 1). In the hypercube, we can further distinguish a layered structure. Let us label two opposite vertices of the hypercube ENTRANCE and EXIT. We say, that vertex a is in the layer k, if its Hamming distance from the ENTRANCE vertex is k. We can see, that any k-th level vertex has neighbours only in levels $k - 1$ or $k + 1$. Also note, that ENTRANCE vertex has Hamming distance 0 and EXIT vertex has Hamming distance n.

Starting at the ENTRANCE vertex, our aim is to get to the EXIT as fast as possible. As between
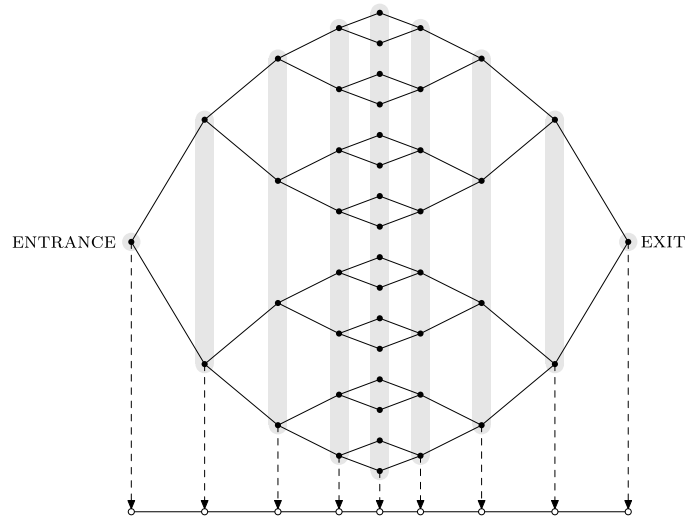
Fig. 2.3. A graph made from two glued trees graph can be efficiently traversed by a quantum walk (or by a clever classical algorithm). However, this graph is not (efficiently) penetrable by a classical random walk. The efficiency of the quantum walk algorithm comes from the graph symmetry – the walk on this graph is equivalent to a walk on a line of "column" states.

ENTRANCE and EXIT vertices there is rougly 2^n other vertices, usual random walk approach would find EXIT only in time exponential in n — the hitting time is $O(2^n)$ and as a result mixing time is even longer.

We can however radically shorten the time needed when we employ the walker with memory helping the walker to advance through the hypercube and increase her Hamming distance from the ENTRANCE in each step — see algorithm in Fig. 2.4. Let us label vertices the walker visits as a_k, with index k denoting the step (and layer as well). From the ENTRANCE vertex, denoted as a_0, the walker moves to random neighbour a_1, remembering the vertex it came from in memory

---

Set $k = 1$, the initial position a_1 to a random neighbor of the ENTRANCE vertex and $S_0 = \{a_0\}$ then for $n - 1$ times repeat:

1. set $k := k + 1$

2. randomly choose a_k to be neighbor of a_{k-1} such that $a_k \notin S_{k-2}$

3. set S_{k-1} to be the set of all neighbors of a_k that have a neighbor also in S_{k-2}

The resulting vertex is EXIT.

Fig. 2.4. Algorithm for traversing the n-hypercube. Starting at position ENTRANCE the walker uses structural oracle to navigate through the graph. She has to remember only $O(n)$ vertices in each step which help her propagate thorugh the graph increasing the layer in each step.

$S_0 = \{a_0\}$. The memory will contain all the vertices that are from lower layer, than actual vertex. Clearly, for S_0 this is true.

On each step $k + 1$ the walker chooses new neighbour a_{k+1} of actual vertex a_k from the next $(k+1)$-th layer by excluding the vertices from memory S_{k-1} from the selection process, as these are from layer $k - 1$. New memory S_k is constructed as a set of all neighbouring vertices of a_{k+1} that are also neighbours of some of the vertices in S_{k-1}.

To see that it works as intended, let us say, that a_{k+1} was obtained from a_k by flipping bit at position q. From definition of S_{k-1} we know, that $a_k \notin S_{k-1}$ and so all vertices contained in this set differ from a_{k+1} in two bits, q and some $r \neq q$. Now, to get from a_{k+1} to lower level, we would have to go either to a_k, which is a neighbour of S_{k-1}, or we would flip the bit on position r taking it to one-bit-flip (q) from some vertex from S_{k-1}.

It is easy to see now, that S_k, as a memory, is a set of all neighbours of vertex a_{k+1} that are at level k. In this manner the walker in each step increases its distance from the ENTRANCE vertex and decreases its distance from the EXIT vertex, thus needing n steps to traverse the graph with $O(n)$ searches in neighbourhood every step. Totally the efficiency of the algorithm is polynomial in n with memory of $O(n)$.

Example 3 (Traversing glued trees) Another example, where a usual random walker without memory fails is in traversing a graph made from two glued binary n-leveled trees, depicted in Fig. 2.3. As in the previous example, the random walker gets stuck in exponentially many vertices, that comprise the "body" of the graph and she will not be able to reach EXIT vertex efficiently. However, with memory the walker can exploit the difference between the central vertices and the rest and proceed through the graph as follows (see also Fig. 2.5). Starting from the ENTRANCE vertex, by simply randomly choosing neighbours other than the previous one, it reaches one of central "glued" vertices in time n. These vertices are easily recognised since they

---

Set $a_1 =$ ENTRANCE and for $n - 1$ times repeat:

  1. set $k := k + 1$

  2. randomly choose a_k to be neighbor of a_{k-1} such that $a_k \neq a_{k-2}$

Set $k = 0$ and repeat until $a_{k+n} =$ EXIT:

  1. set $k := k + 1$

  2. randomly choose a_{k+n} to be neighbor of a_{k+n-1} such that $a_{k+n} \neq a_{k+n-2}$

  3. if vertex a_k has only two neighbors return to $a_{k/2+n}$ (by setting $a_{k+n} := a_{k/2+n}$ and $k := k/2$)

Fig. 2.5. Algorithm for traversing glued trees. The walker can easily navigate with the help of structural oracle from the ENTRANCE vertex to the glued vertices. There she knows for certain her position as these vertices differ from the rest in the number of neighbors being only two. After she passes glued vertices, she can navigate in $O(n^2)$ steps to the EXIT vertex, as everytime she encounters glued vertex she knows, that halfway between previous encounter of the glued vertex she did a wrong choice of direction. This requires memory of size $O(n)$.

have only two neighbours. Traversing further through the graph, the walker either reaches the EXIT vertex or finds itself back in the central region. This region is reached after $2k$ steps after being there for the first time and in this case it is easy to backtrack the last k steps and choose the remaining neighbour. Performing this walk with memory (polynomial in n) for $O(n^2)$ steps will certainly give us the walker ending in the EXIT vertex whereas in the quantum case we need to perform $O(\text{poly}(n))$ steps. Thus, there is no great speedup coming from a quantum walk. However, this example is just the first step to an actual exponential speedup, discussed in Section 6.1.2.

2.3.2 Solving Satisfiability Problems

A prime example of locally constrained problems is Satisfiability. Its easiest variant, 2-SAT is to determine whether one can find an n-bit string $x_1 x_2 \cdots x_n$ that satisfies a boolean formula[6] with exactly 2 literals (bits or their negations) per clause. A 2-SAT instance with m clauses on n bits has the following general form of the boolean function:

$$\phi = (a_1 \vee b_1) \wedge (a_2 \vee b_2) \wedge \cdots \wedge (a_m \vee b_m), \tag{2.10}$$

where $a_i, b_i \in \{x_1, x_2, \ldots, x_n, \neg x_1, \neg x_2, \ldots \neg x_n\}$.

There exists a deterministic algorithm for solving 2-SAT, but we also know a beautiful random-walk approach to the same problem. The algorithm (first analyzed by Papadimitriou) is this:

1. start with $x_1 x_2 \cdots x_n = 11 \cdots 1$.

2. **while** ϕ contains an unsatisfied clause (and #steps $< 2n^2$) **do**
   pick an unsatisfied clause C arbitrarily
   pick one of the two literals in C uniformly at random and flip its value
   if ϕ is now satisfied **then** output "yes" and stop

3. output "no"

This algorithm performs a random walk on the space of strings. One can move from string s_1 to string s_2 if they differ by a single bit value x_t, and bit t belongs to a clause that is unsatisfied for string s_1. Why does this algorithm work? Assume there is a solution to the 2-SAT instance. Call $d(s)$ the Hamming distance of string s from the correct solution. In each step of the algorithm, $d(s_{k+1}) = d(s_k) \pm 1$, because we flip a single bit. However, we claim that the probability of the Hamming distance to the correct solution decreases in each step is $\geq \frac{1}{2}$. Here's the reason for this. We know that when we choose the clause C, it is unsatisfied, so both its bits cannot have the correct value. Imagine both of the bits in the chosen clause C were wrong. We then surely decrease d. If one of the bits in C was wrong, we have a 50% chance of choosing correctly. Together, this means the chance of decreasing d in each step are $\geq \frac{1}{2}$. Our random walk algorithm then performs no worse than a random walk on a line graph with $n+1$ vertices $\{0, 1, \ldots, n\}$, corresponding to Hamming distances of strings from the solution. When we hit 0, we have the solution! The expected number of steps it takes to hit the endpoint on a line of length $n+1$ is

---

[6]in conjunctive normal form

n^2. Therefore, using Markov's inequality we can show that running the algorithm for $2n^2$ steps results in finding the satisfying assignment (if it exists) with probability $\geq \frac{1}{2}$.

Go to 3-SAT. Basic algorithm and simple analysis gives probability of success $3^{-n/2}$, which translates to runtime $(1.78)^n$. If you think about taking $3n$ steps, the probability of success increases a lot – see Luca Trevisan's notes, Schöning's algorithm (1999). Taking $3k$ steps (when starting k-away from the solution), we can calculate the probability of success to be at least $\frac{1}{2^k \sqrt{k}}$. When we sum over all starting points and take $3n$ steps, we arrive at an algorithm with runtime $(1.33)^n$.

This can be further improved, small changes – GSAT, WALKSAT – restarts, greedy choices of which clause to flip, etc.

2.3.3   Markov Chain Monte Carlo Algorithms

Previously mentioned algorithms are based on a Markov chains. These algorithms are designed for finding a solution within many instances. On the other hand there is a large class of physical problems that do not require the knowledge of one particular instance of the state of the system. In these problems we are interested in statistical properties[7] of various, often physical, systems, such as knowing averages of various quantities describing the system under consideration. For such problems an efficient method of *Monte Carlo* has been devised. We can view random walks as a special approach within this method, that employs Markovianity in the search within the phase space of the problem. The system in these simulations undergoes Markov evolution with specific properties creating a chain of states that are sampled from desired probability distribution.

If we want to estimate a time average of some quantity of a system in equilibrium we either have the option to study the actual time evolution, or we can use *ergodic hypothesis* and estimate this time-average by estimating an average over an ensemble of such systems, where the actual distribution is known but, due to computational restrictions it is difficult to use it directly in (analytical) computation of averages. In simulations, when the sampling process has to be discretized, to find average of some state $\mathbf{x}$ quantity Q, we can use formula

$$\langle Q \rangle \simeq \sum_{j=1}^{N} Q(\mathbf{x}_j) p[\mathcal{H}(\mathbf{x})],$$

where $p[\mathcal{H}(\mathbf{x})]$ is the probability (depending only on the energy) of being in the near vicinity of state $\mathbf{x}_j$. These N states are sampled uniformly from the phase space. This is, however, not useful, when only states from a small part of phase space make a major contribution to the average. This makes the sum to converge very slowly.

In order to overcome this drawback, a technique, called *importance sampling* is employed, when the states are not sampled from uniform distribution, but from a distribution that is close to the desired $p[\mathcal{H}(\mathbf{x})]$. Quite surprisingly, this is not so difficult to achieve, as we will see soon, and when we are finally sampling states $\mathbf{x}_j$ from the desired distribution, we can estimate the

---

[7]As we will see in the following section, such statistical properties can be, on the other hand, useful in finding optima of various functions.

average as

$$\langle Q \rangle \simeq \frac{1}{N} \sum_{j=1}^{N} Q(\mathbf{x}_j)\big|_{\mathrm{pdf}(\mathbf{x}_j=\mathbf{x})=p[\mathcal{H}(\mathbf{x})]}$$

with N being the normalization for the average.

As said before, random walks come into the play in the form of Markov processes where a walk in the phase space according to preferred distribution is performed. Employing the notation defined in Sec. 2.1 we will show, how to construct such walk in discrete space and time. In general, the change of a distribution p in a step is described by "continuity" equation

$$\Delta p_j(m) = -\sum_{j'} M_{j',j} p_j(m) + \sum_{j'} M_{j,j'} p_{j'}, \qquad (2.11)$$

where M is the stochastic transition matrix. This equation just states, that the part of probability that leaves state j (first term) and the part of probability that "flows" into the state j correspond to the change of the probability in the state at time m, $p_j(m)$. This condition in the case of a system in equilibrium, when $p_j(m) \equiv p_j$ reads

$$\sum_{j'} M_{j',j} p_j = \sum_{j'} M_{j,j'} p_{j'}$$

stating that whatever probability flows out of the state j must be replenished. This condition is called *global balance* and is still quite complex for simple utilisation. However, the necessary condition called *detailed balance*

$$\frac{M_{j',j}}{M_{j,j'}} = \frac{p_{j'}}{p_j}$$

is suitable for computational purposes.

Indeed, if we take, for example, the choice of Metropolis (and Hastings) [10, 11],

$$M_{j',j} = \min\left\{1, \frac{p_{j'}}{p_j}\right\}, \qquad (2.12)$$

we have a way how to effectively sample the phase space under given distribution — being in state j we randomly choose state j' and conditioned on transition rate from Eq. (2.12), which is easily computable if p_j is known, we replace the state j by j'. See also Fig. 2.6.

Example 4 (Sampling from the Gibbs canonical distribution) This is a special case of the Metropolis-Hastings algorithm. The Gibbs distribution is given as

$$p_j = \frac{1}{\mathcal{Z}} \exp[-\mathcal{H}(j)/k_B T],$$

where $\mathcal{Z}$ is partition function, $\mathcal{H}$ is energy of the system in state j, k_B is Boltzmann constant and T is temperature. In this case

$$M_{j',j} = \min\left\{1, \exp\left[-\frac{\mathcal{H}(j') - \mathcal{H}(j)}{k_B T}\right]\right\}. \qquad (2.13)$$

Initialize $\mathbf{x}_0 = \mathbf{x}$, $k = 0$, $\langle Q \rangle = 0$ and for N steps repeat:

1. choose random state $\mathbf{x}_{k'}$
2. evaluate $M_{k',k}$ according to Eq. (2.13)
3. generate random number q
4. if $q \leq M_{k',k}$ set $\mathbf{x}_{k+1} := \mathbf{x}_{k'}$ otherwise $\mathbf{x}_{k+1} := \mathbf{x}_k$
5. set $k := k + 1$ and if $k > N - N'$ evaluate $\langle Q \rangle := \langle Q \rangle + Q(\mathbf{x}_k)/N'$
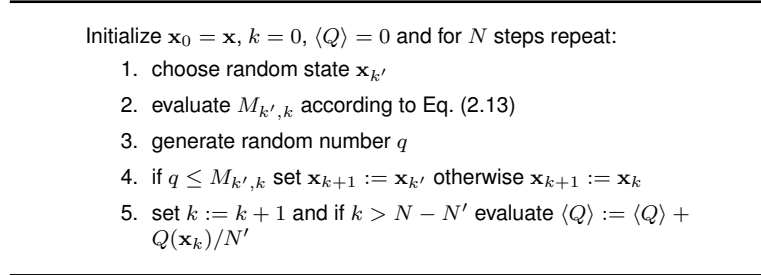

Fig. 2.6. Monte Carlo algorithm with Metropolis acceptance criterion using Gibbs distribution for estimating the average of Q - this is done only in last N' steps to allow relaxation of the system. Inputs of the algorithm are the initial state $\mathbf{x}$ and temperature. The choice of $\mathbf{x}'$ in every step is usually according to some distribution that places $\mathbf{x}'$ close to $\mathbf{x}$.

This tells us, that if the energy in the test state j' is lower than actual energy, then accept the state j' as new state, otherwise accept it only with probability

$$\exp\left[-\frac{\mathcal{H}(j') - \mathcal{H}(j)}{k_B T}\right].$$

Note that these results are in principle applicable also to a continuous phase space, with proper modifications to the above equations. Also Eq. (2.11) can be rewritten to a continuous time form by exchanging the difference in probabilities by a time derivative, and instead of the stochastic matrix M, using a transition matrix containing the rates of changes (see Sec. 6).

Preparing and sampling from the Gibbs canonical distribution is interesting, as it is the basis of MCMC methods (based on telescoping sums) for estimating partition functions, allowing one in turn to approximately count the number of ground states of a system. When applied to the Potts model on a particular graph, it becomes a tool for combinatorial problems whose goal is counting (e.g. estimating the number of perfect matchings in a graph). In Section 5.4.2, we will investigate and look at MCMC algorithms and their quantum counterparts in detail.

2.3.4 Simulated Annealing

Simulated annealing [12,13] is a physically motivated method that allows us to search for ground states of simulated systems in the same way as in experiments slowly cooled material tends to get to its ground state (see also Fig. 2.7). In a more abstract way we can say, that we perform a search for (global) minima[8] of some fitness function $\mathcal{F}$ that represents energy in these models.

If we look at the annealing as a succession of system states that relaxes under slowly decreased temperature, we can easily devise its computational analog. The succession of system states is the crucial part, where on one hand we tend to choose a state that is close to the previous one. This, in fact, defines us a graph the system (walker) walks on, even more, when the system is described by discrete variables such as spins — a step in such case may be described as a spin-flip on some position. On the other hand for the walk we need to have the transitional

---

[8]Note that search for maxima of function $\mathcal{F}$ is equivalent to the search for minima of function $-\mathcal{F}$.

Initialize $\mathbf{x}_0$ randomly and set T to high enough value. Then while $T > T_{m}in$ repeat:

1. do MC algorithm (with N interations) from Fig. 2.6 with input parameters of $\mathbf{x}_0$ and given T

2. set $T := \mu T$ and $\mathbf{x}_0 := \mathbf{x}_N$

Output $\mathbf{x}_0$ as (sub)minimum

Fig. 2.7. Simulated annealing algorithm for finding (sub)optimal value of $\mathbf{x}$. Value of $0 \ll \mu < 1$ should be chosen so that the decrease of the temperature would not be fast and the temperature T should be initially high enough; T_{min} should be chosen close enough to zero to prevent jumps to other local minima of the energy function. Also during the runnig of the algorithm one can look at averages of interesting observables.

probabilities defined. The sampling of these states is then simply governed by the Metropolis algorithm and its utilization for sampling from Gibbs distribution, which is suitable for the task, as in the limit of zero temperature it becomes a distribution on the ground states of $\mathcal{F}$ only. So, in the end of the day Eq. (2.13) is used with fitness function $\mathcal{F}$ in the role of the energy $\mathcal{H}$.

The complete algorithm of simulated annealing starts with the initialization of the system (on random) with the temperature being high[9]. Then the system is let to evolve by the above-mentioned procedure. When we are sure that the system is in equilibrium in this cycle we decrease the temperature a little. Then, again we let the system evolve for another cycle and decrease temperature further, e.g. by following rule $(kT)^{-1} = \sqrt{\#cycles}$ or just by setting $T := \mu T$ for $0 \ll \mu < 1$. This process is repeated until acceptably small temperature T_{min} is achieved, when the system should be close to its ground state. All the parameters of the annealing have to be chosen carefully so that the cooling would be slow enough to allow the system to get to the state with lowest energy, yet not as slow as not to allow reasonable runnign time of the simulation.

2.4   Summary

In this section we have briefly described random walks, their properties and some applications. In the discrete case the evolution is governed by stochastic matrices while the system is described by a probability distribution on given state space. Theoretically studied properties of random walks were successfully applied in various scientific fields. Most prominent but not exclusive are their algorithmic application for probabilistic computations — Markov Chain Monte Carlo. We have also studied few basic properties of the drukard's walk that will be later used to show the difference between classical random and quantum walks. Such quantities (mixing time, limiting distribution, etc.) serve on the other hand as a merit of efficiency with which the walks can be used in more practical applications.

---

[9]What "high" means is out of the scope of this paper and a lot of attention is devoted to setting all the parameters of the annealing right.

3 Quantum Walks: Using Coins

Correctly employed, the non-classical features of quantum mechanics can offer us advantages over the classical world in the areas of cryptography, algorithms and quantum simulations. In the classical world, random walks have often been found very practical. It is then quite natural to ask whether there are quantum counterparts to random walks exist and whether it would be possible to utilize them in any way to our profit. As many simple questions, these two also require more than just simple answers. In the following sections we will attempt to construct answers for these questions in small steps. Our starting point will be one of the simplest translations of random walks to the quantum domain — looking at discrete-time quantum walks taking place both in discrete space and discrete time. Their evolution will be described by an iterative application of a chosen unitary operator, advancing the walk by one step.

The first notions of a discrete-time quantum walk can be traced to Ref. [14]. The authors considered the spatial evolution of a system controlled by its internal spin-1/2 state, defined by the unitary $U = \exp(-\mathrm{i}S_z P \delta/\hbar)$. The operators P and S_z correspond to the momentum and the z-component of the spin of the particle. The initial state $|\psi(x_0)\rangle(c_+|\uparrow\rangle + c_-|\downarrow\rangle)$ under the action of U evolves into the state

$$|\Psi\rangle = c_-|\psi(x_0 - \delta)\rangle|\downarrow\rangle + c_+|\psi(x_0 + \delta)\rangle|\uparrow\rangle, \tag{3.1}$$

where $|\psi(x)\rangle$ corresponds to the wave function of the particle centered around position x. The evolution described by (3.1) will accompany us throughout this whole section as a key ingredient of discrete-time quantum walks — see e.g. (3.3). However, we will diverge from [14] on an important point – the way we measure the system. In the original paper, the authors studied repeated applications of the unitary U alongside with the measurement of the spin system and its repeated preparation. As we will see, this course of actions leads to "classical" random-walk like evolution (that is why the paper is called *Quantum random walks*). The authors have shown that using a well-chosen evolution [similar to (3.1) up to the choice of basis] one can steer the system in a desired direction. Together with a choice of a the initial state that has width much larger than δ, one can move this state spatially further than just the distance δ. This is explained as a result of interference and can be used for example to drastically reduce (or amplify) the average number of photons in a cavity, produced by the detection of a single atom after it interacts resonantly with the cavity field.

In quantum walks, the measurement is performed only once, at the end of the evolution. A repetitive measurement process (as the one in [14]) destroys quantum superposition and correlations that emerge during the evolution. Thus, we will use the definitions of quantum walks found in later references [15, 16], where we perform the measurement only at the end of the experiment. Similarly, continuous-time quantum walks (see Chapter 6), introduced by Farhi and Gutmann in [78] involve a unitary (continuous-time) evolution according to the Schrödinger equation for some time, followed by a single measurement.

3.1 Drawing an Analogy from the Classical Case

Let us consider the walk on a linear chain again and attempt to quantize it. We will start with an unsuccessful attempt and then learn from the mistake and find a meaningful way to do it.

In the classical case, a random walk on a linear chain is defined over the set of integers $\mathbb{Z}$. For a quantum walk, we shall consider a Hilbert space defined over $\mathbb{Z}$, i.e. $\mathcal{H} = \ell^2(\{|x\rangle :\ x \in \mathbb{Z}\})$, with the states $|x\rangle$ forming an orthonormal basis. Instead of being using a stochastic matrix, let us now try to describe the evolution by a unitary matrix. This simple "quantization" looks as a direct translation of random walks into the quantum world. However, it is easy to see that this model does not work as intended. Just as for the drunkard's walk, we require translational invariance of the unitary evolution. Thus if we start in the state $|x - 1\rangle$, in the next step it will turn into $\alpha|x - 2\rangle + \beta|x\rangle$ for some (complex) α and β. Similarly, the state $|x + 1\rangle$ will be mapped to the state $\alpha|x\rangle + \beta|x + 2\rangle$ with the same α and β. Note that the two initially orthogonal states $|x - 1\rangle$ and $|x + 1\rangle$ should remain orthogonal under any unitary evolution. However, this is now possible only if one of the coefficients α or β is zero. We can agree that such an evolution is even simpler than an evolution of the random walker (and quite boring).

The first attempt gave us a hint that using only the position space will not be enough for something interesting. Let us then try again and add another degree of freedom – a *coin space*, describing the direction[10] of the walker. This additional space will be a set $\Xi = \{\uparrow, \downarrow\}$ so that the state of a particle is described by a tuple $(x, c) \in \mathbb{Z} \times \Xi$. Because of this addition, we expand the Hilbert space to a tensor product $\mathcal{H} = \mathcal{H}_{\mathrm{P}} \otimes \mathcal{H}_{\mathrm{C}}$, where $\mathcal{H}_{\mathrm{P}} = \ell^2(\mathbb{Z})$ determines a position of the walker and $\mathcal{H}_{\mathrm{C}} = \ell^2(\Xi)$ is the introduced *coin space*[11] (in this example, the coin could correspond to the spin degree of freedom of a spin-$\frac{1}{2}$ particle). Therefore, $\mathcal{H}$ is spanned by the orthonormal basis[12] $\{|x\rangle \otimes |c\rangle :\ x \in \mathbb{Z}, c \in \Xi\}$. How will an initial state $|\psi_0\rangle \in \mathcal{H}$ evolve? We choose to describe a single evolution step by a unitary evolution operator

$$U = S\,(\mathbb{I} \otimes C). \tag{3.2}$$

It is a composition of $\mathbb{I} \otimes C$ acting nontrivially only on the coin space, and S, which involves the whole Hilbert space $\mathcal{H}$. The operator S describes spatial translation of the walker, while C correspond to coin throwing. Let us look at them in detail.

Translation operator S. This operator acts on the Hilbert space $\mathcal{H}$ as a conditional position shift operator with coin being control qubit. It changes the position of the particle one position up if the coin points up and moves the particle one position down, if the coin points down. It can be written in the form

$$S = \sum_{x \in \mathbb{Z}} \left(|x + 1\rangle\, \langle x| \otimes |\uparrow\rangle\, \langle\uparrow| + |x - 1\rangle\, \langle x| \otimes |\downarrow\rangle\, \langle\downarrow| \right). \tag{3.3}$$

We see that the structure of the graph (in this case the line) is reflected in this operator by the allowed transitions in position space only to the nearest neighbours.

Coin operator $\mathbb{I} \otimes C$. This unitary operator acts nontrivially only on the coin space $\mathcal{H}_{\mathrm{C}}$ and corresponds to a "flip" of the coin. For the quantum walk on line, C is a 2×2 unitary matrix. A

---

[10]Note that now we are diverging from a classical memory-less random walk, where a walker had "no idea" where it came from, or where it would be going in the next step.

[11]Discrete-time quantum walks using coins are also called *coined quantum walks*. Sometimes we will refer to them in this way.

[12]When using the states $|x\rangle \otimes |c\rangle$ we will often omit the tensor product symbol $\otimes$ to shorten the notation.

usual choice for C is the *Hadamard coin*[13]

$$C \equiv H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{3.4}$$

viewed in a different notation as

$$H : |\uparrow\rangle \mapsto \frac{1}{\sqrt{2}} (|\uparrow\rangle + |\downarrow\rangle), \qquad H : |\downarrow\rangle \mapsto \frac{1}{\sqrt{2}} (|\uparrow\rangle - |\downarrow\rangle).$$

Observe that this coin can be called *unbiased*, since the states $|\uparrow\rangle$ and $|\downarrow\rangle$ of the coin are evenly distributed into their equal superpositions, up to a phase factor. We shall return to the coin later in Sec. 3.4. In the next Section, we will investigate the evolution of the walker under the influence of the Hadamard coin.

Let us recall that a single step of a coined quantum walk is described by the operator U in (3.2). The state after n steps will thus be described by a vector from $\mathcal{H}$ as

$$|\psi_n\rangle = U^n |\psi_0\rangle. \tag{3.5}$$

3.2 Dispersion of the Hadamard Quantum Walk on Line

In this section we will investigate the quantum walk in 1D using the Hadamard coin (3.4) and compare it to the classical random walk. First, let us argue why we even talk about a connection of the Hadamard walk to the drunkard's walk (for more on the topic see Refs. [17, 18, 19, 20] and Section 3.5.4). Suppose the state of the system is $|x\rangle|c\rangle$ with $x \in \mathbb{Z}$ and c being one of the basis states $|\uparrow\rangle$ and $|\downarrow\rangle$. Then a single step of the quantum walk gives us

$$\begin{aligned} U|x\rangle|c\rangle &= S(\mathbb{I} \otimes H)|x\rangle|c\rangle = S(|x\rangle \otimes H|c\rangle) = \frac{1}{\sqrt{2}} S\left[|x\rangle \otimes (|\uparrow\rangle \pm |\downarrow\rangle)\right] = \\ &= \frac{1}{\sqrt{2}} (S|x\rangle|\uparrow\rangle \pm S|x\rangle|\downarrow\rangle) = \frac{1}{\sqrt{2}}|x+1\rangle|\uparrow\rangle \pm \frac{1}{\sqrt{2}}|x-1\rangle|\downarrow\rangle. \end{aligned}$$

From this result we may observe that starting in the state $|\psi_0\rangle = |x\rangle|\uparrow\rangle$ and performing a measurement on the position state space[14] right after an application of U, the probability of finding $(x+1)$ equals the probability of measuring $(x-1)$. Therefore, if we perform a position measurement after each[15] application of U, we end up with a classical random walk where we shift the position to either side with equal probability $\frac{1}{2}$.

However, as we described at the beginning of Chapte 3, we do not want to measure the system after each step. What will happen if we leave the system evolve for some time and measure the position of the particle then? Let us look at three applications of U. After some algebraic manipulation we arrive at

$$U^3|0\rangle|\uparrow\rangle = \frac{1}{\sqrt{8}}\left[|3\rangle|\uparrow\rangle + |1\rangle\big(|\downarrow\rangle + 2|\uparrow\rangle\big) - |-1\rangle|\uparrow\rangle + |-3\rangle|\downarrow\rangle\right]. \tag{3.6}$$

---

[13]Quantum walk on line using the Hadamard coin is often called the *Hadamard walk*.

[14]Note that we can choose to measure the coin register instead and also end up with $|x+1\rangle$ or $|x-1\rangle$ in the position register with equal probability. Tis is because the two registers become entangled after the application of U.

[15]We could imagine performing the measurement only after a certain number of steps or probabilistically, giving us a continuum of behaviors between quantum and classical. We explore this in Section 3.5.4.
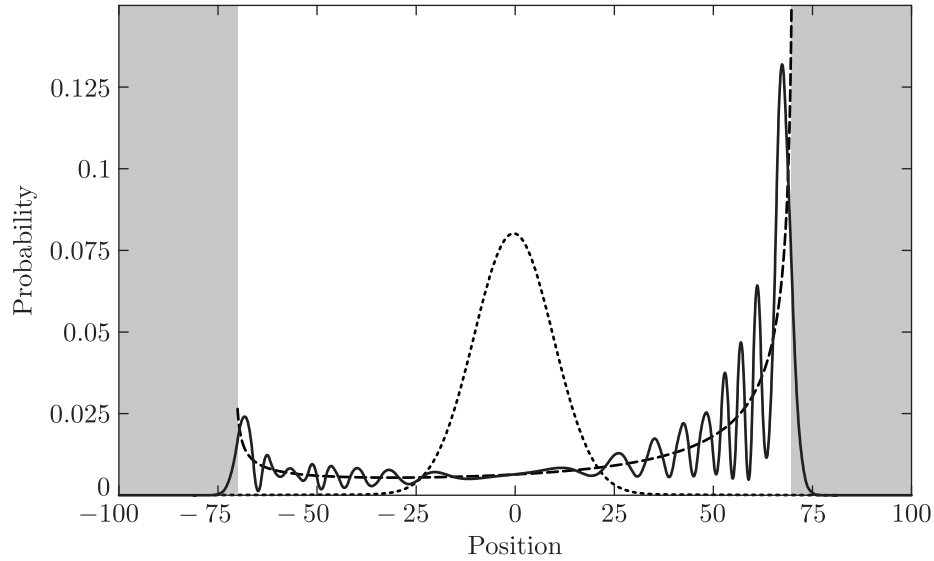
Fig. 3.1. Comparison of probability distributions after 100 steps for the Hadamard walk on even positions (black line) starting in state $|0\rangle|\uparrow\rangle$ and the drunkard's walk (starting on position 0). Spreading of quantum walk (determined by the variance) is quadratically faster than that of random walk. Hadamard walk can be in region $[-100/\sqrt{2}; 100/\sqrt{2}]$ approximated by function P^{100} (dashed line) given by Eq. (3.14) suitable also for evaluation of moments. Outside the region (grey areas), the probability drops exponentially fast. Distribution of the classical walker is depicted by dotted line.

This already shows that the amplitudes are not symmetric and start to show interference effects. Let us look at the system after m steps. In the final measurement, we do not care about the state of the coin. Thus to get the probability for the particle being on the position x after m steps, we trace the coin register out, and look at

$$p^m(x) = |(\langle x| \otimes \langle\uparrow|)U^m|\psi_0\rangle|^2 + |(\langle x| \otimes \langle\downarrow|)U^m|\psi_0\rangle|^2 . \tag{3.7}$$

Looking at three steps of evolution of the Hadamard walk in (3.6), we can read out the values of these probabilities after $n = 3$ steps (when starting from $|0\rangle|\uparrow\rangle$):

$$p^3(-1) = p^3(3) = p^3(-3) = \frac{1}{8}, \qquad p^3(1) = \frac{5}{8}$$

and zero otherwise. Notice the asymmetry between $p^3(1)$ and $p^3(-1)$. We do not observe asymmetrical behavior for classical random walks with an unbiased coin. This is the first difference between quantum and classical random walks that we have seen. The situation is even more interesting for longer evolution times, as depicted in Fig. 3.1 where the system is shown after 100 steps. We see that the distribution for the Hadamard quantum walk starting from $|0\rangle|\uparrow\rangle$ is indeed asymmetrical. What is more interesting, the probability spreads faster than for a classical walk.

We will follow [16,21] to show that the spreading in the case of the quantum walk on a line is linear with time m (the number of steps), meaning that the standard deviation grows as $\sigma \sim m$ as

opposed to the classical case where $\sigma \sim \sqrt{m}$ as shown in (2.4). Moreover, we can see in Fig. 3.1 that the distribution for the quantum walk gets "close" to uniform[16] in the interval $\left[-\frac{m}{\sqrt{2}}, \frac{m}{\sqrt{2}}\right]$.

The method we will use for the analysis is based on switching[17] to the *Fourier basis* in the position space, where

$$|\tilde{k}\rangle = \sum_{x=-\infty}^{\infty} \mathrm{e}^{\mathrm{i}kx}|x\rangle, \qquad k \in [-\pi; \pi].$$

A converse transformation gives us

$$|y\rangle = \frac{1}{2\pi} \int_{-\pi}^{\pi} \mathrm{e}^{-\mathrm{i}ky}|\tilde{k}\rangle \, \mathrm{d}k,$$

with the special case

$$|0\rangle = \frac{1}{2\pi} \int_{-\pi}^{\pi} |\tilde{k}\rangle \, \mathrm{d}k. \tag{3.8}$$

The Fourier basis is useful since the states $|\tilde{k}\rangle \otimes |\uparrow\rangle$ and $|\tilde{k}\rangle \otimes |\downarrow\rangle$ are eigenvectors of the translation operator S corresponding to the eigenvalues $\mathrm{e}^{\mp \mathrm{i}k}$ respectively.

For the Hadamard walk defined in the previous section, with the Hadamard operator used as the coin operator (3.4), and choosing the initial state $|\psi_0^c\rangle = |0\rangle \otimes |c\rangle$ (where c can be either the spin up, or spin down state) and expressing it in the Fourier basis, we find

$$|\psi_m^c\rangle = U^m|\psi_0^c\rangle = U^m(|0\rangle \otimes |c\rangle) = \frac{1}{2\pi} \int_{-\pi}^{\pi} |\tilde{k}\rangle \otimes M_k^m|c\rangle \mathrm{d}k, \tag{3.9}$$

where M_k denotes the 2×2 matrix

$$M_k = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathrm{e}^{-\mathrm{i}k} & \mathrm{e}^{-\mathrm{i}k} \\ \mathrm{e}^{\mathrm{i}k} & -\mathrm{e}^{\mathrm{i}k} \end{bmatrix}$$

It turns out that all we really need to know to analyze the evolution is the eigenspectrum of the operator M_k. Introducing new variables $\omega_k \in [-\pi/2; \pi/2]$ and setting $\sin k = \sqrt{2} \sin \omega_k$, we find that the eigenvalues of M_k are $\lambda_- = \mathrm{e}^{-\mathrm{i}\omega_k}$ and $\lambda_+ = -\mathrm{e}^{\mathrm{i}\omega_k}$. The corresponding eigenvectors are

$$|\phi_-\rangle = \frac{1}{N_-} \begin{bmatrix} 1 \\ \sqrt{2}\mathrm{e}^{\mathrm{i}(k-\omega_k)} - 1 \end{bmatrix} \quad \text{and} \quad |\phi_+\rangle = \frac{1}{N_+} \begin{bmatrix} 1 \\ -\sqrt{2}\mathrm{e}^{\mathrm{i}(k+\omega_k)} - 1 \end{bmatrix}$$

with the normalization factors given by

$$N_\pm^2 = 2(1 + \cos^2 k \pm \cos k \sqrt{1 + \cos^2 k}).$$

---

[16]When considering decoherence in quantum walks as in [18], the distribution of the quantum walker may get even closer to a uniform distribution (see also Section 3.5.4).

[17]There are also other ways how to deduce the asymptotic behaviour of the quantum walk on line, especially employing the knowledge from the theory of path integrals. Note here, that the Fourier basis used is not normalizable, however, under careful manipulation is very useful.

We can expand the expression $M_k^m |c\rangle$ in (3.9) in the eigenbasis of the operator M_k as

$$M_k^m |c\rangle = \left(\lambda_-^m \langle \phi_- | c\rangle\right) |\phi_-\rangle + \left(\lambda_+^m \langle \phi_+ | c\rangle\right) |\phi_+\rangle.$$

Inserting it into (3.9) we arrive at

$$|\psi_m^c\rangle = \sum_x |x\rangle \otimes \left[A_c^m(x) |\uparrow\rangle + B_c^m(x) |\downarrow\rangle \right]. \tag{3.10}$$

Exercise 3 Prove the following equalities

$$
\begin{aligned}
A_\uparrow^m(x) &= \frac{1 + (-1)^{m+x}}{2} \left[\alpha^m(x) + \beta^m(x) \right], \\
A_\downarrow^m(x) &= \frac{1 + (-1)^{m+x}}{2} \left[\beta^m(x) - \gamma^m(x) \right], \\
B_\uparrow^m(x) &= \frac{1 + (-1)^{m+x}}{2} \left[\beta^m(x) + \gamma^m(x) \right], \\
B_\downarrow^m(x) &= \frac{1 + (-1)^{m+x}}{2} \left[\alpha^m(x) - \beta^m(x) \right],
\end{aligned}
$$

where

$$\alpha^m(x) = \int_{-\pi}^{\pi} \frac{dk}{2\pi} e^{i(kx - m\omega_k)}, \tag{3.11a}$$

$$\beta^m(x) = \int_{-\pi}^{\pi} \frac{dk}{2\pi} \frac{\cos k}{\sqrt{1 + \cos^2 k}} e^{i(kx - m\omega_k)}, \tag{3.11b}$$

$$\gamma^m(x) = \int_{-\pi}^{\pi} \frac{dk}{2\pi} \frac{\sin k}{\sqrt{1 + \cos^2 k}} e^{i(kx - m\omega_k)}. \tag{3.11c}$$

Exercise 4 Prove that all $a^m(x)$, $b^m(x)$ and $c^m(x)$ are real and that

$$a^m(x) = (-1)^x a^m(x),\ b^m(x) = -(-1)^x b^m(x),\ c^m(x) = -(-1)^x c^m(x).$$

At this point we may observe the *modularity* property of the walk (previously mentioned in Section 2.2): after an even (odd) number of steps, the walker cannot be on odd (even) positions.

The results we obtained so far are valid only for states initialized with a computational basis state of the coin – spin up or down. If we would like the initial state to be completely general, having form

$$|\psi_0\rangle = |0\rangle \otimes \left[\sqrt{q} |\uparrow\rangle + \sqrt{1 - q}\, e^{i\sigma} |\downarrow\rangle \right] \tag{3.12}$$

parametrized by real parameters q and σ, the state after m steps will be a superposition of evolved states whose initial coins were set to be either spin-up or spin-down, i.e.

$$
\begin{aligned}
|\psi_m\rangle = U^m |\psi_0\rangle \ = \sum_x \ |x\rangle \otimes \Big\{ &\left[\sqrt{q} A_\uparrow^m(x) + \sqrt{1 - q}\, e^{i\sigma} A_\downarrow^m(x) \right] |\uparrow\rangle \\
&+ \left[\sqrt{q} B_\uparrow^m(x) + \sqrt{1 - q}\, e^{i\sigma} B_\downarrow^m(x) \right] |\downarrow\rangle \Big\}.
\end{aligned}
$$

The probability to find the walker at position x after m steps is then expressed as

$$p^m(x) = \left| \sqrt{q} A_\uparrow^m(x) + \sqrt{1-q}\,\mathrm{e}^{\mathrm{i}\sigma} A_\downarrow^m(x) \right|^2 + \left| \sqrt{q} B_\uparrow^m(x) + \sqrt{1-q}\,\mathrm{e}^{\mathrm{i}\sigma} B_\downarrow^m(x) \right|^2.$$

Exercise 5 Show that

$$
\begin{aligned}
p^m(\pm x) &= [\alpha^m(x)]^2 + 2\,[\beta^m(x)]^2 + [\gamma^m(x)]^2 \pm (4q-2)\beta^m(x)\,[\alpha^m(x) \pm \gamma^m(x)] \\
&\quad \pm 4\sqrt{1-q}\cos\sigma\,\beta^m(x)\,[\alpha^m(x) \mp \gamma^m(x)].
\end{aligned}
\tag{3.13}
$$

The integrals in (3.11) can be approximated by employing the method of stationary phase (see Appendix B) and we find that the functions $a^m(x)$, $b^m(x)$ and $c^m(x)$ are mostly concentrated within the interval $\left[-\frac{m}{\sqrt{2}}; \frac{m}{\sqrt{2}} \right]$ and they quickly decrease beyond the bounding values of this interval. This also holds for the probability $p^m(x)$, which oscillates around the function

$$P^m(x) = \frac{2m}{\pi(m-x)\sqrt{m^2 - 2x^2}}, \tag{3.14}$$

where we dropped the vanishing part coming from modularity. For details of deriving (3.14), see Appendix B.2.

The function $P^m(x)$ allows us to approximately evaluate the averages of position x-dependent functions $f^m(x)$ in the m-th step of the walk as

$$\langle f^m(x)\rangle \simeq \frac{1}{2} \int_{-\frac{m}{\sqrt{2}}}^{\frac{m}{\sqrt{2}}} f^m(x) P^m(x)\,\mathrm{d}x. \tag{3.15}$$

The factor $\frac{1}{2}$ comes from modularity (indeed it is easy to check that using (3.15), we correctly obtain $\langle 1 \rangle = 1$. For more interesting x-dependent functions we obtain the approximations

$$
\begin{aligned}
\langle x \rangle &\simeq m\frac{\sqrt{2}-1}{\sqrt{2}}, \\
\langle |x| \rangle &\simeq \frac{m}{2}, \\
\langle x^2 \rangle &\simeq m^2\frac{\sqrt{2}-1}{\sqrt{2}}.
\end{aligned}
$$

The results shows also that the dispersion grows as $\sqrt{\langle x^2 \rangle} \sim m$ for large m, increasing linearly with time. This is quadratically faster than the classical random walk (2.4), whose dispersion grows with the number of steps as $\sqrt{m}$.

We can use (3.13) to find several other properties of the evolution. When tuning the parameter σ in the initial coin state (3.12), we can find many possibilities that result in a symmetrical probability distribution $p^m(x) = p^m(-x)$. For example, we can choose $\cos\sigma = 0$, which gives a symmetrical final distribution for $q = \frac{1}{2}$, i.e. when the initial state is

$$|\psi_0\rangle = |0\rangle \otimes \frac{1}{\sqrt{2}}[|\uparrow\rangle \pm \mathrm{i}|\downarrow\rangle]. \tag{3.16}$$

Similarly, we can choose $\cos\sigma = \pm 1$, in which case $q = (2 \mp \sqrt{2})/4$, corresponding to $\sqrt{q} = \sin\frac{\pi}{8}$ or $\sqrt{q} = \cos\frac{\pi}{8}$, again results in a symmetrical distribution.

Instead of looking at symmetrical evolutions, we can just as well try to find the most asymmetrical one. Let us look at the first moment of the probability distribution,

$$\langle x \rangle = \sum_{x=-\infty}^{\infty} x p^m(x) \sum_{x=1}^{\infty} 4x\alpha^m(x)\beta^m(x) \left[(2q-1) + 2\sqrt{q(1-q)}\cos\sigma \right].$$

From Appendix B.2 we know that $\alpha^m(x)\beta^m(x)$ is positive for our range of positions. This tells us that the maximal right asymmetry can be obtained for $\cos\sigma = 1$ and $q = (2 + \sqrt{2})/4$, while the maximal left asymmetry can be obtained for $\cos\sigma = -1$ and $q = (2 - \sqrt{2})/4$. All these results are in correspondence with the results of [27].

Exercise 6 Looking at the mean position is not a good general indicator showing the asymmetry of a distribution. We were able to use it above, as the evolution began in the position zero. In general, the asymmetry of a distribution is given by its *skewness*

$$\mathrm{Skew}(x) = \left\langle \left(\frac{x - \langle x \rangle}{\langle x^2 \rangle} \right)^3 \right\rangle.$$

Using the approximations (B.3) given in Appendix B.2, compute the skewness for the Hadamard walk and find the most asymmetric one [Hint: it should be the initial state choice we already found above].

Example 5 We can illustrate the result about fast spreading of the quantum Hadamard walk also by considering the *(Shannon) entropy* defined on distributions as

$$S(m) = - \sum_{x=-\infty}^{\infty} p^m(x) \ln p^m(x), \tag{3.17}$$

on the probabilities $p^m(x)$ for the quantum walk defined by (3.7). This quantity gives us yet another characterization of how the probability of the walker's position is distributed and we shall study its dependence on the number m of steps taken in a given walk. Let us look at what the entropy looks like for certain cases. First, if only one position x_0 was possible (and predicted with certainty) for the particle at a particular moment, then $p(x) = \delta_{x,x_0}$ and the sum in (3.17) would be zero. Second, we can imagine the distribution function evolving so that after m steps it would be uniformly distributed (maximally mixed) between all possible locations in the interval $[-m, m]$, taking modularity into account (excluding unreachable points for both random and quantum walks). For such an evolution we may write

$$p^m(x) = \begin{cases} \frac{1 + (-1)^{m+x}}{m+1} & \text{if } -m \leq x \leq m, \\ 0 & \text{otherwise.} \end{cases}$$

The entropy for such a distribution computed from (3.17) gives us
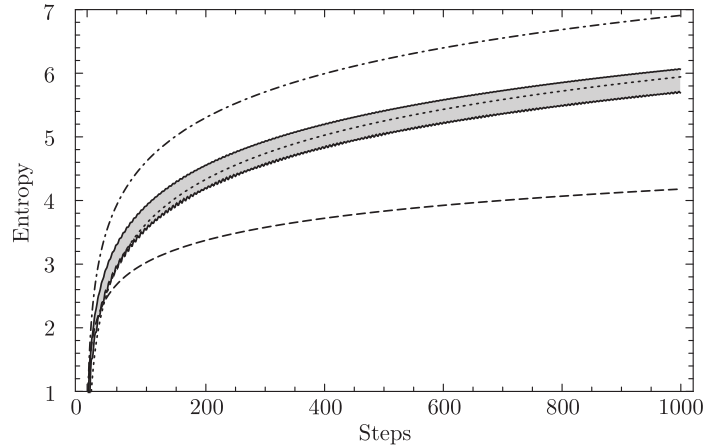
$$S^{\max}(m) = \ln(m+1).$$

Fig. 3.2. Shannon entropy [Eq. (3.17)] vs. number of steps taken for probability distributions of different walks, provided the walker starts at position 0 (state $|0\rangle|c\rangle$). The dot-dashed line is an upper boundary for entropy reachable by any walk, the dashed line is the entropy of the drunkard's walk distribution. The shaded area represents the region for entropy values of quantum walks. Its upper boundary is given by symmetric and lower boundary by maximally asymmetric coined quantum walks (see Sec. 3.4 for further information on symmetry of the distribution). The dotted line is the Shannon entropy for the probability distribution coming from the state (3.10), the Hadamard walk evolution of the initial state $|0\rangle|\uparrow\rangle$.

This is, in fact, an upper bound on the entropy that is achievable by any walk on a given graph (see the dot-dashed line in Fig. 3.2). Third, in the classical drunkard's walk, the probability distribution approaches a normal distribution given by (2.3), and its entropy can be easily evaluated as

$$S^c(m) = \frac{1}{2}\left(1 + \ln\frac{\pi m}{2}\right).$$

Finally, in the quantum case, we can observe from Fig. 3.2 that the entropy is larger than in the classical case given by the dashed line. Even though this entropy depends on the initial coin state, it still tells us that coined quantum walks spread faster than classical random walks also with respect to the Shannon entropy.

3.3   Coined Quantum Walks on General Graphs

Equations (3.2)-(3.4) describe quantum walks on a line using the unitary update U composed from a coin-diffusing operation and a subsequent translation. The generalization to more general graph structures requires just a slight modification to these equations and only slightly more explanation. We can view the coin degree of freedom Ξ in the 1D walk, given by the two states $\uparrow$ and $\downarrow$, as a coloring of a directed version of the graph. For each x, the "color" $\downarrow$ was assigned to the directed edges of the graph pointing from x to $x - 1$, while the "color" $\uparrow$ was assigned to the directed edges of the graph pointing from x to $x + 1$. Therefore, even though the graph (line) on which we walked is not directed, each edge can be interpreted as a set of two oppositely directed edges.

Let us present a simple extrapolation of the 1D formalism which allows us to define quantum walks on d-regular (undirected) graphs $\mathcal{G} = (\mathcal{N}, E)$, where $\mathcal{N}$ is the set of vertices and $E \subseteq \mathcal{N} \times \mathcal{N}$ is the set of edges defining the graph. First, we interpret each edge as two oppositely directed edges and then, as described in [15], we assign a "color" number from 1 to d to each of these directed edges in such a way that all the directed edges of one color form a permutation of the vertices. In other words, each vertex has exactly one outgoing and one incident edge with a given color. Such a coloring is always possible using d colors. Thus, in addition to the position space of vertices $\mathcal{H}_\mathrm{P}$, we expand the Hilbert space with the coin subspace $\mathcal{H}_\mathrm{C}$ with dimension d as $\mathcal{H} = \mathcal{H}_\mathrm{P} \otimes \mathcal{H}_\mathrm{C}$.

The motion of the walker will be described by the translation operator S, generalized to

$$S = \sum_{x \in \mathcal{N}} \sum_{c=1}^{d} |x \oplus c\rangle\langle x| \otimes |c\rangle\langle c|,$$

where $x \oplus c$ is the vertex accessible from x by the edge with color c.

The coin-diffusion $C^{(d)}$ now involves a d-dimensional coin space. Furthermore, we can imagine that the coin-flipping operation can be position-dependent

$$C' = \sum_{x \in \mathcal{N}} |x\rangle\langle x| \otimes C_x^{(d)}, \tag{3.18}$$

with a different $d \times d$ matrix $C_x^{(d)}$ for different vertices x. The evolution of a general graph quantum walk is then governed by the two-step unitary $U = SC'$. Compare this C' to the position-independent (translationally-invariant) coin operator $\mathbb{I} \otimes C$ from Section 3.1 and observe that we recover it if we choose $C_x^{(d)} \equiv C$.

This generalization is often used in the literature, however, there is another (isomorphic) approach called *Scattering Quantum Walks*, first introduced by Hillery et al. [21], with the proof of the isomorphism provided in Ref. [22]. We describe this approach in the following Section. The basic similarities of the two approaches are easier seen after realizing the following points:

▶ The condition that all edges with some color a form a permutation of the vertices meansthat for every vertex x and for every color a there is exactly one vertex x_- from which you can get to x by the edge with color a and exactly one vertex[18] x_+ which is accessible from x by the edge with color a. This also means that every state $|x\rangle \otimes |c\rangle$ uniquely describes some edge from the directed graph.

▶ The coin C_x "determines" how is the amplitude of particle that came from edge with color a distributed to other edges, while S still has the "trivial" role of nudging the walker forward.

These facts can be translated as essentially describing scattering process, where

▶ The incident direction of a particle corresponds to the edge color.

---

[18]Vertices x_+ and x_- need not be the same; e.g. in a regular graph with an odd number of vertices and no loops, all (for all vertices and colors) x_+ and x_- cannot coincide. If they did, the edges with the same color would connect pairs of vertices and all these pairs would be disjoint, but if the number of vertices is odd, one cannot create this structure.

▶ A scattering process on a given vertex x corresponds to the action of the coin C_x. Note that the translation operator S is superfluous in this picture.

Up to this point, we have described the need for a separate coin in discrete-time quantum walks. Having a coin such as in (3.18) is viable only for d-regular graphs, even though we can generalize it and make it position-dependent. If the underlying graph structure is not regular, a simple description of the coin begins to be difficult and the irregularity of the graph structure becomes problematic as well. There are several approaches to alleviate this problem. One of them is to introduce a position dependent coin with a variable dimension [15, 23]. However, it means the Hilbert space cannot be factorized into the coin space and the position space anymore. The possible irregularity of the graph structure is in this case embedded in the evolution process effectively acting as an oracle [24]. The action of the position-dependent coin, however, has to be in correspondence with the graph structure as well. It is then not such great a leap to start thinking of also embedding the coin operator into the oracle. This second approach is in fact the scattering model of quantum walks we are about to define.

3.3.1   Scattering Quantum Walks (SQW)

Scattering quantum walks (SQW) describe a particle moving around a graph, scattering off its vertices. The state of the particle lives (is located on) the edges of a graph $\mathcal{G} = (\mathcal{N}, E)$, defined by a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of vertices and a set of edges $E \subseteq \mathcal{N} \times \mathcal{N}$. The Hilbert space $\mathcal{H}$ for such a walk on $\mathcal{G}$ is then defined as

$$\mathcal{H} = \ell^2(\{|m, l\rangle : m, l \in \mathcal{N}, ml \in E\}), \tag{3.19}$$

where ml is a short-hand notation for the edge connecting vertices m and l. This definition gives us a Hilbert space which is a span of all the *edge states*, which form its orthonormal basis. The state $|m, l\rangle$ can then be interpreted as a particle going from vertex m to vertex l.

Exercise 7 Show that the dimension of the Hilbert space for a scattering quantum walk is the same as for a corresponding coined quantum walk.

Let us look at the structure of the Hilbert space for a SQW. First, we have the subspaces A_k spanned by all the edge-states originating in the vertex k,

$$A_k = \ell^2(\{|k, m\rangle : m \in \mathcal{N}, km \in E\}).$$

Second, we have Ω_k, the subspaces spanned by all the edge-states that end in the vertex k,

$$\Omega_k = \ell^2(\{|m, k\rangle : m \in \mathcal{N}, mk \in E\}). \tag{3.20}$$

These subspaces don't overlap, as $A_k \cap A_l = \emptyset$ and $\Omega_k \cap \Omega_l = \emptyset$ for $k \neq l$. Moreover, $|\Omega_k| = |A_k|$ for all k, as the graph $\mathcal{G}$ is not oriented[19]. The dynamics of the quantum walk are described by *local unitary evolutions* scattering the walker "on vertex" k — describing the transition from the walker entering vertex k to the walker leaving it. Using our notation for the incoming and outgoing subspaces, the local unitary evolutions act as $U^{(k)} : \Omega_k \to A_k$, as depicted in Figure 3.3.

---

[19]Note that we could describe a SQW coming from an oriented graph as in e.g. [25,26], but not without complications. We thus decide to talk only about QW coming from undirected graphs here.
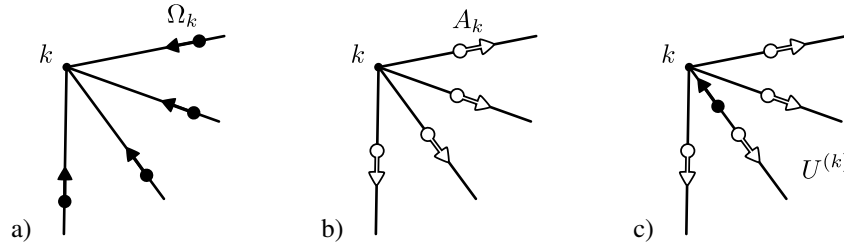
Fig. 3.3. a) The subspace Ω_k of walker states entering vertex k. b) The subspace A_k of walker states exiting vertex k. c) The action of the local unitary $U^{(k)} : \Omega_k \to A_k$ can be viewed as scattering on vertex k.

Example 6 The simplest example of a local unitary evolution $U^{(k)}$ for a 1D graph transforms a "right-moving" state $|k-1, k\rangle$ (moving from $k-1$ to k) into the uniform superposition $\frac{1}{\sqrt{2}} |k, k-1\rangle + \frac{1}{\sqrt{2}} |k, k+1\rangle$, while a "left-moving" state $|k+1, k\rangle$ similarly changes into $\frac{1}{\sqrt{2}} |k, k-1\rangle - \frac{1}{\sqrt{2}} |k, k+1\rangle$, with the minus sign required for unitarity. This transformation corresponds to the Hadamard coin (3.4) in DTQW.

We will see more local "coins" in the following Section 3.4, and then a general approach for finding such transformations coming from classical Markov Chains in Section 5.

The overall unitary U describing one step of the SQW acting on the system is then the combined action of the local unitary evolutions:

$$U = \Gamma \bigoplus_{k \in \mathcal{N}} U^{(k)},$$

where Γ is just a permutation on the basis elements so that their order would be restored. This can be done as

$$\bigcap_{k \in \mathcal{N}} \Omega_k = \bigcap_{k \in \mathcal{N}} A_k$$

gives the whole computational basis set. In other words, as all $U^{(k)}$ act unitarily on disjoint subsets of the Hilbert space, the overall operation U they define is also unitary and the restriction of U to Ω_k is just $U^{(k)}$. When the initial state of the system is $|\psi_0\rangle$, the state after m steps is given by $|\psi_m\rangle = U^m |\psi_0\rangle$ and the probability of finding the particle (walker) in the state $|k, l\rangle$ is then $|\langle k, l | \psi_m \rangle|^2$.

Let us compare discrete-time quantum walks to SQW. The action of the coin in discrete-time quantum walks is an analogue of the local unitary evolutions in SQW, transforming a single "incoming" state into several "outgoing" states from a particular vertex. In addition, the discrete-time quantum walk then requires the action of the translation operator, while in the SQW formalism this is already taken care of by switching the description of the vertex k into the second register as

$$U^{(k)} |j, k\rangle = \sum_{l=1}^{d} U_{j,k}^{(k)} |k, l\rangle. \tag{3.21}$$
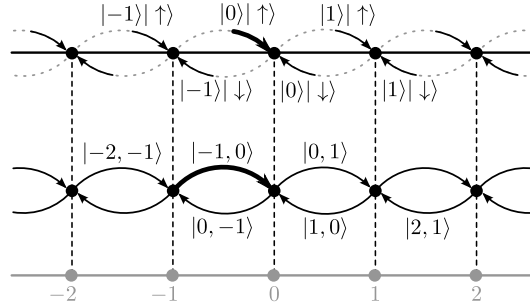
Fig. 3.4. Correspondence between coined quantum walk (top, for better readability shown without direct product symbol) and scattering quantum walk on a line. For example (thick arrows), state $|0\rangle \otimes |\uparrow\rangle$ represents the walker in coined quantum walk positioned on vertex 0 with coin pointing "up" stating that this was the direction the walker used to get to the vertex. This state corresponds in the scattering quantum walk formalism to the walker going from vertex -1 to vertex 0.

We conclude with a correspondence between the states (see Fig. 3.4) of a coined discrete-time QW and a scattering QW, given by

$$|x\rangle \otimes |c\rangle \quad \leftrightarrow \quad |x \ominus c, x\rangle.$$

It means the state "at" vertex x with a coin in the state $|c\rangle$ is nothing but a SQW state going "from" the vertex $x \ominus c$ "into" the vertex x. Moreover, when we talk only about the position of the walker at vertex x in coined quantum walks, it means that we are not interested in the coin state (where is the walker entering x from). In the language of scattering quantum walks this translates to talking about the particle entering vertex x, i.e. when the state of the walker is from subspace Ω_x.

3.4   More on Coins

We have seen that for both coined quantum walks and scattering quantum walks, the dynamics of evolution depends on a set of local unitary operators — position-dependent coins. There are many choices for them, but some turn out to be much more convenient than others for various uses. First, we will discuss the quantum walk properties arising from using different coins (or initial coin states). Second, we will investigate the usual choices of coins utilized in algorithmic applications.

3.4.1   Two-dimensional Coins

In Sec. 3.1, we had the oportunity to notice the need for an additional coin degree of freedom, in order to obtain a non-trivial evolution. We have looked at the quantum walk with the Hadamard coin operator (3.4), resulting in an asymmetrical distribution (see Fig. 3.1). If instead of the Hadamard coin we used the *balanced* coin operator

$$C = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix}, \tag{3.22}$$
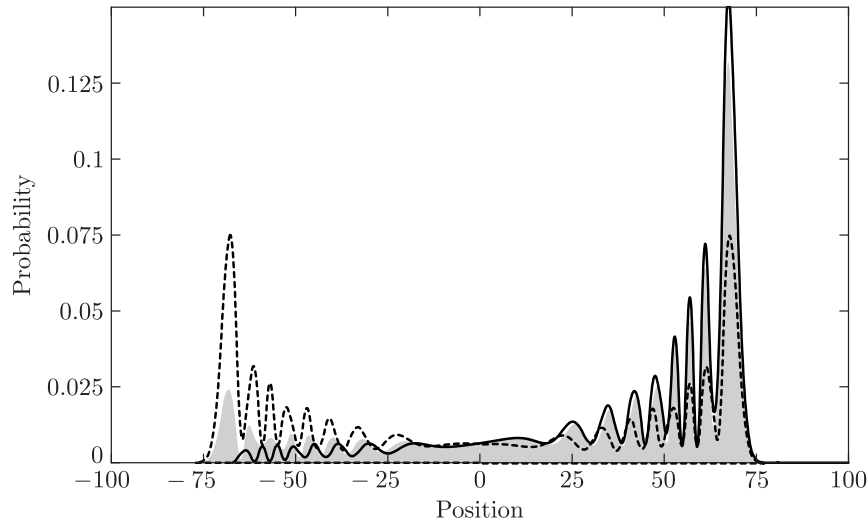
Fig. 3.5. Comparison of Hadamard walks after 100 steps, each starting at position 0 and differing only in the initial state of the coin. If the initial coin state is $|\uparrow\rangle$ (gray shading), the probability distribution is asymmetric. More asymmetry (solid line) is obtained by setting the initial coin state in (3.12) as a particular superposition with $\cos\sigma = 1$ and $q = \cos^2 \pi/8$. On the other hand, symmetrical distributions can be obtained by choosing $\cos\sigma = 0$ and $q = 1/2$ (dashed line). These are not, however, unique choices, and other are possible.

with the coin initial state prepared in the superposition $(|\uparrow\rangle + |\downarrow\rangle)/\sqrt{2}$, we would obtain a symmetrical quantum walk. However, in Sec. 3.2 we also saw that the choice of the initial coin state in the Hadamard walk on the line determines the final distribution of the walker (see Fig. 3.5) – from a right-skewed distribution through a symmetrical distribution to a left-skewed distribution, even though the Hadamard coin itself is *unbiased*. Under the unitary evolution, the information about the choice of the initial state is thus transferred to the final state (before the measurement).

Is the choice of initial coin state general enough, or do different coin operator choices result in qualitatively different behavior? The authors of [27] considered this question using similar techniques as in Sec. 3.2, where we switched into the Fourier basis. They found out that the full range of quantum walk behavior (possible with an unbiased coin) on the line can be achieved within the Hadamard walk, combined with choosing the initial coin state.

In these cases the shape of distributions is determined by two factors — by interference, as is the case for the symmetric Hadamard walk where the initial state is real-valued, or by a combination of probabilities from two mirror-image orthogonal components. Two examples of the latter case are the symmetric Hadamard walk with the initial state (3.16) and the balanced coin walk (3.22).

Although the choice of an *unbiased*[20] coin operator was shown to have only little importance on the line, its impact is significant in other cases. When considering the Hadamard walk on a

---

[20]All unbiased coins are equivalent, according to [27]. For biased coins, we do not expect to be able to reproduce a symmetric evolution.

cycle, the (limiting[21]) distribution depends on the parity of the number of nodes. On the other hand, it can be shown [27] that the limiting distribution can be also modified by choosing a different coin and leaving the number of nodes constant.

3.4.2   General Coins

Moving away from 1D, we now turn our attention to walks on general graphs with d-dimensional vertices, where the coin is described by a $d \times d$ unitary matrix. Not only is the coin space larger, the choice of the coin operator starts to make a difference. For a coined quantum walk on a two-dimensional lattice, a variety of interesting coins were discovered numerically in [27]. Each of those (combined with the choice of initial state) affects the characteristics of the walk. The difference between them is mainly in the extent to which the coin can affect them. For further details, see also Ref. [28].

We will now investigate several types of coins commonly used for walks on d-regular[22] graphs. First, a generalization of the Hadamard coin, then a family of unbiased coins related to the Fourier transform, and finally some symmetric coins.

In two-dimensional coin space, we have looked at the Hadamard coin (which is capable of reproducing all possible behaviors coming from unbiased coins). On l-dimensional lattices ($d = 2^l$), the Hadamard coin can be generalized to $C_{WH} = H \otimes H \otimes \ldots \otimes H$. Also called *Walsh-Hadamard operator*, it can be rewritten as

$$C_{WH} = \frac{1}{\sqrt{2^l}} \sum_{k=0}^{2^l-1} \sum_{m=0}^{2^l-1} (-1)^{\bar{k} \odot \bar{m}} |k\rangle \langle m|, \tag{3.23}$$

where $\bar{k} \odot \bar{m}$ is the parity of the bitwise dot product of l-bit binary strings representing k and m,

$$\bar{k} \odot \bar{m} = \left(\sum_{j=0}^{l-1} k_j m_j \right) \mathrm{mod}\, 2.$$

For some applications, we would like the coin operator to be *unbiased*, i.e. producing equal splitting of the probability of the walker into target vertices. One example of such a coin is the discrete-Fourier-transform (DFT) coin

$$C_{DFT} = \frac{1}{\sqrt{d}} \sum_{\mu=0}^{d-1} \sum_{\nu=0}^{d-1} \exp \frac{2\pi i \mu \nu}{d} |\nu\rangle\langle\mu|,$$

which as a matrix has the form

$$C_{DFT} = \frac{1}{\sqrt{d}} \begin{bmatrix} 1 & 1 & 1 & 1 & \ldots \\ 1 & e^{i\omega} & e^{2i\omega} & e^{3i\omega} & \ldots \\ 1 & e^{2i\omega} & e^{4i\omega} & e^{6i\omega} & \ldots \\ 1 & e^{3i\omega} & e^{6i\omega} & e^{9i\omega} & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

---

[21] Will be defined in Eq. (3.27).

[22] Note that for general d-regular graphs which are not symmetric, choosing the coin operator involves taking the directional information into account. For graphs that are not regular, finding a proper coin is even more problematic.

Although unbiased (all elements of the matrix have same magnitude), this coin is *asymmetric* — different directions of the walker are treated differently, acquiring various phases (all elements of the matrix do not have the same amplitude). Note that for $d = 2$ (corresponding to each vertex having two neighbors), the Fourier transform *is* the Hadamard transform, so we have $C_{DFT}|_{d=2} = H$.

Symmetry of the coin is also often a desirable property. In general, such coins are written as

$$C = \begin{bmatrix} -r & t & t & \dots \\ t & -r & t & \dots \\ t & t & -r & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

The coefficients t and r need to be chosen so that C is unitary, obeying the conditions

$$|r|^2 + (d-1)|t|^2 = 1,$$
$$-r^*t - t^*r + (d-2)|t|^2 = 0.$$

A specific choice of parameters, biased for all $d \neq 4$, treating the "return" direction differently from all others, is

$$t = \frac{2}{d}, \qquad r = 1 - t = 1 - \frac{2}{d}. \tag{3.24}$$

The coin C_G with these coefficients is the asymmetrical coin farthest from the identity [29]. It was used by Grover in [30] in his celebrated quantum algorithm for unstructured search[23] (see Sec. 4.1). The role of C_G is a *reflection about the average* state $|s\rangle = \frac{1}{\sqrt{d}} \sum_{j=0}^{d-1} |j\rangle$. We can see it by observing that

$$C_G = -\mathbb{I} + \frac{2}{d} \sum_{m,n=0}^{d-1} |m\rangle \langle n| = -\mathbb{I} + 2|s\rangle \langle s|, \tag{3.25}$$

acting on an input state $|\psi\rangle$ as $C_G |\psi\rangle = -|\psi\rangle + 2\langle s|\psi\rangle |\psi\rangle = \langle s|\psi\rangle |\psi\rangle - (|\psi\rangle - \langle s|\psi\rangle |\psi\rangle)$, which is the aforementioned reflection of $|\psi\rangle$ about the average state $|s\rangle$. Besides the algorithm for unstructured search, Grover's coin is used in many other search and walk-based algorithms, such as [23]. For further reference, note that when the Hilbert space dimension is a power of 2 (i.e. $d = 2^l$ for some l), we can express C_G as

$$C_G = -C_{WH} R_0 C_{WH}^\dagger = -\mathbb{I} + 2|s\rangle\langle s|$$

where $R_0 = \mathbb{I} - 2|0\rangle \langle 0|$ is the reflection about the state $|0\rangle^{\otimes l}$ and C_{WH} is the Walsh-Hadamard transformation defined by (3.23). The coin we just defined is the same unitary as used in Sec. 4.1 but you can find its use in many other places as it plays a vital role in (quantum-walk) searches of all kinds.

---

[23]Grover's search is the optimal quantum algorithm for the unstructured search problem: given a quantum oracle $R_w = \mathbb{I} - 2|w\rangle \langle w|$ acting as $-\mathbb{I}$ on the target state $|w\rangle$ and as an identity on all other states, the goal is to prepare the marked state $|w\rangle$ with the smallest amount of calls to the oracle R_w. For details, see Sec. 4.1.

If we are restricted to lattice of dimension D one obtains a unique feature — where other graphs have for each coin state c only one direction, lattices have two. So when in Grover coin Eq. (3.25) a large part of the walker was returned to the previous position, on lattices we can modify this coin to move the walker further. When we employ the language of coins, this means, that each coin state is given as $|c, \pm\rangle$ where c determines the direction and the sign determines whether we go "up" or "down". Grover coin as defined previously would for example change the state $|c, +\rangle$ to

$$C_G|c, +\rangle = -r|c, -\rangle + t|c, +\rangle + t\sum_{e \neq c}(|e, +\rangle + |e, -\rangle)$$

with t and r being given by Eq. (3.24) with $d = 2D$. The *flip-flop* coin C_{ff} that repulses the walker from previous state does not change the direction of the walker to head back, but keeps it the same. That means, that on the state $|c, +\rangle$ it acts as follows:

$$C_{ff}|c, +\rangle = -r|c, +\rangle + t|c, -\rangle + t\sum_{e \neq c}(|e, +\rangle + |e, -\rangle) \qquad (3.26)$$

Such coin was used in Ref. [43] to speed-up the search on lattices. See also Sec. 4.6.

3.5   Characteristics of Quantum Walks

The properties of quantum walks given in previous sections suggest that quantum walks could be useful in devising efficient algorithms. From the algorithmic point of view, the properties we have seen so far are quite vague and so to get a more direct feeling about the usefulness of quantum walks, we now turn to some more elaborate properties. We will present them here for discrete-time quantum walks and compare them with their classical analogues introduced in Sec 2.2. Furthermore, based on Sec. 3.3.1, the readers should be able to translate these concepts also to the language of scattering quantum walks.

3.5.1   Limiting Distribution and Mixing Time

The *mixing time* is a quantity that shall give us qualitatively the same information as the aforementioned Shannon entropy given by (3.17), yet taken from a different perspective. In the classical case for connected, non-bipartite graphs, the distribution of random walk always converges (see p. 612 in Sec. 2.2) to the stationary distribution independent of the initial state. Hence, it is possible to define a *mixing time* which tells us the minimal time after which the distribution is ϵ-close to the stationary one as in (2.8). In the quantum case, such a definition is not straightforward, because in general the $m \to \infty$ limits of $U^m|\psi_0\rangle$ and $p^m(x)$ do not exist. Nevertheless, if we average the distribution over time, in the limit of infinite upper bound on time it does converge to a probability distribution which can be evaluated.

Generalising (3.7), the distribution of the walker position x after m steps, given the initial state is $|\psi_0\rangle$, is given by

$$p^m(x) = \sum_{c \in \Xi} |(\langle x| \otimes \langle c|)U^m|\psi_0\rangle|^2.$$

Formally, then, the *time-averaged distribution* $\bar{p}^T(x)$ for the walker starting in state $|\psi_0\rangle$ is defined as the average over all distributions up to time T

$$\bar{p}^T(x) = \frac{1}{T} \sum_{t=0}^{T-1} p^m(x)$$

The interpretation of this quantity is simple. We start the walk in the state $|\psi_0\rangle$ and let it evolve for time m uniformly chosen from the set $\{0, 1, \ldots, T-1\}$. Then the probability of finding the quantum walker at position x is given by $\bar{p}^T(x)$. The *limiting distribution* (for $T \to \infty$) then is

$$\pi(x) \equiv \bar{p}^\infty(x) = \lim_{T \to \infty} \bar{p}^T(x). \tag{3.27}$$

For finding the limiting distribution we refer the reader to [15]. Here we present only a small portion of their results.

Theorem 2 *For an initial state $|\psi_0\rangle = \sum_j a_j|\phi_j\rangle$, the limiting distribution is*

$$\pi(x) = \sum_{i,j:\, \lambda_i = \lambda_j} \sum_c a_i a_j^* ((\langle x| \otimes \langle c|)|\phi_i\rangle\langle\phi_j|(|x\rangle \otimes |c\rangle)), \tag{3.28}$$

where c is the coin state.

If all the eigenvalues are distinct, (3.28) simplifies to

$$\pi(x) = \sum_j |a_j|^2 p_j,$$

where $p_j = \sum_c |(\langle x| \otimes \langle c|)|\phi_j\rangle|^2$ is the probability to measure the initial state in the eigenstate $|\phi_j\rangle$. We may notice that unlike in the classical case, where the limiting distribution did not depend on the initial state, this does not hold anymore in the general quantum case. One of the examples where one still gets a limiting distribution independent of the initial state is for a walk on the Cayley graph of an Abelian group such that all of the eigenvalues are distinct.

Example 7 For the Hadamard walk on a cycle with N nodes (vertices), the limiting distribution depends on the parity of the number of nodes N. The distribution is uniform only if N is odd [15].

We are ready to define the *mixing time* for a discrete-time quantum walk. It is the smallest time after which the time averaged distribution is ϵ-close to the limiting distribution:

$$\mathcal{M}_\epsilon^q = \min\{T : \forall m \geq T, |\pi - \bar{p}^m|_{tvd} \leq \epsilon\}, \tag{3.29}$$

where $|\cdot|_{tvd}$ is the total variational distance of distributions π and $\bar{p}^m$ defined by (2.9). The quantum mixing time, defined in (3.29), tells us the same thing as mixing time $\mathcal{M}_\epsilon^c$ in the classical case defined in (2.8). The difference is that while in the classical case we use the actual distribution to determine $\mathcal{M}_\epsilon^c$, in the quantum case we use only the time averaged distribution.

We point out yet another difference between classical and quantum walks which concerns eigenvalues. In the classical case, the difference between the two largest eigenvalues governs the mixing time. In the quantum case where the eigenvalues of the quantum walk unitary step all have amplitude one, we find a different relationship between the elements of the spectrum and the mixing time.

Theorem 3 *For any initial state $|\psi_0\rangle = \sum_j a_j |\phi_j\rangle$ the total variation distance between the average probability distribution and the limiting distribution satisfies*

$$\left|\pi - \bar{p}^T\right|_{tvd} \leq 2 \sum_{i,j:\ \lambda_i \neq \lambda_j} \frac{|a_i|^2}{T|\lambda_i - \lambda_j|}.$$

3.5.2 Hitting Time

Another important quantity is the *hitting time*. In the classical case, it is the time when we can first observe the particle at a given position (2.6). For a quantum walk, the measurement is destructive, not allowing us to meaningfully define the hitting time in the same manner. In [31], two ways of defining a hitting time were proposed. One of them is called the *one-shot hitting time* connected with some probability p and time T. We say, that a quantum walk has a (T, p) one-shot hitting time between positions $|k\rangle$ and $|j\rangle$, if $\left|\langle j|U^T|k\rangle\right|^2 \geq p$ irrespective[24] of the coin degree. This quantity tells us that when p is a "reasonable" number ($0 < p \leq 1$), we need only $\mathrm{poly}(T, p)$ number of steps to reach the vertex j from k. In fact, the usual definition (2.6) is not possible, the an additional parameter T needs to be specified. However, one can use these two values to devise a similar quantity given by (2.7), corresponding to the average number of steps needed when repeating the experiment (each time running it for T steps) until the desired position j is hit.

In the second case, we perform a measurement with two projectors $\Pi_0 = \Pi(j) = |j\rangle \langle j| \otimes \mathbb{I}$ and $\Pi_1 = \mathbb{I} - \Pi_0$ after each step, determining whether vertex j is "hit". If Π_0 is measured, the process is stopped, otherwise another step is applied. The action of this operation on a general state ρ can be written as $\Phi_q(\rho) = \Pi_q U \rho U^\dagger \Pi_q$ for $q = 0, 1$. This definition allows us to express the probability of reaching the vertex j in the m-th step and no sooner, provided the initial state is ρ_0, as

$$p^m(j) = \mathrm{tr}[(\Phi_0 \Phi_1^{m-1})(\rho_0)]. \tag{3.30}$$

Exercise 8 The expression (3.30) might not be intuitive for everybody. The process of the measurement, instead of using trace-decreasing operations Φ_q, can be also defined as follows. In each step, a projective measurement given by operators Π_0 and Π_1 is performed. If we find the walker at position j, the process is stopped. Otherwise, the state ρ changes to

$$\rho' = \frac{\Pi_1 U \rho U^\dagger \Pi_1}{\mathrm{tr}[\Pi_1 U \rho U^\dagger]}.$$

The resulting (normalized) state is then again evolved and a measurement is performed. Show, that this process hits the vertex j in the m-th step and no sooner with probability given by (3.30).

We could readily use the expression for the m-th step probability of reaching j (3.30)) and plug it into the classical definition (2.6). However, the usual choice (in order to correspond to the first definition above) is slightly different. The *concurrent hitting time* for a given probability p

---

[24]In the end state, usually any coin state is accepted, while in the initial state the choice of the coin usually respects the topology and symmetry of the graph.

is the time T for which the process stops with probability higher than p at a time $m \leq T$. This can be written as

$$h_c^q(j) = \min \left\{ T : \sum_{m=1}^{T} p^m(j) \geq p \right\}. \tag{3.31}$$

Example 8 Using the quantities defined above, it was shown [31] that a quantum walk on a hypercube has an exponentially faster hitting time (when traversing from one end to another – see Fig. 2.2) than the corresponding classical random walk, even though the mixing time might be exponentially large in the number of layers. As was pointed out in [33], however, this does not mean that every classical process is exponentially slower – there is a classical algorithm capable of traversing the m-hypercube in time polynomial in m, which is of the same efficiency as that for the quantum walk – see Exercise 2 in Sec. 2.3.1. For a graph-traversing quantum walk algorithm provably exponentially faster than a classical one, see Sec. 2.3.1 and 6.1.2.

3.5.3 Absorbing Boundary

We now turn our attention to the Hadamard quantum walk on a line with an *absorbing boundary* and compare the results with the classical ones from Sec. 2.2. In the classical case, we looked at whether a walker starting at position 1 eventually reaches position 0, and found it is so. Following [16], we now want to compute the probability with which the quantum walker starting in the state $|\psi_0\rangle = |1\rangle \otimes |\uparrow\rangle$ hits an absorbing boundary at position 0 (after an arbitrary number of steps). Comparing with (3.31) for the concurrent hitting time, we can also say that we are looking for the smallest p (which we denote p_{10}) for which the hitting time is infinite[25].

The absorbing boundary is a repeated projective measurement with $\Pi_0 = |0\rangle \langle 0| \otimes \mathbb{I}$ and $\Pi_1 = \mathbb{I} - \Pi_0$, telling us whether the walker reached 0 in a given step or not. To find the overall probability of this eventually happening, we will not employ the Fourier transform technique we used before, because we now only look at a semi-infinite line comprised of non-negative positions for the walker. Instead, we shall use the technique of counting paths.

Exercise 9 Show that this type of measurement process does not allow the walker to tunnel through the position 0. i.e. if she starts at positive position that she cannot pass to negative positions.

Let us start with a short analysis of the problem. The walker starts at position 1 and we want to know, what the amplitude is for him to be at position 0 after m steps – let us denote this amplitude $a_{10}(m)$. It can be written as the sum of amplitudes corresponding to all possible classical paths (not going through 0). Each classical path is described by an m-tuple $(q_1, q_2, \ldots, q_m)$ with $q_j \in \{\uparrow, \downarrow\}$ standing for the direction the walker went in the j-th step. The amplitude for each path is $\pm 2^{-m/2}$, as the Hadamard coin (3.4) has only the coefficients $\pm 1/\sqrt{2}$. We notice that the coefficient can be negative only in the case we go left in two consecutive steps (let us call this event a *doublet*). In each such case, the amplitude acquires a minus sign. If the path has

---

[25]This means there is no number of steps that would guarantee the walker has reached position 0 with probability greater than p_{10}

even number of doublets, the final amplitude for the path is positive and when the path has odd number of doublets, the final amplitude for the path is negative.

We denote the set of all paths of length m with even number of doublets as A_m^+ and the set of all paths of length m with odd number of doublets as A_m^-. The amplitude $a_{10}(m)$ can now be written as

$$a_{10}(m) = \sum_{p \in A_m^+} \frac{1}{2^{m/2}} - \sum_{p \in A_m^-} \frac{1}{2^{m/2}} = \frac{1}{2^{m/2}}(|A_m^+| - |A_m^-|),$$

where $|A|$ denotes the cardinality of set A. The probability for the walker to hit the boundary at all is now expressed as

$$p_{10} = \sum_{m=1}^{\infty} |a_{10}(m)|^2 = \sum_{m=1}^{\infty} \frac{1}{2^m}(|A_m^+| - |A_m^-|)^2. \tag{3.32}$$

In order to find the coefficients $|A_m^+| - |A_m^-|$, we construct the so called generating function[26] for these[27] coefficients,

$$f(x) = \sum_{m=1}^{\infty} (|A_m^+| - |A_m^-|)x^m.$$

If the walker were at position 2 and then got back to 1 by some path (not going to 0), the generating function would be the same. Now if the walker is at position 1, then moves right to position 2 and then back to 1 by some path, we find that the generating function for this process is $xf(x)$. By joining k such paths together we can obtain the generating function for all the paths that get to position 1 after leaving it exactly k-times, obtaining $[xf(x)]^k$.

There are two ways to get from position 1 to position 0. First, we could go left one step. Second, we could leave 1, move around and return to 1, all this k-times, and then finally take a step left to 0. When we enter 1 for the k-th time and then go left once more, the amplitude acquires an additional minus sign. Overall, we can write this in the following way,

$$f(x) = x - x \sum_{k=1}^{\infty} [xf(x)]^k = x - \frac{x^2 f(x)}{1 - xf(x)}.$$

---

[26] This is a different concept from the generating functions for the moments of distributions. Although this topic is very interesting from the mathematical point of view, for our purposes we only focus on a few points. For any sequence of numbers $\{a_j\}_{j=0}^{\infty}$ we can construct the power series at $x = 0$,

$$g(x) = \sum_{j=0}^{\infty} a_j x^j.$$

All we will need is that when we have two generating functions $f(x)$ and $g(x)$, and know the coefficients $\{a_j\}_{j=0}^{\infty}$ of $g(x)$, then if we can find a functional relation between $f(x)$ and $g(x)$, we can also find a relation between their coefficients $\{a_j\}_{j=0}^{\infty}$ and $\{b_j\}_{j=0}^{\infty}$.

[27] In this Section we will always consider coefficients of the type $|A_m^+| - |A_m^-|$ for various conditions, and thus we will use the notion of a generating function in a slightly abusive form just by saying it corresponds to some process instead of always explaining that it corresponds to the coefficients of the type $|A_m^+| - |A_m^-|$ for a given process.

Solving this quadratic equation for $f(x)$ yields

$$f(x) = \frac{1 + 2x^2 - \sqrt{1 + 4x^4}}{2x}.$$

By a simple comparison with (C.2), we find that the generating function $f(x)$ is connected with the generating function $c(x)$ for the Catalan numbers (see Appendix C) as

$$f(x) = x - x^3 c\left(-x^4\right) = x + \sum_{k=0}^{\infty} (-1)^{k+1} C_k x^{4k+3},$$

where C_k are the Catalan numbers,

$$C_k = \frac{1}{1+k} \binom{2k}{k}.$$

In other words, we have

$$|A_m^+| - |A_m^-| = \begin{cases} 1 & \text{for } m = 1, \\ (-1)^{k+1} C_k & \text{for } m = 4k+3, \\ 0 & \text{otherwise.} \end{cases}$$

This results in the expression for the probability of eventually hitting the boundary (3.32):

$$p_{10} = \frac{1}{2} + \frac{1}{8} \sum_{k=0}^{\infty} \frac{C_k^2}{2^{4k}} = \frac{1}{2} + \frac{1}{8} \left(\frac{16}{\pi} - 4\right) = \frac{2}{\pi},$$

using (3.33) and (C.4), obtained for example by employing Stirling's formula.

We found that $p_{10} = 2/\pi$ which is different from the random walk case for which the probability is 1 (see p. 611), meaning the classical random walker will eventually come to the position 0. We see again that interference of amplitudes instead of adding up probabilities plays a crucial role in quantum theory, leading to a different behavior of systems when compared to the classical case.

Exercise 10 Show that

$$\sum_{k=0}^{M} \frac{C_k^2}{2^{4k}} = (16M^3 + 36M^2 + 24M + 5) \frac{C_M^2}{2^{4M}} - 4, \tag{3.33}$$

using induction and (C.6) from Appendix C.

3.5.4   Quantum-to-classical Transition and Decoherence

The definition of discrete-time quantum walks is based mostly on an analogy with classical random walks. So fare, we have done no real attempt to make a canonical quantization of random walks – this will be done later in Chapter 5. However, we might think about the opposite processes. Consider now a quantum walk whose evolution is not unitary anymore, but some amount
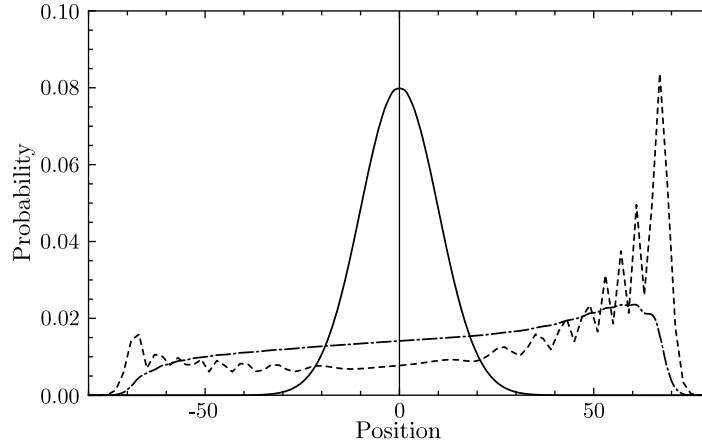
Fig. 3.6. Probability distributions for quantum walks on a line with additional decoherence, after 100 steps of evolution. A pure quantum walk with $p = 1.0$ (dashed line), a pure classical random walk $p = 0.0$ (solid line) and a non-unitary walk with $p = 0.96$ leading to a distribution with a maximal Shannon entropy.

of decoherence comes into play. Imagine the state unitarily evolves for one step, and then with some small probability $(1 - p)$ a projective measurement with projectors Π_x gets performed, transforming the state ρ into

$$\mathcal{E}(\rho) = pU\rho U^\dagger + (1 - p) \sum_x \Pi_x U \rho U^\dagger \Pi_x. \tag{3.34}$$

The projectors Π_x can be either tied to different coin states ($\Pi_\uparrow$, $\Pi_\downarrow$) or work in the position basis of the (coined) walker (when $\Pi_x = |x\rangle\langle x|$), or even in both of these (for a SQW, x would correspond to different directed edges). This prescription thus states on top of performing a quantum walk, occasionally we measure either the state of the coin, the position of the walker, or the edge state for a SQW, depending on the choice of projectors Π_x.

Let us look on the quantum walk on a line in the SQW model (see Sec. 3.3.1), concentrating on the Shannon entropy of the walks given by (3.17). Consider $\mathcal{N}$ to be the set of all edge-states and $(1 - p)$ the probability of performing a measurement after a step of the walk. It is clear, that $p = 1$ gives an unperturbed unitary evolution. On the other hand, for $p = 0$ the evolution is devoid of interference, as it results in a diagonal density matrix. In each step, no matter what the state of the coin might be, the walker moves to the right or left with equal probabilities. The measurements performed after every step thus give us a classically describable evolution, as any superposition that occurs in the coin evolution is broken by the measurement. Such a walk is governed by the same rules as the drunkard's walk. Therefore, in the extreme case of maximal decoherence, a quantum walk becomes classical. This is a conclusion that can be simply extrapolated into any type of graph and a quantum walk on it, with the high-decoherence limit resulting in a classical walk with a transition rule derivable from the corresponding coin operator.

The situation for general (intermediate) values of p now deserves our attention and is reviewed in [17]. In short, from Fig. 3.6 we see that by varying p, we smoothly get from a two-peaked pure quantum-walk distribution to the normal distribution. However, as the computation
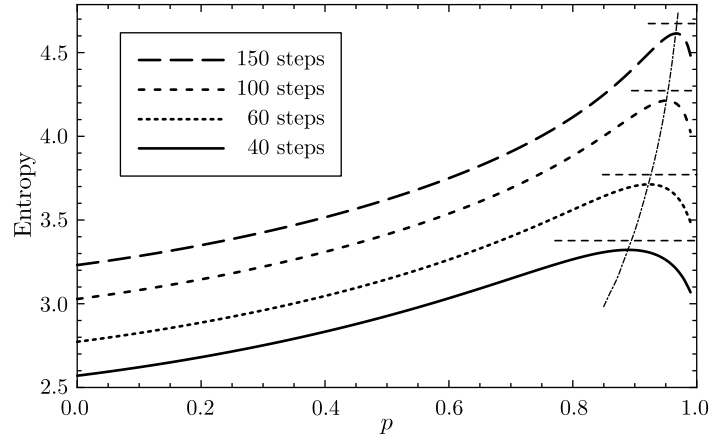
Fig. 3.7. Shannon entropy of a probability distribution coming from a quantum walk, depending on the unitarity rate p, for various numbers of steps taken. Dashed horizontal lines correspond to the maximal achievable entropy of a (uniform) distribution restricted to the position interval $[-x/\sqrt{2}; x/\sqrt{2}]$ after taking x steps. The thin dot-dashed line denotes the maximal achievable entropy for a given p. Obviously, as a quantum walk without decoherence does not revert to classical side, the maximum of the entropy for $p = 1$ goes to infinity.

of Shannon entropy in Fig. 3.7 suggests, an intermediate point in p gives us an almost uniform distribution, something far from both the purely quantum and purely classical behavior. The point in which Shannon entropy is maximal depends not only on the value of p, but also on the number of steps taken. Before reaching this point, the distribution spreads ballistically, while at larger times, the spreading slows down as the distribution approaches a normal distribution and the spreading becomes diffusive. Thus on long timescales, any amount of decoherence leads to the destruction of quantum interference effects. Note that in the limit $p \to 1$, the point of maximal entropy diverges in the number of steps and so the evolution remains (as is expected) ballistic all the time.

We encourage the reader to consult [17] for a deep review, considering also different types of measurement (showing that it makes a difference whether we measure the coin, position or both), different types of graphs (such as a hypercube or a cycle) and different types of decoherence as well. We conclude that although discrete time quantum walks are not quantized versions of classical random walks, they approach the behavior of classical random walks when undergoing decoherence at an increasing rate. One should keep in mind one additional thing. Even if the decoherence is maximal in both the position and the coin registers ($p = 0$ and we measure in every step), the usual unbiased random walk given by (2.2) is recovered only in special cases. The walker always remembers the direction it came from and so it treats all directions equally (as a random walk does) only if the coin is unbiased. Otherwise, the walk obtained by decoherence corresponds to a random walk with memory [20].

Exercise 11 We could also think about what happens to quantum walks under a different type of decoherence. In [20], every step of a SQW evolution is affected by a random phase-shift on every

edge-state. Thus, after evolving the state ρ by the step unitary U, another unitary $\Phi(\hat{\phi}) = \mathrm{diag}\,\mathrm{e}^{\mathrm{i}\hat{\phi}}$ is applied (with a random vector of phase-shifts $\hat{\phi}$). The "random" effectively means that such a mapping is described by

$$\mathcal{E}(\rho) = \int \pi(\hat{\phi})\Phi(\hat{\phi})U\rho U^{\dagger}[\Phi(\hat{\phi})]^{\dagger}\,\mathrm{d}\hat{\phi}\,,$$

where π is the distribution of phases, which we assume to be symmetric and $\hat{\phi}$-components uncorrelated, meaning that the choices of components are independent, i.e. $\pi(\hat{\phi}) = \prod_{x \in \mathcal{N}} \pi_0(\phi_x)$. Under this assumption, show that the state after one step of evolution with random phase flips is again described by (3.34) with p dependent on the parameters of noise, still being from interval $[0;1]$, with $p = 1$ standing for no noise and $p = 0$ for maximal noise described by the choice of phase-shifts sampled uniformly from whole interval $[-\pi;\pi]$.

3.6   Summary

In this section we have presented the simplest and most naïve transition from classical random walks to processes we call quantum walks. The definition of these discrete time quantum walks was made on analogy with their classical counterparts rather than by quantization. Nevertheless, there is a closer correspondence between them, as the classical random walk can be obtained from the quantum walk by adding decoherence – classical random walks are indeed the classical limits of discrete time coined quantum walks (possibly with memory).

On the other hand, quantum walks display a variety of differences from classical random walk, as their spreading is ballistic and not diffusive as is the case for classical random walks. This difference is the result of interference effects and leads to faster mixing times. This in turn can be used to obtain efficient algorithms. The most prominent algorithmic application of discrete-time quantum walks is for searches on graphs, where a quadratic speedup can be obtained if the "database" we search is unstructured (Grover's search on a complete graph), but also in many other cases. Although their immediate application doesn't seem apparent, they can in turn be used as subroutines in more complicated tasks such as k-subset finding, where they provide a provable speedup over the best possible classical algorithm.

The aim of this section was also to showcase the methods for analysis useful not only in this area – Fourier transforms (applicable on all Cayley graphs) and counting paths. We looked at the most common graphs: a line, a cycle, a hypercube. They were used to get the reader acquainted with the most usual properties studied in these types of problems – hitting time, mixing time, reaching an absorbing boundary or finding the (limiting) distribution. We then compared the obtained quantities to their classical counterparts for random walks. The only problem arising when defining these quantities is the different approach to measurement – whereas in the classical case a measurement does not change the state of the system, in the quantum case measurement destroys the state and leads to the collapse of the wave function. Furthermore, occasional measurement can be used to model decoherence that destroys the superpositions, the key ingredient of quantum walks.

4   Quantum Walks and Searches

In the previous Section we defined discrete-time quantum walks and explored some of their basic properties. Now it is time to show their applications in computational problems. As the title of the section suggests, a large part of the research in this area concerns applying quantum walks for searching. The mixing properties of quantum walks are behind this — fast mixing leads to a natural suggestion that these walks could perform searches faster than classical algorithms. This approach has led to several important quantum walk search algorithms with provable speedups over their classical counterparts.

The first quantum walk algorithm performing a search for a target vertex was given in Ref. [23] for a hypercube. Soon after that, several new algorithms for searching on various graphs emerged, utilizing the symmetric properties of the underlying graphs to analytically describe the unitary evolutions. The common theme resembles the algorithm for unstructured database search by Grover [30]. In fact, the problems of searches on graphs go beyond the framework of amplitude amplification method [37] which solves the problem by showing that its evolution is restricted to a two-dimensional subspace of the Hilbert space. On one hand one can use results of Ref. [43] that approximate the evolution as a restriction to a two-dimensional Hilbert space or one can find higher-dimensional restrictions and describe the evolution within this subspace. This method is described in Sec. 4.3 and is applicable for highly symmetrical problems.

Research in the area of quantum walk searches progressed also with Ambainis' algorithm for element distinctness, utilizing quantum walks on a graph with a more involved structure [38] (the vertices of this graph are sets of vertices of the original graph). This specific algorithm was later generalized to the problem of subset (e.g. triangle) finding [39, 40]. Although a general lower bound is still not known in the general case, this approach is the best known up to date. Moreover, it can be modified for the task of solving several other problems such as verification of matrix products [41] or testing the commutativity of a black-box group [42].

4.1   Grover Search

We begin this Section with Grover's search [30], the algorithm whose power will be used throughout the whole section for comparison and as a point of reference.

In 1997, Lov K. Grover introduced [30] a quantum search algorithm for a marked (target) element from an unstructured database. The algorithm is provably quadratically faster than any classical algorithm for the given problem, also in the generalized case soon described in [47]. There exists also a fixed point "version" of Grover's search — see Ref. [53, 54] and Appendix D. This quadratic speedup comes in the number of oracle calls, where one oracle call gives us only the information whether the questioned-about element is the target one or not. The oracle model is key for Grover's search which serves as a base for the comparison of different algorithms. Therefore, we will now provide a short introduction to oracles and after that we shall analyze Grover's search.

4.1.1   Oracles and Searches

In the previous Sections, we had a few opportunities to notice oracles but left them unexplained. Here we provide some basic information on this topic that concerns Grover's search and further

applications.

Within a classical computation, the use of the so-called *oracles* (also called black-boxes), is quite abundant. The oracle is a device that performs a specialized task and the number of its calls is used for comparison of efficiencies of algorithms (their query complexity). The algorithmic efficiency is then expressed in the number of calls of such oracle. In the quantum case we can follow a similar path, where we compare different algorithms with respect to the number of oracle calls, where the oracle can now be quantum (i.e. a unitary operation such as a reflection about a vector characteristic for the black box but unknown to us).

A very common task in both classical and quantum setting is a search for a marked/desired element. Here the oracle serves as a device telling us whether the element we are examining is the one we are searching for (and nothing more). This task may be justified in the following way. If we consider a database (structured or non-structured), there are certain kinds of queries for which the database may be viewed as unstructured. For example a phone book (yellow pages) is a database designed so that if you know a name, it is easy to find the corresponding phone number. However, having only the a phone number, it is a tedious task to find the corresponding name, since the phone numbers in the phone book are positioned practically in a random fashion.

Following the previous discussion, we may also construct an oracle that tells us an answer for a query – given an index of an element from a database, the oracle tells us whether the element accommodates the given query or not. Sticking to the yellow pages analogy, the oracle tells us whether the k-th record (name) corresponds to the number we are looking for. The analogy is suitable also to show that such oracle doesn't have to be something magical – it is a relatively easy task to find the k-th name in a phone book – a well structured database from this point of view.

Let us now define an oracle in a more abstract fashion, denoting the set of all possible choices of queried elements as $\mathcal{N}$ and the subset of this set that accommodates a given query as $\mathcal{K} \subset \mathcal{N}$. The oracle[28] then is a function $f_{\mathcal{K}}(x)$ defined for all $x \in \mathcal{N}$ such that

$$f_{\mathcal{K}}(x) = \begin{cases} 1, & \text{if } x \in \mathcal{K}, \\ 0, & \text{if } x \in \mathcal{N} \setminus \mathcal{K}. \end{cases} \tag{4.1}$$

The oracle function $f_{\mathcal{K}}$ is determined by the set $\mathcal{K}$ (*target* or *special set*) and it is not necessary to state this fact in the function definition, yet it will be helpful on occasions. For practical purposes, if $\mathcal{K}$ contains only one element k, we will write it as $f_{\mathcal{K}}(x) \equiv f_k(x)$.

In a quantum world, we can define a unitary quantum oracle based upon a classical one $f_{\mathcal{K}}$, following the constructions in [47]. Given the function $f_{\mathcal{K}}$, we can consider a unitary controlled operator $\mathcal{C}V_{\mathcal{K}}$ defined for a bipartite (two-register) system,

$$\mathcal{C}V_{\mathcal{K}} : |x\rangle \otimes |m\rangle \mapsto |x\rangle \otimes |m \oplus f_{\mathcal{K}}(x)\rangle. \tag{4.2}$$

We can think of the first register as providing a query (asking about the element x), while the second subsystem is a qubit to whose value we add (in binary) the result of the oracle function $f_{\mathcal{K}}$ evaluated on a state of the first subsystem. This second register is needed for reversibility, as we would like the quantum oracle to be unitary. Such a quantum oracle, no matter what its actual

---

[28]In general, oracle may be any function. For our purposes it will be enough to restrict ourselves to boolean functions.

implementation is, is widely used to demonstrate the difference in query complexity between classical and quantum algorithms.

Observe that the quantum oracle we just defined is a unitary operation, and thus can also act on superpositions. However, if we query it classically (i.e. if input states are of the form $|x\rangle \otimes |0\rangle$ with $x \in \mathcal{N}$) and measure the result on the second register afterwards, we always get the result of $f_\mathcal{K}(x)$. Thus, in this classical-like usage, this quantum oracle does not give us any extra power for finding an element from $\mathcal{K}$. On the other hand, as we will show soon, quantum properties such as interference help us enhance the efficiency of search algorithms – the only way to achieve a speedup is to use a quantum oracle with quantum inputs.

Finally, there is one interesting feature of the quantum oracle from (4.2). When we intialize the second register to

$$|m\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \equiv |-\rangle,$$

we find that

$$\mathcal{C}V_\mathcal{K}|x\rangle \otimes |-\rangle = (-1)^{f_\mathcal{K}(x)}|x\rangle \otimes |-\rangle.$$

Surprisingly, when the second register is intialized as $|-\rangle$, after the application of $\mathcal{C}V_\mathcal{K}$ it does not change and thus is not needed in further mathematical considerations. Therefore, we may just as well use

$$R_\mathcal{K} : |x\rangle \mapsto (-1)^{f_\mathcal{K}(x)}|x\rangle$$

as our quantum oracle operation. The operator $R_\mathcal{K}$ can be further written as

$$R_\mathcal{K} = \mathbb{I} - 2 \sum_{j \in \mathcal{K}} |j\rangle\langle j|, \tag{4.3}$$

flipping the sign of the basis states from $\mathcal{K}$ and preserving the rest – it is a *conditional phase-flip*.

As the searches we will study are restricted to graph structures it is also necessary for similar tasks to consider "graph" oracles that will be later used. Besides mentioned oracle that marks some vertices we may consider also oracles that give us information about graph structure. We will find use of such oracles later as well. The walk on graph can be viewd also as a query problem, where walker, being on position j queries the *neighbor* oracle to tell him possible neighbors of j. Another type of oracle might be called *edge* oracle as it gives just information whether some edge jk exists — the information is provided in a similar way as in the case of vertex-marking oracle from Eg. (4.2) with the difference of taking two vertices on input.

4.1.2   Grover's Algorithm

Let us now present the optimal quantum algorithm for unstructured search. Having sets $\mathcal{N}$ and $\mathcal{K}$ as defined in the previous section, with $N = |\mathcal{N}|$ and $k = |\mathcal{K}|$, our goal is to find a state from the marked set $\mathcal{K}$. We initialize our system in the equal superposition of all states of the computational basis

$$|\psi_0\rangle = |s\rangle = \frac{1}{\sqrt{N}} \sum_{j \in \mathcal{N}} |j\rangle. \tag{4.4}$$
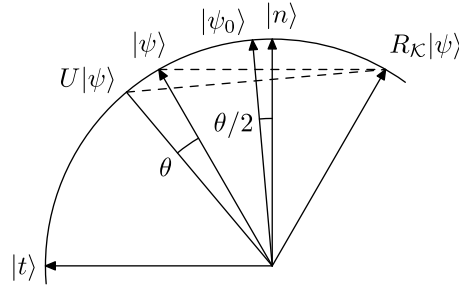
Fig. 4.1. The Grover algorithm performs a rotation of the state vector $|\psi\rangle$ by angle θ by first reflecting the vector around $|n\rangle$ and then around $|\psi_0\rangle$.

We define the unitary step operator

$$U = C_G R_{\mathcal{K}}, \tag{4.5}$$

where

$$C_G = -\mathbb{I} + 2|s\rangle\langle s| = -R_s \tag{4.6}$$

is an *inversion about the average* introduced[29] in Sec. 3.4, and the operator[30] $R_{\mathcal{K}}$ is defined in 4.3. Defining the states

$$|t\rangle = \frac{1}{\sqrt{k}} \sum_{j \in \mathcal{K}} |j\rangle, \qquad |n\rangle = \frac{1}{\sqrt{N-k}} \sum_{j \in \mathcal{N}\setminus\mathcal{K}} |j\rangle,$$

simplifies the analysis greatly, as the application of U (4.5) on a state from the subspace spanned by these two vectors leaves the state in this subspace. In particular, it performs a rotation (see also Fig. 4.1)

$$\begin{aligned} U|n\rangle &= \cos\theta|n\rangle + \sin\theta|t\rangle, \\ U|t\rangle &= -\sin\theta|n\rangle + \cos\theta|t\rangle, \end{aligned}$$

where

$$\cos\theta = \frac{N-2k}{N}. \tag{4.7}$$

Moreover, the uniform superposition initial state $|\psi_0\rangle$ from (4.4) is also a superposition of the $|t\rangle$ and $|n\rangle$ vectors:

$$|\psi_0\rangle = |s\rangle = \sin\frac{\theta}{2}|t\rangle + \cos\frac{\theta}{2}|n\rangle.$$

---

[29]There, we also show how to implement it using the Walsh-Hadamard operation and a reflection about the all-zero state $|0\cdots0\rangle$.

[30]Note, that $R_{\{k\}} \equiv R_k$, which also justifies Eq. (4.6).

Initialize state of the system according to Eq. (4.4) and for $\tilde{m}$ steps [given by Eq. (4.9)] repeat:

1. apply the oracle $R_\mathcal{K}$ to the state

2. perform inversion about average C_G on the system

Measure the system in computational basis; if element from $\mathcal{K}$ is not found repeat the algorithm

Fig. 4.2. Grover algorithm which searches for oracle-selected elements.

As a consequence, repeated application of U rotates the initial state towards the state $|t\rangle$ as

$$|\psi_m\rangle \equiv U^m|\psi_0\rangle = \sin(2m+1)\frac{\theta}{2}|t\rangle + \cos(2m+1)\frac{\theta}{2}|n\rangle. \qquad (4.8)$$

Thus, if we pick the number of applications of U to satisfy $(2\tilde{m}+1)\theta/2 = \pi/2$, we get $U^{\tilde{m}}|\psi_0\rangle = |t\rangle$ and a subsequent measurement of the system gives us an element from $\mathcal{K}$. For $k \ll N$, this happens for

$$\tilde{m} \simeq \frac{\pi}{4}\sqrt{\frac{N}{k}}. \qquad (4.9)$$

This number of steps is quadratically smaller than the corresponding number of steps we would need when using any classical algorithm where it is $O(N/k)$.

Note that it is important for this algorithm (see Fig. 4.2) to start in the equal superposition and to know the number of searched elements k in advance. Any deviation from the initial state (4.4) decreases the probability of successfully identifying the target elements. As a rule of thumb we can say that in order to obtain high enough probabilities, one has to start in a highly superposed initial state (in computational basis). On the other hand, there exist ways to modify the algorithm if the number of searched-for elements is unknown, e.g. by running the algorithm with expecting large k and progressively dividing the expected k after each unsuccessful run.

The Grover algorithm is a special instance of a more general class of amplitude amplification method introduced in Ref. [37]. The algorithm presented there amplifies amplitude a of target state obtainable by quantum algorithm $\mathcal{A}$. In the classical approach one would measure the system after applying algorithm $\mathcal{A}$ and obtain target state with probability $|a|^2$. So to find the target state one would have to repeat algorithm $\mathcal{A}$ for roughly $|a|^{-2}$ times. Approach of the amplitude amplification method repeats the algorithm $\mathcal{A}$ (adding some intermediate steps) for only $O(|a|^{-1})$ times with measurement process only at the very end to obtain the target with constant probability close to 1.

Exercise 12 A nice conclusion drawn from the previous result, often exploited in experiments verifying the functionality of Grover's search, is obtained by setting $k = \frac{N}{4}$ (meaning that one fourth of the elements from the set $\mathcal{N}$ are target elements). Show that in this case a single application of the unitary U (4.5) is needed to reach a target state, as opposed to two oracle queries one needs to make (on average) in the classical case. In particular, this means we can find a single marked element in a list of four by a single application of U (meaning using only a single call to the oracle).

---

1. Initialize state of the system according to Eq. (4.13)

2. for $\tilde{m}$ steps [given by Eq. (4.17)] apply U

3. measure the position (edge) of the walker; if the resulting edge does not have any target end vertex from $\mathcal{K}$, restart the algorithm

---

Fig. 4.3. Searches on graphs usually start in highly superposed initial state in the position basis. After $\tilde{m}$ iteration the walker is with high probability located at an edge connected to (at least) one target vertex. This algorithm is given for particular case of a search on the complete graph.

Example 9 Considering different types of oracles defined at the end of Sec. 4.1.1 one can notice that the edge oracle can be used together with Grover's search to construct neighbor oracle. If the degree of the graph is much smaller than the number of vertices the time expense of such construction is $O(\sqrt{N})$ as Grover's search has to look through N vertices of the graph to determine which of them are neighbors of given vertex.

4.2 Searches on Graphs

One of the first attempts to perform a search with quantum walks used a discrete time quantum walk on a hypercube, with a single type of coin (the Grover coin) for all vertices except the marked one, which had it perturbed to minus identity (effectively acting as minus the Grover coin due to the symmetry of the graph). This work by Shenvi, Kempe and Whaley [23] showed the possibility of quantum speedups in quantum walks on graphs, compared to classical search. The analysis of the hypercube search problem is similar to the one we used for Grover's search [30] in the previous Section, yet the dimension of the subspace into which the evolution is restricted to is larger than the two dimensions of amplitude amplification method of Ref. [43]. In this way, the approach was unique at the time.

Sarching for marked elements on a graph is the focus of many other works. Ambainis, Kempe and Rivosh [43] , also considered a search on a complete graph with loops, where they found another incarnation of Grover's search. More importantly, they studied searches on regular lattices and found an algorithm with a quadratic speedup over classical search. In the field of continuous quantum walks (which will be introduced in Sec. 6) Childs and Goldstone in Ref. [44] also proposed a search on a complete graph with loops and obtained similar results. Again, more importantly, the dialogue between continuous and discrete-time quantum walks inspired the quadratically faster (than classical) continuous-time quantum walk search algorithm on regular lattices by Childs and Goldstone [80].

We will now look at scattering quantum walks, following the definitions in Sec. 3.3.1. We will combine this formalism with searches on graphs, looking at the general algorithm given in Fig. 4.3 and compare it to Grover's search (see Fig. 4.2). In particular we will follow [45], studying scattering quantum walk search on the (symmetric) complete graph. We will call the vertices corresponding to the desired elements from set $\mathcal{K}$ *targets*, while the rest of the (unmarked) vertices will be called *normal*. Our goal again is to to find at least one element from set $\mathcal{K}$ by identifying some target vertex.

Considering a vertex l, let $\Gamma(l)$ be the set of vertices connected to l by an edge, and if $k \in \Gamma(l)$, let $\Gamma(l; k)$ be the set of vertices connected to l by an edge but excluding k. The local unbiased unitary evolutions (coins) corresponding to both the normal and target vertices act as follows

$$U^{(l)}|k, l\rangle = -r^{(l)}|l, k\rangle + t^{(l)} \sum_{m \in \Gamma(l;k)} |l, m\rangle, \tag{4.10a}$$

where $r^{(l)}$ and $t^{(l)}$ are reflection and transmission coefficients to be chosen in such way that $U^{(l)}$ is unitary. For normal (unmarked) vertices, our choice is the Grover coin (3.24) with

$$t^{(l)} = \frac{2}{|\Gamma(l)|}, \qquad r^{(l)} = 1 - t^{(l)}. \tag{4.10b}$$

The targed vertices are marked by a special coin, where we choose

$$r^{(l)} = -e^{i\phi}, \qquad t^{(l)} = 0, \tag{4.10c}$$

with a general phase-shift $e^{i\phi}$ (meaning a walker going into a target vertex is reflected completely, gaining some phase shift in the process). Here $|\Gamma(l)| = d(l)$ is the degree of vertex l, i.e. the number of vertices in the set $\Gamma(l)$. For both of these choices, the operator $U^{(l)}$ is unitary, and, as we shall see in next section, these choices also guarantee that the quantum walk has the same symmetry group as the graph.

4.3   Symmetry Considerations

Symmetry plays quite an important role in being able to determine the basic evolution properties of quantum walks. In this Section, we present a general framework of utilizing symmetries, following [45]. Suppose we have a graph $\mathcal{G} = (V, E)$ with vertices of two types[31] — previously called targets and normal vertices. Let $\mathcal{A}$ be the group of automorphisms of the graph that also preserve vertex types. An automorphism a of $\mathcal{G}$ is a mapping $a : V \to V$ such that for any two vertices $v_1, v_2 \in V$, there is an edge connecting $a(v_1)$ and $a(v_2)$ if and only if there is an edge connecting v_1 and v_2. Each automorphism a induces a unitary mapping U_a on the Hilbert space of the graph $\mathcal{G}$, such that $U_a|v_1, v_2\rangle = |a(v_1), a(v_2)\rangle$. Suppose now that $\mathcal{H}$ can be decomposed into m subspaces,

$$\mathcal{H} = \bigoplus_{j=1}^{m} \mathcal{H}_j,$$

where each $\mathcal{H}_j$ is the span of some subset B_j of the canonical basis elements and is invariant under U_a for all $a \in \mathcal{A}$. We shall also assume that each $\mathcal{H}_j$ does not contain any smaller invariant subspaces. This can always be done — in the worst case, the subsets B_j are exactly single edge states, while in other cases the subspaces are larger and help us to obtain a considerable reduction of the dimensionality of the problem.

---

[31]The number of types of vertices can be arbitrary, but for our purposes two suffices. If the number would be higher, the automorphisms that we define would map vertices of each type onto vertices of the same type.

Next, in each invariant subspace we form a vector

$$|w_j\rangle = \frac{1}{\sqrt{d_j}} \sum_{|v_1,v_2\rangle \in B_j} |v_1, v_2\rangle \tag{4.11}$$

that is the sum of all of the canonical basis elements in the subspace, with d_j the dimension of $\mathcal{H}_j$. This vector satisfies $U_a|w_j\rangle = |w_j\rangle$ for all $a \in \mathcal{A}$. Moreover, it is the only vector in $\mathcal{H}_j$ that satisfies this condition. With the help of the vectors $|\psi_j\rangle$, we define the space $\mathcal{S} = \ell^2(\{|w_j\rangle : j = 1, 2, \ldots, m\})$, and note that $\mathcal{S} = \{|\psi\rangle \in \mathcal{H} : U_a|\psi\rangle = |\psi\rangle, \forall a \in \mathcal{A}\}$. The space $\mathcal{S}$ has dimension equal to the number of invariant subspaces B_j.

Now suppose that the quantum walk operator U commutes with the automorphisms, i.e. $[U, U_a] = 0$ for all $a \in \mathcal{A}$. This implies that if $U_a|\psi\rangle = |\psi\rangle$, then $U_aU|\psi\rangle = U|\psi\rangle$. Thus if $|\psi\rangle \in \mathcal{S}$, then $U|\psi\rangle \in \mathcal{S}$, meaning the subspace $\mathcal{S}$ is closed under the action of the step operator U. Correspondingly, if the initial state of the walk is in $\mathcal{S}$, then we only need to consider states in $\mathcal{S}$ to describe the state of the walk at any time. It is useful when the automorphism group is large because $\mathcal{S}$ then can have a much smaller dimension than $\mathcal{H}$, simplifying the analysis greatly.

Now let us demonstrate that the unitary operator U defined by the local unitary operators in (4.10a) does, in fact, commute with all of the automorphisms of a graph that leave the target vertices fixed. It should be clear that this holds, as the construction of the evolution unitary is based on the structure of the graph. Nevertheless, let us see it directly. If these operators commute when applied to all of the elements of the canonical basis, then they commute. As before, let $\Gamma(v)$ be the set of vertices in V that are connected to the vertex v, and if $v' \in \Gamma(v)$ then $\Gamma(v; v') = \Gamma(v) \setminus \{v'\}$. Finally, let $|\Gamma(v)|$ be the number of elements in $\Gamma(v)$. Then we have

$$U_aU|v_1, v_2\rangle = -r^{(v_2)}|a(v_2), a(v_1)\rangle + t^{(v_2)} \sum_{v \in \Gamma(v_2;v_1)} |a(v_2), a(v)\rangle,$$

while also knowing that

$$UU_a|v_1, v_2\rangle = -r^{[a(v_2)]}|a(v_2), a(v_1)\rangle + t^{[a(v_2)]} \sum_{v \in \Gamma(a(v_2);a(v_1))} |a(v_2), v\rangle.$$

First, note that the reflection and transmission amplitudes in this equation are the same as those in the previous equation, i.e. $r^{(v_2)} = r^{[a(v_2)]}$ and $t^{(v_2)} = t^{[a(v_2)]}$. This is a consequence of $|\Gamma(v_2; v_1)| = |\Gamma(a(v_2); a(v_1))|$ – the key properties of the local unitaries are conserved as the vertices of some type are mapped to the vertices of the same type, thus also the degree of the vertices is conserved. Second, we also know that $\Gamma(a(v_2); a(v_1)) = \{a(v) : v \in \Gamma(v_2; v_1)\}$, so that the sums in the two equations are identical. Therefore, $U_aU|v_1, v_2\rangle = UU_a|v_1, v_2\rangle$, implying $[U, U_a] = 0$.

Symmetries of a graph thus allow us to analyze the evolution under a quantum walk in a reduced subspace, if the initial state belongs to $\mathcal{S}$. This is the case for the uniform superposition, which is the reason we use it as our starting point in quantum walk searches. In the next Section, we showcase how this works for the complete graph, where the symmetries dictate that the dimensionality of $\mathcal{S}$ is only 4.
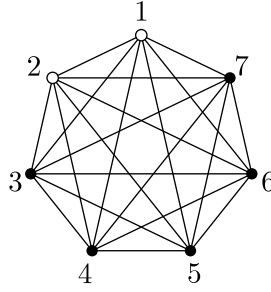
Fig. 4.4. An example of a complete graph with $N = 7$ vertices out of which $k = 2$ are targets (white ones). A solution for the scattering-quantum-walk search on such graph leads to a reduction in dimensionality of the problem to only four dimensions.

4.4   Search on a Complete Graph

Let us now consider a specific search problem where we will show basic manipulations and methods used to solve these problems. We consider a complete graph with N vertices comprising the set $\mathcal{N}$ (see Fig. 4.4). This is a specific graph with each vertex connected to all of the other vertices by an edge. The graph has thus $N(N-1)/2$ edges which define the Hilbert space of dimension $\dim \mathcal{H} = N(N-1)$ for a scattering quantum walk, according to (3.19). Let k be the number of special vertices. Recalling what was said in Sec. 4.3, the elements of the set $\mathcal{K}$ of these target vertices can be labeled as $j = 1, 2, \ldots, k$ and corresponding local unitary evolutions will be defined by (4.10a) and (4.10c). We label the normal vertices $j = k+1, k+2, \ldots, N$, and define their local unitary evolutions by (4.10a) and (4.10b). The transmission and reflection coefficients for all normal vertices j are the same, with $|\Gamma(j)| = N - 1$.

There are four types of edge states here: those directed from normal vertices to special vertices, from special vertices to normal vertices, connecting normal vertices and connecting special vertices. Because of the symmetry of the complete graph, these four types also define the basis vectors of the subspace $\mathcal{S}$ according to (4.11):

$$|w_1\rangle = \frac{1}{\sqrt{k(N-k)}} \sum_{a=k+1}^{N} \sum_{b=1}^{k} |a,b\rangle,$$

$$|w_2\rangle = \frac{1}{\sqrt{k(N-k)}} \sum_{a=1}^{k} \sum_{b=k+1}^{N} |a,b\rangle,$$

$$|w_3\rangle = \frac{1}{\sqrt{(N-k)(N-k-1)}} \sum_{a=k+1}^{N} \sum_{\substack{b=k+1 \\ b \neq a}}^{N} |a,b\rangle,$$

$$|w_4\rangle = \frac{1}{\sqrt{k(k-1)}} \sum_{a=1}^{k} \sum_{\substack{b=1 \\ b \neq a}}^{k} |a,b\rangle.$$

These are equal superpositions of all edge states directed from normal to special, from special to normal, connecting only normal and connecting only special vertices. The unitary evolution of a uniform superposition of edge states, as shown in Sec. 4.3, happens within the subspace $\mathcal{S}$ spanned by the four vectors $|w_k\rangle$, $k = 1, 2, 3, 4$, and is given by a 4×4 matrix

$$U|_{\mathcal{S}} = \begin{bmatrix} 0 & q & s & 0 \\ e^{i\phi} & 0 & 0 & 0 \\ 0 & s & -q & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{bmatrix}, \tag{4.12}$$

where

$$\begin{aligned} q &= -r + t(k-1) = -1 + \frac{2k}{N-1}, \\ s &= \sqrt{1 - q^2} = t\sqrt{k(N-k-1)}. \end{aligned}$$

Note that the subspace spanned by the vectors $|w_k\rangle$, $k = 1, 2, 3$ is decoupled from the subspace spanned by the vector $|w_4\rangle$. Indeed, the vector $|w_4\rangle$ (a superposition of edge states from target to target vertices) is invariant under the step unitary, up to a phase factor.

In analogy with Grover's search [see (4.4)], we now take the equal superposition of all edge states for our initial state:

$$|\psi_0\rangle = \frac{1}{\sqrt{N(N-1)}} \sum_{|j,l\rangle \in \mathcal{H}} |j, l\rangle. \tag{4.13}$$

This is on one hand a necessity in order for the search to be efficient, but on the other hand it is a state where no prior information about any target vertex is used, as we do not expect to know it beforehand. We can write $|\psi_0\rangle$ as a superposition of the states $|w_k\rangle$ (thus it belongs to $\mathcal{S}$) as

$$\begin{aligned} |\psi_0\rangle &= \sqrt{\frac{k(N-k)}{N(N-1)}} \left(|w_1\rangle + |w_2\rangle\right) + \\ &\quad + \sqrt{\frac{(N-k)(N-k-1)}{N(N-1)}} |w_3\rangle + \sqrt{\frac{k(k-1)}{N(N-1)}} |w_4\rangle. \end{aligned}$$

We shall analyze what happens for two choices of the phase-shift on the target vertices: $\phi = 0$ and $\phi = \pi$. First, we start with a bad choice. For $\phi = 0$ (target vertices do not scatter, just reflect without a phase shift), we find that the initial state can be written as a superposition of eigenstates of U with an eigenvalue equal to unity, in particular,

$$\begin{aligned} |\tilde{u}_0\rangle &= \frac{1}{\sqrt{3+q}} \begin{bmatrix} \sqrt{1+q} \\ \sqrt{1+q} \\ \sqrt{1-q} \\ 0 \end{bmatrix}, \\ |\tilde{u}_0'\rangle &= (0, 0, 0, 1)^T, \end{aligned}$$

and the initial state can be expressed in terms of them as

$$|\psi_0\rangle = \sqrt{\frac{(N-v)(N+v-1)}{N(N-1)}}|\tilde{u}_0\rangle + \sqrt{\frac{v(v-1)}{N(N-1)}}|\tilde{u}_0'\rangle.$$

This, however, means that the initial state is an eigenstate of the step unitary U as well. In this case, the quantum walk gives us no advantage over a classical search, as the measurement in any time gives us only a random edge state.

Second, we pick the value of $\phi = \pi$, and see that the behavior of the quantum walk is quite different. The state of the walk after m steps is derived by employing the decomposition formula

$$|\psi_m\rangle = \sum_\lambda \left(\lambda^m\langle\mu_\lambda|\psi_0\rangle\right)|\mu_\lambda\rangle, \tag{4.14}$$

where $|\mu_\lambda\rangle$ are the orthogonal eigenvectors corresponding to the eigenvalue λ of the unitary operator U. In the limit $N \gg k \geq 1$ we find that

$$U^m|\psi_0\rangle \simeq \frac{1}{\sqrt{2}}\begin{bmatrix} \sin(2m+1)\frac{\theta}{2} \\ -\sin(2m-1)\frac{\theta}{2} \\ \sqrt{2}\cos m\theta \\ 0 \end{bmatrix}, \tag{4.15}$$

where

$$\tan\theta = \frac{\sqrt{k(2N-k-2)}}{N-k-1}. \tag{4.16}$$

We find that the probability amplitudes for edge states not connected to special vertices (the third component in the vector) are approximately equal to zero when $\theta m = \pi/2$. If we measure the walker after such a choice of steps, with probability close to unity we will find an edge connected to one of the target vertices. Therefore, the number of steps needed to find one of the target vertices with reasonable probability is of the order $O(\sqrt{N/k})$ for large N, in particular

$$\tilde{m} \simeq \frac{\pi}{2\sqrt{2}}\sqrt{\frac{N}{k}}. \tag{4.17}$$

This is a quadratic speedup over any classical algorithm that needs at least $O(N/k)$ steps to do the task when searching an unstructured database.

It should be noted that for the case $k = 1$, the vector $|w_4\rangle$ is not defined and the dimension of the problem is reduced to three. In this special case, the analysis becomes even simpler and the quadratic speedup result remains valid.

To put the problem into perspective, let us compare it with the results of [23], studying search on a hypercube. They present an algorithm for finding a single target vertex with success probability of approximately $1/2$, which is in contrast with our finding an edge with a special state with probability close to unity. In our case, the probability is (almost) equally split between two possible sets of edge states, those *leaving* the special vertex and those *entering* the special

vertex (states $|w_1\rangle$ and $|w_2\rangle$). If we would reformulate our search in the coined quantum walk language, we would find that our results correspond to either finding the particle on the special vertex (with an arbitrary coin state, i.e. our state $|w_1\rangle$) or finding it on one of the neighboring vertices with the coin pointing to the special one (our state $|w_2\rangle$).

In [43], a quantum walk that performs an exact Grover search is discussed. The graph the quantum walk is performed on is a complete graph that has loops added to each vertex. The coins for normal vertices are chosen to be of the Grover type. However, the coins for the special vertices are "minus" the Grover coins (adding a phase shift of π). This leads to almost the same evolution as in the case of Grover's search; the only difference is that one step of Grover's algorithm corresponds to two steps of the quantum walk.

Finally, let us summarize the situation as it presently stands. Quantum-walk searches have two figures of merit, the number of steps necessary to find a special vertex and the probability of finding it after a specific number of steps. On a complete graph without loops [the result we obtained in (4.15)) one needs $\sqrt{2}$-times as many steps as in the Grover search for the corresponding problem (a search within N elements), and after this many number of steps the probability of finding the special vertex is equal to unity. On a complete graph with loops [43], twice as many steps as in the corresponding Grover search are required, and the probability of finding the special vertex is again equal to unity. A rigorous comparison of these properties on a hypercube can be found in Ref. [46], where adding loops to the graph again results in the increase if the necessary number of walk steps by a factor of $\sqrt{2}$. The result holds also for the complete graph and explains the differences stated.

Exercise 13 Find the evolution of an equal superposition initial state for a quantum scattering walk on a complete graph with loops using the Grover coin on normal vertices and minus the Grover coin on target vertices.

4.4.1   Oracle Controlled Evolution

In this Section we will make a small detour from the particular results obtained in previous pages. We stated that quantum walk search is faster than the classical one, yet we have not really supported the statement. In order to be able to make such a comparison more rigorously we will use oracles defined in Sec. 4.1.1 and the number of their calls as a mean of comparison. Oracles are the bearers of information about searched-for targets and they provide a way to quantify the resources needed to perform a search. It is the number of times we call an oracle that tells us how efficient the search is. Let us have a closer look on how oracles fit into the problems of quantum-walk searches.

In particular, the oracle $\mathcal{C}V_f$ we will use is given by (4.2) from Sec. 4.1.1. On one hand, when marking target elements, it can be used to count the resources used in a search algorithm such as Grover's search. On the other hand, we have devised and discussed quantum walks with position-dependent coins (scattering quantum walks) in the previous Sections. If we want to compare the necessary resources, we need to connect the quantum oracle concept with the notion of a coin for a quantum walk. Just as in scattering quantum walks the state of the walker is given by two vertices (an edge connecting them), we will use the oracle in two steps. In the first step, the information about one of the vertices contained in the state of the walker will be extracted to one ancillary system. This information will be then used as the input to the oracle.
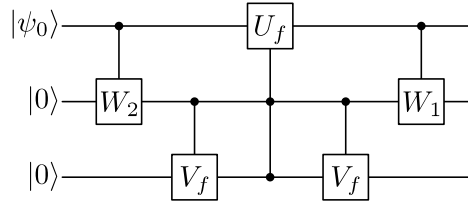
Fig. 4.5. A logical circuit (network) that implements a single step of a scattering quantum walk search, making use of the quantum oracle $\mathcal{C}$. The first input corresponds to a quantum walker originally prepared in the state $|\psi_0\rangle$. The second input represents a vertex state, while the third input represents an ancillary qubit.

The oracle will tell us whether the vertex is a target or not, adding this information to the second ancillary system. Finally, this information will be used to determine what type of coin will be used on a given vertex.

In particular, if l is a vertex of the graph, $f(l) = 0$ corresponds to a normal vertex and $f(l) = 1$ corresponds to a target vertex, i.e. the type of vertex we are trying to find. If l is a normal vertex, the local unitary operator corresponding to it will be denoted by $U_0^{(l)}$, and if it is a target vertex the local unitary will be denoted by $U_1^{(l)}$.

Our quantum circuit will act on a tensor product of the Hilbert space $\mathcal{H}$ for the quantum walk (3.19), the Hilbert space for vertices, $\mathcal{H}_v$, and a qubit Hilbert space, $\mathcal{H}_2$. The vertex space is

$$\mathcal{H}_v = \ell^2(\{|l\rangle : l \in V\}). \tag{4.18}$$

We now define an operator $\mathcal{C}U$ acting on in the following way on the edge register and the two ancillary ones as

$$\mathcal{C}U(|k,l\rangle \otimes |l\rangle \otimes |c\rangle) = \left(U_c^{(l)}|k,l\rangle\right) \otimes |l\rangle \otimes |c\rangle. \tag{4.19}$$

This equation does not completely specify the actions of $\mathcal{C}U$. In particular, it does not specify its action on states of the form $|k,l\rangle \otimes |l'\rangle \otimes |c\rangle$, where $l \neq l'$, but we will only need to consider its action on states of the form given in the previous equation as it can always be expanded to a unitary operator on the whole Hilbert space.

The quantum circuit that implements one step of our quantum walk search is given in Fig. 4.5. The first input stands for a state of the quantum walker (i.e. any superposition of edge states), the second for a vertex state, and the third for an ancillary qubit. The input state is $|\psi_{\text{init}}\rangle = |\psi_0\rangle \otimes |0\rangle \otimes |0\rangle$ where $|\psi_0\rangle$ is a general state in the Hilbert space $\mathcal{H}$ of edge states,

$$|\psi_0\rangle = \sum_{ml \in E} a_{ml}|m,l\rangle, \qquad \sum_{ml \in E} |a_{ml}|^2 = 1.$$

The state $|0\rangle$ in the second slot (input) in Fig. 4.5 is one of the vertex states, which, besides labeling a particular vertex, will also serve as a reference "zero" state. First, we apply the operator $\mathcal{C}W_2$ which maps the state $|m,l\rangle \otimes |0\rangle$ in $\mathcal{H} \otimes \mathcal{H}_v$ to $|m,l\rangle \otimes |l\rangle$. Such a unitary can be

implemented, e.g. as presented in [47] or [48]. After this operator is applied, we get

$$|\psi_0\rangle \mapsto \sum_{ml\in E} a_{ml}|m,l\rangle \otimes |l\rangle \otimes |0\rangle \equiv |\psi_1\rangle.$$

Next, we apply the quantum oracle (4.2) to the vertex state and the qubit, yielding

$$|\psi_1\rangle \mapsto \sum_{ml\in E} a_{ml}|m,l\rangle \otimes |l\rangle \otimes |f(l)\rangle \equiv |\psi_2\rangle.$$

Now we can apply the $\mathcal{C}U$ operator (which coin gets applied is controlled by the last two registers) from (4.19)to the state, producing

$$|\psi_2\rangle \mapsto \sum_{ml\in E} a_{ml}\left(U_{f(l)}^{(l)}|m,l\rangle\right) \otimes |l\rangle \otimes |f(l)\rangle \equiv |\psi_3\rangle.$$

Both of the ancillary systems acted as controls, and they need to be reset before we can make another quantum walk step. This erasure is the task of the remaining two gates in the circuit. Since the local unitary operator $U_{f(l)}^{(l)}$ acts only on the edge space and maps Ω_l to A_l as given in Sec. 3.3.1, the state $|\psi_3\rangle$ can be rewritten as

$$|\psi_3\rangle = \sum_{lm\in E} b_{lm}|l,m\rangle \otimes |l\rangle \otimes |f(l)\rangle.$$

A second application of the quantum oracle on the last two register resets the last qubit state to $|0\rangle$. However, to reset the vertex state in the second register, we cannot use the $\mathcal{C}W_2$ operation as before, since the information contained in the edge state about the vertex state has moved from the second to the first position. Therefore, we need the operator $\mathcal{C}W_1$, which would map $|l,m\rangle \otimes |l\rangle$ in $\mathcal{H} \otimes \mathcal{H}_v$ to $|l,m\rangle \otimes |0\rangle$, giving

$$|\psi_3\rangle \mapsto \sum_{lm\in E} b_{lm}|l,m\rangle \otimes |0\rangle \otimes |0\rangle.$$

This $\mathcal{C}W_1$ operation can be constructed in the same manner as operation $\mathcal{C}W_2$ in [47], using now the first of the edge states as a control.

We have thus performed one step of the walk and reseted the ancillas, so that the circuit can be applied again to perform additional steps of the walk. The oracle in these problems is a resource – giving us additional information every time we call it. The number of necessary oracle calls then tells us how efficient our algorithm is. Here we see that at most $2m$ oracle calls are needed in a quantum walk search algorithm that requires m walk steps.

4.5 Other Examples of Searches on Graphs

The complete graph search example from Sec. 4.4 is just one of many possibilities. In [45], the reader can find a number of possibilities for the choice of graph suitable for quantum walk search (with at most quadratic speedups). On one hand these examples are artificial, however one should bear in mind the results of Sec. 4.4.1, that the oracle determining the selected elements
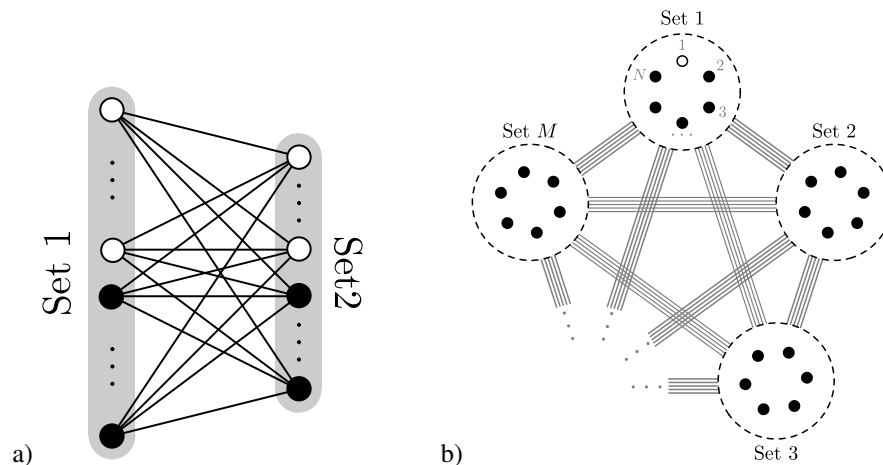
Fig. 4.6. Two types of graphs suitable for quantum-walk searches: a) A bipartite graph consisting of two sets of vertices where each vertex from one set is connected to all of the vertices from the other set, while there are no connections within a set. There may also be a different number of target vertices in each set. b) An M-partite complete graph consisting of M sets of vertices, where each set contains N vertices. There exists an edge for every pair of vertices not belonging to the same set, while there is no edge connecting any two vertices within the same set. Here we look for one special vertex in one of the sets.

is independent of the choice of the graph we make. Hence, whatever choice of graph with a proper number of vertices we make, we can always use the same oracle to perform the search. It is however crucial to make a good graph choice, as the efficiency of the walk algorithm is dependent on this choice. Indeed, taking a circular graph clearly cannot give us any means to speed up classical search, as the "information" about the target has to "travel" at least along half of the vertices (the amplitude needs to rise significantly at the target vertex). On the other side of the spectrum of graph choices is the complete graph which makes the search quadratically faster than in the classical case.

One interesting example of a graph to search on is the complete bipartite graph depicted in Fig. 4.6a. It is a graph with vertices belonging to two sets. Each vertex belongs to a particular set, is connected to all the vertices belonging to the other set, but not with any vertices within the same set. Another graph of interest is the complete M-partite graph (Fig. 4.6b). This graph is composed of M sets, each containing N vertices. Here the vertices from one set are connected to all the vertices from any other set, but again, vertices within a set are not connected. Clearly, taking $M = 2$ we recover the complete bipartite graph and taking $N = 1$ we recover the complete graph. In all of these cases, a quadratic speedup (in the number of oracle calls) over the best classical search algorithm is possible.

Until now, we had the opportunity to study quantum-walk searches where target vertices were marked by an additional phase-shift, utilizing the oracle from (4.2) – the oracle tells us which vertices are special, while the choice of the graph is ours. There are, however, also examples where the goal of the search is to find a distinctive topological feature of the graph. In those
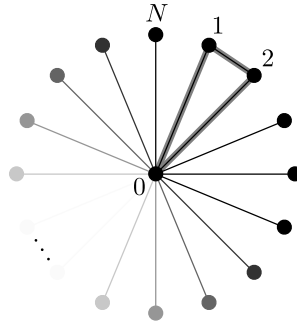
Fig. 4.7. A star graph having two arm vertices connected; these vertices are purely transmissive. Other arm vertices are purely reflective and the central vertex is Grover-like. The evolution, when starting from equal superposition, gives us again a speedup in locating the extra edge with high probability and the state can be localised almost entirely on the grey triangle.

cases, we may also employ an oracle, albeit in a different form – giving us the information about the neighbors that the walker can visit. This also means, that the structure of the graph is now a part of the oracle and not our choice anymore.

Let us now take a look at the graph depicted in Fig. 4.7 (see also Ref. [49]) – a star graph with N spikes, where two of the spike vertices (say 1 and 2) are connected. The Hilbert space is again defined as a span of all edge states of the graph. We choose the central node labeled 0 to have a Grover-like coin, obeying (3.25) with $t = 2/(N+1)$. The local unitary evolution for the outside vertices is the same as before (without a phase-shift), except for the two connected vertices 1 and 2, which obviously have two outgoing edges. There we set the local unitary evolution U_0 to

$$U_0|0,1\rangle = r_0|1,0\rangle + t_0|1,2\rangle, \qquad U_0|0,2\rangle = r_0|2,0\rangle + t_0|2,1\rangle,$$
$$U_0|2,1\rangle = t_0|1,0\rangle - r_0|1,2\rangle, \qquad U_0|1,2\rangle = t_0|2,0\rangle - r_0|2,1\rangle,$$

with $t_0 = \sqrt{1 - r_0^2}$ and r_0 chosen from the interval $[-1; 1]$, with $r_0 = -1$ corresponding to Grover's search ("marked" edges get a -1 phase), $r_0 = 1$ giving a repetitive application of Grover's coin without any phase-flips and $r_0 = 0$ for purely transmitting vertices. The quantum walk evolution is based on an oracle that determines the neighbors of the vertex we enter as an input. We further suppose that the (name of the) central vertex is known to us beforehand.

To classically find the extra edge, we would have to sift through the outlying vertices to find one that has two neighbors. Note that looking at a typical vertex with only 1-neighbor, we only get the knowledge that we missed. Thus, this is unstructured search, which classically requires $O(N)$ calls to the oracle giving out the neighbors of a given vertex. Again, as we will see, we can obtain a quadratic improvement but now without using any phase-flips. The results of Sec. 4.3 once again give us a way to find an invariant subspace, which is now five-dimensional, spanned

by the vectors

$$
\begin{aligned}
|w_1\rangle &= \frac{1}{\sqrt{2}}(|0,1\rangle + |0,2\rangle), \\
|w_2\rangle &= \frac{1}{\sqrt{2}}(|1,0\rangle + |2,0\rangle), \\
|w_3\rangle &= \frac{1}{\sqrt{N-2}}\sum_{j=3}^{N}|0,j\rangle, \\
|w_4\rangle &= \frac{1}{\sqrt{N-2}}\sum_{j=3}^{N}|j,0\rangle, \\
|w_5\rangle &= \frac{1}{\sqrt{2}}(|1,2\rangle + |2,1\rangle).
\end{aligned}
$$

Note, that it is desirable to end in $|w_5\rangle$ or at least in the states $|w_1\rangle$ or $|w_2\rangle$. Meanwhile, the probability of ending in the states $|w_3\rangle$ and $|w_4\rangle$ should be as small as possible, as these states correspond to edges not connected to either of the vertices 1 or 2.

The choice of initial state,

$$
|\psi_0\rangle = \frac{1}{\sqrt{2N}}\sum_{j=1}^{N}(|0,j\rangle - |j,0\rangle), \tag{4.20}
$$

will lead us to the evolution described by the state after m steps,

$$
|\psi_m\rangle = U^m|\psi_0\rangle \simeq \frac{(-1)^m \Delta}{\sqrt{2}}
\begin{bmatrix}
\sin m\Delta\sqrt{t} \\
\sin m\Delta\sqrt{t} \\
\Delta^{-1}\cos m\Delta\sqrt{t} \\
-\Delta^{-1}\cos m\Delta\sqrt{t} \\
-\frac{t_0}{1-r_0}\sin m\Delta\sqrt{t}
\end{bmatrix},
$$

where

$$
\Delta = \sqrt{\frac{2(1-r_0)}{3-r_0}}.
$$

We chose the initial state (4.20) in such a way that the contributions from different eigenvectors do not cancel in the components for states $|w_3\rangle$ and $|w_4\rangle$ (which would happen if the intitial state was the equal superposition), but rather add up.

The different choices of r_0 result in various behavior of the walk. For $r_0 = 0$ (when the vertices 1 and 2 are purely transmitting), we get a quantum walk algorithm taking $\tilde{m} = \pi\Delta^{-1}\sqrt{N/8}$ steps, ending up with the walker equally distributed on the triangle 012, with equal probability of $1/3$ on every edge of this triangle.

When increasing r_0, the probability to find the walker in the state $|w_5\rangle$, i.e. on the edge 1–2, rises. It might seem that taking $r_0 \to 1$ will give us the walker positioned entirely on the extra edge. However, the parameter Δ depends on r_0 as well. Taking $r_0 \to 1$ gives $\Delta \to 0$,

increasing the required number of steps $\tilde{m}$ necessary to assure us the maximum probability of success. Evidently, $r_0 = 1$ causes the extra edge to become "invisible", making the vertices 1 and 2 no different from others – in this case, the evolution is trivial (up to a global phase).

On the other hand, taking $r_0 = -1$ is a good choice, as it results in $\Delta = 1$. In this case, the extra edge is again not "visible", but the vertices 1 and 2 get a phase-shift π leading to Grover's search on a star graph with $k = 2$ target vertices. Note that for every step of Grover's algorithm we need two steps of the walk on the star graph, so the number of steps needed $\tilde{m} = \pi\sqrt{N/8}$ corresponds to Grover's result in [30].

Let us go back to the most natural $r_0 = 0$ case. There is no active phase-shift occurring anywhere in the graph, yet for the Grover search a phase-flip is a very important element. Where is then the place where something like that occurs in this case? The graph in Fig. 4.7 is not bipartite. The part of the walker leaving the central vertex (outgoing edges) and the part entering it (incoming edges) have a way to interfere via the extra edge. Notice that in the initial state (4.20) has the incoming and outcoming parts initialized with opposite signs. Therefore, the directly reflected amplitude and the amplitude transmitted "around" the triangle can have different signs, combining as if a part that was reflected from one of the special vertices gained a π phase-shift. Thus, in this case a phase-shift action of a unitary was replaced by an interference effect of phase-shifted (otherwise) non-interfering parts of the walker.

Example 10 Search on the star graph with an extra edge can be also used as a lower bound for a triangle-finding algorithm — see Sec. 4.7.2. In the triangle problem for a graph G with N vertices, we are supposed to find out whether it contains a triangle. The graph is given to us via an oracle $O_G \ket{x,y} \ket{z} = \ket{x,y} \ket{z \oplus E(x,y)}$ that holds the information whether vertices x and y have an edge between them. A clever quantum algorithm for this was given by Magniez et al. [40], taking $O(N^{\frac{13}{10}})$ queries of the edge oracle. ...However, we do not know whether this is an optimal algorithm. The best lower bound for this problem comes from the star graph. Finding the extra edge between a pair of spike vertices certifies the existence of the triangle. Any quantum algorithm for this unstructured problem has to take $O(\sqrt{N(N-1)})$ queries of oracle O_G, as there are $N(N-1)$ possible edges between the N spike vertices in the star graph. Thus, the best algorithm for the triangle problem can not take fewer than $O(N)$ edge oracle O_G queries. Note that if we have an oracle giving us a list of neighbors instead, we can find the special vertex in $\sqrt{N}$ queries using Grover's search with unit cost of asking for its neighbors, as was done above in this Section.

4.6   Abstract Search Algorithm and Spatial Search

All the kinds of search algorithms introduced previously were analyzed in the same way. First the unitary U is spectrally decomposed and then Eq. (4.14) is used to find the evolution of the initial state. This approach is quite universal, however sometimes such analysis is overcomplicated and some other approaches can be used. One such approach was presented in Ref. [43] — abstract search algorithm. It is not as universal, yet it is still quite broad to help solve many search problems easier than by determining the whole evolution. This approach to searches view them from a different perspective than the generalized approach of amplitude amplification [37].

Suppose you have evolution driven by unitary $U = VR$, where R is controlled phase-flip on a single target element $\ket{t}$ given by Eq. (4.3) and V is real unitary operation with a unique (real)

eigenvector $|\psi_0\rangle$ with eigenvalue 1.

As V is real unitary matrix, its non-±1 eigenvalues $\lambda_j^{\pm}$ come in pairs of complex conjugate numbers $e^{\pm i\theta_j}$. The eigenvector with eigenvalue 1 is $|\psi_0\rangle$; let the complex eigenvalues $\lambda_j^{\pm}$ have eigenvectors $|\phi_j^{\pm}\rangle$. One can show that $|\phi_j^{+}\rangle^* = |\phi_j^{-}\rangle$. Finally, let the eigenstates of eigenvalue -1 be $|\rho_k\rangle$. Setting $a_j^{\pm} = \langle\phi_j^{\pm}|t\rangle$ one can also see that the global phase of states $|\phi_j^{\pm}\rangle$ can be set so that $(a_j^{+})^* = a_j^{-} \equiv a_j$. From this it follows that the expansion of state $|t\rangle$ in the basis of V, which we will need, is

$$|t\rangle = a|\psi_0\rangle + \sum_j a_j(|\phi_j^{+}\rangle + |\phi_j^{-}\rangle) + \sum_k a_k|\rho_k\rangle. \tag{4.21}$$

This expansion is used to analyze the operator U. Taking $\theta_{\min}$, the smallest phase of the eigenvalues of V one can find [43] that the most important eigenvalues of U are $e^{\pm i\alpha}$, where[32]

$$\alpha = \Theta\left(a\left(\sum_j \frac{a_j^2}{1 - \cos\theta_j} + \frac{A^2}{4}\right)^{-1/2}\right),$$

where the sum goes only over the complex eigenvalues indices and

$$A = \sqrt{\sum_k a_k^2}$$

is the contribution from -1 eigenvalue expansion coefficients. When $\theta_{\min}$ is small, which is ususal, then also α is small. Let us set also states

$$|\alpha^{\pm}\rangle = \frac{1}{\sqrt{2}}(|\alpha\rangle \pm |-\alpha\rangle),$$

where $|\pm\alpha\rangle$ are eigenvectors of U corresponding to eigenvalues $e^{\pm i\alpha}$. If $\alpha < \theta_{\min}/2$, then the initial state $|\psi_0\rangle$ is close to the state $|\alpha^{-}\rangle$, in particular

$$|\langle\psi_0|\alpha^{-}\rangle| \geq 1 - \Theta\left(\alpha^4 \sum_j \frac{a_j^2}{a^2} \frac{1}{(1 - \cos\theta_j)^2}\right) - \Theta\left(\frac{A^2\alpha^4}{a^2}\right), \tag{4.22a}$$

while the target state $|t\rangle$ falls out to be close to the state $|\alpha^{+}\rangle$,

$$|\langle t|\alpha^{+}\rangle|^2 = \Theta\left(\min\left\{\left(\sum_j a_j^2 \cot^2\frac{\theta_j}{4}\right)^{-1/2}, 1\right\}\right). \tag{4.22b}$$

This means, that one can use this general procedure to find the approximate evolution of the system: if we apply m-times operator U on the initial state, keeping in mind approximations of

---

[32]Symbol $\Theta(g)$ means that the function f is bounded both above and below by g asymptotically, i.e. there exist positive numbers a and b such that $ag(x) \leq f(x) \leq bg(x)$.

Eqs. (4.22), we find

$$U^m|\psi_0\rangle \simeq U^m|\alpha^-\rangle = U^m \frac{1}{\sqrt{2}}(|\alpha\rangle - |-\alpha\rangle) = \frac{1}{\sqrt{2}}\left(e^{im\alpha}|\alpha\rangle - e^{-im\alpha}|-\alpha\rangle\right).$$

When we take $m\alpha = \pi/2$ we find that

$$U^m|\psi_0\rangle \simeq \frac{i}{\sqrt{2}}(|\alpha\rangle + |-\alpha\rangle) = i|\alpha^+\rangle.$$

So the method tells us, that if we start in state $|\psi_0\rangle$ and apply operation U for $\lfloor \pi/2\alpha \rfloor$-times, Eq. (4.22b) will tell us, how close we are to the state $|t\rangle$.

Example 11 Let us consider a lattice with dimension $d \geq 3$ with N vertices arranged as $\sqrt[d]{N} \times \ldots \times \sqrt[d]{N}$ with periodic boundary conditions. Then we can use flip-flop grover coin from Eq. (3.26) in quantum walk search given by algorithm in Fig. 4.3 to find one marked vertex in (optimal) time $O(\sqrt{N})$ starting from initial state of the equal superposition on all sites — such initial state can be constructed from localized state with time expense of $O(\sqrt[d]{N})$. Eqs. (4.22) tell us that initial state is almost $|\alpha^-\rangle$ and the probability of success is constant, i.e. few repetitions of the algorithm suffice to find targeted vertex.

Interestingly continuous-time quantum walks for a long time did not succeed to find an efficient continuous-time alternative to this discrete-time algorithm. Only after introducing spin degree of freedom into the walk researches suceeded to get the same limits.

Example 12 Spatial search on two-dimensional lattice of $\sqrt{N} \times \sqrt{N}$ vertices with periodic boundary can be analyzed with the abstract search algorithm as well, yet the results are not as optimistic as in higher dimensions. For such quantum walk there is a $T = O(\sqrt{N \log N})$ such that after T steps the probability to determine the target vertex is $p = O(1/\log N)$ [43]. By using the method of amplitude amplification (see Sec. 4.1.2) one can obtain constant probability of success with the running time of the algorithm of $O(\sqrt{N} \log N)$. Deeper analysis of the abstract search algorithm and introduction of properly chosen ancillary system and allowed Tulsi [50] to boost the probability of succes to constant thus reducing the running time of the algorithm even more to $O(\sqrt{N \log N})$ steps.

4.7 Subset Finding and Related Problems

Probably the most ingenious algorithm showing the usefulness of discrete-time quantum walks is their application to various subset finding problems. This range of algorithms is based on Grover-like evolution on specially constructed graphs allowing for better efficiency of these algorithms. The history begins in 2003 when Ambainis [38] gave an algorithm for element k-distinctness. It determines whether a given set contains k elements with the same assigned value provided by an oracle, and finds such a set if there is one. Building on this work, Magniez, Santha and Szegedy [40] then provided a triangle-finding algorithm, deciding whether a given graph contains a triangle. This approach was generalized in [39], where an algorithm for subset-finding was provided. A better efficiency was then provided in the updated version of [40], where the authors performed a deeper algebraic analysis of the algorithm. Here we present the algorithm in its most up-to-date and efficient form.
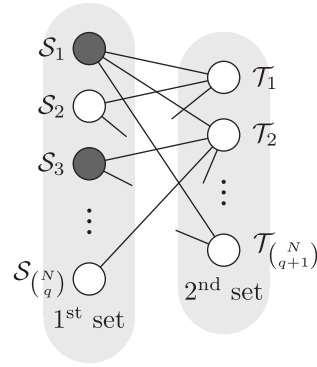
Fig. 4.8. The algorithm for subset finding performs a walk on a bipartite graph whose vertices are identified with subsets of $\mathcal{N}$ having either q elements (on the left), or $q + 1$ elements (on the right). The vertices are connected only if the sets they correspond to differ in exactly one element. The evolution on the graph is given by the Grover coin on most vertices, combined with phase-flips on vertices (q-subsets) containing k-subsets with elements having with property $\mathcal{P}$.

4.7.1   Algorithm for k-subset Finding

Let us discuss the algorithm for the following problem, introduced in [38]. Consider a set $\mathcal{N}$ with N elements combined with values from a finite set $\mathcal{R}$ assigned by a function $f : \mathcal{N} \to \mathcal{R}$. The problem is to determine, whether a k-subset with a given *property* $\mathcal{P} \subset (\mathcal{N} \times \mathcal{R})^k$ exists in the set $\mathcal{N}$. For example, in the *collision problem*, we are given a list of vertices and their assigned colors, and we're asked to find two vertices ($k = 2$) of the same color (if they exist). Another example is the *triangle problem*: given a list of edges in a graph, determine whether we can find a set of 3 edges (here $k = 3$) that form a triangle (if such a set exists).

The information hidden in the function f is given to us in the form of a classical or quantum oracle. Evaluating whether a k-subset has a given property $\mathcal{P}$ (such as: do these three edges form a triangle) should be simple once we know the values of f on the vertices of the k-subset. This allows us to determine the efficiency of a given algorithm using query complexity – counting the number of oracle calls required to find the desired k-subset with a given property.

The aim of the algorithm is to output a subset $\mathcal{K} = \{x_1, x_2, \ldots, x_k\} \subset \mathcal{N}$ such that the l-tuple $((x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_l, f(x_l))) \in \mathcal{P}$ if it exists, when we are given the set $\mathcal{N}$ of elements, a description of an easily computable property $\mathcal{P}$ and the oracle computing the function f on vertices. If there is none such l-tuple, the algorithm should say so. In the classical case, it we have to query the oracle $O(N)$ times to determine the solution to the problem, as the very last element we query could be the one that completes a k-tuple with the property $\mathcal{P}$.

The k-subset finding quantum algorithm is a quantum walk on a specially constructed bipartite graph, depicted in Fig. 4.8. The vertices on the left of this graph correspond to all the possible q-element subsets $\mathcal{S}_j$ of $\mathcal{N}$, while the vertices on the right label all the possible $(q + 1)$-element subsets $\mathcal{T}_j$ of $\mathcal{N}$. Two vertices are connected only if the corresponding sets differ in exactly one element. In the following, we will not make any further distiction between the vertex and its

corresponding set, i.e. $\mathcal{S}$ will be both the set and the vertex it determines.

The size of the parameter q shall be determined later to provide the best efficiency. The simplest choice would be $q = k$, but that does not give much benefit over straightfoward Grover's search over k-tuples, looking whether they have the property $\mathcal{P}$. It will be much better to think of rather large subsets $q = N^\mu$ with $\mu < 1$.

The Hilbert space the algorithm runs in will be spanned by the orthonormal basis

$$
\begin{aligned}
|\mathcal{S}\rangle \otimes |r\rangle \otimes |j\rangle &\equiv |\mathcal{S}, r, j\rangle, \text{ where } |\mathcal{S}| = q, \text{ and } j \notin \mathcal{S}, \\
|\mathcal{T}\rangle \otimes |r\rangle \otimes |j\rangle &\equiv |\mathcal{T}, r, j\rangle, \text{ where } |\mathcal{T}| = q + 1, \text{ and } j \in \mathcal{T},
\end{aligned}
$$

whose second (*data*) register r stores information about $\mathcal{S}$ or $\mathcal{T}$ obtained from the oracle. This part of the system is just an ancillary subspace of suitable dimension. We will also use the notation $f(\mathcal{S}) = \{f(j) : j \in \mathcal{S}\}$ for simplicity. The third register holding $|j\rangle$ functions somewhat like a coin – the state $|j\rangle$ unambiguously points to some neighbor of a subset, so we can use it for moving between sets of the type $\mathcal{S}$ and $\mathcal{T}$. For the vertex $\mathcal{S}$ from the first set it is the element that shall be added to get to vertex $\mathcal{T}$ from the second set (that is why $j \notin \mathcal{S}$). On the other hand, for vertex $\mathcal{T}$ from the second set, the element $j \in \mathcal{T}$ can be removed to get to $\mathcal{S}$.

The graph structure and the Hilbert space we introduced are very useful, as they require only a single query of the oracle when using the translation operator S between the vertices:

$$
\begin{aligned}
S|\mathcal{S}, f(\mathcal{S}), j\rangle &= |\mathcal{S} \cup \{j\}\rangle |f(x_1), \ldots, f(x_q), f(j)\rangle |j\rangle = |\mathcal{S} \cup \{j\}, f(\mathcal{S} \cup \{j\}), j\rangle, \\
S|\mathcal{T}, f(\mathcal{T}), j\rangle &= |\mathcal{T} \setminus \{j\}, f(\mathcal{T} \setminus \{j\}), j\rangle.
\end{aligned}
$$

This operation changes only the element j, so its addition to the set $\mathcal{S}$ requires a single call of the oracle f to determine $f(j)$. On the deletion of the element j, a single call of the oracle is also needed to clean up the register previously holding $f(j)$.

The translation operation handles states that have the oracle information stored in the data register. Before starting the quantum walk, this register needs to be initialized, which takes q calls to the oracle.

The overall evolution is specified as

$$
\left[(SC_G)^{2\tau_1} P \right]^{\tau_2},
$$

where C_G is the standard Grover coin from (3.25), with $d = N - q$ for the first set (the number of elements we can possibly add to the chosen set $\mathcal{S}$) and $d = q + 1$ for the second set (the number of elements we can remove from the chosen set $\mathcal{T}$). In particular,

$$
\begin{aligned}
C_G|\mathcal{S}, f(\mathcal{S}), j\rangle &= -|\mathcal{S}, f(\mathcal{S}), j\rangle + \frac{2}{N-q} \sum_{l \in \mathcal{N} \setminus \mathcal{S}} |\mathcal{S}, f(\mathcal{S}), l\rangle, \\
C_G|\mathcal{T}, f(\mathcal{T}), j\rangle &= -|\mathcal{T}, f(\mathcal{T}), j\rangle + \frac{2}{q+1} \sum_{l \in \mathcal{T}} |\mathcal{T}, f(\mathcal{T}), l\rangle.
\end{aligned}
$$

This operation only affects the coin state (last register), thus it does not require a query to the oracle. After flipping the coin (chosing which element to add/remove), we apply the translation operator S. When this happens $2\tau_1$ times, we use the conditional phase-flip operator P:

$$
P|\mathcal{S}, f(\mathcal{S}), j\rangle = \begin{cases} -|\mathcal{S}, f(\mathcal{S}), j\rangle & \text{if some subset of } \mathcal{S} \text{ has the property } \mathcal{P}, \\ |\mathcal{S}, f(\mathcal{S}), j\rangle & \text{otherwise.} \end{cases}
$$

---

1. prepare initial state $|\psi\rangle = |\psi_0\rangle$ according to Eq. (4.23)
2. for τ_2 steps repeat:

   (a) for $2\tau_1$ times apply SC_G on the state $|\psi\rangle$

   (b) aplly operation P on state $|\psi\rangle$

3. measurement should with high probability return $\mathcal{S}$ such that colliding elements are in the set

---

Fig. 4.9. Algorithm for distinctness problem repeats Grover-like evolution for time τ_1 without applying phase-flip P. The phase-flip is applied only afterwards and then the process is repeated for τ_2 times.

The construction that will be presented in a moment is such that P doesn't need to be specified for the states from the second set, we may still assume it acts as identity there. Finally, the algorithm consists of repeating the "walk $2\tau_1$ times, do a conditional phase flip" combination τ_2 times (see also Fig. 4.9).

The initial state is chosen as an equal superposition of all subsets of size q, with their data registers initialized, and the third register (the walker) uniformly spread through the available positions not the corresponding set $\mathcal{S}$ from the first register,

$$|\psi_0\rangle = \frac{1}{\sqrt{c}} \sum_{\substack{\mathcal{S}\in\mathcal{N} \\ |\mathcal{S}|=q}} \sum_{j\in\mathcal{N}\setminus\mathcal{S}} |\mathcal{S}, f(\mathcal{S}), j\rangle, \tag{4.23}$$

with the normalization constant $c = \binom{N}{q}(N-q)$. The choice of this state on one hand makes no assumptions about the target vertex we shall end in. To assure we end up in a state of the type $|\mathcal{S}, f(\mathcal{S}), j\rangle$, the operation SC_G is repeated an even number of times.

Up to now, we decided what the initial state will be and how the evolution proceeds. It is time to specify τ_1 and τ_2. A quite lengthy derivation in [39] shows a choice of these parameters leading to an evolution that transforms the initial state $|\psi_0\rangle$ to a state with $\mathcal{S}$ containing a set with property $\mathcal{P}$ with high probability, if such a set it exists. It is

$$\tau_1 = \left\lfloor \frac{\pi}{2}\sqrt{\frac{q}{k}} \right\rceil, \qquad \tau_2 = \left\lfloor \frac{\pi}{4}\left(\frac{N}{q}\right)^{k/2} \right\rceil, \tag{4.24}$$

where $\lfloor x \rceil$ means the closest integer to x. These required repetition numbers are approximations under the assumptions $N, q \gg k \geq 1$. Putting the things together and setting $q = N^\mu$, one finds (see also Tab. 4.1) that the efficiency of the algorithm is

$$O(q + 2\tau_1\tau_2) = O\left(N^\mu + N^{\frac{1}{2}[(1-k)\mu+k]}\right). \tag{4.25}$$

This efficiency is smallest (with respect to the choice of μ) when both terms have the same exponents, i.e. for $\mu = k/(k+1)$. The efficiency of this quantum algorithm then is $O(N^{\frac{k}{k+1}})$ of oracle calls. Again, we stress that here the process of determining whether the set $\mathcal{S}$ contains

a subset with the property $\mathcal{P}$ is considered to be "fast", i.e. it requires no resources[33].

Finally, we can compare the query complexity $O\left(N^{\frac{k}{k+1}}\right)$ of the quantum subset-finding algorithm to the best classical one, where $O(N)$ queries are required both on average and in the worst case if there exists a collision of two elements. If there is no collision, one has to query the oracle N times in order to find, that there is no collision.

Example 13 When we set $k = 1$ in (4.25), we get an algorithm with query complexity $O\left(N^{\mu}+N^{\frac{1}{2}}\right)$. We are free to choose μ from the interval $[0; 1/2]$ where the efficiency remains the same as in Grover's search. In fact, by setting $\mu = 0$ we make the vertices singleton sets and the algorithm becomes Grover's search by a quantum walk on a complete graph as in Sec. 4.4.

4.7.2   Algorithm for Finding k-cliques in Graphs

The following is an extension of the above results, also presented in [38, 39], dealing with a problem of finding complete subgraphs (cliques) of size k in a graph[34] $\mathcal{G}$ with N vertices. The oracle in this problem is a device that takes two vertices j, m of the graph $\mathcal{G}$ as input and answers whether the pair jm is an edge of $\mathcal{G}$.

Again, the quantum walk algorithm will work with subsets of $\mathcal{G}$'s vertices containing q, resp. $q + 1$ vertices. There are, however, a few necessary changes to what we saw in the previous Section. First, the costs of initialization and application of S increase. When initializing the system, we need to know the information (stored in the ancillary data register) about the edges between the vertices contained in the subset, which now requires $O(q^2)$ queries to the oracle. Also, when performing a translation with the operator S, the addition (or removal) of a vertex j requires $O(q)$ queries to the oracle for the potential edges connected to the vertex j. This does not change the exponent μ, but raises the query complexity to $O\left(N^{\frac{2k}{k+1}}\right)$ oracle calls (see Tab. 4.1).

This is not the best that one can do, and the efficiency can be improved further by an ingenious approach from [38, 39]. Instead of searching for k-cliques, the crucial idea is to look for only $(k-1)$-cliques. In particular, we search for subsets with $(k-1)$ elements that fulfill the property that these elements form a $(k-1)$-clique (this changes τ_2, see Tab. 4.1). At the same time, we also want all these vertices to be connected to one other vertex. This is a "redefinition" of the property $\mathcal{P}$ which now has an impact on the complexity of performing the operation P. Effectively, P works as an oracle that takes some $\mathcal{S}$ on input (and either lets it be or gives it a -1 phase) and we will thus call it oracle$_2$ as opposed to the oracle giving us information about edges of graph $\mathcal{G}$ which, for the time being, we shall call oracle$_1$.

The oracle$_2$ can be implemented by an algorithm that performs a search for a vertex that is fully connected to some $(k-1)$-clique within $\mathcal{S}$. Whether some vertex j has this property shall be provided to us by another oracle$_3$. If we had it at hand, then searching for the special vertex could be done by a Grover search on $O(N)$ vertices (elements), having to query oracle$_3$ $O(\sqrt{N})$ times.

---

[33]It doesn't require calling the oracle, only some extra computational resources, such as for the collision problem we need to sort the list $f(x_1), \ldots, f(x_q)$ and see if any two of the values match.

[34]Not to be confused with the bipartite graph from Fig. 4.8 which just shows a process of construction of the walk performing the search for the clique.

	k-subset	k-clique	recursive k-clique
initialization	q	q^2	q^2
S	1	q	q
C_G	0	0	0
P	0	0	$\sqrt{N} \times N^{\frac{k-1}{k}}$
τ_1	$\sim \sqrt{q}$	$\sim \sqrt{q}$	$\sim \sqrt{q}$
τ_2	$\sim \left(\frac{N}{q}\right)^{\frac{k}{2}}$	$\sim \left(\frac{N}{q}\right)^{\frac{k}{2}}$	$\sim \left(\frac{N}{q}\right)^{\frac{k-1}{2}}$
query complexity	$q + 2\tau_1\tau_2$	$q^2 + 2q\tau_1\tau_2$	$q + \tau_2\left(2q\tau_1 + \sqrt{N} \times q^{\frac{k-1}{k}}\right)$
optimal μ	$\frac{k}{k+1}$	$\frac{2k}{k+1}$	$\frac{5k-2}{2k+4} = 1.3$ for $k = 3$ $\frac{2(k-1)}{k}$ for $k > 3$

Tab. 4.1. Summary of the query complexity for quantum walk algorithms for k-subset finding, k-clique finding and recursive k-clique finding. The algorithms involve a walk on subsets of size $q = N^\mu$. The bottom part of the table shows the summary complexity and the best choice for the exponent μ, resulting in query complexity N^μ.

We are able to construct (at least in principle) this oracle$_3$, having some vertex j and a subset $\mathcal{S}$ as input. We know how to search for a $(k-1)$-clique within the set $\mathcal{S}$, having q elements. The search for this $(k-1)$ clique connected to j is application of a standard $(k-1)$-subset finding procedure within $\mathcal{S}$, where the oracle$_1$ sets the property $\mathcal{P}$ for this sub-search. Checking whether some $(k-1)$-element subset of $\mathcal{S}$ connected to j is a clique requires no additional queries to oracle$_1$, as this information is already stored in the ancillary data register state corresponding to $\mathcal{S}$. The oracle$_3$ thus needs to call oracle$_1$ $O\big(q^{\frac{k-1}{k}}\big)$ times. Knowing the complexity of oracle$_3$, we can determine the complexity of calling oracle$_2$, i.e. of the operation P – it is $O\big(\sqrt{N} \times q^{\frac{k-1}{k}}\big)$. Putting it all together, the combined algorithm for k-subset finding requires

$$O\left(q + \tau_2\left(2q\tau_1 + \sqrt{N} \times q^{\frac{k-1}{k}}\right)\right)$$

calls to the[35] oracle$_1$.

Let us look at the result for various k. For $k = 2$ (we look for an edge) the optimal efficiency is that of Grover's search on the $N(N-1)$ possible edges, i.e. $O(N)$ calls to the oracle. However, for $k > 2$, the presented algorithm is more efficient than direct Grover's search for k-cliques which requires $O(N^{k/2})$ calls to the oracle. First, a brute force querying of all the edges and then performing a search on the received information has query complexity $\Omega(N^2)$, besting direct Grover's search for $k > 4$. However, the presented algorithm does even better. For $k = 3$ (triangle finding) the recursive algorithm given above can find a triangle with $O(N^{1.3})$ calls to the oracle (although the lower bound for this problem is so far only $O(N)$ - see Example 10). For $k > 3$, its complexity goes as $O\big(N^{\frac{2(k-1)}{k}}\big)$, but again, it is not known whether this is the optimum.

---

[35]From this point on oracle$_1$ is again called only "the oracle".

Note that all these algorithms were presented under the assumption that there is exactly one solutions or none at all. If there would be more solutions, we could use the approach given in [38], which preserves the efficiencies and speedups over the classical case. There is a number of discrete quantum walk algorithms that are more efficient than the best possible classical ones and are based on the algorithm of k-subset finding. Two of the examples are the verification of matrix products [41] and testing the commutativity of a black-box group [42].

4.8   Summary

In this Chapter, we explored the potential of discrete time quantum walks, focusing on search algorithms. We have seen they offer enough potential for devising new and more efficient algorithms for several problems — searches on hypercube [23], complete graph [45], lattices [43] or for anomalies in symmetry [49], collision problem [38], finding triangles in graphs [40], verifying matrix products [41] or testing the commutativity of a black-box group [42].

Viewed in a slightly abstract manner, quantum walks can be used in many oracle problems spanning from unstructured search to searching for graph substructures. Separating the oracle (which holds information about a set of elements) and the graph underlying the quantum walk allows one to make good choices resulting in clever algorithms as the one for k-subset finding. The simpler oracle algorithms can be subsequently employed as subroutines in more elaborate algorithms. We have looked at k-subset finding, where the search is performed on subsets of the set of elements rather then on elements themselves. This algorithm was used as a subroutine in the algorithm that finds k-cliques in a graph.

5 Quantizing Markov Chains

In this Chapter, we will look at how to obtain discrete quantum walks from *any Markov Chain*, which will result in a quantum speedup for many classical algorithms. It is hard to quantize Markov chains that are not regular. In particular, we would have to define a different "coin" at each vertex, which presents encoding difficulties. That's why Szegedy [51] took a different approach, using the state of the quantum walk itself as the basis for diffusion instead of an external coin register. Taking this route again results in a quantum walk that has some kind of "memory", as the unitarity of each transformation implies dependence on where we came from, in contrast to classical random walks.

We start with the definition of quantum walks on systems with two registers, analyze their spectra and prove some speedup results for hitting times. We then turn our attention to sampling (Monte-Carlo Markov Chain) algorithms. Finally, we take a look at quantum Metropolis sampling [56, 57].

5.1 Walks on Two Registers

Let us recall a discrete-time quantum walk on a regular degree-d graph which uses the Grover coin C_G. The state of the system is contained in two registers (vertex, coin), with Hilbert space of dimension Nd. When we start walking at vertex $|x\rangle$ with the coin in the state $|c\rangle$, a step of the walk results in

$$|x\rangle\,|c\rangle_{coin} \xrightarrow{\text{diffuse}} |x\rangle \sum_{\tilde{c}=1}^{d}(C_G)_{c\tilde{c}}\,|\tilde{c}\rangle_{coin} \xrightarrow{\text{shift}} \sum_{\tilde{c}=1}^{d}(C_G)_{c\tilde{c}}\,|x\oplus\tilde{c}\rangle\,|\tilde{c}\rangle_{coin}\,, \qquad (5.1)$$

a superposition over the neighbors of $|x\rangle$ in the vertex register, with corresponding states of the coin in the coin register. In Section 3.3.1, we viewed it as a scattering quantum walk in a system with two registers containing a target vertex and a source vertex. The Hilbert space for such a walk has dimension $\mathbb{C}^N \otimes \mathbb{C}^N$. When a scattering walk starts in the state that "moves" from vertex $|x\oplus c\rangle$ towards the vertex $|x\rangle$, one step brings it to

$$|x\rangle\,|x\oplus c\rangle \xrightarrow{\text{diffuse}} |x\rangle \sum_{\tilde{c}=1}^{d}(C_G)_{c\tilde{c}}\,|x\oplus\tilde{c}\rangle \xrightarrow{\text{swap registers}} \sum_{\tilde{c}=1}^{d}(C_G)_{c\tilde{c}}\,|x\oplus\tilde{c}\rangle\,|x\rangle\,, \qquad (5.2)$$

a superposition of states originating in $|x\rangle$ and "going" towards the neighbors of $|x\rangle$. Note that the diffusion in (5.1) and (5.2) is governed by the same $d \times d$ diffusion matrix C_G. It turns out we can express the unitary operator for the Grover diffusion step as a reflection operator on the whole two-register Hilbert space. Let us define the state

$$|\alpha_x\rangle \quad = \quad |x\rangle \sum_{\tilde{c}=1}^{d} \frac{1}{\sqrt{d}}\,|x\oplus\tilde{c}\rangle\,. \qquad (5.3)$$

The reflection of the state $|x\rangle \, |x \oplus c\rangle$ about $|\alpha_x\rangle$ is

$$(2 \, |\alpha_x\rangle \, \langle \alpha_x| - \mathbb{I}) \, |x\rangle \, |x \oplus c\rangle \quad = \quad \frac{2}{\sqrt{d}} \, |\alpha_x\rangle - |x\rangle \, |x \oplus c\rangle \tag{5.4}$$

$$= \quad |x\rangle \left[\left(\frac{2}{d} - 1 \right) |x \oplus c\rangle + \frac{2}{d} \sum_{\tilde{c} \neq c} |x \oplus \tilde{c}\rangle \right] \tag{5.5}$$

$$= \quad |x\rangle \sum_{\tilde{c}=1}^{d} (C_G)_{c\tilde{c}} \, |x \oplus \tilde{c}\rangle \,, \tag{5.6}$$

recalling the definition of the Grover coin C_G from (3.25), where we analyzed coins for discrete-time walks. Not restricting ourselves to a specific vertex x, we now define the projector

$$\Pi_G = \sum_x |\alpha_x\rangle \, \langle \alpha_x| \,. \tag{5.7}$$

A generalization of the above computation shows that the reflection $2\Pi_G - \mathbb{I}$ is exactly the diffusion step on the whole two-register space of our scattering quantum walk. The scattering quantum walk with the Grover coin on a d-regular graph can thus conveniently be written as

$$W_G = S(2\Pi_G - \mathbb{I}), \tag{5.8}$$

where $S = \sum_{x,y} |x, y\rangle \, \langle y, x|$ is an operator swapping the two registers. Such two-register scattering quantum walks (SQW) on regular graphs have a connection to classical walks whose next step chooses uniformly at random among the neighbors of each vertex. It remains a problem to concisely describe quantum walks that correspond to asymmetric classical walks, or to walks with a general stochastic[36] transition matrix $P_{x,y}$.

In the case the diffusion is different at different vertices, we will again utilize a system with two-registers (vertex,vertex), related to scattering quantum walks. Using Szegedy's generalization [51], instead of reflecting about the uniform states $|\alpha_x\rangle$, we now define a state

$$|\phi_x\rangle \quad = \quad |x\rangle \otimes \left(\sum_{y \in X} \sqrt{P_{x,y}} \, |y\rangle \right) \tag{5.9}$$

for every vertex x. We then choose the unitary for the diffusion to be a reflection about the subspace spanned by all the states $|\phi_x\rangle$, i.e.

$$R_1 = 2 \left(\sum_{x \in X} |\phi_x\rangle \, \langle \phi_x| \right) - \mathbb{I}. \tag{5.10}$$

As for the SQW, the diffusion step is followed by S, a swap of the registers. Applying this twice, we get Szegedy's quantization of a Markov chain with a transition matrix P. The (composed) step of the walk is then

$$W = SR_1 SR_1 = R_2 R_1, \tag{5.11}$$

---

[36] each row sums to 1

a product of two reflections, with R_2 given by

$$|\psi_y\rangle \quad = \quad \left(\sum_{x \in X} \sqrt{P_{x,y}}\,|x\rangle\right) \otimes |y\rangle, \tag{5.12}$$

$$R_2 \quad = \quad 2\left(\sum_{y \in Y} |\psi_y\rangle \langle\psi_y|\right) - \mathbb{I}. \tag{5.13}$$

We measure the position of the walker in a two-register state $|\psi\rangle$ by measuring only the first register. The probability of finding the walker at vertex x is thus

$$p(x) = \langle\psi|\,(|x\rangle \langle x| \otimes \mathbb{I})\,|\psi\rangle. \tag{5.14}$$

Note that for a symmetric Markov chain (with $P_{x,y} = P_{y,x}$), the only state invariant under both of these reflections (and thus under $W = R_2 R_1$) is

$$|\psi\rangle = \frac{1}{\sqrt{N}}\sum_{x \in X} |\phi_x\rangle = \frac{1}{\sqrt{N}}\sum_{y \in Y} |\psi_y\rangle = \frac{1}{\sqrt{N}}\sum_{x \in X}\sum_{y \in Y}\sqrt{P_{x,y}}\,|x\rangle\,|y\rangle, \tag{5.15}$$

a state with probability $p(x) = \frac{1}{\sqrt{N}}$ for each x (because $\sum_{y \in Y} P_{x,y} = 1$). This quantum state corresponds to the uniform-superposition stationary state of the classical Markov chain, and obeys the detailed balance equations

$$P_{x,y}\,p(x) = P_{y,x}\,p(y). \tag{5.16}$$

5.2   The Spectrum of the Walk

We have written the quantum walk (5.11) as a product of two reflections. Let us investigate the "mixing" properties of this type of walk and how we can connect them to the mixing properties of the original Markov Chain.

In many classical algorithms, Markov Chains are used for their fast mixing towards their stationary states. Do quantum walks bring anything new to the table? In what sense do unitary walks mix? We have discussed mixing previously in Section 3.5.1, where we defined mixing towards a limiting distribution in a time-averaged sense. We now want to look at how fast (and how close) we are getting towards the state that encodes the stationary distribution. For this, we will need to understand the spectrum of the quantum walk unitary operator. Let us start by investigating the action of two reflections. It will be helpful to recall Jordan's lemma.

Lemma 1 (Jordan '75) *For any two Hermitian projectors Π_1 and Π_2, there exists an orthogonal decomposition of the Hilbert space into one dimensional and two dimensional subspaces that are invariant under both Π_1 and Π_2. Moreover, inside each two-dimensional subspace, Π_1 and Π_2 are rank-one projectors.*

The two reflections R_1 and R_2 reflect around the subspaces defined by the projectors Π_1 and Π_2. Jordan's lemma implies that we can rewrite the Hilbert space as an orthogonal sum of 1D and 2D subspaces invariant under Π_1 and Π_2. Consequently, these subspaces are also invariant under

the reflections R_1 and R_2. Moreover, within each 2D subspace, two reflections compose into a rotation.

Szegedy proved a spectral theorem for the quantum walk $W = R_2 R_1$. We now present a slightly different version, including the 1D and 2D invariant space decomposition intuition from Jordan's lemma.

Theorem 4 *Consider two Hermitian projectors Π_1, Π_2 and the identity operator $\mathbb{I}$. The unitary operator $(2\Pi_2 - \mathbb{I})(2\Pi_1 - \mathbb{I})$ has eigenvalues $e^{\pm i 2\theta_j}$, $0 < \theta_j < \frac{\pi}{2}$ in the two-dimensional subspaces S_i invariant under Π_1 and Π_2, and it has eigenvalues ± 1 in the one-dimensional subspaces invariant under Π_1 and Π_2.*

The following proof comes from [70]. The 1D invariant subspaces are spanned by the common eigenvectors of Π_1 and Π_2, making the product of two reflections either $\mathbb{I}$ or $-\mathbb{I}$. What is more interesting, in each 2D subspace invariant under the two projectors, a product of two reflections is a rotation by $2\theta_j$, the angle between the axes of the two bases of the 2D subspace related to Π_1 and Π_2. We can find the Hilbert space decomposition and the angles θ_j with the help of the Hermitian *discriminant matrix* with entries

$$D_{x,y} = \sqrt{P_{x,y} P_{y,x}} \tag{5.17}$$

of the Markov chain. In the case of a symmetric chain, it is equal to the matrix P itself. Let the eigenvectors of D be $|\lambda_j\rangle$ with eigenvalues λ_j. Define an isometry from $\mathbb{C}^n$ to $\mathbb{C}^n \otimes \mathbb{C}^n$ by

$$T = \sum_x |\psi_x\rangle \langle x| = \sum_{x,y} \sqrt{P_{x,y}} |x\rangle |y\rangle \langle x|. \tag{5.18}$$

We claim that each two-dimensional invariant subspaces can be constructed as $\mathrm{span}\{|\tilde\lambda_j\rangle, S|\tilde\lambda_j\rangle\}$, where the vectors $|\tilde\lambda_j\rangle = T|\lambda_j\rangle$ come from the eigenvectors of D with the help of the isometry T. To see that this is indeed so, we first observe that

$$TT^\dagger = \sum_{x,y} |\psi_x\rangle \langle x|y\rangle \langle \psi_y| = \sum_x |\psi_x\rangle \langle \psi_x| = \Pi_1, \tag{5.19}$$

$$T^\dagger T = \sum_{x,y} |x\rangle \langle \psi_x|\psi_y\rangle \langle y| = \sum_{x,y,w,z} \sqrt{P_{x,w} P_{y,z}} |x\rangle \langle x|y\rangle \langle w|z\rangle \langle y|$$

$$= \sum_{x,w} P_{x,w} |x\rangle \langle x| = \sum_x |x\rangle \langle x| = \mathbb{I}, \tag{5.20}$$

$$T^\dagger S T = \sum_{x,y} |x\rangle \langle \psi_x| S |\psi_y\rangle \langle y| = \sum_{x,y,z,w} \sqrt{P_{x,z}} \sqrt{P_{y,w}} \langle x| \langle x| \langle z| S |y\rangle |w\rangle \langle y|$$

$$= \sum_{x,y} \sqrt{P_{x,y} P_{y,x}} |x\rangle \langle y| = D, \tag{5.21}$$

and then check that each subspace $\text{span}\{|\tilde{\lambda}_j\rangle, S|\tilde{\lambda}_j\rangle\}$ is indeed invariant:

$$
\begin{aligned}
R_1|\tilde{\lambda}_j\rangle &= (2\Pi_1 - \mathbb{I})T\,|\lambda_j\rangle = T\,|\lambda_j\rangle = |\tilde{\lambda}_j\rangle, & (5.22) \\
R_1 S|\tilde{\lambda}_j\rangle &= (2\Pi_1 - \mathbb{I})ST\,|\lambda_j\rangle = (2TT^\dagger - \mathbb{I})ST\,|\lambda_j\rangle \\
&= (2TD - ST)\,|\lambda_j\rangle = 2\lambda_j|\tilde{\lambda}_j\rangle - S|\tilde{\lambda}_j\rangle, & (5.23) \\
R_2|\tilde{\lambda}_j\rangle &= (SR_1 S)|\tilde{\lambda}_j\rangle = 2\lambda_j S|\tilde{\lambda}_j\rangle - |\tilde{\lambda}_j\rangle, & (5.24) \\
R_2 S|\tilde{\lambda}_j\rangle &= SR_1 SS|\tilde{\lambda}_j\rangle = SR_1|\tilde{\lambda}_j\rangle = S|\tilde{\lambda}_j\rangle. & (5.25)
\end{aligned}
$$

This in turn helps us relate the eigenvalues of $W = R_2 R_1$ to the eigenvalues of D. Jordan's lemma implies the action of W is a rotation in each 2D subspace, so its eigenvalues are $e^{\pm i\theta_j}$, where θ_j is the angle between the $+1$ eigenvectors of Π_1 and Π_2 in this subpspace. Let us calculate these θ_j. First, recall that $\Pi_1|\tilde{\lambda}_j\rangle = |\tilde{\lambda}_j\rangle$ and $\Pi_2 = S\Pi_1 S$, which means that $T^\dagger \Pi_2 T = T^\dagger STT^\dagger ST = D^2$. Thus

$$
\cos\theta_j = \frac{\langle\tilde{\lambda}_j|\Pi_2\Pi_1|\tilde{\lambda}_j\rangle}{\sqrt{\langle\tilde{\lambda}_j|\Pi_2|\tilde{\lambda}_j\rangle\langle\tilde{\lambda}_j|\Pi_1|\tilde{\lambda}_j\rangle}} = \sqrt{\langle\lambda_j|T^\dagger\Pi_2 T|\lambda_j\rangle} = \sqrt{\langle\lambda_j|D^2|\lambda_j\rangle} = \lambda_j. \quad (5.26)
$$

The eigenvalues of W are thus ± 1 in the 1D invariant subspaces and

$$
e^{\pm i2\theta_j} = e^{\pm i2\arccos\lambda_j} = (2\lambda_j^2 - 1) \pm i2\lambda_j\sqrt{1 - \lambda_j^2}, \quad (5.27)
$$

in the 2D invariant subspaces (this is a form seen in Szegedy's papers).

5.3 Speeding up Searching for Marked Vertices

With an understanding of the spectrum of the quantum walk unitary, we now turn our attention to possible square root speedups over its classical counterpart. While the number of necessary classical Markov Chain steps depends on the gap δ of the classical chain as $f(\epsilon, \delta)$. Szegedy has shown that the quantum walk will need to only take $f(\sqrt{\epsilon}, \sqrt{\delta})$ steps. The speedups thus appear in shorter hitting times when searching, and in the later sections we will see them in shorter required times to get within distance ϵ of the stationary distribution.

Let us look at a rather generic search algorithm based on a Markov chain P (with spectral gap δ) on set X, with a marked subset M. The goal is to find one of the vertices of the subset M. We will see that a quantum version of this algorithm has a "square-root speedup" over its classical counterpart. This section also follows [70].

First, we start with a classical algorithm, and modify the Markov chain so that it stays in a marked vertex $x \in M$ once we hit it. For this, we choose the transition matrix to be

$$
P'_{x,y} = \begin{cases} 0 & x \in M, x \neq y, \\ 1 & x \in M, x = y, \\ P_{x,y} & x \notin M. \end{cases} \quad (5.28)
$$

It has block form

$$
P' = \begin{bmatrix} P_M & 0 \\ B & \mathbb{I} \end{bmatrix}, \quad (5.29)
$$

where P_M corresponds to the rows/columns of the original P that are not marked (in M). We will now show that when the fraction of marked vertices is $\frac{|M|}{N} \leq \epsilon$ and the second largest eigenvalue of P is lower than $(1-\delta)$, the classical hitting time is lower bounded by $O\left(\frac{1}{\delta\epsilon}\right)$. The decision problem (is there a marked vertex at all) complexity will also have the same lower bound.

First, let us analyze the classical chain. When we take t steps of the walk, we get

$$(P')^t = \begin{bmatrix} P_M^t & 0 \\ B + BP_M + BP_M^2 + \cdots + BP_M^t & \mathbb{I} \end{bmatrix} = \begin{bmatrix} P_M^t & 0 \\ B\frac{P_M^t - \mathbb{I}}{P_M - \mathbb{I}} & \mathbb{I} \end{bmatrix}. \tag{5.30}$$

We start in the uniform distribution over unmarked vertices (by choosing a random unmarked vertex as the starting point). Denoting $|o\rangle = \frac{1}{\sqrt{N-|M|}} \sum_{x \notin M} |x\rangle$, we can express the probability of *not* reaching a marked vertex after t steps as

$$p_t^- = \langle o| P_M^t |o\rangle \leq \left\| P_M^t \right\| = \| P_M \|^t, \tag{5.31}$$

using the operator norm (largest eigenvalue) of the matrix. When $\| P_M \| \leq 1 - \Delta$,

$$p_t^+ = 1 - p_t^- \geq 1 - \| P_M \|^t \geq 1 - (1-\Delta)^t. \tag{5.32}$$

It is then enough to take $t = O\left(\frac{1}{\Delta}\right)$ to ensure an $\Omega(1)$ success probability.

Finally, we can relate $\| P_M \|$ to the second largest eigenvalue $(1-\delta)$ of the original transition matrix P, and the fraction $\frac{|M|}{N} \leq \epsilon$ of marked vertices. The original matrix P is symmetric, so its principal eigenvector with eigenvalue 1 is $|s\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$. Let $|w\rangle \in \mathbb{C}^N$ be the principal eigenvector of the matrix P_M (which has size $N-M$), padded with zeros on the marked vertices.

Using the Cauchy-Schwartz inequality, we can upper bound the overlap

$$|\langle s|w\rangle|^2 = |\langle s| \Pi_{x \notin M} |w\rangle|^2 \leq \| \Pi_{x \notin M} |s\rangle \| \cdot \| |w\rangle \|^2 = \frac{N - |M|}{N} = 1 - \epsilon. \tag{5.33}$$

Next, using the eigenvectors $|\lambda\rangle$ of P, we can express

$$|w\rangle = \langle s|w\rangle |s\rangle + \sum_{\lambda \neq 1} \langle \lambda|w\rangle |\lambda\rangle, \tag{5.34}$$

$$\| P_M \|^2 = \| P |w\rangle \|^2 = |\langle s|w\rangle|^2 + \sum_{\lambda \neq 1} |\langle \lambda|w\rangle|^2 \lambda^2 \tag{5.35}$$

$$\leq |\langle s|w\rangle|^2 + (1-\delta)^2 \sum_{\lambda \neq 1} |\langle \lambda|w\rangle|^2 \tag{5.36}$$

$$= |\langle s|w\rangle|^2 [1 - (1-\delta)^2] + (1-\delta)^2 \sum_{\text{all } \lambda} |\langle \lambda|w\rangle|^2 \tag{5.37}$$

$$\leq (1-\epsilon)(2\delta - \delta^2) + (1-\delta)^2 \tag{5.38}$$

$$\leq 1 - 2\epsilon\delta, \tag{5.39}$$

because the first eigenvalue of P is 1 and all the other eigenvalues are upper bounded by $(1-\delta)$. Therefore, $\| P_M \| \leq \sqrt{1 - 2\delta\epsilon} \leq 1 - \delta\epsilon$. Together with (5.32), this means we need to take

$O\left(\frac{1}{\delta\epsilon}\right)$ steps of the walk to get a constant success probability for this classical random walk search algorithm.

Let us now look at a quantum walk algorithm for the same task. If we start in a uniform superposition and measure whether we get a marked vertex, success is unlikely and we end up in a uniform superposition over unmarked vertices. However, we can now perform phase estimation of the unitary quantum walk W instead. We have seen how we can relate its eigenvalues to the discriminant matrix, which in this case is

$$D = \left[\begin{array}{cc} P_M & 0 \\ 0 & \mathbb{I} \end{array} \right]. \tag{5.40}$$

If there are no marked vertices, $|o\rangle$ (5.31) is the principal eigenvector of the quantum walk W with eigenvalue one, and phase estimation [71] of W returns 0. On the other hand, if marked vertices exist, $|o\rangle$ lives in the "busy" subspace of the quantum walk (corresponding to eigenvalues $e^{\pm i2\theta_j}$, where $\lambda_j^M = \cos\theta_j$ are the eigenvalues of P_M). Phase estimation of W would thus return a phase greater than

$$\phi_0 = 2\min_j |\theta_j| \geq 2\min_j \arccos \lambda_j^M \geq 2\min_j \sqrt{1 - \lambda_j^M} = 2\sqrt{1 - \|A\|} \geq 2\sqrt{\delta\epsilon}. \tag{5.41}$$

Phase estimation of a unitary W with precision ϕ_0 takes $O\left(\phi_0^{-1}\right)$ evaluations of W, so a quantum walk algorighm deciding between no marked vertices and a fraction of ϵ marked vertices will take $O\left(\frac{1}{\sqrt{\delta\epsilon}}\right)$ steps, a quadratic speedup over its classical counterpart described above. This is the result of Szegedy's $\delta\epsilon$-paper [51].

Note that we did not claim the classical algorithm was optimal for a particular search problem – we only compared classical and quantum versions of these generic approaches. Furthermore, we have analyzed a quantum walk algorithm that only *decides* whether marked vertices exist, and does not output one. However, by adding an additional layer of "marking" vertices, we could obtain a marked-vertex-identifying algorithm with at most logarithmic overhead. On the other hand, sampling from the set of marked vertices (or according to some probability distribution) is a different and interesting problem, and we will look at it in the following Sections.

5.4   Walks and Sampling

We have seen that quantization of classical Markov chains has been crucial in the design of efficient quantum algorithms for a wide range of search problems [52] that outperform their classical counterparts. We now extend the scope of use of quantum walks (quantized Markov chains) beyond search problems. They can be employed to speed up sampling from probability distributions. This results in a variety of quantum algorithms, including quantum simulated annealing, fully polynomial-time quantum approximation schemes for partition functions, and the quantum Metropolis algorithm.

Sampling from the stationary distributions of Markov Chains is a strong classical algorithmic tool, useful for counting (#P) problems, as described in Section 5.4.2. Classically, Aldous has shown [65] that the mixing time (the number of steps guaranteeing closeness to the stationary distribution) for a Markov Chain is related to its spectral gap δ and the minimum of the distribution π_* as $O(\delta^{-1}\log 1/\pi_*)$. The question is whether we can do better using quantum walks. Richter [72], introducing decoherence into quantum walks generated a classically converging

Markov Chain, and proved that on a periodic lattice $\mathbb{Z}_n^d$, the mixing time gets a square root speedup (with respect to the spectral gap) to $O(\sqrt{\delta^{-1}} \log 1/\pi_*)$. Another interesting application of quantization of classical MC's is simulated annealing which involves Markov Chains whose stationary distributions correspond to Gibbs distributions at particular temperatures. Bringing quantum mechanics to the picture, Somma et al. [60] proposed a quantum simulation of classical annealing, using quantized Markov chains, and relying on the quantum Zeno effect and phase estimation. They sequentially prepare coherent state encodings of the stationary distribution of the MC's, using a simulation of projective measurements. This results in a quantum algorithm with an $O(\sqrt{\delta^{-1}})$ dependence on the spectral gap (again a square root speedup), with some additional factors. We discuss these two approaches to sampling in Section 5.4.1, describe the Monte-Carlo Markov Chain (MCMC) method in detail in 5.4.2, look at a general approach to its quantization [69] in 5.4.3 and showcase some of its applications and speedups. We then turn our attention to the quantum Metropolis algorithm [56] in Section 5.4.4.

5.4.1   Speeding up Mixing Using Quantum Walks

We have seen that sampling from stationary distributions of Markov Chains has very useful applications. How could one speed up the preparation of these states using quantum walks? Unitarity prohibits us from talking about mixing in the classical sense. However, according to Richter [72] we can use a quantum walk to get a random process by combining

1. time evolution according to the transition rule U (discrete with U^t or continuous with $U(t)$), and

2. a measurement in the computational basis, evaluated at a time chosen randomly according to a measurement rule, given by a probability distribution ω_T (e.g. the uniform distribution $\bar{\mu}_T = \frac{1}{T} \chi_{[0,T]}$, the delta-function distribution $\delta_T(t) = \delta(t - T)$, or other).

Thus, we can view quantum walks as the pair $\langle U, \omega_T \rangle$. Repeating the unitary evolution and random-time measurement many times gives us an algorithm which probabilistically outputs[37] a vertex – the probability distribution for this output vertex converges to a stationary distribution just as its classical Markov Chain counterpart did. Let us ask how fast this happens.

   The Aldous theorem for a classical reversible, ergodic Markov Chains with stationary distribution π and spectral gap δ states that the mixing time obeys

$$\frac{1}{\delta} \leq \tau_{mix} \leq \frac{1}{\delta} \log \frac{1}{\pi_*}, \tag{5.42}$$

where $\pi_* = \min_x \pi(x)$. Could we speed up this dependence on the spectral gap in a square root fashion by using quantum walks? Richter [72] showed that on a periodic lattice $\mathbb{Z}_n^d$, the mixing time for the quantum decohering walk gains a square root speedup (with respect to the spectral gap), as it converges with mixing time $O(\sqrt{\delta^{-1}} \log 1/\pi_*)$. It remains open whether this speedup can be achieved in general, for (m)any quantized Markov Chains.

   Classical simulated annealing [12, 13] imitates the process where a metal is heated to a high temperature and then slowly cooled down. This is supposed to allow thermal excitations to jump

---

[37]For each sample, we pick some simple initial state (or the previously measured vertex), let the quantum walk run for some time, and measure. With this measurement, the superposition randomly collapses to a vertex, producing a sample.

out of local minima, letting the system end up in a low-energy state at the end of the process. The thermalization is modeled by a Markov Chain, whose stationary distribution corresponds to the Gibbs distribution at a given temperature. Again, we recall the Aldous theorem which says the mixing time for a MC scales with the spectral gap δ as $O(\delta^{-1})$. Could we speed this process up using quantized Markov Chains? In [60], Somma et al. use a sequence of quantized (two-register) Markov chains, but instead of preparing probability distributions π_i, they look at coherent states $|\pi_i\rangle = \sum_x \sqrt{\pi_i(x)} |x\rangle$ encoding them. When we have access to the state $|\pi_i\rangle$, could we use it to prepare the next state $|\pi_{i+1}\rangle$? As we lower the temperature slowly, the states are close. Thus, it is likely that a projection onto $|\pi_{i+1}\rangle$ would succeed. This projective measurement is simulated by phase estimating the walk operator W_{i+1} for the next quantum MC, as $|\pi_{i+1}\rangle$ is its eigenvector with eigenvalue 1. The quantum Zeno effect [71] and large overlaps $\langle \pi_i | \pi_{i+1} \rangle$ are responsible for the high success probability of each step, and consequently, the overall procedure. The resulting algorithm [60] requires $O(\sqrt{\delta^{-1}})$ implementations of a step of a quantum MC (with some additional factors).

5.4.2   Markov Chain Monte Carlo (MCMC) Methods

Sampling from stationary distributions of a sequence of Markov chains, combined with simulated annealing (progressive lowering of a temperature parameter) lies at the heart of many important classical approximation algorithms. These methods are in general called *Markov Chain Monte Carlo* (MCMC). Some out of the many examples include the approximation algorithms for the volume of convex bodies [61], the permanent of a non-negative matrix [62], and the partition function of statistical physics models such as the Ising model [63] and the Potts model [64]. Each of these algorithms is a *fully polynomial randomized approximation scheme* (FPRAS), outputting a random number $\hat{Z}$ within a factor of $(1 \pm \epsilon)$ of the real value Z, with probability greater than $\frac{3}{4}$, i.e.

$$\Pr\left[(1 - \epsilon)Z \leq \hat{Z} \leq (1 + \epsilon)Z\right] \geq \frac{3}{4}, \tag{5.43}$$

in a number of steps[38] polynomial in $1/\epsilon$ and the problem size.

The classical algorithms can be outlined as follows. Consider a physical system with state space Ω and an energy function $E : \Omega \to \mathbb{R}$, assigning each state $\sigma \in \Omega$ an energy $E(\sigma)$. Our task is to estimate the Gibbs partition function

$$Z(T) = \sum_{\sigma \in \Omega} e^{-\frac{E(\sigma)}{kT}} \tag{5.44}$$

at a low final temperature T_F. The value of Z at zero temperature is interesting, as it is equal to the number of the system configurations with zero energy[39]. This could be a solution to a counting problem, such as providing us the number of graphs with a particular property.

---

[38]Note that these FPRAS can not be used to *solve* counting problems when Z is exponentially large (in the problem size). The precision ϵ would then have to be exponentially small, and the number of required algorithmic steps would grow exponentially.

[39]This relationship is used e.g. in the algoritm [62] for approximating the permanent of a non-negative matrix – one can find the value of the permanent by counting the number of perfect matchings of a particular bipartite graph, which in turn is equal to the zero-temperature partition function of a certain spin system.

The partition function $Z(T)$ encodes the thermodynamical properties of the system in equilibrium at temperature T, where the probability of finding the system in state σ is given by the Boltzmann distribution

$$\pi(\sigma) = \frac{1}{Z(T)} e^{-\frac{E(\sigma)}{kT}}. \tag{5.45}$$

It is hard to estimate $Z(T)$ directly – using a single Markov Chain (whose stationary distribution is a low temperature Gibbs state) and letting it thermalize could take a very long time. The schemes we want to speed up thus attempt to reach the low-temperature thermal states in several stages. Consider a sequence of decreasing temperatures $T_0 \geq T_1 \geq \cdots \geq T_\ell$, where T_0 is a very high starting temperature and $T_\ell = T_F$ is the desired final temperature. The final partition function $Z(T_F)$ can then be expressed as a telescoping product

$$Z(T_F) \quad = \quad Z_0 \frac{Z_1}{Z_0} \cdots \frac{Z_{\ell-1}}{Z_{\ell-2}} \frac{Z_\ell}{Z_{\ell-1}} = Z_0 \underbrace{(\alpha_0 \alpha_1 \cdots \alpha_{\ell-2} \alpha_{\ell-1})}_{\alpha}, \tag{5.46}$$

where $Z_i = Z(T_i)$ stands for the Gibbs partition function at temperature T_i and $\alpha_i = Z_{i+1}/Z_i$ is the ratio of successive Z's. To start, it is easy to calculate the partition function $Z_0 = Z(T_0)$ at high temperature (its value is the volume of the state space). Next, for each i, we can estimate the ratio α_i by sampling from a distribution that is sufficiently close to the Boltzmann distribution π_i (5.45) at temperature T_i. This is possible by using a rapidly-mixing Markov chain P_i whose stationary distribution is equal to the Boltzmann distribution π_i.

To be efficient, these classical schemes require that

1. we use a cooling schedule such that the resulting ratios $\alpha_i = Z(T_{i+1})/Z(T_i)$ are lower bounded by a constant c^{-1} (to simplify the presentation, we use $c = 2$ from now on),

2. the spectral gaps of the Markov chains P_i are bounded from below by δ.

The time complexity of such FPRAS, i.e., the number of times we have to invoke an update step for a Markov chain from $\{P_1, \ldots, P_{\ell-1}\}$, is

$$\tilde{O}\left(\frac{\ell^2}{\delta \cdot \epsilon^2}\right), \tag{5.47}$$

where $\tilde{O}$ means up to logarithmic factors.

We will now follow the presentation in [64, Section 2.1] and describe the classical approximation schemes in more detail, starting with the partition function as a telescoping product (5.46). At $T_0 = \infty$, the partition function Z_0 is equal to

$$Z_0 = |\Omega|, \tag{5.48}$$

the size of the state space. On the other hand, for each $i = 0, \ldots, \ell-1$, we can estimate the ratio

$$\alpha_i = \frac{Z_{i+1}}{Z_i} \tag{5.49}$$

in (5.46) by sampling from the Boltzmann distribution π_i as follows. Let $X_i \sim \pi_i$ denote a random state chosen according to the Boltzmann distribution π_i, i.e.,

$$\Pr(X_i = \sigma) = \pi_i(\sigma). \tag{5.50}$$

Define a new random variable Y_i by

$$Y_i = e^{-(\beta_{i+1} - \beta i)\, E(X_i)}, \tag{5.51}$$

where $\beta_i = (kT_i)^{-1}$ is the inverse temperature (k is the Boltzmann constant). This Y_i is an unbiased estimator for α_i since

$$\mathbf{E}\left(Y_i\right) \;=\; \sum_{\sigma \in \Omega} \pi_i(\sigma)\, e^{-(\beta_{i+1} - \beta i)\, E(\sigma)} = \sum_{\sigma \in \Omega} \frac{e^{-\beta_i E(\sigma)}}{Z_i}\, e^{-(\beta_{i+1} - \beta i)\, E(\sigma)} \tag{5.52}$$

$$\;=\; \sum_{\sigma \in \Omega} \frac{e^{-\beta_{i+1} E(\sigma)}}{Z_i} = \frac{Z_{i+1}}{Z_i} = \alpha_i. \tag{5.53}$$

Assume now that we have an algorithm for generating states X_i according to π_i. We draw $m := 64\ell/\epsilon^2$ samples of X_i and take the mean $\overline{Y}_i$ of their corresponding estimators Y_i. Then, assuming $\frac{1}{2} \le \alpha_i \le 1$, the mean $\overline{Y}_i$ satisfies

$$\frac{\mathbf{Var}\left(\overline{Y}_i\right)}{\left(\mathbf{E}\left(\overline{Y}_i\right)\right)^2} = \frac{\epsilon^2}{64\ell}\, \frac{\mathbf{Var}\left(Y_i\right)}{\left(\mathbf{E}\left(Y_i\right)\right)^2} \le \frac{\epsilon^2}{16\ell}. \tag{5.54}$$

We can now compose such estimates of α_i. Define a new random variable $\overline{Y}$ by

$$\overline{Y} = \overline{Y}_{\ell-1}\overline{Y}_{\ell-2}\cdots\overline{Y}_0 \tag{5.55}$$

Since all $\overline{Y}_i$ are independent, we have

$$\mathbf{E}\left(\overline{Y}\right) \;=\; \mathbf{E}\left(Y_{\ell-1}\right)\mathbf{E}\left(Y_{\ell-2}\right)\cdots\mathbf{E}\left(Y_0\right) = \alpha_{\ell-1}\alpha_{\ell-2}\cdots\alpha_0 = \alpha, \tag{5.56}$$

Moreover, $\overline{Y}$ has the property

$$
\begin{aligned}
\frac{\mathbf{Var}\left(\overline{Y}\right)}{\left(\mathbf{E}\left(\overline{Y}\right)\right)^2} \;&=\; \frac{\mathbf{E}\left(\overline{Y}_{\ell-1}^2\right)\cdots\mathbf{E}\left(\overline{Y}_0^2\right) - \mathbf{E}\left(\overline{Y}_{\ell-1}\right)^2\cdots\mathbf{E}\left(\overline{Y}_0\right)^2}{\mathbf{E}\left(\overline{Y}_{\ell-1}^2\right)^2\cdots\mathbf{E}\left(\overline{Y}_0\right)^2} \\[2mm]
&=\; \left(1 + \frac{\mathbf{Var}\left(\overline{Y}_{\ell-1}\right)}{\left(\mathbf{E}\left(\overline{Y}_{\ell-1}\right)\right)^2}\right)\cdots\left(1 + \frac{\mathbf{Var}\left(\overline{Y}_0\right)}{\left(\mathbf{E}\left(\overline{Y}_0\right)\right)^2}\right) - 1 \\[2mm]
&\le\; \left(e^{\epsilon^2/16\ell}\right)^{\ell} - 1 \le \epsilon^2/8,
\end{aligned}
\tag{5.57}
$$

where we used $1 + x \le e^x$ (true for all x) and $e^x - 1 \le 2x$ (true for all $x \in [0,1]$) in the last two steps, respectively. Chebyshev's inequality now implies that the value of $\overline{Y}$ is in the interval $[(1-\epsilon)\alpha, (1+\epsilon)\alpha]$ with probability at least $\frac{7}{8}$.

Of course, we are not able to obtain perfect samples X_i from π_i. Assume now that we have X_i' that are from a distribution with a variation distance from π_i smaller than $d := \epsilon^2/(512\ell^2)$. Let $\overline{Y}'$ be defined as $\overline{Y}$ as above, but instead of X_i we use X_i'. Then, with probability at least $\frac{7}{8}$, we have $\overline{Y} = \overline{Y}'$. To derive this, observe that the algorithm can be thought to first take a sample from a product probability distribution π on the $(m\ell)$-fold direct product of Ω. We denote the probability distribution in the case of imperfect samples by π'. The total variation distance between π and π' is then bounded from above by

$$d \cdot m \cdot \ell = \frac{\epsilon^2}{512\ell^2} \cdot \frac{64\ell}{\epsilon^2} \cdot \ell = \frac{1}{8} \,. \tag{5.58}$$

Therefore, $\overline{Y}'$ is in the interval $[(1-\epsilon)\mathbf{E}\left(Y\right), (1+\epsilon)\mathbf{E}\left(Y\right)]$ with probability at least $\frac{3}{4}$.

We obtain the samples X_i' by applying rapidly mixing Markov chains P_i whose limiting distributions are equal to π_i. Constructing such Markov chains is a hard task, but it has been done for the Ising model [63] and the Potts model [64], resulting in FPRAS for the partition functions for these models.

Thus, using a sequence of rapidly-mixing Markov Chains, it is possible to prepare low-temperature Gibbs states. Not only that, sampling from these states and estimating the variable (5.51) allows us to estimate partition functions by a telescoping product (5.46), because of (5.53). Note that for each sample from a state close to the thermal state for a given MC, we need to prepare it using a sequence of MC's. The run is then discarded. Could we somehow "reuse" the mixing we have done so far using quantum mechanics? It turns out it is possible, and that estimating the ratios α_i using (5.51) is possible by phase estimation of a certain quantum walk operator. The method described in the next section, developed by Wocjan et al. in [68, 69] is related to the one used by Somma [60], and brings an additional square root speedup in the approximation precision parameter.

5.4.3 Quantizing MCMC Methods for Approximating Partition Functions

In Section 5.4.2, we have seen how classical MCMC methods work and that they require $\tilde{O}\left(\frac{\ell^2}{\delta \cdot \epsilon^2}\right)$ steps (5.47), where δ is the lower bound on the spectral gap of the MC's involved and ϵ is the desired approximation precision for the FPRAS. Let us now turn to the quantum world and put quantum walks to use, following [69]. We will see that the resulting quantized algorithm gets a square root speedup in both the parameter δ and ϵ. Let us summarize the result:

Theorem 5 *Consider a classical FPRAS for approximating the Gibbs partition function of a physical system at temperature T_F, satisfying the above conditions. Then, there exists a fully polynomial quantum approximation scheme that uses*

$$\tilde{O}\left(\frac{\ell^2}{\sqrt{\delta} \cdot \epsilon}\right) \tag{5.59}$$

applications of a controlled version of a quantum walk operator from $\{W(P_1), \ldots, W(P_{\ell-1})\}$.

The reduction in complexity (in comparison to the classical FPRAS (5.47)) for the quantum algorithm we will now look at is twofold. First, the factor $1/\delta$ is reduced to $1/\sqrt{\delta}$ by using

$$|\pi_0\rangle \longrightarrow \boxed{\diagup} \; \mathcal{M}_0 \quad (\text{obtain } \alpha_0)$$
$$|\pi_0\rangle \;\rightarrow\; |\tilde{\pi}_1\rangle \longrightarrow \boxed{\diagup} \; \mathcal{M}_1 \quad (\text{obtain } \alpha_1)$$
$$\vdots$$
$$|\pi_0\rangle \;\rightarrow\; |\tilde{\pi}_1\rangle \;\rightarrow\; \cdots \;\rightarrow\; |\tilde{\pi}_{\ell-1}\rangle \boxed{\diagup} \; \mathcal{M}_{\ell-1} \; (\text{obtain } \alpha_{\ell-1})$$
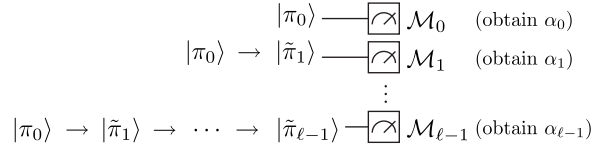
Fig. 5.1. Structure of the quantum algorithm.

quantum walks instead of classical Markov chains (for preparing the distributions to sample from), and utilizing the quadratic relation between spectral and phase gaps. This relation is the basis of success for many quantum search algorithms based on quantum walks [52]. Thus, instead of letting a Markov chain P_i mix towards its stationary distribution π_i, we choose to approximately prepare the state

$$|\pi_i\rangle = \sum_{\sigma \in \Omega} \sqrt{\pi_i(\sigma)} \, |\sigma\rangle \tag{5.60}$$

a *coherent encoding* of the Boltzmann distribution. The preparation method [68] is based on Grover's $\frac{\pi}{3}$-fixed-point search [53, 54], described in Appendix D, efficiently driving the state $|\pi_0\rangle$ towards the desired state $|\pi_i\rangle$ through a sequence of intermediate states.

Second, we speed up the way to determine the ratios α_i. Instead of using classical samples from the stationary distribution π_i of a Markov chain P_i, we approximate α_i by phase-estimating a certain unitary (related to quantum walks) on the state $|\pi_i\rangle$. This results in the reduction of the factor $1/\epsilon^2$ to $1/\epsilon$.

The structure of the algorithm is depicted in Fig. 5.1. It consists of successive approximate preparations of $|\pi_i\rangle$ followed by a quantum circuit outputting a good approximation to α_i (with high probability). We will now show how to quantize the the classical algorithm, assuming that we can take perfect samples X_i from π_i. The interested reader is invited to look into [69] for the full quantum algorithm which deals with the fact that samples from π_i can be taken only approximately, as the states we can prepare are only approximations (but good ones) to $|\pi_i\rangle$. However, the errors can be handled and collected in such a way, that the result of Theorem 5 holds, giving a fully quantum FPRAS with a double (in δ and ϵ) square root speedup over the classical method.

To estimate the ratios α_i in (5.46), the classical algorithm generates random states X_i from π_i and computes the mean $\overline{Y}_i$ of the random variables Y_i. The process of generating a random state X_i from π_i is equivalent to preparing the mixed state $\rho_i = \sum_{\sigma \in \Omega} \pi_i(\sigma)|\sigma\rangle\langle\sigma|$. Instead of this, we choose to prepare the pure states

$$|\pi_i\rangle = \sum_{\sigma \in \Omega} \sqrt{\pi_i(\sigma)}|\sigma\rangle \,. \tag{5.61}$$

We call these states *quantum samples* since they coherently encode the probability distributions π_i. Let us for now assume that we can prepare these exactly and efficiently.

The random variable Y_i can be viewed as the outcome of the measurement of the observable

$$A_i = \sum_{\sigma \in \Omega} y_i(\sigma)|\sigma\rangle\langle\sigma| \tag{5.62}$$

where $y_i(\sigma) = e^{-(\beta_{i+1}-\beta_i)E(\sigma)}$, for the state ρ_i. This interpretation implies that to estimate α_i classically, we need to estimate the expected value $\mathrm{Tr}(A_i\rho_i)$ by repeating the above measurement many times and outputting the mean of the outcomes. However, we can quantize this process. We add an ancilla qubit to our quantum system in which the quantum samples $|\pi_i\rangle$ live. For each $i = 0, \ldots, \ell - 1$, we define the unitary

$$V_i = \sum_{\sigma \in \Omega} |\sigma\rangle\langle\sigma| \otimes \begin{pmatrix} \sqrt{y_i(\sigma)} & \sqrt{1 - y_i(\sigma)} \\ -\sqrt{1 - y_i(\sigma)} & \sqrt{y_i(\sigma)} \end{pmatrix}. \tag{5.63}$$

This V_i can be efficiently implemented, as it is a rotation on the extra qubit controlled by the state of the first tensor component. Let us label

$$|\psi_i\rangle = V_i(|\pi_i\rangle \otimes |0\rangle). \tag{5.64}$$

Consider now the expected value of the projector

$$P = \mathbb{I} \otimes |0\rangle\langle 0| \tag{5.65}$$

in the state $|\psi_i\rangle$. We find

$$\langle\psi_i|P|\psi_i\rangle = \langle\pi_i|A_i|\pi_i\rangle = \alpha_i. \tag{5.66}$$

We now show how to speed up the process of estimating α_i with a method that generalizes quantum counting [55]. Assuming efficient preparation of $|\pi_i\rangle$ implies that we can also efficiently implement the reflections

$$R_i = 2|\pi_i\rangle\langle\pi_i| - \mathbb{I}. \tag{5.67}$$

Thus, we arrive at the existence of a quantum FPRAS for estimating the partition function, assuming efficient and perfect preparation of $|\pi_i\rangle$, summed in Theorem 6:

Theorem 6 *There is a fully polynomial quantum approximation scheme $\mathcal{A}$ for the partition function Z. Its output Q satisfies*

$$\Pr\left[(1-\epsilon)Z \leq Q \leq (1+\epsilon)Z\right] \geq \frac{3}{4}. \tag{5.68}$$

For each $i = 0, \ldots, \ell - 1$, the scheme $\mathcal{A}$ uses $O(\log \ell)$ perfectly prepared quantum samples $|\pi_i\rangle$, and applies the controlled-R_i operator $O\left(\frac{\ell}{\epsilon}\log \ell\right)$ times, where R_i is as in (5.67).

The proof of Theorem 6 builds on the following three technical results.

Lemma 2 (Quantum ratio estimation) *Let $\epsilon_{pe} \in (0,1)$. For each $i = 0, \ldots, \ell - 1$ there exists a quantum approximation scheme $\mathcal{A}'_i$ for α_i. Its output Q'_i satisfies $\Pr\left[(1-\epsilon_{pe})\alpha_i \leq Q'_i \leq (1+\epsilon_{pe})\alpha_i\right] \geq \frac{7}{8}$. The scheme $\mathcal{A}'_i$ requires one copy of the quantum sample $|\pi_i\rangle$ and invokes the controlled-R_i operator $O\left(\epsilon_{pe}^{-1}\right)$ times, where R_i is as in (5.67).*

We can boost the success probability of the above quantum approximation scheme for the ratio α_i by applying the *powering lemma* from [66], which we state here for completeness:

Lemma 3 (Powering lemma for approximation schemes) *Let $\mathcal{B}'$ be a (classical or quantum) approximation scheme whose estimate W' is within $\pm\epsilon_{pe}q$ to some value q with probability $\frac{1}{2}+\Omega(1)$. Then, there is an approximation scheme $\mathcal{B}$ whose estimate W satisfies $\Pr\left[(1-\epsilon_{pe})q \le W \le (1+\epsilon_{pe})q\right] \ge 1-\delta_{boost}$. It invokes the scheme $\mathcal{B}'$ as a subroutine $O\left(\log\delta_{boost}^{-1}\right)$ times.*

Lemma 3 ensures we can get precise estimates of α_i, which we can than compose into an approximation for the partition function (5.46).

Lemma 4 (Composing ratio estimates) *Let $\epsilon > 0$. Assume we have approximation schemes $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_{\ell-1}$ such that their estimates $Q_0, Q_1, \ldots, Q_{\ell-1}$ satisfy $\Pr\left[|Q_i - \alpha_i| \le \frac{\epsilon\alpha_i}{2\ell}\right] \ge 1-\frac{1}{4\ell}$. Then, there is a simple approximation scheme $\mathcal{A}$ for the product $\alpha = \alpha_0\alpha_1\cdots\alpha_{\ell-1}$. The result $Q = Q_0Q_1\cdots Q_{\ell-1}$ satisfies*

$$\Pr\left[(1-\epsilon)\alpha \le Q \le (1+\epsilon)\alpha\right] \ge \frac{3}{4} . \tag{5.69}$$

We are now finally ready to prove Theorem 6. For each $i = 0,\ldots,\ell-1$, we can apply Lemma 2 with the state $|\psi_i\rangle$ (5.64) and the projector P (5.65). This gives us a quantum approximation scheme for α_i. Note that to prepare $|\psi_i\rangle$, it suffices to prepare $|\pi_i\rangle$ once. Also, to realize a controlled reflection around $|\psi_i\rangle$, it suffices to invoke the controlled reflection around $|\pi_i\rangle$ once.

We now use the reflection $2|\psi_i\rangle\langle\psi_i|-\mathbb{I}$ and set $\epsilon_{pe} = \epsilon/(2\ell)$ in Lemma 2. With these settings, we can apply Lemma 3 to the resulting approximation scheme for α_i with $\delta_{boost} = 1/(4\ell)$. This gives us approximation schemes $\mathcal{A}_i$ outputting Q_i with high precision and probability of success that can be used in Lemma 4. The composite result $Q = Q_0\cdots Q_{\ell-1}$ is thus an approximation for $\alpha = \alpha_0\cdots\alpha_{\ell-1}$ with the property $\Pr\left[(1-\epsilon)\alpha \le Q \le (1+\epsilon)\alpha\right] \ge \frac{3}{4}$.

Finally, we obtain the estimate for Z by multiplying Q with Z_0. Let us summarize the costs from Lemmas 2-4. For each $i = 0,\ldots,\ell-1$, this scheme uses $\log\delta_{boost}^{-1} = O(\log\ell)$ copies of the state $|\pi_i\rangle$, and invokes $\left(\log\delta_{boost}^{-1}\right)\epsilon_{pe}^{-1} = O\left(\frac{\ell}{\epsilon}\log\ell\right)$ reflections around $|\pi_i\rangle$.

Finally, this is where quantum walks come into play. So far, we have assumed that we can prepare the quantum samples $|\pi_i\rangle$ and implement the controlled reflections $R_i = 2|\pi_i\rangle\langle\pi_i| - \mathbb{I}$ about these states perfectly and efficiently. These assumptions can be released and accomplished approximately with the help of quantum walk operators. The errors arising from these approximate procedures do not significantly decrease the success probability of the algorithm.

Using the fact that the consecutive states $|\pi_i\rangle$ and $|\pi_{i+1}\rangle$ are close, we can utilize Grover's $\frac{\pi}{3}$ fixed-point search [53] to drive the starting state $|\pi_0\rangle$ towards the desired state $|\pi_i\rangle$ through multiple intermediate steps[40]. Moreover, to be able to perform this kind of Grover search, we have to be able to apply selective phase shifts of the form $S_i = \omega|\pi_i\rangle\langle\pi_i| + (\mathbb{I} - |\pi_i\rangle\langle\pi_i|)$ for $\omega = e^{i\pi/3}$ and $\omega = e^{-i\pi/3}$. There is an efficient way to apply these phase shifts approximately, based on quantum walks and phase estimation [68].

The important condition for this method to work, the overlap of two consecutive quantum samples $|\pi_i\rangle$ and $|\pi_{i+1}\rangle$ has to be large. This is satisfied when $\alpha_i = Z_{i+1}/Z_i$ is bounded from below by $\frac{1}{2}$, since

$$|\langle\pi_i|\pi_{i+1}\rangle|^2 = \left|\sum_{\sigma\in\Omega} \frac{\sqrt{e^{-\beta_i E(\sigma)}\, e^{-\beta_{i+1}E(\sigma)}}}{\sqrt{Z_i\,Z_{i+1}}}\right|^2 \ge \left|\frac{\sum_{\sigma\in\Omega} e^{-\beta_{i+1}E(\sigma)}}{\sqrt{2Z_{i+1}}\sqrt{Z_{i+1}}}\right|^2 = \frac{1}{2} . \tag{5.70}$$

---

[40]Compare this to the projections of $|\pi_i\rangle$ onto $|\pi_{i+1}\rangle$ and the quantum Zeno effect in [73] discussed in Section 5.4.1.

We can then use additional ancilla qubits and the phase estimation of a quantum walk operator to implement the selective phases required for Grover's fixed-point search, with $O\left(\frac{\ell}{\sqrt{\delta}}\log^2\ell\right)$ applications of a quantum walk operator. Second, the reflections $R_i = 2|\pi_i\rangle\langle\pi_i| - \mathbb{I}$ can to done approximately and effectively, again using phase estimation of a controlled-quantum walk operator. Altogether, with proper choices for the precision for the phase estimations required in the state preparation and reflections, the resulting cost of this scheme (the number of times we have to invoke the controlled quantum walk operators) is $\tilde{O}\left(\frac{\ell^2}{\epsilon\sqrt{\delta}}\right)$. It remains open how to quantize FPRAS which are based on sequences of MC's which do not obey the condition (5.70), or use adaptive steps (not knowing the range into which α_i will fall in advance).

5.4.4 Quantum Metropolis Sampling

The preparation of ground states and (sampling from) Gibbs states is generally a hard task, as finding them is related to optimization problems. However, for classical systems (Hamiltonians), the Metropolis algorithm, described in Section 2.3.3, is widely used and often efficient. When we want to use it to prepare Gibbs (thermal) states of a system, the strategy is to perform a random walk on the states of a system, changing a few local parameters in each step. There is a simple rule for accepting or rejecting the change, depending on the difference in energies of the two states and on the system temperature, which we slowly cool down.

The big question is whether we could we use a quantum computer and quantum walks to prepare Gibbs states of *quantum* systems, especially those plagued by the sign problem (and thus unfit for Quantum Monte Carlo methods). We would thus like to prepare a sample from the eigenstates of a quantum system (according to their energies). However, the energy eigenbasis is now not equal to the computational basis – and making a Markov Chain jumping between the (unknown) energy eigenstates, according to a Metropolis rule, is a hard task. The most serious obstacle to quantizing the Metropolis algorithm is the necessity of rejecting a state change, when the energy of the proposed new state is much larger than the energy of the original state. Is a return from the undesired quantum state to the original (but uknown) one possible and could we do it coherently? Recently, Temme et al. [56] have discovered a way of doing exactly that, using a property of two reflections that is essential to quantum walks, but also found uses in quantum complexity theory [58, 59]. Later, Yung et al. [57] made the algorithm even more quantum, providing a square root speedup in the dependence of the convergence time on δ, the gap of the associated classical Markov Chain.

Let us now briefly sketch the principle of the Quantum Metropolis algorithm due to Temme et al. [56]. For simplicity, let us assume we would like to prepare the Gibbs state of a system of n 2-level particles, i.e. Ising spins. We would like to set up a rapidly mixing Markov chain, which samples from the configurations x with the corresponding probabilities

$$\pi(x) = \frac{e^{-\beta E_x}}{Z}. \tag{5.71}$$

We start from a random state and run the Markov chain for many steps. In each step, we start with some state x with energy E_x, flip a few (a local change) randomly selected spins, and obtain a state y. We accept this move if the energy decreases. On the other hand, if the energy E_y of the new state is higher, we accept the move only with probability $e^{-\beta(E_y - E_x)}$. This Markov chain

obeys detailed balance, because the following is true when $E_y > E_x$:

$$\pi(x) = \pi(y)e^{-\beta(E_x - E_y)}. \tag{5.72}$$

Each step of the quantum version of the Metropolis algorithm now takes as input an energy eigenstate $|\psi_i\rangle$ and applies a random local unitary C to it, producing $\sum_k x_k^i |\psi_k\rangle$, some superposition of energy eigenstates. To be able to decide whether we want to take a step or not, let us also add two extra energy-labeling registers in the state $|E_i\rangle |0\rangle$. Without touching the register with $|E_i\rangle$, we can use phase estimation on the first and third register to produce the label E_k, giving

$$\sum_k x_k^i |\psi_k\rangle |E_i\rangle |E_k\rangle. \tag{5.73}$$

We would now like to accept the states with energies $E_k > E_i$ with probability $e^{-\beta(E_k - E_i)}$. However, a direct measurement of the energy register would collapse the superposition, and this would disallow us to return back to the state $|\psi_i\rangle$. We can work around this obstacle by adding another ancilla register denoting the acceptance of the move, using a unitary F on the energy registers and the acceptance ancilla to locally transform (5.73) into

$$\underbrace{\sum_k x_k^i \sqrt{f_k^i} |\psi_k\rangle |E_i\rangle |E_k\rangle |1\rangle}_{|\psi_i^+\rangle} + \underbrace{\sum_k x_k^i \sqrt{1 - f_k^i} |\psi_k\rangle |E_i\rangle |E_k\rangle |0\rangle}_{|\psi_i^-\rangle}, \tag{5.74}$$

where $f_k^i = \min\left(1, e^{-\beta(E_k - E_i)}\right)$ corresponds to the Metropolis rule (2.12) for the Gibbs distribution[41]. If we now perform a projective measurement Q on the acceptance register and obtain 1, we get exactly what we wanted, as the amplitudes $x_k^i \sqrt{f_k^i}$ correspond to the classical Metropolis rule transition probabilities $|x_k^i|^2 f_k^i$. However, if we measure 0 in the acceptance register, we project into a state that we did not want. Nevertheless, there is a way of undoing this, similar to the procedure used in QMA amplification [58, 59]. The projective measurement of the acceptance register gave us the state $|\psi_i^-\rangle |0\rangle$. The key observation is that this projective measurement left the state within the 2D subspace spanned by $|\psi_i^+\rangle |1\rangle$ and $|\psi_i^-\rangle |0\rangle$, which has a different basis as well:

$$|\varphi_+\rangle = |\psi_i^+\rangle |1\rangle + |\psi_i^-\rangle |0\rangle. \tag{5.75}$$

$$|\varphi_-\rangle = \frac{|\psi_i^+\rangle |1\rangle - |\varphi_+\rangle \langle \varphi_+ | \psi_i^+\rangle |1\rangle}{\sqrt{1 - |\langle \varphi_+ | \psi_i^+\rangle |1\rangle|^2}} \tag{5.76}$$

If we could devise a projective measurement according to the projector $P = |\varphi_+\rangle \langle \varphi_+|$, we could get back into business (obtain the state $|\psi_i^+\rangle |1\rangle$) in the following way. Because the 2D subspace we are in is invariant under the projectors P and Q, performing alternating projective measurements P and Q (as in the Marriott-Watrous scheme) gives a rotation in this 2D subspace. More than that, it will eventually result in the $+1$ eigenstate of Q, which is our desired state $|\psi_i^+\rangle |1\rangle$.

The last necessary ingredient is the projective measurement according to P. It can be performed as follows. We uncompute the unitary F, uncompute the energy label E_k, undo the local

---

[41] Note that the Metropolis algorithm is general, not restricted to use the Gibbs distribution.

mixing unitary C and uncompute the energy label E_i. We can now project onto $|0\rangle\,|0\rangle\,|0\rangle$ on the ancilla registers, and again compute E_i, apply C, compute E_k and apply F. Observe that on the state $|\varphi_+\rangle$ this projection acts as an identity, while the state $|\varphi_-\rangle$ is in its kernel. The required number of steps for applying this rejection (and state repair) procedure is worked out thoroughly in [56], resulting in a quantum algorithm for preparing the Gibbs states of quantum systems. The required number of steps of the algorithm again depends on the inverse of the spectral gap δ of the Markov Chains involved. We believe natural systems thermalize easily, but it is unlikely that MC's whose Hamiltonians correspond to hard computational problems have gaps that scale favorably (as inverse polynomials) with the system size.

The quantum Metropolis algorithm receives a square root speedup (exchanging δ for $\sqrt{\delta}$) in Yung et al.'s work [57], where the authors use quantized (two-register) MC's and quantum simulated annealing instead of the random walk described in this Section which is classical at heart, although running on quantum states and combined with a quantum evaluation and recovery procedure.

5.5   Summary

In this Chapter, we have seen how to implement a quantum walk corresponding to a particular (classical) Markov Chain given by its transition matrix. Instead of appending a coin to the position of a walker, we did it with a system with two-registers both having ranges on the vertices (positions) of the underlying graph. A step of the quantum walk is then made by using one of the registers to govern the diffusion (according to (5.10)) in the other one. We then swap the registers and continue.

The unitary quantum walk we obtain has nice spectral properties, relating the spectra of the Quantum MC to the original spectra of the underlying MC's as seen in Section 5.2. This underlies the quadratic speedups for many algorithms, including phase estimation. Taking it further, Quantum MC's form the basis of sampling algorithms, culminating with the quantum Metropolis algorithm [56].

6   Continuous Time Quantum Walks

In this Chapter we will look at a different approach to quantum walks, with a system undergoing continuous-time evolution according to the Schrödinger equation, governed by a fixed "hopping" Hamiltonian related to an underlying graph. This is a more natural description related to the dynamics of a single excitation in condensed-matter systems in physical systems. We start with a review of the development of continuous-time quantum walks, discuss their behavior, and present several algorithms (one with an exponential speedup) utilizing graph symmetries. We then show how continuous-time quantum walks (CTQW) can be used as a universal model for quantum computation, and wrap up with the comparison (and connection) of continuous-time to discrete-time quantum walks.

6.1   Quantizing Continuous Random Walks

So far, we have investigated classical and quantum walks on graphs in *discrete* time. They are described by a particular update rule – a transition matrix (classical Markov Chains), a unitary coin-toss and shift matrices (usual discrete-time quantum walks), or a product of two reflections (quantized Markov chains). Where classical walks involve the evolution of probabilities, quantum discrete-time walks involve the evolution of amplitudes (of quantum states) and require an extra coin register to ensure unitary evolution. They describe discrete quantum diffusion processes (retaining at least some degree of coherence) on graphs. We will now look for a quantum process described by a continuous-time equation similar to the diffusion equation, and call what we find a *continuous-time quantum walk*. In contrast to discrete-time quantum walks, there will be no need for the extra register ("coin" or spin degree of freedom), but using one can be helpful, as shown by the algorithm for spatial search in Section 6.2.

Let us consider a graph $G = (V, E)$ where V is the set of its vertices and E the set of its edges. Our goal si to model a continuous-time diffusion process on this graph. We start with a classical process, where in each small time-step, vertex j leaks probability to its neighbors (there are $\deg(j)$ of them), while collecting some probability back from them. We can describe this process by the diffusion equation

$$\frac{\mathrm{d}}{\mathrm{d}t}p_j(t) = \sum_{k \in V} L_{j,k} p_k(t), \tag{6.1}$$

where L is the Laplacian of G, given by

$$L_{j,k} = \begin{cases} -\deg(j) & j = k, \\ 1 & (j,k) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{6.2}$$

This matrix respects the graph structure, and is a representation of the discrete Laplace operator ∇^2. For example, on a cycle of length 5, the Laplacian is

$$L_{\circ_5} = \begin{bmatrix} -2 & 1 & 0 & 0 & 1 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 1 & 0 & 0 & 1 & -2 \end{bmatrix}. \tag{6.3}$$

We will now define a quantum process involving the evolution of amplitudes (instead of probabilities), described by an equation similar to (6.1). The matrix L is Hermitian, so we can also think of it as a Hamiltonian in the Schrödinger equation as

$$i\frac{\mathrm{d}}{\mathrm{d}t}\left|\psi(t)\right\rangle = L\left|\psi(t)\right\rangle. \tag{6.4}$$

The essential difference from (6.1) is the appearance of the imaginary i (inducing unitary evolution), and the interpretation of $\left|\psi(t)\right\rangle$ as the amplitudes of the system's wavefunction. We will soon see this has interesting consequences, just as using unitary update rules gave discrete quantum walks their rich structure.

This formulation (using the Laplacian matrix) was used by Farhi and Gutmann in the first paper on continuous-time quantum walks [78], where they investigated the transition/reflection properties of binary trees. We could also choose a different matrix for our Hamiltonian. One example also containing the information about the graph is the adjacency matrix[42] of G

$$A_{j,k} = \begin{cases} 1 & (j,k) \in E, \\ 0 & (j,k) \notin E. \end{cases} \tag{6.5}$$

It is often suitable to use as the Hamiltonian, but the analogy with the classical diffusion is now lost. It is not always clear how to use A as a generator of a continuous time *classical* random walk, where one may need to use its "lazy" version generated by $\mathbb{I} - A$.

6.1.1   Walking in 1D and Mixing

The discrete quantum walk in 1D behaves quite differently than the classical drunken-sailor walk. The continuous version of the 1D walk resembles the quantum walk we have seen in Section 3.2. How fast the probability is spreading is proportional to t instead of the classical $\sqrt{t}$. We will now show this, and then discuss the sense in which we can talk about mixing for continuous-time quantum walks.

The system we will investigate is a cycle of length N (periodic boundary conditions). The behavior of the continuous-time quantum walk on a cycle is similar for both basic choices of the Hamiltonian — the Laplacian or the adjacency matrix, as they are related by $L = A - 2\mathbb{I}$. For simplicity, let us then choose

$$H_N = -A_N, \tag{6.6}$$

the negative of the adjacency matrix of the cycle, acting on the N basis states $\left|x\right\rangle$ as

$$\begin{aligned} H_N\left|x\right\rangle &= -\left|x-1\right\rangle - \left|x+1\right\rangle, \quad \text{for } 2 \leq x \leq N-1, &\tag{6.7}\\ H_N\left|1\right\rangle &= -\left|N\right\rangle - \left|2\right\rangle, \\ H_N\left|N\right\rangle &= -\left|N-1\right\rangle - \left|1\right\rangle. \end{aligned}$$

One obvious eigenvector of H_N is the uniform superposition over all states $\left|x\right\rangle$.

---

[42]also with size $|V| \times |V|$

Exercise 14 Show that all of the eigenvectors of H_N are plane waves, i.e.

$$|\phi_k\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^{N} e^{ip_k x} |x\rangle, \tag{6.8}$$

with the corresponding eigenvalues (energies)

$$E_k = -\left(e^{-ip_k} + e^{ip_k}\right) = -2\cos p_k. \tag{6.9}$$

Show also that the periodic boundary conditions constrain the momenta to values

$$p_k = \frac{2\pi k}{N}, \tag{6.10}$$

for $0 \le k \le N-1$, or alternatively, $-\lfloor \frac{N}{2} \rfloor \le k \le \lceil \frac{N}{2} \rceil$.

Let us start in a state concentrated at vertex x and let it evolve for time t according to the Schrödinger equation with Hamiltonian H_N. Using an expansion in terms of the eigenvectors, the amplitude at vertex y at time t is

$$\langle y| e^{-iH_N t} |x\rangle = \langle y| e^{-iH_N t} \left(\sum_{k=0}^{N-1} |\phi_k\rangle \langle \phi_k| \right) |x\rangle \tag{6.11}$$

$$= \sum_{k=0}^{N-1} e^{-i(-2\cos p_k)t} \langle y|\phi_k\rangle\langle \phi_k|x\rangle \tag{6.12}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} e^{i2t \cos p_k} e^{ip_k(y-x)}. \tag{6.13}$$

For large N, we can approximate the sum by an integral, and obtain [82] the Bessel function of the first kind of order $(y-x)$ as

$$\langle y| e^{-iH_L t} |x\rangle \approx i^{(y-x)} J_{y-x}(2t), \tag{6.14}$$

$$p_{y-x}(t) \approx |J_{y-x}(2t)|^2 \tag{6.15}$$

We can now look at the asymptotics of the Bessel function [77]. For $2t$ large, but still obeying $2t \ll y-x$, we have $J_{y-x}(2t) \approx \frac{t^{y-x}}{(y-x)!}$, which is exponentially small. However, for $2t \approx y-x$, the values of the Bessel function start to rise, and $J_{y-x}(y-x)$ is on the order of $(y-x)^{-\frac{1}{3}}$. Furthermore, for $2t > y-x$ the function $J_{y-x}(2t)$ becomes qualitatively a cosine wave with amplitude decreasing as $\frac{1}{\sqrt{2t}}$. For illustration, we plot the probabilities arising from the transition amplitudes for a fixed large time t, and for a fixed large distance $y-x$ in Figure 6.1.

There is a significant probability of transition $x \to y$ for times of order $2t \approx y-x$, which corresponds to the wavefront moving with a constant speed of the walker[43] equal to 2. Just as for the discrete quantum walk, the average distance of the walker for the continuous quantum walk on a line rises linearly with time. Contrast this to the classical drunkards' walk, in which the

---

[43]Note that the speed of spreading for the discrete quantum walk in 1D is proven to be $\frac{1}{\sqrt{2}}$.

$$p_{y-x}(t) = |J_{y-x}(2t)|^2 \quad \text{for fixed } t=1000$$

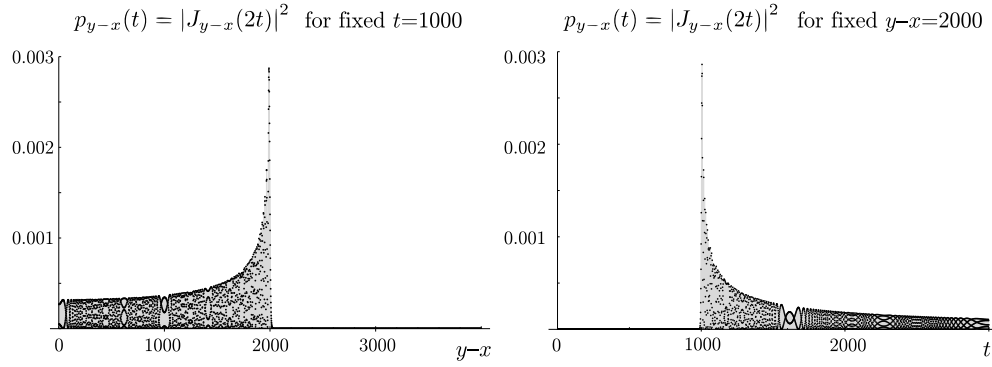$$p_{y-x}(t) = |J_{y-x}(2t)|^2 \quad \text{for fixed } y-x=2000$$

Fig. 6.1. Plotting the probabilities of being at vertex y when starting at $x = 0$ for a continuous-time quantum walk on an infinite line, approximated with the help of Bessel functions as $p_{y-x}(t) \approx |J_x(2t)|^2$. Note that at time $t = 1000$, the wavefront of the walk is at position $y - x = 2000 = 2t$, agreeing with the speed of spreading equal to 2 distance points per unit of time, coming from the largest eigenvalue of H, which is 2.

average distance rises as a square root of the number of steps (or equivalently, time). The similarities between continuous and discrete quantum walks are discussed in more detail in Section 6.5. We refer the reader to Figure 6.8, where we compare the probabilities of finding the walker at a given vertex for both types of quantum walks in 1D.

The evolution according to the Schrödinger equation – and thus also the continuous quantum walk – is unitary. There is no mixing towards a stationary distribution – a wavefunction does not change with time only if we start in an eigenstate. However, for finite graphs we can again talk about its mixing in a time-averaged sense, just as we did for discrete-time quantum walks in Section 3.5.1, and follow Aharonov et al. [15]. The probability of being at vertex y at time t when starting from vertex x is

$$p_t(x \rightarrow y) = \left| \langle y | e^{-iHt} | x \rangle \right|^2, \tag{6.16}$$

and this probability does not converge. However, when we pick a random time t between 0 and some large T, we can talk about a *time-averaged distribution*

$$\bar{p}_T(x \rightarrow y) = \frac{1}{T} \int_0^T \left| \langle y | e^{-iHt} | x \rangle \right|^2 dt. \tag{6.17}$$

In the $T \rightarrow \infty$ limit, this gives rise (and converges) to the *limiting distribution*

$$\pi(x \rightarrow y) = \lim_{T \rightarrow \infty} \bar{p}_T(x \rightarrow y). \tag{6.18}$$

This limiting distribution can be computed using the expansion to energy eigenstates $|\phi_k\rangle$, simi-

larly to what we did in (6.13) as

$$\pi(x \to y) = \lim_{T \to \infty} \int_0^T \left| \langle y | e^{-iHt} | x \rangle \right|^2 dt \tag{6.19}$$

$$= \lim_{T \to \infty} \int_0^T \sum_{k,m} \langle y | \phi_k \rangle \langle \phi_k | e^{-iHt} | x \rangle \langle x | \phi_m \rangle \langle \phi_m | e^{-iHt} y | y \rangle dt \tag{6.20}$$

$$= \sum_{k,m} \langle y | \phi_k \rangle \langle \phi_k | x \rangle \langle x | \phi_m \rangle \langle \phi_m | y \rangle \underbrace{\lim_{T \to \infty} \frac{1}{T} \int_0^\infty e^{-i(E_m - E_k)t} dt}_{\delta_{E_k - E_m}} \tag{6.21}$$

$$= \sum_{E_k = E_m} \langle y | \phi_k \rangle \langle \phi_k | x \rangle \langle x | \phi_m \rangle \langle \phi_m | y \rangle. \tag{6.22}$$

How fast does the time-averaged distribution converge to the limiting one? When computing

$$\Delta_T(x \to y) = \bar{p}_T(x \to y) - \pi(x \to y), \tag{6.23}$$

the terms that do not subtract out come from the finite-T integral in (6.22), and only from those terms where we sum over nonequal pairs of energies, i.e.

$$\bar{p}_T(x \to y) - \pi(x \to y) = \sum_{E_k \neq E_m} \langle y | \phi_k \rangle \langle \phi_k | x \rangle \langle x | \phi_m \rangle \langle \phi_m | y \rangle \frac{e^{-i(E_m - E_k)T} - 1}{-i(E_m - E_k)T}. \tag{6.24}$$

Thus for the continuous-time quantum walk starting at vertex x, the total probability distribution distance

$$\|\bar{p}_T(x) - \pi(x)\| = \sum_y |\bar{p}_T(x \to y) - \pi(x \to y)| \tag{6.25}$$

from the limiting distribution will converge towards 0 with growing T as a function of the gaps in the energy spectrum. The *mixing time*[44] $\mathcal{M}_\epsilon^q$ is then a time beyond which the total probability distribution distance from the limiting distribution is smaller than a precision parameter ϵ, i.e.

$$\forall T \geq \mathcal{M}_\epsilon^q : \quad \|\bar{p}_T(x) - \pi(x)\| \leq \epsilon. \tag{6.26}$$

Note that it is not governed only by the energy gap between the ground state and in the first excited state, as for classical Markov chains, but grows because of small gaps anywhere in the spectrum.

Continuing with our example — the CTQW on a cycle of length N, we can compute the limiting distribution plugging (6.8) and (6.9) into (6.22).

Exercise 15 For a cycle with odd length N, with an initial state concentrated on a single vertex x, show that the limiting distribution is

$$\pi(x \to y) = \frac{1 + \delta_{y,x}}{N} - \frac{1}{N^2}, \tag{6.27}$$

---

[44]This is analogous to (3.29), where we defined the mixing time for discrete-time quantum walks.

while for even-length cycles, we get

$$\pi(x \to y) = \frac{1 + \delta_{y,x} + \delta_{y,x+N/2}}{N} - \frac{2}{N^2}. \tag{6.28}$$

Both of these are not uniform, having small corrections of the order N^{-2} at each vertex, which make up for the different amplitude at the initial vertex (or the vertex directly opposite x for the even-cycle case). Analyzing the inverses of differences of the non-equal eigenvalues in (6.24), it can be shown that the mixing time for the CTQW on a cycle scales as

$$\mathcal{M}_\epsilon^{\mathrm{q,cycle}} \leq \epsilon^{-1} N \log N. \tag{6.29}$$

This (superlinear scaling in N) can be connected to the linear spreading of the maximum of the wavefront as seen in Fig.6.1. This behavior is interesting for uses in the computational models related to the Feynman computer, e.g. in [89].

The scaling of the mixing times for CTQW has been also investigated for necklace graphs, which are cyclic graphs made by connecting many copies of a certain subgraph (pearl) [90], and also utilized in a quantum-walk based model of quantum computation [88].

6.1.2 Symmetries and Continuous-time Quantum Walks

When the graphs we walk on have symmetries, analyzing the dynamics simplifies a lot. The coherent evolution of superpositions also often gives interesting results, when constructive interference at specific times can significantly raise the probability of being at specific vertices.

For example, let us look at the continuous-time quantum walk on the *hypercube* (see also Section 2.3.1 and Figure 2.2). The adjacency matrix of an n-dimensional hypercube is conveniently written in terms of the Pauli matrices as

$$A = \sum_{j=1}^{n} \sigma_x^{(j)}. \tag{6.30}$$

Schrödinger time evolution of the initial state $|00\cdots0\rangle$ with the Hamiltonian $H = -A$ is surprisingly simple, as all the $\sigma_x^{(j)}$ matrices commute. The evolution is thus an independent rotation on each qubit:

$$e^{-iHt}|00\cdots0\rangle = \bigotimes_{j=1}^{k}\left(e^{it\sigma_x^{(j)}}|0\rangle_j\right) = \bigotimes_{j=1}^{k}\left(\cos t\,|0\rangle_j + i\sin t\,|1\rangle_1\right). \tag{6.31}$$

At times $t = \frac{(2k+1)\pi}{2}$, all the qubits are rotated into the state $|1\rangle$, so the continuous-time quantum walk starting in the state $|00\cdots0\rangle$ traverses the hypercube completely in constant time. However, it is important to remember that it is not only the time that counts as 'cost' of a continuous-time quantum walk algorithm. Rather, it is the dimensionless parameter $\|H\|\,t$.

Exercise 16 Show that the norm (largest eigenvalue) of the Hamiltonian A is linear in n.
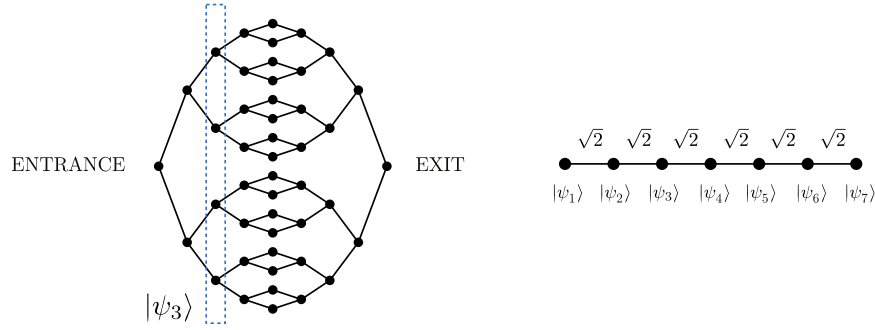
Fig. 6.2. Two binary trees with 4 layers make up the graph G_4 when glued together. The line on the right depicts the equivalent quantum walk on a line of the column states $|\psi_n\rangle$, with weight $\sqrt{2}$ on each link.

When we rescale $A \to \frac{1}{\|A\|} A$ to give it norm 1, the time required for traversal becomes $\frac{\pi n}{2}$, linear in n, as we have seen in Section 2.3.1 for the discrete quantum walk on a hypercube. Compare this to the classical random walk on a hypercube, where the walk rapidly mixes towards the uniform distribution, where the probability of finding the string $11 \cdots 1$ is 2^{-n}.

What was the symmetry that comes into play in this example? Due to the permutational symmetry of the hypercube (and a Hamiltonian that preserves it), one can also view the evolution as a quantum walk on a line of states (the $n+1$ symmetric superpositions with Hamming weight $k = 0, \ldots, n$), with transition coefficients $\sqrt{(n-k)(k+1)}$.

The next example of symmetries simplifying the evolution comes from Childs et al. [79]. Imagine we walk on a graph G_n that is made from two glued binary threes of depth n (with 2^{n-1} leaves each) as in Figure 6.2. This is reminiscent of (but much simpler than) the graph[45] encountered by the WALKSAT algorithm for 3-SAT (Section 2.3.2), with each node having 2 ways to go towards the center, but only 1 way to go towards the endpoints. However, what does a continuous-time quantum walk see here? When we choose the Hamiltonian to be the negative of the adjacency matrix

$$H_{tt} = -A_{tree_1} - A_{tree_2}, \tag{6.32}$$

its action turns out to be very simple if we view it in terms of the "column" states

$$|\psi_k\rangle = \frac{1}{\sqrt{n_k}} \sum_{j=1}^{n_k} |x_j^{(k)}\rangle. \tag{6.33}$$

Each $|\psi_k\rangle$ is a uniform superposition of the n_k states located at the vertices of the k-th column of the graph. In this basis $\{|\psi_k\rangle\}_{k=1}^{2n-1}$, the Hamiltonian is the adjacency matrix of a quantum walk

---

[45]The actual graph there is much more complicated, and we don't yet know how to naturally quantize the WALKSAT algorithm besides the general "Groverizing" approach.
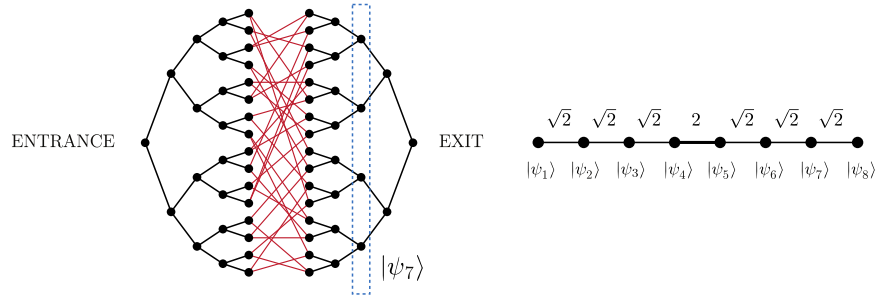
Fig. 6.3. Two binary trees with 4 layers, glued by a randomly chosen cycle of length 2^{n+1} with vertices alternating between the two trees. Every vertex (besides the endpoints) in the graph now has three neighbors. The line on the right depicts the equivalent quantum walk on a line of the column states $|\psi_n\rangle$ with modified weights.

on a line of length $2n - 1$ with weights $\sqrt{2}$, as

$$H_{tt}|\psi_k\rangle = -\sqrt{2}|\psi_{k+1}\rangle - \sqrt{2}|\psi_{k-1}\rangle, \qquad 2 \le k \le 2n - 2, \tag{6.34}$$

$$H_{tt}|\psi_1\rangle = -\sqrt{2}|\psi_2\rangle, \tag{6.35}$$

$$H_{tt}|\psi_{2n-1}\rangle = -\sqrt{2}|\psi_{2n-2}\rangle. \tag{6.36}$$

We have already seen how this walk behaves[46] in Section 6.1.1, because the extra weight $\sqrt{2}$ only multiplies the whole Hamiltonian. Thus the time it takes for the "walker" to move from the left endpoint to the right endpoint (with constant probability) scales as $O\left(\frac{2n-1}{2\sqrt{2}}\right)$, only linearly with the tree depth n.

Compare this to a classical random walk without memory, which will (highly probably) get stuck in the center region of the graph. Note though, that there exists a classical recursive $O(n^2)$ algorithm for traversing this type of graph (see Section 2.3.1). It crucially depends on the possibility of identifying the center column vertices by their degree.

Soon afterwards, Childs et al. [33] modified the previous construction, gluing the two trees by a cycle of length 2^{n+1} alternating between the leaves of the two trees as in Figure 6.3, ensuring the degree of the central vertices is also 3. This gave them one of the few examples with a provable exponential separation in query complexity between the best possible classical algorithm and a simple quantum-walk algorithm for an oracle problem. Let us thus investigate the unitary evolution governed by the Hamiltonian

$$H_{tt2} = -A_{tree_1} - A_{cycle} - A_{tree_2}, \tag{6.37}$$

where A_{tree} is the adjacency matrix of a tree and A_{cycle} the adjacency matrix of the central cycle. Because of symmetry, we can again choose a basis of $2n$ "column" states $|\psi_n\rangle$ (6.33), with

$$n_k = \begin{cases} 2^{k-1} & 1 \le k \le n, \\ 2^{2n-k} & n+1 \le k \le 2n. \end{cases} \tag{6.38}$$

---

[46]Recall that we started the walk from the center of a finite line (or any point on a cycle) in Section 6.1.1. Nevertheless, if one starts the walk from an endpoint, the probability to be in another vertex again starts to spread linearly with time.

The action of (6.37) on $|\psi_n\rangle$ differs from (6.35) only in the middle, where accounting for the cycle connections we obtain

$$
\begin{aligned}
H_{tt2} |\psi_n\rangle &= -\sqrt{2} |\psi_{n-1}\rangle - 2 |\psi_{n+1}\rangle, & (6.39) \\
H_{tt2} |\psi_{n+1}\rangle &= -2 |\psi_n\rangle - \sqrt{2} |\psi_{n+2}\rangle. & (6.40)
\end{aligned}
$$

Therefore, the evolution with H_{tt2} (in the symmetric subspace) is the same as the continuous-time quantum walk on a line depicted in Figure 6.3, with weights $\sqrt{2}, \ldots, \sqrt{2}, 2, \sqrt{2}, \ldots, \sqrt{2}$. When we let the state $|\psi_1\rangle$ evolve, it now starts to be reasonable to expect that after time $O(n)$, there will be a significant probability of measuring the state located at the root of the tree on the right, i.e. for $e^{-iH_{tt}t} |\psi_1\rangle$ to have a large overlap with $|\psi_{2n}\rangle$. As shown in [33], we can solve for eigenvectors and eigenvalues of this walk similarly to what we did on a line, and then show a mixing result to prove the probability of reaching EXIT in $O(n)$ time is substantial.

The quantum walk on two-trees glued by a cycle solves a Hamiltonian oracle problem – to find the vertex named EXIT. We can turn it into an oracle problem in the usual quantum circuit model. For this, we consider a black-box oracle returning the names of neighbors of a given vertex. We can then efficiently simulate [32] the quantum walk with the Hamiltonian (6.37), because the Hamiltonian is sparse.

Unlike the two simply glued trees traversal problem in Figure (6.2), the oracle problem with two trees glued by a cycle as in Figure (6.3) is difficult classically [33]. Here is a sketch of the proof. First, we can think of the vertices having random names of length $3n$. Only 2^{2n} out of those 2^{3n} strings thus correspond to graph vertices, so that it is hard to guess a name of a vertex actually belonging to the graph. This restricts any classical algorithm to explore only a connected part of the graph around the ENTRANCE (as the oracle can supply the names of the neighbors of a given vertex). However, whichever way we explore the graph, we can embed the part we have seen into the glued trees + cycle at random. For a number of queries that is exponential in n but much less than the number of vertices of one of the trees, the probability of finding the EXIT (or even reaching the same vertex we already have been in by a different path) will remain exponentially small.

Recently, another quantum algorithm for traversing randomly glued trees has been found [87] in the adiabatic quantum algorithm (AQC) model. It is inspired by the quantum walk-based one presented above. However, the AQC algorithm differs from the usual approaches [75] in that it utilizes two lowest energy levels (instead of only the ground state), and does not care that the eigenvalue gap between these two levels is exponentially small.

6.2   Spatial Search

We have seen how to perform searches for marked vertices on graphs using coined quantum walks in Section 2.3.1. The goal was to concentrate the amplitude of the evolved state on a specially marked vertex (which had different coin-flipping or scattering properties). We can perform the same feat with continuous-time quantum walks. In fact, the development of the corresponding algorithms was often parallel, with continuous quantum walks sometimes leading the way [81] and sometimes catching up [80] to the competing discrete-time method.

Search algorithms based on continuous-time quantum walks are based on the following common idea. We start with a uniform superposition $|s\rangle$ over all vertices and let it evolve with the

Hamiltonian

$$H_{search} = -\gamma A_{graph} - \sum_{w \in W} |w\rangle \langle w|, \tag{6.41}$$

where A is the adjacency matrix (or sometimes, the Laplacian) of the graph, W is a set of marked vertices and γ is a tunable parameter. Often, the ground state of (6.41) and its first excited state will both have large overlap with the uniform superposition $|s\rangle$ and some marked state $|w\rangle$. When we let the state $|s\rangle$ evolve, due to quantum tunneling, the state $|s\rangle$ will then transform into a state with a large overlap with the state $|w\rangle$, on the timescale of $\frac{1}{E_1 - E_0}$. Let us look at a few examples of how this happens, starting with the unstructured search problem, and a quantum walk algorithm that is an analog analogue of Grover's algorithm [76].

6.2.1   Complete Graph

An unstructured search of N items is equivalent to a spatial search on a complete graph with N vertices (where we can freely jump from any vertex to any other vertex) as in Figure 4.4. For simplicity, let us now think only of a single marked vertex. In Section 4.4, we have seen how this problem is solved by a scattering quantum walk. Let us now look at it in a Hamiltonian formulation, with the marked state specified by an oracle Hamiltonian

$$H_w = - |w\rangle \langle w|, \tag{6.42}$$

where the special vertex $|w\rangle$ (the "winner") gets a decrease in energy compared to the other vertices. Our goal is to prepare the state $|w\rangle$ (if there is such a state). We will follow Farhi et al. [76], who worked this out soon after Grover's algorithm appeared. The approach is to add an instance-independent Hamiltonian – the rescaled adjacency matrix of the full graph. We will rescale the adjacency matrix A, so that its norm is $O(1)$ (corresponding to $\gamma = \frac{1}{N}$ in (6.41)). The reason for this is the time-energy tradeoff, as increasing the norm of the Hamiltonian obviously decreases the required runtime. Together with the oracle Hamiltonian, we will thus use

$$H = -\frac{1}{N} A + H_w = - |s\rangle \langle s| - |w\rangle \langle w|, \tag{6.43}$$

where $|s\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} |j\rangle$ is the uniform superposition of computational basis states.

The algorithm is rather simple. We start in the uniform superposition $|s\rangle$ over all vertices, and let the system evolve (according to the Schrödinger equation) for time $O\left(\sqrt{N}\right)$. Let us look at why it works. The Hamiltonian (6.43) never takes the evolution of $|s\rangle$ outside of the two-dimensional Hilbert space spanned by $|w\rangle$ and $|s\rangle$. We can thus rewrite the Hamiltonian in (and restricted to) the basis $\{|w\rangle, |w^\perp\rangle\}$, where

$$|w^\perp\rangle = \frac{|s\rangle - \delta |w\rangle}{\sqrt{1 - \delta^2}}, \tag{6.44}$$

with $\delta = \frac{1}{\sqrt{N}}$ (this generalizes to $\sqrt{M/N}$ if there are M marked states). In the new basis, the Hamiltonian reads

$$\begin{aligned} H &= -\left(|w\rangle + \sqrt{1 - \delta^2}|w^\perp\rangle\right)\left(\langle w| + \sqrt{1 - \delta^2}\langle w^\perp|\right) - |w\rangle \langle w| \\ &= -\begin{bmatrix} \delta^2 + 1 & \delta\sqrt{1 - \delta^2} \\ \delta\sqrt{1 - \delta^2} & 1 - \delta^2 \end{bmatrix} = -\mathbb{I} + \delta\left(\delta\sigma_z + \sqrt{1 - \delta^2}\sigma_x\right). \end{aligned} \tag{6.45}$$

Using $e^{i\alpha\vec{\hat{n}}\cdot\vec{\sigma}} = (\cos\alpha)\mathbb{I} + i(\sin\alpha)\vec{\hat{n}}\cdot\vec{\sigma}$, the time evolution of $|s\rangle = \delta|w\rangle + \sqrt{1-\delta^2}|w^\perp\rangle$, up to an insignificant phase is

$$
\begin{aligned}
e^{-iHt}|s\rangle &= \left[\begin{array}{cc} \cos(\delta t) - i\delta\sin(\delta t) & -i\sqrt{1-\delta^2}\sin(\delta t) \\ -i\sqrt{1-\delta^2}\sin(\delta t) & \cos(\delta t) + i\delta\sin(\delta t) \end{array}\right] \left[\begin{array}{c} \delta \\ \sqrt{1-\delta^2} \end{array}\right] \\
&= \left[\begin{array}{c} \delta\cos(\delta t) - i\sin(\delta t) \\ \sqrt{1-\delta^2}\cos(\delta t) \end{array}\right] = \cos(\delta t)|s\rangle - i\sin(\delta t)|w\rangle. \quad (6.46)
\end{aligned}
$$

Therefore, it is enough to wait time $T = \frac{\pi}{2\delta} = \frac{\pi}{2}\sqrt{N}$ to obtain the marked state $|w\rangle$ with probability 1.

In fact, this algorithm is optimal, and the reader can find the instructive proof – an alternative to the BBBV proof [74] in the Hamiltonian oracle model – in the paper[47] by Farhi et al. [76], who were quick to realize they got the Grover search algorithm in continuous time soon after Grover published his original paper [30]. The proof is based on analyzing the evolution of a known state towards a marked state $|w\rangle$ with the oracle Hamiltonian $-|w\rangle\langle w|$ plus an arbitrary driver Hamiltonian (without knowledge of the solution). For this algorithm (which doesn't know w) to work for all N possible choices of w, the evolution of an initial state must substantially differ for all of those. That in turn lower bounds the required evolution time T by $T \geq O\left(\frac{\sqrt{N}}{\|H\|}\right)$.

6.2.2   Searching on the Hypercube and on Finite-dimensional Lattices

The next example where continuous-time quantum search provides a fast algorithm is search on the hypercube (cf. Section 3.5.2 for the discrete-time approach). Following Childs et al., [44,80], we will investigate the Hamiltonian

$$
H = -\gamma A_{hc} - |w\rangle\langle w|, \quad (6.47)
$$

where $A_{hc} = \sum_{j=1}^n \sigma_x^{(j)}$ is the adjacency matrix of a n-dimensional hypercube (described by n-qubit states). The spectrum of this Hamiltonian was analyzed by Farhi et al. in [75]. It turns out that for a specific choice of $\gamma = \frac{2}{n} + O(n^{-2})$, the energy gap is $\frac{1}{\sqrt{2^n}}[1 + O(n^{-1})]$, and the ground and the first excited state of (6.47) are $\frac{1}{\sqrt{2}}(|w\rangle \pm |s\rangle)$ up to terms of order $O(n^{-1})$. Thus, evolving the state $|s\rangle$ for time of order $\sqrt{2^n}$, the probability of finding $|w\rangle$ is of order 1. This is again a quadratic speedup over a classical algorithm, just as the one using discrete-time quantum walk search [31].

Let us now move to a d-dimensional lattice. Childs et al. [44] investigate the Hamiltonian (6.41) and find a critical point in γ, where the gap is small, but the ground and the excited states again have large overlap with $|w\rangle$ and $|s\rangle$. Altogether, their algorithm has running time $O\left(\sqrt{2^n \log 2^n}\right)$ in dimensions $d \geq 4$. There it gives a quadratic speedup (up to a log-factor for $d = 4$) over the best classical algorithm. However, the continuous quantum walk search does not work for $d < 4$.

The discrete-time lattice search algorithm [23] has runtime $\sqrt{2^n}$ all the way down to $d = 2$. Can continuous quantum walks work there too? It turns out yes, but it requires augmentation. Childs et al. [80] showed that adding an extra degree of freedom (spin) of the walking particle and

---

[47]An Analog Analogue of a Digital quantum computation

Fig. 6.4. A game tree for a game with two players alternating moves, starting at the root of the tree. If Alice (who chooses her move first) ends up at a leaf labeled 1, she wins. Given a leaf labeling, we fill the vertices using the NAND operation (as in the marked rectangles). For this particular game, Alice does not have a winning strategy, assuming best possible play from Bob.

replacing the Laplacian (or the adjacency matrix) with the massless Dirac Hamiltonian helps. The dynamics is now governed by the Dirac equation (for a massless particle). The energy spectrum near $E = 0$ now has linear (instead of quadratic) dispersion (dependence on the momentum). This is the underlying reason for the gap at the critical point being large enough to give a speedup for the continuous-time quantum walk algorithm. The result is then an algorithm with running time $O\left(\sqrt{2^n \log 2^n}\right)$ in dimensions $d = 2$ and $d = 3$, matching the discrete-time walk, at the cost of a larger Hilbert space.

6.3 NAND Trees and Games

Let us have a look at another application of continuous-time quantum walks. In the early days of quantum computing, the paper *Quantum Computation and Decision Trees* [78] investigated a system whose Hamiltonian is the adjacency matrix of a binary tree, and looked at the transmission and spreading of wavepackets (and their possible concentration on searched-for vertices). Here, we will see how the transmission-reflection properties of a graph (under time-evolution with an adjacency-matrix Hamiltonian) are affected by a certain graph property. The result of this analysis is a continuous quantum walk-based algorithm [83] for evaluating NAND trees of depth $n = \log_2 N$, whose query complexity $O\left(N^{0.5}\right)$ beats the best possible classical algorithm which requires $O\left(N^{0.753}\right)$ oracle queries. This quantum algorithm developed in the Hamiltonian oracle model has been quickly translated into the more common discrete-query-model [84], and then generalized to arbitrary trees and optimized by Ambainis et al. [85].

Alice (who starts) and Bob (going second) are playing a simple game for two alternating players, making one of two possible moves. This game can be described by a binary tree of depth n, with the $N = 2^n$ leaves labeled by the results of the game as in Figure 6.4. Does Alice (the first player to move) have a winning strategy, ensuring she ends up at a leaf labeled 1? We can figure it out by working our way down the tree from the leaves to the root. If Alice reaches a leaf labeled 1, she wins, if she gets to a leaf with a 0, she loses. When Bob chooses his move on the layer $n - 1$, he will surely choose a win for himself by sending Alice to a 0-labeled leaf
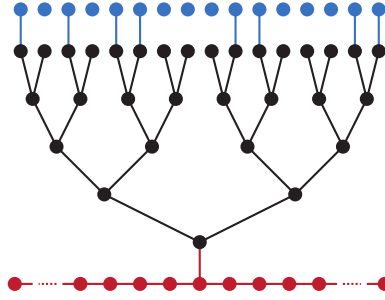
Fig. 6.5. A graph for the quantum walk of the NAND-tree algorithm, corresponding to the game in Figure 6.4. Its centerpart is a balanced binary tree. Above it we have an extra row of vertices, connected to the tree by edges encoding a particular game (there is an extra edge only for those tree vertices that originally carried a "1" label). Finally, the tree is connected to a $\sqrt{N}$-long line (runway).

(if one like that exists). Thus, if there is a 0-leaf above his position, he has a winning strategy, and we can label that graph vertex 1. On the other hand, if both leaves above Bob's vertex are 1, he can not win (with best play from the opponent), and so we will label it 0 as in the marked rectangles in Figure 6.4. This works at any level of the tree, so we can work our way down to the root using the NAND function on the bits on the children of each node $x = \neg(c_1 \wedge c_2)$. The problem of determining the existence of a winning strategy for Alice in this game is thus equivalent to evaluating a binary NAND tree with a set of game results fixed on the leaves.

The provably optimal [86] classical algorithm for this problem uses randomized recursive branch evaluation. To evaluate a given vertex, we randomly pick one of its children and continue with the recursive evaluation. If we conclude with a 0, the value at the original vertex is certainly 1 (because of the property of the NAND). Only if the evaluation produces a 1, we need to do more work and evaluate the other branch as well. For a game with a balanced[48] binary tree, this recursive procedure requires $O(N^{0.753})$ calls to the oracle encoding the game results at the leaves.

The basis of the quantum algorithm of Farhi et al. [83] is a continuous-time quantum walk on a graph, whose main part is a balanced tree with extra leaves at the vertices with game results equal to 1. This tree is rooted above the middle of a long line of vertices[49] as in Figure 6.5. On the runway (the long line), the eigenvectors of an adjacency-matrix Hamiltonian have to be combinations of plane waves. When we look at eigenvectors with energies close to 0 (and momenta close to $\pi/2$), we find that the graph possesses interesting scattering properties. We look for eigenvectors

$$|\psi\rangle \propto \sum_{x \leq 0} e^{-ipx} |x\rangle + R \sum_{x \leq 0} e^{ipx} |x\rangle + T \sum_{x > 0} e^{-ipx} |x\rangle + |\varphi_{tree}\rangle \,, \qquad (6.48)$$

which are a combination of a right-moving wave with a reflected and transmitted part, plus something inside the tree. For NAND-trees evaluating to 1 (with a winning strategy for Alice, the

---

[48]In a balanced tree, the lengths of the subtrees from a node do not differ by more than 1.
[49]As shown in [85], this long line can be shortened to just one vertex on each side, with additional weight $\sqrt{N}$.
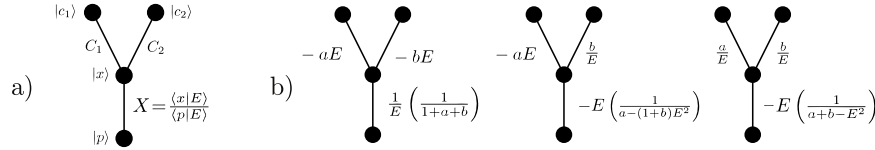
Fig. 6.6. a) Defining the ratios of amplitudes at the vertex x and its children c_i and parent p for a NAND tree. b) Evaluating the ratio X from known ratios C_i using (6.53). When at least one of the C_i's is large and positive, we get a small X, just as (at least) a single 0 for a child implies a 1 on a NAND tree.

starting player), Farhi et al. claim that the close-to-zero-energy eigenvectors have little support on the root of the tree. The eigenvalue equation at the vertex $x = 0$ right below the tree root then reads

$$\langle x = 0| H |\psi\rangle = e^{ip} + Re^{-ip} + Te^{ip} = \left(e^{-ip} + e^{ip}\right) + T\left(e^{ip} - e^{-ip}\right). \tag{6.49}$$

Continuity dictates $1 + R = T$. Assuming $E|0\rangle \approx 0$ then implies $T = 1$, perfect transmission for close to E=0 wavepackets. Small overlap on the root of the tree thus implies support on the exit runway of the graph. On the other hand, if the NAND-tree evaluates to 0, the close-to-zero-energy eigenvectors do have significant support on the root of the tree, which in turn implies no support on the exit line for these states. Viewed as a scattering problem, we can think of putting a close-to-zero-energy wavepacket on the input line of the graph, let it evolve, and then measure the system to determine whether the wavepacket has reflected from the tree (NAND=0), or passed through it (NAND=1). Let us sketch why this is the case, utilizing a the recursive evaluation of the tree.

When we look for the eigenstates of the adjacency matrix of the tree, we would like to have $H|E\rangle = E|E\rangle$. At vertex x, it means that

$$\langle x| H |E\rangle \;=\; E\langle x|E\rangle \tag{6.50}$$
$$\langle x| H |E\rangle \;=\; \langle p|E\rangle + \sum_i \langle c_i|E\rangle, \tag{6.51}$$

where c_i are the children of x and p is the parent vertex of x. The values of $\langle c_i|E\rangle$ and $\langle x|E\rangle$ thus determine $\langle p|E\rangle$ as

$$\langle p|E\rangle \;=\; E\langle x|E\rangle - \sum_c \langle c|E\rangle, \tag{6.52}$$

which is especially simple for $E = 0$. Denoting the ratios of amplitudes $C_i = \frac{\langle c_i|E\rangle}{\langle x|E\rangle}$ and $X = \frac{\langle x|E\rangle}{\langle p|E\rangle}$ as in Figure 6.6a) translates (6.52) to

$$X = \frac{1}{E - C_1 - C_2}. \tag{6.53}$$

We will now think about eigenstates with small $E > 0$. We now want to show the following correspondence between the ratios X, the overlaps $\langle x|E\rangle$ and the NAND value of the subtree

rooted at vertex x:

$$\text{small } X \quad \longleftrightarrow \quad \text{negligible overlap } \langle x|E\rangle \quad \longleftrightarrow \quad \text{NAND=1 at vertex } x$$
$$\text{large } X \quad \longleftrightarrow \quad \text{reasonable overlap } \langle x|E\rangle \quad \longleftrightarrow \quad \text{NAND=0 at } x, \text{NAND=1 at } p.$$

Let us start at a vertex x which is a leaf. It has no children (whose C_i's are thus zero), making its X large and positive. Using (6.53), we can work out the ratios down the tree as in Figure 6.6b). When both C_i's are small and not positive, X is going to be large and positive. On the other hand, if at least one of the C_i's is large, X is going to be small and negative. In this sense, (6.53) evaluates the NAND operation, as described by the mapping above. What does it mean for the case when x is the root of the tree, and p, its parent is already part of the runway? Small X implies that the overlap $\langle x|E\rangle$ is tiny, while the whole tree has NAND=1. On the other hand, a large ratio X would imply NAND=0 for the whole tree, and $\langle x|E\rangle$ not entirely small.

This was a sketch of the actual proof of Farhi et al. [83], who show that for eigenstates of H with $|E| < \frac{\epsilon}{16\sqrt{N}}$, the overlap at the root in the NAND=1 case is no larger than ϵ. This happens because a "small" overlap on some far-out vertices can not grow much more than by doubling when we go down the tree from the leaves two levels at a time using (6.53).

The peculiar scattering properties of the NAND tree (arising from the small overlap of small E states on the root of the NAND=1 trees) shown by Farhi et al. are retained for non-regular NAND trees, as shown by Ambainis et al. [85]. One extra ingredient in their approach is that they shorten the runway to just 3 vertices, and add extra weight to the connections. Furthermore, evaluating whether the NAND=0 or 1 for the tree is done by phase-estimation of the quantum walk on this graph.

6.4 Quantum Walks and Universal Quantum Computation

We have investigated continuous-time quantum walks with Hamiltonians that are adjacency matrices of graphs, i.e. they contain only 0's and 1's (or possibly, with some weights at the edges). The time-evolution of simple initial states can be used for searching (on regular lattices), to traverse graphs in interesting (e.g. for glued trees) ways, or to investigate graph properties (such as evaluating NAND trees). Thanks to Childs et al. [91,92], there is one more surprising application – *universal quantum computation*. It comes in two varieties.

First, there is a way to translate an arbitrary unitary transformation on n-qubits into the evolution of a continuous-time quantum walk on a non-weighted graph with maximum vertex degree 3 [91]. This idea has also been later translated to the *discrete* quantum walk model by Lovett et al. [93]. Although this first model is universal for quantum computation, we can not hope to construct the graph physically, as the number of vertices is exponential in n. However, this implies that simulating a quantum walk (the evolution of a single excitation) on a sparse, easily computable and describable, unweighted graph is as difficult as general quantum computation. Thus, it is one more example of a problem that can be asked purely classically, in terms of $0/1$ matrices, but which has a deep relationship to quantum computation (cf. for example [94]).

Second, a novel idea recently appeared in [92]. Instead of using a single walker (a wavepacket with a certain momentum) and a "wire" for each of the 2^n computational basis states, Childs et al. decided to use multiple walkers – one walker per qubit. The state of a qubit is encoded by
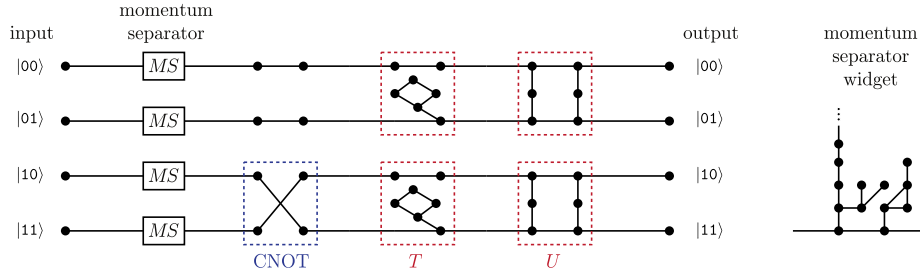
Fig. 6.7. A graph corresponding to a 2-qubit circuit has 4 wires. It starts with momentum separator widgets (depicted in detail on the right), and then 3 gate widgets. First, the CNOT gate between the two qubits, then the T gate (6.55) on the second qubit, and finally a U gate (6.56) on the second qubit.

an excitation moving in real time on a dual-rail[50]. Single-qubit gates are performed just as in the first model, while they found a way to implement a 2-qubit gate (C-PHASE) by scattering of excitations on two neighboring wires via a gadget graph connecting the two wires (utilizing an extra ancillary wavepacket with a different momentum). The overall size of the graph is no longer exponential, now it is only $O(nL)$, with n the number of qubits used and L the number of gates in the circuit. The analysis of the underlying scattering process is beyond the scope of this article, so we choose to explain only the first, simpler (but exponential in space) construction.

Let us now sketch the first idea. How can we achieve arbitrary unitary transformations just from letting a continuous quantum walk run on a non-weighted graph? For an n-qubit unitary, we will need 2^n quantum wires (corresponding to the 2^n basis states) of $poly(n)$ length, as in Figure 6.7. These wires will be connected by gate widgets, according to what quantum circuit we want to apply. If we worked with infinitely long wires, the eigenvectors of this Hamiltonian would be plane waves $|\tilde{k}\rangle$ characterized by their momenta k, with

$$\langle x|\tilde{k}\rangle = e^{ikx}, \tag{6.54}$$

at location x and normalized to $\langle \tilde{k}'|\tilde{k}\rangle = 2\pi\delta(k - k')$, corresponding to eigenvalues $2\cos k$, as we have seen for the walk on a line in Section 6.1.1. The graph here consists of incoming and outgoing lines (not infinite), plus the the graph widgets on which the plane waves can scatter. On the lines, the eigenvectors of H can only be linear combinations of the plane waves traveling right or left (with momenta $\pm k$), possibly also with imaginary momenta, giving bound states. Given an incoming plane wave with particular momentum, scattering theory allows us to compute the transition (and reflection) coefficients for the wave on the incoming/outgoing lines. The graph widgets in [91] are constructed in such a way that the reflection coefficients for a plane wave with momentum $k = -\frac{\pi}{4}$ are exactly zero, while the transmission amplitudes correspond to desired unitary transformations.

Universal computation requires interaction of qubits. This part is simple, as a CNOT gate between qubits l and m will be implemented by a crossing of the wires $\cdots 1_l \cdots 0_m \cdots$ and

---

[50]A dual rail construction has two wires per qubit. An excitation on the first line corresponds to the state $|0\rangle$, while an excitation on the second wire corresponds to $|1\rangle$. The state on a dual-rail graph can also be in a coherent superposition of $|0\rangle$ and $|1\rangle$ – encoding a qubit.

$\cdots 1_l \cdots 1_m \cdots$ as in Figure 6.7. However, besides the CNOT operation, we also need a universal set of single-qubit gates. A single qubit gate on the m-th qubit can be implemented by adding a graph widget to the $\cdots 0_m \cdots$ and $\cdots 1_m \cdots$ wires. Having two types of gates at hand is sufficient for single-qubit universality. First, we need the $\frac{\pi}{8}$ gate[51]

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}, \tag{6.55}$$

which can be implemented by the widget in Figure 6.7 (repeated for all values of qubits not involved in the T gate). Second, we can utilize a basis-changing gate

$$U = -\frac{1}{\sqrt{2}} \begin{bmatrix} i & 1 \\ 1 & i \end{bmatrix}, \tag{6.56}$$

implemented as in Figure 6.7. While (T, U) is is not the usual universal gate set (x and z rotations, or phase gate and Hadamard), observe that we can implement the Hadamard gate[52] H using T and U as $H = iT^2UT^2$, and the phase gate simply as T^2.

Finally, because the single qubit gate widgets require a plane wave with specific momentum to work, we need to use a momentum separator/filter. This is a graph widget with transmission coefficients close to zero for momenta away from $k = -\frac{\pi}{4}$ and $\frac{3\pi}{4}$, while it lets the selected ones through completely. Its second part separates wavepackets with these two momenta in time, as the effective path length of the widget is different for them. The computation then consists of initializing the system with wave packets on corresponding input lines (e.g. only on the $00 \cdots 0$ line, and letting the system evolve with the graph adjacency matrix Hamiltonian. After time linear in the number of gate applications, we measure the amplitude of the desired output line[53].

6.5   Connecting Continuous Time and Discrete Time Quantum Walks

It is now time to look at the similarities and differences for the two approaches to quantum walks. Coined or two-register discrete-time quantum walks are directly implementable using quantum circuits. On the other hand, continuos-time quantum walks are more natural in their interpretation as the dynamics of an excitation in a physical system, are generally easier to analyze, but require small degree of the vertices. Both methods have brought forth successful algorithms (discrete: graph search, element distinctness, continuous: glued trees traversal, NAND tree evaluation), many of which have been soon translated from one model to the other. We can see a clear analogy for the search algorithm on a 2D lattice, where the continuous-time quantum walk [80] needs a "coin" – an extra spin degree of freedom – to work as fast as the discrete quantum walk algorithm. Meanwhile, a successful translation of the glued-trees traversal walk to a discrete-time quantum walk utilizing a 3-headed coin [27] should be possible, but we don't have a rigorous analysis showing it. Worse than that, until recently we didn't know how to formulate a CTQW algorithm for the element-distinctness algorithm discussed in Section 4.7.

---

[51]Up to an overall phase, it is equivalent to multiplying the 0-amplitude by $e^{i\frac{\pi}{8}}$ and the 1-amplitude by $e^{-i\frac{\pi}{8}}$.

[52]The Hadamard gate switches between the x and z-bases as $H|x\pm\rangle = |z\pm\rangle$.

[53]Note that for BQP universality it is enough to be able to initialize the system in the state $|0\rangle^{\otimes n}$, apply a quantum circuit and make a projective measurement onto the state $|0\rangle^{\otimes n}$ afterwards.
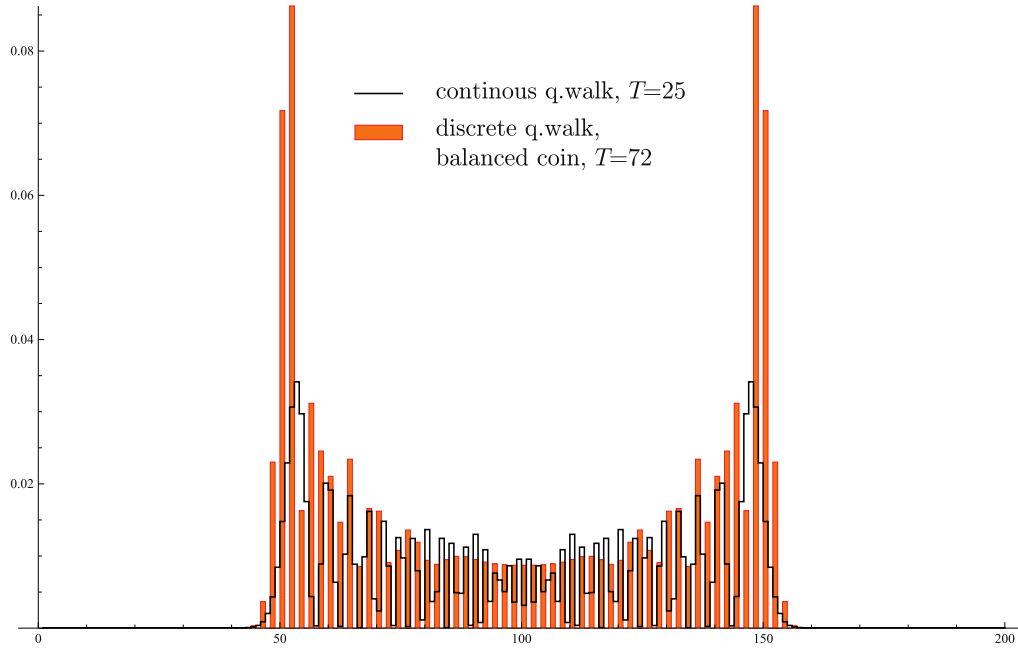
Fig. 6.8. Comparison of probabilities of finding the walker at a given spot for continuous and discrete quantum walks (balanced coin (3.22), initial state $|x = 100\rangle \otimes (|\leftarrow\rangle + |\rightarrow\rangle)/\sqrt{2}$) on a line, starting at the center ($x = 100$). Note that the discrete quantum walk has nonzero probabilities only at odd sites. The speed of spreading is linear in both cases: 2 for the continuous and $1/\sqrt{2}$ for the discrete quantum walk.

Does there exist a simple correspondence between the two models, arriving at CTQW's as a short time-step limit of some DTQW? This can't be possible directly, as DTQW's are implemented in a larger Hilbert space – including a coin register besides a position register (or on two position registers in scattering quantum walks or in Szegedy's formulation of quantized Markov chains, see Chapter 5). However, the behavior of DTQW's and CTQW's is remarkably similar in several cases, e.g. in 1D (see Figure 6.8). In [95], Andrew Childs has found a way of reproducing the dynamics of any Hamiltonian (thus including a CTQW) as a limit of discrete-time quantum walks. This in turn has implications for Hamiltonian simulation. Let us shortly sketch this construction.

Our goal is to take a CTQW with Hamiltonian H, and find a corresponding DTQW (in a limiting sense). The Hilbert space for the CTQW is spanned by the states $|j\rangle$. First, we will find a particular DTQW in Szegedy's two-register formulation, implemented in the space spanned by the vectors $|j, k\rangle$. To go between the state space of the CTQW and the DTQW, we will use an isometry

$$T := \sum_{j=1}^{N} |\psi_j\rangle \langle j|, \tag{6.57}$$

defined in the following way. Let $|d\rangle = \sum_{j=1}^{N} d_j |j\rangle$ be the largest-eigenvalue eigenvector of the matrix $\mathrm{abs}(H) = \sum_{j,k} |H_{jk}| |j\rangle \langle k|$ (whose elements are the absolute values of the elements of H). From the Perron-Frobenius theorem, we know that all d_j must be nonnegative. We denote the largest eigenvalue of $\mathrm{abs}(H)$ as $\|\mathrm{abs}(H)\|$. The orthonormal (easily checkable) states $|\psi_j\rangle$ in the isometry (6.57) are then defined as

$$|\psi_j\rangle := \frac{1}{\sqrt{\|\mathrm{abs}(H)\|}} \sum_{k=1}^{N} \sqrt{H_{jk}^* \frac{d_k}{d_j}} \, |j,k\rangle. \tag{6.58}$$

Using the states $|\psi_j\rangle$, we can define this discrete-time quantum walk:

1. Map an initial state $|j\rangle$ to a two-register state $|\psi_j\rangle$ using the isometry T.

2. Apply a reflection about the span of the states $|\psi_j\rangle$, swap the registers and add a $\frac{\pi}{2}$ phase, i.e. apply

$$U = iS\left(2 \sum_j |\psi_j\rangle \langle \psi_j| - \mathbb{I}\right) = iS(2TT^\dagger - \mathbb{I}) \tag{6.59}$$

3. Repeat step 2 (a given number of times). After that, use the inverse isometry $T^\dagger$ to get back to the original state space.

This DTQW is interesting, because if the rescaled eigenvalues λ of $\frac{H}{\|\mathrm{abs}(H)\|}$ are small, the eigenvalues of the DTQW can be shown to be approximately $\mu_\pm \approx \pm e^{\pm i\lambda}$. Furthermore, with additional rotations at the beginning and end, we obtain an approximation (valid up to $O(\lambda)$) of the evolution according to a CTQW for time τ

$$e^{-i\frac{H}{\mathrm{abs}(H)}\tau} = \sum_\lambda e^{-i\lambda\tau} |\lambda\rangle \langle \lambda| \approx T^\dagger \frac{(1-iS)}{\sqrt{2}} (iU)^\tau \frac{(1+iS)}{\sqrt{2}} T, \tag{6.60}$$

in the form of a DTQW with τ reflections and register-swaps.

Still, this is working only when all of the λ's are small, i.e. when $h = \frac{\|H\|}{\|\mathrm{abs}(H)\|}$ is small. We can make h small by taking a lazy walk instead of the original one. A lazy walk makes a step only with a very small probability ϵ. The trick is to enlarge the Hilbert space, and instead of T (6.57), use the isometry T_ϵ with modified states

$$|\psi_j^\epsilon\rangle := \sqrt{\epsilon} |\psi_j\rangle + \sqrt{1-\epsilon} |\perp_j\rangle, \tag{6.61}$$

where $|\perp_j\rangle$'s is a set of orthogonal states, orthogonal also to all of the $|\psi_j\rangle$'s and the states that come from $|\psi_j\rangle$'s by swapping their registers. The reflection about the span of $|\psi_j^\epsilon\rangle$ will thus have only a small part that corresponds to the reflection about the span of $|\psi_j\rangle$. It can be shown that the modified simulation procedure acts the same way as the previous one, but as if we evolved with ϵH instead of H.

Altogether, when we want to simulate H for time t up to precision δ, we choose a large enough number of simulation steps $\tau = O\left((\|H\| t)^{\frac{3}{2}} \delta^{-\frac{1}{2}}, \|\mathrm{abs}(H)\| t\right)$, and use a "lazy" walk

with $\epsilon = \|\mathrm{abs}(H)\| \frac{t}{\tau}$. For this all to work, we need to be able to analyze the principal eigenvector of $\mathrm{abs}(A)$. However, it is also possible to use different states $|\psi_j\rangle$ without the knowledge of the principal eigenvector, but that might in turn result in some increase of the required simulation resources [95]. Using this translation from CTQW to DTQW (and vice versa!), Childs has obtained a CTQW version of the element distinctness algorithm with $O(N^{\frac{2}{3}})$ queries (in the Hamiltonian query model) and a DTQW version of the randomly glued-tree traversal algorithm, with a polynomial runtime.

6.6   Summary

Continuous-time quantum walks (CTQW) are a powerful tool for describing the dynamics of an excitation in a system. The Hamiltonian of the system can be as simple as the adjacency matrix of a graph. We have seen that the spreading of a wavepacket in this model can be similar to DTQW and that it can be used in lattice-search algorithms (see Section 6.2). Furthermore, the dynamics have interesting properties and can be analyzed with usual quantum-mechanical tools – utilizing symmetries allowed us to see the simple movement of superpositions in the randomly-glued tree traversal problem (see Section 6.1.2), while investigating the scattering properties of graphs resulted in the NAND-tree evaluation algorithm (see Section 6.3). In fact, scattering properties of simple graphs can be used as a universal model of quantum computation (see Section 6.4). Finally, we have looked at how CTQW can be connected to discrete-time quantum walks in Section 6.5.

Acknowledgements

Appendices

A   Limiting Distribution of Classical Random Walks

To show the results of Sec. 2.2, let us consider random walk on graph $\mathcal{G} = (V, E)$, $|V| = N$, given by eq. (2.2) where at each position the walker decides unbiasedly, where to go next. Further suppose, that the graph is connected (for unconnected graphs we would consider its connected subgraphs) and non-bipartite. The transition matrix M of this random walk can be expressed as $M = AD^{-1}$ where

$$D = \mathrm{diag}\,(d(1), d(2), \ldots, d(N))$$

is the diagonal matrix having degrees of vertices on its diagonal and A is the adjacency matrix

$$A_{ij} = \begin{cases} 1, \text{ if } ij \in E, \\ 0, \text{ otherwise.} \end{cases}$$

For undirected graphs M is not symmetric in general and may not have spectral decomposition. On the other hand A is symmetric. Applying a similarity transformation on M,

$$D^{-1/2}MD^{1/2} = D^{-1/2}AD^{-1/2} \equiv Q,$$

we see, that M can be spectrally decomposed as well as it has the same eigenvalues as Q. Let $\lambda_1, \lambda2, \ldots, \lambda_N$ be the eigenvalues of Q and $v_j, j = 1, 2, \ldots, N$ its corresponding eigenvectors. Then eigenvectors $w_j, j = 1, 2, \ldots, N$ of M are connected to corresponding eigenvectors of Q by equality $w_j = D^{1/2}v_j$. We can further suppose that the eigenvalues are ordered,

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N.$$

Now we can write

$$Q = \sum_j \lambda_j v_j v_j^T$$

with vectors v_j fulfilling normalization $v_j^T v_i = \delta_{ij}$ and for the purpose of the evaluation of evolution $p(m) = M^m p(0)$ we write

$$
\begin{aligned}
p(m) &= M^m p(0) = \left(D^{1/2}QD^{-1/2}\right)^m p(0) = D^{1/2}Q^m D^{-1/2}p(0) \\
&= \sum_j \lambda_j^m D^{1/2}v_j v_j^T D^{-1/2}p(0).
\end{aligned}
\tag{A.1}
$$

Now that we know, that M can be spectrally decomposed, we can concetrate on its eigenvalues, which are the same also for Q. We will show, that all the eigenvalues lie in the range $-1 \leq \lambda_j \leq 1$ and that π is the unique eigenvector with eigenvalue 1.

Lemma 5 *Eigenvalues of matrix M lie in the interval $[-1; 1]$ and the vector π is the unique eigenvector of M with eigenvalue* 1.

To prove this, let us consider some eigenvector w with eigenvalue λ and choose such j, that $|w(j)|/d(j)$ is maximal, where $w(j)$ is the j-th component of vector w. In other words we consider inequality

$$\frac{|w(j)|}{d(j)} \geq \frac{|w(i)|}{D(i)}$$

for all indices i. We also know, that

$$\lambda w(j) = \sum_{i:\ ij \in E} M_{ji} w(i) = \sum_{i:\ ij \in E} \frac{w(i)}{d(i)}.$$

Now we write

$$
\begin{aligned}
|\lambda||w(j)| &= \left| \sum_{i:\ ij \in E} \frac{w(i)}{d(i)} \right| \leq \sum_{i:\ ij \in E} \frac{|w(i)|}{d(i)} \\
&\leq \sum_{i:\ ij \in E} \frac{|w(j)|}{d(j)} = |w(j)|.
\end{aligned}
\tag{A.2}
$$

This shows, that $|\lambda_j| \leq 1$ for all eigenvectors v_j. Moreover, equality can be obtained only when

$$\frac{|w(j)|}{d(j)} = \frac{|w(i)|}{D(i)} \tag{A.3}$$

for all neighbors i of j in the graph. The equality Eq. (A.3) can be easily extended to all vertices, i.e. to all indices i, as the graph is connected and finite, meaning, that whenever the equality would not hold, there would be at least one edge in the graph connecting vertices with different ratios which would be in contradiction with the proof. We see, that eigenvalue $\lambda_1 = 1$ uniquely determines the eigenstate, as the one fulfilling condition Eq. (A.3): we can always consider $w_1(1)$ to be positive, in which case we can proceed in a similar manner as in Eq. (A.2), but without the absolute values to find, that

$$\frac{w_1(j)}{d(j)} = \frac{w_1(i)}{D(i)}.$$

Taking this ratio to be equal to $1/2|E|$ we recover π:

$$\pi = \frac{1}{2|E|} (d(1), d(2), \ldots, d(N))^T.$$

This concludes the proof.

As π is a multiple of $w_1 = D^{1/2} v_1$ we can also find, that

$$v_1 = \sqrt{2|E|} D^{-1/2} \pi.$$

We can conclude now, that

$$1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_N \geq -1.$$

Furthermore, from Eq. (A.1) we can observe, that in the limiting case of $m \to \infty$ all the terms with eigenvalues λ_j, $j = 2, 3, \ldots, N$ tend to zero and so

$$\lim_{m \to \infty} p(m) = D^{1/2} v_1 v_1^T D^{-1/2} p(0) = \pi \left(2|E| \pi^T D^{-1} p(0)\right) = \pi.$$

This shows, that the limiting distribution is indeed π.

Exercise 17 Show that $2|E| \pi^T D^{-1} p = 1$ for any probability distribution p.

Exercise 18 In previous proofs we have disregarded the possibility for eigenvalue of M to be -1. Show, that we could do it, as only bipartite graphs lead to this case. [Hint: Take the eigenvector with eigenvalue -1 and apply M^2 to it]

B   Evolution of Hadamard Walk in Detail

B.1   Method of Stationary Phase

The method of mathematical analysis called *the method of stationary phase*, taken e.g. from Ref. [16] and from advanced mathematical textbooks, allows us to estimate our integrals, Eqs. (3.11). Based on a fact, that in integrals of type

$$I(m) = \int_a^b g(k)e^{im\phi(k)}dk$$

the largest contribution comes from areas, where the oscillations in phase are small and that is near stationary points of function $\phi(k)$. Basically, under reasonable suppositions of smooth $g(k)$ that does not vanish in the only stationary point a of the order p of interval $[a; b]$ and with $t \to \infty$ we can make approximation

$$I(m) \sim g(a)\exp\left\{im\phi(a) + \text{sgn}\left[\phi^{(p)}(a)\right]\frac{i\pi}{2p}\right\}\left[\frac{p!}{m\left|\phi^{(p)}(a)\right|}\right]^{\frac{1}{p}}\frac{\Gamma\left(\frac{1}{p}\right)}{p},$$

where $\phi^{(p)}(a) \neq 0$ from assumption and sgn is a sign function.

Especially for the case of $p = 2$ we get

$$I(m) \sim \sqrt{\frac{\pi}{2m\left|\phi''(a)\right|}}g(a)\exp\left\{im\phi(a) + \text{sgn}\left[\phi''(a)\right]\frac{i\pi}{4}\right\}. \tag{B.1}$$

B.2   Hadamard Walk Evolution Approximation

Integrals from Eqs. (3.11) can be transformed to suit Eq. (B.1) first by setting $x = \lambda m$, when these integrals obtain form

$$I(m; \lambda) = \frac{1}{2\pi}\int_{-\pi}^{\pi} g(k)e^{i\phi(k;\lambda)}mdk, \tag{B.2}$$

where $\phi(k; \lambda) = k\lambda - \omega_k$ and $g(k)$ is either an even or an odd function. In this slightly generalised case, there are no stationary points for $\lambda > 1/\sqrt{2}$ or $\lambda < -1/\sqrt{2}$ and $I(m; \lambda)$ by getting with λ further from zero decreases exponentially fast. For[54] $\lambda \in (-1/\sqrt{2}; 1/\sqrt{2})$ we find stationary points $\pm k_\lambda \in [0; \pi]$ of order $p = 2$, where

$$\cos k_\lambda = \frac{\lambda}{\sqrt{1 - \lambda^2}}$$

and

$$\frac{\partial^2\phi}{\partial k^2}(\pm k_\lambda; \lambda) = \pm(1 - \lambda^2)\sqrt{1 - 2\lambda^2} \qquad (= -\omega''_{k_\lambda}).$$

---

[54]We do not care about boundary points much, as these do not play an important role.

Under these conditions, and by dividing the integration range $[-\pi; \pi]$ in Eq. (B.2) into four subintervals by points 0 and $\pm k_\lambda$ we find

$$I(m; \lambda) = \frac{2g(k_\lambda)}{\sqrt{2\pi m \left| \omega''_{k_\lambda} \right|}} \begin{cases} \cos \left[m\phi(k_\lambda; \lambda) + \dfrac{\pi}{4} \right], & \text{for } g \text{ even,} \\ \mathrm{i} \sin \left[m\phi(k_\lambda; \lambda) + \dfrac{\pi}{4} \right], & \text{for } g \text{ odd.} \end{cases}$$

Finally, for Eqs. (3.11), we obtain

$$\alpha^m(\lambda m) \quad \sim \quad \frac{2}{\sqrt{2\pi m \left| \omega''_{k_\lambda} \right|}} \cos \left[m\phi(k_\lambda; \lambda) + \frac{\pi}{4} \right], \tag{B.3a}$$

$$\beta^m(\lambda m) \quad \sim \quad \frac{2\lambda}{\sqrt{2\pi m \left| \omega''_{k_\lambda} \right|}} \cos \left[m\phi(k_\lambda; \lambda) + \frac{\pi}{4} \right], \tag{B.3b}$$

$$\gamma^m(\lambda m) \quad \sim \quad -\frac{2\sqrt{1 - 2\lambda^2}}{\sqrt{2\pi m \left| \omega''_{k_\lambda} \right|}} \sin \left[m\phi(k_\lambda; \lambda) + \frac{\pi}{4} \right]. \tag{B.3c}$$

Now the probability of being at position λt after m steps is

$$P^m(\lambda m) \sim \frac{2(1 + \lambda)}{\pi m (1 - \lambda^2)\sqrt{1 - 2\lambda^2}} [1 + \lambda \sqrt{2} \cos \theta], \tag{B.4}$$

where

$$\theta = 2m\phi(k_\lambda; \lambda) + \frac{\pi}{2} + \mu$$

and

$$\tan \mu = \frac{\sqrt{1 - 2\lambda^2}}{1 + 2\lambda}.$$

Note also following equality for further reference:

$$\mathrm{sgn} \left[\alpha^m(x) \beta^m(x) \right] = \mathrm{sgn}\, x. \tag{B.5}$$

Exercise 19 Prove Eq. (B.5) [Hint: use approximated formulas].

C   Catalan Numbers

Catalan numbers C_n [97] comprise a sequence of numbers that occur in combinatorics in many problems based on recurrence relations. For example, they enumerate the paths of a particle on a semi-infinite line starting from position 0 and returning to this position after $2n$ steps, i.e. when only the non-negative positions are available. In Section 2.2, we utilize Catalan numbers for computing a hitting time for a classical random walk, and in Section 3.5.3, they help us analyze a quantum walk on a line with an absorbing boundary.

Considering all the paths returning to origin after $2k$ steps ($k = 1, 2, \ldots, n$) we can arrive at recurrence relation

$$C_n = \sum_{k=1}^{n} C_{k-1} C_{n-k} = \sum_{k=0}^{n-1} C_k C_{(n-1)-k}$$

with $C_0 = 1$. Writing the generating function for Catalan numbers, $c(x) = \sum_{n=0}^{\infty} C_n x^n$ we can find, that

$$c(x) = 1 + x c^2(x). \tag{C.1}$$

Solving the quadratic equation we get

$$c(x) = \frac{1 - \sqrt{1 - 4x}}{2x} = \frac{2}{1 + \sqrt{1 - 4x}}. \tag{C.2}$$

The other solution of the quadratic equation is not acceptable, as $c(x)$ is a power series at $x = 0$ and so it cannot have a pole.

On the other hand we know, that power series for $\sqrt{1 + y}$ at $y = 0$ is

$$\sqrt{1 + y} = 1 - 2 \sum_{n=1}^{\infty} \binom{2n - 2}{n - 1} \frac{(-1)^n}{4^n} \frac{y^n}{n}.$$

Setting $y = -4x$ we can obtain also power series for Catalan generating function from Eq. (C.1),

$$c(x) = \frac{1 - \sqrt{1 - 4x}}{2x} = \sum_{n=0}^{\infty} \frac{1}{n + 1} \binom{2n}{n} x^n.$$

This shows, that Catalan numbers have form

$$C_n = \frac{1}{n + 1} \binom{2n}{n}. \tag{C.3}$$

For large n one can use approximation

$$C_n \simeq \frac{4^n}{n^{3/2} \sqrt{\pi}}. \tag{C.4}$$

Exercise 20 By employing

$$\sqrt{2\pi}n^{n+1/2}\mathrm{e}^{-n} \le n! \le \mathrm{e}n^{n+1/2}\mathrm{e}^{-n}$$

show that

$$C_{n+1} \ge \frac{2\sqrt{\pi}}{\mathrm{e}^2} \frac{4^k}{(k+1)^3/2}. \tag{C.5}$$

Exercise 21 Show that

$$C_{n+1} = \frac{2(2n+1)}{n+2}C_n. \tag{C.6}$$

D Grover's Fixed-point Search

Grover's algorithm [30] for searching for one of M marked items among N elements in its original form is not a fixed-point search. If we run it for longer than its optimal time, the probability of finding a marked element begins to decrease, and after some time, we even get back to our original initial state. In [96], Gil Brassard wrote "The quantum search algorithm is like baking a souffle. . . you have to stop at just the right time or else it gets burnt." When we view the Grover algorithm as a quantum walk, the same is true — the walker concentrates on a marked vertex at some special time, but then moves away from it. Could we modify the quantum algorithm (and the walk) to stay near a solution?

This algorithm can be modified [53, 54] to work as a fixed-point one, producing a solution with high probability even if we "overshoot" the runtime. However, it loses the square root speedup (in number of oracle calls). Nevertheless, it can be useful when the number of marked items is not small[55]. Let us sketch this modification, based on iterative phase-$\pi/3$ search.

First, let us assume we have an algorithm (a unitary transformation U plus a final measurement) which fails to produce a marked state with probability f, when starting in the uniform superposition state $|s\rangle$. We will iteratively intersperse its uses with other (oracle-calling) transformations to increase the probability of finding a solution. The additional ingredients are the selective phase shifts

$$R_s^\theta = e^{i\theta}\Pi_s + (\mathbb{I} - \Pi_s), \tag{D.1}$$

$$R_M^\phi = e^{i\phi}\Pi_M + (\mathbb{I} - \Pi_M), \tag{D.2}$$

for the uniform superposition $|s\rangle$ and the marked subspace M. Note that for $\theta = \phi = \pi$ these are the original reflection operators (up to a sign) from Grover's algorithm. However, let us now choose $\theta = \phi = \frac{\pi}{3}$ and look at the operation

$$U_1 = U R_s^{\frac{\pi}{3}} U^\dagger R_M^{\frac{\pi}{3}} U. \tag{D.3}$$

The original algorithm U fails with probability f (which can be even very close to 1). The unitary U thus transforms $|s\rangle$ as

$$U|s\rangle = \sqrt{f}|w'\rangle + \sqrt{1-f}|w\rangle, \tag{D.4}$$

with $|w\rangle$ a vector from the marked subspace, and $|w'\rangle$ some unmarked vector. Because of unitarity of U, there exists a vector $|s'\rangle$ orthogonal to $|s\rangle$, which is transformed by U as

$$U|s'\rangle = \sqrt{1-f}|w'\rangle - \sqrt{f}|w\rangle, \tag{D.5}$$

which allows us to express

$$|w\rangle = U\left(\sqrt{1-f}|s\rangle - \sqrt{f}|s'\rangle\right). \tag{D.6}$$

---

[55]When this algorithm is used as a subroutine in Section 5.4.3, the ratio of marked/unmarked items is a constant.

To compute the failure probability of the new algorithm, let us apply U_1 to the initial state $|s\rangle$.

$$U_1 |s\rangle = U R_s^{\frac{\pi}{3}} U^\dagger R_M^{\frac{\pi}{3}} U |s\rangle = U R_s^{\frac{\pi}{3}} U^\dagger R_M^{\frac{\pi}{3}} \left(\sqrt{f} |w'\rangle + \sqrt{1-f} |w\rangle \right) \tag{D.7}$$

$$= U R_s^{\frac{\pi}{3}} U^\dagger \left(\sqrt{f} |w'\rangle + e^{i\pi/3} \sqrt{1-f} |w\rangle \right) \tag{D.8}$$

$$= U R_s^{\frac{\pi}{3}} U^\dagger \left(\sqrt{f} |w'\rangle + \sqrt{1-f} |w\rangle + \left(e^{i\pi/3} - 1 \right) \sqrt{1-f} |w\rangle \right) \tag{D.9}$$

$$= U R_s^{\frac{\pi}{3}} U^\dagger \left(U |s\rangle + e^{2i\pi/3} \sqrt{1-f}\, U \left(\sqrt{1-f} |s\rangle - \sqrt{f} |s'\rangle \right) \right) \tag{D.10}$$

$$= U R_s^{\frac{\pi}{3}} \left(\left(1 + e^{2i\pi/3}(1-f) \right) |s\rangle - e^{2i\pi/3} \sqrt{(1-f)f} |s'\rangle \right) \tag{D.11}$$

$$= U \left(e^{i\pi/3} \left(1 + e^{2i\pi/3}(1-f) \right) |s\rangle - e^{2i\pi/3} \sqrt{(1-f)f} |s'\rangle \right) \tag{D.12}$$

$$= \left(e^{i\pi/3} - 1 + f \right) U |s\rangle - e^{2i\pi/3} \sqrt{(1-f)f}\, U |s'\rangle \tag{D.13}$$

After the final substitution for $U |s\rangle$ and $U |s'\rangle$, we collect the coefficients in front of $|w'\rangle$ and obtain the probability of failure for the new algorithm

$$|\langle w'| U_1 |s\rangle|^2 = \left| \sqrt{f} \left(e^{i\pi/3} - 1 + f \right) - e^{2i\pi/3}(1-f)\sqrt{f} \right|^2 \tag{D.14}$$

$$= f \left| \left(e^{i\pi/3} - 1 - e^{2i\pi/3} \right) + f \left(1 + e^{2i\pi/3} \right) \right|^2 \tag{D.15}$$

$$= f^3 \left| 1 + e^{2i\pi/3} \right|^2 = f^3. \tag{D.16}$$

The original failure probability was $f < 1$, so the new failure probability f^3 is smaller. This composite algorithm (D.3) reminds us of an error-correcting scheme.

Let us compute how the failure probability decreases with the number of queries. At each level of composition, it decreases as $f \to f^3$. Assuming the original algorithm U takes $c_0 = n$ oracle calls, the cost of the new approach U_1 is $c_1 = 3n + 1$. At the next level, composing $U_2 = U_1 R_s^{\pi/3} U_1^\dagger R_M^{\pi/3} U_1$ uses $c_2 = 9n + 3 + 1$ oracle calls.

Exercise 22 Show that the number of oracle calls for k-levels of composition becomes $q_k = 3^k n + \frac{1}{2} \left(3^k - 1 \right)$.

It turns out that we can even use $U_0 = \mathbb{I}$ (without any oracle calls) for our starting algorithm. Alternatively, we could use a single iteration from Grover's algorithm $U_G = R_s^\pi R_M^\pi$.

Exercise 23 For the strategy U_0 with $c_0 = 0$, show that the failure probability at the k-th level of composition $f_k = f_0^{3^k}$ decreases with the number of oracle calls as $f(q_k) = f_0^{2q_k+1} = \left(1 - \frac{M}{N} \right)^{2q_k+1}$.

Thus, the fixed-point quantum algorithm needs need $O \left(\frac{N}{M} \right)$ queries to find a marked item with reasonable probability. This is obviously inferior to the $O \left(\sqrt{\frac{N}{M}} \right)$ scaling of Grover's original algorithm, but can still be useful for ratios $\frac{M}{N}$ that are not too small. Finally, let us compare this scaling to the classical case, where the failure probability after querying q elements decreases with the number of oracle calls as $f(c_k) = \binom{N-M}{c_k} / \binom{N}{c_k} \approx \left(1 - \frac{M}{N} \right)^{c_k}$ for $M, c_k \ll N$.

References

[1] F. Galton: *Natural Inheritance* (The Macmillan Company, London, 1889).

[2] L. Bachelier: *Theorie de la speculation* (PhD thesis, Annales Scientifiques de l'Ecole Superieure **III-17,** 21, 1900).

[3] N. Wiener: *Differential space*, J. Math. Phys. **2,** 131 (1923).

[4] A. Einstein: *Über die von der molekularkinetischen Theorie der wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen*, Ann. Physik **17,** 549 (1905).

[5] R. Brown: *A brief account of microscopical observations made in the months of June, July and August, 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies*, Phil. Mag. **4,** 161 (1928).

[6] J. Perrin: *Mouvement brownien et réalité moléculaire*, Ann. Chim. Phys. **18,** 5 (1909).

[7] B.D. Hughes: *Random Walks and Random Environments*, Clarendon Press (1995)

[8] A. Sinclair: *Algorithms for Random Generation and Counting, a Markov Chain Approach*, Birkhauser (1993).

[9] Rajeev Motwani and Prabhakar Raghavan, Randomized algorithms, ACM Comput. Surv. **28, 1** 33–37 (March 1996).

[10] N. Metropolis, A.W. Rosenbluth, N.N. Rosenbluth, A.H. Teller, E. Teller: *Equation of State Calculations for Fast Computing Machines*, J. Chem. Phys. **21,** 1087 (1953).

[11] W.K. Hastings: *Monte Carlo Sampling Methods Using Markov Chains and Their Applications*, Biometrika **57,** 97 (1970).

[12] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi: *Optimization by Simulated Annealing*, Science **220,** 671 (1983).

[13] J. Černý: *Thermodynamical Approach to the Travelling Salesman Problem*, J. Opt. Theory Appl. **45,** 41 (1985).

[14] Y. Aharonov, L. Davidovich, N. Zagury: *Quantum random walks*, Phys. Rev. A **48,** 1687 (1993).

[15] D. Aharonov, A. Ambainis, J. Kempe, U. Vazirani: *Quantum walks on graphs*, in STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing, 50–59 (New York, NY, USA, ACM, 2001).

[16] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous: *One-dimensional quantum walks*, in STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing, 37–49 (New York, NY, USA, ACM, 2001).

[17] V. Kendon: *Decoherence in quantum walks — a review*, Mathematical Structures in Computer Science **17,** 1169 (2007).

[18] V. Kendon, B. Tregenna: *Decoherence can be useful in quantum walks*, Phys. Rev. A **67,** 042315 (2003).

[19] T.A. Brun, H.A. Carteret, A. Ambainis: *Quantum random walks with decoherent coins*, Phys. Rev. A **67,** 032304 (2003).

[20] J. Košík, V. Bužek, and M. Hillery: *Quantum walks with random phase shifts*, Phys. Rev. A **74,** 022310 (2006).

[21] M. Hillery, J. Bergou, E. Feldman: *Quantum walks based on an interferometric analogy*, Phys. Rev. A **68,** 032314 (2003).

[22] F.M. Andrade, M.G.E. da Luz: *Equivalence between discrete quantum walk models in arbitrary topologies*, Phys. Rev. A **80,** 052301 (2009).

[23] N. Shenvi, J. Kempe, K.B. Whaley: *Quantum random-walk search algorithm*, Phys. Rev. A **67,** 052307 (2003).

[24] V. Kendon: *Quantum Walks on General Graphs*, Int. J. Quantum Info. **4,** 791 (2006).

[25] S. Severini: *On the digraph of a unitary matrix*, SIAM J. Matrix Anal. Appl. (SIMAX) **25,** 295 (2003).

[26] A. Montanaro: *Quantum walks on directed graphs*, Quantum Information and Computation **7,** 93 (2007).

[27] B. Tregenna, W. Flanagan, R. Maile, V. Kendon: *Controlling discrete quantum walks: coins and initial states*, New Journal of Physics **5,** 83 (2003).

[28] Y. Omar, N. Paunkovic, L. Sheridan, S. Bose: *Quantum walk on a line with two entangled particles*, Phys. Rev. A **74,** 042304 (2006).

[29] C. Moore, A. Russell: *Quantum walks on the hypercube*, in RANDOM, 164–178 (2002).

[30] L.K. Grover: *Quantum mechanics helps in searching for a needle in a haystack*, Phys. Rev. Lett. **79,** 325 (1997).

[31] J. Kempe: *Discrete quantum walks hit exponentially faster*, Probability Theory and Related Fields **133,** 215 (2005).

[32] A. M. Childs, R. Kothari, *Simulating sparse Hamiltonians with star decompositions*, In proceedings of TQC 2010, Lecture Notes in Computer Science 6519, pp. 94-103 (2011).

[33] A.M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, D.A. Spielman, *Exponential algorithmic speedup by quantum walk*, Proceedings of the 35th ACM Symposium on Theory of Computing, pp. 59–68 (2003).

[34] E. Bach, S. Coppersmith, M.P. Goldschen, R. Joynt, J. Watrous: *One-dimensional quantum walks with absorbing boundaries*, Journal of Computer and System Sciences **69,** 562 (2004).

[35] T. Yamasaki, H. Kobayashi, H. Imai: *Analysis of absorbing times of quantum walks*, Phys. Rev. A **68,** 012302 (2003).

[36] L.K. Grover: *Quantum Computers Can Search Rapidly by Using Almost Any Transformation*, Phys. Rev. Lett. **80,** 4329 (1998).

[37] G. Brassard, P. Høyer, M. Mosca, A. Tapp: *Quantum Amplitude Amplification and Estimation*, preprint `arXiv:quant-ph/0005055`.

[38] A. Ambainis: *Quantum walk algorithm for element distinctness*, SIAM J. Comp. **37,** 210 (2007).

[39] A.M. Childs, J.M. Eisenberg: *Quantum algorithms for subset finding*, Quantum Information and Computation **5,** 593 (2005).

[40] F. Magniez, M. Santha, M. Szegedy: *Quantum Algorithms for the Triangle Problem*, in SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, 1109 (Philadelphia, PA, USA, 2005).

[41] H. Buhrman, R. Špalek: *Quantum Verification of Matrix Products*, in SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, 880 (New York, NY, USA, 2006).

[42] F. Magniez, A. Nayak: *Quantum Complexity of Testing Group Comutativity*, Algorithmica **48,** 221 (2007).

[43] A. Ambainis, J. Kempe, A. Rivosh: *Coins make quantum walks faster*, in SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, 1099 (Philadelphia, PA, USA, 2005).

[44] A. M. Childs, J. Goldstone, *Spatial search by quantum walk*, Physical Review A 70, 022314 (2004).

[45] D. Reitzner, M. Hillery, E. Feldman, V. Bužek: *Quantum Searches on Highly Symmetric Graphs*, Phys. Rev. A **79,** 012323 (2009).

[46] V. Potoček, A. Gábris, T. Kiss, I. Jex: *Optimized quantum random-walk search algorithms on the hypercube*, Phys. Rev. A, **79,** 012325 (2009).

[47] M. Boyer, G. Brassard, P. Høyer, A. Tapp: *Tight bounds on quantum searching*, Fortschritte der Physik **46,** 493 (1998).

[48] S.L. Braunstein, V. Bužek, M. Hillery: *Quantum-information distributors: Quantum network for symmetric and asymmetric cloning in arbitrary dimension and continuous limit,* Phys. Rev. A **63,** 052313 (2001).

[49] E. Feldman, M. Hillery, H.-W. Lee, D. Reitzner, H. Zheng, V. Bužek: *Finding structural anomalies in graphs by means of quantum walks*, Phys. Rev. A **82,** 040301(R) (2010).

[50] A. Tulsi: *Faster quantum-walk algorithm for the two-dimensional spatial search,* Phys. Rev. A **78,** 012310 (2008).

[51] M. Szegedy, *Quantum Speed-up of Markov Chain Based Algorithms*, Proc. of 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 32–41 (2004).

[52] M. Santha, *Quantum Walk Based Search Algorithms*, Proc. of 5th Theory and Applications of Models of Computation (TAMC08), Lectures Notes on Computer Science, vol. **4978,** pp. 31–46, 2008.

[53] L. K. Grover *A Different Kind of Quantum Search*, arXiv:quant-ph/0503205.

[54] T. Tulsi, L. K. Grover, and A. Patel, *A new algorithm for fixed point quantum search*, Quantum Info. Comput. **6 (6),** 483–494 (2006).

[55] G. Brassard, P. Høyer, and A. Tapp, *Quantum Counting*, Proc. of 25th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science, vol. **1443,** pp. 820–831, 1998. quant-ph/9805082.

[56] K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, F. Verstraete, *Quantum Metropolis sampling*, Nature **471,** 87–90, http://arXiv:0911.3635.

[57] M.-H.Yung, A. Aspuru-Guzik, *A Quantum-Quantum Metropolis Algorithm*, arXiv:1011.1468.

[58] C. Marriott and J.Watrous, *Quantum Arthur-Merlin games*, Computational Complexity, **14 (2)** 122–152 (2005).

[59] D. Nagaj, P. Wocjan, Y. Zhang, *Fast QMA Amplification*, QIC Vol. **9** No.11&12 (2009), 1053-1068 [arXiv:0904.1549].

[60] R. Somma, S. Boixo, H. Barnum, E. Knill, *Quantum Simulations of Classical Annealing Processes*, arXiv:0804.1571.

[61] L. Lovász and S. Vempala, *Simulated Annealing in Convex Bodies and an $O^*(n^4)$ Volume Algorithm*, Journal of Computer and System Sciences, vol. **72,** issue 2, pp. 392–417, 2006.

[62] M. Jerrum, A. Sinclair, and E. Vigoda, *A Polynomial-Time Approximation Algorithm for the Permanent of a Matrix Non-Negative Entries*, Journal of the ACM, vol. **51,** issue 4, pp. 671–697, 2004.

[63] M. Jerrum and A. Sinclair, *Polynomial-Time Approximation Algorithms for the Ising Model*, SIAM Journal on Computing, vol. **22,** pp. 1087–1116, 1993.

[64] I. Bezáková, D. Štefankovič, V. Vazirani and E. Vigoda, *Accelerating Simulated Annealing for the Permanent and Combinatorial Counting Problems*, SIAM Journal on Computing, vol. **37,** No. 5, pp. 1429–1454, 2008.

[65] D. Aldous, J. London Math. Soc. **253,** 564 (1982).

[66] M. Jerrum, L. Valiant and V. Vazirani, *Random Generation of Combinatorial Structures from a Uniform Distribution*, Theoretical Computer Science, vol. **43,** issue 2-3, pp. 169–188, 1986.

[67] D. Štefankovič, S. Vempala, and E. Vigoda, *Adaptive Simulated Annealing: A Near-Optimal Connection between Sampling and Counting*, Proc. of the 48th Annual IEEE Symposium on Foundations of Computer Science, pp. 183-193, 2007.

[68] P. Wocjan and A. Abeyesinghe, *Speed-up via Quantum Sampling*, Physical Review A, vol. **78,** pp. 042336, 2008.

[69] P. Wocjan et.al., *Quantum Algorithm for Approximating Partition Functions*, Physical Review A **80,** 022340 (2009).

[70] A. Childs, *Quantum Algorithms*, lecture notes, University of Waterloo, http://www.math.uwaterloo.ca/ amchilds/teaching/w11/qic823.html (2011).

[71] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

[72] P. Richter, *Quantum Speed-Up of Classical Mixing Processes*, Physical Review A, vol. **76,** 042306, 2007.

[73] R. Somma, S. Boixo, and H. Barnum, *Quantum Simulated Annealing*, arXiv:0712.1008.

[74] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, *Strengths and Weaknesses of Quantum Computing*, SIAM J. Comput. **26,** 5 (October 1997), 1510-1523.

[75] E. Farhi, S. Gutmann, *Analog Analogue of Digital Quantum Computation*, Phys. Rev. A **57,** 2403-2406 (1998).

[76] E. Farhi, J. Goldstone, S. Gutmann, M. Sipser, *Quantum computation by adiabatic evolution*, quant-ph/0001106 (2000).

[77] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C*, Chapter 6.5, Cambridge University Press (1992).

[78] E. Farhi, S. Gutmann, *Quantum Computation and Decision Trees*, Phys. Rev. A **58 (2),** 915 (1998).

[79] A. M. Childs, E. Farhi, and S. Gutmann, *An example of the difference between quantum and classical random walks*, Quantum Information Processing **1,** 35–43 (2002).

[80] A. M. Childs, J. Goldstone, *Spatial search and the Dirac equation*, Physical Review A **70,** 042312 (2004).

[81] A. M. Childs, E. Deotto, E. Farhi, J. Goldstone, S. Gutmann, A. J. Landahl, *Quantum search by measurement*, Physical Review A **66,** 032314 (2002).

[82] A. M. Childs, *Quantum Information Processing in Continuous Time*, Ph.D. Thesis, Massachusetts Institute of Technology (2004).

[83] E. Farhi, J. Goldstone, S. Gutmann, *A quantum algorithm for the Hamiltonian NAND tree*, Theory of Computing, Vol. **4,** no. 1, pp.169-190, 2007; quant-ph/0702144.

[84] A. M. Childs, R. Cleve, S. P. Jordan, D. L. Yonge-Mallo, *Discrete-query quantum algorithm for NAND trees*, Theory of Computing , vol. **5,** no. 1, pp. 119-123, 2009.

[85] A. Ambainis, A. M. Childs, B. W. Reichardt, R. Špalek, S. Zhang, *Every NAND formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer*, 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), 2007.

[86] P. Hoyer, R. Špalek, *Lower bounds on Quantum Query Complexity, Bulletin of the EATCS* **87,** 2005; quant-ph/0509153.

[87] D. Nagaj, R. D. Somma, M. Kieferová, *Quantum Speedup by Quantum Annealing* arXiv:1202.6257 (2012).

[88] D. Nagaj, *Universal 2-local Hamiltonian Quantum Computing*, Physical Review A **85,** 032330 (2012).

[89] D. Nagaj, *Local Hamiltonians in Quantum Computation*, Ph.D. thesis, MIT (2008), arXiv:0808.2117.

[90] M. Kieferová, D. Nagaj, *Quantum Walks on Necklaces and Mixing*, International Journal of Quantum Information Vol. **10,** Issue 2, 1250025 (2012).

[91] A. M. Childs, *Universal computation by quantum walk* Physical Review Letters **102,** 180501 (2009).

[92] A. M. Childs, D. Gosset, Z. Webb, *Universal computation by multi-particle quantum walk* arXiv:1205.3782 (2012).

[93] N. B. Lovett, S. Cooper, M. Everitt, M. Trevers, V. Kendon, *Universal quantum computation using the discrete-time quantum walk*, Physical Review A **81,** 042330 (2010).

[94] D. Janzing, P. Wocjan, *A PromiseBQP-complete String Rewriting Problem*, Quantum Information and Computation, vol. **10,** no. 3&4, pp. 234-257 (2010).

[95] A. Childs, On the Relationship Between Continuous- and Discrete-Time QuantumWalk, Commun. Math. Phys. **294,** 581-603 (2010).

[96] G. Brassard, *Searching a quantum phone book*, Science **275,** 627 (1997).

[97] *Catalan number*, Wikipedia, `http://en.wikipedia.org/wiki/Catalan_number` (2012).

Dr. Daniel Reitzner received his PhD in General and mathematical physics from Slovak Academy of Sciences and Comenius University in 2010. He was employed at the Research Center for Quantum Information between years 2010 and 2011. Since 2011 he is employed at the Technical University in Munich. He specializes in quantum walks and their algorithmic applications as well as in some foundational aspects of quantum mechanics connected to joint measurability and compatibility.
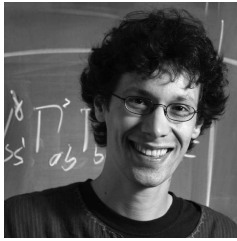
Dr. Daniel Nagaj received his PhD in theoretical physics from MIT in 2008. Between 2008 and 2012, he was employed at the Research Center for Quantum Information at the Institute of Physics of the Slovak Academy of Sciences. He mainly studies the computational capabilities of nature (quantum mechanical) - adiaibatic quantum computation, quantum walks, and Hamiltonian complexity. He is also interested in numerical methods for condensed-matter physics based on DMRG and tensor product states.

Prof. Vladimír Bužek has graduated at the Moscow State University (both MSc and PhD). His research interests include quantum optics, quantum information sciences, foundations of quantum theory, quantum thermodynamics, and quantum measurement theory. He is an author and co-author of more than 220 research papers and 15 chapters in monographs and books. These papers have been cited more than 7000 times ($H = 42$). During his academic career he was a visiting professor at a number of academic institutions including the Imperial College, London (UK), the National University of Ireland, Maynooth (Ireland), and the University of Ulm (Germany). He has been serving on various national and international advisory, scientific, and editorial boards. Prof. Bužek is the president of the Learned Society of the Slovak Academy of Sciences and a foreign corresponding member of the Austrian Academy of Sciences. He is a fellow of the Institute of Physics (UK) and of the Optical Society of America (USA). For his research achievements he was awarded the Ernst Abbe Medal and the International Commission for Optics Prize, the Humboldt Research Award (Germany) and the E.T.S Walton Award (Ireland).