Quantum algorithms for searching, resampling, and hidden shift problems

by

Māris Ozols

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Combinatorics & Optimization – Quantum Information

Waterloo, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis is on quantum algorithms. It has three main themes: (1) quantum walk based search algorithms, (2) quantum rejection sampling, and (3) the Boolean function hidden shift problem. The first two parts deal with generic techniques for constructing quantum algorithms, and the last part is on quantum algorithms for a specific algebraic problem.

In the first part of this thesis we show how certain types of random walk search algorithms can be transformed into quantum algorithms that search quadratically faster. More formally, given a random walk on a graph with an unknown set of marked vertices, we construct a quantum walk that finds a marked vertex in a number of steps that is quadratically smaller than the hitting time of the random walk. The main idea of our approach is to interpolate the random walk from one that does not stop when a marked vertex is found to one that stops. The quantum equivalent of this procedure drives the initial superposition over all vertices to a superposition over marked vertices. We present an adiabatic as well as a circuit version of our algorithm, and apply it to the spatial search problem on the 2D grid.

In the second part we study a quantum version of the problem of resampling one probability distribution to another. More formally, given query access to a black box that produces a coherent superposition of unknown quantum states with given amplitudes, the problem is to prepare a coherent superposition of the same states with different specified amplitudes. Our main result is a tight characterization of the number of queries needed for this transformation. By utilizing the symmetries of the problem, we prove a lower bound using a hybrid argument and semidefinite programming. For the matching upper bound we construct a quantum algorithm that generalizes the rejection sampling method first formalized by von Neumann in 1951. We describe quantum algorithms for the linear equations problem and quantum Metropolis sampling as applications of quantum rejection sampling.

In the third part we consider a hidden shift problem for Boolean functions: given oracle access to $f(x+s)$, where $f(x)$ is a known Boolean function, determine the hidden shift s. We construct quantum algorithms for this problem using the "pretty good measurement" and quantum rejection sampling. Both algorithms use the Fourier transform and their complexity can be expressed in terms of the Fourier spectrum of f (in particular, in the second case it relates to "water-filling" of the spectrum). We also construct algorithms for variations of this problem where the task is to verify a given shift or extract only a single bit of information about it.

Acknowledgements

First and most of all I would like to acknowledge my supervisors—I was extremely lucky to have *four* of them! Andrew Childs and Debbie Leung supervised me at University of Waterloo, and Jérémie Roland and Martin Rötteler supervised me at NEC Laboratories America in Princeton. I would also like to acknowledge all my close friends—you have taught me many lessons that are far more important than what I have learned at university. I cherish the time we have spent together—it has shaped my personality in significant ways. I would not have become who I am without you all.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

This thesis consists of six chapters that are grouped into three parts (see Fig. 1.1). All three parts are self-contained and can be read independently from each other. The last part uses a result from the previous part, but it is not essential to know the details of this result to understand the application.

	1. Introduction
Part I	2. Markov chains [KOR10, KMOR10] 3. Quantum search (adiabatic version) [KOR10] 4. Quantum search (circuit version) [KMOR10]
Part II	5. Quantum rejection sampling [ORR12]
Part III	6. Hidden shift problem

Figure 1.1: Structure of this thesis.

1.1.1 Part I

The first part of this thesis is on quantizing Markov chains and is based on joint work with Hari Krovi, Frédéric Magniez, and Jérémie Roland. Most of the research for this

part was done in the summer of 2009 during my internship at NEC Laboratories America, and it was completed during subsequent short visits to Princeton. It is based on papers [KOR10, KMOR10] and consists of three chapters (see Fig. 1.1). Chapter 2 is purely classical, while Chapter 3 and Chapter 4 describe an adiabatic and a circuit-based version of a quantum walk algorithm that finds a marked vertex in a graph quadratically faster than any randomized classical algorithm.

Chapter 2 contains results on Markov chains which are common to papers [KOR10] and [KMOR10]. Their proofs do not require any knowledge on quantum computing, but use only linear algebra and probability theory. The main result of this chapter is the formula

$$\mathrm{HT}(s) = \frac{p_M^2}{(1-s(1-p_M))^2}\,\mathrm{HT}(P,M) \tag{1.1}$$

from Theorem 2.22. It relates $\mathrm{HT}(P,M)$, the hitting time to Markov chain P, to the extended hitting time $\mathrm{HT}(s)$. This formula is used to analyze the quantum algorithms in Chapter 3 and Chapter 4, where it is important that $\mathrm{HT}(s)$ is monotonically increasing as a function of s.

In Chapter 3 we describe an adiabatic quantum algorithm for finding a marked vertex in a graph that has some set M of its vertices marked. A classical version of this algorithm is a random walk with transition matrix $P(s) = sP + (1-s)P'$, where parameter s slowly changes from $s = 0$ to $s = 1$ as the walk proceeds. During this evolution the transition matrix $P(s)$ changes from initial matrix P to the absorbing matrix P' which has no outgoing transitions from marked vertices. The main result of Chapter 3 is Theorem 3.6, which states that this algorithm finds a marked element with high probability in time $O(\sqrt{\mathrm{HT}(P,M)})$ where $\mathrm{HT}(P,M)$ is the corresponding time for the classical random walk P'.

In Chapter 4 we use ideas from the previous chapter to design a search algorithm in the quantum circuit model, which is based on eigenvalue estimation. This algorithm also achieves a quadratic speed-up over the classical case, which is the main result of this chapter (see Theorem 4.10). We also provide several variations of this algorithm that relax the assumptions in the main theorem, as well as apply these results for the spatial search problem on the 2D grid.

1.1.2 Part II

The second part of this thesis is on quantum rejection sampling. It consists of Chapter 5 and is almost identical to [ORR12], which is joint work with Martin Rötteler and Jérémie

Roland. This result originally came about as a byproduct of an algorithm for solving the Boolean function hidden shift problem, which is discussed in the third part of this thesis. I started to work on the hidden shift problem during my second internship at the NEC Laboratories America in the summer of 2010, when we discovered a "Grover-like" approach for solving this problem (see Sect. 6.4.2). The main ingredient in this approach is an amplitude amplification subroutine, which uses an oracle to implement a certain transformation between two unknown quantum states. This subroutine seemed to be of independent interest, so we decided to deviate from the original problem and study the abstract quantum state conversion problem solved by this subroutine. Only later we came to realize that it is the quantum equivalent of a simple probabilistic procedure known as rejection sampling which was studied by von Neumann [vN51].

Chapter 5 contains three main results: a query lower bound for the quantum resampling problem (see Sect. 5.3), the quantum rejection sampling algorithm (see Sect. 5.4), and applications for the linear systems of equations and quantum Metropolis sampling problems (see Sect. 5.5).

1.1.3 Part III

The third part of this thesis is based on unpublished work together with Andrew Childs, Martin Rötteler, and Jérémie Roland. It consists of Chapter 6 where we study a version of the hidden shift problem for Boolean functions. We assume that the underlying function is known and consider upper bounds on quantum query complexity by constructing quantum algorithms for this problem.

The most important part of Chapter 6 is Sect. 6.4 where three different approaches for solving this problem are considered. The first approach is based on the "pretty good measurement"—it corresponds to making all queries in parallel and performing a joint measurement of the obtained states (see Sect. 6.4.1). The second approach is based on quantum rejection sampling—it uses amplitude amplification and performs all queries sequentially (see Sect. 6.4.2). The third approach is due to [Röt10, GRR11] and resembles Simon's algorithm [Sim94]—it makes independent queries, each immediately followed by a measurement, and classically post-processes the obtained data (see Sect. 6.4.3).

We also consider the problem of extracting one bit of information about the hidden shift, namely, determining the inner product with a given string (see Sect. 6.5.1) as well as the problem of verifying a given shift (see Sect. 6.5.2). The idea of using decision trees to construct Boolean functions that have large fraction of the Fourier spectrum equal to zero was suggested by Dmitry Gavinsky (see Sect. 6.6.2).

1.2 Mathematical preliminaries

We assume that the reader is familiar with linear algebra and basics of quantum computing (for an introduction to quantum computing see [KLM07, KSV02]; for a more comprehensive overview see [NC10]). In particular, the reader should be familiar with two basic tools for constructing quantum algorithms: *quantum amplitude amplification* (see [KLM07, p. 163] or the original paper [BHMT00]) and *eigenvalue estimation* (see [KSV02, p. 125], [KLM07, p. 125.], [NC10, p. 221], or the papers [Kit95, CEMM98]).

Chapter 2

Semi-absorbing Markov chains and the extended hitting time

Contents

In this chapter we study a special type of Markov chains that can be described by a one-parameter family $P(s)$ corresponding to convex combinations of some chain P and the corresponding absorbing chain P'. Intuitively, $P(s)$ has states that are hard to escape

which is controlled by the interpolation parameter s. For this reason we call such chains "semi-absorbing". We will consider various properties of these chains as a function of the interpolation parameter s.

We begin by discussing some preliminaries and defining basic concepts related to Markov chains, such as ergodicity (Sect. 2.1). Next, we define the interpolated Markov chain $P(s)$ and consider various its properties, such as the stationary distribution and reversibility (Sect. 2.2). We proceed by applying these concepts to define and study the discriminant matrix of $P(s)$ which encodes all relevant properties, such as eigenvalues and the principal eigenvector of $P(s)$, but has a much more convenient form (Sect. 2.3). Finally, we define the hitting time HT and the extended hitting time $\mathrm{HT}(s)$ and relate the two via Theorem 2.22, which is the main result of this chapter (Sect. 2.4).

Results from this chapter will be used in Chapter 3 and Chapter 4 to construct quantum algorithms based on adiabatic evolution and discrete-time quantum walks, respectively.

2.1 Classical random walks

2.1.1 Preliminaries

Let us consider a Markov chain[1] on a discrete state space X of size n. Its transition probabilities can be described by a *row-stochastic matrix* P, i.e., an $n \times n$ matrix whose entries are real and non-negative and each row sums to one:

$$\forall x \in X : \sum_{y \in X} P_{xy} = 1. \tag{2.1}$$

Here P_{xy} is the probability to go from state x to y.

A Markov chain P with state space X has a corresponding *underlying directed graph* with n vertices labelled by elements of X, and directed arcs labelled by *non-zero* probabilities P_{xy} (see Fig. 2.1).

We will represent probability distributions by *row* vectors whose entries are also real, non-negative, and sum to one. If p is the initial probability distribution, then the probability distribution p' after one step of P is obtained by multiplying p by the transition

[1] We will use terms "random walk", "Markov chain", and "stochastic matrix" interchangeably. The same holds for "state", "vertex", and "element".

$$P = \begin{pmatrix} 0.7 & 0.3 & 0 \\ 0 & 0 & 1.0 \\ 0 & 0.8 & 0.2 \end{pmatrix}$$

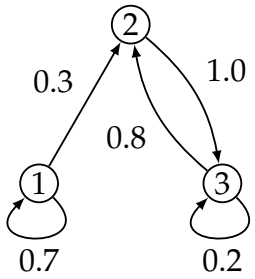

Figure 2.1: Markov chain P and the corresponding graph with transition probabilities.

matrix from the left hand side: $p' = pP$. A probability distribution π that satisfies $\pi P = \pi$ is called the *stationary distribution* of P.

For more background on Markov chains see, *e.g.*, [GS97, KS60, KS07].

2.1.2 Ergodicity

Definition 2.1. A Markov chain is called

- *irreducible*, if any state in the underlying directed graph can be reached from any other by a finite number of steps (*i.e.*, the graph is strongly connected);
- *aperiodic*, if there is no integer $k > 1$ that divides the length of every directed cycle of the underlying directed graph;
- *ergodic*, if it is both irreducible and aperiodic.

Equivalently, a Markov chain P is ergodic if there exists some integer $k_0 \geq 1$ such that all entries of P^{k_0} (and, in fact, of P^k for any $k \geq k_0$) are strictly positive. Some authors call such chains *regular* and use the term "ergodic" already for irreducible chains [GS97, KS60]. From now on we will almost exclusively consider only ergodic Markov chains.

Even though some of the Markov chain properties are independent from each other (such as irreducibility and aperiodicity), usually they are imposed in a specific order which is summarized in Fig. 2.2. As we impose more conditions, more can be said about the spectrum of P as discussed in the next section.

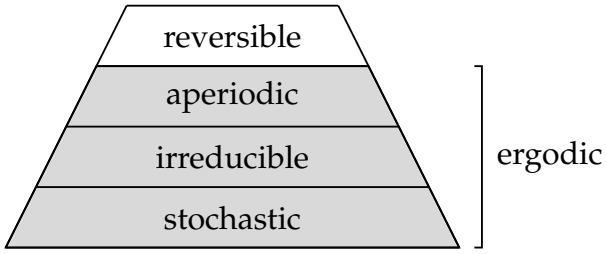

Figure 2.2: The order in which Markov chain properties from Definition 2.1 are typically imposed. Reversibility will be defined in Sect. 2.2.3.

2.1.3 Perron–Frobenius theorem

The following theorem will be very useful for us. It is essentially the standard Perron–Frobenius theorem [HJ90, Theorem 8.4.4, p. 508], but adapted for Markov chains. (This theorem is also known as the "Ergodic Theorem for Markov chains" [KS07, Theorem 5.9, p. 72].) The version presented here is based on the extensive overview of Perron–Frobenius theory in [Mey00, Chapter 8].

Theorem (Perron–Frobenius)**.** *Let P be a stochastic matrix. Then*

- *all eigenvalues of P are at most 1 in absolute value and 1 is an eigenvalue of P;*
- *if P is irreducible, then the 1-eigenvector is unique and strictly positive (i.e., it is of the form $c\pi$ for some non-vanishing probability distribution π and $c \neq 0$);*
- *if in addition to being irreducible, P is also aperiodic (i.e., P is ergodic), then the remaining eigenvalues of P are strictly smaller than 1 in absolute value.*

If P is irreducible but not aperiodic, it has some complex eigenvalues on the unit circle (which can be shown to be roots of unity) [Mey00, Chapter 8]. However, when in addition we also impose aperiodicity (and hence ergodicity), we are guaranteed that there is a unique eigenvalue of absolute value 1 and, in fact, it is equal to 1.

2.2 Semi-absorbing Markov chains

2.2.1 Definition

Assume that a subset $M \subset X$ of size $m := |M|$ of the states are marked (throughout this chapter we assume that M is not empty). Let P' be the Markov chain obtained from P by turning all outgoing transitions from marked states into self-loops (see Fig. 2.3). We call P' the *absorbing* version of P (see [KS60, Chapter III] and [GS97, Sect. 11.2]). Note that P' differs from P only in the rows corresponding to the marked states (where it contains all zeros on non-diagonal elements, and ones on the diagonal). If we arrange the states of X so that the unmarked states $U := X \setminus M$ come first, matrices P and P' have the following block structure:

$$P := \begin{pmatrix} P_{UU} & P_{UM} \\ P_{MU} & P_{MM} \end{pmatrix}, \qquad P' := \begin{pmatrix} P_{UU} & P_{UM} \\ 0 & I \end{pmatrix}, \tag{2.2}$$

where P_{UU} and P_{MM} are square matrices of size $(n-m) \times (n-m)$ and $m \times m$, respectively, while P_{UM} and P_{MU} are matrices of size $(n-m) \times m$ and $m \times (n-m)$, respectively.

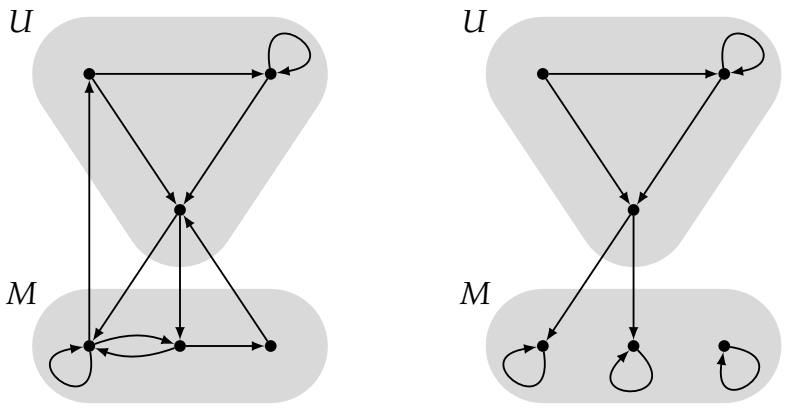

Figure 2.3: Directed graphs underlying Markov chain P (left) and the corresponding absorbing chain P' (right). Outgoing arcs from vertices in the marked set M have been turned into self-loops in P'.

Let us define an *interpolated* Markov chain that interpolates between P and P':

$$P(s) := (1-s)P + sP', \quad 0 \le s \le 1. \tag{2.3}$$

This expression has some resemblance with adiabatic quantum computation where similar interpolations are usually defined for quantum Hamiltonians [FGGS00]. Indeed, we will use the interpolated Markov chain $P(s)$ in Chapter 3 to construct an adiabatic quantum algorithm. Note that $P(0) = P$, $P(1) = P'$, and $P(s)$ has the following block structure:

$$P(s) = \begin{pmatrix} P_{UU} & P_{UM} \\ (1-s)P_{MU} & (1-s)P_{MM} + sI \end{pmatrix}. \tag{2.4}$$

Proposition 2.2. *If P is ergodic then so is $P(s)$ for $s \in [0,1)$. $P(1)$ is not ergodic.*

Proof. A non-zero transition probability in P remains non-zero also in $P(s)$ for $s \in [0,1)$. Thus the ergodicity of P implies that $P(s)$ is also ergodic for $s \in [0,1)$. However, $P(1)$ is not irreducible, since states in U are not reachable from M. Thus $P(1)$ is *not* ergodic. □

Proposition 2.3. $(P'^t)_{UU} = P_{UU}^t$.

Proof. Let us derive an expression for P'^t, the matrix of transition probabilities corresponding to t applications of P'. Notice that $\left(\begin{smallmatrix} a & b \\ 0 & 1 \end{smallmatrix}\right)\left(\begin{smallmatrix} a & b \\ 0 & 1 \end{smallmatrix}\right) = \left(\begin{smallmatrix} a^2 & ab+b \\ 0 & 1 \end{smallmatrix}\right)$. By induction we get

$$P'^t = \begin{pmatrix} P_{UU}^t & \sum_{k=0}^{t-1} P_{UU}^k P_{UM} \\ 0 & I \end{pmatrix}. \tag{2.5}$$

When restricted to U, it acts as P_{UU}^t. □

Proposition 2.4 ([GS97, Theorem 11.3, p. 417])**.** *If P is irreducible then* $\lim_{k\to\infty} P_{UU}^k = 0$.

Intuitively this means that the sub-stochastic process defined by P_{UU} eventually dies out or, equivalently, that the unmarked states of P' eventually get absorbed (by Prop. 2.3).

Proof. Let us fix an unmarked initial state x. Since P is irreducible, we can reach a marked state from x in a finite number of steps. Note that this also holds true for P'. Let us denote the smallest number of steps by l_x and the corresponding probability by p_x. Thus in $l := \max_x l_x$ steps of P' we are guaranteed to reach a marked state with probability at least $p := \min_x p_x > 0$, independently of the initial state $x \in U$. Notice that the probability to still be in an unmarked state after l^k steps is at most $(1-p)^k$ which approaches zero as we increase k. □

Proposition 2.5 ([KS60, Theorem 3.2.1, p. 46])**.** *If P is irreducible then $I - P_{UU}$ is invertible.*

Proof. Notice that

$$(I - P_{UU}) \cdot (I + P_{UU} + P_{UU}^2 + \cdots + P_{UU}^{k-1}) = I - P_{UU}^k \tag{2.6}$$

and take the determinant of both sides. From Prop. 2.4 see that $\lim_{k\to\infty} \det(I - P_{UU}^k) = 1$. By continuity, there exists k_0 such that $\det(I - P_{UU}^{k_0}) > 0$, so the determinant of the left-hand side is non-zero as well. Using multiplicativity of determinant, we conclude that $\det(I - P_{UU}) \neq 0$ and thus $I - P_{UU}$ is invertible. □

In the Markov chain literature $(I - P_{UU})^{-1}$ is called the *fundamental matrix* of P.

2.2.2 Stationary distribution

From now on let us demand that P is ergodic. Then according to the Perron–Frobenius Theorem it has a unique and non-vanishing stationary distribution π. Let π_U and π_M be row vectors of length $n - m$ and m that are obtained by restricting π to sets U and M, respectively. Then

$$\pi = \begin{pmatrix} \pi_U & \pi_M \end{pmatrix}, \qquad \pi' := \begin{pmatrix} 0_U & \pi_M \end{pmatrix} \tag{2.7}$$

where 0_U is the all-zeroes row vector indexed by elements of U and π' satisfies $\pi' P' = \pi'$.

Let $p_M := \sum_{x\in M} \pi_x$ be the probability to pick a marked element from the stationary distribution. In analogy to the definition of $P(s)$ in Eq. (2.3), let $\pi(s)$ be a convex combination of π and π', appropriately normalized:

$$\pi(s) := \frac{(1-s)\pi + s\pi'}{(1-s) + sp_M} = \frac{1}{1 - s(1-p_M)} \begin{pmatrix} (1-s)\pi_U & \pi_M \end{pmatrix}. \tag{2.8}$$

Proposition 2.6. *$\pi(s)$ is the unique stationary distribution of $P(s)$ for $s \in [0,1)$. At $s = 1$ any distribution with support only on marked states is stationary, including $\pi(1)$.*

Proof. Notice that

$$(\pi - \pi')(P - P') = \begin{pmatrix} \pi_U & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ P_{MU} & P_{MM} - I \end{pmatrix} = 0 \tag{2.9}$$

which is equivalent to

$$\pi P' + \pi' P = \pi P + \pi' P'. \tag{2.10}$$

Using this equation we can check that $\pi(s)P(s) = \pi(s)$ for any $s \in [0,1]$:

$$\big((1-s)\pi + s\pi'\big)\big((1-s)P + sP'\big) \tag{2.11}$$
$$= (1-s)^2\pi P + (1-s)s(\pi P' + \pi' P) + s^2\pi' P' \tag{2.12}$$
$$= (1-s)^2\pi + (1-s)s(\pi + \pi') + s^2\pi' \tag{2.13}$$
$$= \big((1-s)\pi + s\pi'\big)\big((1-s) + s\big) \tag{2.14}$$
$$= (1-s)\pi + s\pi'. \tag{2.15}$$

Recall from Prop. 2.2 that $P(s)$ is ergodic for $s \in [0,1)$ so $\pi(s)$ is the unique stationary distribution by Perron–Frobenius Theorem. Since P' acts trivially on marked states, any distribution with support only on marked states is stationary for $P(1)$. □

2.2.3 Reversibility

Definition 2.7. Markov chain P is called *reversible* if it is ergodic and satisfies the so-called *detailed balance condition*

$$\forall x, y \in X : \pi_x P_{xy} = \pi_y P_{yx} \tag{2.16}$$

where π is the unique stationary distribution of P.

Intuitively this means that the net flow of probability in the stationary distribution between every pair of states is zero. Note that Eq. (2.16) is equivalent to

$$\operatorname{diag}(\pi)\, P = P^{\mathsf{T}} \operatorname{diag}(\pi) = \big(\operatorname{diag}(\pi)P\big)^{\mathsf{T}} \tag{2.17}$$

where $\operatorname{diag}(\pi)$ is a diagonal matrix whose diagonal is given by vector π. Thus Eq. (2.16) is equivalent to saying that matrix $\operatorname{diag}(\pi)P$ is symmetric.

Proposition 2.8. *If P is reversible then so is $P(s)$ for any $s \in [0,1]$. Hence, $P(s)$ satisfies the extended detailed balance equation*

$$\forall s \in [0,1],\ \forall x, y \in X : \pi_x(s) P_{xy}(s) = \pi_y(s) P_{yx}(s). \tag{2.18}$$

Proof. First, notice that the absorbing walk P' is reversible[2] since

$$\operatorname{diag}(\pi')P' = \begin{pmatrix} 0 & 0 \\ 0 & \operatorname{diag}(\pi_M) \end{pmatrix} \begin{pmatrix} P_{UU} & P_{UM} \\ 0 & I \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & \operatorname{diag}(\pi_M) \end{pmatrix} = \operatorname{diag}(\pi') \tag{2.19}$$

[2]Strictly speaking, the definition of reversibility also includes ergodicity for the stationary distribution to be uniquely defined. However, we will relax this requirement for P' since, by continuity, π' is the natural choice of the "unique" stationary distribution.

which is symmetric. Next, notice that

$$\operatorname{diag}(\pi-\pi')(P-P') = \begin{pmatrix}\operatorname{diag}(\pi_U) & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}0 & 0\\ P_{MU} & P_{MM}-I\end{pmatrix} = 0 \tag{2.20}$$

which gives us an analogue of Eq. (2.10):

$$\operatorname{diag}(\pi')P + \operatorname{diag}(\pi)P' = \operatorname{diag}(\pi)P + \operatorname{diag}(\pi')P'. \tag{2.21}$$

Here the right-hand side is symmetric due to reversibility of P and P', thus so is the left-hand side. Using this we can check that $P(s)$ is reversible:

$$\operatorname{diag}\big((1-s)\pi + s\pi'\big)\big((1-s)P + sP'\big) \tag{2.22}$$

$$= (1-s)^2 \operatorname{diag}(\pi)P + (1-s)s\big(\operatorname{diag}(\pi)P' + \operatorname{diag}(\pi')P\big) + s^2 \operatorname{diag}(\pi')P' \tag{2.23}$$

where the first and last terms are symmetric since P and P' are reversible, but the middle term is symmetric due to Eq. (2.21). □

2.3 Discriminant matrix

2.3.1 Definition

The *discriminant matrix* of a Markov chain $P(s)$ is

$$D(s) := \sqrt{P(s) \circ P(s)^{\mathsf{T}}}, \tag{2.24}$$

where the Hadamard product "$\circ$" and the square root are computed entry-wise. This matrix was introduced by Szegedy in [Sze04a, Sze04b]. We prefer to work with $D(s)$ rather than $P(s)$ since the matrix of transition probabilities is not necessarily symmetric while its discriminant matrix is.

Proposition 2.9. *If P is reversible then*

$$D(s) = \operatorname{diag}\big(\sqrt{\pi(s)}\,\big)\, P(s)\, \operatorname{diag}\big(\sqrt{\pi(s)}\,\big)^{-1}, \qquad \forall s \in [0,1); \tag{2.25}$$

$$D(1) = \begin{pmatrix}\operatorname{diag}\big(\sqrt{\pi_U}\,\big)\, P_{UU}\, \operatorname{diag}\big(\sqrt{\pi_U}\,\big)^{-1} & 0\\ 0 & I\end{pmatrix}. \tag{2.26}$$

Here the square roots are also computed entry-wise and M^{-1} denotes the matrix inverse of M. Notice that for $s \in [0,1)$ the right-hand side of Eq. (2.25) is well-defined, since $P(s)$ is ergodic by Prop. 2.2 and thus according to the Perron–Frobenius Theorem has a unique and non-vanishing stationary distribution. However, recall from Prop. 2.6 that $\pi(1)$ vanishes on U, so the right-hand side of Eq. (2.25) is no longer well-defined at $s = 1$. For this reason we have an alternative expression for $D(1)$.

Proof (of Prop. 2.9). For a reversible Markov chain P the extended detailed balance condition in Eq. (2.18) implies that $D_{xy}(s) = \sqrt{P_{xy}(s)P_{yx}(s)} = P_{xy}(s)\sqrt{\pi_x(s)/\pi_y(s)}$. This is equivalent to Eq. (2.25).

At $s = 1$ from Eq. (2.24) we have:

$$D(1) = \sqrt{P(1) \circ P(1)^{\mathsf{T}}} = \sqrt{\begin{pmatrix} P_{UU} \circ P_{UU}^{\mathsf{T}} & 0 \\ 0 & I \end{pmatrix}} = \begin{pmatrix} \sqrt{P_{UU} \circ P_{UU}^{\mathsf{T}}} & 0 \\ 0 & I \end{pmatrix}. \tag{2.27}$$

In the same way one can use Eq. (2.24) to compute the upper left block of $D(s)$ for any s, and notice that it does not depend on s:

$$D_{UU}(s) = \sqrt{P_{UU} \circ P_{UU}^{\mathsf{T}}} = D_{UU}(0) = \operatorname{diag}\big(\sqrt{\pi_U}\,\big)\, P_{UU}\, \operatorname{diag}\big(\sqrt{\pi_U}\,\big)^{-1} \tag{2.28}$$

where the last equality follows from Eq. (2.25) at $s = 0$. Together with Eq. (2.27) this gives us the desired expression in Eq. (2.26). □

2.3.2 Spectral decomposition

Recall from Eq. (2.24) that $D(s)$ is real and symmetric. Therefore, its eigenvalues are real and it has an orthonormal set of real eigenvectors. Let

$$D(s) = \sum_{i=1}^{n} \lambda_i(s) |v_i(s)\rangle\langle v_i(s)| \tag{2.29}$$

be the spectral decomposition of $D(s)$ with eigenvalues $\lambda_i(s)$ and eigenvectors[3] $|v_i(s)\rangle$. Moreover, let us arrange the eigenvalues so that

$$\lambda_1(s) \leq \lambda_2(s) \leq \cdots \leq \lambda_n(s). \tag{2.30}$$

[3]There is no need to use bra-ket notation at this point; nevertheless we adopt it since vectors $|v_i(s)\rangle$ later will be used as quantum states.

From now on we will assume that P is reversible (and hence ergodic) without explicitly mentioning it. Under this assumption the matrices $P(s)$ and $D(s)$ are similar (see Prop. 2.10 below). This means that $D(s)$ essentially has the same properties as $P(s)$, but in addition it also admits a spectral decomposition with orthogonal eigenvectors. This will be very useful in Chapter 3, where we will find the spectral decomposition of Hamiltonian $H(s)$ in terms of that of $D(s)$, and use it to relate properties of $H(s)$ and $P(s)$.

Proposition 2.10. *The matrices $P(s)$ and $D(s)$ are similar for any $s \in [0,1]$ and therefore have the same eigenvalues. In particular, the eigenvalues of $P(s)$ are real.*

Proof. From Eq. (2.25) we see that the matrices $D(s)$ and $P(s)$ are similar for $s \in [0,1)$. From Eq. (2.26) we see that $D(1)$ is similar to $\tilde{P} := \left(\begin{smallmatrix} P_{UU} & 0 \\ 0 & I \end{smallmatrix}\right)$. To verify that $\tilde{P}$ and $P(1) = \left(\begin{smallmatrix} P_{UU} & P_{UM} \\ 0 & I \end{smallmatrix}\right)$ are similar, let $M := \left(\begin{smallmatrix} P_{UU}-I & P_{UM} \\ 0 & I \end{smallmatrix}\right)$. One can check that $MP(1)M^{-1} = \tilde{P}$ where $M^{-1} = \left(\begin{smallmatrix} (P_{UU}-I)^{-1} & -(P_{UU}-I)^{-1}P_{UM} \\ 0 & I \end{smallmatrix}\right)$ exists, since $P_{UU} - I$ is invertible according to Prop. 2.5. By transitivity, $D(1)$ is also similar to $P(1)$. □

Proposition 2.11. *The largest eigenvalue of $D(s)$ is 1. It has multiplicity 1 when $s \in [0,1)$ and multiplicity m when $s = 1$. In other words,*

$$\lambda_{n-1}(s) < \lambda_n(s) = 1, \quad \forall s \in [0,1), \tag{2.31}$$

$$\lambda_{n-m}(1) < \lambda_{n-m+1}(1) = \cdots = \lambda_n(1) = 1. \tag{2.32}$$

Proof. Let us argue about $P(s)$, since it has the same eigenvalues as $D(s)$ by Prop. 2.10. From the Perron–Frobenius Theorem we have that $\forall i : \lambda_i(s) \leq 1$ and $\lambda_n(s) = 1$. In addition, by Prop. 2.2 the Markov chain $P(s)$ is ergodic for any $s \in [0,1)$, so $\forall i \neq n : \lambda_i(s) < 1$. Finally, note by Eq. (2.26) that for $s = 1$ eigenvalue 1 has multiplicity at least m. Recall from Eq. (2.28) that $D_{UU}(1)$ and P_{UU} are similar. From Prop. 2.5 we conclude that all eigenvalues of P_{UU} are strictly less than 1. Thus the multiplicity of eigenvalue 1 of $D(1)$ is exactly m. □

2.3.3 Principal eigenvector

Let us prove an analogue of Prop. 2.6 for the matrix $D(s)$.

Proposition 2.12. *$\sqrt{\pi(s)}^{\mathsf{T}}$ is the unique $(+1)$-eigenvector of $D(s)$ for $s \in [0,1)$. At $s = 1$ any vector with support only on marked states is a $(+1)$-eigenvector, including $\sqrt{\pi(1)}^{\mathsf{T}}$.*

Proof. Since $P(s)$ is row-stochastic, $P(s)\,1_X^\mathsf{T} = 1_X^\mathsf{T}$ where 1_X is the all-ones row vector. Thus we can check that for $s \in [0,1)$,

$$D(s)\sqrt{\pi(s)^\mathsf{T}} = \operatorname{diag}\Big(\sqrt{\pi(s)}\Big)\,P(s)\,\operatorname{diag}\Big(\sqrt{\pi(s)}\Big)^{-1}\sqrt{\pi(s)^\mathsf{T}} \tag{2.33}$$

$$= \operatorname{diag}\Big(\sqrt{\pi(s)}\Big)\,P(s)\,1_X^\mathsf{T} \tag{2.34}$$

$$= \operatorname{diag}\Big(\sqrt{\pi(s)}\Big)\,1_X^\mathsf{T} \tag{2.35}$$

$$= \sqrt{\pi(s)^\mathsf{T}}. \tag{2.36}$$

Uniqueness for $s \in [0,1)$ follows by the uniqueness of $\pi(s)$ and Prop. 2.10. For the $s = 1$ case, notice from Eq. (2.26) that $D(1)$ acts trivially on marked elements and recall from Eq. (2.8) that $\pi(1) = (0_U \;\; \pi_M)/p_M$. □

According to the above Proposition, for any $s \in [0,1]$ we can choose the principal eigenvector $|v_n(s)\rangle$ in the spectral decomposition of $D(s)$ in Eq. (2.29) to be

$$|v_n(s)\rangle := \sqrt{\pi(s)^\mathsf{T}}. \tag{2.37}$$

We would like to have an intuitive understanding of how $|v_n(s)\rangle$ evolves as a function of s. Let us introduce some useful notation that we will also need later.

Let 0_U and 1_U (respectively, 0_M and 1_M) be the all-zeros and all-ones row vectors of dimension $n - m$ (respectively, m) whose entries are indexed by elements of U (respectively, M). Furthermore, let

$$\tilde{\pi}_U := \pi_U/(1 - p_M), \qquad \tilde{\pi}_M := \pi_M/p_M \tag{2.38}$$

be the normalized row vectors describing the stationary distribution π restricted to unmarked and marked states. Let us also define the following unit vectors in $\mathbb{R}^n$:

$$|U\rangle := \sqrt{(\tilde{\pi}_U \;\; 0_M)^\mathsf{T}} = \frac{1}{\sqrt{1 - p_M}} \sum_{x \in U} \sqrt{\pi_x}|x\rangle, \tag{2.39}$$

$$|M\rangle := \sqrt{(0_U \;\; \tilde{\pi}_M)^\mathsf{T}} = \frac{1}{\sqrt{p_M}} \sum_{x \in M} \sqrt{\pi_x}|x\rangle. \tag{2.40}$$

Then we can express $|v_n(s)\rangle$ as a linear combination of $|U\rangle$ and $|M\rangle$.

Proposition 2.13. $|v_n(s)\rangle = \cos\theta(s)|U\rangle + \sin\theta(s)|M\rangle$ *where*

$$\cos\theta(s) = \sqrt{\frac{(1-s)(1-p_M)}{1-s(1-p_M)}}, \qquad \sin\theta(s) = \sqrt{\frac{p_M}{1-s(1-p_M)}}. \tag{2.41}$$

Proof. By substituting $\pi(s)$ from Eq. (2.8) into Eq. (2.37) we get

$$|v_n(s)\rangle = \sqrt{\pi(s)^\mathsf{T}} = \sqrt{\frac{\big((1-s)\pi_U \;\; \pi_M\big)^\mathsf{T}}{1-s(1-p_M)}} = \sqrt{\frac{\big((1-s)(1-p_M)\tilde{\pi}_U \;\; p_M\tilde{\pi}_M\big)^\mathsf{T}}{1-s(1-p_M)}} \tag{2.42}$$

which is the desired expression. □

Thus $|v_n(s)\rangle$ lies in the two-dimensional subspace $\text{span}\{|U\rangle, |M\rangle\}$ and is subject to a rotation as we change the parameter s (see Fig. 2.4). In particular,

$$|v_n(0)\rangle = \sqrt{1-p_M}|U\rangle + \sqrt{p_M}|M\rangle, \qquad |v_n(1)\rangle = |M\rangle. \tag{2.43}$$

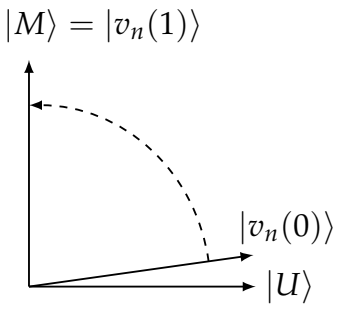

Figure 2.4: As s changes from zero to one, the evolution of the principal eigenvector $|v_n(s)\rangle$ corresponds to a rotation in the two-dimensional subspace $\text{span}\{|U\rangle, |M\rangle\}$.

Proposition 2.14. $\theta(s)$ *and its derivative* $\dot{\theta}(s) := \frac{d}{ds}\theta(s)$ *are related as follows:*

$$2\dot{\theta}(s) = \frac{\sin\theta(s)\cos\theta(s)}{1-s}. \tag{2.44}$$

Proof. Notice that

$$\frac{d}{ds}\big(\sin^2\theta(s)\big) = 2\dot{\theta}(s)\sin\theta(s)\cos\theta(s). \tag{2.45}$$

On the other hand, according to Eq. (2.41) we have

$$\frac{d}{ds}\big(\sin^2\theta(s)\big) = \frac{d}{ds}\left(\frac{p_M}{1-s(1-p_M)}\right) = \frac{p_M(1-p_M)}{(1-s(1-p_M))^2} = \frac{\sin^2\theta(s)\cos^2\theta(s)}{1-s}. \quad (2.46)$$

By comparing both equations we get the desired result. □

2.3.4 Derivative

Proposition 2.15. *$D(s)$ and its derivative $\dot{D}(s) := \frac{d}{ds}D(s)$ are related as follows:*

$$\dot{D}(s) = \frac{1}{2(1-s)}\big\{\Pi_M, I - D(s)\big\} \quad (2.47)$$

where $\{X, Y\} := XY + YX$ is the anticommutator of X and Y, and $\Pi_M := \sum_{x\in M} |x\rangle\langle x|$ is the projector onto the m-dimensional subspace spanned by marked states M.

Proof. Recall from Eq. (2.24) that $D(s) = \sqrt{P(s)\circ P(s)^\mathsf{T}}$. The block structure of $P(s)$ is given in Eq. (2.4). First, let us derive an expression for $D_{MM}(s)$, the lower right block of $D(s)$:

$$D_{MM}(s) = \sqrt{P_{MM}(s)\circ P_{MM}(s)^\mathsf{T}} \quad (2.48)$$

$$= \sqrt{\big((1-s)P_{MM} + sI\big)\circ\big((1-s)P_{MM}^\mathsf{T} + sI\big)}. \quad (2.49)$$

Let us separately consider the diagonal and off-diagonal entries of $D_{MM}(s)$. For $x, y \in M$ we have

$$D_{xy}(s) = \begin{cases} (1-s)\sqrt{P_{xy}P_{yx}} & \text{if } x \neq y, \\ (1-s)P_{xx} + s & \text{if } x = y. \end{cases} \quad (2.50)$$

Thus we can write $D_{MM}(s)$ as

$$D_{MM}(s) = (1-s)\sqrt{P_{MM}\circ P_{MM}^\mathsf{T}} + sI. \quad (2.51)$$

Expressions for the remaining blocks of $D(s)$ can be derived in a straightforward way. By putting all blocks together we get

$$D(s) = \begin{pmatrix} \sqrt{P_{UU}\circ P_{UU}^\mathsf{T}} & \sqrt{(1-s)(P_{UM}\circ P_{MU}^\mathsf{T})} \\ \sqrt{(1-s)(P_{MU}\circ P_{UM}^\mathsf{T})} & (1-s)\sqrt{P_{MM}\circ P_{MM}^\mathsf{T}} + sI \end{pmatrix}. \quad (2.52)$$

When we take the derivative with respect to s we find

$$\dot{D}(s) = \begin{pmatrix} 0 & -\frac{1}{2\sqrt{1-s}}\sqrt{P_{UM} \circ P_{MU}^{\mathsf{T}}} \\ -\frac{1}{2\sqrt{1-s}}\sqrt{P_{MU} \circ P_{UM}^{\mathsf{T}}} & I - \sqrt{P_{MM} \circ P_{MM}^{\mathsf{T}}} \end{pmatrix}. \tag{2.53}$$

To relate $\dot{D}(s)$ and the original matrix $D(s)$, observe that

$$\Pi_M D(s) + D(s)\Pi_M = \begin{pmatrix} 0 & \sqrt{(1-s)(P_{UM} \circ P_{MU}^{\mathsf{T}})} \\ \sqrt{(1-s)(P_{MU} \circ P_{UM}^{\mathsf{T}})} & 2(1-s)\sqrt{P_{MM} \circ P_{MM}^{\mathsf{T}}} + 2sI \end{pmatrix} \tag{2.54}$$

which can be seen by overlaying the second column and row of $D(s)$ given in Eq. (2.52). When we rescale this by an appropriate constant, we get

$$-\frac{1}{2(1-s)}\{\Pi_M, D(s)\} = \begin{pmatrix} 0 & -\frac{1}{2\sqrt{1-s}}\sqrt{P_{UM} \circ P_{MU}^{\mathsf{T}}} \\ -\frac{1}{2\sqrt{1-s}}\sqrt{P_{MU} \circ P_{UM}^{\mathsf{T}}} & -\sqrt{P_{MM} \circ P_{MM}^{\mathsf{T}}} - \frac{s}{1-s}I \end{pmatrix}. \tag{2.55}$$

This is very similar to the expression for $\dot{D}(s)$ in Eq. (2.53), except for a slightly different coefficient for the identity matrix in the lower right corner. We can correct this by adding Π_M with an appropriate constant: $-\frac{1}{2(1-s)}\{\Pi_M, D(s)\} + \frac{1}{1-s}\Pi_M = \dot{D}(s)$. □

2.4 Hitting time

2.4.1 Definition

To define the hitting time of Markov chain P, let us consider a simple classical algorithm for finding a marked element in the state space X using a random walk based on P.

Random Walk Algorithm

1. Sample a vertex $x \in X$ according to the stationary distribution π of P.
2. If x is marked, output x and exit.
3. Otherwise, update x according to P and go back to step 2.

The hitting time of P is the expected number of applications of P during this algorithm (notice that the algorithm stops as soon as a marked element is reached, thus effectively it uses the absorbing Markov chain P'). Here is a more formal definition:

Definition 2.16. Let P be an ergodic Markov chain, and M be a set of marked states. The *hitting time* of P with respect to M, denoted by $\mathrm{HT}(P, M)$, is the expected number of executions of the last step of the **Random Walk Algorithm**, conditioned on the initial vertex being unmarked.

Proposition 2.17. *The hitting time of Markov chain P with respect to marked set M is*

$$\mathrm{HT}(P, M) = \sum_{t=0}^{\infty} \langle U | D^t(1) | U \rangle. \tag{2.56}$$

Proof. The expected number of iterations in the **Random Walk Algorithm** is

$$\begin{aligned}
\mathrm{HT}(P, M) &:= \sum_{l=1}^{\infty} l \cdot \Pr[\text{need } \textit{exactly } l \text{ steps}] && (2.57)\\
&= \sum_{l=1}^{\infty} \sum_{t=1}^{l} \Pr[\text{need } \textit{exactly } l \text{ steps}] && (2.58)\\
&= \sum_{t=1}^{\infty} \sum_{l=t}^{\infty} \Pr[\text{need } \textit{exactly } l \text{ steps}] && (2.59)\\
&= \sum_{t=1}^{\infty} \Pr[\text{need } \textit{at least } t \text{ steps}] && (2.60)\\
&= \sum_{t=0}^{\infty} \Pr[\text{need } \textit{more} \text{ than } t \text{ steps}] && (2.61)
\end{aligned}$$

where the region corresponding to the double sum is shown in Fig. 2.5.

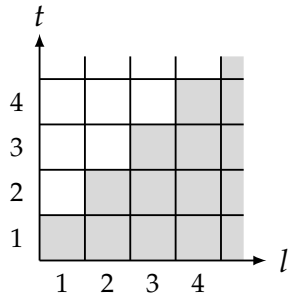

Figure 2.5: The region corresponding to the double sum in Eqs. (2.58) and (2.59).

Hence, we can consider the probability that no marked vertex is found after t steps starting from an unmarked vertex distributed according to $\tilde{\pi}_U = \pi_U/(1-p_M)$. Let us find an explicit expression for this probability. The distribution of vertices at the first execution of step 3 of the **Random Walk Algorithm** is $(\tilde{\pi}_U \;\; 0_M)$, hence

$$\Pr[\text{need } \textit{more} \text{ than } t \text{ steps}] = (\tilde{\pi}_U \;\; 0_M) P'^t (1_U \;\; 0_M)^\mathsf{T}. \tag{2.62}$$

Recall from Prop. 2.3 that $(P'^t)_{UU} = P^t_{UU}$ so we can simplify Eq. (2.62) as follows:

$$\begin{aligned}
\Pr[\text{need } \textit{more} \text{ than } t \text{ steps}] &= (\tilde{\pi}_U \;\; 0_M) P'^t (1_U \;\; 0_M)^\mathsf{T} && (2.63)\\
&= \frac{\pi_U}{1-p_M} P^t_{UU} 1_U^\mathsf{T} && (2.64)\\
&= \sqrt{\tfrac{\pi_U}{1-p_M}}\, \mathrm{diag}\big(\sqrt{\pi_U}\big) P^t_{UU}\, \mathrm{diag}\big(\sqrt{\pi_U}\big)^{-1} \sqrt{\tfrac{\pi_U^\mathsf{T}}{1-p_M}} && (2.65)\\
&= \langle U | D^t(1) | U \rangle, && (2.66)
\end{aligned}$$

where the last equality follows from the expression for the discriminant matrix $D(1)$ in Eq. (2.26). By plugging this back in Eq. (2.61) we get the desired result. □

2.4.2 Extended hitting time

Let us define the following extension of the hitting time based on Eq. (2.56):

$$\mathrm{HT}(s) := \sum_{t=0}^{\infty} \langle U | \big(D^t(s) - |v_n(s)\rangle\langle v_n(s)| \big) | U \rangle. \tag{2.67}$$

Note that $\mathrm{HT}(1) = \mathrm{HT}(P, M)$ since $\langle U | v_n(1) \rangle = \langle U | M \rangle = 0$. This justifies calling $\mathrm{HT}(s)$ *extended hitting time*. We can use the similarity transformation between $D(s)$ and $P(s)$ from Prop. 2.9 to obtain an alternative expression for $\mathrm{HT}(s)$:

$$\mathrm{HT}(s) = \sum_{t=0}^{\infty} (\tilde{\pi}_U \;\; 0_M) \big(P^t(s) - Q(s) \big) (1_U \;\; 0_M)^\mathsf{T} \tag{2.68}$$

where $Q(s) := \lim_{t\to\infty} P^t(s)$ is a stochastic matrix whose all rows are equal to $\pi(s)$. Intuitively, $\mathrm{HT}(s)$ may be understood as the time it takes for $P(s)$ to converge to its stationary distribution $\pi(s)$, starting from $(\tilde{\pi}_U \;\; 0_M)$. For $s = 1$, the walk $P(1) = P'$ converges to the (non-unique) stationary distribution $(0_U \;\; \tilde{\pi}_M)$, which only has support over marked elements.

Proposition 2.18. *The extended hitting time can be expressed as*

$$\mathrm{HT}(s) = \langle U|A(s)|U\rangle, \qquad A(s) := \sum_{k:\,\lambda_k(s)\neq 1} \frac{|v_k(s)\rangle\langle v_k(s)|}{1-\lambda_k(s)}. \tag{2.69}$$

Proof. Rewrite Eq. (2.67) using the spectral decomposition of $D(s)$ from Eq. (2.29):

$$\mathrm{HT}(s) = \sum_{t=0}^{\infty}\sum_{k\neq n} \lambda_k^t(s)\langle U|v_k(s)\rangle\langle v_k(s)|U\rangle = \sum_{k:\,\lambda_k(s)\neq 1} \frac{|\langle v_k(s)|U\rangle|^2}{1-\lambda_k(s)} \tag{2.70}$$

where we exchanged the sums and used the expansion $(1-x)^{-1} = \sum_{t=0}^{\infty} x^t$. □

For technical reasons it will be important later that all eigenvalues of $P(s)$ are non-negative. We can guarantee this using a standard trick—we replace the original Markov chain P with the "lazy" walk $(P+I)/2$ where I is the $n \times n$ identity matrix. In fact, we can assume without loss of generality that the original Markov chain already is "lazy", since this affects the hitting time only by a constant factor, as shown below.

Proposition 2.19. *Let P be an ergodic and reversible Markov chain. Then for any $s \in [0,1]$ the eigenvalues of $(P(s)+I)/2$ are between 0 and 1. Moreover, if the extended hitting time of P is $\mathrm{HT}(s)$, then the extended hitting time of $(P+I)/2$ is $2\,\mathrm{HT}(s)$.*

Proof. Since P is reversible, so is $P(s)$ by Prop. 2.8. Thus the eigenvalues of $P(s)$ are real by Prop. 2.10. If $\lambda_k(s)$ is an eigenvalue of $P(s)$ then $\lambda_k(s) \in [-1,1]$ according to Perron–Frobenius Theorem. Thus, the eigenvalues of $(P(s)+I)/2$ satisfy $(\lambda_k(s)+1)/2 \in [0,1]$.

Recall from Prop. 2.10 that $P(s)$ and $D(s)$ are similar. Thus, the discriminant matrix of $(P(s)+I)/2$ is $(D(s)+I)/2$, which has the same eigenvectors as $D(s)$. By Prop. 2.18 we see that the extended hitting time of $(P(s)+I)/2$ is given by

$$\sum_{k:\,\lambda_k(s)\neq 1} \frac{|\langle v_k(s)|U\rangle|^2}{1-\frac{\lambda_k(s)+1}{2}}. \tag{2.71}$$

Since $1 - \frac{\lambda_k(s)+1}{2} = \frac{1-\lambda_k(s)}{2}$, the above expression is equal to $2\,\mathrm{HT}(s)$ as claimed. □

The following property of $A(s)$ will be useful on several occasions.

Proposition 2.20. $A(s)|M\rangle = -\frac{\cos\theta(s)}{\sin\theta(s)} A(s)|U\rangle$.

Proof. Recall from Prop. 2.11 that $\lambda_n(s) = 1$, so $A(s)|v_n(s)\rangle = 0$ by definition. If we substitute $|v_n(s)\rangle = \cos\theta(s)|U\rangle + \sin\theta(s)|M\rangle$ from Prop. 2.13 in this equation, we get the desired formula. □

2.4.3 Dependence on s

The goal of this section is to express $\mathrm{HT}(s)$ as a function of s and the hitting time $\mathrm{HT}(P, M)$ of the original Markov chain. The main idea is to relate $\frac{d}{ds}\mathrm{HT}(s)$ to $\mathrm{HT}(s)$ and then solve the resulting differential equation.

Lemma 2.21. *The derivative of* $\mathrm{HT}(s)$ *is related to* $\mathrm{HT}(s)$ *as*

$$\frac{d}{ds}\mathrm{HT}(s) = \frac{2(1-p_M)}{1-s(1-p_M)}\mathrm{HT}(s) \tag{2.72}$$

where p_M *is the probability to pick a marked state from the stationary distribution* π *of* P*.*

Proof. Recall from Prop. 2.18 that $\mathrm{HT}(s) = \langle U|A(s)|U\rangle$ where $A(s)$ may be written as

$$A(s) = B(s)^{-1} - \Pi_n(s) \text{ where } B(s) := I - D(s) + \Pi_n(s),\ \Pi_n(s) := |v_n(s)\rangle\langle v_n(s)|. \tag{2.73}$$

Recall from Sect. 2.3.2 that $|v_n(s)\rangle$ is the unique $(+1)$-eigenvector of $D(s)$ for $s \in [0,1)$, thus $B(s)$ is indeed invertible when s is in this range.

From now on we will not write the dependence on s explicitly. We will also often use $\dot{f}(s)$ as a shorthand form of $\frac{d}{ds}f(s)$. Let us start with

$$\frac{d}{ds}\mathrm{HT} = \langle U|\dot{A}|U\rangle \tag{2.74}$$

and expand $\dot{A}$ using Eq. (2.73). To find $\frac{d}{ds}(B^{-1})$, take the derivative of both sides of $B^{-1}B = I$ and get $\frac{d}{ds}(B^{-1})\cdot B + B^{-1}\cdot\frac{d}{ds}B = 0$. Thus $\frac{d}{ds}(B^{-1}) = -B^{-1}\dot{B}B^{-1}$ and

$$\dot{A} = -B^{-1}\dot{B}B^{-1} - \dot{\Pi}_n. \tag{2.75}$$

Notice from Eq. (2.73) that $\dot{B} = -\dot{D} + \dot{\Pi}_n$, thus $\dot{A} = -B^{-1}(-\dot{D} + \dot{\Pi}_n)B^{-1} - \dot{\Pi}_n$ and $\frac{d}{ds}\mathrm{HT} = h_1 + h_2 + h_3$ where

$$h_1 := \langle U|B^{-1}\dot{D}B^{-1}|U\rangle, \tag{2.76}$$

$$h_2 := -\langle U|B^{-1}\dot{\Pi}_n B^{-1}|U\rangle, \tag{2.77}$$

$$h_3 := -\langle U|\dot{\Pi}_n|U\rangle. \tag{2.78}$$

Let us evaluate each of these terms separately.

To evaluate the first term h_1, we substitute $\dot{D} = \frac{1}{2(1-s)}\{\Pi_M, I - D\}$ from Prop. 2.15 and replace $I - D$ by $B - \Pi_n$ according to Eq. (2.73):

$$2(1-s)h_1 = \langle U|B^{-1}\{\Pi_M, B - \Pi_n\}B^{-1}|U\rangle \tag{2.79}$$
$$= \langle U|B^{-1}\big(\{\Pi_M, B\} - \{\Pi_M, \Pi_n\}\big)B^{-1}|U\rangle \tag{2.80}$$
$$= \langle U|\{B^{-1}, \Pi_M\}|U\rangle - \langle U|B^{-1}\{\Pi_M, \Pi_n\}B^{-1}|U\rangle. \tag{2.81}$$

Recall that $\Pi_M = \sum_{x\in M}|x\rangle\langle x|$ is the projector onto the marked states. Thus $\Pi_M|U\rangle = 0$ and the first term vanishes. Note that B has the same eigenvectors as D. In particular, $B^{-1}|v_n\rangle = |v_n\rangle$ and thus $B^{-1}\Pi_n = \Pi_n = \Pi_n B^{-1}$. Using this we can expand the anti-commutator in the second term: $B^{-1}\{\Pi_M, \Pi_n\}B^{-1} = B^{-1}\Pi_M\Pi_n + \Pi_n\Pi_M B^{-1}$. Since all three matrices in this expression are real and symmetric and $|U\rangle$ is also real, both terms of the anti-commutator have the same contribution, so we get

$$2(1-s)h_1 = -2\langle U|B^{-1}\Pi_M\Pi_n|U\rangle. \tag{2.82}$$

Recall from Prop. 2.13 that $|v_n\rangle = \cos\theta|U\rangle + \sin\theta|M\rangle$, so we see that $\Pi_M\Pi_n|U\rangle = \Pi_M|v_n\rangle\cdot\langle v_n|U\rangle = \sin\theta|M\rangle\cdot\cos\theta$. Moreover, $B^{-1} = A + \Pi_n$ according to Eq. (2.73), so

$$2(1-s)h_1 = -2\sin\theta\cos\theta\langle U|(A + \Pi_n)|M\rangle. \tag{2.83}$$

Recall from Prop. 2.20 that $\sin\theta\langle U|A|M\rangle = \cos\theta\langle U|A|U\rangle$. To simplify the second term, notice that $\langle U|\Pi_n|M\rangle = \langle U|v_n\rangle\cdot\langle v_n|M\rangle = \cos\theta\cdot\sin\theta$. When we put this together, we get

$$2(1-s)h_1 = 2\cos^2\theta\langle U|A|U\rangle - 2\sin^2\theta\cos^2\theta \tag{2.84}$$

or simply

$$h_1 = \frac{\cos^2\theta}{1-s}\big(\langle U|A|U\rangle - \sin^2\theta\big). \tag{2.85}$$

Let us now consider the second term $h_2 = -\langle U|B^{-1}\dot{\Pi}_n B^{-1}|U\rangle$. First, we compute $\dot{\Pi}_n = |\dot{v}_n\rangle\langle v_n| + |v_n\rangle\langle\dot{v}_n|$. Using $B^{-1}|v_n\rangle = |v_n\rangle$ we get $B^{-1}\dot{\Pi}_n B^{-1} = B^{-1}|\dot{v}_n\rangle\langle v_n| + |v_n\rangle\langle\dot{v}_n|B^{-1}$. Since $\langle v_n|U\rangle = \cos\theta$ we have

$$h_2 = -2\langle U|B^{-1}|\dot{v}_n\rangle\cos\theta \tag{2.86}$$

where the factor two comes from the fact that all vectors involved are real and matrix B^{-1} is real and symmetric. Let us compute

$$|\dot{v}_n\rangle = \dot{\theta}\big(-\sin\theta|U\rangle + \cos\theta|M\rangle\big). \tag{2.87}$$

Notice that $\langle v_n|\dot{v}_n\rangle = 0$ and thus $\Pi_n|\dot{v}_n\rangle = 0$. By substituting $B^{-1} = A + \Pi_n$ from Eq. (2.73) we get

$$h_2 = -2\langle U|A|\dot{v}_n\rangle \cos\theta. \tag{2.88}$$

Next, we substitute $|\dot{v}_n\rangle$ and get

$$h_2 = -2\dot{\theta}\big(-\sin\theta\langle U|A|U\rangle + \cos\theta\langle U|A|M\rangle\big)\cos\theta. \tag{2.89}$$

Now we use Prop. 2.20 to substitute $A|M\rangle$ by $A|U\rangle$:

$$h_2 = -2\dot{\theta}\left(-\sin\theta - \frac{\cos^2\theta}{\sin\theta}\right)\langle U|A|U\rangle\cos\theta = 2\dot{\theta}\frac{\cos\theta}{\sin\theta}\langle U|A|U\rangle. \tag{2.90}$$

Finally, we substitute $2\dot{\theta} = \frac{\sin\theta\cos\theta}{1-s}$ from Eq. (2.44) and get

$$h_2 = \frac{\cos^2\theta}{1-s}\langle U|A|U\rangle. \tag{2.91}$$

For the last term $h_3 = -\langle U|\dot{\Pi}\Pi_n|U\rangle$ we observe that $\langle U|\dot{v}_n\rangle\langle v_n|U\rangle = -\dot{\theta}\sin\theta\cdot\cos\theta$ thus $h_3 = 2\dot{\theta}\sin\theta\cos\theta$ where the factor two comes from symmetry. After substituting $2\dot{\theta}$ from Eq. (2.44) we get

$$h_3 = \frac{\cos^2\theta}{1-s}\sin^2\theta. \tag{2.92}$$

When we compare Eqs. (2.85), (2.91), and (2.92) we notice that $h_2 = h_1 + h_3$. Thus the derivative of the hitting time is $\frac{d}{ds}\,\mathrm{HT} = h_1 + h_2 + h_3 = 2h_2$. Recall from Eq. (2.69) that $\mathrm{HT} = \langle U|A|U\rangle$. Thus

$$\frac{d}{ds}\,\mathrm{HT}(s) = 2\frac{\cos^2\theta(s)}{1-s}\,\mathrm{HT}(s). \tag{2.93}$$

By substituting $\cos\theta(s)$ from Eq. (2.41) we get the desired result. □

Theorem 2.22. *The extended hitting time* $\mathrm{HT}(s)$ *is related to* $\mathrm{HT}(P, M)$*, the hitting time of Markov chain* P *with marked states* M*, as follows:*

$$\mathrm{HT}(s) = \frac{p_M^2}{(1-s(1-p_M))^2}\,\mathrm{HT}(P, M) \tag{2.94}$$

where p_M *is the probability to pick a marked state from the stationary distribution* π *of* P*.*

Proof. We will prove this theorem by solving the differential equation from Lemma 2.21. In particular, let us consider Eq. (2.93). Recall from Eq. (2.44) that $2\dot{\theta} = \frac{\sin\theta\cos\theta}{1-s}$, so we can rewrite the coefficient in this equation as

$$2\frac{\cos^2\theta}{1-s} = 2 \cdot \frac{\sin\theta\cos\theta}{1-s} \cdot \frac{\cos\theta}{\sin\theta} = 4\dot{\theta}\frac{\cos\theta}{\sin\theta} = 4\frac{\frac{d}{ds}(\sin\theta)}{\sin\theta}. \tag{2.95}$$

Now we can rewrite the differential equation as

$$\frac{\frac{d}{ds}\mathrm{HT}(s)}{\mathrm{HT}(s)} = 4\frac{\frac{d}{ds}(\sin\theta(s))}{\sin\theta(s)}. \tag{2.96}$$

By integrating both sides we get

$$\ln|\mathrm{HT}(s)| = 4\ln|\sin\theta(s)| + C \tag{2.97}$$

for some constant C. Recall from Sect. 2.4.2 that $\mathrm{HT}(1) = \mathrm{HT}(P, M)$ and from Eq. (2.41) that $\sin\theta(1) = 1$, so the boundary condition at $s = 1$ gives us $C = \ln|\mathrm{HT}(P, M)|$. Since all quantities are non-negative, we can omit the absolute value signs. After exponentiating both sides we get

$$\mathrm{HT}(s) = \sin^4\theta(s) \cdot \mathrm{HT}(P, M). \tag{2.98}$$

We get the desired expression when we substitute $\sin\theta(s)$ from Eq. (2.41). □

In the next two chapters we consider several quantum search algorithms whose running time depends on $\mathrm{HT}(s)$ for some values of $s \in [0, 1]$. Theorem 2.22 is a crucial ingredient in analysis of these algorithms, since it relates $\mathrm{HT}(s)$ to the usual hitting time $\mathrm{HT}(P, M)$. In particular, it is important that $\mathrm{HT}(s)$ is monotonically increasing as a function of s (some example plots of $\mathrm{HT}(s)$ are shown in Fig. 2.6).

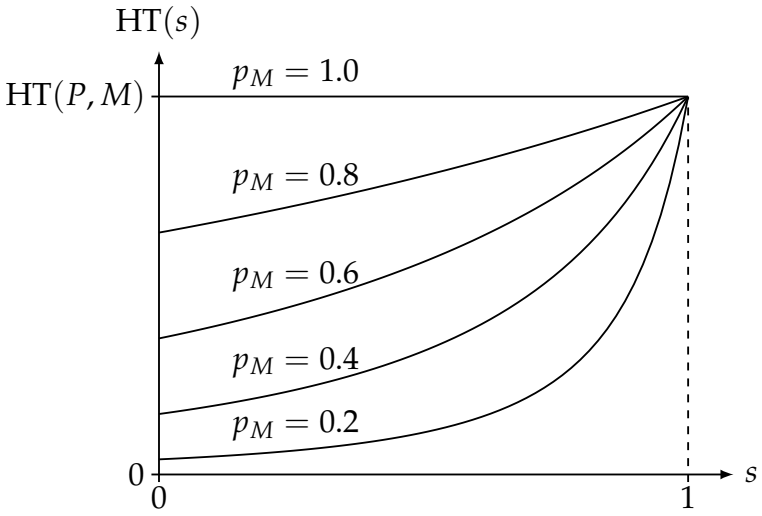

Figure 2.6: Extended hitting time $\mathrm{HT}(s)$ as a function of s for several values of p_M.

Chapter 3

Adiabatic condition and the quantum hitting time of Markov chains

Contents

3.1 Introduction

In Chapter 2, we studied semi-absorbing Markov chains which are of the form $P(s) = sP + (1-s)P'$ where P' is the absorbing version of P. This is very reminiscent of adiabatic quantum computing, which is the main topic of this chapter.

Adiabatic quantum computation is an approach for solving problems on a quantum computer by performing a continuous-time evolution according to some time-dependent Hamiltonian. It was introduced in 2000 by Farhi, Goldstone, Gutmann, and Sipser [FGGS00]. Let us recall the basic idea. Suppose that the solution of a computational problem can be encoded in the ground state of a *problem Hamiltonian* H_P. To find this solution, we start in the ground state of some other Hamiltonian H_0, known as the *initial Hamiltonian*. Usually, H_0 is much simpler than H_P and does not depend on the solution to the problem, so its ground state is easy to construct. Then we slowly change the Hamiltonian from H_0 to H_P, say, by taking a convex combination of the two Hamiltonians: $H(s) = (1-s)H_0 + sH_P$ where $0 \leq s \leq 1$. If this is done slowly enough, the Adiabatic Theorem of quantum mechanics [Mes59] guarantees that the intermediate state of the system stays close to the ground state of $H(s)$ at all points in the evolution. Thus, at $s = 1$ we will end up close to the ground state of H_P and thus be able to recover the solution of the problem.

In this chapter we study the similarities between semi-absorbing Markov chains and adiabatic quantum evolutions. We carry over some of the classical concepts to the quantum case by using results from the previous chapter. In particular, we define a quantum version of the classical hitting time, and show how the two notions can be related using the *folk* Adiabatic Theorem.

3.1.1 Overview of the main result

In this section we briefly summarize the main result of this chapter, explain how we arrived at it and what is the intuition behind our approach.

Informally, our result can be described as follows. Given a Markov chain P that finds a marked element from a set M in time $\mathrm{HT}(P, M)$, we construct an adiabatic quantum algorithm that performs the same task in time $\sqrt{\mathrm{HT}(P, M)}$. The main idea of our approach is the following. Instead of considering a quantum analogue of either P or P', we use the semi-absorbing Markov chain $P(s)$ and construct a quantum Hamiltonian $H(s)$ that adiabatically interpolates between the two cases. Let explain why this is a reasonable thing to do and give some more details on how it works.

Recall from Sect. 2.4.1 that the original classical **Random Walk Algorithm** consists in applying the absorbing walk P' on the stationary distribution π of P. Since any stationary distribution of P' has support only on marked elements, the initial distribution π is far from being stationary, say, in the statistical distance. Intuitively this means that initially the system is far from equilibrium. As the system evolves, the random walk P'

damps any non-stationary contributions of the initial distribution until the final distribution has most of its weight on marked states. However, a quantum walk based either on P or P' seems to have trouble with performing such irreversible task. Thus, it is not immediately clear how to generalize the classical algorithm to the quantum case.

However, there is a situation in quantum mechanics where a similar phenomenon does occur. Consider an adiabatic evolution based on Hamiltonian $H(s)$, and assume that the system is initialized close to the ground state of the initial Hamiltonian $H(0)$. If $H(s)$ varies slowly, the *folk* Adiabatic Theorem ensures that contributions from excited states will cancel out and the system will remain close to its ground state.

Therefore, our strategy is to first modify the classical Markov chain, so that the system stays close to equilibrium throughout the evolution, and then translate it into an adiabatic quantum algorithm. We will use the interpolated Markov chain $P(s) = (1-s)P + sP'$ from Section 2.2.1 to drive the stationary distribution from π to π' by slowly switching s from 0 to 1. Then the system at all times will remain close to the stationary distribution $\pi(s)$ of $P(s)$. Intuitively, this modification of the **Random Walk Algorithm** should not create too much overhead, so the new classical algorithm still runs in time $O(\mathrm{HT}(P, M))$. However, now the system remains close to equilibrium at all times, so we are in a better shape for designing a quantum analogue that relies on the Adiabatic Theorem.

To design a Hamiltonian $H(s)$ that corresponds to the Markov chain $P(s)$, we use the construction by Somma and Ortiz [SO10], which is based on (and can be thought of as a Hamiltonian equivalent of) the original construction due to Szegedy [Sze04a]. It maps $P(s)$ to a Hamiltonian $H(s)$ that acts on the "edge space" of the transition graph underlying $P(s)$ (this will be discussed in more detail in Sect. 3.2.1). Intuitively, $H(s)$ implements a continuous-time quantum walk on the edges of the graph.

Here is how our **Adiabatic Search Algorithm** proceeds. First, we initialize the system in state $|\pi\rangle|\bar{0}\rangle$, where $|\pi\rangle$ is a quantum version of the stationary distribution π of P and $|\bar{0}\rangle$ is some reference state. Then we evolve it according to $H(s)$ with some schedule $s(t)$, where time t varies from 0 to T. The Hamiltonian $H(s)$ is constructed so that according to the *folk* Adiabatic Theorem the intermediate state of the system remains close to $|\pi(s)\rangle|\bar{0}\rangle$, where $|\pi(s)\rangle$ is the quantum version of the stationary distribution $\pi(s)$ of $P(s)$. In fact, the schedule $s(t)$ is chosen so that $|\pi(s(t))\rangle$ rotates with constant angular velocity from $|\pi\rangle$ to a superposition over marked elements $|M\rangle$ as t ranges from 0 to T. Finally, we measure the first register of the resulting state in the standard basis to get a marked element. Here is a summary of the algorithm:

Adiabatic Search Algorithm

1. Prepare the state $|\pi\rangle|\bar{0}\rangle$.
2. Apply a time-dependent Hamiltonian $H(s)$ with schedule $s(t)$ from $t = 0$ to $t = T$.
3. Measure the first register in the standard basis.

The main result of this chapter is Theorem 3.6, which states that the above algorithm finds a marked element with high probability if $T = O(\sqrt{\mathrm{HT}(P, M)})$. This corresponds to a quadratic speed-up over the classical case. While this result relies on the *folk* Adiabatic Theorem, a similar statement can be made in the circuit-based model of quantum computing, where no such condition is necessary—this will be the topic of Chapter 4. Results in the present chapter are parallel to those in Chapter 4 and are not necessary for constructing or understanding the circuit-based quantum algorithm. Nevertheless, the intuition behind this algorithm originates from the adiabatic version presented here.

The rest of this chapter is structured as follows. In Sect. 3.2 we construct the interpolating Hamiltonian $H(s)$ and evaluate its spectrum. We determine the relevant subspace for the adiabatic evolution in Sect. 3.3, as well as state a formal Adiabaticity Requirement that captures the intuition behind the *folk* Adiabatic Theorem. In Sect. 3.4 we choose the evolution schedule $s(t)$, impose the adiabatic condition, and determine the running time of the resulting adiabatic quantum algorithm. At the end of this section we prove Theorem 3.6, which is the main result of this chapter. It relates the classical hitting time of P and the running time of the adiabatic evolution, thus providing a quadratic speed-up in the quantum case. Finally, we conclude in Sect. 3.5.

3.2 Interpolating Hamiltonian

3.2.1 Definition

Szegedy [Sze04a] proposed a general method to map a random walk to a unitary operator that defines a quantum walk. Somma and Ortiz [SO10] showed how Szegedy's method may be adapted to build a Hamiltonian. We apply this method to the semi-absorbing random walk $P(s)$ introduced in Sect. 2.2.1.

The first step of Szegedy's construction is to map the rows of $P(s)$ to quantum states. Let X be the state space of $P(s)$ and $\mathcal{H} := \mathrm{span}\{|x\rangle : x \in X\}$ be a complex Euclidean

space of dimension $n = |X|$ with basis states labelled by elements of X. For every $x \in X$ we define the following state in $\mathcal{H}$:

$$|p_x(s)\rangle := \sum_{y \in X} \sqrt{P_{xy}(s)}|y\rangle. \tag{3.1}$$

Notice that these states are correctly normalized, since $P(s)$ is row-stochastic. Following Szegedy [Sze04a], we define a unitary operator $V(s)$ acting on $\mathcal{H} \otimes \mathcal{H}$ as

$$V(s)|x, \bar{0}\rangle := |x\rangle|p_x(s)\rangle = \sum_{y \in X} \sqrt{P_{xy}(s)}|x, y\rangle, \tag{3.2}$$

when the second register is in some reference state $|\bar{0}\rangle \in \mathcal{H}$, and arbitrarily otherwise. It will not be relevant to us how $V(s)$ is extended from $\mathcal{H} \otimes |\bar{0}\rangle$ to $\mathcal{H} \otimes \mathcal{H}$. The only constraint we impose is that $V(s)$ is continuous as a function of s, which is a reasonable assumption from a physical point of view.

Let $S : |x, y\rangle \mapsto |y, x\rangle$ be the gate that swaps the two registers and let $\Pi_0 := I \otimes |\bar{0}\rangle\langle\bar{0}|$ be the projector that keeps only the component containing the reference state $|\bar{0}\rangle$ in the second register.

Proposition 3.1. *When restricted to $|\bar{0}\rangle$ in the second register, operator $V^\dagger(s)SV(s)$ acts as the discriminant matrix $D(s)$ introduced in Sect. 2.3.1:*

$$\Pi_0 V^\dagger(s) S V(s) \Pi_0 = D(s) \otimes |\bar{0}\rangle\langle\bar{0}|. \tag{3.3}$$

Proof. From Eq. (3.2) we have:

$$\begin{aligned}
\langle x, \bar{0}|V^\dagger(s)SV(s)|y, \bar{0}\rangle &= \langle x, p_x(s)|S|y, p_y(s)\rangle && (3.4)\\
&= \langle p_x(s)|y\rangle\langle x|p_y(s)\rangle && (3.5)\\
&= \sqrt{P_{xy}(s)P_{yx}(s)} && (3.6)\\
&= D_{xy}(s) && (3.7)
\end{aligned}$$

where last equality follows from Eq. (2.24). □

Following Somma and Ortiz [SO10], we define the Hamiltonian $H(s)$ on $\mathcal{H} \otimes \mathcal{H}$ as

$$H(s) := i\big[V^\dagger(s)SV(s), \Pi_0\big] \tag{3.8}$$

where $[A, B] := AB - BA$ is the commutator.

3.2.2 Spectral decomposition

To understand the properties of the Hamiltonian $H(s)$, let us find its spectral decomposition. Our strategy will be to express the eigenvalues and eigenvectors of $H(s)$ in terms of those of $D(s)$.

Recall from Eq. (2.29) that $D(s) = \sum_{i=1}^{n} \lambda_i(s)|v_i(s)\rangle\langle v_i(s)|$ is the spectral decomposition of $D(s)$. Let us consider the following subspaces of $\mathcal{H} \otimes \mathcal{H}$ defined in terms of the eigenvectors of $D(s)$ and the operator $V^\dagger(s)SV(s)$:

$$\mathcal{B}_k(s) := \operatorname{span}\{|v_k(s), \bar{0}\rangle, V^\dagger(s)SV(s)|v_k(s), \bar{0}\rangle\}, \quad k \in \{1, \dots, n-1\}, \tag{3.9}$$

$$\mathcal{B}_n(s) := \operatorname{span}\{|v_n(s), \bar{0}\rangle\}, \tag{3.10}$$

$$\mathcal{B}^\perp(s) := \left(\bigoplus_{k=1}^{n} \mathcal{B}_k(s)\right)^\perp. \tag{3.11}$$

Normally the subspaces $\mathcal{B}_k(s)$ for $k \neq n$ have dimension two, but in some special cases they might have dimension one. To take care of this, we will impose continuity on $\mathcal{B}_k(s)$, so that it is two-dimensional for any $s \in [0, 1]$.

Assume that $\lambda_k(s) \neq 1$. Then by unitarity of $V^\dagger(s)SV(s)$ and Prop. 3.1,

$$V^\dagger(s)SV(s)|v_k(s), \bar{0}\rangle = \lambda_k(s)|v_k(s), \bar{0}\rangle + \sqrt{1 - \lambda_k(s)^2}|v_k(s), \bar{0}\rangle^\perp \tag{3.12}$$

for some unit vector $|v_k(s), \bar{0}\rangle^\perp$ orthogonal to $|v_k(s), \bar{0}\rangle$ and lying in the subspace $\mathcal{B}_k(s)$. Note that $|v_k(s), \bar{0}\rangle^\perp$ depends on how operator $V(s)$, defined in Eq. (3.2), is extended to the rest of the space $\mathcal{H} \otimes \mathcal{H}$. The only case when $|v_k(s), \bar{0}\rangle^\perp$ is not well-defined is when $\lambda_k(s) = 1$. By Prop. 2.11, this is exactly when $s = 1$ and $n - m < k < n$. Since $V(s)$ is continuous, we can assume that so is $|v_k(s), \bar{0}\rangle^\perp$ and define $|v_k(1), \bar{0}\rangle^\perp := \lim_{s\to 1} |v_k(s), \bar{0}\rangle^\perp$ and include this vector in the corresponding subspace $\mathcal{B}_k(1)$. This ensures that $\mathcal{B}_k(s)$ remains two-dimensional for all $s \in [0, 1]$ and $k \neq n$.

Let us also find how $V^\dagger(s)SV(s)$ acts on $|v_k(s), \bar{0}\rangle^\perp$. If we apply $V^\dagger(s)SV(s)$ to both sides of Eq. (3.12), we get

$$|v_k(s), \bar{0}\rangle = \lambda_k(s)V^\dagger(s)SV(s)|v_k(s), \bar{0}\rangle + \sqrt{1 - \lambda_k(s)^2}V^\dagger(s)SV(s)|v_k(s), \bar{0}\rangle^\perp. \tag{3.13}$$

We regroup the terms and substitute Eq. (3.12):

$$\sqrt{1 - \lambda_k(s)^2}V^\dagger(s)SV(s)|v_k(s), \bar{0}\rangle^\perp = |v_k(s), \bar{0}\rangle - \lambda_k(s)V^\dagger(s)SV(s)|v_k(s), \bar{0}\rangle \tag{3.14}$$

$$= |v_k(s), \bar{0}\rangle - \lambda_k(s)\Big(\lambda_k(s)|v_k(s), \bar{0}\rangle + \sqrt{1 - \lambda_k(s)^2}|v_k(s), \bar{0}\rangle^\perp\Big). \tag{3.15}$$

After cancellation we get

$$V^\dagger(s)SV(s)|v_k(s),\bar{0}\rangle^\perp = \sqrt{1-\lambda_k(s)^2}|v_k(s),\bar{0}\rangle - \lambda_k(s)|v_k(s),\bar{0}\rangle^\perp. \tag{3.16}$$

Proposition 3.2. *Subspaces $\mathcal{B}_1(s),\ldots,\mathcal{B}_n(s)$, and $\mathcal{B}^\perp(s)$ are mutually orthogonal and invariant under $H(s)$ for all $s\in[0,1]$.*

Proof. Clearly, $\mathcal{B}^\perp(s)$ is orthogonal to the other subspaces. Vectors $|v_k(s),\bar{0}\rangle$ are also mutually orthogonal for $k\in\{1,\ldots,n\}$, since they form an orthonormal basis of $\mathcal{H}\otimes|\bar{0}\rangle$. Finally, note from Prop. 3.1 that

$$\langle v_j(s),\bar{0}|\cdot V^\dagger(s)SV(s)|v_k(s),\bar{0}\rangle = \langle v_j(s)|D(s)|v_k(s)\rangle = \delta_{jk}\lambda_k(s), \tag{3.17}$$

so $V^\dagger(s)SV(s)|v_k(s),\bar{0}\rangle$ is orthogonal to $|v_j(s),\bar{0}\rangle$ for any $j\neq k$. Thus all of the above subspaces are mutually orthogonal.

Let us show that these subspaces are invariant under the Hamiltonian $H(s)$. From the definition of $H(s)$ in Eq. (3.8) we see that it suffices to check the invariance of each subspace under $V^\dagger(s)SV(s)$ and Π_0 separately.

First, let us argue the invariance under $V^\dagger(s)SV(s)$. Since the swap gate S squares to identity, then so does $V^\dagger(s)SV(s)$ and hence $\mathcal{B}_k(s)$ is invariant under $V^\dagger(s)SV(s)$ for any $k<n$. Next, $\mathcal{B}_n(s)$ is invariant, since $V^\dagger(s)SV(s)$ acts trivially on $|v_n(s),\bar{0}\rangle$ by Prop. 3.1. Finally, $\mathcal{B}^\perp(s)$ is invariant, since it is the orthogonal complement of invariant subspaces.

Let us now show the invariance under Π_0. First, let us argue that

$$\langle v_j(s),\bar{0}|v_k(s),\bar{0}\rangle^\perp = 0, \quad \forall j\in\{1,\ldots,n\}. \tag{3.18}$$

These vectors lie in subspaces $\mathcal{B}_j(s)$ and $\mathcal{B}_k(s)$ which are mutually orthogonal when $j\neq k$. For $j=k$ this holds by definition of $|v_k(s),\bar{0}\rangle^\perp$. Since $\mathrm{span}\{|v_k(s),\bar{0}\rangle\}_{k=1}^n = \mathcal{H}\otimes|\bar{0}\rangle$, we conclude that

$$\Pi_0|v_k(s),\bar{0}\rangle^\perp = 0. \tag{3.19}$$

From Eq. (3.12) we get

$$\Pi_0 V^\dagger(s)SV(s)|v_k(s),\bar{0}\rangle = \lambda_k(s)|v_k(s),\bar{0}\rangle, \tag{3.20}$$

hence $\mathcal{B}_k(s)$ is invariant under Π_0 for $k<n$. Next, $\mathcal{B}_n(s)$ is invariant since $\Pi_0|v_n(s),\bar{0}\rangle = |v_n(s),\bar{0}\rangle$. Finally, $\mathcal{B}^\perp(s)$ is invariant by being the orthogonal complement of invariant subspaces. □

Now we can find the eigenvalues and eigenvectors of $H(s)$ by restricting its action to each of the invariant subspaces. The following lemma is in the spirit of Jordan's lemma [NWZ09] and appears in the work of Somma and Ortiz [SO10] who consider the same Hamiltonian. Later we will prove a unitary version of this result by finding the eigenvalues of a product of two reflections (see Lemma 4.7 in Sect. 4.3.1, originally due to Szegedy [Sze04a]).

Lemma 3.3 ([SO10]). *$H(s)$ has the following eigenvalues and eigenstates.*

$$\text{On } \mathcal{B}_k(s): \quad E_k^{\pm}(s) := \pm\sqrt{1-\lambda_k(s)^2}, \quad |\Psi_k^{\pm}(s)\rangle := \frac{|v_k(s),\bar{0}\rangle \pm i|v_k(s),\bar{0}\rangle^{\perp}}{\sqrt{2}}. \tag{3.21}$$

$$\text{On } \mathcal{B}_n(s): \quad E_n(s) := 0, \quad |\Psi_n(s)\rangle := |v_n(s),\bar{0}\rangle. \tag{3.22}$$

$$\text{On } \mathcal{B}^{\perp}(s): \quad F_j(s) := 0, \quad |\Phi_j(s)\rangle. \tag{3.23}$$

Here $k \neq n$ and $\{|\Phi_j(s)\rangle : j = 1, \ldots, (n-1)^2\}$ is an arbitrary orthonormal basis of $\mathcal{B}^{\perp}(s)$.

Proof. We consider the case $s \in [0,1)$; the case $s = 1$ follows by continuity. By Prop. 3.1, $V^{\dagger}(s)SV(s)|v_n(s),\bar{0}\rangle = D(s)|v_n(s)\rangle \otimes |\bar{0}\rangle = |v_n(s),\bar{0}\rangle$, so $|v_n(s),\bar{0}\rangle$ is an eigenstate of $H(s)$ with eigenvalue 0. For $k \neq n$, from Eqs. (3.12) and (3.19) we get

$$V^{\dagger}(s)SV(s)\Pi_0|v_k(s),\bar{0}\rangle = \lambda_k(s)|v_k(s),\bar{0}\rangle + \sqrt{1-\lambda_k(s)^2}|v_k(s),\bar{0}\rangle^{\perp}, \tag{3.24}$$

$$\Pi_0 V^{\dagger}(s)SV(s)|v_k(s),\bar{0}\rangle = \lambda_k(s)|v_k(s),\bar{0}\rangle. \tag{3.25}$$

Similarly, for $|v_k(s),\bar{0}\rangle^{\perp}$ we have

$$V^{\dagger}(s)SV(s)\Pi_0|v_k(s),\bar{0}\rangle^{\perp} = 0, \tag{3.26}$$

$$\Pi_0 V^{\dagger}(s)SV(s)|v_k(s),\bar{0}\rangle^{\perp} = \sqrt{1-\lambda_k(s)^2}|v_k(s),\bar{0}\rangle, \tag{3.27}$$

where we used Eq. (3.16). By combining these expressions we get

$$H(s)|v_k(s),\bar{0}\rangle = i\sqrt{1-\lambda_k(s)^2}|v_k(s),\bar{0}\rangle^{\perp}, \tag{3.28}$$

$$H(s)|v_k(s),\bar{0}\rangle^{\perp} = -i\sqrt{1-\lambda_k(s)^2}|v_k(s),\bar{0}\rangle, \tag{3.29}$$

where the second line can be obtained directly, or from the fact that $H(s)$ is Hermitian, traceless and preserves $\mathcal{B}_k(s)$. If $\sigma_y := \left(\begin{smallmatrix} 0 & -i \\ i & 0 \end{smallmatrix}\right)$ is the Pauli y matrix, then $H(s)$ acts as $\sqrt{1-\lambda_k(s)^2}\,\sigma_y$ in the basis $\{|v_k(s),\bar{0}\rangle, |v_k(s),\bar{0}\rangle^{\perp}\}$ of $\mathcal{B}_k(s)$. This yields Eq. (3.21).

Finally, $H(s)$ restricted to $\mathcal{B}^{\perp}(s)$ is equal to zero, since $\mathcal{B}^{\perp}(s) \subset \mathcal{H} \otimes \{|\bar{0}\rangle\}^{\perp}$ but $H(s)$ involves Π_0. Hence, the remaining $n^2 - (2n-1) = (n-1)^2$ eigenvalues of $H(s)$ are zero. □

3.3 The adiabatic condition

In adiabatic quantum computing it is a common practice to associate the intermediate state of the computation with the ground state (*i.e.*, the lowest energy eigenstate) of the Hamiltonian. However, from Lemma 3.3 we see that the spectrum of $H(s)$ is symmetric about zero and the state that we are interested in lies in the middle of the spectrum. Hence, we will not use the ground state of $H(s)$, which has negative energy, but instead we will use the zero-eigenvector $|\Psi_n(s)\rangle = |v_n(s), \bar{0}\rangle$. Indeed, recall from Eq. (2.37) in Sect. 2.3.3 that $|v_n(s)\rangle = \sqrt{\pi(s)^\mathsf{T}}$, so this state is closely related to $\pi(s)$, the stationary distribution of $P(s)$. In particular, the problem of finding a marked vertex would be solved if we can reach the state $|\Psi_n(1)\rangle$, as measuring the first register of this state yields a vertex distributed according to $\pi(1)$, which only has support on marked vertices.

3.3.1 The relevant subspace for adiabatic evolution

We initially prepare the system in the zero-eigenvector $|\Psi_n(0)\rangle$ of $H(0)$ and then start to change the Hamiltonian $H(s)$ by slowly increasing the parameter s from 0 to 1 according to some *schedule* $s(t)$. If the schedule $s(t)$ is chosen so that it satisfies certain conditions, the system is guaranteed to stay close to the intermediate zero-eigenstate $|\Psi_n(s)\rangle$ of $H(s)$. Then, at $s = 1$, the state will be close to $|\Psi_n(1)\rangle = |v_n(1), 0\rangle$, where the first register only has overlap over marked vertices, so that a measurement yields a marked vertex with high probability.

Recall from Lemma 3.3 that the zero-eigenspace $\mathcal{B}_n(s) \cup \mathcal{B}^\perp(s)$ of $H(s)$ has a huge dimension. Thus, before we apply the adiabatic condition, we have to make sure that the non-relevant part $\mathcal{B}^\perp(s)$ is totally decoupled from $|\Psi_n(s)\rangle$, the only zero-eigenvector that is relevant for our algorithm. In particular, we want to show that

$$\langle \Phi_j(s)| \cdot \frac{d}{dt} |\Psi_n(s)\rangle = 0 \tag{3.30}$$

for any $j \in \{1, \ldots, (n-1)^2\}$, since this would imply that during the evolution $|\Psi_n(s)\rangle$ is not leaked into the subspace $\mathcal{B}^\perp(s)$ spanned by states $|\Phi_j(s)\rangle$. To see that this is indeed the case, note that

$$|\Phi_j(s)\rangle \perp \text{span}\{|\Psi_1^\pm(s)\rangle, \ldots, |\Psi_{n-1}^\pm(s)\rangle, |\Psi_n(s)\rangle\} \tag{3.31}$$

for any $j \in \{1, \ldots, (n-1)^2\}$, since the eigenvectors of $H(s)$ form an orthonormal basis. In particular, from Eq. (3.21) we get

$$|\Phi_j(s)\rangle \perp \text{span}\{|\Psi_1^+(s)\rangle + |\Psi_1^-(s)\rangle, \ldots, |\Psi_{n-1}^+(s)\rangle + |\Psi_{n-1}^-(s)\rangle, |\Psi_n(s)\rangle\} \tag{3.32}$$

$$= \text{span}\{|v_1(s), \bar{0}\rangle, \ldots, |v_{n-1}(s), \bar{0}\rangle, |v_n(s), \bar{0}\rangle\}. \tag{3.33}$$

Recall from Eq. (3.22) that $\frac{d}{dt}|\Psi_n(s)\rangle = \frac{d}{dt}|v_n(s)\rangle|\bar{0}\rangle$, so the inner product in Eq. (3.30) indeed vanishes. Thus, we can safely apply the adiabatic condition only for the relevant subspace $\bigoplus_{k=1}^{n} \mathcal{B}_k(s)$ in which the zero-eigenstate is not degenerate.

3.3.2 The quantum adiabatic theorem

To formalize the intuition behind the *folk* Quantum Adiabatic Theorem [Mes59], let us state a specific adiabaticity requirement for Hamiltonian $H(s)$.

Definition 3.4 (Adiabaticity Requirement)**.** Assume a quantum system starts in the zero-eigenstate $|\Psi_n(0)\rangle$ of Hamiltonian $H(0)$ and evolves according to $H(s)$ with schedule $s(t)$. We say that $H(s)$ satisfies the *adiabatic condition* for $\varepsilon > 0$ if

$$\forall t : \sum_{\sigma=\pm 1} \sum_{k=1}^{n-1} \frac{\left|\langle \Psi_k^\sigma(s)| \cdot \frac{d}{dt}|\Psi_n(s)\rangle\right|^2}{\left(E_k^\sigma(s) - E_n(s)\right)^2} \leq \varepsilon^2. \tag{3.34}$$

Moreover, if $|\psi(t)\rangle$ (the state of the system at time t) stays close to the intermediate zero-eigenstate $|\Psi_n(s(t))\rangle$ throughout the evolution, more precisely,

$$\forall t : \left|\langle \Psi_n(s(t))|\psi(t)\rangle\right|^2 \geq 1 - \varepsilon^2, \tag{3.35}$$

we say that the *Adiabaticity Requirement* is satisfied.

While the condition in Eq. (3.34) is known not to be sufficient in full generality (see, *e.g.*, the discussion in [JRS07]), we will assume that it can be applied in our setup. We will discuss how this assumption may be suppressed in Sect. 3.5 and provide a quantum algorithm that does not rely on it in Chapter 4.

Let us write the adiabatic condition for our Hamiltonian $H(s)$ explicitly, by inserting the eigenvalues and eigenvectors from Lemma 3.3 into Eq. (3.34):

$$\forall t : \sum_{k:\, \lambda_k(s) \neq 1} \frac{\left|\langle v_k(s)| \cdot \frac{d}{dt}|v_n(s)\rangle\right|^2}{1 - \lambda_k^2(s)} \leq \varepsilon^2. \tag{3.36}$$

Note that this condition is purely in terms of the eigenvalues and eigenvectors of the discriminant matrix $D(s)$.

3.4 Analysis of running time

In this section we will analyze the running time of an adiabatic quantum algorithm which evolves according to the Hamiltonian $H(s)$. We will make a specific choice of the evolution schedule $s(t)$, plug it into the adiabatic condition from the Adiabaticity Requirement, and derive a bound on the required running time T.

3.4.1 Choice of the schedule

Recall from Prop. 2.13 in Sect. 2.3.3 that

$$|v_n(s)\rangle = \cos\theta(s)|U\rangle + \sin\theta(s)|M\rangle, \tag{3.37}$$

so the evolution of the zero-eigenstate $|\Psi_n(s)\rangle = |v_n(s), \bar{0}\rangle$ corresponds to a rotation in the two-dimensional subspace $\text{span}\{|U,\bar{0}\rangle, |M,\bar{0}\rangle\}$. Since the second register of $|\Psi_n(s)\rangle$ is always in the state $|\bar{0}\rangle$, we will omit it from the analysis for simplicity. Let us choose schedule $s : [0,T] \to [0,1]$ so that $|v_n(s)\rangle$ rotates with constant angular velocity from $|v_n(0)\rangle$ to $|M\rangle$ in time T.

Proposition 3.5. *Let* $\theta_0 := \arcsin\sqrt{p_M}$ *and* $\omega := (\arccos\sqrt{p_M})/T$. *If*

$$s(t) := \frac{1}{1-p_M}\left(1 - \frac{p_M}{\sin^2(\omega t + \theta_0)}\right) \tag{3.38}$$

then $\theta(s(t)) = \omega t + \theta_0$.

Proof. Recall from Eq. (2.41) that

$$\sin\theta(s) = \sqrt{\frac{p_M}{1 - s(1-p_M)}} = \sin(\omega t + \theta_0), \tag{3.39}$$

where the second equality is what we desire. By solving for s we get Eq. (3.38). To find the constants θ_0 and ω, we use the boundary conditions at $s = 0$ and $s = 1$ ($t = 0$ and $t = T$, respectively). When we plug $s = 0$ and $t = 0$ in the above equation, we get $\sqrt{p_M} = \sin\theta_0$. The other boundary condition gives us $\omega T + \theta_0 = \pi/2$, so $\omega T = \pi/2 - \theta_0 = \arccos\sqrt{p_M}$. □

3.4.2 Running time

In this section we will use the adiabatic condition from Eq. (3.36) to determine the running time of our algorithm when schedule $s(t)$ from Eq. (3.38) is used.

First, we will find the derivative $\frac{d}{dt}|v_n(s)\rangle$. Let

$$|v_n^{\perp}(s)\rangle := -\sin\theta(s)|U\rangle + \cos\theta(s)|M\rangle \tag{3.40}$$

be a unit vector such that $\{|v(s)\rangle, |v_n^{\perp}(s)\rangle\}$ is an orthonormal basis of $\mathrm{span}\{|U\rangle, |M\rangle\}$ for each s. Then from Eq. (3.37) and Prop. 3.5 we get

$$\frac{d}{dt}|v_n(s)\rangle = \frac{d}{dt}\theta(s) \cdot |v_n^{\perp}(s)\rangle = \omega|v_n^{\perp}(s)\rangle = \frac{1}{T}\arccos\sqrt{p_M}|v_n^{\perp}(s)\rangle. \tag{3.41}$$

To slightly simplify things, notice that $\arccos\sqrt{p_M} \leq \frac{\pi}{2}$. Now we can rewrite the adiabatic condition in Eq. (3.36) as follows:

$$\forall s : \frac{\pi^2}{4\varepsilon^2} \sum_{k:\,\lambda_k(s)\neq 1} \frac{|\langle v_k(s)|v_n^{\perp}(s)\rangle|^2}{1-\lambda_k^2(s)} \leq T^2. \tag{3.42}$$

If this condition is satisfied, then from the Adiabaticity Requirement it follows that at time $t = T$ we obtain a state $|\psi(T)\rangle$ which is close to $|\Psi_n(1)\rangle = |v_n(1)\rangle|\bar{0}\rangle = |M\rangle|\bar{0}\rangle$. In particular, measuring the first register of $|\psi(T)\rangle$ yields a marked vertex with probability at least $1-\varepsilon^2$, according to Eq. (3.35).

Intuitively, we want to change the parameter s slowly for the evolution to be adiabatic. This corresponds to choosing T big enough so that the inequality in Eq. (3.42) holds. Recall from Prop. 2.19 that we can replace P by $(P+I)/2$ to ensure that $\lambda_k(s) \geq 0$. Thus, $1-\lambda_k^2(s) = \big(1+\lambda_k(s)\big)\big(1-\lambda_k(s)\big) \geq 1-\lambda_k(s)$. Let us impose a slightly stronger condition on T in Eq. (3.42) by replacing $1-\lambda_k^2(s)$ with $1-\lambda_k(s)$. In addition, let us choose the smallest T that still satisfies the inequality and use it as the running time of our adiabatic algorithm:

$$T := \frac{\pi}{2\varepsilon} \max_{0\leq s\leq 1} \sqrt{\sum_{k:\,\lambda_k(s)\neq 1} \frac{|\langle v_k(s)|v_n^{\perp}(s)\rangle|^2}{1-\lambda_k(s)}}. \tag{3.43}$$

It turns out that there is a simple relationship between this quantity and $\mathrm{HT}(P, M)$, the hitting time of the Markov chain P with marked states M (see Sect. 2.4.1 for definition).

3.4.3 Relation to the classical hitting time

Theorem 3.6. *Let P be an ergodic and reversible Markov chain with a set of marked vertices M, and assume that the Adiabaticity Requirement holds for the Hamiltonian $H(s)$. Then the **Adiabatic Search Algorithm** finds a marked vertex with probability at least $1 - \varepsilon^2$ in time $T = \frac{\pi}{2\varepsilon}\sqrt{\mathrm{HT}(P, M)}$, where $\mathrm{HT}(P, M)$ is the hitting time of the classical Markov chain P with respect to the set of marked vertices M.*

Proof. Let us first express the running time T from Eq. (3.43) in terms of the extended hitting time $\mathrm{HT}(s)$ defined in Sect. 2.4.2. Recall from Prop. 2.18 that

$$\mathrm{HT}(s) = \langle U|A(s)|U\rangle, \qquad A(s) := \sum_{k:\,\lambda_k(s)\neq 1} \frac{|v_k(s)\rangle\langle v_k(s)|}{1-\lambda_k(s)}. \tag{3.44}$$

Note from Eq. (3.43) that T can also be expressed using $A(s)$:

$$T = \frac{\pi}{2\varepsilon} \max_{0\le s\le 1} \sqrt{\langle v_n^\perp(s)|A(s)|v_n^\perp(s)\rangle}. \tag{3.45}$$

To relate these two expressions, we should relate the way $A(s)$ acts on $|U\rangle$ and $|v_n^\perp(s)\rangle$. Recall from Prop. 2.20 that $A(s)|M\rangle = -\frac{\cos\theta(s)}{\sin\theta(s)}A(s)|U\rangle$. Thus,

$$\begin{aligned} A(s)|v_n^\perp(s)\rangle &= -\sin\theta(s)A(s)|U\rangle + \cos\theta(s)A(s)|M\rangle && (3.46)\\ &= \frac{1}{\sin\theta(s)}\big(-\sin^2\theta(s) - \cos^2\theta(s)\big)A(s)|U\rangle && (3.47)\\ &= -\frac{1}{\sin\theta(s)}A(s)|U\rangle. && (3.48) \end{aligned}$$

If we multiply both sides by $\langle v_n^\perp(s)|$, use the Hermiticity of $A(s)$, and apply the same equation again, we get

$$\langle v_n^\perp(s)|A(s)|v_n^\perp(s)\rangle = \frac{\langle U|A(s)|U\rangle}{\sin^2\theta(s)}. \tag{3.49}$$

Thus, we get the following relationship between T and $\mathrm{HT}(s)$:

$$T = \frac{\pi}{2\varepsilon} \max_{0\le s\le 1} \frac{\sqrt{\mathrm{HT}(s)}}{\sin\theta(s)}. \tag{3.50}$$

Finally, we express $\mathrm{HT}(s)$ in terms of $\mathrm{HT}(P, M)$ using Theorem 2.22 from Chapter 2. In particular, recall from Eq. (2.98) that $\mathrm{HT}(s) = \mathrm{HT}(P, M) \sin^4 \theta(s)$. When we substitute this in Eq. (3.50), we get $T = \frac{\pi}{2\varepsilon} \sqrt{\mathrm{HT}(P, M)} \cdot \max_{0 \leq s \leq 1} \sin \theta(s)$. Now it only remains to observe from the definition of $\sin \theta(s)$ in Eq. (2.41) that the maximum is reached at $s = 1$ and is equal to $\sin \theta(1) = 1$. □

3.5 Conclusion and discussion

Our adiabatic quantum algorithm defines a new notion of quantum hitting time, which is quadratically smaller than the classical hitting time for any reversible Markov chain and any set of marked elements. Our algorithm only requires minimal assumptions, while previous approaches were subject to various restrictions, *e.g.*, the quantum algorithm could only detect the presence of marked elements [Sze04a], did not always provide full quadratic speed-up [MNRS07], or could only be applied for state-transitive Markov chains with a unique marked element [MNRS12].

Indeed, let us argue why the only remaining condition, reversibility, is necessary. Let us consider the Markov chain on a cycle $P = (I + C)/2$, where C implements a clockwise shift, *i.e.*, $C|x\rangle = |(x + 1) \mod n\rangle$. This Markov chain is ergodic but not reversible. While its classical hitting time is $\Theta(n)$, a simple locality argument implies that any quantum operator acting locally on the cycle requires time $\Omega(n)$ to find a marked vertex, so that a quadratic speed-up cannot be achieved. Magniez *et al.* [MNRS12] have also shown that under reasonable conditions the quadratic speed-up is optimal. This provides evidence that our result is both as strong and as general as possible.

While our result relies on the assumption that the *folk* adiabatic condition is sufficient, this assumption could be suppressed in different ways. One option would be to actually prove that the Adiabaticity Requirement holds in our setup, as was previously done for the adiabatic version of Grover's algorithm [JRS07]. Another option would be to circumvent adiabatic evolution altogether, by using a sequence of measurements or quantum Zeno effect as in [CDF+02]. Their technique provides a quantum circuit realizing the same evolution as the adiabatic approach with a similar running time, but without relying on the adiabatic condition. This leads to a quantum circuit algorithm described in Chapter 4.

Finally, note that in order to design the schedule $s(t)$, our algorithm requires knowledge of p_M and the order of magnitude of $\mathrm{HT}(P, M)$. These assumptions are standard in other quantum algorithms for this problem. In particular, a similar issue arises in

Grover's algorithm when the number of marked elements is unknown. In Grover's case, there are techniques to deal with this issue [BBHT98], and similar techniques could be applied in our case. While we do not provide a full answer to these questions here, they do not present any new technical difficulty. More details are given in Chapter 4 where we consider a quantum circuit version of our adiabatic algorithm.

Chapter 4

Finding is as easy as detecting for quantum walks

Contents

4.1 Introduction

Many randomized classical algorithms rely heavily on random walks or Markov chains. This technique has been extended to the quantum case and is called *quantum walk*. Ambainis [Amb04] was the first to solve a natural problem—the element distinctness problem—using a quantum walk. Following this, many other quantum walk algorithms were discovered, for example, [MSS05, BŠ06, MN07].

A common class of problems that are typically solved using a random walk are the so-called *spatial search problems*. In such problems, the displacement constraints are modelled by edges of an undirected graph G, which has some desired subset of vertices M that are marked. The goal of a spatial search problem is to find one of the marked vertices by traversing the graph along its edges. Classically, a simple strategy for finding a marked vertex is to perform a random walk on G, by repeatedly applying some stochastic matrix P until one of the marked vertices is reached. This is exactly how the **Random Walk Algorithm** that we discussed in Sect. 2.4.1 works. The expected running time of this algorithm is called the *hitting time* of P and is denoted by $\mathrm{HT}(P, M)$.

Quantum walk algorithms for the spatial search problem were studied in [AA05]. This problem has also been considered for several specific graphs, such as the hypercube [SKW03] and the grid [CG04a, AKR05]. The notion of the hitting time has been carried over to the quantum case in [AKR05, Kem05, Sze04a, KB06, MNRS07, MNRS12, VKB08] by generalizing the classical notion in different ways. Usually, the quantum hitting time has a quadratic improvement over the classical one. However, until the present work several serious restrictions were imposed for this to be the case. A quantum algorithm could only solve the *detection problem* of deciding whether there are marked vertices or not [Sze04a], but for being able to *find* them, the Markov chain had to be reversible, state-transitive, and with a unique marked vertex [Tul08, MNRS12]. The detection algorithm is quite intuitive and well understood, whereas the finding algorithm requires an elaborate proof whose intuition is not clear. This is due in part to a modification of the quantum walk, so that the resulting walk is not a quantum analogue of a Markov chain anymore.

Whether this quadratic speed-up for finding a marked element also holds for all reversible Markov chains and for multiple marked elements was an open question. In this chapter we give a positive answer to this question by providing a quantum algorithm for solving this problem.

4.1.1 Related work

Inspired by Ambainis' quantum walk algorithm for solving the element distinctness problem [Amb04], Szegedy [Sze04a] has introduced a powerful way of constructing quantum analogues of Markov chains which led to new quantum walk-based algorithms. He showed that for any symmetric Markov chain a quantum walk could detect the presence of marked vertices in at most the square root of the classical hitting time. However, showing that a marked vertex could also be found in the same time (as is the case for the classical algorithm) proved to be a very difficult task. Magniez *et al.* [MNRS07] extended Szegedy's approach to the larger class of ergodic Markov chains, and proposed a quantum walk-based algorithm to find a marked vertex, but its complexity may be larger than the square root of the classical hitting time. A typical example where their approach fails to provide a quadratic speed-up is the 2D grid, where their algorithm has complexity $\Theta(n)$, whereas the classical hitting time is $\Theta(n \log n)$. Ambainis *et al.* [AKR05] and Szegedy's [Sze04a] approaches yield a complexity of $\Theta(\sqrt{n} \log n)$ in this special case, for a unique marked vertex. Childs and Goldstone [CG04b, CG04a] also obtained a similar result using a continuous-time quantum walk.

However, whether a full quadratic speed-up was possible in the 2D grid case remained an open question, until Tulsi [Tul08] proposed a solution involving a new technique. Magniez *et al.* [MNRS12] extended Tulsi's technique to any reversible state-transitive Markov chain, showing that for such chains, it is possible to find a unique marked vertex with a full quadratic speed-up over the classical hitting time. However, the state-transitivity is a strong symmetry condition, and furthermore their technique cannot deal with multiple marked vertices. Recently [ABN+11] have suggested to modify the original [AKR05] algorithm in the case of the 2D grid with a single marked element, by replacing amplitude amplification with classical search in a neighbourhood of the final vertex. This results in a $\sqrt{\log n}$ speed-up over the original algorithm from [AKR05] and yields complexity $O(\sqrt{n \log n})$ as in the case of [Tul08, MNRS12].

It seems implausible that one has to rely on involved techniques to solve the finding problem under such restricted conditions, while the classical **Random Walk Algorithm** is conceptually simple and works under general conditions. Recall from Sect. 2.4.1 that it works as follows: starting with the stationary distribution π of P, we repeatedly apply the absorbing walk P' until most of the probability is absorbed in marked vertices, at which point the state is close to a stationary distribution of P'.

Previous attempts at providing a quantum speed-up over this classical algorithm have followed one of these two approaches:

- Combining a quantum version of P with a reflection through marked vertices to mimic a Grover operation [AKR05, Amb04, MNRS07].
- Directly applying a quantum version of P' [Sze04a, MNRS12].

The problem with these approaches is that they would only be able to find marked vertices in very restricted cases. We explain this by the different nature of random and quantum walks: while both have a stable state, *i.e.*, the stationary distribution for the random walk and the eigenstate with eigenvalue 1 for the quantum walk, the way both walks act on other states is dramatically different.

Indeed, an ergodic random walk will converge to its stationary distribution from any initial distribution. This apparent robustness may be attributed to the inherent randomness of the walk, which will smooth out any initial perturbation. After many iterations of the walk, non-stationary contributions of the initial distribution will be damped and only the stationary distribution will survive (this can be attributed to the thermodynamical irreversibility[1] of ergodic random walks).

On the other hand, this is not true for quantum walks, because in the absence of measurements a unitary evolution is deterministic (and in particular thermodynamically reversible): the contributions of the other eigenstates will not be damped but just oscillate with different frequencies, so that the overall evolution is quasi-periodic. As a consequence, while iterations of P' always lead to a marked vertex, it may happen that iterations of the quantum analogue of P' will never lead to a state with a large overlap over marked vertices, unless the walk exhibits a strong symmetry (as is the case for a state-transitive walk with only one marked element, which could be addressed by previous approaches).

4.1.2 Our approach and contributions

The main result of this chapter is that a quadratic speed-up for finding a marked element via quantum walk holds for any reversible Markov chain with multiple marked elements. We provide several algorithms for different versions of the problem. Compared to previous results, our algorithms are more general and conceptually clean. The intuition behind our main algorithm is based on the adiabatic algorithm presented in Chapter 3. However, the algorithms presented in this chapter are circuit-based and thus do not suffer from the drawbacks of the adiabatic algorithm discussed in Sect. 3.5.

[1] Note that when we consider *reversible Markov chains* as defined in Sect. 2.2.3, this corresponds to a different notion of reversibility than in the usual thermodynamical sense. Actually, even a "reversible" Markov chain is thermodynamically irreversible.

We choose an approach that is different from the ones described above: first, we directly modify the original random walk P, and then construct a quantum analogue of the modified walk. We choose the modified walk to be the interpolated Markov chain $P(s) = (1-s)P + sP'$ that interpolates between P and the absorbing walk P' whose outgoing transitions from marked vertices have been replaced by self-loops. Thus, we can still use our intuition from the classical case, but at the same time also get simpler proofs and more general results in the quantum case.

All of the quantum walk algorithms presented in this chapter are based on eigenvalue estimation performed on operator $W(s)$, a quantum analogue of Markov chain $P(s)$. We consider the $(+1)$-eigenstate $|\Psi_n(s)\rangle$ of $W(s)$ that plays the role of the stationary distribution in the quantum case. We use the interpolation parameter s to tune the length of projections of $|\Psi_n(s)\rangle$ onto marked and unmarked vertices. If both projections are large, our algorithm succeeds with large probability in $\mathrm{O}\bigl(\sqrt{\mathrm{HT}(s)}\bigr)$ steps (Theorem 4.10).

We also provide several modifications of the main algorithm. In particular, we show how to make a suitable choice of s to balance the overlap of $|\Psi_n(s)\rangle$ on marked and unmarked vertices even if some of the parameters required by the main algorithm are unknown and the rest are either approximately known (Theorem 4.12 and Theorem 4.13) or bounded (Theorem 4.14 and Theorem 4.15). In all cases a marked vertex is found in $\mathrm{O}\bigl(\sqrt{\mathrm{HT}(P,M)}\bigr)$ steps. Finally, we use these results to make progress on an open problem from [AA05, Tul08] related to the spatial search on the 2D grid (Corollary 4.16).

In Sect. 4.2 we introduce the spatial search problem and provide some preliminaries on random and quantum walks and their hitting times. In Sect. 4.3 we describe Szegedy's method for constructing quantum analogues of Markov chains, and provide a circuit for implementing the quantum walk operator $W(s)$. Finally, Sect. 4.4 describes our algorithms and contains the main results of this chapter.

4.2 Preliminaries

In this chapter we shift our focus from Markov chains to search algorithms on graphs. In Chapter 2 and Chapter 3 our main objects of study were Markov chains, but their underlying graphs of transition probabilities were only of a secondary importance. In this chapter, however, we study the problem of traversing a given graph. Thus, our main object of study is the given graph, whereas a particular Markov chain for traversing

it is a derived concept. Another difference is that we put our results in a more general algorithmic framework: we consider an abstract search algorithm, break it up into elementary building blocks, and associate different costs to each of them.

4.2.1 Spatial search on graphs

Throughout this chapter, let us fix an undirected graph $G = (X, E)$ with $n := |X|$ vertices and a set of edges E. As before, let $M \subseteq X$ be a set of marked vertices of size $m := |M|$. We insist that the current vertex is stored in a distinguished *vertex register*. Our goal is to find any of the marked vertices in M using only evolutions that preserve the locality of G on the vertex register, *i.e.*, to perform a *spatial search* on G [AA05] (here we define an even more restricted notion of locality than the ones in [AA05], but it is more intuitive and sufficiently powerful for our purpose).

We allow two types of operations on the vertex register:

- *static transformations*, that can be conditioned on the state of the vertex register, but do not modify it;
- SHIFT, that exchanges the value of the vertex register and another register.

To impose locality, we want to restrict the execution of SHIFT only to the edges of G.

Definition 4.1. Let

$$\text{SHIFT} : (x, y) \mapsto \begin{cases} (y, x), & \text{if } (x, y) \in E, \\ (x, y), & \text{otherwise.} \end{cases} \tag{4.1}$$

In the first case we say that SHIFT *succeeds*, but in the second case it *fails* (we assume that SHIFT always succeeds if $x = y$).

Definition 4.2 (Search problems)**.** Under the restriction that only static transformations and SHIFT are allowed, consider the following problems:

- DETECT(G): Detect if there is a marked vertex in G;
- FIND(G): Find any marked vertex in G, with the promise that $M \neq \emptyset$.

Let us also define the following variations of the above problems:

- DETECT$^{(k)}(G)$: problem DETECT(G) with the promise that either $m = 0$ or $m = k$;
- FIND$^{(k)}(G)$: problem FIND(G) with the promise that $m = k$.

Similarly, let DETECT$^{(\geq k)}(G)$ and FIND$^{(\geq k)}(G)$ denote the corresponding problems with equality $m = k$ replaced by inequality $m \geq k$.

4.2.2 Random walks

A natural approach to searching on a graph consists in using a random walk. Intuitively, a random walk is an alternation of coin flips and shifts. More precisely, a coin is flipped according to the current state $x \in X$ of the vertex register, its value describes the target vertex y, and SHIFT performs a move from x to y. Let P_{xy} be the probability that x is shifted to y. Then SHIFT always succeeds if $P_{xy} = 0$ whenever $(x, y) \notin E$. In such case, we say that $P = (P_{xy})_{x,y \in X}$ is a *Markov chain on graph* G.

We assume from now on that P is an ergodic Markov chain (see Definition 2.1). Therefore, by the Perron–Frobenius Theorem, P has a unique stationary distribution π. We also assume that P is reversible: $\pi_x P_{xy} = \pi_y P_{yx}$, for all $x, y \in X$ (see Definition 2.7).

To measure the complexity of implementing a random walk corresponding to P, we introduce the following black-box operations:

- $\texttt{Check}(M)$: check if a given vertex is marked;
- $\texttt{Setup}(P)$: draw a sample from the stationary distribution π of P;
- $\texttt{Update}(P)$: perform one step of P.

Each of these black-box operations have the corresponding associated implementation cost, which we denote by C, S, and U, respectively.

4.2.3 Quantum walks

The setup in the quantum case is as follows. As in Chapter 3, the evolution will take place in space $\mathcal{H} \otimes \mathcal{H}$ where $\mathcal{H} := \text{span}\{|x\rangle : x \in X\}$ is the n-dimensional complex Euclidean space spanned by elements of set X. Let the first register be the *vertex register* that stores the current vertex of the walk. We call a unitary transformation *static* if it is controlled by this register, *i.e.*, it is of the form $\sum_{x \in X} |x\rangle\langle x| \otimes U_x$ for some unitaries U_x. The quantum version of the SHIFT operation is obtained by extending the expression in Definition 4.1 by linearity.

A *quantum walk* on G is a composition of static unitary transformations and SHIFT. In addition, we require that it respects the local structure of G, *i.e.*, whenever SHIFT is applied to a state, it must completely lie within the subspace of $\mathcal{H} \otimes \mathcal{H}$ where SHIFT is guaranteed to succeed.

As in Eq. (3.2) in Chapter 3, let $|\bar{0}\rangle \in \mathcal{H}$ be a fixed reference state, and let $V(P)$ be a

unitary operation that satisfies

$$V(P)|x\rangle|\bar{0}\rangle = |x\rangle|p_x\rangle = |x\rangle \sum_{y\in X} \sqrt{P_{xy}}|y\rangle \tag{4.2}$$

for all $x \in X$.

We will only consider quantum walks built from quantum analogues of reversible Markov chains, so we extend the operations `Check`, `Setup`, and `Update` to the quantum setting as follows.

- `Check`(M): map $|x\rangle|b\rangle$ to $|x\rangle|b\rangle$ if $x \notin M$ and $|x\rangle|b \oplus 1\rangle$ if $x \in M$, where $|x\rangle$ is the vertex register and $b \in \{0,1\}$;
- `Setup`(P): construct the superposition $|\pi\rangle = \sum_{x\in X} \sqrt{\pi_x}|x\rangle$;
- `Update`(P): apply any of $V(P)$, $V(P)^\dagger$, or SHIFT.

Implicitly, we also allow any controlled version of `Check`(M), `Setup`(P), and `Update`(P), on which we access via oracle.

In terms of the number of applications of SHIFT, `Update` has complexity 1 while `Setup` has complexity equal to the diameter of graph G. Nonetheless, in many algorithmic applications, the situation is more complex and the number of applications of SHIFT is not the only relevant cost; see for instance [Amb04, MSS05].

4.2.4 Classical hitting time

Let P be a Markov chain with a set of marked elements M. Recall from Definition 2.16 that the classical hitting time of P with respect to M, denoted by $\mathrm{HT}(P, M)$, is the expected number of applications of P required to hit a marked vertex in M. We assume that the initial vertex is distributed according to the stationary distribution π of P restricted to unmarked vertices. This definition is based on the **Random Walk Algorithm** from Sect. 2.4.1, which can be used to solve the DETECT and FIND problems classically.

Proposition 4.3. *Let $k \geq 1$. $\mathrm{DETECT}^{(\geq k)}(G)$ can be solved with high probability and classical complexity of order*

$$\mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C}), \quad \textit{where} \quad T = \max_{|M'|=k} \mathrm{HT}(P, M'). \tag{4.3}$$

$\mathrm{FIND}(G)$ *can be solved with high probability and expected classical complexity of order*

$$\mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C}), \quad \textit{where} \quad T = \mathrm{HT}(P, M). \tag{4.4}$$

4.2.5 Quantum hitting time

Quantum walks have been successfully used for detecting the presence of marked vertices quadratically faster than random walks. Nonetheless, only little is known on the problem of finding a marked vertex. Below we illustrate the state of the art.

Theorem 4.4 ([Sze04a]). *Let $k \geq 1$. $\textsc{Detect}^{(\geq k)}(G)$ can be solved with high probability and quantum complexity of order*

$$\mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C}), \quad \textit{where} \quad T = \max_{|M'|=k} \sqrt{\mathrm{HT}(P, M')}. \tag{4.5}$$

When P is state-transitive and there is a unique marked vertex z (i.e., $m = 1$), $\mathrm{HT}(P, \{z\})$ is independent of z and one can also find z:

Theorem 4.5 ([Tul08, MNRS12]). *Assume that P is state-transitive. $\textsc{Find}^{(1)}(G)$ can be solved with high probability and quantum complexity of order*

$$\mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C}), \quad \textit{where} \quad T = \sqrt{\mathrm{HT}(P, \{z\})}. \tag{4.6}$$

Using standard techniques, such as in [AA05], Theorem 4.5 can be generalized to any number of marked vertices, with an extra logarithmic multiplicative factor. Nonetheless, the complexities of the corresponding algorithms do not decrease when the size of M increases, contrary to the random walk search algorithm (Prop. 4.3) and the quantum walk detecting algorithm (Theorem 4.4).

Corollary 4.6. *Assume that P is state-transitive. $\textsc{Find}(G)$ can be solved with high probability and quantum complexity of order*

$$\log(n) \cdot \big(\mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C})\big), \quad \textit{where} \quad T = \sqrt{\mathrm{HT}(P, \{z\})}, \textit{ for any } z. \tag{4.7}$$

4.3 Discrete-time quantum walk

4.3.1 Szegedy's construction

We follow the construction of Szegedy [Sze04a] and define a quantum analogue of a reversible Markov chain P. Recall from Eq. (4.2) that $V(P)|x\rangle|\bar{0}\rangle = |x\rangle|p_x\rangle$ where $|\bar{0}\rangle$ is an arbitrary reference state in $\mathcal{H}$. Let $\mathcal{X} := \mathcal{H} \otimes \mathrm{span}\{|\bar{0}\rangle\} = \mathrm{span}\{|x\rangle|\bar{0}\rangle : x \in X\}$ and

$$\mathrm{ref}_{\mathcal{X}} := 2 \sum_{x \in X} |x\rangle\langle x| \otimes |\bar{0}\rangle\langle\bar{0}| - I = I \otimes (2|\bar{0}\rangle\langle\bar{0}| - I) \tag{4.8}$$

be the reflection in $\mathcal{H} \otimes \mathcal{H}$ with respect to the subspace $\mathcal{X}$. The *quantum walk operator* corresponding to Markov chain P is[2]

$$W(P) := V(P)^\dagger \, \textsc{Shift} \, V(P) \cdot \mathrm{ref}_{\mathcal{X}}. \tag{4.9}$$

Notice that $W(P)$ requires 3 calls to $\texttt{Update}(P)$.

We will choose an initial state that lies in subspace $\mathcal{X}$. Thus, to simplify the analysis, we will always restrict the action of $W(P)$ to the smallest subspace that contains $\mathcal{X}$ and is invariant under $W(P)$. We call this subspace the *walk space* of $W(P)$. In Lemma 4.7 below we show that this subspace is spanned by $\mathcal{X}$ and $W(P)\mathcal{X}$, and that SHIFT is guaranteed to succeed when $W(P)$ is applied to a state in the walk space.

The goal of this section is to find the spectral decomposition of the quantum walk operator corresponding to $P(s)$:

$$W(s) := V(s)^\dagger \, \textsc{Shift} \, V(s) \cdot \mathrm{ref}_{\mathcal{X}} \tag{4.10}$$

where $V(s) := V(P(s))$. Recall from Sect. 2.3.2 that $\lambda_k(s)$ and $|v_k(s)\rangle$ are the eigenvalues and eigenvectors of the discriminant matrix $D(s)$ of $P(s)$. The following lemma by Szegedy [Sze04a] is a unitary equivalent of Lemma 3.3. It provides the spectral decomposition of $W(s)$ in terms of that of $D(s)$.

Lemma 4.7 ([Sze04a])**.** *Let $\mathcal{B}_k(s)$ for $k = 1, \ldots, n$ be the subspaces defined in Sect. 3.2.2, and let $\varphi_k(s) \in [0, \pi]$ be such that*

$$\lambda_k(s) = \cos \varphi_k(s). \tag{4.11}$$

Then $W(s)$ has the following eigenvalues and eigenvectors.

$$\text{On } \mathcal{B}_k(s): \qquad e^{\pm i\varphi_k(s)}, \qquad |\Psi_k^\pm(s)\rangle := \frac{|v_k(s), \bar{0}\rangle \pm i|v_k(s), \bar{0}\rangle^\perp}{\sqrt{2}}. \tag{4.12}$$

$$\text{On } \mathcal{B}_n(s): \qquad 1, \qquad |\Psi_n(s)\rangle := |v_n(s), \bar{0}\rangle. \tag{4.13}$$

In particular, $\bigcup_{k=1}^n \mathcal{B}_k(s)$ is the walk space of $W(s)$ and the remaining eigenvectors of $W(s)$ lie in the orthogonal complement $\mathcal{B}^\perp(s)$.

Proof. We would like to reuse results from Sect. 3.2.2. However, we have to argue that they still hold if we replace the swap operator $S : |x, y\rangle \mapsto |y, x\rangle$ with SHIFT, which is defined in Eq. (4.1) and respects the local structure of the graph.

[2] In [Sze04a] the quantum walk operator corresponding to P is defined as $\big(V(P)\, W(P)\, V(P)^\dagger\big)^2$ where $W(P)$ is defined in Eq. (4.9).

First, let us show that $V^\dagger(s)SV(s)$ and $V^\dagger(s)\,\textsc{Shift}\,V(s)$ act in the same way on subspace $\mathcal{X}$. For any $x \in X$ we have

$$V^\dagger(s)SV(s)\cdot|x,\bar{0}\rangle = V^\dagger(s)S\sum_{y\in X}\sqrt{P_{xy}(s)}|x,y\rangle. \tag{4.14}$$

Since $P(s)$ is a Markov chain on G, $P_{xy}(s) = 0$ when xy is not an edge of G, so the resulting state is the same if we replace S by SHIFT.

According to this observation, we can rewrite Eqs. (3.12) and (3.16) as

$$V^\dagger(s)\,\textsc{Shift}\,V(s)\cdot|v_k(s),\bar{0}\rangle \quad = \lambda_k(s)|v_k(s),\bar{0}\rangle + \sqrt{1-\lambda_k(s)^2}|v_k(s),\bar{0}\rangle^\perp, \tag{4.15}$$
$$V^\dagger(s)\,\textsc{Shift}\,V(s)\cdot|v_k(s),\bar{0}\rangle^\perp = \sqrt{1-\lambda_k(s)^2}|v_k(s),\bar{0}\rangle - \lambda_k(s)|v_k(s),\bar{0}\rangle^\perp. \tag{4.16}$$

Clearly, $\mathrm{ref}_{\mathcal{X}}|v_k(s),\bar{0}\rangle = |v_k(s),\bar{0}\rangle$, and recall from Eq. (3.19) that $\Pi_0|v_k(s),\bar{0}\rangle^\perp = 0$, so $\mathrm{ref}_{\mathcal{X}}|v_k(s),\bar{0}\rangle^\perp = -|v_k(s),\bar{0}\rangle^\perp$. Thus, Eqs. (4.15) and (4.16) give us

$$W(s)\cdot|v_k(s),\bar{0}\rangle \quad = \lambda_k(s)|v_k(s),\bar{0}\rangle + \sqrt{1-\lambda_k(s)^2}|v_k(s),\bar{0}\rangle^\perp, \tag{4.17}$$
$$W(s)\cdot|v_k(s),\bar{0}\rangle^\perp = -\sqrt{1-\lambda_k(s)^2}|v_k(s),\bar{0}\rangle + \lambda_k(s)|v_k(s),\bar{0}\rangle^\perp. \tag{4.18}$$

Recall from Prop. 3.2 that subspaces $\mathcal{B}_k(s)$ are mutually orthogonal. From Eqs. (4.17) and (4.18) we also see that they are invariant under $W(s)$. In fact, $W(s)$ acts as

$$\begin{pmatrix} \lambda_k(s) & -\sqrt{1-\lambda_k(s)^2} \\ \sqrt{1-\lambda_k(s)^2} & \lambda_k(s) \end{pmatrix} \tag{4.19}$$

in the basis $\{|v_k(s),\bar{0}\rangle, |v_k(s),\bar{0}\rangle^\perp\}$ of $\mathcal{B}_k(s)$. This is an orthogonal matrix whose eigenvalues are

$$\lambda_k(s) \pm i\sqrt{1-\lambda_k(s)^2} = e^{\pm i\varphi_k(s)}. \tag{4.20}$$

Finally, according to Prop. 3.1,

$$\langle v_n(s),\bar{0}|\cdot V^\dagger(s)\,\textsc{Shift}\,V(s)\cdot|v_n(s),\bar{0}\rangle = 1, \tag{4.21}$$

so $|v_n(s),\bar{0}\rangle$ is an eigenvector of $W(s)$ with eigenvalue 1. □

4.3.2 Quantum circuit for $W(s)$

Recall that $\texttt{Update}(P)$ can be used to implement the quantum walk operator $W(P)$. However, we would also like to be able to implement the quantum analogue of $P(s)$ for any $s \in [0,1]$. Recall from Eq. (4.10) that it is given by

$$W(s) = V(s)^\dagger \, \textsc{Shift} \, V(s) \cdot \mathrm{ref}_{\mathcal{X}}. \tag{4.22}$$

We know how to implement SHIFT and $\mathrm{ref}_{\mathcal{X}}$, so we only need to understand how to implement $V(s)$ using $V(P)$. Recall from Eq. (4.2) that

$$V(s)|x\rangle|\bar{0}\rangle = |x\rangle|p_x(s)\rangle = |x\rangle \sum_{y \in X} \sqrt{P_{xy}(s)}|y\rangle. \tag{4.23}$$

In the following lemma, we assume that we know p_{xx} for every x. This is reasonable since in practice the probability of self-loops is known. In many cases, it is even independent of x. For the rest of this chapter, we assume that this is not an obstacle (we can assume that one call to $\texttt{Update}(P)$ allows to learn p_{xx} for any x).

Lemma 4.8. *Assuming that p_{xx} is known for every x, **Interpolation**(P, M, s) implements $V(s)$ with quantum complexity $2\mathsf{C} + \mathsf{U}$. Thus, $\texttt{Update}(P(s))$ has quantum complexity of order $\mathsf{C} + \mathsf{U}$.*

Proof. We explain only how to implement $V(s)$ using one call to $V(P)$ and two calls to $\texttt{Check}(M)$. The algorithm for $V(s)^\dagger$ is obtained from the reverse algorithm.

Our algorithm uses four registers: R_1, R_2, R_3, R_4. The first two registers have underlying state space $\mathcal{H}$ each, but the last two store a qubit in $\mathbb{C}^2$ each. Register R_3 is used to store if the current vertex x is marked, but R_4 is used for performing rotations. Let

$$R_\alpha := \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{4.24}$$

denote the rotation by angle α. An algorithm for implementing the transformation $|x\rangle|\bar{0}\rangle \mapsto |x\rangle|p_x(s)\rangle$ is given below.

Interpolation(P, M, s)

1. Let the initial state be $|x\rangle|\bar{0}\rangle|0\rangle|0\rangle$.
2. Apply $\texttt{Check}(M)$ to $\mathsf{R}_1\mathsf{R}_3$ (then $\mathsf{R}_3 = 1$ if and only if $x \in M$).
3. If $\mathsf{R}_3 = 0$, apply $V(P)$ to $\mathsf{R}_1\mathsf{R}_2$ and get $|x\rangle|p_x\rangle|0\rangle|0\rangle$.
4. Otherwise:
   (a) The state is $|x\rangle|\bar{0}\rangle|1\rangle|0\rangle$ where $x \in M$.
   (b) Apply R_α with $\alpha = \arcsin\sqrt{s}$ on R_4: $|x\rangle|\bar{0}\rangle|1\rangle(\sqrt{1-s}|0\rangle + \sqrt{s}|1\rangle)$.
   (c) If $\mathsf{R}_4 = 0$, apply $V(P)$ on $\mathsf{R}_1\mathsf{R}_2$. Otherwise, use CNOT to copy R_1 to R_2 in the standard basis: $|x\rangle(\sqrt{1-s}|p_x\rangle|1\rangle|0\rangle + \sqrt{s}|x\rangle|1\rangle|1\rangle)$.
   (d) If $\mathsf{R}_1 = \mathsf{R}_2$, apply R_α with $\alpha = -\arcsin\sqrt{s/((1-s)P_{xx}+s)}$ to R_4. Otherwise, do nothing: $|x\rangle|p_x(s)\rangle|1\rangle|0\rangle$.
5. Apply $\texttt{Check}(M)$ to $\mathsf{R}_1\mathsf{R}_3$ to uncompute R_3 and get $|x\rangle|p_x(s)\rangle|0\rangle|0\rangle$.

Recall from Eq. (2.3) that $P(s)$ has the following block structure:

$$P(s) = \begin{pmatrix} P_{UU} & P_{UM} \\ (1-s)P_{MU} & (1-s)P_{MM} + sI \end{pmatrix}. \tag{4.25}$$

We will analyze the cases $x \in M$ and $x \in U$ separately. Then the general case will hold by linearity.

If $x \in U$ then the corresponding row of $P(s)$ does not depend on s, so $|p_x(s)\rangle = |p_x\rangle$. In this case step 4 of the above algorithm is never executed and the remaining steps effectively apply $V(P)$ to produce the correct state.

When $x \in M$ the algorithm is more involved. Let us analyze only step 4 where most of the work is done. During this step the state gets transformed as follows:

$$\begin{aligned} |x\rangle|\bar{0}\rangle|1\rangle|0\rangle &\mapsto |x\rangle|\bar{0}\rangle|1\rangle(\sqrt{1-s}|0\rangle + \sqrt{s}|1\rangle) && (4.26)\\ &\mapsto |x\rangle\big(\sqrt{1-s}|p_x\rangle|1\rangle|0\rangle + \sqrt{s}|x\rangle|1\rangle|1\rangle\big) && (4.27)\\ &\mapsto |x\rangle|p_x(s)\rangle|1\rangle|0\rangle. && (4.28) \end{aligned}$$

The first two transformations are straightforward, so let us focus only on the last one which corresponds to step 4d. The state at the beginning of this step is

$$\begin{aligned} &|x\rangle\big(\sqrt{1-s}|p_x\rangle|1\rangle|0\rangle + \sqrt{s}|x\rangle|1\rangle|1\rangle\big) && (4.29)\\ &= |x\rangle\left[\sqrt{1-s}\sum_{y\in X\setminus\{x\}}\sqrt{P_{xy}}|y\rangle|1\rangle|0\rangle + |x\rangle|1\rangle\Big(\sqrt{(1-s)P_{xx}}|0\rangle + \sqrt{s}|1\rangle\Big)\right]. && (4.30) \end{aligned}$$

Note from the second row of matrix $P(s)$ in Eq. (4.25) that all its elements have acquired a factor of $1-s$, except the diagonal ones. Thus in step 4d we perform a rotation only when $\mathsf{R}_1 = \mathsf{R}_2$. This rotation affects only the second half of the state in Eq. (4.30) and transfers all amplitude to $|0\rangle$ in the last register:

$$|x\rangle\left[\sqrt{1-s}\sum_{y\in X\setminus\{x\}}\sqrt{P_{xy}}|y\rangle+\sqrt{(1-s)P_{xx}+s}|x\rangle\right]|1\rangle|0\rangle=|x\rangle|p_x(s)\rangle|1\rangle|0\rangle. \tag{4.31}$$

Finally, step 5 uncomputes R_3 to $|0\rangle$ and the final state is $|x\rangle|p_x(s)\rangle|0\rangle|0\rangle$ as desired. □

4.4 Quantum search algorithms

In this section we provide several quantum search algorithms. They are all based on a procedure known as *eigenvalue estimation* and essentially run it different numbers of times with different values of parameters. Here is a formal statement of what eigenvalue estimation does.

Theorem 4.9 (Eigenvalue estimation [Kit95, CEMM98])**.** *For any unitary operator W and precision $t \in \mathbb{N}$, there exists a quantum circuit* ***Eigenvalue Estimation***(W,t) *that uses 2^t calls to the controlled-W operator and $O(t^2)$ additional gates, and acts on eigenstates $|\Psi_k\rangle$ of W as*

$$|\Psi_k\rangle \mapsto |\Psi_k\rangle\frac{1}{2^t}\sum_{l,m=0}^{2^t-1} e^{-\frac{2\pi i l m}{2^t}} e^{i\varphi_k l}|m\rangle, \tag{4.32}$$

where $e^{i\varphi_k}$ is the eigenvalue of W corresponding to $|\Psi_k\rangle$.

By linearity, **Eigenvalue Estimation**(W,t) resolves any state as a linear combination of the eigenstates of W and attaches to each term a second register holding an approximation of the first t bits of the binary decomposition of $\frac{1}{2\pi}\varphi_k$, where φ_k is the phase of the corresponding eigenvalue. We will be mostly interested in the component along the eigenvector $|\Psi_n\rangle$ which corresponds to phase $\varphi_n = 0$. In that case, the second register is in the state $|0^t\rangle$ and the estimation is exact.

Our search algorithms will be based on **Eigenvalue Estimation**$(W(s),t)$ for some values of parameters s and t. The value of the interpolation parameter $s \in [0,1]$ will be related to p_M, the probability to pick a marked vertex from the stationary distribution

Result	p_M	$\mathrm{HT}(P, M)$
Theorem 4.10	known	known
Theorem 4.12	approximation known	known
Theorem 4.13	approximation known	not known
Theorem 4.14	bound known	bound known
Theorem 4.15	not known	bound known

Table 4.1: Summary of results on quantum search algorithms. Assumptions on p_M and $\mathrm{HT}(P, M)$ are listed in the last two columns.

π of P. Precision $t \in \mathbb{N}$, or the number of binary digits in eigenvalue estimation, will be related to $\mathrm{HT}(P, M)$, the hitting time of P.

We consider several scenarios where different knowledge of the values of parameters p_M and $\mathrm{HT}(P, M)$ is available, and for each case we provide an algorithm. The list of all results and the corresponding assumptions is given in Table 4.1.

4.4.1 Algorithm with known values of p_M and $\mathrm{HT}(P, M)$

For simplicity, let us first assume that the values of p_M and $\mathrm{HT}(P, M)$ are known. In this case we provide a quantum algorithm that solves $\textsc{Find}(G)$ (*i.e.*, outputs a marked vertex if there is any) with success probability and running time that depends on two parameters ε_1 and ε_2.

Let us first recall how the classical **Random Walk Algorithm** from Sect. 2.4.1 works. It starts with the stationary distribution π of P and applies the absorbing walk P' until most of the probability is absorbed in marked vertices and thus the state is close to a stationary distribution of P'.

In quantum case a natural starting state is $|\pi\rangle|\bar{0}\rangle = |v_n\rangle|\bar{0}\rangle = |\Psi_n(0)\rangle$, which is the stationary superposition of $W(P)$. By analogy, we would like to end in its projection onto marked vertices, namely $|M\rangle|\bar{0}\rangle$, which is also the stationary superposition of $W(P')$. However, at this point the analogy breaks down, since we do not want to apply $W(P')$ to reach the final state. The reason is that in many cases, including the 2D grid, every iteration of $W(P')$ on $|\pi\rangle|\bar{0}\rangle$ may remain far from $|M\rangle|\bar{0}\rangle$. Instead, our approach consists in quantizing a new random walk, namely an interpolation $P(s)$ between P and P'. This technique is drastically different from the approach of [Tul08, MNRS12], and up to our knowledge new.

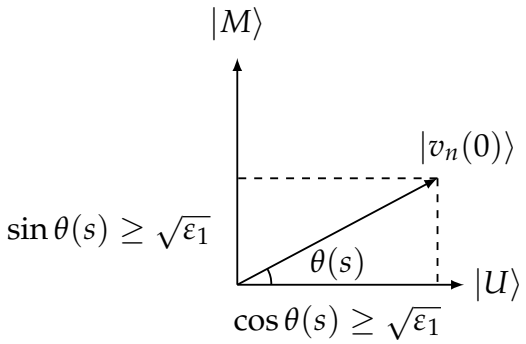

Figure 4.1: Vectors $|U\rangle$, $|M\rangle$, and $|v_n(s)\rangle = \cos\theta(s)|U\rangle + \sin\theta(s)|M\rangle$. We want to choose s so that $\langle U|v_n(s)\rangle = \cos\theta(s) \geq \sqrt{\varepsilon_1}$ and $\langle M|v_n(s)\rangle = \sin\theta(s) \geq \sqrt{\varepsilon_1}$.

Intuitively, our quantum algorithm works as follows. We fix some value of $s \in [0,1]$ and map $|U\rangle$ to $|v_n(s)\rangle$ using a quantum walk based on $P(s)$, and then measure $|v_n(s)\rangle$ in the standard basis to get a marked vertex. For this to work with a good probability of success, we have to choose the interpolation parameter s so that $|v_n(s)\rangle$ has a large overlap on both $|U\rangle$ and $|M\rangle$ (see Fig. 4.1). Recall from Prop. 2.13 that $|v_n(s)\rangle = \cos\theta(s)|U\rangle + \sin\theta(s)|M\rangle$, so we will demand that $\cos\theta(s)\sin\theta(s) \geq \varepsilon_1$ for some parameter ε_1. The second parameter ε_2 controls the precision of phase estimation.

Theorem 4.10. *Assume that the values of p_M and $\mathrm{HT}(P,M)$ are known, and let $s \in [0,1]$, $T \geq 1$, and $\frac{1}{2} \geq \varepsilon_1 \geq \varepsilon_2 \geq 0$ be some parameters. If*

$$\cos\theta(s)\sin\theta(s) \geq \varepsilon_1 \qquad \text{and} \qquad T \geq \frac{\pi}{\sqrt{2}\varepsilon_2}\sqrt{\mathrm{HT}(s)} \tag{4.33}$$

then ***Search***$(P, M, s, \lceil \log T \rceil)$ *solves* $\textsc{Find}(G)$ *with success probability at least*

$$p_M + (1 - p_M)(\varepsilon_1 - \varepsilon_2)^2 \tag{4.34}$$

and complexity of order $\mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C})$.

Proof. Let $t = \lceil \log T \rceil$ be the precision in the eigenvalue estimation. Our algorithm uses two registers: R_1 and R_2 with underlying state space $\mathcal{H}$ each. Occasionally we will attach the third register R_3 initialized in $|0\rangle \in \mathbb{C}^2$ to check if the current vertex is marked.

Search(P, M, s, t)

1. Prepare the state $|\pi\rangle|\bar{0}\rangle$.
2. Attach R_3, apply $\texttt{Check}(M)$ to $\mathsf{R}_1\mathsf{R}_3$, and measure R_3.
3. If $\mathsf{R}_3 = 1$, measure R_1 (in the vertex basis) and output the outcome.
4. Otherwise, discard R_3 and:
   (a) Apply **Eigenvalue Estimation**$(W(s), t)$ on $\mathsf{R}_1\mathsf{R}_2$.
   (b) Attach R_3, apply $\texttt{Check}(M)$ to $\mathsf{R}_1\mathsf{R}_3$, and measure R_3.
   (c) If $\mathsf{R}_3 = 1$, measure R_1 (in the vertex basis) and output the outcome. Otherwise, output: `No marked vertex`.

Notice that step 1 has complexity S, but **Eigenvalue Estimation**$(W(s), t)$ in step 4a has complexity of the order $2^t \cdot (\mathsf{U} + \mathsf{C})$ according to Theorem 4.9 and Lemma 4.8. Thus, the total complexity is of the order $\mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C})$, and it only remains to bound the success probability.

Observe that the overall success probability is of the form $p_M + (1 - p_M)q$ where q is the probability to find a marked vertex in step 4. Thus, it remains to show that $q \geq (\varepsilon_1 - \varepsilon_2)^2$.

We assume that **Search**(P, M, s, t) reaches step 4a, otherwise a marked vertex is already found. At this point the state is $|U\rangle|\bar{0}\rangle$. Let us expand the first register of this state in the eigenbasis of the discriminant matrix $D(s)$. From now on we will omit the explicit dependence on s when there is no ambiguity. Let

$$\alpha_k := \langle v_k | U \rangle \tag{4.35}$$

and observe from Eq. (4.12) that $|v_k\rangle|\bar{0}\rangle = \frac{1}{\sqrt{2}}(|\Psi_k^+\rangle + |\Psi_k^-\rangle)$. Then

$$|U\rangle|\bar{0}\rangle = \alpha_n |v_n\rangle|\bar{0}\rangle + \sum_{k \neq n} \alpha_k |v_k\rangle|\bar{0}\rangle = \alpha_n |\Psi_n\rangle + \frac{1}{\sqrt{2}} \sum_{k \neq n} \alpha_k \big(|\Psi_k^+\rangle + |\Psi_k^-\rangle\big). \tag{4.36}$$

Recall from Lemma 4.7 that the eigenvalues corresponding to $|\Psi_n\rangle$ and $|\Psi_k^\pm\rangle$ are 1 and $e^{\pm i\varphi_k}$, respectively. From Eq. (4.32) we see that **Eigenvalue Estimation**$(W(s), t)$ in step 4a acts as follows:

$$|\Psi_n\rangle \mapsto |\Psi_n\rangle|0^t\rangle, \tag{4.37}$$

$$|\Psi_k^\pm\rangle \mapsto |\Psi_k^\pm\rangle|\xi_k^\pm\rangle, \tag{4.38}$$

where $|\xi_k^{\pm}\rangle$ is a t-qubit state that satisfies

$$\langle 0^t|\xi_k^{\pm}\rangle = \frac{1}{2^t}\sum_{l=0}^{2^t-1} e^{\pm i\varphi_k l} =: \delta_k^{\pm}. \tag{4.39}$$

Thus, the state after eigenvalue estimation lies in $\mathcal{H}\otimes\mathcal{H}\otimes\mathbb{C}^{2^t}$ and is equal to

$$|\Phi\rangle := \alpha_n|\Psi_n\rangle|0^t\rangle + \frac{1}{\sqrt{2}}\sum_{k\neq n}\alpha_k\big(|\Psi_k^+\rangle|\xi_k^+\rangle + |\Psi_k^-\rangle|\xi_k^-\rangle\big). \tag{4.40}$$

Recall that q denotes the probability to obtain a marked vertex by measuring the first register of $|\Phi\rangle$ in step 4c. To lower bound q, we in addition require that the last register of $|\Phi\rangle$ is in the state $|0^t\rangle$ (*i.e.*, the phase is estimated to be 0). Then

$$\begin{aligned}
\sqrt{q} &= \|(\Pi_M\otimes I\otimes I)|\Phi\rangle\| && (4.41)\\
&\geq \|(\Pi_M\otimes I\otimes|0^t\rangle\langle 0^t|)|\Phi\rangle\| && (4.42)\\
&\geq \|\alpha_n(\Pi_M\otimes I)|\Psi_n\rangle\| - \frac{1}{\sqrt{2}}\Big\|(\Pi_M\otimes I)\sum_{k\neq n}\alpha_k\big(\delta_k^+|\Psi_k^+\rangle + \delta_k^-|\Psi_k^-\rangle\big)\Big\| && (4.43)\\
&\geq \|\alpha_n(\Pi_M\otimes I)|\Psi_n\rangle\| - \frac{1}{\sqrt{2}}\Big\|\sum_{k\neq n}\alpha_k\big(\delta_k^+|\Psi_k^+\rangle + \delta_k^-|\Psi_k^-\rangle\big)\Big\|. && (4.44)
\end{aligned}$$

Recall from Eq. (4.13) and Prop. 2.13 that $|\Psi_n\rangle = |v_n\rangle|\bar{0}\rangle = (\cos\theta|U\rangle + \sin\theta|M\rangle)|\bar{0}\rangle$. Hence, we find that $\alpha_n = \langle v_n|U\rangle = \cos\theta$ and $\|(\Pi_M\otimes I)|\Psi_n\rangle\| = \sin\theta$. Moreover, recall from Lemma 4.7 that vectors $|\Psi_1^{\pm}\rangle,\ldots,|\Psi_k^{\pm}\rangle$ are mutually orthogonal. Thus we can simplify Eq. (4.44) as follows:

$$\sqrt{q} \geq \cos\theta\sin\theta - \sqrt{\sum_{k\neq n}|\alpha_k|^2\delta_k^2} \tag{4.45}$$

where $\delta_k := |\delta_k^+| = |\delta_k^-|$ (note from Eq. (4.39) that δ_k^+ and δ_k^- are complex conjuagtes). Now we will bound the second term in Eq. (4.45).

Let us compute the sum of the geometric series in Eq. (4.39):

$$\delta_k^2 = \left|\frac{1}{2^t}\sum_{l=0}^{2^t-1}e^{i\varphi_k l}\right|^2 = \frac{1}{2^{2t}}\left|\frac{1-e^{i\varphi_k 2^t}}{1-e^{i\varphi_k}}\right|^2 = \frac{1}{2^{2t}}\left|\frac{e^{-i\frac{\varphi_k}{2}2^t} - e^{i\frac{\varphi_k}{2}2^t}}{e^{-i\frac{\varphi_k}{2}} - e^{i\frac{\varphi_k}{2}}}\right|^2 = \frac{\sin^2(\frac{\varphi_k}{2}2^t)}{2^{2t}\sin^2(\frac{\varphi_k}{2})}. \tag{4.46}$$

We can upper bound the numerator in the final expression by one. To bound the denominator, we use $\sin \frac{x}{2} \geq \frac{x}{\pi}$ for $x \in [0, \pi]$. Hence, we get

$$\delta_k^2 \leq \frac{\pi^2}{2^{2t}\varphi_k^2} \leq \frac{\pi^2}{T^2\varphi_k^2} \tag{4.47}$$

since we chose $t = \lceil \log T \rceil$.

Recall from Prop. 2.18 that the extended hitting time is given by

$$\mathrm{HT}(s) = \sum_{k:\, \lambda_k(s) \neq 1} \frac{|\langle v_k(s)|U\rangle|^2}{1 - \lambda_k(s)}. \tag{4.48}$$

If we substitute $\langle v_k(s)|U\rangle = \alpha_k(s)$ and $\lambda_k(s) = \cos\varphi_k(s)$ from Eqs. (4.39) and (4.11), and omit the dependence on s, we get

$$\mathrm{HT}(s) = \sum_{k\neq n} \frac{|\alpha_k|^2}{1 - \cos\varphi_k} = \sum_{k\neq n} \frac{|\alpha_k|^2}{2\sin^2(\frac{\varphi_k}{2})} \geq 2\sum_{k\neq n} \frac{|\alpha_k|^2}{\varphi_k^2} \tag{4.49}$$

since $x \geq \sin x$ for $x \in [0, \pi]$.

By combining Eqs. (4.47) and (4.49) we get

$$\sum_{k\neq n} |\alpha_k|^2 \delta_k^2 \leq \sum_{k\neq n} |\alpha_k|^2 \frac{\pi^2}{T^2\varphi_k^2} = \frac{\pi^2}{T^2} \sum_{k\neq n} \frac{|\alpha_k|^2}{\varphi_k^2} \leq \frac{\pi^2}{2}\frac{\mathrm{HT}(s)}{T^2}. \tag{4.50}$$

Thus, Eq. (4.45) becomes

$$\sqrt{q} \geq \cos\theta(s)\sin\theta(s) - \frac{\pi}{\sqrt{2}}\frac{\sqrt{\mathrm{HT}(s)}}{T} \geq \varepsilon_1 - \varepsilon_2, \tag{4.51}$$

where the last inequality follows from our assumptions. Thus $q \geq (\varepsilon_1 - \varepsilon_2)^2$. □

4.4.2 Algorithms with approximately known p_M

In this section we show that a good approximation p^* of p_M suffices to guarantee that the constraint $\cos\theta(s)\sin\theta(s) \geq \varepsilon_1$ in Theorem 4.10 is satisfied. Our strategy is to make a specific choice of the interpolation parameter s, based on p^*.

Intuitively, we want to choose s so that $\cos\theta(s)\sin\theta(s)$ is large (recall Fig. 4.1), since this will increase the success probability according to Eq. (4.51), and make it easier to satisfy the constraint on ε_1 in Theorem 4.10. The maximal value of $\cos\theta(s)\sin\theta(s)$ is achieved when $\sin\theta(s) = \cos\theta(s) = 1/\sqrt{2}$, and from Eq. (2.41) we get that the optimal value of s as a function of p_M is

$$s(p_M) := 1 - \frac{p_M}{1 - p_M}. \tag{4.52}$$

Thus, when only an approximation p^* of p_M is known, we will choose the interpolation parameter to be

$$s^* := s(p^*) = 1 - \frac{p^*}{1 - p^*}. \tag{4.53}$$

Since we want $s^* \geq 0$, we have to always make sure that $p^* \leq 1/2$. In fact, from now we will also assume that $p_M \leq 1/2$. This is without loss of generality, since one can always prepare the initial state $|\pi\rangle$ at cost S and measure it in the standard basis. If $p_M \geq 1/2$, this yields a marked vertex with probability at least $1/2$.

Proposition 4.11. *If $p_M, \varepsilon_1 \in [0, \frac{1}{2}]$ and p^* satisfy*

$$2\varepsilon_1 p_M \leq p^* \leq 2(1 - \varepsilon_1)p_M, \tag{4.54}$$

then $\cos\theta(s^*)\sin\theta(s^*) \geq \varepsilon_1$ where $s^* := 1 - \frac{p^*}{1-p^*}$.

Proof. To get the desired result, we will show that the two inequalities in Eq. (4.54) imply that $\cos^2\theta(s^*) \geq \varepsilon_1$ and $\sin^2\theta(s^*) \geq \varepsilon_1$, respectively, where

$$\cos^2\theta(s^*) = \frac{(1 - p_M)p^*}{p_M + p^* - 2p_M p^*}, \qquad \sin^2\theta(s^*) = \frac{p_M(1 - p^*)}{p_M + p^* - 2p_M p^*} \tag{4.55}$$

according to Eq. (2.41).

From Eq. (4.55), we have $\sin^2\theta(s^*) \geq \varepsilon_1$ if and only if

$$p^* \leq \frac{(1 - \varepsilon_1)p_M}{\varepsilon_1 + p_M - 2\varepsilon_1 p_M}. \tag{4.56}$$

Since $p_M, \varepsilon_1 \leq 1/2$, the denominator is upper bounded as

$$\varepsilon_1 + (1 - 2\varepsilon_1)p_M \leq \varepsilon_1 + \frac{1 - 2\varepsilon_1}{2} = \frac{1}{2}. \tag{4.57}$$

Therefore, $p^* \leq 2(1-\varepsilon_1)p_M$ implies Eq. (4.56), which is equivalent to $\sin^2\theta(s^*) \geq \varepsilon_1$.

Similarly from Eq. (4.55) we have $\cos^2\theta(s^*) \geq \varepsilon_1$ if and only if

$$p^* \geq \frac{\varepsilon_1 p_M}{1-\varepsilon_1-p_M+2\varepsilon_1 p_M}, \tag{4.58}$$

where the denominator is lower bounded as

$$1-\varepsilon_1-(1-2\varepsilon_1)p_M \geq 1-\varepsilon_1-\frac{1-2\varepsilon_1}{2} = \frac{1}{2}. \tag{4.59}$$

Therefore, $p^* \geq 2\varepsilon_1 p_M$ implies Eq. (4.58), which is equivalent to $\cos^2\theta(s^*) \geq \varepsilon_1$. □

4.4.2.1 Known $\mathrm{HT}(P,M)$

Now we will use Prop. 4.11 to show how an approximation p^* of p_M can be used to make a specific choice of the parameters ε_1, ε_2, s, and T in Theorem 4.10, so that our quantum search algorithm succeeds with constant probability.

To be more specific, we assume that we have an approximation p^* of p_M such that

$$|p^*-p_M| \leq \frac{1}{3}p_M, \tag{4.60}$$

where the constant $1/3$ is an arbitrary choice. Notice that

$$\frac{1}{3}p_M \geq p^*-p_M \quad\Longleftrightarrow\quad \frac{4}{3}p_M \geq p^*, \tag{4.61}$$

$$\frac{1}{3}p_M \geq p_M-p^* \quad\Longleftrightarrow\quad p^* \geq \frac{2}{3}p_M, \tag{4.62}$$

so Eq. (4.60) is equivalent to

$$\frac{2}{3}p_M \leq p^* \leq \frac{4}{3}p_M. \tag{4.63}$$

Theorem 4.12. *Assume that an approximation p^* of p_M such that $|p^*-p_M| \leq p_M/3$ and the value of $\mathrm{HT}(P,M)$ is known. If $T \geq 14\sqrt{\mathrm{HT}(P,M)}$ then **Search**$(P,M,s^*,\lceil\log T\rceil)$ solves $\textsc{Find}(G)$ with probability at least $1/36$ and complexity of order $\mathsf{S}+T\cdot(\mathsf{U}+\mathsf{C})$.*

Proof. We are given p^* that satisfies Eq. (4.63). If we choose $\varepsilon_1 := 1/3$, then this is equivalent to Eq. (4.54). Without loss of generality $p_M \leq 1/2$, so from Prop. 4.11 we get that $\cos\theta(s^*)\sin\theta(s^*) \geq \varepsilon_1$. Thus, the first condition in Eq. (4.33) of Theorem 4.10 is satisfied.

Next, we choose $\varepsilon_2 := 1/6$ somewhat arbitrarily. Recall from Theorem 2.22 that $\mathrm{HT}(s^*) \leq \mathrm{HT}(P, M)$, thus

$$\frac{\pi}{\sqrt{2}}\frac{1}{\varepsilon_2}\sqrt{\mathrm{HT}(s^*)} \leq \pi 3\sqrt{2}\sqrt{\mathrm{HT}(P,M)} \leq 14\sqrt{\mathrm{HT}(P,M)} \leq T, \tag{4.64}$$

so the second condition in Eq. (4.33) is also satisfied.

Hence, according to Theorem 4.10, **Search**$(P, M, s^*, \lceil \log T \rceil)$ solves FIND(G) with success probability at least

$$p_M + (1-p_M)(\varepsilon_1-\varepsilon_2)^2 \geq (\varepsilon_1-\varepsilon_2)^2 = \left(\frac{1}{3}-\frac{1}{6}\right)^2 = \frac{1}{36} \tag{4.65}$$

and complexity of order $\mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C})$. □

4.4.2.2 Unknown $\mathrm{HT}(P, M)$

Recall from Theorem 4.12 in previous section that a marked vertex can be found if p^*, an approximation of p_M, and $\mathrm{HT}(P, M)$ are known. In this section we show that a marked vertex can still be found (with essentially the same expected complexity), even if the requirement to know $\mathrm{HT}(P, M)$ is relaxed.

Theorem 4.13. *Given p^* such that $|p^* - p_M| \leq p_M/3$, **Incremental Search**$(P, M, s^*, 50)$ solves* FIND(G) *with expected quantum complexity of order*

$$\log(T) \cdot \mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C}), \quad \textit{where} \quad T = \sqrt{\mathrm{HT}(P,M)}. \tag{4.66}$$

Proof. The idea is to repeatedly use **Search**(P, M, s^*, t) with increasing accuracy of the eigenvalue estimation. We start with $t = 1$ and in every iteration increase it by one. Once t is above some threshold t_0, any subsequent iteration outputs a marked element with probability that is at least a certain constant. To boost the success probability of the **Search**(P, M, s^*, t) subroutine, for each value of t we call it $k = 50$ times.

Incremental Search(P, M, s^*, k)

1. Let $t = 1$.
2. Call k times **Search**(P, M, s^*, t).
3. If no marked vertex is found, set $t \leftarrow t + 1$ and go back to step 2.

Let t_0 be the smallest integer that satisfies

$$14\sqrt{\mathrm{HT}(P, M)} \leq 2^{t_0}. \tag{4.67}$$

Assume that variable t has reached value $t \geq t_0$, but **Incremental Search**$(P, M, s^*, 50)$ has not terminated yet. By Theorem 4.12, each execution of **Search**(P, M, s^*, t) outputs a marked vertex with probability at least $1/36$. Let p_{fail} be the probability that none of the $k = 50$ executions in step 2 succeeds. Notice that

$$p_{\text{fail}} \leq (1 - 1/36)^{50} \leq 1/4. \tag{4.68}$$

Let us assume that **Incremental Search**$(P, M, s^*, 50)$ terminates with the final value of t equal to t_f. Recall from Theorem 4.10 that **Search**(P, M, s^*, t) has complexity of order $\mathsf{S} + 2^t \cdot (\mathsf{U} + \mathsf{C})$, so the expected complexity of **Incremental Search**$(P, M, s^*, 50)$ is of order

$$N_1 \cdot \mathsf{S} + N_2 \cdot (\mathsf{U} + \mathsf{C}), \tag{4.69}$$

where N_1 is the expectation of t_f, and N_2 is the expectation of $2 + 4 + \cdots + 2^{t_f}$.

To upper bound N_1, we assume that the first $t_0 - 1$ iterations fail. Since each of the remaining iterations fails with probability at most p_{fail}, we get

$$N_1 \leq (t_0 - 1) + \sum_{t=t_0}^{\infty} p_{\text{fail}}^{1+(t-t_0)} \tag{4.70}$$

$$= (t_0 - 1) + \frac{p_{\text{fail}}}{1 - p_{\text{fail}}} \tag{4.71}$$

$$\leq (t_0 - 1) + \frac{1/4}{3/4} \tag{4.72}$$

$$\leq t_0. \tag{4.73}$$

We use the same strategy to upper bound N_2:

$$N_2 \leq \sum_{t=1}^{t_0-1} 2^t + \sum_{t=t_0}^{\infty} p_{\text{fail}}^{1+(t-t_0)} 2^t \tag{4.74}$$

$$= (2^{t_0} - 2) + p_{\text{fail}} \cdot \sum_{t=0}^{\infty} p_{\text{fail}}^t 2^{t+t_0} \tag{4.75}$$

$$\leq (2^{t_0} - 2) + \frac{1}{4} \cdot \sum_{t=0}^{\infty} \left(\frac{1}{4} \cdot 2\right)^t \cdot 2^{t_0} \tag{4.76}$$

$$= (2^{t_0} - 2) + \frac{1}{4} \cdot 2 \cdot 2^{t_0} \tag{4.77}$$

$$\leq 2 \cdot 2^{t_0}. \tag{4.78}$$

We plug the bounds on N_1 and N_2 in Eq. (4.69) and get that the expected complexity is of order $t_0 \cdot \mathsf{S} + 2^{t_0+1} \cdot (\mathsf{U} + \mathsf{C})$. Since t_0 satisfies Eq. (4.67), this concludes the proof. □

4.4.3 Algorithms with a given bound on p_M or $\mathrm{HT}(P, M)$

In previous section we considered the case when we know a *relative* approximation of p_M, *i.e.*, a value p^* such that $|p^* - p_M| \leq p_M/3$. In this section we consider the case when we are given an *absolute* lower bound $p_{\min}$ such that $p_{\min} \leq p_M$, an *absolute* upper bound $\mathrm{HT}_{\max} \geq \mathrm{HT}(P, M)$, or both. In particular, for problem $\textsc{Find}(G)^{(\geq k)}$ we can set $p_{\min} := \min_{M':|M'|=k} p_{M'}$ and $\mathrm{HT}_{\max} := \max_{M':|M'|=k} \mathrm{HT}(P, M')$.

4.4.3.1 Assuming a bound on p_M

Theorem 4.14. *Given $p_{\min}$ such that $p_{\min} \leq p_M$, $\textsc{Find}(G)$ can be solved with expected quantum complexity of order*

$$\sqrt{\log(1/p_{\min})} \cdot \left[\log(T) \cdot \mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C})\right], \quad \textit{where} \quad T = \sqrt{\mathrm{HT}(P, M)}. \tag{4.79}$$

Moreover, if $\mathrm{HT}_{\max}$ such that $\mathrm{HT}_{\max} \geq \mathrm{HT}(P, M)$ is also given, then $\textsc{Find}(G)$ can be solved with quantum complexity of order

$$\sqrt{\log(1/p_{\min})} \cdot \left[\mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C})\right], \quad \textit{where} \quad T = \sqrt{\mathrm{HT}_{\max}}. \tag{4.80}$$

Proof. We prove the first part of the theorem. The second part is similar except one has to use **Search**(P, M, s^*, T) instead of **Incremental Search**$(P, M, s^*, 50)$.

To apply Theorem 4.13, it is enough to obtain an approximation p^* of p_M such that $|p^* - p_M| \leq p_M/3$. Recall from Eq. (4.63) that this is equivalent to finding p^* such that

$$\frac{2}{3}p_M \leq p^* \leq \frac{4}{3}p_M. \tag{4.81}$$

Let l be the largest integer such that $p_M \leq 2^{-l}$. Then

$$\frac{1}{2} \cdot 2^{-l} \leq p_M \leq 2^{-l} \tag{4.82}$$

and hence

$$\frac{2}{3}p_M \leq \frac{2}{3} \cdot 2^{-l} = \frac{4}{3} \cdot \left(\frac{1}{2} \cdot 2^{-l}\right) \leq \frac{4}{3}p_M. \tag{4.83}$$

We can make sure that Eq. (4.81) is satisfied by choosing $p^* := \frac{2}{3} \cdot 2^{-l}$. Unfortunately, we do not know the value of l. However, we know that $p_{\min} \leq p_M$ and without loss of generality we can assume that $p_M \leq 1/2$. Thus, it only suffices to check all values of l from 1 to $\lfloor \log(1/p_{\min}) \rfloor$.

To find a marked vertex, we replace step 2 in the **Incremental Search** algorithm by a loop over the $\lfloor \log(1/p_{\min}) \rfloor$ possible values of p^*:

For $l = 1$ to $\lfloor \log(1/p_{\min}) \rfloor$ do:

- Let $p^* := \frac{2}{3} \cdot 2^{-l}$.
- Call k times **Search**$(P, M, s(p^*), t)$.

Recall from Theorem 4.10 that the complexity of **Search**(P, M, s^*, t) depends only on t. Hence, the analysis of the modified algorithm is the same, except that now the complexity of step 2 is multiplied by a factor of order $\log(1/p_{\min})$. In fact, this is the only non-trivial step of the **Incremental Search** algorithm, so the overall complexity increases by this multiplicative factor. Finally, note that instead of trying all possible values of p^*, we can search for the right value using Grover's algorithm, following the approach of [HMdW03], therefore reducing the multiplicative factor to $\sqrt{\log(1/p_{\min})}$. □

4.4.3.2 Assuming a bound on $\mathrm{HT}(P, M)$

Theorem 4.15. *Given* $\mathrm{HT}_{\max}$ *such that* $\mathrm{HT}_{\max} \geq \mathrm{HT}(P, M)$, $\textsc{Find}(G)$ *can be solved with expected quantum complexity of order*

$$\log(1/p_M) \cdot \big[\mathsf{S} + T \cdot (\mathsf{U} + \mathsf{C})\big], \quad \textit{where} \quad T = \sqrt{\mathrm{HT}_{\max}}. \tag{4.84}$$

Proof. We use **Search**(P, M, s^*, t) with $t = \left\lceil \log \sqrt{\mathrm{HT}_{\max}} \right\rceil$ and perform a dichotomic search for an appropriately chosen value of p^*. This dichotomic search uses backtracking, since the branching in the dichotomy is with bounded error, similarly to the situation in [FRFU94].

Initially we set $a = 0$ and $b = 1$. Then for testing the current value of $p^* = (a + b)/2$, we run **Search**$(P, M, s(p^*), t)$ a constant number of times. If a marked vertex is found we stop. Otherwise, if **Eigenvalue Estimation**$(W(s(p^*)), t)$ outputs a minority of 0s, we set $a = p^*$, otherwise we set $b = p^*$. The details of the analysis are given in [FRFU94]. □

4.4.4 Application to the 2D grid

Consider a random walk on graph G which is a rectangular 2D grid (torus) of size $\sqrt{n} \times \sqrt{n}$ with self-loops at each vertex. In this section we consider only the complexity in terms of the number of uses of `Check` and $\textsc{Shift}$. The previous best known result on quantum complexity of $\textsc{Find}(G)^{(k)}$ and $\textsc{Find}(G)^{(\geq k)}$ was $\mathrm{O}\big(\sqrt{n}(\log n)^{3/2}\big)$, from Corollary 4.6. Since the grid is a 5-regular graph (4 directions and 1 self-loop), P is symmetric. Thus, the stationary distribution of P is uniform and we simply have $p_M = m/n$. Then `Setup` is realized with $\sqrt{n}$ uses of $\textsc{Shift}$, and $\mathrm{HT}(P, \{z\}) = \Theta(n \log n)$, for any z. Therefore we get the following corollary of Theorem 4.10 and Theorem 4.14, by upper bounding $\mathrm{HT}(P, M) = \mathrm{O}(n \log n)$.

Corollary 4.16. *Let* G *be the* 2D *grid of size* $\sqrt{n} \times \sqrt{n}$*, and let* $k \geq 1$*. Then* $\textsc{Find}(G)^{(k)}$ *can be solved with expected quantum complexity* $\mathrm{O}\big(\sqrt{n \log n}\big)$*, and* $\textsc{Find}(G)^{(\geq k)}$ *with expected quantum complexity* $\mathrm{O}\big(\sqrt{n \log n \log(n/k)}\big)$.

Chapter 5

Quantum rejection sampling

Contents

5.1 Introduction

We address the problem of preparing a desired target quantum state into the memory of a quantum computer. It is of course unreasonable to try to find an efficient quan-

tum algorithm to achieve this for general quantum states. Indeed, if any state could be prepared efficiently such difficult tasks as preparing witnesses for QMA-complete problems [KKR06] could be solved efficiently, a task believed to be impossible even if classical side-information about the quantum state is provided [AD10]. On the other hand, many interesting computational problems can be related to quantum state generation problems that carry some additional *structure* which might be exploited by an efficient algorithm.

Among the most tantalizing examples of problems that are reducible to quantum state generation is the GRAPH ISOMORPHISM problem [KST93] which could be solved by preparing the quantum state $|\Gamma\rangle = \frac{1}{\sqrt{n!}} \sum_{\pi \in S_n} |\Gamma^\pi\rangle$, *i.e.*, the uniform superposition of all the permutations of a given graph Γ. By generating such states for two given graphs, one could then use the standard SWAP-test [BCWdW01] to check whether the two states are equal or orthogonal and therefore decide whether the graphs are isomorphic or not. Furthermore, it is known that all problems in statistical zero knowledge (SZK) can be reduced to instances of quantum state generation [ATS03], along with gap-versions of closest lattice vector problems [Reg04a, AR05] and subgroup membership problems for arbitrary subgroups of finite groups [Wat00, Wat01, FIM+02]. Aside from brute-force attempts that try to solve quantum state preparation by applying sequences of controlled rotations (typically of exponential length) to fix the amplitudes of the target state one qubit at a time, not much is known regarding approaches to tackle the quantum state generation problem while exploiting inherent structure.

In this regard, the only examples we are aware of are:

1. A direct approach to generate states described by efficiently computable amplitudes [GR02].
2. An approach via adiabatic computing [ATS03] in which a sequence of local Hamiltonians has to be found such that the desired target state is the ground state of a final Hamiltonian and the overlap between intermediate ground states is large.
3. Quantum analogues of classical annealing processes [BKS09, SBBK08] and quantum Metropolis sampling procedure [TOV+11, YAG10].

Conversely, for some problems a *lower* bound on the complexity of solving a corresponding quantum state generation problem would be desirable, for instance to provide further evidence for the security of quantum money schemes, see *e.g.* [Aar09, FGH+12]. Unfortunately, except for a recent result that generalizes the adversary method to a particular case of quantum state generation problems (see [LMR+11] and [AMRR11]), very little is known about lower bounds for quantum state generation problems in general.

5.1.1 Rejection sampling

The classical rejection sampling method[1] was introduced by von Neumann [vN51] to solve the following *resampling* problem: given the ability to sample according to some probability distribution P, one is asked to produce samples from some other distribution S. Conceptually, the method is extremely simple and works as follows: let $\gamma \leq 1$ be the largest scaling factor such that γS lies under P, formally, $\gamma = \min_k(p_k/s_k)$. We accept a sample k from P with probability $\gamma s_k/p_k$, otherwise we reject it and repeat. The expected number T of samples from P to produce one sample from S is then given by $T = 1/\gamma = \max_k(s_k/p_k)$. See also [Dev86] for further details and analysis of the method for various special cases of P and S. In a setting where access to the distribution P is given by a black box, this has been proved to be optimal by Letac [Let75]. The rejection sampling technique is at the core of many randomized algorithms and has a wide range of applications, ranging from computer science to statistical physics, where it is used for Monte Carlo simulations, the most well-known example being the Metropolis algorithm [MRR+53].

In the same way that quantum state preparation can be considered a quantum analogue of classical sampling, it is natural to study a quantum analogue of the classical resampling problem, *i.e.*, the problem of sampling from a distribution S given the ability to sample from another distribution P. We call this problem *quantum resampling* and define it to be the following analogue of the classical resampling problem: given an oracle generating a quantum state $|\boldsymbol{\pi}^{\xi}\rangle = \sum_{k=1}^{n} \pi_k |\xi_k\rangle|k\rangle$, where the amplitudes π_k are known but the states $|\xi_k\rangle$ are unknown, the task is to prepare a target state $|\boldsymbol{\sigma}^{\xi}\rangle = \sum_{k=1}^{n} \sigma_k |\xi_k\rangle|k\rangle$ with (potentially) different amplitudes σ_k but the same states $|\xi_k\rangle$. Note that while both the initial amplitudes π_k and the final amplitudes σ_k are fixed and known, the fact that the states $|\xi_k\rangle$ are unknown makes the problem non-trivial.

5.1.2 Related work

The query complexity of quantum state generation problems was studied in [AMRR11], where the adversary method was extended to this model and used to prove a tight lower bound on a specific quantum state generation problem called INDEXERASURE. The adversary method was later extended to quantum state *conversion* problems—where the goal is to convert an initial state into a target state—and shown to be nearly tight in the bounded error case for any problem in this class, which includes as special cases state

[1] It is also known as the accept/reject method or "hit-and-miss" sampling.

generation and the usual model of function evaluation [LMR+11]. In all these cases, the input to the problem is classical, as the oracle encodes some hidden classical data. This is where the quantum resampling problem differs from those models, as in that case the oracle encodes unknown quantum states.

Grover [Gro00] considered a special case of the quantum resampling problem, where the initial state

$$|\pi\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle \tag{5.1}$$

is a uniform superposition and one is given access to an oracle that for given input x provides a classical description of σ_x, the amplitude in the target state

$$|\sigma\rangle = \sum_x \sigma_x |x\rangle. \tag{5.2}$$

We significantly extend the scope of Grover's technique by considering any initial superposition and improving the efficiency of the algorithm when only an approximate preparation of the target state is required.

Techniques related to quantum resampling have already been used implicitly as a useful primitive for building quantum algorithms. For instance, it was used in a paper by Harrow, Hassidim, and Lloyd [HHL09] for the problem of solving a system of linear equations $Ax = b$, where A is an invertible matrix over the real or complex numbers whose entries are efficiently computable, and b is a vector. The quantum algorithm in [HHL09] solves the problem of preparing the state $|x\rangle = A^{-1}|b\rangle$ by applying the following three basic steps:

1. Use phase estimation on the state $|b\rangle = \sum_k b_k |\psi_k\rangle$ to prepare $\sum_k b_k |\lambda_k\rangle|\psi_k\rangle$, where $|\psi_k\rangle$ and λ_k denote the eigenvectors and eigenvalues of A.
2. Convert this state to $\sum_k b_k \lambda_k^{-1} |\lambda_k\rangle|\psi_k\rangle$ (up to normalization).
3. Undo the phase estimation to obtain the target state $A^{-1}|b\rangle = \sum_k b_k \lambda_k^{-1} |\psi_k\rangle$.

The second step of this procedure performs the transformation

$$\sum_k b_k |\lambda_k\rangle|\psi_k\rangle \mapsto \sum_k b_k \lambda_k^{-1} |\lambda_k\rangle|\psi_k\rangle \tag{5.3}$$

which can be seen as an instance of quantum resampling. Note that other works, such as [Chi08, SMM09], have used a similar technique—i.e., using phase estimation to simulate some function of an operator—to apply a unitary on an unknown quantum state, rather than preparing one particular state.

Temme *et al.* [TOV+11] have proposed a quantum Metropolis sampling algorithm to solve the problem of preparing the thermal state of a quantum Hamiltonian. As it is heavily inspired by the classical Metropolis algorithm, the main step uses an accept/reject rule on random moves between eigenstates of the Hamiltonian. The main complication comes from reverting rejected moves, as the no-cloning principle prevents us from keeping a copy of the previous eigenstate. We will show that this step actually reduces to a quantum resampling problem, and that quantum rejection sampling leads to an alternative solution which also provides a speed-up over the technique proposed in [TOV+11].

Finally, another type of quantum resampling problem has been considered in a paper by Kitaev and Webb [KW08] in which the task is to prepare a superposition of basis states with Gaussian-distributed weights along a low-dimensional strip inside a high-dimensional space. The authors solve this problem by applying a sequence of lattice transformation and phase estimation steps.

For us, another important case in point are hidden shift problems over an abelian group A. Here it is easy to prepare a quantum state of the form

$$|\boldsymbol{\pi}^{\xi}\rangle = \sum_{w \in A} \hat{f}(w)\chi_w(s)|w\rangle, \tag{5.4}$$

where χ_w denotes the characters of A and $\hat{f}$ denotes the Fourier transform of f (see, *e.g.*, [vDHI03, Iva08, Röt10]). If we could eliminate the Fourier coefficients $\hat{f}(w)$ from the state $|\boldsymbol{\pi}^{\xi}\rangle$, we would obtain a state

$$|\boldsymbol{\sigma}^{\xi}\rangle = |A|^{-1/2} \sum_{w \in A} \chi_w(s)|w\rangle \tag{5.5}$$

from which the hidden shift s can be easily recovered by applying another Fourier transform. Note that in this case the states $|\xi_k\rangle$ are actually just the complex phases $\chi_w(s)$. We discuss an application of our general framework to the special case of the Boolean function hidden shift problem in Sect. 6.4.2 in Chapter 6.

5.1.3 Our results

We denote the classical resampling problem mentioned above by $\textsc{Sampling}_{P \to S}$, where P and S are probability distributions on the set $[n]$. Note that in its simplest form, this problem is not meaningful in the context of query complexity (indeed, if distribution S

is known to begin with, there is no need to use the ability to sample from P). However, there is a natural modification of the problem, that actually models realistic applications, which does not suffer from this limitation. In this version of the problem, there is a function ξ that deterministically associates some unknown data with each sample, and the problem is to sample pairs $(k, \xi(k))$, where k follows the target distribution. Formally, the problem is therefore defined as follows: given oracle access to a black box producing pairs $(k, \xi(k)) \in [n] \times [d]$ such that k is distributed according to P, where $\xi : [n] \to [d]$ is an unknown function, the problem is to produce a sample $(k, \xi(k))$ such that k is distributed according to S. Note that in this model it is not possible to produce the required samples without using the access to the oracle that produces the samples from P, and the algorithm is therefore restricted to act as in Fig. 5.1.

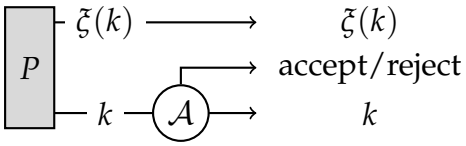

Figure 5.1: Classical rejection sampling: A black box produces samples k according to a known probability distribution P and accompanied by some unknown classical data $\xi(k)$. The algorithm $\mathcal{A}$ either accepts or rejects these samples, so that accepted samples are distributed according to a target distribution S.

The problem studied in this chapter is a quantum analogue of $\text{SAMPLING}_{P\to S}$, where probability distributions are replaced by quantum superpositions. More precisely, let $\boldsymbol{\pi}, \boldsymbol{\sigma} \in \mathbb{R}^n$ be such that $\|\boldsymbol{\pi}\|_2 = \|\boldsymbol{\sigma}\|_2 = 1$ and $\pi_k, \sigma_k \geq 0$ for all $k \in [n]$. Let O be a unitary that acts on a default state $|\bar{0}\rangle_{dn} \in \mathbb{C}^d \otimes \mathbb{C}^n$ as $O : |\bar{0}\rangle_{dn} \mapsto |\boldsymbol{\pi}^\xi\rangle := \sum_{k=1}^n \pi_k |\xi_k\rangle|k\rangle$, where $|\xi_k\rangle \in \mathbb{C}^d$ are some fixed unknown normalized quantum states. Given oracle access to unitary black boxes $O, O^\dagger$, the $\text{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ problem is to prepare the state $|\boldsymbol{\sigma}^\xi\rangle := \sum_{k=1}^n \sigma_k |\xi_k\rangle|k\rangle$. Note that a special case of this problem is the scenario $d = 1$, when $\xi_k \in \mathbb{C}$ are just unknown phases (complex numbers of absolute value 1).

The main result of this chapter is a tight characterization of the query complexity of $\text{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ for any success probability p:

Theorem 5.1. *For $p \in [p_{\min}, p_{\max}]$, the quantum query complexity of $\text{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ with success probability p is $Q_{1-p}(\text{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}) = \Theta(1/\|\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}\|_2)$, where $p_{\min}$, $p_{\max}$, and $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ are given in Definition 5.8. For $p \leq p_{\min}$, the query complexity is 1, and for $p > p_{\max}$, it is infinite.*

The vector $\boldsymbol{\varepsilon}^{p}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$, as well as the probabilities $p_{\min}$ and $p_{\max}$, will be defined in Sect. 5.2 (intuitively, the vector $\boldsymbol{\varepsilon}^{p}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ characterizes the amplitudes of the final state prepared by the best algorithm having success probability p).

Let us note that when $p = p_{\max} = 1$, the query complexity reduces to $\max_k(\sigma_k/\pi_k)$ in analogy with the classical case, except that amplitudes replace probabilities. The lower bound comes from an extension of the automorphism principle (originally introduced in the context of the adversary method [AMRR11, HLŠ07]) to our framework of quantum state generation problems with quantum oracles. The upper bound follows from an algorithm based on amplitude amplification that we call quantum rejection sampling. We also show that a modification of this algorithm can also solve a quantum state conversion problem, which we call strong quantum resampling (SQSAMPLING).

Next, we illustrate the technique by providing different applications. We first show that the main steps in two recent algorithms, namely the quantum algorithm for solving linear systems of equations [HHL09] and the quantum Metropolis sampling algorithm [TOV+11], consists in solving quantum state conversion problems which we call QLINEAREQS$_\kappa$ and QMMOVE$_{\mathcal{C}}$. We then observe that these problems reduce to SQSAMPLING, and can therefore be solved using quantum rejection sampling.

Theorem 5.2. *For any $\tilde{\kappa} \in [1,\kappa]$, there is a quantum algorithm that solves* QLINEAREQS$_\kappa$ *with success probability $p = (\boldsymbol{w}^{\mathsf{T}} \cdot \tilde{\boldsymbol{w}})/(\|\boldsymbol{w}\|_2 \cdot \|\tilde{\boldsymbol{w}}\|_2)$ using an expected number of queries $O(\tilde{\kappa}/\|\tilde{\boldsymbol{w}}\|_2)$, where $w_j := b_j/\lambda_j$, $\tilde{w}_j := b_j/\tilde{\lambda}_j$, and $\tilde{\lambda}_j := \max\{\tilde{\kappa}^{-1}, \lambda_j\}$.*

Theorem 5.3. *There is a quantum algorithm that solves* QMMOVE$_{\mathcal{C}}$ *with success probability 1 using an expected number of queries $O(1/\|\boldsymbol{w}^{(i)}\|_2)$.*

Let us note that while the quantum algorithm for linear systems of equations was indeed using this technique implicitly, this was not the case for quantum Metropolis sampling, where quantum rejection sampling provides some speed-up over the original algorithm.

Finally, quantum rejection sampling can also be used to solve the Boolean function hidden shift problem. We give a short summary on this application in Sect. 5.5.3. For more discussion on this see Sect. 6.4.2 in the next chapter.

5.2 Definition of the problem

In this section, we define different notions related to the quantum resampling problem. It is important to note that this problem goes beyond the usual model of quantum query

complexity, where the goal is to compute a function depending on some unknown classical data that can be accessed via an oracle (see [BdW02] for a comprehensive survey). In the usual model, the algorithm is therefore quantum but both its input and output are classical. A first extension of this model is the case were the output is also quantum, that is, the goal is to generate a target quantum state depending on some unknown classical data hidden by the oracle. The quantum adversary method has recently been extended to this model by [AMRR11], where is was used to characterize the query complexity if a quantum state generation problem called INDEXERASURE. In both the usual model and this extension, the oracle acts as $O_x : |b\rangle|i\rangle \mapsto |b + x_i\rangle|i\rangle$, where x is the hidden classical data. However, the quantum resampling problem corresponds to another extension of these models, where the input is also quantum, in the sense that the oracle hides unknown quantum states instead of classical data. Let us now define this extended model more precisely.

Definition 5.4 (Quantum state generation problem)**.** Let $\mathcal{O} := \{O_x : x \in \mathcal{X}\}$ and $\Psi := \{|\psi_x\rangle : x \in \mathcal{X}\}$, respectively, be sets of quantum oracles (*i.e.*, unitaries) and target quantum states labeled by elements of some set $\mathcal{X}$. Then $\mathcal{P} := (\mathcal{O}, \Psi, \mathcal{X})$ describes the following *quantum state generation problem:* given an oracle O_x for some unknown value of $x \in \mathcal{X}$, prepare a state

$$|\psi\rangle = \sqrt{p}|\psi_x\rangle|\bar{0}\rangle + |\text{error}_x\rangle, \tag{5.6}$$

where p is the desired success probability, $|\bar{0}\rangle$ is a normalized standard state for some workspace and $|\text{error}_x\rangle$ is an arbitrary error state with norm at most $\sqrt{1-p}$. The quantum query complexity of $\mathcal{P}$ is the minimum number of queries to O_x or $O_x^\dagger$ necessary to solve $\mathcal{P}$ with success probability p and will be denoted by $Q_{1-p}(\mathcal{P})$.

Intuitively, we want the final state $|\psi\rangle$ to have a component of length at least $\sqrt{p}$ in the direction of $|\psi_x\rangle|\bar{0}\rangle$. We can restate the condition $\||\text{error}_x\rangle\|_2 \leq \sqrt{1-p}$ as follows:

$$1 - p \geq \big\||\psi\rangle - \sqrt{p}|\psi_x\rangle|\bar{0}\rangle\big\|_2^2 = 1 + p - 2\,\text{Re}\big[\langle\psi| \cdot \sqrt{p}|\psi_x\rangle|\bar{0}\rangle\big], \tag{5.7}$$

or equivalently:

$$\text{Re}\big[\langle\psi| \cdot |\psi_x\rangle|\bar{0}\rangle\big] \geq \sqrt{p}. \tag{5.8}$$

Note that the main difference of the above definition with the usual model of quantum query complexity, and its extension to quantum state generation in [AMRR11], is that the oracle is not restricted to act as $O_x : |b\rangle|i\rangle \mapsto |b + x_i\rangle|i\rangle$.

We now formally define $\text{QSAMPLING}_{\pi\to\sigma}$ as a special case of the quantum state generation problem. Throughout this article, we fix positive integers d, n and we assume

that $\boldsymbol{\pi}, \boldsymbol{\sigma} \in \mathbb{R}^n$ are vectors such that $\|\boldsymbol{\pi}\|_2 = \|\boldsymbol{\sigma}\|_2 = 1$ and $\pi_k, \sigma_k \geq 0$ for all $k \in [n]$. We also use the notation $|\boldsymbol{\pi}\rangle := \sum_{k=1}^n \pi_k |k\rangle$ and $|\boldsymbol{\sigma}\rangle := \sum_{k=1}^n \sigma_k |k\rangle$. For simplicity, we assume that $\sigma_k > 0$ for all $k \in [n]$, but the general case can easily be obtained by taking the limit $\sigma_k \to 0$.

Let $\xi = (|\xi_k\rangle \in \mathbb{C}^d : k \in [n])$ be an ordered list of normalized quantum states. Then any unitary that maps a default state $|\bar{0}\rangle_{dn}$ to $|\boldsymbol{\pi}^\xi\rangle := \sum_{k=1}^n \pi_k |\xi_k\rangle|k\rangle$ is a valid oracle for QSAMPLING$_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$. Therefore, we will label valid oracles by a pair (ξ, u), where ξ denotes the states hidden by the oracle, and u defines how the oracle acts on states that are orthogonal to $|\bar{0}\rangle_{dn}$. More explicitly, we fix a default oracle $O \in \mathrm{U}(dn)$ as a unitary that acts on $|\bar{0}\rangle_{dn}$ as $O|\bar{0}\rangle_{dn} = |\bar{0}\rangle_d |\boldsymbol{\pi}\rangle$, and arbitrarily on the orthogonal complement. We then use O as a reference point to define the remaining oracles:

$$O_{\xi,u} := V_\xi \, O \, u, \tag{5.9}$$

where $u \in \mathrm{U}(dn)$ is a unitary such that $u|\bar{0}\rangle_{dn} = |\bar{0}\rangle_{dn}$ and V_ξ is a unitary that acts on $|\bar{0}\rangle_d |k\rangle$ as $V_\xi |\bar{0}\rangle_d |k\rangle = |\xi_k\rangle|k\rangle$ for any $k \in [n]$, and arbitrarily on the orthogonal complement of these states, so that

$$O_{\xi,u}|\bar{0}\rangle_{dn} = V_\xi \, O|\bar{0}\rangle_{dn} = V_\xi \sum_{k=1}^n \pi_k |\bar{0}\rangle_d |k\rangle = \sum_{k=1}^n \pi_k |\xi_k\rangle|k\rangle = |\boldsymbol{\pi}^\xi\rangle \tag{5.10}$$

(note that how O and V_ξ are defined on the orthogonal complements is irrelevant as it only affects the exact labeling, but not the set of valid oracles).

Definition 5.5 (Quantum resampling problem)**.** *Quantum resampling problem*, denoted by QSAMPLING$_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$, is an instance of quantum state generation problem $(\mathcal{O}, \Psi, \mathcal{X})$ with

$$\mathcal{X} := \big\{(\xi, u) : \xi = (|\xi_1\rangle, \ldots, |\xi_n\rangle) \in (\mathbb{C}^d)^n, u \in S\big\}, \tag{5.11}$$

$$S := \big\{u \in \mathrm{U}(dn) : u|\bar{0}\rangle_{dn} = |\bar{0}\rangle_{dn}\big\} \cong \mathrm{U}(dn-1). \tag{5.12}$$

Oracles in $\mathcal{O}$ that are hiding the states $|\boldsymbol{\pi}^\xi\rangle$ are defined according to Eq. (5.9) and the corresponding target states are defined by $|\boldsymbol{\sigma}^\xi\rangle := V_\xi |\bar{0}\rangle_d |\boldsymbol{\sigma}\rangle = \sum_{k=1}^n \sigma_k |\xi_k\rangle|k\rangle$.

Let us note that the target states only depend on the index ξ, and not u. Moreover, note that amplitudes π_k and σ_k can be assumed to be real and positive without loss of generality, as any phase can be corrected using a controlled-phase gate, which does not require any oracle call since $\boldsymbol{\pi}$ and $\boldsymbol{\sigma}$ are fixed and known.

In [LMR+11], Lee *et al.* have proposed another extension of the query complexity model for quantum state generation of [AMRR11] by considering quantum state

conversion, where the problem is now to convert a given quantum state into another quantum state, instead of generating a target quantum state from scratch. They have extended the adversary method to this model and showed that it is approximately tight in the bounded-error case for any quantum state conversion problem with a classical oracle. Here, we define a model that combines both extensions (from classical to quantum oracles as well as from state generation to state conversion), hence subsuming all previous models (see Fig. 5.2).

Definition 5.6 (Quantum state conversion problem)**.** Let $\mathcal{O} := \{O_x : x \in \mathcal{X}\}$, $\Phi := \{|\phi_x\rangle : x \in \mathcal{X}\}$ and $\Psi := \{|\psi_x\rangle : x \in \mathcal{X}\}$, respectively, be sets of quantum oracles (*i.e.*, unitaries), initial quantum states and target quantum states labelled by elements of some set $\mathcal{X}$. Then $\mathcal{P} := (\mathcal{O}, \Phi, \Psi, \mathcal{X})$ describes the following *quantum state conversion problem:* given an oracle O_x for some unknown value of $x \in \mathcal{X}$ and a copy of the corresponding initial state $|\phi_x\rangle$, prepare a state

$$|\psi\rangle = \sqrt{p}|\psi_x\rangle|\bar{0}\rangle + |\text{error}_x\rangle, \tag{5.13}$$

where p is the desired success probability, $|\bar{0}\rangle$ is a normalized standard state for some workspace and $|\text{error}_x\rangle$ is an arbitrary error state with norm at most $\sqrt{1-p}$. Again, $Q_{1-p}(\mathcal{P})$ will denote the quantum query complexity of $\mathcal{P}$.

We also define a strong version of the quantum resampling problem, which is a special case of the state conversion problem. Compared to the original resampling problem, it is made harder due to two modifications. First, instead of being given access to an oracle that maps $|\bar{0}\rangle_{dn}$ to $|\boldsymbol{\pi}^{\xi}\rangle$, we are only provided with one copy of $|\boldsymbol{\pi}^{\xi}\rangle$ and an oracle that reflects through it, making this a quantum state conversion problem. The second extension assumes that we only know the ratios between the amplitudes π_k and σ_k for each k, instead of the amplitudes themselves. More precisely, instead of vectors $\boldsymbol{\pi}, \boldsymbol{\sigma} \in \mathbb{R}^n$ specifying the initial and target amplitudes, we fix a single vector $\boldsymbol{\tau} \in \mathbb{R}^n$ such that $\tau_k \geq 0$ and $\max_k \tau_k = 1$, specifying the ratios between those amplitudes. Let us now formally define the stronger version of the quantum resampling problem (this definition is motivated by the applications that will be presented in Sect. 5.5.1 and Sect. 5.5.2).

Definition 5.7 (Strong quantum resampling problem)**.** Let $P := \{\boldsymbol{\pi} \in \mathbb{R}^n : \|\boldsymbol{\pi}\|_2 = 1, \forall k : \pi_k > 0\}$. The *strong quantum resampling problem* $\text{SQSAMPLING}_{\boldsymbol{\tau}}$ is a quantum state conversion problem $(\mathcal{O}, \Phi, \Psi, \mathcal{X})$, where $\mathcal{X} := \{(\xi, \boldsymbol{\pi}) : \xi = (|\xi_1\rangle, \ldots, |\xi_n\rangle) \in (\mathbb{C}^d)^n, \boldsymbol{\pi} \in P\}$, oracles in $\mathcal{O}$ are defined by $O_{\xi,\boldsymbol{\pi}} := \text{ref}_{|\boldsymbol{\pi}^{\xi}\rangle} = I - 2|\boldsymbol{\pi}^{\xi}\rangle\langle\boldsymbol{\pi}^{\xi}|$ with the corresponding initial and target states being $|\boldsymbol{\pi}^{\xi}\rangle$ and $|\boldsymbol{\sigma}^{\xi}\rangle = \sum_{k=1}^{n} \sigma_k |\xi_k\rangle|k\rangle$, respectively, where $\boldsymbol{\sigma} := \boldsymbol{\pi} \circ \boldsymbol{\tau} / \|\boldsymbol{\pi} \circ \boldsymbol{\tau}\|_2$ so that $\sigma_k / \pi_k \propto \tau_k$.

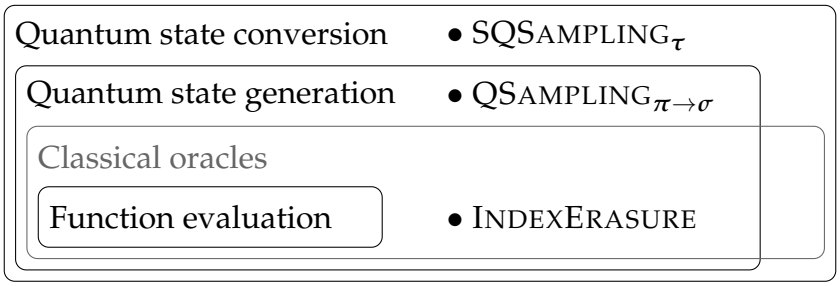

Figure 5.2: Comparison of different classes of problems in quantum query complexity. The case of function evaluation has been extensively studied in the literature. The extension to quantum state generation with classical oracles, as well as the problem INDEXERASURE which belongs to that class, have been studied in [AMRR11]. The adversary method has been extended to the case of quantum state conversion with classical oracles in [LMR+11]. The problems QSAMPLING$_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ and SQSAMPLING$_{\boldsymbol{\tau}}$ studied in this article use quantum oracles and therefore belong to yet another extension of the quantum query complexity model.

The relationship between different classes of query complexity problems introduced above, and strong and regular quantum rejection sampling as special instances of them are summarized in Fig. 5.2. Our main result is that the quantum query complexities of QSAMPLING$_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ and SQSAMPLING$_{\boldsymbol{\tau}}$ for any success probability p depend on a vector $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ defined as follows.

Definition 5.8. For any $\boldsymbol{\pi}, \boldsymbol{\sigma}$, let us define the following quantities

$$p_{\min} := (\boldsymbol{\sigma}^{\mathsf{T}} \cdot \boldsymbol{\pi})^2, \qquad \gamma_{\min} := \min_{k:\pi_k>0} (\pi_k/\sigma_k), \tag{5.14}$$

$$p_{\max} := \sum_{k:\pi_k>0} \sigma_k^2, \qquad \gamma_{\max} := \max_k (\pi_k/\sigma_k). \tag{5.15}$$

For any $\gamma \in [\gamma_{\min}, \gamma_{\max}]$, let us define a vector $\boldsymbol{\varepsilon}(\gamma)$ and a scalar $p(\gamma)$ by

$$\varepsilon_k(\gamma) := \min\{\pi_k, \gamma\sigma_k\}, \qquad p(\gamma) := \left(\frac{\boldsymbol{\sigma}^{\mathsf{T}} \cdot \boldsymbol{\varepsilon}(\gamma)}{\|\boldsymbol{\varepsilon}(\gamma)\|_2}\right)^2. \tag{5.16}$$

For $p \in [p_{\min}, p_{\max}]$, let $\bar{\gamma} \in [\gamma_{\min}, \gamma_{\max}]$ be such that $p(\bar{\gamma}) = p$ and define a vector $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}} := \boldsymbol{\varepsilon}(\bar{\gamma})$.

To see that $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ is well-defined, note that $\|\boldsymbol{\varepsilon}(\gamma)\|_2$ is monotonically increasing with γ, while $p(\gamma)$ is monotonically decreasing with γ. More precisely, for $\gamma = \gamma_{\min}$, the

vector $\boldsymbol{\varepsilon}(\gamma)$ has components $\varepsilon_k(\gamma) = \gamma\sigma_k$ if $\pi_k \neq 0$ or zero otherwise, and $p(\gamma) = p_{\max}$. For $\gamma = \gamma_{\max}$, we have $\boldsymbol{\varepsilon}(\gamma) = \boldsymbol{\pi}$ and $p(\gamma) = p_{\min}$. Between these extreme cases, $p(\gamma)$ interpolates from $p_{\max}$ to $p_{\min}$, which means that for any $p \in [p_{\min}, p_{\max}]$, there exists a value $\bar{\gamma}$ such that $p(\bar{\gamma}) = p$, which uniquely defines $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$.

Intuitively, $\boldsymbol{\varepsilon}(\gamma)$ may be interpreted as a "water-filling" vector, where γ defines the water level, and π_k defines the height of "tank" number k. As γ increases from 0 to $\gamma_{\min}$, we have $\varepsilon_k(\gamma) = \gamma\sigma_k$, meaning that all tanks are progressively filled proportionally to γ. When $\gamma > \gamma_{\min}$, some tanks are filled ($\gamma\sigma_k > \pi_k$) and cannot hold more water, while others continue to get filled.

5.3 Query complexity of quantum resampling

Let us first show that $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ defines an optimal feasible point of a certain semidefinite program (SDP). Afterwards we will show that the same SDP characterizes the quantum query complexity of $\text{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$.

Lemma 5.9. *Let $p \in [p_{\min}, p_{\max}]$, and $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$. Then, the following SDP*

$$\max_{M\succeq 0} \operatorname{Tr} M \quad s.t. \qquad \begin{array}{l} \forall k : \pi_k^2 \geq M_{kk}, \\ \operatorname{Tr}\big[(\boldsymbol{\sigma}\cdot\boldsymbol{\sigma}^{\mathsf{T}} - pI)M\big] \geq 0. \end{array} \tag{5.17}$$

has optimal value $\|\boldsymbol{\varepsilon}\|_2^2$, which is achieved by the rank-1 matrix $M = \boldsymbol{\varepsilon}\cdot\boldsymbol{\varepsilon}^{\mathsf{T}}$.

Proof sketch. We first show that $M = \boldsymbol{\varepsilon}\cdot\boldsymbol{\varepsilon}^{\mathsf{T}}$, where $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$, satisfies the constraints of SDP in Eq. (5.17) and therefore constitutes a feasible point. Therefore, the optimal value of SDP in Eq. (5.17) is at least $\operatorname{Tr} M = \|\boldsymbol{\varepsilon}\|_2^2$. Secondly, we dualize the SDP, and provide a dual-feasible point achieving the same objective value. The fact that this objective value is feasible for both the primal and the dual then implies that this is the optimal value. The details of the proof are given in Appendix A. □

Let us prove that SDP in Eq. (5.17) provides a lower bound for the $\text{QSAMPLING}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ problem. In Sect. 5.4, we will also show that this lower bound is tight by providing an explicit algorithm.

Let us emphasize that a lower bound cannot be obtained from standard methods such as the adversary method [Amb00, HLŠ07] (which has recently been proved to be

tight for evaluating functions [Rei09, Rei11, LMR+11]), nor from its extension to quantum state generation problems [AMRR11, LMR+11], because in this case the oracle is also quantum, in the sense that it encodes some unknown quantum state rather than some unknown classical data. To prove lower bounds it is useful to exploit possible symmetries of the problem. We extend the notion of automorphism group [AMRR11, HLŠ07] to our framework of quantum state generation problems:

Definition 5.10 (Automorphism group)**.** We say that G is the *automorphism group* of problem $(\mathcal{O}, \Psi, \mathcal{X})$ if:

1. G acts on $\mathcal{X}$ (and thus implicitly also on $\mathcal{O}$ as $g : O_x \mapsto O_{g(x)}$).
2. For any $g \in G$ there is a unitary U_g such that $U_g|\psi_x\rangle = |\psi_{g(x)}\rangle$ for all $x \in \mathcal{X}$.
3. For any given $g \in G$ it is possible to simulate the oracles $O_{g(x)}$ for all $x \in \mathcal{X}$, using only a constant number of queries to the black box O_x.

While for the standard model of quantum query complexity, where the oracle encodes some unknown classical data, the automorphism group is restricted to be a permutation group and is therefore always finite, in this more general framework the automorphism group can be continuous. For example, in the case of $\text{QSAMPLING}^p_{\pi\to\sigma}$ it will involve the unitary group. Taking such symmetries into account might significantly simplify the analysis of algorithms for the corresponding problem and is the key to prove our lower bound.

Lemma 5.11. *Any quantum algorithm for* $\text{QSAMPLING}^p_{\pi\to\sigma}$ *with* $p \in [p_{\min}, p_{\max}]$ *requires at least* $\Omega(1/\|\varepsilon^p_{\pi\to\sigma}\|_2)$ *queries to* O *and* $O^\dagger$.

Proof. The proof proceeds as follows: we first define a subset of oracles that are sufficiently hard to distinguish to characterize the query complexity of the problem. Exploiting the automorphism group of this subset of oracles, we then show that any algorithm may be symmetrized in such a way that the real part of the amplitudes of the final state prepared by the algorithm does not depend on the specific oracle it was given. These amplitudes define a vector $\bar{\gamma}$ that should satisfy some constraints for the algorithm to have success probability p. Moreover, we can use the hybrid argument to show that the components of $\bar{\gamma}$ should also satisfy some constraints for the algorithm to be able to generate the corresponding state in at most T queries. Putting all these constraints together in an optimization program, we then show that such a vector $\bar{\gamma}$ cannot exist unless T is large enough. This optimization program is then shown to be equivalent to the semidefinite program in Lemma 5.9, which proves the theorem.

Let us now give the details of the proof. For given $\pi, \sigma \in \mathbb{R}^n$, let us choose a subset of oracles $\mathcal{O}'_{\pi,\sigma} \subset \mathcal{O}_{\pi,\sigma}$ that are hard to distinguish. We choose the states hidden inside oracles to be of the form $|\xi_k\rangle = (-1)^{x_k}|\bar{0}\rangle_d$, where phases are given by some unknown string $x \in \mathbb{F}_2^n$. We also choose u so that any oracle in the subset acts trivially on the d-dimensional register holding the unknown states. In that case, this register is effectively one-dimensional, so we will omit it and write $(-1)^{x_k}$ as a relative phase. More precisely, we consider a set of oracles $\mathcal{O}'_{\pi,\sigma} := \{O_{x,u} : x \in \mathbb{F}_2^n, u \in S\}$, where

$$S := \{u \in \mathrm{U}(n) : u|\bar{0}\rangle_n = |\bar{0}\rangle_n\} \cong \mathrm{U}(n-1). \tag{5.18}$$

As in the general case, we define the first oracle $O_{0,I}$ as a unitary that acts on $|\bar{0}\rangle_n$ as $O_{0,I}|\bar{0}\rangle_n = |\pi\rangle$, and arbitrarily on the orthogonal complement, and we use $O_{0,I}$ as a reference point to define the remaining oracles:

$$O_{x,u} := V_x O_{0,I} u, \qquad \text{where} \qquad V_x := \sum_{k=1}^{n} (-1)^{x_k} |k\rangle\langle k|. \tag{5.19}$$

The set of target states is $\Psi'_{\pi,\sigma} := \{|\sigma(x)\rangle : x \in \mathbb{F}_2^n, u \in S\}$ where $|\sigma(x)\rangle := V_x|\sigma\rangle = \sum_{k=1}^{n}(-1)^{x_k}\sigma_k|k\rangle$. For the quantum state generation problem corresponding to the restricted set of oracles $\mathcal{O}'_{\pi,\sigma}$, the automorphism group is $G = \mathbb{F}_2^n \times \mathrm{U}(n-1)$ and it acts on itself, *i.e.*, $\mathcal{X} = G$. Note that the target states depend only on x, but u is used for parameterizing the oracles. Intuitively, the reason we need this parameter is that the algorithm should not depend on how the black box acts on states other than $|\bar{0}\rangle_n$ (or how its inverse acts on states other than $|\pi^{\xi}\rangle$). To make this intuition formal, we will later choose the parameter u for different oracles adversarially.

Let us consider an algorithm that uses T queries to the black box $O_{x,u}$ and its inverse, and let us denote the final state of this algorithm by $|\psi_T(x,u)\rangle$. If we expand the first register in the standard basis, we can express this state as

$$|\psi_T(x,u)\rangle = \sum_{k=1}^{n} (-1)^{x_k} |k\rangle|\gamma_k(x,u)\rangle. \tag{5.20}$$

Here the workspace vectors $|\gamma_k(x,u)\rangle$ can have arbitrary dimension and are not necessarily unit vectors, but instead satisfy the normalization constraint $\sum_{k=1}^{n} \||\gamma_k(x,u)\rangle\|_2^2 = 1$. If the algorithm succeeds with probability p, then according to Eq. (5.8) for any x and

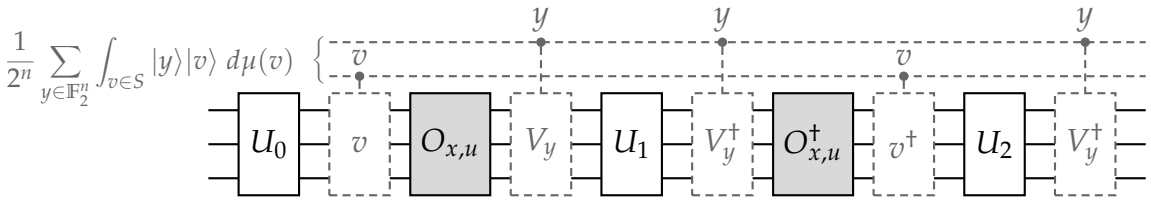

Figure 5.3: Symmetrized algorithm. We symmetrize the algorithm by introducing unitaries V_y and v, controlled by an extra register prepared in the uniform superposition over all $|y\rangle$ and $|v\rangle$.

u we have

$$\sqrt{p} \le \mathrm{Re}\Big[\langle\sigma(x)|\langle\bar{0}|\cdot|\psi_T(x,u)\rangle\Big] \tag{5.21}$$

$$= \mathrm{Re}\left[\sum_{k=1}^{n}\sigma_k\cdot\langle\bar{0}|\gamma_k(x,u)\rangle\right] \tag{5.22}$$

$$= \sigma^{\mathsf{T}}\cdot\gamma(x,u), \tag{5.23}$$

where $\gamma(x,u)$ is a real vector whose components are given by

$$\gamma_k(x,u) := \mathrm{Re}\Big[\langle\bar{0}|\gamma_k(x,u)\rangle\Big]. \tag{5.24}$$

Note that $\|\gamma(x,u)\|_2 \le 1$.

Next, let us show that we can symmetrize the algorithm without decreasing its success probability. We do this by replacing each oracle call by $O_{x+y,uv} = V_y O_{x,u} v$ and correcting the final state by applying $V_y^\dagger$ (see Fig. 5.3). Let μ be the Haar measure on the set S defined in Eq. (5.18). We define an operation that symmetrizes a set of states:

$$\overline{|\phi(x,u)\rangle} := \frac{1}{\sqrt{2^n}}\sum_{y\in\mathbb{F}_2^n}\int_{v\in S}\Big[(V_y^\dagger\otimes I)|\phi(x+y,uv)\rangle\Big]|y\rangle|v\rangle\, d\mu(v). \tag{5.25}$$

If we symmetrize the final state $|\psi_T(x,u)\rangle$, we get

$$\overline{|\psi_T(x,u)\rangle} = \frac{1}{\sqrt{2^n}}\sum_{y\in\mathbb{F}_2^n}\int_{v\in S}\sum_{k=1}^{n}(-1)^{x_k}|k\rangle|\gamma_k(x+y,uv)\rangle|y\rangle|v\rangle\, d\mu(v). \tag{5.26}$$

Note that the target state $|\sigma(x)\rangle|\bar{0}\rangle$ is already symmetric, so symmetrization only introduces an additional workspace register in a default state (uniform superposition):

$$\overline{|\sigma(x)\rangle|\bar{0}\rangle} = |\sigma(x)\rangle|\bar{0}\rangle\frac{1}{\sqrt{2^n}}\sum_{y\in\mathbb{F}_2^n}\int_{v\in S}|y\rangle|v\rangle\, d\mu(v). \tag{5.27}$$

The success probability of the symmetrized algorithm is

$$\sqrt{\bar{p}} := \mathrm{Re}\Big[\overline{\langle\sigma(x)|\langle\bar{0}|}\cdot\overline{|\psi_T(x,u)\rangle}\Big] \tag{5.28}$$

$$= \sum_{k=1}^{n}\sigma_k\cdot\frac{1}{2^n}\sum_{y\in\mathbb{F}_2^n}\int_{v\in S}\mathrm{Re}\Big[\langle\bar{0}|\gamma_k(x+y,uv)\rangle\Big]\,d\mu(v) \tag{5.29}$$

$$= \sigma^{\mathsf{T}}\cdot\bar{\gamma}, \tag{5.30}$$

where, by changing variables, we get that $\bar{\gamma}$ is the average of vectors $\gamma(y,v)$ and thus does not depend on x and u:

$$\bar{\gamma} := \frac{1}{2^n}\sum_{y\in\mathbb{F}_2^n}\int_{v\in S}\gamma(y,v)\,d\mu(v). \tag{5.31}$$

Note that $\|\bar{\gamma}\|_2 \le 1$ by triangle inequality. Also, note that $\bar{p}\ge p$, since the mean is at least as large as the minimum. Thus without loss of generality we can consider only symmetrized algorithms.

Let $x,x'\in\mathbb{F}_2^n$ and $u,u'\in S$. The difference of final states of the symmetrized algorithm that uses oracles $O_{x,u}$ and $O_{x',u'}$ is

$$\Big\|\overline{|\psi_T(x,u)\rangle}-\overline{|\psi_T(x',u')\rangle}\Big\|_2^2 \tag{5.32}$$

$$= \Big\|\sum_{k=1}^{n}\frac{1}{\sqrt{2^n}}\sum_{y\in\mathbb{F}_2^n}\int_{v\in S}|k\rangle\big((-1)^{x_k}|\gamma_k(x+y,uv)\rangle-(-1)^{x'_k}|\gamma_k(x'+y,u'v)\rangle\big)|y\rangle|v\rangle\,d\mu(v)\Big\|_2^2$$

$$= \sum_{k=1}^{n}\frac{1}{2^n}\sum_{y\in\mathbb{F}_2^n}\int_{v\in S}\Big\|(-1)^{x_k}|\gamma_k(x+y,uv)\rangle-(-1)^{x'_k}|\gamma_k(x'+y,u'v)\rangle\Big\|_2^2\,d\mu(v) \tag{5.33}$$

$$\ge \sum_{k=1}^{n}\frac{1}{2^n}\sum_{y\in\mathbb{F}_2^n}\int_{v\in S}\big((-1)^{x_k}\gamma_k(x+y,uv)-(-1)^{x'_k}\gamma_k(x'+y,u'v)\big)^2\,d\mu(v) \tag{5.34}$$

$$\ge \sum_{k=1}^{n}\big((-1)^{x_k}\bar{\gamma}_k-(-1)^{x'_k}\bar{\gamma}_k\big)^2 \tag{5.35}$$

$$= \sum_{k:x_k\neq x'_k}\big(2\bar{\gamma}_k\big)^2. \tag{5.36}$$

Here the two inequalities were obtained from the following facts:

1. If $|\bar{0}\rangle$ is a unit vector then for any $|\gamma\rangle$ we have:

$$\||\gamma\rangle\|_2^2 \geq \||\bar{0}\rangle\langle\bar{0}|\gamma\rangle\|_2^2 = |\langle\bar{0}|\gamma\rangle|^2 \geq \big(\mathrm{Re}[\langle\bar{0}|\gamma\rangle]\big)^2. \tag{5.37}$$

2. For any function $\gamma(y, v)$ by Cauchy–Schwarz inequality we have:

$$\frac{1}{2^n}\sum_{y\in\mathbb{F}_2^n}\int_{v\in S}\gamma(y,v)^2\,d\mu(v) \geq \left(\frac{1}{2^n}\sum_{y\in\mathbb{F}_2^n}\int_{v\in S}\gamma(y,v)\,d\mu(v)\right)^2. \tag{5.38}$$

Recall the *hybrid argument* of [BBBV97, Vaz98]. If O and O' are unitary matrices, then for any vectors $|\psi\rangle$ and $|\psi'\rangle$ it holds that

$$\begin{aligned}\big\|O|\psi\rangle - O'|\psi'\rangle\big\|_2 &= \big\|O|\psi\rangle - O'|\psi\rangle + O'|\psi\rangle - O'|\psi'\rangle\big\|_2 &(5.39)\\ &\leq \big\|(O-O')|\psi\rangle\big\|_2 + \|O'(|\psi\rangle - |\psi'\rangle)\|_2 &(5.40)\\ &\leq \big\|O-O'\big\|_\infty + \big\||\psi\rangle - |\psi'\rangle\big\|_2. &(5.41)\end{aligned}$$

By induction, we get the following upper bound:

$$\Big\|\overline{|\psi_T(x,u)\rangle} - \overline{|\psi_T(x',u')\rangle}\Big\|_2 \leq T\cdot\|O_{x,u} - O_{x',u'}\|_\infty, \tag{5.42}$$

where $\|\cdot\|_\infty$ denotes the usual operator norm. Bounds from Eqs. (5.36) and (5.42) together imply that for any $x, x' \in \mathbb{F}_2^n$ and $u, u' \in S$:

$$T \geq \frac{\sqrt{\sum_{k:x_k\neq x'_k}\big(2\bar{\gamma}_k\big)^2}}{\|O_{x,u} - O_{x',u'}\|_\infty}. \tag{5.43}$$

To obtain a good lower bound, we want to choose oracles $O_{x,u}$ and $O_{x',u'}$ to be as similar as possible. First, let us fix $u := I$, $x := 0$ and $x' := e_l$, where e_l is the l-th standard basis vector. Then, the numerator in Eq. (5.43) is simply $2\bar{\gamma}_l$. Let us choose u' in order to minimize the denominator. Recall that $O_{0,I}|\bar{0}\rangle_n = |\boldsymbol{\pi}\rangle$ and note that any unitary matrix that fixes $|\boldsymbol{\pi}\rangle$ can be written as $O_{0,I}u'(O_{0,I})^\dagger$ for some choice of u' fixing $|\bar{0}\rangle_n$. Since $O_{x',u'}|\bar{0}\rangle_n = V_{e_l}O_{0,I}u'|\bar{0}\rangle_n = V_{e_l}|\boldsymbol{\pi}\rangle$, we also have $O_{x',u'}(O_{0,I})^\dagger|\boldsymbol{\pi}\rangle = V_{e_l}|\boldsymbol{\pi}\rangle$, and any unitary matrix that sends $|\boldsymbol{\pi}\rangle$ to $V_{e_l}|\boldsymbol{\pi}\rangle$ can be expressed as $O_{x',u'}(O_{0,I})^\dagger$ for some choice of u'. Let us choose u' so that $O_{x',u'}(O_{0,I})^\dagger$ acts as a rotation in the two-dimensional subspace $\mathrm{span}\{|\boldsymbol{\pi}\rangle, V_{e_l}|\boldsymbol{\pi}\rangle\}$ and as identity on the orthogonal complement. If θ denotes

the angle of this rotation, then $\cos\theta = \langle \boldsymbol{\pi} | V_{e_l} | \boldsymbol{\pi} \rangle = \sum_{k=1}^{n} \pi_k^2 - 2\pi_l^2 = 1 - 2\pi_l^2$ and $\sin\theta = \sqrt{1-(1-2\pi_l^2)^2} = 2\pi_l\sqrt{1-\pi_l^2}$. Then

$$\begin{aligned} \|O_{0,I} - O_{x',u'}\|_\infty &= \|I - O_{x',u'}(O_{0,I})^\dagger\|_\infty && (5.44)\\ &= \left\| I - \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \right\|_\infty && (5.45)\\ &= 2\pi_l \left\| \begin{pmatrix} \pi_l & \sqrt{1-\pi_l^2} \\ -\sqrt{1-\pi_l^2} & \pi_l \end{pmatrix} \right\|_\infty && (5.46)\\ &= 2\pi_l, && (5.47) \end{aligned}$$

where the singular values of the last matrix are equal to 1, since it is a rotation. By plugging this back in Eq. (5.43), we get that for any $l \in [n]$:

$$T \geq \frac{|\bar{\gamma}_l|}{\pi_l}. \tag{5.48}$$

Thus any quantum algorithm that solves $\text{QSAMPLING}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ with T queries and success probability p gives us some vector $\bar{\gamma}$ such that

$$\|\bar{\gamma}\|_2 \leq 1, \qquad \boldsymbol{\sigma}^\mathsf{T} \cdot \bar{\gamma} \geq \sqrt{p}, \qquad \forall l : |\bar{\gamma}_l| \leq T\pi_l. \tag{5.49}$$

To obtain a lower bound on T, we have to find the smallest possible t such that there is still a feasible value of $\bar{\gamma}$ that satisfies Eqs. (5.49) (with T replaced by t). We can state this as an optimization problem:

$$\begin{aligned} T \geq \min_{\bar{\gamma}} t \quad \text{s.t.} \quad & \|\bar{\gamma}\|_2 \leq 1, \\ & \forall l : |\bar{\gamma}_l| \leq t\pi_l, \\ & \boldsymbol{\sigma}^\mathsf{T} \cdot \bar{\gamma} \geq \sqrt{p}. \end{aligned} \tag{5.50}$$

Finally, let us show that we can take a feasible solution $\bar{\gamma}$ of problem in Eq. (5.50) and modify its components, without increasing the objective value or violating any of the constraints, so that $\forall l : \bar{\gamma}_l \geq 0$ and $\|\bar{\gamma}\|_2 = 1$. Clearly, making all components of $\bar{\gamma}$ non-negative does not affect the objective value and makes the last constraint only easier to satisfy since $\sigma_k \geq 0$ for all k. To turn $\bar{\gamma}$ into a unit vector, first observe that not all of the constraints $\bar{\gamma}_l \leq t\pi_l$ can be saturated (indeed, in that case we would have $\bar{\gamma} = t\boldsymbol{\pi}$ with $t < 1$ since $\|\bar{\gamma}\|_2 < \|\boldsymbol{\pi}\|_2 = 1$, but the last constraint then implies $\boldsymbol{\sigma}^\mathsf{T} \cdot \boldsymbol{\pi} > \sqrt{p}$, which contradicts the assumption $p \geq p_{\min}$). If $\|\bar{\gamma}\|_2 < 1$, let j be such that $\bar{\gamma}_j < t\pi_j$.

We increase $\bar{\gamma}_j$ until either $\|\bar{\gamma}\|_2 = 1$ or $\bar{\gamma}_j = t\pi_j$. We then repeat with another j such that $\bar{\gamma}_j < t\pi_j$, until we reach $\|\bar{\gamma}\|_2 = 1$. Note that while doing so, the other constraints remain satisfied. Therefore, the program reduces to

$$T \geq \min_{\bar{\gamma}} t \quad \text{s.t.} \quad \begin{array}{l} \|\bar{\gamma}\|_2 = 1, \\ \forall l : 0 \leq \bar{\gamma}_l \leq t\pi_l, \\ \boldsymbol{\sigma}^{\mathsf{T}} \cdot \bar{\gamma} \geq \sqrt{p}. \end{array} \tag{5.51}$$

Note that we need $p \leq p_{\max}$, otherwise this program has no feasible point. Setting $\boldsymbol{\varepsilon} = \bar{\gamma}/t$, we may rewrite this program as in Eq. (A.1) in Appendix A:

$$\tfrac{1}{T^2} \leq \max_{\varepsilon_k \geq 0} \|\boldsymbol{\varepsilon}\|_2^2 \quad \text{s.t.} \quad \begin{array}{l} \forall k : \pi_k \geq \varepsilon_k \geq 0, \\ \boldsymbol{\sigma}^{\mathsf{T}} \cdot \boldsymbol{\varepsilon} \geq \sqrt{p}\|\boldsymbol{\varepsilon}\|_2, \end{array} \tag{5.52}$$

Finally, setting $M = \boldsymbol{\varepsilon} \cdot \boldsymbol{\varepsilon}^{\mathsf{T}}$, this program becomes the same as the SDP in Eq. (5.17), with the additional constraint that M is rank-1. However, we know from Lemma 5.9 that the SDP in Eq. (5.17) admits a rank-1 optimal point, therefore adding this constraint does not modify the objective value, which is also $\|\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}\|_2^2$ by Lemma 5.9. □

5.4 Quantum rejection sampling algorithm

In this section we describe quantum rejection sampling algorithms for $\text{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ and $\text{SQSAMPLING}_{\boldsymbol{\tau}}$ problems. We also explain the intuition behind our method and its relation to the classical rejection sampling. Our algorithms are based on amplitude amplification [BHMT00] and can be seen as an extension of the algorithm in [Gro00].

5.4.1 Intuitive description of the algorithm

The workspace of our algorithm is $\mathbb{C}^d \otimes \mathbb{C}^n \otimes \mathbb{C}^2$, where the last register can be interpreted as a quantum coin that determines whether a sample will be rejected or accepted (this corresponds to basis states $|0\rangle$ and $|1\rangle$, respectively). Our algorithm is parametrized by a vector $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ ($0 \leq \varepsilon_k \leq \pi_k$ for all k) that characterizes how much of the amplitude from the initial state will be used for creating the final state (in classical rejection sampling ε_k^2 corresponds to the probability that a specific value of k is drawn from the initial distribution and accepted).

We start in the initial state $|\bar{0}\rangle_d|\bar{0}\rangle_n|0\rangle$ and apply the oracle O to prepare $|\boldsymbol{\pi}^{\xi}\rangle$ in the first two registers:

$$O|\bar{0}\rangle_{dn} \otimes |0\rangle = |\boldsymbol{\pi}^{\xi}\rangle|0\rangle = \sum_{k=1}^{n} \pi_k |\xi_k\rangle|k\rangle|0\rangle. \tag{5.54}$$

Next, for each k let $R_{\boldsymbol{\varepsilon}}(k)$ be a single-qubit unitary operation defined[2] as follows (this is a rotation by an angle whose sine is equal to ε_k/π_k):

$$R_{\boldsymbol{\varepsilon}}(k) := \frac{1}{\pi_k} \begin{pmatrix} \sqrt{|\pi_k|^2 - \varepsilon_k^2} & -\varepsilon_k \\ \varepsilon_k & \sqrt{|\pi_k|^2 - \varepsilon_k^2} \end{pmatrix}. \tag{5.55}$$

Let $R_{\boldsymbol{\varepsilon}} := \sum_{k=1}^{n} |k\rangle\langle k| \otimes R_{\boldsymbol{\varepsilon}}(k)$ be a block-diagonal matrix that performs rotations by different angles in mutually orthogonal subspaces. Then $I_d \otimes R_{\boldsymbol{\varepsilon}}$ corresponds to applying $R_{\boldsymbol{\varepsilon}}(k)$ on the last qubit, controlled on the value of the second register being equal to k. This operation transforms state from Eq. (5.54) into

$$|\Psi_{\boldsymbol{\varepsilon}}\rangle := (I_d \otimes R_{\boldsymbol{\varepsilon}}) \cdot |\boldsymbol{\pi}^{\xi}\rangle|0\rangle \tag{5.56}$$

$$= \sum_{k=1}^{n} |\xi_k\rangle|k\rangle\Big(\sqrt{|\pi_k|^2 - \varepsilon_k^2}\,|0\rangle + \varepsilon_k|1\rangle\Big). \tag{5.57}$$

If we would measure the coin register of $|\Psi_{\boldsymbol{\varepsilon}}\rangle$ in the basis $\{|0\rangle, |1\rangle\}$, the probability of outcome $|1\rangle$ ("accept") and the corresponding post-measurement state would be

$$q_{\boldsymbol{\varepsilon}} := \Big\|\big(I_d \otimes I_n \otimes |1\rangle\langle 1|\big)|\Psi_{\boldsymbol{\varepsilon}}\rangle\Big\|_2^2 = \sum_{k=1}^{n} \varepsilon_k^2 = \|\boldsymbol{\varepsilon}\|_2^2, \tag{5.58}$$

$$|\Psi_{\Pi,\boldsymbol{\varepsilon}}\rangle := \frac{1}{\|\boldsymbol{\varepsilon}\|_2} \sum_{k=1}^{n} \varepsilon_k |\xi_k\rangle|k\rangle|1\rangle. \tag{5.59}$$

Note that if the vector of amplitudes $\boldsymbol{\varepsilon}$ is chosen to be close to $\boldsymbol{\sigma}$, then the reduced state on the first two registers of $|\Psi_{\Pi,\boldsymbol{\varepsilon}}\rangle$ has a large overlap on the target state $|\boldsymbol{\sigma}^{\xi}\rangle$, more precisely,

$$\sqrt{p_{\boldsymbol{\varepsilon}}} := \big(\langle\boldsymbol{\sigma}^{\xi}| \otimes \langle 1|\big)|\Psi_{\Pi,\boldsymbol{\varepsilon}}\rangle = \boldsymbol{\sigma}^{\mathsf{T}} \cdot \frac{\boldsymbol{\varepsilon}}{\|\boldsymbol{\varepsilon}\|_2}, \tag{5.60}$$

Depending on the choice of $\boldsymbol{\varepsilon}$, this can be a reasonably good approximation, so our strategy will be to prepare a state close to $|\Psi_{\Pi,\boldsymbol{\varepsilon}}\rangle$.

[2]For those k, for which $\pi_k = 0$, operation $R_{\boldsymbol{\varepsilon}}(k)$ can be defined arbitrarily.

In principle, we could obtain $|\Psi_{\Pi,\varepsilon}\rangle$ by repeatedly preparing $|\Psi_\varepsilon\rangle$ and measuring its coin register until we get the outcome "accept" (we would succeed with high probability after $O(1/q_\varepsilon)$ steps). To speed up this process, we can use amplitude amplification [BHMT00] to amplify the amplitude of the "accept" state $|1\rangle$ in the coin register of the state in Eq. (5.57). This allows us to increase the probability of outcome "accept" arbitrarily close to 1 in $O(1/\sqrt{q_\varepsilon})$ steps.

5.4.2 Amplitude amplification subroutine and quantum rejection sampling algorithm

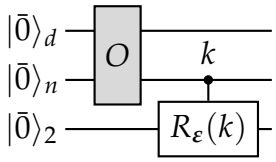

Figure 5.4: Quantum circuit for implementing U_ε.

We will use the following amplitude amplification subroutine extensively in all algorithms presented in this paper:

$$\mathcal{S}_{\mathrm{QRS}}(\mathrm{ref}_{|\pi^\xi\rangle|0\rangle}, \varepsilon, t) := \left(\mathrm{ref}_{|\Psi_\varepsilon\rangle} \cdot \mathrm{ref}_{I_d\otimes I_n\otimes|1\rangle\langle 1|}\right)^t, \tag{5.61}$$

where reflections through the two subspaces are defined as follows:

$$\mathrm{ref}_{I_d\otimes I_n\otimes|1\rangle\langle 1|} := I_d \otimes I_n \otimes \left(I_2 - 2|1\rangle\langle 1|\right) = I_d \otimes I_n \otimes Z, \tag{5.62}$$

$$\mathrm{ref}_{|\Psi_\varepsilon\rangle} := (I_d \otimes R_\varepsilon)\ \mathrm{ref}_{|\pi^\xi\rangle|0\rangle}\ (I_d \otimes R_\varepsilon)^\dagger. \tag{5.63}$$

Depending on the application, we will either be given an oracle O for preparing $|\pi^\xi\rangle|0\rangle$ or an oracle $\mathrm{ref}_{|\pi^\xi\rangle|0\rangle}$ for reflecting through this state. Note that we can always use the preparation oracle to implement the reflection $\mathrm{ref}_{|\pi^\xi\rangle|0\rangle}$ as

$$(O \otimes I_2)\ \left(I_d \otimes I_n \otimes I_2 - 2|\bar{0},\bar{0},0\rangle\langle\bar{0},\bar{0},0|\right)\ (O \otimes I_2)^\dagger. \tag{5.64}$$

Amplitude amplification subroutine $\mathcal{S}_{\mathrm{QRS}}(\mathrm{ref}_{|\pi^\xi\rangle|0\rangle}, \varepsilon, t)$ for quantum rejection sampling

Perform the following steps t times:

1. Perform $\mathrm{ref}_{I_d \otimes I_n \otimes |1\rangle\langle 1|}$ by applying Pauli Z on the coin register.
2. Perform $\mathrm{ref}_{|\Psi_\varepsilon\rangle}$ by applying $R_\varepsilon^\dagger$ on the last two registers, applying $\mathrm{ref}_{|\pi^\xi\rangle|0\rangle}$, and then undoing R_ε.

The quantum rejection sampling algorithm $\mathcal{A}_{\mathrm{QRS}}(O, \boldsymbol{\pi}, \boldsymbol{\varepsilon})$ starts with the initial state $|\bar{0}\rangle_d|\bar{0}\rangle_n|0\rangle$. First, we transform it into $|\Psi_\varepsilon\rangle$ defined in Eq. (5.57), by applying $U_\varepsilon := (I_d \otimes R_\varepsilon) \cdot (O \otimes I_2)$ (see Fig. 5.4). Then we apply the amplitude amplification subroutine $\mathcal{S}_{\mathrm{QRS}}(\mathrm{ref}_{|\pi^\xi\rangle|0\rangle}, \varepsilon, t)$ with $t = O\big(1/\|\varepsilon\|_2\big)$. Finally, we measure the last register: if the outcome is $|1\rangle$, we output the first two registers, otherwise we output "Fail". To prevent the outcome "Fail" we can slightly adjust the angle of rotation in amplitude amplification so that the target state is reached exactly. More precisely, we prove that one can choose $\varepsilon = r \cdot \varepsilon^p_{\pi\to\sigma}$ for some bounded constant r, so that amplitude amplification succeeds with probability 1 (*i.e.*, the outcome of the final measurement is always $|1\rangle$). Moreover, such algorithm is optimal as its cost matches the lower bound in Lemma 5.11.

Quantum rejection sampling algorithm $\mathcal{A}_{\mathrm{QRS}}(O, \boldsymbol{\pi}, \boldsymbol{\varepsilon})$

1. Start in initial state $|\bar{0}\rangle_d|\bar{0}\rangle_n|0\rangle$.
2. Apply U_ε.
3. Apply the amplitude amplification subroutine $\mathcal{S}_{\mathrm{QRS}}(\mathrm{ref}_{|\pi^\xi\rangle|0\rangle}, \varepsilon, t)$ where $\mathrm{ref}_{|\pi^\xi\rangle|0\rangle}$ is implemented according to Eq. (5.64) and $t = O(1/\|\varepsilon\|_2)$.
4. Measure the last register. If the outcome is $|1\rangle$, output the first two registers, otherwise output "Fail".

Lemma 5.12. *For any $p_{\min} \le p \le p_{\max}$, there is a constant $r \in [\frac{1}{2}, 1]$, so that the algorithm $\mathcal{A}_{QRS}(O, \boldsymbol{\pi}, \boldsymbol{\varepsilon})$ with $\boldsymbol{\varepsilon} = r \cdot \boldsymbol{\varepsilon}^p_{\pi\to\sigma}$ solves $\mathrm{QSAMPLING}_{\pi\to\sigma}$ with success probability p using $O\big(1/\|\boldsymbol{\varepsilon}^p_{\pi\to\sigma}\|_2\big)$ queries to O and $O^\dagger$.*

Proof. By Definition 5.8, we have $0 \le \varepsilon_k \le \pi_k$ for all k, therefore $\boldsymbol{\varepsilon}^p_{\pi\to\sigma}$ is a valid choice of vector $\boldsymbol{\varepsilon}$ for the algorithm. Instead of using $\boldsymbol{\varepsilon}^p_{\pi\to\sigma}$ itself, we slightly scale it down by a factor r so that the amplitude amplification never fails. Note that if we were to use $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^p_{\pi\to\sigma}$, the probability that the amplitude amplification succeeds after t steps would be $\sin^2\big((2t+1)\theta\big)$, where $\theta := \arcsin\|\boldsymbol{\varepsilon}^p_{\pi\to\sigma}\|_2$ (see *e.g.* [BBHT98, BHMT00] for details).

Note that this probability would be equal to one at $t = \frac{\pi}{4\theta} - \frac{1}{2}$, which in general might not be an integer. However, following an idea from [BHMT00, p.10], we can ensure this by slightly decreasing θ to $\tilde{\theta} := \frac{\pi}{2(2\tilde{t}+1)}$ where $\tilde{t} := \lceil \frac{\pi}{4\theta} - \frac{1}{2} \rceil$. This can be done by setting $\boldsymbol{\varepsilon} := r \cdot \boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ with the scaling-down factor $r := \frac{\sin\tilde{\theta}}{\sin\theta}$. One can check that r satisfies $\frac{1}{2} \leq r \leq 1$ (this follows from $0 \leq \theta \leq \frac{\pi}{2}$).

Together with Eq. (5.60), Definition 5.8 also implies that for this choice, the algorithm solves $\text{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ with success probability $\frac{\boldsymbol{\sigma}^\mathsf{T}\cdot\boldsymbol{\varepsilon}}{\|\boldsymbol{\varepsilon}\|_2} = \frac{\boldsymbol{\sigma}^\mathsf{T}\cdot\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}}{\|\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}\|_2} = \sqrt{p}$. It therefore remains to prove that the cost of the algorithm is $O(1/\|\boldsymbol{\varepsilon}\|_2)$, which follows immediately from its description: we need one query to implement the operation $U_{\boldsymbol{\varepsilon}}$ and two queries to implement $\text{ref}_{|\boldsymbol{\pi}^{\vec{\xi}}\rangle|0\rangle}$, thus in total we need $2t+1 = O\big(1/\sqrt{q_{\boldsymbol{\varepsilon}}}\big) = O(1/\|\boldsymbol{\varepsilon}\|_2)$ calls to oracles O and $O^\dagger$. □

We now have all the elements to prove Theorem 5.1.

Theorem 5.1. *For $p \in [p_{\min}, p_{\max}]$, the quantum query complexity of $\text{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ with success probability p is $Q_{1-p}(\text{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}) = \Theta(1/\|\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}\|_2)$, where $p_{\min}$, $p_{\max}$, and $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ are given in Definition 5.8. For $p \leq p_{\min}$, the query complexity is 1, and for $p > p_{\max}$, it is infinite.*

Proof. When $p \leq p_{\min}$, the state $|\boldsymbol{\pi}^{\vec{\xi}}\rangle$ is already close enough to $|\boldsymbol{\sigma}^{\vec{\xi}}\rangle$ to satisfy the constraint on the success probability, therefore one call to O is sufficient, which is clearly optimal. When $p > p_{\max}$, the oracle gives no information about some of the unknown states $|\vec{\xi}_k\rangle$ (when $\pi_k = 0$), but the target state should have some overlap on $|\vec{\xi}_k\rangle|k\rangle$ to satisfy the constraint on the success probability, therefore the problem is not solvable.

For the general case $p_{\min} \leq p \leq p_{\max}$, the upper bound comes from the algorithm in Lemma 5.12, and the matching lower bound is given in Lemma 5.11. □

5.4.3 Strong quantum rejection sampling algorithm

Let us now describe how the algorithm can be modified to solve the stronger problem $\text{SQSAMPLING}_{\boldsymbol{\tau}}$. The first modification follows from the observation that in the previous algorithm, the oracle is only used in two different ways: it is used once to create the state $|\boldsymbol{\pi}^{\vec{\xi}}\rangle$, and then only to reflect through that state. This means that we can still solve the problem if, instead of being given access to an oracle that maps $|\bar{0}\rangle_{dn}$ to $|\boldsymbol{\pi}^{\vec{\xi}}\rangle$, we are provided with one copy of $|\boldsymbol{\pi}^{\vec{\xi}}\rangle$ and an oracle that reflects through it.

In order to solve SQSAMPLING$_{\boldsymbol{\tau}}$, we should also be able to deal with the case where we only know the ratios between the amplitudes π_k and σ_k for each k, instead of the amplitudes themselves. As we will see, in that case we cannot use the algorithm given above anymore, as we do not know in advance how many steps of amplitude amplification are required. There are different approaches to solve this issue, one of them being to estimate q_ε, and therefore the required number of steps, by performing a phase estimation on the amplitude amplification operator (this is sometimes referred to as amplitude estimation or quantum counting, see [BBHT98, BHMT00]). Another option, also proposed by [BBHT98, BHMT00], is to repeat the algorithm successively with an increasing number of steps until it succeeds. One advantage of the first option would be that it provides an estimation of the initial acceptance probability q_ε, which might be useful for some applications. Since this is not required for SQSAMPLING$_{\boldsymbol{\tau}}$, we will rather describe an algorithm based on the second option, as it is more direct. Note that for both options, we need to adapt the algorithms in [BBHT98, BHMT00] as they require to use a fresh copy of the initial state after each failed attempt, whereas for SQSAMPLING$_{\boldsymbol{\tau}}$ we only have one copy of that state. This issue can be solved by using the state left over from the previous unsuccessful measurement instead of a fresh copy of the state. More precisely, we can use the following algorithm.

Strong quantum rejection sampling algorithm $\mathcal{A}_{\mathrm{SQRS}}(|\boldsymbol{\pi}^{\xi}\rangle, \varepsilon, c)$

1. Append an extra qubit prepared in the state $|0\rangle$ to the input state $|\boldsymbol{\pi}^{\xi}\rangle$, and apply $I_d \otimes R_\varepsilon$ on the resulting state.
2. Measure the last register of the current state. If the outcome is $|1\rangle$, output the first two registers and stop. Otherwise, set $l = 0$ and continue.
3. Let $T_l := \lceil c^l \rceil$ and pick an integer $t \in [T_l]$ uniformly at random.
4. Apply the amplitude amplification subroutine $\mathcal{S}_{\mathrm{QRS}}(\mathrm{ref}_{|\boldsymbol{\pi}^{\xi}\rangle|0\rangle}, \varepsilon, t)$.
5. Measure the last register of the current state. If the outcome is $|1\rangle$, output the first two registers and stop. Otherwise, increase l by one and go back to step 3.

Lemma 5.13. *For any $\alpha \geq 1$, there is a quantum algorithm that solves* SQSAMPLING$_{\boldsymbol{\tau}}$ *with success probability $p(\gamma)$ using an expected number of queries $O(1/\|\boldsymbol{\varepsilon}(\gamma)\|_2)$, where $\gamma = \alpha\|\boldsymbol{\pi} \circ \boldsymbol{\tau}\|_2$. In particular, for $\alpha = 1$ the expected number of queries is $O(1/\|\boldsymbol{\pi} \circ \boldsymbol{\tau}\|_2)$ and the success probability is equal to one.*

Here the parameter α allows us to control the trade-off between the success probability and the required number of queries. However, we cannot predict the actual values

of both quantities, because they depend on $\boldsymbol{\pi}$ and $\boldsymbol{\tau}$, but only $\boldsymbol{\tau}$ is known to us. The only exception is $\alpha = 1$, when the success probability is equal to one. Also, increasing α above $1/(\min_{k:\tau_k>0} \tau_k)$ will no longer affect the query complexity and success probability of the algorithm.

Proof. We show that for some choice of $c > 1$ and $\boldsymbol{\varepsilon}$ the algorithm $\mathcal{A}_{\mathrm{SQRS}}(|\boldsymbol{\pi}^{\xi}\rangle, \boldsymbol{\varepsilon}, c)$ described above solves the problem.

Let us first verify that we can actually perform all steps required in the algorithm. We need one copy of the state $|\boldsymbol{\pi}^{\xi}\rangle$, which is indeed provided as an input for the SQSAMPLING$_{\boldsymbol{\tau}}$ problem. Note that for applying R_{ε} in step 1 it suffices to know only the ratio ε_k/π_k (see Eq. (5.55)), which can be deduced from τ_k as follows. Let $\boldsymbol{\varepsilon} := r \cdot \boldsymbol{\varepsilon}(\gamma)$ for some $r < 1$, and recall from Definition 5.8 and Definition 5.7 that $\varepsilon_k(\gamma) = \min\{\pi_k, \gamma\sigma_k\}$ and $\sigma_k = \pi_k\tau_k/\|\boldsymbol{\pi} \circ \boldsymbol{\tau}\|_2$, respectively. Then

$$\frac{\varepsilon_k}{\pi_k} = r \min\left\{1, \gamma \frac{\sigma_k}{\pi_k}\right\} \tag{5.65}$$

$$= r \min\left\{1, \gamma \frac{\tau_k}{\|\boldsymbol{\pi} \circ \boldsymbol{\tau}\|_2}\right\} \tag{5.66}$$

$$= r \min\{1, \alpha\tau_k\}, \tag{5.67}$$

where we substituted $\gamma = \alpha\|\boldsymbol{\pi} \circ \boldsymbol{\tau}\|_2$ from the statement of the Lemma. Note that once r is chosen, the final expression in Eq. (5.67) is completely known. Finally, applying $\mathcal{S}_{\mathrm{QRS}}(\mathrm{ref}_{|\boldsymbol{\pi}^{\xi}\rangle|0\rangle}, \boldsymbol{\varepsilon}, t)$ in step 4 also requires the ability to apply R_{ε}, as well as $\mathrm{ref}_{|\boldsymbol{\pi}^{\xi}\rangle|0\rangle}$, which can be done by using one oracle query. Therefore, we have all we need to implement the algorithm.

We now show that the algorithm has success probability $p(\gamma)$. Recall from Eq. (5.57) that step 1 of the algorithm prepares the state

$$|\Psi_{\boldsymbol{\varepsilon}}\rangle = \sum_{k=1}^{n} |\xi_k\rangle|k\rangle\left(\sqrt{|\pi_k|^2 - \varepsilon_k^2}\,|0\rangle + \varepsilon_k|1\rangle\right) \tag{5.68}$$

$$= \sin\theta|\Psi_{\Pi,\boldsymbol{\varepsilon}}\rangle + \cos\theta|\Psi_{\Pi,\boldsymbol{\varepsilon}}^{\perp}\rangle, \tag{5.69}$$

where $\theta := \arcsin\|\boldsymbol{\varepsilon}\|_2$ and unit vectors

$$|\Psi_{\Pi,\boldsymbol{\varepsilon}}\rangle := \frac{1}{\sin\theta}\sum_{k=1}^{n} \varepsilon_k|\xi_k\rangle|k\rangle|1\rangle, \tag{5.70}$$

$$|\Psi_{\Pi,\boldsymbol{\varepsilon}}^{\perp}\rangle := \frac{1}{\cos\theta}\sum_{k=1}^{n} \sqrt{|\pi_k|^2 - \varepsilon_k^2}|\xi_k\rangle|k\rangle|0\rangle \tag{5.71}$$

are orthogonal and span a 2-dimensional subspace. In this subspace $\mathrm{ref}_{I_d \otimes I_n \otimes |1\rangle\langle 1|}$ and $\mathrm{ref}_{|\Psi_{\Pi,\varepsilon}\rangle}$ act in the same way, so each iteration of the amplitude amplification subroutine consists of a product of two reflections. Since both reflections preserve this subspace, $\mathcal{S}_{\mathrm{QRS}}(\mathrm{ref}_{|\pi^{\tilde{\xi}}\rangle|0\rangle}, \varepsilon, t)$ corresponds to a rotation by angle $2t\theta$ in this subspace. Measurements in step 2 and step 5 either project on $|\Psi_{\Pi,\varepsilon}\rangle$, when the outcome is $|1\rangle$, or on $|\Psi^{\perp}_{\Pi,\varepsilon}\rangle$, when the outcome is $|0\rangle$. Therefore, the algorithm always outputs the first two registers of the state $|\Psi_{\Pi,\varepsilon}\rangle$, and by Eq. (5.60), the success probability is $p_\varepsilon = p(\gamma)$, as claimed. In particular, for $\alpha = 1$ from Eq. (5.67) we get $\varepsilon_k = r\pi_k\tau_k$ as $\tau_k \leq 1$. Thus, $\varepsilon = r(\pi \circ \tau)$ and since $\sigma = (\pi \circ \tau)/\|\pi \circ \tau\|_2$, we get $p_\varepsilon = (\sigma^{\mathsf{T}} \cdot \varepsilon/\|\varepsilon\|_2)^2 = 1$.

Let us now bound the expected number of oracle queries. We follow the proof of Theorem 3 in [BHMT00], but there is an important difference: a direct analogue of the algorithm in [BHMT00, Theorem 3] would use a fresh copy of $|\pi^{\tilde{\xi}}\rangle$ each time the measurement fails to give a successful outcome, whereas in this algorithm we start from the state left over from the previous measurement, since we only have one copy of $|\pi^{\tilde{\xi}}\rangle$. Note that $\mathcal{S}_{\mathrm{QRS}}(\mathrm{ref}_{|\pi^{\tilde{\xi}}\rangle|0\rangle}, \varepsilon, t)$ in step 4 is always applied on $|\Psi^{\perp}_{\Pi,\varepsilon}\rangle$, since it is the post-measurement state corresponding to the unsuccessful outcome. Therefore, the state created by step 4 is $\sin(2t\theta)|\Psi_{\Pi,\varepsilon}\rangle + \cos(2t\theta)|\Psi^{\perp}_{\Pi,\varepsilon}\rangle$, and the next measurement will succeed with probability $\sin^2(2t\theta)$. Since t is picked uniformly at random between 1 and T_l, the probability that the l-th measurement fails is

$$p_l = \frac{1}{T_l}\sum_{t=1}^{T_l}\cos^2(2t\theta) \tag{5.72}$$

$$= \frac{1}{2} + \frac{1}{2T_l}\sum_{t=1}^{T_l}\cos(4t\theta) \tag{5.73}$$

$$\leq \frac{1}{2} + \frac{1}{2T_l\|\varepsilon\|_2}, \tag{5.74}$$

where the upper bound is obtained as follows:

$$\sum_{t=1}^{T} \cos(4t\theta) = \mathrm{Re}\left(e^{i4\theta} \sum_{t=0}^{T-1} e^{i4t\theta} \right) \tag{5.75}$$

$$= \mathrm{Re}\left(e^{i4\theta} \cdot \frac{1 - e^{i4T\theta}}{1 - e^{i4\theta}} \right) \tag{5.76}$$

$$= \mathrm{Re}\left(e^{i2(T+1)\theta} \cdot \frac{e^{-i2T\theta} - e^{i2T\theta}}{e^{-i2\theta} - e^{i2\theta}} \right) \tag{5.77}$$

$$= \cos\big(2(T+1)\theta\big) \frac{\sin(2T\theta)}{\sin(2\theta)} \tag{5.78}$$

$$\leq \frac{1}{\sin(2\theta)} \leq \frac{1}{\sin\theta} = \frac{1}{\|\boldsymbol{\varepsilon}\|_2}, \tag{5.79}$$

where we forced the last inequality by picking $r := \sqrt{3}/2$, so that $\sin\theta = \|\boldsymbol{\varepsilon}\|_2 \leq \sqrt{3}/2$ and thus $0 \leq \theta \leq \pi/3$. Recall from the algorithm that $T_l = \lceil c^l \rceil$ for some $c > 1$, so it is increasing and goes to infinity as l increases. Let $\bar{T} := 1/(2\Delta\|\boldsymbol{\varepsilon}\|_2)$ for some $\Delta > 0$ and let $\bar{l}$ be the smallest integer such that $T_l \geq \bar{T}$ for all $l \geq \bar{l}$. Then according to Eq. (5.74) we get that $p_l \leq 1/2 + 1/(2\bar{T}\|\boldsymbol{\varepsilon}\|_2) = 1/2 + \Delta =: \bar{p}$ for all $l \geq \bar{l}$. Note that the l-th execution of the subroutine uses at most $2T_l$ oracle queries, so the expected number of oracle calls is at most $2T_0 + p_0(2T_1 + p_1(2T_2 + \ldots))$. This can be upper bounded by

$$\sum_{l=0}^{\bar{l}} 2T_l + \sum_{d=1}^{\infty} 2T_{\bar{l}+d}\bar{p}^d = \sum_{l=0}^{\bar{l}} 2\lceil c^l \rceil + \sum_{d=1}^{\infty} 2\lceil c^{\bar{l}+d} \rceil \bar{p}^d \tag{5.80}$$

$$\leq 4\left(\sum_{l=0}^{\bar{l}} c^l + c^{\bar{l}} \sum_{d=1}^{\infty} (c\bar{p})^d \right) \tag{5.81}$$

$$= 4\left(\frac{c^{\bar{l}+1} - 1}{c - 1} + c^{\bar{l}} \frac{c\bar{p}}{1 - c\bar{p}} \right) \tag{5.82}$$

$$\leq 4c^{\bar{l}+1}\left(\frac{1}{c-1} + \frac{\bar{p}}{1 - c\bar{p}} \right) \tag{5.83}$$

$$\leq \frac{2c^2}{\Delta\|\boldsymbol{\varepsilon}\|_2}\left(\frac{1}{c-1} + \frac{\bar{p}}{1 - c\bar{p}} \right), \tag{5.84}$$

where the first and last inequality is obtained from the following two observations, respectively:

1. $\lceil c^l \rceil = c^l + \delta$ for some $0 \leq \delta < 1$, so $T_l = \lceil c^l \rceil < c^l + 1 < 2c^l$ as $c > 1$.
2. $c^{\bar{l}+1} \leq c^2 \lceil c^{\bar{l}-1} \rceil = c^2 T_{\bar{l}-1} < c^2 \bar{T} = c^2/(2\Delta \|\boldsymbol{\varepsilon}\|_2)$ by the choice of $\bar{l}$.

Finally, we have to make a choice of $c > 1$ and $\Delta > 0$, so that the geometric series in Eq. (5.81) converges, *i.e.*, $c\bar{p} < 1$ or equivalently $c < 2/(1 + 2\Delta)$. By choosing $c := 8/7$ and $\Delta := 1/4$ we minimize the upper bound in Eq. (5.84) and obtain $128/\|\boldsymbol{\varepsilon}\|_2 = O(1/\|\boldsymbol{\varepsilon}(\gamma)\|_2)$. In particular, for $\alpha = 1$ this becomes $O(1/\|\boldsymbol{\pi} \circ \boldsymbol{\tau}\|_2)$. □

5.5 Applications

5.5.1 Linear systems of equations

As a first example of application, we show that quantum rejection sampling was implicitly used in the quantum algorithm for linear systems of equations proposed by Harrow, Hassidim, and Lloyd [HHL09]. This algorithm solves the following quantum state generation problem: given the classical description of an invertible $d \times d$ matrix A and a unit vector $|b\rangle \in \mathbb{C}^d$, prepare the quantum state $|x\rangle/\||x\rangle\|_2$, where $|x\rangle$ is the solution of the linear system of equations $A|x\rangle = |b\rangle$. As shown in [HHL09], we can assume without loss of generality that A is Hermitian. Similarly to classical matrix inversion algorithms, an important factor of the performance of the algorithm is the condition number κ of A, which is the ratio between the largest and smallest eigenvalue of A. We will assume that all eigenvalues of A are between κ^{-1} and 1, and we denote by $|\psi_j\rangle$ and λ_j the eigenvectors and eigenvalues of A, respectively. We also define[3] the amplitudes $b_j := \langle \psi_j | b \rangle$, so that $|b\rangle = \sum_{j=1}^d b_j |\psi_j\rangle$. Then, the problem is to prepare the state $|x\rangle := A^{-1}|b\rangle = \sum_{j=1}^d b_j \lambda_j^{-1} |\psi_j\rangle$ (up to normalization).

We now show how this problem reduces to the quantum state conversion problem SQSAMPLING$_{\boldsymbol{\tau}}$. Since A is Hermitian, we can use Hamiltonian simulation techniques [BACS05, Chi08, CK11] to simulate the unitary operator e^{iAt} on any state. Using quantum phase estimation [Kit95, CEMM98] on the operator e^{iAt}, we can implement an operator E_A that acts in the eigenbasis of A as $E_A : |\psi_j\rangle|\bar{0}\rangle \mapsto |\psi_j\rangle|\lambda_j\rangle$, where $|\lambda_j\rangle$ is a quantum state encoding an approximation of the eigenvalue λ_j. Here, we will assume that this can be done exactly, that is, we assume that $|\lambda_j\rangle$ is a computational basis state that exactly encodes λ_j (we refer the reader to [HHL09] for a detailed analysis of the error introduced by this approximation). Under this assumption, the problem reduces

[3] We choose the global phase of each eigenvector $|\psi_j\rangle$ so that b_j is real and non-negative.

to a quantum state conversion problem that we will call QLINEAREQS$_\kappa$. Its definition requires fixing a set of possible eigenvalues $\Lambda_\kappa \subset [\kappa^{-1}, 1]$ of finite cardinality $n := |\Lambda_\kappa|$. Let us denote the set of $d \times d$ Hermitian matrices by $\mathrm{Herm}(d)$ and the eigenvalues of A by $\mathrm{spec}(A)$.

Definition 5.14 (Quantum linear system of equations)**.** QLINEAREQS$_\kappa$, the *quantum linear system of equations problem* is a quantum state conversion problem $(\mathcal{O}, \Phi, \Psi, \mathcal{X})$ with $\mathcal{X} := \{(|b\rangle, A) \in \mathbb{C}^d \times \mathrm{Herm}(d) : \mathrm{spec}(A) \in \Lambda_\kappa^d\}$, oracles in $\mathcal{O}$ being pairs $(O_{|b\rangle,A}, E_A)$ where $O_{|b\rangle,A} := \mathrm{ref}_{|b\rangle}$ and E_A acts as $E_A : |\psi_j\rangle|\bar{0}\rangle_n \mapsto |\psi_j\rangle|\lambda_j\rangle$ where $|\psi_j\rangle$ are the eigenvectors of A, and the corresponding initial and target states being $|b\rangle$ and $A^{-1}|b\rangle / \|A^{-1}|b\rangle\|_2$.

Using Lemma 5.13 we can prove the following result.

Theorem 5.2. *For any $\tilde{\kappa} \in [1, \kappa]$, there is a quantum algorithm that solves* QLINEAREQS$_\kappa$ *with success probability $p = (\boldsymbol{w}^\mathsf{T} \cdot \tilde{\boldsymbol{w}}) / (\|\boldsymbol{w}\|_2 \cdot \|\tilde{\boldsymbol{w}}\|_2)$ using an expected number of queries $O(\tilde{\kappa} / \|\tilde{\boldsymbol{w}}\|_2)$, where $w_j := b_j / \lambda_j$, $\tilde{w}_j := b_j / \tilde{\lambda}_j$, and $\tilde{\lambda}_j := \max\{\tilde{\kappa}^{-1}, \lambda_j\}$.*

Proof. Following [HHL09], the algorithm for this problem consists of three steps:

1. Apply the phase estimation operation E_A on $|b\rangle$ to obtain the state $\sum_{j=1}^d b_j |\psi_j\rangle|\lambda_j\rangle$.
2. Convert this state to $\sum_{j=1}^d w_j |\psi_j\rangle|\lambda_j\rangle / \|\boldsymbol{w}\|_2$.
3. Undo the operation E_A to obtain the target state $\sum_{j=1}^d w_j |\psi_j\rangle / \|\boldsymbol{w}\|_2$.

We see that step 2 is an instance of SQSAMPLING$_{\boldsymbol{\tau}}$, where the basis states $\{|\lambda\rangle : \lambda \in \Lambda_\kappa\}$ of the phase estimation register play the role of the states $|k\rangle$ and the vector $\boldsymbol{\tau}$ of the ratios between the initial and final amplitudes is given by $\tau_\lambda := (\kappa\lambda)^{-1}$ (here the normalization factor κ is to make sure that $\max_\lambda \tau_\lambda = 1$). The rest of the reduction is summarized in Table 5.1. Therefore, we can use the algorithm from Lemma 5.13 to perform step 2.

If we set $\alpha := \kappa / \tilde{\kappa}$ then from Table 5.1 we get

$$\begin{aligned} \varepsilon_{\lambda_j}(\gamma) &= \pi_{\lambda_j} \min\{1, \alpha\tau_{\lambda_j}\} && (5.85)\\ &= b_j \min\{1, (\tilde{\kappa}\lambda_j)^{-1}\} && (5.86)\\ &= \frac{b_j}{\tilde{\kappa} \max\{\tilde{\kappa}^{-1}, \lambda_j\}} && (5.87)\\ &= \frac{\tilde{w}_j}{\tilde{\kappa}}, && (5.88) \end{aligned}$$

$\text{SQSAMPLING}_{\boldsymbol{\tau}}$	$\text{QLINEAREQS}_{\kappa}$
$\lvert k\rangle$	$\lvert\lambda\rangle$
π_k	$\pi_\lambda := \begin{cases} b_j & \text{if } \lambda = \lambda_j \in \operatorname{spec}(A) \\ 0 & \text{if } \lambda \notin \operatorname{spec}(A) \end{cases}$
$\lvert\xi_k\rangle$	$\lvert\xi_\lambda\rangle := \begin{cases} \lvert\psi_j\rangle & \text{if } \lambda = \lambda_j \in \operatorname{spec}(A) \\ \text{n/a} & \text{if } \lambda \notin \operatorname{spec}(A) \end{cases}$
τ_k	$\tau_\lambda := (\kappa\lambda)^{-1}$

Table 5.1: Reduction from step 2 in the linear system of equations algorithm to the quantum resampling problem $\text{SQSAMPLING}_{\boldsymbol{\tau}}$.

thus $\boldsymbol{\varepsilon}(\gamma) = \tilde{\boldsymbol{w}}/\tilde{\kappa}$ and the expected number of queries is $O(1/\|\boldsymbol{\varepsilon}(\gamma)\|_2) = O(\tilde{\kappa}/\|\tilde{\boldsymbol{w}}\|_2)$. Recall that the amplitudes of the target state are given by $\sigma_j = w_j/\|\boldsymbol{w}\|_2$, so $\sigma = \boldsymbol{w}/\|\boldsymbol{w}\|_2$ and the success probability is

$$p = \left(\frac{\boldsymbol{\sigma}^\mathsf{T} \cdot \boldsymbol{\varepsilon}(\gamma)}{\|\boldsymbol{\varepsilon}(\gamma)\|_2}\right)^2 = \frac{\boldsymbol{w}^\mathsf{T}}{\|\boldsymbol{w}\|_2} \cdot \frac{\tilde{\boldsymbol{w}}}{\|\tilde{\boldsymbol{w}}\|_2} \tag{5.89}$$

as claimed. □

Note that even though we have a freedom to choose $\tilde{\kappa}$, we cannot predict the query complexity in advance, since it depends on $\tilde{w}_j = \langle\psi_j|b\rangle/\tilde{\lambda}_j$, which in turn is determined by the lengths of projections of $|b\rangle$ in the eigenspaces of A, weighted by the corresponding truncated eigenvalues $\tilde{\lambda}_j$. Similarly, we cannot predict the success probability p. However, by choosing $\tilde{\kappa} = \kappa$ we can at least make sure that $p = 1$ (since $\tilde{\lambda}_j = \lambda_j$ and thus $\tilde{\boldsymbol{w}} = \boldsymbol{w}$). In this case step 2 is performed exactly (assuming an ideal phase estimation black box) and the expected number of queries is $O(\kappa/\|\boldsymbol{w}\|_2)$. By noting that $\lambda_j \leq 1$ for all j, we see that $\|\boldsymbol{w}\|_2^2 = \sum_{j=1}^d b_j^2 \lambda_j^{-2} \geq 1$ and thus we can put a cruder upper bound of $O(\kappa)$, which coincides with the bound given in [HHL09] for that step of the algorithm. For ill-conditioned matrices, *i.e.*, matrices with a high condition number κ, the approach taken by [HHL09] is to ignore small eigenvalues $\lambda_j \leq \tilde{\kappa}^{-1}$, for some cut-off $\tilde{\kappa}^{-1} \geq \kappa^{-1}$, which reduces the cost of the algorithm to $O(\tilde{\kappa})$, but introduces some extra error. In our case, by choosing $\alpha = \kappa/\tilde{\kappa}$ we obtained bound $O(\tilde{\kappa}/\|\tilde{\boldsymbol{w}}\|_2)$, where

$\|\tilde{w}\|_2 \geq 1$. Again, here $\tilde{w}$ depends on additional structure of the problem and cannot be predicted beforehand.

In practical applications, we will of course not be given access to the ideal phase estimation operator E_A, but we can still approximate it by using the phase estimation algorithm [Kit95, CEMM98] on the operator A. It is shown in [HHL09] that if A is s-sparse, this approximation can be implemented with sufficient accuracy at a cost $\tilde{O}\big(\log(d)s^2\tilde{\kappa}/\epsilon\big)$, where ϵ is the overall additive error introduced by this approximation throughout the algorithm. Therefore, the total cost of the algorithm is at most $\tilde{O}\big(\log(d)s^2\tilde{\kappa}^2/\epsilon\big)$ (see [HHL09] for details).

5.5.2 Quantum Metropolis sampling

Since rejection sampling lies at the core of the (classical) Metropolis algorithm, it seems natural to use quantum rejection sampling to solve the corresponding problem in the quantum case. The quantum Metropolis sampling algorithm presented in [TOV+11] follows the same lines as the classical algorithm by setting up a (classical) random walk between eigenstates of the Hamiltonian, where each move is either accepted or rejected depending on the value of some random coin. The main complication compared to the classical version comes from the case where the move has to be rejected, since we cannot keep a copy of the previous eigenstate due to the no-cloning theorem. The solution proposed by Temme *et al.* [TOV+11] is to use an unwinding technique based on successive measurements to revert to the original state. Here, we show that quantum rejection sampling can be used to avoid this step, as it allows to amplify the amplitude of the "accept" state of the coin register, effectively eliminating rejected moves. This yields a more efficient algorithm as it eliminates the cost of reverting rejected moves and provides a quadratic speed-up on the overall cost of obtaining an accepted move.[4]

Before describing in more details how quantum rejection sampling can be used to design a new quantum Metropolis algorithm, let us recall how the standard (classical) Metropolis algorithm works [MRR+53]. The goal is to solve the following problem: given a classical Hamiltonian associating energies E_j to a set of possible configurations j, sample from the Gibbs distribution $p(j) = \exp(-\beta E_j)/Z(\beta)$, where β is the inverse temperature and $Z(\beta) = \sum_j \exp(-\beta E_j)$ is the partition function. Since the size of the

[4] Martin Schwarz has pointed out to us that this is similar to how [NWZ09] provides a speed-up over [MW05], and that our technique can also be used to speed-up the quantum algorithm in [STV11] for preparing PEPS.

configuration space is exponential in the number of particles, estimating the Gibbs distribution itself is not an option, therefore the Metropolis algorithm proposes to solve this problem by setting up a random walk on the set of configurations that converges to the Gibbs distribution. More precisely, the random walk works as follows:

1. If i is the current configuration with energy E_i, choose a random move to another configuration j (*e.g.*, for a system of spins, a random move could consist in flipping a random spin), and compute the associated energy E_j.
2. The random move is then accepted or rejected according to the following rule:
   - if $E_j \leq E_i$, then the move is always accepted;
   - if $E_j > E_i$, then the move is only accepted with probability $\exp\big(\beta(E_i - E_j)\big)$.

It can be shown that this random walk converges to the Gibbs distribution.

The quantum Metropolis sampling algorithm by Temme *et al.* [TOV+11] follows the same general lines as the classical algorithm. It aims at solving the equivalent problem in the quantum case, where we need to generate the thermal state of a Hamiltonian H, that is, we need to generate a random eigenstate $|\psi_j\rangle$ where j is sampled according to the Gibbs distribution. The fact that the Hamiltonian is quantum, however, adds a few obstacles, since the set of eigenstates $|\psi_j\rangle$ is not known to start with. The main tool to overcome this difficulty is to use quantum phase estimation [Kit95, CEMM98] which, applied on the Hamiltonian H, allows to project any state on an eigenstate $|\psi_j\rangle$, while obtaining an estimate of the corresponding eigenenergy E_j. Similarly to the previous section, we will assume for simplicity that this can be done exactly, that is, we have access to a quantum circuit that acts in the eigenbasis of H as $E_H : |\psi_j\rangle|\bar{0}\rangle \mapsto |\psi_j\rangle|E_j\rangle$, where $|E_j\rangle$ exactly encodes the eigenenergy E_j. We will also assume that the eigenenergies of H are nondegenerate, so that each eigenenergy E_j corresponds to a single eigenstate $|\psi_j\rangle$, instead of a higher dimensional eigenspace. The quantum Metropolis sampling algorithm also requires to choose a set of quantum gates $\mathcal{C}$ that will play the role of the possible random moves between eigenstates. In this case, a given quantum gate $C_l \in \mathcal{C}$ will not simply move an initial eigenstate $|\psi_i\rangle$ to another eigenstate $|\psi_j\rangle$, but rather to a superposition $C_l|\psi_i\rangle = \sum_j c_{ij}^{(l)}|\psi_j\rangle$ where $c_{ij}^{(l)} := \langle\psi_j|C_l|\psi_i\rangle$.

We can now give a high-level description of the quantum Metropolis sampling algorithm by Temme *et al.* [TOV+11]. Let $|\psi_i\rangle|E_i\rangle$ be an initial state, that can be prepared by applying the phase estimation operator E_H on an arbitrary state, and measuring the energy register. The algorithm implements each random move by performing the following steps:

1. Apply a random gate $C_l \in \mathcal{C}$ on the first register to prepare the state $\big(C_l|\psi_i\rangle\big)|E_i\rangle = \sum_j c_{ij}^{(l)}|\psi_j\rangle|E_i\rangle$.
2. Apply the phase estimation operator E_H on the $|\psi_j\rangle$ register and an ancilla register initialized in the default state $|\bar{0}\rangle$ to prepare the state $\sum_j c_{ij}^{(l)}|\psi_j\rangle|E_i\rangle|E_j\rangle$.
3. Add another ancilla qubit prepared in the state $|0\rangle$ and apply a controlled-rotation on this register to create the state $\sum_j c_{ij}^{(l)}|\psi_j\rangle|E_i\rangle|E_j\rangle\left[\sqrt{f_{ij}}|1\rangle + \sqrt{1-f_{ij}}|0\rangle\right]$, where $f_{ij} := \min\{1, \exp(\beta(E_i - E_j))\}$.
4. Measure the last qubit. If the outcome is 0, reject the move by reverting the state to $|\psi_i\rangle|E_i\rangle$ (see [TOV+11] for details) and go back to step 1. Otherwise, continue.
5. Discard the $|E_i\rangle$ register and measure the $|E_j\rangle$ register to project the state onto a new eigenstate $|\psi_j\rangle|E_j\rangle$.

It is shown in [TOV+11] that by choosing a universal set of quantum gates for the set of moves $\mathcal{C}$, the algorithm simulates random walk on the set of eigenstates of H that satisfies a quantum detailed balanced condition, which ensures that the walk converges to the Gibbs distribution, as in the classical case.

For a given initial state $|\psi_i\rangle|E_i\rangle$, the probability (over all choices of the randomly chosen gate C_l) that the measurement in step 4 succeeds is $\frac{1}{|\mathcal{C}|}\sum_{j,l} f_{ij}|c_{ij}^{(l)}|^2$. If we define a vector $\boldsymbol{w}^{(i)}$ whose components are

$$w_{jl}^{(i)} := \sqrt{\frac{f_{ij}}{|\mathcal{C}|}}\, c_{ij}^{(l)}, \tag{5.90}$$

then this probability is simply $\|\boldsymbol{w}^{(i)}\|_2^2$. Hence, after one execution of the algorithm (step 1 – step 5) the initial state $|\psi_i\rangle|E_i\rangle$ gets mapped to state $|\psi_j\rangle|E_j\rangle$ with probability $\sum_l |w_{jl}^{(i)}|^2/\|\boldsymbol{w}^{(i)}\|_2^2$. We could achieve the same random move by converting the initial state $|\psi_i\rangle|E_i\rangle$ to

$$\sum_{j,l}\frac{w_{jl}^{(i)}}{\|\boldsymbol{w}^{(i)}\|_2}|l\rangle|\psi_j\rangle|E_i\rangle|E_j\rangle = \frac{1}{\|\boldsymbol{w}^{(i)}\|_2}\sum_j\sqrt{\frac{f_{ij}}{|\mathcal{C}|}}\left[\sum_l c_{ij}^{(l)}|l\rangle\right]|\psi_j\rangle|E_i\rangle|E_j\rangle \tag{5.91}$$

and discarding the $|l\rangle$ and $|E_i\rangle$ registers and measuring the $|E_j\rangle$ register to project on the state $|\psi_j\rangle|E_j\rangle$ with the correct probability. This implies that one random move reduces to a quantum state conversion problem that we will call QMMOVE$_{\mathcal{C}}$. This problem assumes that we are able to perform a perfect phase estimation on the Hamiltonian H.

Therefore, similarly to the previous section, we fix a set of possible eigenenergies $\mathcal{E}$ of finite cardinality $n := |\mathcal{E}|$.

Definition 5.15 (Quantum Metropolis move)**.** The *quantum Metropolis move problem*, denoted by QMMOVE$_\mathcal{C}$, is a quantum state conversion problem $(\mathcal{O}, \Phi, \Psi, \mathcal{X})$, where

$$\mathcal{X} := \{(H, i) \in \mathrm{Herm}(d) \times [d] : \mathrm{spec}(H) \in \mathcal{E}^d\}. \tag{5.92}$$

Oracles in $\mathcal{O}$ act as $E_H : |\psi_j\rangle|\bar{0}\rangle_n \mapsto |\psi_j\rangle|E_j\rangle$, with the corresponding initial states $|\psi_i\rangle$, the eigenvectors of H, and target states $\sum_{j,l} w_{jl}^{(i)} / \|\boldsymbol{w}^{(i)}\|_2 |l\rangle|\psi_j\rangle$ where

$$w_{jl}^{(i)} := \sqrt{\frac{f_{ij}}{|\mathcal{C}|}}\, c_{ij}^{(l)}, \qquad f_{ij} := \min\{1, \exp(\beta(E_i - E_j))\}, \qquad c_{ij}^{(l)} := \langle\psi_j|C_l|\psi_i\rangle. \tag{5.93}$$

A critical part of the algorithm from [TOV+11] described above is how to revert a rejected move in step 4. Temme *et al.* show how this can be done by using an unwinding technique based on successive measurements, but we will not describe this technique in detail, as we now show how this step can be avoided by using quantum rejection sampling. Intuitively, this can be done by using amplitude amplification to ensure that the measurement in step 4 always projects on the "accept" state $|1\rangle$. This also avoids having to repeatedly attempt random moves until one is accepted, and the number of steps of amplitude amplification will be quadratically smaller than the number of random moves that have to be attempted until one is accepted. This leads to the following statement:

Theorem 5.3. *There is a quantum algorithm that solves* QMMOVE$_\mathcal{C}$ *with success probability 1 using an expected number of queries* $O(1/\|\boldsymbol{w}^{(i)}\|_2)$.

Proof. The modified algorithm follows the same lines as the original algorithm, except that step 3 – step 4 is replaced by a quantum rejection sampling step. We use a quantum coin to choose the random gate in step 1 in order to make it coherent. The algorithm starts by applying the phase estimation oracle E_H on the initial state to prepare the state $|\psi_i\rangle|E_i\rangle$, and then proceeds with the following steps:

1. Prepare an extra register in the state $\frac{1}{\sqrt{|\mathcal{C}|}} \sum_l |l\rangle$. Conditionally on this register, apply the gate $C_l \in \mathcal{C}$ on the eigenstate register to prepare the state

$$\frac{1}{\sqrt{|\mathcal{C}|}} \sum_j \left[\sum_l c_{ij}^{(l)} |l\rangle\right] |\psi_j\rangle|E_i\rangle. \tag{5.94}$$

2. Apply the phase estimation operator E_H on the second register and an ancilla register initialized in the default state $|\bar{0}\rangle_n$ to prepare the state

$$\frac{1}{\sqrt{|\mathcal{C}|}} \sum_j \left[\sum_l c_{ij}^{(l)} |l\rangle \right] |\psi_j\rangle |E_i\rangle |E_j\rangle. \tag{5.95}$$

3. Convert this state to the state given in Eq. (5.91):

$$\frac{1}{\|\boldsymbol{w}^{(i)}\|_2} \sum_j \sqrt{\frac{f_{ij}}{|\mathcal{C}|}} \left[\sum_l c_{ij}^{(l)} |l\rangle \right] |\psi_j\rangle |E_i\rangle |E_j\rangle. \tag{5.96}$$

4. Discard $|E_i\rangle$ and uncompute $|E_j\rangle$ by using one call to the phase estimation oracle $E_H^\dagger$.

Note that step 3 is an instance of SQSAMPLING$_{\boldsymbol{\tau}}$, where the pair of basis states $|E\rangle|E'\rangle$ of the phase estimation registers plays the role of the states $|k\rangle$, the initial amplitudes $\pi_{E,E'}$ are given by $\frac{1}{\sqrt{|\mathcal{C}|}} \sqrt{\sum_l |c_{ij}^{(l)}|^2}$ for $(E, E') = (E_i, E_j)$ or 0 for values (E, E') that do not correspond to a pair of eigenvalues of H, the states $\left[\sum_l c_{ij}^{(l)} |l\rangle\right] |\psi_j\rangle / \sqrt{\sum_l |c_{ij}^{(l)}|^2}$ play the role of the unknown states $|\xi_k\rangle$, and the ratio between the initial and target amplitudes is given by $\tau_{E,E'} = \sqrt{\min\{1, \exp(\beta(E - E'))\}}$ (the reduction is summarized in Table 5.2). Therefore, this step may be performed using the algorithm in Lemma 5.13. Here, we choose $\alpha = 1$ since the full Quantum Metropolis Sampling algorithm requires to apply a large number of successive random moves, therefore each instance of QMMOVE$_{\mathcal{C}}$ should be solved with high success probability. Choosing $\alpha = 1$ ensures that each random move will have success probability 1 (under our assumption that the phase estimation oracle is perfect), using an expected number of phase estimation oracles $O(1/\|\boldsymbol{w}^{(i)}\|_2)$. □

Note that in this case it is critical that the algorithm only requires one copy of the initial state, hence solving the quantum state conversion problem SQSAMPLING$_{\boldsymbol{\tau}}$ (in contrast, the quantum algorithm for linear systems of equations used a unitary to create multiple copies of the initial state, which is allowed only in the weaker quantum state generation problem QSAMPLING$_{\boldsymbol{\pi} \to \boldsymbol{\sigma}}$). Indeed, creating the initial state requires one copy of the previous eigenstate $|\psi_i\rangle$, which cannot be cloned as it is unknown. Here, the algorithm only requires to reflect through the initial state, which can be done by inverting step 1 and step 2, applying a phase "-1" conditionally on the eigenenergy

$\text{SQSAMPLING}_{\boldsymbol{\tau}}$	$\text{QMMOVE}_{\mathcal{C}}$
$\lvert k\rangle$	$\lvert E\rangle\lvert E'\rangle$
π_k	$\pi_{E,E'} := \begin{cases} \frac{1}{\sqrt{\lvert\mathcal{C}\rvert}}\sqrt{\sum_l \lvert c_{ij}^{(l)}\rvert^2} & \text{if } (E,E') = (E_i,E_j) \text{ where } E_i, E_j \in \text{spec}(H) \\ 0 & \text{if } E \notin \text{spec}(H) \text{ or } E' \notin \text{spec}(H) \end{cases}$
$\lvert\xi_k\rangle$	$\lvert\xi_{E,E'}\rangle := \begin{cases} \frac{1}{\sqrt{\sum_l \lvert c_{ij}^{(l)}\rvert^2}} \sum_l c_{ij}^{(l)} \lvert l\rangle\lvert\psi_j\rangle & \text{if } (E,E') = (E_i,E_j) \text{ where } E_i, E_j \in \text{spec}(H) \\ \text{n/a} & \text{if } E \notin \text{spec}(H) \text{ or } E' \notin \text{spec}(H) \end{cases}$
τ_k	$\tau_{E,E'} := \sqrt{\min\{1, \exp(\beta(E-E'))\}}$

Table 5.2: Reduction from step 3 in the new quantum Metropolis sampling algorithm to the quantum resampling problem $\text{SQSAMPLING}_{\boldsymbol{\tau}}$.

register being in the state $\lvert E_i\rangle$ (which is possible since E_i is known), and applying step 1 and step 2 again.

Repeating the algorithm for $\text{QMMOVE}_{\mathcal{C}}$ a large number of times will simulate the same random walk on the eigenstates of H as the original quantum Metropolis sampling algorithm in [TOV+11], except that we have a quadratic speed-up over the number of attempted moves necessary to obtain an accepted move. In order to converge to the Gibbs distribution, we need to take into account this quadratic speed-up in order to decide when to stop the algorithm, effectively assuming that each move takes quadratically longer than it actually does. Another option would be to modify the algorithm for $\text{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ so that it also estimates $\|\boldsymbol{w}^{(i)}\|_2$ by using amplitude estimation or quantum counting [BBHT98, BHMT00]. We leave the full analysis of these technical issues for future work.

5.5.3 Boolean function hidden shift problem

Our final application of quantum rejection sampling is a new quantum algorithm for the Boolean function hidden shift problem. Here we only give a brief overview of the results. See Sect. 6.4.2 in Chapter 6 for more details.

Definition 5.16. The *Boolean function hidden shift problem* (BFHSP) for function f, denoted by BFHSP_f, is defined as follows. Let $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ be a Boolean function, which is assumed to be completely known. Furthermore, we are given oracle access to *shifted function* $f_s(x) := f(x+s)$ for some unknown bit string $s \in \mathbb{Z}_2^n$, with the promise that there exists x such that $f(x+s) \neq f(x)$. The task is to find the bit string s.

In Sect. 6.4.2 we reduce this problem to solving the $\mathrm{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ problem for $\boldsymbol{\pi}$ corresponding to the Fourier spectrum of f, and $\boldsymbol{\sigma}$ being a uniformly flat vector. This leads to the following upper bound which expresses the complexity of our quantum algorithm for BFHSP_f in terms of the "water-filling" vector $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ (see Definition 5.8) of the Fourier spectrum of f.

Theorem 5.17. *Let f be a Boolean function and $\hat{F}$ be its Fourier transform. For any success probability $p \in [p_{\min}, p_{\max}]$, we have $Q_{1-p}(\mathrm{BFHSP}_f) = O(1/\|\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}\|_2)$, where $p_{\min}$, $p_{\max}$, and $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ are given in Definition 5.8, and components of $\boldsymbol{\pi}$ and $\boldsymbol{\sigma}$ are given by $\pi_w = |\hat{F}(w)|$ and $\sigma_w = 1/\sqrt{2^n}$ for $w \in \mathbb{Z}_2^n$.*

As special cases of this theorem we obtain the quantum algorithms for hidden shift problem for delta functions, which leads to the Grover search algorithm [Gro96], and for bent functions, which are functions that have perfectly flat absolute values in their Fourier spectrum [Röt10].

In general, the complexity of the algorithm is limited by the smallest Fourier coefficient of the function. By ignoring small Fourier coefficients, we can decrease the complexity of the algorithm, at the cost of a lower success probability. The final success probability can nevertheless be amplified using either parallel queries or repetition. For more details see Sect. 6.4.2 in Chapter 6.

5.6 Conclusion and open problems

We provide an algorithm for solving the quantum resampling problem. Our algorithm can be viewed as a quantum version of the classical rejection sampling technique. It relies on amplitude amplification [BHMT00] to increase the amplitude of some target "accept" state, and its query complexity is given by a semidefinite program. The solution of this SDP and hence the cost of the algorithm depends on the ratio between the amplitudes of the initial and target states, similarly to the case of the classical rejection sampling where the cost is given by the ratio of probabilities. Using the automorphism

principle over a unitary group, we derive an SDP for the lower bound that is identical to the one for the upper bound, showing that our algorithm has optimal query complexity.

While the original adversary method cannot be applied as is for this quantum state generation problem because the oracle encodes an unknown quantum state instead of some unknown classical data, it is interesting to note that the query complexity of this problem is also characterized by an SDP. Therefore, an interesting open question is whether the adversary method [Amb00, HLŠ07], which has been shown to be tight for evaluating functions [Rei09, Rei11, LMR+11] and nearly tight for quantum state generation or conversion problems with classical oracles [LMR+11], can always be extended and shown to be tight for this more general framework of problems with quantum oracles.

In Sect. 5.5, we illustrate how quantum rejection sampling may be used as a primitive in quantum algorithm design by providing three different applications. We first show that it was used implicitly in the quantum algorithm for linear systems of equations [HHL09]. By assuming a perfect phase estimation operator on the matrix of the system, we show that this problem reduces to a quantum state conversion problem which we call QLINEAREQS$_\kappa$, which itself reduces to SQSAMPLING$_\tau$. An open question is how to combine the quantum rejection sampling approach with the variable time amplitude amplification technique that was proposed by Ambainis [Amb10] to improve on the original algorithm by Harrow *et al.* [HHL09]. In order to do so, we should "open" the phase estimation black box since Ambainis's main idea is to stop some branches of the phase estimation earlier than others.

As a second application, we show that quantum rejection sampling can be used to speed up the main step in the original quantum Metropolis sampling algorithm by Temme *et al.* [TOV+11]. The general idea is to use amplitude amplification to increase the acceptance probability of a move, and therefore quadratically reduce the number of moves that have to be attempted before one is accepted. While this approach also provides some type of quadratic speed-up, it is rather different from the "quantum-quantum" Metropolis algorithm proposed by Yung and Aspuru-Guzik [YAG10]. The main difference is that the approach based on quantum rejection sampling still simulates the same classical random walk on the eigenstates of the Hamiltonian, whereas the quantum-quantum Metropolis algorithm replaces it by a quantum walk. Note that while random walks converge towards their stationary distribution from any initial state, this is not the case for quantum walks as they are reversible by definition. Therefore, while both the original quantum Metropolis sampling algorithm and our variation can start from any initial state and run at a fixed inverse temperature β to converge to the corresponding Gibbs distribution, the quantum-quantum Metropolis sampling al-

gorithm works differently: it starts from a uniform superposition, which corresponds to the Gibbs distribution at $\beta = 0$, and uses a series of measurements to project this state onto superpositions corresponding to Gibbs distributions with increasingly large β, until the desired value is reached.

Finally, as briefly discussed in Sect. 5.5.3, we can apply the quantum rejection sampling technique to solve the hidden shift problem for any Boolean function f (for more details see Sect. 6.4.2). In the limiting cases of flat or highly peaked Fourier spectra we recover the quantum algorithm for bent functions [Röt10] or Grover's algorithm for delta functions [Gro96], respectively. For a general Boolean function the hidden shift problem can be seen as lying somewhere between these two extreme cases. While our algorithm is optimal for the extreme cases of bent and delta functions, its optimality for more general cases remains an open problem.

Chapter 6

Quantum algorithms for the Boolean function hidden shift problem

Contents

6.1 Introduction

In this chapter we study the hidden shift problem for Boolean functions, and provide several quantum algorithms for this problem and some variations of it. To motivate this problem, we first review the related hidden subgroup and hidden shift problems, and highlight connections between these problems and the Boolean hidden shift problem that we consider. For more information on the hidden subgroup and hidden shift problems see the review by Childs and van Dam [CvD10].

6.1.1 Hidden subgroup problem

Many computational problems for which quantum algorithms can achieve superpolynomial speedup over the best known classical algorithms are related to the so-called *hidden subgroup problem* [NC10, KSV02, KLM07, CvD10, Lom04].

Problem (Hidden subgroup problem)**.** Given a finite group G and oracle access to a hiding function $f : G \to X$ that is hiding some subgroup H of G, find a generating set of H. The hiding function f takes values in some finite set X and is constant on cosets of H in G, and distinct on different cosets.

Two early examples of algorithms for this type of problems are Deutsch–Jozsa algorithm [DJ92] and Simon's algorithm [Sim94]. Inspired by the latter, in 1994 Shor discovered efficient quantum algorithms for factoring integers and computing discrete logarithm on a quantum computer [Sho97]. In 1995 Kitaev [Kit95, KSV02] introduced the Abelian stabilizer problem and derived an efficient quantum algorithm for this problem that includes Shor's factoring and discrete logarithm algorithms as special cases. Eventually it was observed that all of the above algorithms are solving special instances of the same problem, the hidden subgroup problem [Joz98, ME99, Joz01].

This early success created a significant interest in studying various instances of the hidden subgroup problem and led to discovery of many other interesting quantum algorithms. For example, period finding over reals was used by Hallgren to construct an efficient quantum algorithm for solving Pell's equation [Hal07, Joz03]. Moreover, the hidden subgroup problem over symmetric and dihedral groups are related to the graph isomorphism problem [BL95, Bea97, Hø97, EH99] and certain lattice problems [Reg04a], respectively, and constructing efficient algorithms for these problems are two major open questions in quantum algorithms. Kuperberg has provided a subexponential-time quantum algorithm for the dihedral subgroup problem [Kup05, Reg04b, Kup11], which has been used by Childs, Jao, and Soukharev to show how elliptic curve isogenies can be constructed in quantum subexponential time [CJS10].

6.1.2 Hidden shift problem

Hidden shift problem (also known as *hidden translation problem*) is a variant of the hidden subgroup problem and is defined as follows.

Problem (Hidden shift problem, injective version)**.** Let G be a finite group. Given oracle access to injective functions $f_0, f_1 : G \to X$ with the promise that $f_0(x) = f_1(x \cdot s)$ for some $s \in G$, determine s.

If G is Abelian, this problem is equivalent to the hidden subgroup problem in the semidirect product group $G \rtimes \mathbb{Z}_2$, where the multiplication is defined by

$$(x_1, b_1) \cdot (x_2, b_2) := \left(x_1 \cdot x_2^{(-1)^{b_1}}, b_1 + b_2\right) \tag{6.1}$$

and the hiding function $f : G \rtimes \mathbb{Z}_2 \to X$ is defined as

$$f[(x, b)] := f_b(x). \tag{6.2}$$

It is a simple calculation to show this. First, notice that the promise $f_0(x) = f_1(x \cdot s)$ is equivalent to $f_1(x) = f_0(x \cdot s^{-1})$. We can summarize both equations as

$$f_b(x) = f_{b+1}\big(x \cdot s^{(-1)^b}\big) \tag{6.3}$$

where $b \in \mathbb{Z}_2$. Using this we can check that

$$f\big[(x, b) \cdot (s, 1)\big] = f\big[(x \cdot s^{(-1)^b}, b + 1)\big] = f_{b+1}\big(x \cdot s^{(-1)^b}\big) = f_b(x) = f[(x, b)]. \tag{6.4}$$

In other words, f is constant on cosets of $H := \langle (s, 1) \rangle$. Moreover, f is distinct on different cosets since f_0 is injective. Thus, f is a hiding function for subgroup H in $G \rtimes \mathbb{Z}_2$ in the sense described in the previous section.

Notice that if $G = \mathbb{Z}_d$ then $G \rtimes \mathbb{Z}_2$ is the dihedral group. Ettinger and Høyer [EH00] have shown that the dihedral hidden subgroup problem for general subgroup H can be reduced to a subgroup of the form $\langle (s, 1) \rangle$. This means that the hidden shift problem in $\mathbb{Z}_d$ is equivalent to the dihedral hidden subgroup problem. This is one of the reasons why the hidden shift problem has been studied for various groups [EH00, vDHI03, FIM+02, MRRS07, CW07, Iva08, CvD10].

It is still open whether quantum computers can solve the dihedral subgroup problem efficiently. However, the study of hidden shift problems has resulted in quantum algorithms that are of independent interest and have cryptographic applications. For example, a quantum algorithm for the shifted Legendre symbol by van Dam, Hallgren, and Ip [vDHI03] can be used to break certain cryptosystems. In fact, negative results potentially would also have cryptographic applications for designing classical cryptosystems that are secure against quantum attacks [Reg04a].

6.1.3 Hidden shift problem for $\mathbb{Z}_d$-valued functions

For the rest of this chapter we will consider only non-injective hidden shift problems, i.e., hidden shift problems where the hiding function f is not injective (recall that in previous section injectivity was assumed as a part of the definition of the problem). Such non-injective problems have been considered, for example, in [vDHI03].

Moreover, we restrict our attention to functions of the form

$$f : \mathbb{Z}_d^n \to \mathbb{Z}_d \tag{6.5}$$

for some integers $n \geq 1$ and $d \geq 2$. For most part of this chapter we will be dealing only with the Boolean ($d = 2$) case. However, often the generalization to arbitrary d is straightforward, so we can relax this restriction. This type of problems have been studied in [Röt09, Röt10, GRR11, Gha11].

Notice that to determine the hidden shift of an injective function f, it suffices to find x_0 and x_1 such that $f_0(x_0) = f_1(x_1)$. However, this is no longer true in the non-injective case, so the problem of verifying a given shift becomes non-trivial (see Sect. 6.5.2). In fact, sometimes the hidden shift cannot be uniquely determined in principle (for more on this see Sect. 6.6.1). On the other hand, by considering functions with domain $\mathbb{Z}_d$ we have more structure than in the case of the hidden subgroup problem and the injective hidden shift problem discussed earlier, where the domain was an arbitrary set. We will exploit this structure by encoding the values of the function as phases and using Fourier transform which will be an indispensable tool for design and analysis of our algorithms.

Here is a formal definition of the problem that we study in this chapter.

Problem (Hidden shift problem for $\mathbb{Z}_d$-valued functions)**.** Given a complete description of a function $f : \mathbb{Z}_d^n \to \mathbb{Z}_d$ and access to an oracle for *shifted function* $f_s(x) := f(x + s)$, determine the *hidden shift* $s \in \mathbb{Z}_d^n$. Here $x + s$ is computed entry-wise and modulo d. When $d = 2$ we call this the *Boolean function hidden shift problem* for f and denote it by BFHSP_f.

Let us highlight the main differences between this problem and the one discussed in the previous section. In the hidden shift problem for $\mathbb{Z}_d$-valued functions:

- the function f is *not* injective, and
- we are given complete description of the unshifted function f instead of having only oracle access to f.

Moreover, we are interested only in the query complexity of the problem and do not consider its time complexity. This means that we can pre-process the description of f at no cost before we start querying the oracle.

To completely specify the problem, we need to formally define the oracles that we are allowed to use. Let $\omega := e^{2\pi i/d}$. We consider two different kinds of quantum oracles:

- the *phase* oracle: $O_{f_s} : |x\rangle \mapsto \omega^{f(x+s)}|x\rangle$,

- the *k-controlled-phase* oracle: $O_{f_{ks}} : |k\rangle|x\rangle \mapsto \omega^{f(x+ks)}|k\rangle|x\rangle$.

In classical case the oracles act in the same way, except the value of the function is computed in a $\mathbb{Z}_d$-valued register instead of a phase. Note that the second oracle is stronger, since it can be used to simulate the first oracle by setting $k = 1$. Also note that in the Boolean case ($d = 2$) the ability to use the second oracle is equivalent to having an oracle for the shifted function $f(x + s)$ and the original unshifted function $f(x)$. Hidden shift problem for oracles that allow to perform arbitrary multiples of the shift s have also been considered in [CvD07], however only for injective functions.

The motivation for studying this problem is similar to that of the hidden shift problem discussed in the previous section—we hope to shed some light on the interesting cases of the hidden subgroup problem (in particular, the dihedral subgroup problem which corresponds to the limiting case $n = 1$). It is also not excluded that our algorithms might have cryptographic applications (especially, for the case of bent functions). Finally, it is worth noting that the quantum algorithm for solving the Boolean hidden shift problem discussed in Sect. 6.4.2 was the original inspiration that led to the discovery of the quantum rejection sampling technique discussed in Chapter 5.

It would be desirable to have a complete characterization of the classical and quantum query complexity of the hidden shift problem for any Boolean function (or more generally, for any function $f : \mathbb{Z}_d^n \to \mathbb{Z}_d$). In this chapter partially address this question by providing several quantum query algorithms for this and related problems. However, it still remains an open problem to prove a lower bound on the quantum query complexity for this problem or prove any non-trivial bounds in the classical case.

For Boolean functions there are two extreme cases, for which this problem has been solved in the quantum case:

- If f is a *delta function*, i.e., $f(x) := \delta_{x,x_0}$ for some $x_0 \in \mathbb{Z}_2^n$, then the hidden shift problem for f is equivalent to Grover's search problem—finding x_0 among the 2^n elements of $\mathbb{Z}_2^n$—so the quantum query complexity in this case is $\Theta(\sqrt{2^n})$ [Gro96, BBBV97].
- If f is a *bent function*, i.e., it has a flat Fourier spectrum, then the quantum query complexity is $\Theta(1)$ [Röt10].

Intuitively, the rest of the Boolean functions lie somewhere between these two extreme cases. This is similar to the weighing matrix problem considered by van Dam [vD08], which interpolates between two extreme cases: the Bernstein-Vazirani problem [BV97] and Grover's search [Gro96].

Apart from delta and bent function, Boolean hidden shift problem has been considered also for several other families. Boolean functions that are quadratic forms or are close to being quadratic have been considered in [Röt09]. Random Boolean functions have been considered in [GRR11, Gha11].

6.1.4 Outline

In this chapter we give an overview of different approaches for solving the hidden shift problem; we generalize the known approaches and provide several new ones. Our main contributions are two quantum algorithms for solving the hidden shift problem. The first algorithm is based on pretty good measurement (see Sect. 6.4.1), but the second algorithm is based on quantum rejection sampling (see Sect. 6.4.2).

The rest of this chapter is organized as follows. First, in Sect. 6.2 we introduce some notation and basic definitions such as Fourier transform, convolution, influence, and bent functions. Next, in Sect. 6.3 we describe subroutines for preparing the t-fold Fourier states $|\Phi^t(s)\rangle$ and $|\Psi^t(s)\rangle$. They are used in Sect. 6.4, which is the most important part of this chapter, where we describe three different approaches (pretty good measurement, and "Grover-like" and "Simon-like" algorithms) for solving the hidden shift problem. In Sect. 6.5 we consider some variations of this problem, where we want to learn only one bit of information about the hidden shift s, or verify if the oracle is hiding a specified shift s (see Sect. 6.5.1 and Sect. 6.5.2, respectively). In Sect. 6.6 we analyze the structure of zero Fourier coefficients of Boolean functions. We show how to construct instances of the hidden shift problem with specific properties, as well as how to modify the quantum rejection sampling algorithm discussed earlier, so that it does not fail on these instances. Finally, Sect. 6.7 contains conclusions and a list of open questions.

6.2 Notation and basic definitions

6.2.1 Boolean Fourier transform

Fourier analysis on the Boolean cube studies the 2^n-dimensional vector space of all *real*-valued functions defined on the n-dimensional Boolean cube $\mathbb{Z}_2^n$. The main idea is that the same vector can be expressed in two complementary bases—the standard basis and

the Fourier basis—and both representations carry useful information about the corresponding function. Intuitively, the standard basis captures local information such as the value of the function at a specific point, but Fourier basis captures global information such as the periodicity of the function. For a good overview of basic properties of the Fourier transform of Boolean functions see [dW08].

First, let us define the Fourier transform of a real-valued function defined on the Boolean hypercube.

Definition 6.1. The *Fourier transform* of $F : \mathbb{Z}_2^n \to \mathbb{R}$ is a function $\hat{F} : \mathbb{Z}_2^n \to \mathbb{R}$ defined as follows:

$$\hat{F}(w) := \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_2^n} (-1)^{w \cdot x} F(x) \tag{6.6}$$

where $w \cdot x = \sum_{i=1}^n w_i x_i$ is the inner product modulo two.

Alternatively, we can encode the values of $F : \mathbb{Z}_2^n \to \mathbb{R}$ as amplitudes of a (non-normalized) quantum state:

$$|F\rangle := \sum_{x \in \mathbb{Z}_2^n} F(x)|x\rangle. \tag{6.7}$$

Then the *Fourier coefficient* of F at $w \in \mathbb{Z}_2^n$ is

$$\hat{F}(w) = \langle w|H^{\otimes n}|F\rangle \tag{6.8}$$

where $H := \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$ is the Hadamard matrix. In particular, Fourier transform is linear:

$$\widehat{F+G} = \hat{F} + \hat{G}. \tag{6.9}$$

The set $\{\hat{F}(w) : w \in \mathbb{Z}_2^n\}$ of all Fourier coefficients is called the *Fourier spectrum* of F.

To define the *Fourier transform of a Boolean function* $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$, we convert its truth table to a normalized quantum state with amplitude at $x \in \mathbb{Z}_2^n$ given by

$$F(x) := \frac{1}{\sqrt{2^n}}(-1)^{f(x)}. \tag{6.10}$$

Then the *Fourier coefficient*[1] at $w \in \mathbb{Z}_2^n$ is

$$\hat{F}(w) = \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{w \cdot x + f(x)}. \tag{6.11}$$

For $d > 2$ the definition of the Fourier transform is similar, except one has to use the quantum Fourier transform QFT_d instead of the Hadamard transform.

[1] Technically $\hat{F}$ is the Fourier transform of the normalized (± 1)-function F. However, we will also refer to $\hat{F}$ as the Fourier spectrum of the original Boolean function f.

6.2.2 Quantum Fourier transform

Throughout this chapter we fix an integer $d \geq 2$ and let $\omega := e^{2\pi i/d}$. The *quantum Fourier transform* (QFT) for qudits is

$$\mathrm{QFT} := \frac{1}{\sqrt{d}} \sum_{x,y \in \mathbb{Z}_d} \omega^{xy} |y\rangle\langle x|. \tag{6.12}$$

Note that QFT is unitary and that $\mathrm{QFT} = H$ when $d = 2$. Also note that

$$\mathrm{QFT}^{\otimes n} := \frac{1}{\sqrt{d^n}} \sum_{x,y \in \mathbb{Z}_d^n} \omega^{x \cdot y} |y\rangle\langle x|, \tag{6.13}$$

where $x \cdot y = \sum_{i=1}^n x_i y_i$ is the inner product modulo d.

In analogy to Eq. (6.6), the *Fourier transform* of function $F : \mathbb{Z}_d^n \to \mathbb{C}$ is a function $\hat{F} : \mathbb{Z}_d^n \to \mathbb{C}$ defined as

$$\hat{F}(w) := \frac{1}{\sqrt{d^n}} \sum_{x \in \mathbb{Z}_d^n} \omega^{w \cdot x} F(x). \tag{6.14}$$

Alternatively,

$$\hat{F}(w) = \langle w | \mathrm{QFT}^{\otimes n} | F \rangle \tag{6.15}$$

where

$$|F\rangle = \sum_{x \in \mathbb{Z}_d^n} F(x) |x\rangle. \tag{6.16}$$

As in the Boolean case, the canonical choice of F corresponding to $f : \mathbb{Z}_d^n \to \mathbb{Z}_d$ is

$$F(x) := \frac{\omega^{f(x)}}{\sqrt{d^n}}. \tag{6.17}$$

Proposition 6.2. *We have:* $\hat{\hat{F}}(z) = F(-z)$. *In particular, for* $d = 2$ *the Fourier transform is self-inverse.*

Proof. We have:

$$\hat{\hat{F}}(z) = \frac{1}{\sqrt{d^n}} \sum_{y \in \mathbb{Z}_d^n} \omega^{z \cdot y} \hat{F}(y) \tag{6.18}$$

$$= \frac{1}{d^n} \sum_{x,y \in \mathbb{Z}_d^n} \omega^{z \cdot y + y \cdot x} F(x) \tag{6.19}$$

$$= \sum_{x \in \mathbb{Z}_d^n} \left(\frac{1}{d^n} \sum_{y \in \mathbb{Z}_d^n} \omega^{y \cdot (z+x)} \right) F(x) \tag{6.20}$$

$$= \sum_{x \in \mathbb{Z}_d^n} \delta_{0,z+x} F(x) \tag{6.21}$$

$$= F(-z). \tag{6.22}$$

If $d = 2$ then $-z = z$ and thus $\hat{\hat{F}} = F$. For $d \geq 3$ Fourier transform has order four. □

6.2.3 Convolution

Definition 6.3. *Convolution* of functions $u, v : \mathbb{Z}_d^n \to \mathbb{C}$ is a function $(u * v) : \mathbb{Z}_d^n \to \mathbb{C}$ which is defined as

$$(u * v)(x) = \sum_{y \in \mathbb{Z}_d^n} u(y) v(x - y). \tag{6.23}$$

Note that convolution is commutative and associative, i.e.,

$$u * v = v * u \qquad \text{and} \qquad (u * v) * w = u * (v * w). \tag{6.24}$$

The following is a standard result that relates Fourier transform and convolution via entry-wise product.

Proposition 6.4. *Fourier transform and convolution satisfy the following identities:*

$$\frac{1}{\sqrt{d^n}} (\hat{u} * \hat{v}) = \widehat{uv}, \qquad \frac{1}{\sqrt{d^n}} \widehat{(u * v)} = \hat{u}\hat{v}, \tag{6.25}$$

where uv is the entry-wise product of functions u and v.

Proof. For any $x \in \mathbb{Z}_d^n$ we have:

$$\frac{1}{\sqrt{d^n}}(\hat{u} * \hat{v})(x) = \frac{1}{\sqrt{d^n}} \sum_{y \in \mathbb{Z}_d^n} \hat{u}(y)\hat{v}(x-y) \tag{6.26}$$

$$= \frac{1}{\sqrt{d^n}} \sum_{y \in \mathbb{Z}_d^n} \left(\frac{1}{\sqrt{d^n}} \sum_{z \in \mathbb{Z}_d^n} \omega^{y \cdot z} u(z) \right) \left(\frac{1}{\sqrt{d^n}} \sum_{w \in \mathbb{Z}_d^n} \omega^{(x-y) \cdot w} v(w) \right) \tag{6.27}$$

$$= \frac{1}{\sqrt{d^n}} \sum_{z,w \in \mathbb{Z}_d^n} \left(\frac{1}{d^n} \sum_{y \in \mathbb{Z}_d^n} \omega^{y \cdot (z-w)} \right) \omega^{x \cdot w} u(z) v(w) \tag{6.28}$$

$$= \frac{1}{\sqrt{d^n}} \sum_{z \in \mathbb{Z}_d^n} \omega^{x \cdot z} u(z) v(z) \tag{6.29}$$

$$= (\widehat{uv})(x). \tag{6.30}$$

The second identity follows from the first one by replacing u and v by $\hat{u}$ and $\hat{v}$, respectively, and taking the Fourier transform of both sides. □

Proposition 6.5. *The t-fold convolution satisfies the following identity:*

$$\left[\frac{\hat{u}}{\sqrt{d^n}} \right]^{*t} = \frac{\widehat{u^t}}{\sqrt{d^n}}. \tag{6.31}$$

Proof. We prove this by induction. The base case $t = 1$ is trivial as $f^{*1} = f$ by definition. We assume that the desired relation holds for $t - 1$. Then

$$\left[\frac{\hat{u}}{\sqrt{d^n}} \right]^{*t} = \left[\frac{\hat{u}}{\sqrt{d^n}} \right] * \left[\frac{\hat{u}}{\sqrt{d^n}} \right]^{*(t-1)} \tag{6.32}$$

$$= \left[\frac{\hat{u}}{\sqrt{d^n}} \right] * \left[\frac{\widehat{u^{t-1}}}{\sqrt{d^n}} \right] \tag{6.33}$$

$$= \frac{\widehat{u^t}}{\sqrt{d^n}} \tag{6.34}$$

according to the first relation in Eq. (6.25). □

6.2.4 Influence

Let us consider the case $d = 2$, i.e., when f is an n-argument Boolean function.

Definition 6.6. The *influence of input* $w \in \mathbb{Z}_2^n$ on f is

$$\mathrm{I}_f(w) := \Pr_x\big[f(x) \neq f(x+w)\big] \tag{6.35}$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} \big[f(x) \oplus f(x+w)\big], \tag{6.36}$$

where "$\oplus$" denotes addition modulo two, but the terms in the sum indexed by x are added as integers. The *influence of function* f is the minimum over the non-zero input influences:

$$\mathrm{I}_f := \min_{w \in \mathbb{Z}_2^n \setminus \{0\}} \mathrm{I}_f(w). \tag{6.37}$$

Note that $F(x) = (-1)^{f(x)}/\sqrt{2^n}$ satisfies the following identity:

$$(F * F)(w) = \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{f(x)+f(x+w)} \tag{6.38}$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} \Big(1 - 2\big[f(x) \oplus f(x+w)\big]\Big) \tag{6.39}$$

$$= 1 - 2\mathrm{I}_f(w). \tag{6.40}$$

6.2.5 Bent functions

Quantum algorithms for finding a hidden shift presented in Sect. 6.4 are particularly efficient for functions having a relatively flat Fourier spectrum, and therefore in particular for bent functions [Dil72, Dil75, MS77, Dob95].

Definition 6.7. We say that a Boolean function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ is *bent* if it has a flat Fourier spectrum, i.e., all Fourier coefficients $\hat{F}(w)$ have the same absolute value.

From the normalization constraint $\sum_{w \in \mathbb{Z}_2^n} |\hat{F}(w)|^2 = 1$ of the Fourier spectrum we get that a Boolean function is bent if

$$\forall w \in \mathbb{Z}_2^n : |\hat{F}(w)| = \frac{1}{\sqrt{2^n}}. \tag{6.41}$$

Recall from Eq. (6.11) that $\hat{F}(w)$ is always an integer multiple of $1/2^n$. Thus, from the above equation it immediately follows that necessary condition for a bent function in n

variables to exist is that n is even [Dil75, MS77]. Moreover, from $|\hat{F}(0)| = 1/\sqrt{2^n}$ we get that $\left|\sum_{w\in\mathbb{Z}_2^n}(-1)^{f(x)}\right| = \sqrt{2^n}$, so a bent function f is close to being balanced:

$$\left|\{x \in \mathbb{Z}_2^n : f(x) = 1\}\right| = \frac{2^n \pm \sqrt{2^n}}{2}. \tag{6.42}$$

A simple example of a bent function is provided by the *inner product function* [Dil72]. The most simple instance is the two-argument function $f(x, y) := xy$ where $x, y \in \mathbb{Z}_2$ which is equivalent to the logical AND. It is easy to verify that f is a bent function. This can be generalized to $2n$ variables [MS77] and we obtain the inner product modulo two:

$$\mathrm{IP}_n(x_1, \ldots, x_n, y_1, \ldots, y_n) := \sum_{i=1}^{n} x_i y_i. \tag{6.43}$$

It is straightforward to check that IP_n is bent.

While many other examples of bent functions have been constructed, see for instance [MS77, Dil75, Dob95], no complete classification is known. Besides the flatness of the Fourier spectrum, many equivalent characterizations of bent functions are known [CS09, Dil72].

6.3 Quantum algorithms for preparing the t-fold Fourier states

In this section we describe two similar quantum algorithms for preparing the following states for any integer $t \geq 1$:

$$|\Phi^t(s)\rangle = \sum_{w\in\mathbb{Z}_d^n} \omega^{s\cdot w} |\mathcal{F}_w^t\rangle |w\rangle, \tag{6.44}$$

$$|\Psi^t(s)\rangle = \sum_{w\in\mathbb{Z}_d^n} |s \cdot w\rangle |\mathcal{F}_w^t\rangle |w\rangle. \tag{6.45}$$

Here s is the hidden shift, but the non-normalized vectors $|\mathcal{F}_w^t\rangle \in \mathbb{C}^{d(t-1)n}$ will be defined in the next section. We will use these algorithms in Sect. 6.4 as subroutines to solve the hidden shift problem.

Informally we refer to the states in Eqs. (6.44) and (6.45) as the *t-fold Fourier states*, since the norm of the vector $|\mathcal{F}_w^t\rangle$ can be interpreted as a t-fold generalization of the

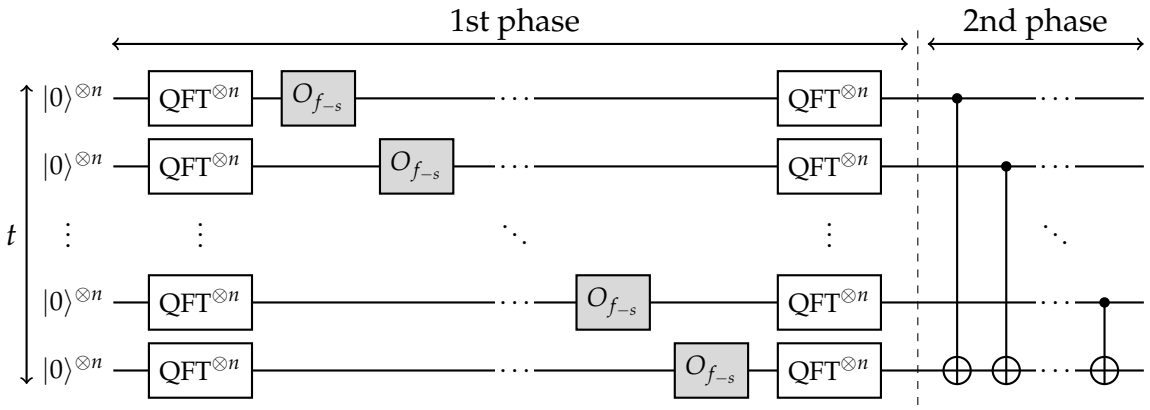

Figure 6.1: Quantum algorithm for preparing the t-fold Fourier sate $|\Phi^t(s)\rangle$ in Eq. (6.44). Definitions of the oracle and "controlled-plus" gates are given in Eqs. (6.46) and (6.47), respectively. The state on the first wire at the end of the first phase is given in Eq. (6.53).

Fourier coefficient at w as discussed later. Here are a couple of differences between the states $|\Phi^t(s)\rangle$ and $|\Psi^t(s)\rangle$ that one should note:

- $|\Phi^t(s)\rangle \in \mathbb{C}^{dtn}$ but $|\Psi^t(s)\rangle \in \mathbb{C}^{d(tn+1)}$.
- Φ-states cannot be used to detect hidden b-shifts (see Sect. 6.6). If s is a b-shift then the Φ-states corresponding to s and 0 differ only by a global phase: $|\Phi^t(s)\rangle = \omega^b|\Phi^t(0)\rangle$.
- We can probabilistically convert a Ψ-state to the corresponding Φ-state by applying QFT on the first register and measuring it in the standard basis. With probability $1/d$ we get outcome 1 and the post-measurement state is $|\Phi^t(s)\rangle$. If the outcome is not 1, we can sometimes still recover the correct value of s at the end of the algorithm. In particular, let φ be Euler's totient function. Then with probability $\varphi(d)/d$ we get outcome $k \in \mathbb{Z}_d$ that has a multiplicative inverse modulo d and the corresponding post-measurement state is $|\Phi^t(ks)\rangle$. Since we know the value of k, we can recover the original hidden shift s at the end of the algorithm if we multiply the obtained value by k^{-1}.

6.3.1 Computing $w \cdot s$ in the *phase* to prepare $|\Phi^t(s)\rangle$

6.3.1.1 Algorithm

Consider the phase oracle

$$O_{f_{-s}} : |x\rangle \mapsto \omega^{f(x-s)}|x\rangle \tag{6.46}$$

that acts on n qudits. We will execute t copies of this oracle in parallel to prepare the state $|\Phi^t(s)\rangle$ given in Eq. (6.44). The circuit of our algorithm is given in Fig. 6.1. It consists of two phases: the first phase prepares t identical copies of the same state by using one oracle call between two quantum Fourier transforms on each wire independently. The second phase entangles these states by applying a sequence of "controlled-plus" gates that are defined as follows:

$$|x\rangle|y\rangle \mapsto |x\rangle|y + x\rangle, \tag{6.47}$$

where addition is modulo d.

6.3.1.2 Analysis

Let us analyze how the initial state evolves. During the first phase of the algorithm the first wire evolves as follows:

$$|0\rangle^{\otimes n} \mapsto \frac{1}{\sqrt{d^n}} \sum_{x \in \mathbb{Z}_d^n} |x\rangle \tag{6.48}$$

$$\mapsto \frac{1}{\sqrt{d^n}} \sum_{x \in \mathbb{Z}_d^n} \omega^{f(x-s)}|x\rangle \tag{6.49}$$

$$\mapsto \frac{1}{d^n} \sum_{x,y \in \mathbb{Z}_d^n} \omega^{f(x-s)+x\cdot y}|y\rangle \tag{6.50}$$

$$= \frac{1}{d^n} \sum_{x,y \in \mathbb{Z}_d^n} \omega^{f(x)+(x+s)\cdot y}|y\rangle \tag{6.51}$$

$$= \sum_{y \in \mathbb{Z}_d^n} \omega^{s\cdot y} \left(\frac{1}{d^n} \sum_{x \in \mathbb{Z}_d^n} \omega^{f(x)+x\cdot y} \right) |y\rangle \tag{6.52}$$

$$= \sum_{y \in \mathbb{Z}_d^n} \omega^{s\cdot y} \hat{F}(y)|y\rangle. \tag{6.53}$$

Since the algorithm in Fig. 6.1 prepares t copies of the same state in parallel, the state at the end of the first phase is given by the t-fold tensor product of the state in Eq. (6.53):

$$\sum_{y_1,\dots,y_t \in \mathbb{Z}_d^n} \omega^{s\cdot(y_1+\cdots+y_t)} \bigotimes_{i=1}^{t} \hat{F}(y_i)|y_i\rangle. \tag{6.54}$$

During the second phase of the algorithm, the "controlled-plus" gates transform this state into

$$\sum_{y_1,\dots,y_t \in \mathbb{Z}_d^n} \omega^{s\cdot(y_1+\cdots+y_t)} \left[\bigotimes_{i=1}^{t-1} \hat{F}(y_i)|y_i\rangle\right] \hat{F}(y_t)|y_1+\cdots+y_t\rangle \tag{6.55}$$

$$= \sum_{y_1,\dots,y_t \in \mathbb{Z}_d^n} \omega^{s\cdot y_t} \left[\bigotimes_{i=1}^{t-1} \hat{F}(y_i)|y_i\rangle\right] \hat{F}\big(y_t - (y_1+\cdots+y_{t-1})\big)|y_t\rangle. \tag{6.56}$$

We can rewrite this state as

$$|\Phi^t(s)\rangle := \sum_{w\in\mathbb{Z}_d^n} \omega^{s\cdot w}|\mathcal{F}_w^t\rangle|w\rangle, \tag{6.57}$$

where the non-normalized state $|\mathcal{F}_w^t\rangle$ on $(t-1)n$ qudits is given by

$$|\mathcal{F}_w^t\rangle := \sum_{y_1,\dots,y_{t-1}\in\mathbb{Z}_d^n} \hat{F}(y_1)\cdots\hat{F}(y_{t-1})\hat{F}\big(w-(y_1+\cdots+y_{t-1})\big)|y_1\rangle\cdots|y_{t-1}\rangle. \tag{6.58}$$

The square of the norm of $|\mathcal{F}_w^t\rangle$ is

$$\||\mathcal{F}_w^t\rangle\|_2^2 = \sum_{y_1,\dots,y_{t-1}\in\mathbb{Z}_d^n} \hat{F}(y_1)^2\cdots\hat{F}(y_{t-1})^2\hat{F}\big(w-(y_1+\cdots+y_{t-1})\big)^2. \tag{6.59}$$

Note that this is the t-fold convolution of $\hat{F}^2$ with itself, evaluated at w:

$$\||\mathcal{F}_w^t\rangle\|_2^2 = \big[\hat{F}^2\big]^{*t}(w). \tag{6.60}$$

In particular, for $t=1$ we have:

$$\||\mathcal{F}_w^1\rangle\|_2 = |\hat{F}(w)|, \tag{6.61}$$

so $\||\mathcal{F}_w^t\rangle\|_2$ can be interpreted as a t-fold generalization of the Fourier spectrum.

6.3.1.3 The t-fold Fourier spectrum

The quantity $\||\mathcal{F}_w^t\rangle\|_2$ will come up quite often in our discussion and play the role of a generalized Fourier coefficient. Let us introduce special notation for it and provide several alternative expressions.

Definition 6.8. The *t-fold Fourier coefficient* of $f : \mathbb{Z}_d^n \to \mathbb{Z}_d$ at $w \in \mathbb{Z}_d^n$ is

$$\mathcal{F}^t(w) := \||\mathcal{F}_w^t\rangle\|_2. \tag{6.62}$$

We can express this quantity in many equivalent ways. For example, according to Prop. 6.4 and Prop. 6.5 we have:

$$\begin{aligned}\left[\mathcal{F}^t(w)\right]^2 &= \left[\hat{F}^2\right]^{*t}(w) && (6.63)\\ &= \left[\frac{1}{\sqrt{d^n}}\widehat{(F * F)}\right]^{*t}(w) && (6.64)\\ &= \frac{1}{\sqrt{d^n}}\widehat{(F * F)^t}\,(w). && (6.65)\end{aligned}$$

When $d = 2$ we can use Eq. (6.40) to write the last expression in terms of the influence:

$$\left[\mathcal{F}^t(w)\right]^2 = \frac{1}{\sqrt{2^n}}\widehat{(1 - 2\mathrm{I}_f)^t}\,(w). \tag{6.66}$$

6.3.2 Computing $w \cdot s$ in the *register* to prepare $|\Psi^t(s)\rangle$

6.3.2.1 Algorithm

This time we will use a more powerful oracle. It is a controlled version of the phase oracle, but we can also control how many times the hidden shift s is applied:

$$O_{f_{ks}} : |k\rangle|x\rangle \mapsto \omega^{f(x+ks)}|k\rangle|x\rangle. \tag{6.67}$$

It acts on $n + 1$ qudits and the value $k \in \mathbb{Z}_d$ in the first register controls the number of times the shift s is applied, but $x \in \mathbb{Z}_d^n$ is the value that we are querying. The workspace of our algorithm consists of $tn + 1$ qudits and its circuit is given in Fig. 6.2.

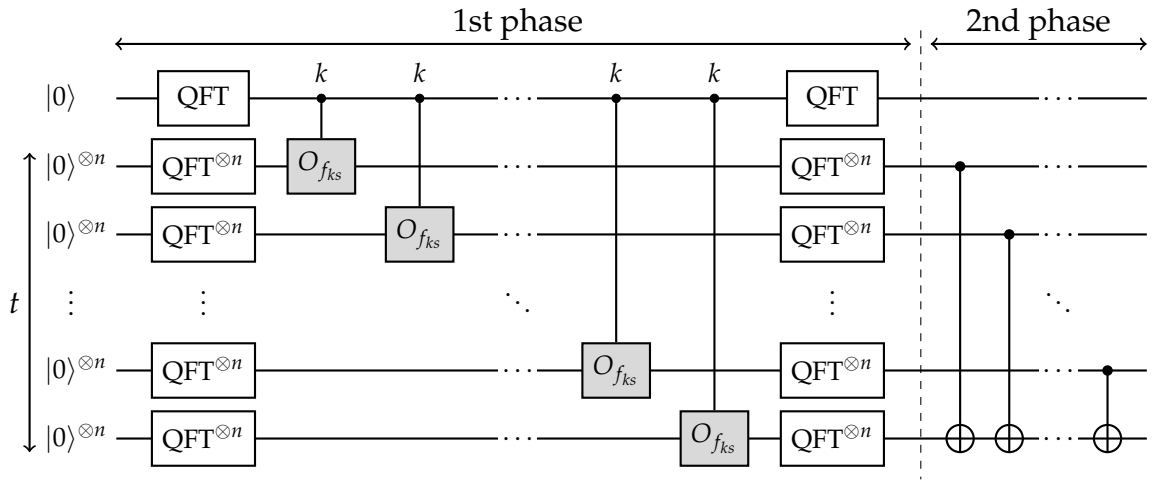

Figure 6.2: Quantum algorithm for preparing the t-fold Fourier sate $|\Psi^t(s)\rangle$ in Eq. (6.45). Definitions of the oracle and "controlled-plus" gates are given in Eqs. (6.67) and (6.47), respectively. The state on the first two wires at the end of the first phase is given in Eq. (6.73).

6.3.2.2 Analysis

Analysis of this algorithm is slightly more complicated, but otherwise very similar to the previous algorithm. The state on the first two wires during the first phase evolves

as follows:

$$|0\rangle|0\rangle^{\otimes n} \mapsto \left(\frac{1}{\sqrt{d}}\sum_{k\in\mathbb{Z}_d}|k\rangle\right)\otimes\left(\frac{1}{\sqrt{d^n}}\sum_{x\in\mathbb{Z}_d^n}|x\rangle\right) \tag{6.68}$$

$$\mapsto \frac{1}{\sqrt{d}}\sum_{k\in\mathbb{Z}_d}|k\rangle\otimes\frac{1}{\sqrt{d^n}}\sum_{x\in\mathbb{Z}_d^n}\omega^{f(x+ks)}|x\rangle \tag{6.69}$$

$$\mapsto \frac{1}{d}\sum_{k,l\in\mathbb{Z}_d}\omega^{kl}|l\rangle\otimes\frac{1}{d^n}\sum_{x,y\in\mathbb{Z}_d^n}\omega^{f(x+ks)+x\cdot y}|y\rangle \tag{6.70}$$

$$= \frac{1}{d}\sum_{k,l\in\mathbb{Z}_d}\omega^{kl}|l\rangle\otimes\frac{1}{d^n}\sum_{x,y\in\mathbb{Z}_d^n}\omega^{f(x)+(x-ks)\cdot y}|y\rangle \tag{6.71}$$

$$= \sum_{y\in\mathbb{Z}_d^n}\sum_{l\in\mathbb{Z}_d}\left(\frac{1}{d}\sum_{k\in\mathbb{Z}_d}\omega^{k(l-s\cdot y)}\right)|l\rangle\otimes\left(\frac{1}{d^n}\sum_{x\in\mathbb{Z}_d^n}\omega^{f(x)+x\cdot y}\right)|y\rangle \tag{6.72}$$

$$= \sum_{y\in\mathbb{Z}_d^n}|s\cdot y\rangle\otimes\hat{F}(y)|y\rangle. \tag{6.73}$$

When we execute t queries in parallel, the first term in Eq. (6.72) picks up a phase of $\omega^{-k(s\cdot y_i)}$ from the ith query. Thus the state on all $t+1$ wires at the end of the first phase is given by the following generalization of the state in Eq. (6.73):

$$\sum_{y_1,\ldots,y_t\in\mathbb{Z}_d^n}|s\cdot(y_1+\cdots+y_t)\rangle\bigotimes_{i=1}^{t}\hat{F}(y_i)|y_i\rangle. \tag{6.74}$$

Repeating essentially the same calculations as before, we see that final state at the end of the second phase is

$$|\Psi^t(s)\rangle := \sum_{w\in\mathbb{Z}_d^n}|s\cdot w\rangle|\mathcal{F}_w^t\rangle|w\rangle, \tag{6.75}$$

where $|\mathcal{F}_w^t\rangle$ was defined in Eq. (6.58).

6.4 Quantum algorithms for finding a hidden shift

In this section we consider how the two basic building blocks, circuits for preparing states $|\Phi^t(s)\rangle$ and $|\Psi^t(s)\rangle$, introduced in Sect. 6.3 can be used to construct quantum algorithms for solving the hidden shift problem. In particular, we discuss three different approaches which we loosely call:

1. the pretty good measurement approach,
2. the "Grover-like" approach,
3. the "Simon-like" approach.

The first approach is based on the Pretty Good Measurement (PGM) which was introduced by Hausladen and Wootters [HW94], and has also been used in the context of the hidden subgroup problem [BCvD05, BCvD06, MR05]. The second approach uses amplitude amplification and is based on quantum rejection sampling. In the special case of delta functions this method reduces to Grover's algorithm [Gro96], so we call it the "Grover-like" approach. Finally, the third approach resembles Simon's algorithm [Sim94], as it first collects data using a simple quantum sampling procedure, and the performs classical post-processing by solving a system of linear equations.

As an alternative to the "Simon-like" approach, Gharibi has recently proposed an "injectivization" technique [Gha11]. It reduces the Boolean hidden shift problem to Simon's algorithm [Sim94] by making the hiding function injective.

6.4.1 The PGM (Pretty Good Measurement) approach

In this approach the main idea is to prepare the state $|\Phi^t(s)\rangle$ for some value of t and then extract the hidden shift s from it via PGM. The same measurement has been used by Bacon, Childs, and van Dam [BCvD05] to efficiently solve the hidden subgroup problem for semidirect product groups, including the Heisenberg group, and is also known to be optimal for the dihedral subgroup problem [BCvD06].

Let us consider a set of mixed states $\{\rho_s^{(t)} : s \in \mathbb{Z}_d^n\}$ where $\rho_s^{(t)}$ is given with probability p_s. The *pretty good measurement* [HW94] for discriminating these states is a POVM with operators $\{E_s : s \in \mathbb{Z}_d^n\} \cup \{E_*\}$ where

$$E_s := E^{-1/2}\, p_s \rho_s^{(t)}\, E^{-1/2}, \qquad\qquad E := \sum_{s \in \mathbb{Z}_d^n} p_s \rho_s^{(t)}, \tag{6.76}$$

and $E_* := I - \sum_{s \in \mathbb{Z}_d^n} E_s$.

In our case the states $\rho_s^{(t)}$ and probabilities p_s are given by

$$\rho_s^{(t)} := |\Phi^t(s)\rangle\langle\Phi^t(s)|, \qquad\qquad p_s := \frac{1}{d^n}, \tag{6.77}$$

where $|\Phi^t(s)\rangle$ was defined in Eq. (6.57).

To find the operators E_s, let us first compute an expression for E:

$$E = \sum_{s \in \mathbb{Z}_d^n} \frac{1}{d^n} \rho_s^{(t)} \tag{6.78}$$

$$= \sum_{s \in \mathbb{Z}_d^n} \frac{1}{d^n} \sum_{w,w' \in \mathbb{Z}_d^n} \omega^{(w-w')\cdot s} |\mathcal{F}_w^t\rangle\langle\mathcal{F}_{w'}^t| \otimes |w\rangle\langle w'| \tag{6.79}$$

$$= \sum_{w \in \mathbb{Z}_d^n} |\mathcal{F}_w^t\rangle\langle\mathcal{F}_w^t| \otimes |w\rangle\langle w| \tag{6.80}$$

$$= \sum_{w \in \mathbb{Z}_d^n} \||\mathcal{F}_w^t\rangle\|_2^2 \cdot \frac{|\mathcal{F}_w^t\rangle\langle\mathcal{F}_w^t|}{\||\mathcal{F}_w^t\rangle\|_2^2} \otimes |w\rangle\langle w|. \tag{6.81}$$

Note that E is a sum of mutually orthogonal rank-1 operators whose eigenvalues are $\||\mathcal{F}_w^t\rangle\|_2^2$. Thus

$$E^{-1/2} = \sum_{w \in \mathbb{Z}_d^n} \frac{1}{\||\mathcal{F}_w^t\rangle\|_2} \cdot \frac{|\mathcal{F}_w^t\rangle\langle\mathcal{F}_w^t|}{\||\mathcal{F}_w^t\rangle\|_2^2} \otimes |w\rangle\langle w|. \tag{6.82}$$

The POVM element E_s is the outer product of vector $|E_s\rangle$ with itself where

$$|E_s\rangle := E^{-1/2}\sqrt{p_s}|\Phi^t(s)\rangle \tag{6.83}$$

$$= \left(\sum_{w \in \mathbb{Z}_d^n} \frac{|\mathcal{F}_w^t\rangle\langle\mathcal{F}_w^t|}{\||\mathcal{F}_w^t\rangle\|_2^3} \otimes |w\rangle\langle w| \right) \frac{1}{\sqrt{d^n}} \left(\sum_{w \in \mathbb{Z}_d^n} \omega^{w\cdot s} |\mathcal{F}_w^t\rangle |w\rangle \right) \tag{6.84}$$

$$= \frac{1}{\sqrt{d^n}} \sum_{w \in \mathbb{Z}_d^n} \omega^{w\cdot s} \frac{|\mathcal{F}_w^t\rangle}{\||\mathcal{F}_w^t\rangle\|_2} \otimes |w\rangle. \tag{6.85}$$

Notice that vectors $|E_s\rangle$ are orthonormal, so the PGM is just an orthogonal measurement on these vectors (and the orthogonal complement of their span). Thus, our algorithm can be summarized as follows:

PGM(f, t)

1. Prepare $|\Phi^t(s)\rangle$ using the circuit shown in Fig. 6.1.
2. Recover s by performing an orthogonal measurement with projectors $\{|E_s\rangle\langle E_s| : s \in \mathbb{Z}_d^n\} \cup \{E_*\}$.

The success probability to recover the hidden shift correctly using PGM from the t-fold Fourier state $|\Phi^t(s)\rangle$ defined in Eq. (6.57) is given by

$$p_{\text{success}} = \left|\langle E_s|\Phi^t(s)\rangle\right|^2 = \left(\frac{1}{\sqrt{d^n}}\sum_{w\in\mathbb{Z}_d^n}\mathcal{F}^t(w)\right)^2, \tag{6.86}$$

where $\mathcal{F}^t(w) = \||\mathcal{F}_w^t\rangle\|_2$ is the t-fold Fourier coefficient introduces in Sect. 6.3.1.3. We can use Eq. (6.65) to get

$$p_{\text{success}} = \frac{1}{d^n}\left(\sum_{w\in\mathbb{Z}_d^n}\sqrt{\frac{1}{\sqrt{d^n}}\widehat{(F*F)^t}(w)}\right)^2. \tag{6.87}$$

Note that for $t = 1$ we can use Eq. (6.61) to simplify Eq. (6.86) as follows:

$$p_{\text{success}} = \frac{1}{d^n}\left(\sum_{w\in\mathbb{Z}_d^n}|\hat{F}(w)|\right)^2. \tag{6.88}$$

6.4.1.1 Boolean delta function

Let us check how well the PGM approach performs for the case of a Boolean delta function. We know that the quantum query complexity in this case should be $\Theta(\sqrt{2^n})$. One can work out from Eq. (6.87) that the success probability is

$$p_{\text{success}} = \frac{1}{2^{2n}}\left((2^n-1)\sqrt{1-\left(\frac{2^n-4}{2^n}\right)^t} + \sqrt{1+(2^n-1)\left(\frac{2^n-4}{2^n}\right)^t}\right)^2. \tag{6.89}$$

Since t is the total number of queries made by the algorithm, let us choose $t = \sqrt{2^n}$. Unfortunately, then the success probability goes to 0 as $n \to \infty$. In fact, the same happens even if $t = c^n$ for any $c < 2$. Only if we take $t = 2^n$, the success probability approaches a positive constant $1 - 1/e^4 \approx 0.98$ as $n \to \infty$. This means that the PGM algorithm does not give us the expected quadratic improvement of query complexity in the limiting case of delta functions. Thus, the PGM algorithm is not optimal in general. However, it performs extremely well in other cases that we discuss next.

6.4.1.2 Bent functions

Let f be a Bent function (in particular, $d = 2$). Recall from Definition 6.7 that its Fourier spectrum is flat: $|\hat{F}(w)| = 1/\sqrt{2^n}$ for all $w \in \mathbb{Z}_2^n$. Thus, from Eq. (6.88) we get that $p_{\text{success}} = 1$ when $t = 1$. Thus, we can find the hidden shift with certainty by measuring $|\Phi^1(s)\rangle$ with the pretty good measurement (recall that preparing $|\Phi^1(s)\rangle$ requires only one query to O_{f_s}). This was first observed by Rötteler in [Röt10].

Theorem 6.9 ([Röt10])**.** *If f is a bent function then a quantum algorithm can solve the Boolean hidden shift problem for f with success probability one using a single query to O_{f_s}. In particular, this can be achieved using the **PGM**$(f, 1)$ algorithm.*

This result seems surprising, since we extract an n-bit string from the oracle by using only a single query. On the other hand, we were given a very strong promise on the oracle—namely, that its underlying function is bent. A similar phenomenon can be observed also in the case of the Bernstein-Vazirani problem [BV97].

6.4.1.3 Random Boolean functions

Recall from Eq. (6.88) that **PGM**$(f, 1)$ solves the hidden shift problem for Boolean function f with success probability

$$p(f) = \frac{1}{2^n} \left(\sum_{w \in \mathbb{Z}_2^n} |\hat{F}(w)| \right)^2 . \tag{6.90}$$

Let us show that if f is picked uniformly at random, then the expected success probability is lower bounded by some constant that does not depend on n, where the expectation is over the choice of f.

Theorem 6.10. *Let $p(f)$ denote the success probability of **PGM**$(f, 1)$ for solving* BFHSP_f *for an n-argument Boolean function f. When f is picked uniformly at random, the expected success probability $\bar{p} := \mathbb{E}_f \big[p(f) \big]$ is lower bounded by a constant: $\bar{p} \geq 1/2$.*

Proof. We have:

$$\bar{p} = \frac{1}{2^{2^n}} \sum_f p(f) = \frac{1}{2^n} \frac{1}{2^{2^n}} \sum_f \left(\sum_{w \in \mathbb{Z}_2^n} |\hat{F}(w)| \right)^2 . \tag{6.91}$$

Since $\sum_{i=1}^{N}|v_i|^2 \geq \frac{1}{N}\left(\sum_{i=1}^{N}|v_i|\right)^2$ for any $v \in \mathbb{C}^N$, we get

$$\bar{p} \geq \frac{1}{2^n}\left(\frac{1}{2^{2^n}}\right)^2\left(\sum_f \sum_{w\in\mathbb{Z}_2^n}|\hat{F}(w)|\right)^2 \tag{6.92}$$

$$= \frac{1}{2^n}\left(\frac{1}{2^{2^n}}\sum_{w\in\mathbb{Z}_2^n}\sum_f\left|\frac{1}{2^n}\sum_{x\in\mathbb{Z}_2^n}(-1)^{w\cdot x+f(x)}\right|\right)^2, \tag{6.93}$$

where we exchanged the two sums and substituted $\hat{F}(w)$ from Eq. (6.11). For each w we can changes the order of summation over f by defining $f'(x) := w \cdot x + f(x)$ and summing over f' instead. Thus the value of the inner sum does not depend on w and we get

$$\bar{p} \geq \frac{1}{2^n}\left(\frac{1}{2^{2^n}}\sum_f\left|\sum_{x\in\mathbb{Z}_2^n}(-1)^{f(x)}\right|\right)^2. \tag{6.94}$$

This can be written simply as

$$\bar{p} \geq \frac{L(2^n)^2}{2^n} \tag{6.95}$$

where

$$L(N) := \frac{1}{2^N}\sum_{z\in\{1,-1\}^N}\left|\sum_{i=1}^{N} z_i\right| \tag{6.96}$$

is the expected distance traveled by N steps of a random walk on a line (each step is of size one and is chosen with equal probability to left or right). It remains to lower bound $L(N)$.

Let $N = 2m$ for some integer $m \geq 1$. Using standard identities for sums of binomial coefficients, we compute that

$$L(2m) = \frac{1}{2^{2m}}\cdot 2\sum_{k=0}^{m}(2m-2k)\binom{2m}{k} \tag{6.97}$$

$$= \frac{1}{2^{2m}}\cdot 2m\binom{2m}{m}. \tag{6.98}$$

Since the central binomial coefficient satisfies [Kos08, p. 48]

$$\binom{2m}{m} \geq \frac{4^m}{\sqrt{4m}}, \tag{6.99}$$

we get that

$$L(2m) \geq \sqrt{m}. \tag{6.100}$$

For $N = 2^n$ this gives $L(2^n) \geq \sqrt{2^n/2}$. We plug this in Eq. (6.95) and get $\bar{p} \geq 1/2$. □

In fact, according to Stirling's formula $\binom{2m}{m} \sim 4^m/\sqrt{\pi m}$ as $m \to \infty$. This means that $L(N) \sim \sqrt{2N/\pi}$ as $N \to \infty$ and our lower bound on $\bar{p}$ approaches $2/\pi \approx 0.63$ as $n \to \infty$. We believe that taking $t = 2$ should suffice to get a lower bound on $\bar{p}$ that approaches 1 as $n \to \infty$.

Conjecture 1. For every $\varepsilon > 0$ there exists an integer n_0 such that if $n \geq n_0$ then **PGM**$(f, 2)$ solves BFHSP$_f$ with expected success probability $\bar{p} \geq 1 - \varepsilon$, where f is an n-argument boolean function chosen uniformly at random.

6.4.1.4 Perturbed functions

In this section we show that the success probability of the **PGM**$(f, 1)$ algorithm does not change much if the truth table of the Boolean function is perturbed only at few locations. Together with Theorem 6.9 this implies that we can solve the BFHSP$_f$ problem with high success probability and a single query for any function that is sufficiently close to a bent function.

Let us first show that if the truth table of a Boolean function f is perturbed at one location, its Fourier spectrum does not change by too much. Let

$$\delta_a(x) = \begin{cases} 1 & \text{if } x = a, \\ 0 & \text{otherwise} \end{cases} \tag{6.101}$$

be the Kronecker delta function.

Proposition 6.11. *Let f and g be n-argument Boolean functions such that $g(x) := f(x) + \delta_a(x)$ for some $a \in \mathbb{Z}_2^n$. Then*

$$\hat{G}(w) = \hat{F}(w) - \frac{2}{\sqrt{2^n}}(-1)^{w \cdot a} F(a). \tag{6.102}$$

Proof. First, observe that the normalized (± 1)-valued functions corresponding to f and g are related as follows:

$$G(x) = \frac{1}{\sqrt{2^n}}(-1)^{f(x)+\delta_a(x)} \tag{6.103}$$

$$= \frac{1}{\sqrt{2^n}}(-1)^{f(x)}\big(1-2\delta_a(x)\big) \tag{6.104}$$

$$= F(x) - 2F(a)\delta_a(x). \tag{6.105}$$

By linearity of the Fourier transform,

$$\hat{G}(w) = \hat{F}(w) - 2F(a)\hat{\delta}_a(w) \tag{6.106}$$

where

$$\hat{\delta}_a(w) = \frac{1}{\sqrt{2^n}} \sum_{x\in\mathbb{Z}_2^n} (-1)^{w\cdot x}\delta_a(x) = \frac{1}{\sqrt{2^n}}(-1)^{w\cdot a}. \tag{6.107}$$

We substitute this in Eq. (6.106) and get the desired result. □

Let us consider how the success probability changes under one perturbation.

Lemma 6.12. *Let $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ be an n-argument Boolean function and $g(x) := f(x) + \delta_a(x)$ be a perturbed function where $\delta_a(x)$ is the Kronecker delta defined in Eq. (6.101). If* ***PGM****$(f,1)$ solves* BFHSP_f *with success probability p_f, then* ***PGM****$(g,1)$ solves* BFHSP_g *with success probability p_g such that*

$$\sqrt{p_g} \geq \sqrt{p_f} - \frac{2}{\sqrt{2^n}}. \tag{6.108}$$

Proof. From Eq. (6.88) and Prop. 6.11 we have:

$$\sqrt{p_g} = \frac{1}{\sqrt{2^n}} \sum_{w\in\mathbb{Z}_2^n} |\hat{G}(w)| \tag{6.109}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{w\in\mathbb{Z}_2^n} \left|\hat{F}(w) - \frac{2}{\sqrt{2^n}}(-1)^{w\cdot a}F(a)\right| \tag{6.110}$$

$$\geq \frac{1}{\sqrt{2^n}} \sum_{w\in\mathbb{Z}_2^n} \left(|\hat{F}(w)| - \frac{2}{\sqrt{2^n}}|F(a)|\right) \tag{6.111}$$

$$= \sqrt{p_f} - 2|F(a)| \tag{6.112}$$

$$= \sqrt{p_f} - \frac{2}{\sqrt{2^n}} \tag{6.113}$$

as claimed. □

As a corollary of the above lemma we get the following result that bounds the success probability after k perturbations.

Corollary 6.13. *If $g(x) := f(x) + \sum_{i=1}^{k} \delta_{a_i}(x)$ for some $a_i \in \mathbb{Z}_2^n$, then **PGM**$(g, 1)$ solves* BFHSP_g *with success probability p_g such that*

$$\sqrt{p_g} \geq \sqrt{p_f} - \frac{2k}{\sqrt{2^n}}, \tag{6.114}$$

*where p_f is the success probability of **PGM**$(f, 1)$ for solving* BFHSP_f.

6.4.2 The "Grover" approach (quantum rejection sampling)

6.4.2.1 The basic algorithm

In this section we show how quantum rejection sampling can be used to solve the Boolean hidden shift problem. In particular, we show how to reduce an instance of BFHSP_f to $\mathrm{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ for some $\boldsymbol{\pi}$ and $\boldsymbol{\sigma}$, and thus prove an upper bound on quantum query complexity of BFHSP_f. The following theorem provides a trade-off between the quantum query complexity and success probability of BFHSP_f for any Boolean function f.

Theorem 5.17. *Let f be a Boolean function and $\hat{F}$ be its Fourier transform. For any success probability $p \in [p_{\min}, p_{\max}]$, we have $Q_{1-p}(\mathrm{BFHSP}_f) = O(1/\|\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}\|_2)$, where $p_{\min}$, $p_{\max}$, and $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ are given in Definition 5.8, and components of $\boldsymbol{\pi}$ and $\boldsymbol{\sigma}$ are given by $\pi_w = |\hat{F}(w)|$ and $\sigma_w = 1/\sqrt{2^n}$ for $w \in \mathbb{Z}_2^n$.*

Proof. Let us set $d = 2$ and $t = 1$ and use the quantum circuit from Sect. 6.3.1 to prepare the state

$$|\Phi^1(s)\rangle = \sum_{w\in\mathbb{Z}_2^n} (-1)^{w\cdot s} \hat{F}(w)|w\rangle. \tag{6.115}$$

If we could eliminate the Fourier coefficients $\hat{F}(w)$ from this state, we would obtain a state

$$|\psi(s)\rangle := \frac{1}{\sqrt{2^n}} \sum_{w\in\mathbb{Z}_2^n} (-1)^{w\cdot s}|w\rangle \tag{6.116}$$

from which the hidden shift s can be easily recovered by applying the Hadamard transform $H^{\otimes n}$. Unfortunately, the operation $\mathrm{diag}\big(\hat{F}(w)^{-1}/\sqrt{2^n} : w \in \mathbb{Z}_2^n\big)$ that uncomputes the Fourier coefficients is unitary only when f is a bent function (see Definition 6.7). Thus, instead we first apply a diagonal unitary[2] $\mathrm{diag}\big(\hat{F}(w)/|\hat{F}(w)| : w \in \mathbb{Z}_2^n\big)$ which corrects the phases of the Fourier coefficients and transforms $|\Phi^1(s)\rangle$ into

$$\sum_{w\in\mathbb{Z}_2^n} (-1)^{w\cdot s}|\hat{F}(w)||w\rangle. \tag{6.117}$$

Note that we can perform this transformation, since we know the function f and therefore its Fourier transform $\hat{F}$. Finally, observe that the problem of transforming the above state to $|\psi(s)\rangle$ is a special case of $\mathrm{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ with

$$\pi_w := |\hat{F}(w)|, \qquad \sigma_w := 1/\sqrt{2^n}, \qquad |\xi_w\rangle := (-1)^{w\cdot s}|0\rangle. \tag{6.118}$$

As a consequence, Theorem 5.1 immediately gives us a quantum algorithm for solving this problem. □

As is the case for classical rejection sampling, quantum rejection sampling can perform poorly if many amplitudes of the initial state are small. In particular, if we want the success probability to be one, the complexity of our algorithm is limited by the smallest Fourier coefficient of the function. If the smallest Fourier coefficient of f is zero, then the resampling approach cannot provide a solution with arbitrary high probability of success, since

$$p_{\max} = \sum_{w:\pi_w\neq 0} \sigma_w^2 = \sum_{w:\hat{F}(w)\neq 0} \frac{1}{2^n} = \frac{|\{w : \hat{F}(w) \neq 0\}|}{2^n}. \tag{6.119}$$

according to Definition 5.8. By ignoring small Fourier coefficients, one can decrease the complexity of the algorithm at the cost of a lower success probability.

Nevertheless, it is possible to boost the success probability of this algorithm. We show that this can be done either by using $t \geq 2$ queries in parallel (see Sect. 6.4.2.2) or by repetition (see Sect. 6.4.2.3). Note that the second strategy requires to construct a procedure for verifying a candidate shift. We propose such procedure based on a controlled-SWAP test in Sect. 6.5.2.

[2] If $\hat{F}(w) = 0$, we can choose the corresponding entry to be 1.

6.4.2.2 The t-fold algorithm

In this section we provide a generalization of the quantum algorithm described in the previous section by considering general values of parameters d and t. That is, we show how to solve the hidden shift problem for function $f : \mathbb{Z}_d^n \to \mathbb{Z}_d$ with any $d \geq 2$ by using quantum rejection sampling that performs $t \geq 1$ queries in parallel. Instead of $|\Phi^1(s)\rangle$, we use the t-fold Fourier state $|\Phi^t(s)\rangle$ from Sect. 6.3.1 for some $t \geq 1$. The advantage of choosing $t > 1$ is that we can avoid the problems that arise when f has small or zero Fourier coefficients. In particular, when $t \geq n$, the t-fold Fourier spectrum $\mathcal{F}^t$ (see Sect. 6.3.1.3) is guaranteed to be non-zero as will be shown in Sect. 6.6. The following result is a generalization of Theorem 5.17 for general values of t and d.

Theorem 6.14. *Let $f : \mathbb{Z}_d^n \to \mathbb{Z}_d$ be a function. For any integer $t \geq 1$ and success probability $p \in [p_{\min}, p_{\max}]$, we have $Q_{1-p}(\mathrm{BFHSP}_f) = O(t/\|\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}\|_2)$, where $p_{\min}$, $p_{\max}$, and $\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ are given in Definition 5.8, and components of $\boldsymbol{\pi}$ and $\boldsymbol{\sigma}$ are given by $\pi_w = \mathcal{F}^t(w)$, the t-fold Fourier spectrum of f (see Sect. 6.3.1.3), and $\sigma_w = 1/\sqrt{d^n}$ for $w \in \mathbb{Z}_d^n$.*

Proof. As in the $t = 1$ case, the main idea is to use quantum rejection sampling to drive the states

$$|\Phi^t(s)\rangle = \sum_{w\in\mathbb{Z}_d^n} \omega^{w\cdot s}|\mathcal{F}_w^t\rangle|w\rangle \tag{6.120}$$

towards a carefully chosen orthonormal basis, from which the hidden shift can be recovered by a measurement. One possibility is to eliminate the dependence on w from the second register of $|\Phi^t(s)\rangle$ and obtain the states

$$\sum_{w\in\mathbb{Z}_d^n} \omega^{w\cdot s}\frac{1}{\sqrt{d^n}}|\bar{0}\rangle|w\rangle \tag{6.121}$$

from which the hidden shift s can be easily recovered by measuring the second register in the Fourier basis. An alternative and equally good choice of the target states are the PGM states

$$|E_s\rangle = \sum_{w\in\mathbb{Z}_d^n} \omega^{w\cdot s}\frac{1}{\sqrt{d^n}}\frac{|\mathcal{F}_w^t\rangle}{\||\mathcal{F}_w^t\rangle\|_2}\otimes|w\rangle \tag{6.122}$$

from Eq. (6.85), which can be transformed into states in Eq. (6.121) by a controlled unitary operation. Thus, we have reduced the hidden shift problem to a quantum resampling problem $\mathrm{QSAMPLING}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ where

$$\pi_w := \mathcal{F}^t(w), \qquad \sigma_w := \frac{1}{\sqrt{d^n}}, \qquad |\xi_w\rangle := \omega^{w\cdot s}\frac{|\mathcal{F}_w^t\rangle}{\||\mathcal{F}_w^t\rangle\|_2}. \tag{6.123}$$

This generalizes the expressions for the $t = 1$ and $d = 2$ case in Eq. (6.118). The desired result immediately follows from Theorem 5.1. □

6.4.2.3 Boosting the success probability

When we want to find the hidden shift of f with success probability p close to one, the basic algorithm with $t = 1$ described in Sect. 6.4.2.1 (see Theorem 5.17) might be quite inefficient. In the special case $p = 1$, its complexity is limited by the smallest Fourier coefficient of f. In particular, when some Fourier coefficients are zero, it is not possible to achieve $p = 1$ using this algorithm.

One possible approach to deal with this problem is to choose $t \geq 2$ and use t queries in parallel as described in the previous section. As shown in Sect. 6.6.3, when t is chosen sufficiently large, the t-fold Fourier spectrum $\mathcal{F}^t$ is non-zero everywhere, so the hidden shift can be recovered with certainty (see Corollary 6.28).

Another approach to find the hidden shift with near certainty is to choose a smaller success probability p and repeat the algorithm until the right shift is obtained. However, this requires a procedure to check a candidate shift. Let us assume that we have such a procedure and it has *one-sided success probability* q, i.e., it accepts the correct shift with probability one and rejects a wrong shift with probability at least q (see Sect. 6.5.2 for more details). We can combine such imperfect checking procedure with the quantum search algorithm with bounded-error inputs from [HMdW03] to get the following result.

Proposition 6.15. *Let f be a Boolean function. Assume that we have a quantum algorithm that finds the hidden shift of f with complexity* F *and success probability p, and another algorithm that can verify a given shift with complexity* C *and one-side success probability q. Then we can combine these algorithms to solve* BFHSP_f *with arbitrary high success probability and complexity*

$$O\left(\frac{1}{\sqrt{p}}\left(\mathrm{F}+\frac{\mathrm{C}}{\sqrt{q}}\right)\right). \tag{6.124}$$

Proof. First, we boost the probability of rejecting a wrong candidate shift arbitrarily close to one using $O(1/\sqrt{q})$ steps of quantum amplitude amplification [BHMT00]. Thus, the total checking cost is $O(\mathrm{C}/\sqrt{q})$.

To boost the success probability p of the finding algorithm arbitrarily close to one, we use the quantum search algorithm with bounded-error inputs from [HMdW03]. Note

that we cannot apply the usual quantum amplitude amplification technique [BHMT00] directly, since this would incur an additional factor of $\log(1/p)$ as the checking operation is imperfect. Therefore, the total complexity comes from repeating $O(1/\sqrt{p})$ times the original algorithm with cost F and the checking operation with cost $O(\mathrm{C}/\sqrt{q})$. □

We will apply this result in Sect. 6.5.2 where we discuss specific procedures for verifying a given shift.

6.4.3 The "Simon" approach (sampling and classical post-processing)

We describe this approach only very briefly (for more details see [Röt10, GRR11]). The main idea is to use the quantum circuit from Sect. 6.3.2 to prepare the state

$$|\Psi^t(s)\rangle := \sum_{w \in \mathbb{Z}_d^n} |s \cdot w\rangle |\mathcal{F}_w^t\rangle |w\rangle \tag{6.125}$$

and measure it in the standard basis (the papers mentioned above consider only the $d = 2$ and $t = 1$ case). This gives us a random string $w \in \mathbb{Z}_d^n$, each with probability $\mathcal{F}^t(w)^2$, and the corresponding inner product $s \cdot w$. We can repeat this several times until the vectors w that we have obtained span the whole space $\mathbb{Z}_d^n$. Then by solving a system of linear equations we can obtain the hidden shift s. Below is a more formal description of this algorithm.

Simon's sampling(t)

1. Set $k := 0$.
2. While $\mathrm{span}\{w_i : 1 \le i \le k\} \neq \mathbb{Z}_d^n$ do:
   (a) Set $k \leftarrow k + 1$.
   (b) Prepare $|\Psi^t(s)\rangle$ with the circuit shown in Fig. 6.2 using t calls to the oracle $O_{f_{ks}}$.
   (c) Measure the first and last register of $|\Psi^t(s)\rangle$ and denote the outcomes by $b_k \in \mathbb{Z}_d$ and $w_k \in \mathbb{Z}_d^n$, respectively.
3. Solve the linear system of equations $w_i \cdot s = b_i$ $(1 \le i \le k)$ over $\mathbb{Z}_d$ to obtain the hidden shift s.

Gavinsky, Roetteler, and Roland [GRR11] bound the expected running time of this algorithm for any Boolean function f in terms of its influence I_f.

Theorem 6.16 ([GRR11]). *Let f be an n-argument Boolean function. **Simon's sampling**(1) solves* BFHSP_f *with success probability one and expected number of queries $O(n/\sqrt{\mathrm{I}_f})$ to the oracle $O_{f_{ks}}$ where I_f is the influence of f (see Sect. 6.2.4).*

Moreover, [GRR11] use this result to shown an exponential separation between average case quantum and classical query complexities.

Theorem 6.17 ([GRR11]). *Let $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ and $s \in \mathbb{Z}_2^n$ be chosen uniformly at random. **Simon's sampling**(1) can solve* BFHSP_f *with bounded error using $O(n)$ queries, whereas any classical algorithm needs $\Omega(\sqrt{2^n})$ queries to succeed with probability at least $1/2$.*

6.5 Quantum algorithms for related problems

In this section we consider two variations of the Boolean hidden shift problem. We provide a quantum algorithm for extracting only one chosen bit of information about the hidden shift, as well as show how to verify if the oracle is hiding a specific shift.

6.5.1 Parity extraction

In this section we consider the problem of extracting only one bit of information about the hidden shift—we want to determine the value of $w \cdot s$ for some specific $w \in \mathbb{Z}_2^n$. Our procedure uses only one query and succeeds with probability which depends on the Fourier coefficient of f corresponding to w.

Lemma 6.18. *Let f be a known Boolean function and $w \in \mathbb{Z}_2^n$ be a given input. Using one query to the oracle $O_{f_{ks}}$, a quantum algorithm can determine the value of $w \cdot s$ with success probability*

$$p = \frac{2\hat{F}(w)^2}{1 + \hat{F}(w)^2}. \tag{6.126}$$

Proof. Our algorithm is based on the procedure for preparing the state $|\Psi^1(s)\rangle$ from Sect. 6.3.2. The only difference is that we initialize the first register in an arbitrary superposition that we optimize over later. To avoid repeating the analysis from Sect. 6.3.2.2, we consider only two steps—the application of the oracle and the Fourier transform right after it.

We use two registers—the first register stores a qubit initialized in state $\alpha_0|0\rangle + \alpha_1|1\rangle$, but the second register stores n qubits initialized in the uniform superposition. Let us consider the transformation implemented by the k-controlled-phase oracle which acts as $O_{f_{ks}} : |k\rangle|x\rangle \mapsto (-1)^{f(x+ks)}|k\rangle|x\rangle$, followed by Fourier transform on the second register:

$$\sum_{k\in\mathbb{Z}_2} \alpha_k|k\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\mathbb{Z}_2^n} |x\rangle \mapsto \sum_{k\in\mathbb{Z}_2} \alpha_k|k\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\mathbb{Z}_2^n} (-1)^{f(x+ks)}|x\rangle \tag{6.127}$$

$$\mapsto \sum_{k\in\mathbb{Z}_2} \alpha_k|k\rangle \otimes \frac{1}{2^n} \sum_{x,y\in\mathbb{Z}_2^n} (-1)^{f(x+ks)+x\cdot y}|y\rangle \tag{6.128}$$

$$= \sum_{k\in\mathbb{Z}_2} \alpha_k|k\rangle \otimes \frac{1}{2^n} \sum_{x,y\in\mathbb{Z}_2^n} (-1)^{f(x)+(x+ks)\cdot y}|y\rangle \tag{6.129}$$

$$= \sum_{y\in\mathbb{Z}_2^n} \sum_{k\in\mathbb{Z}_2} \alpha_k(-1)^{k(s\cdot y)}|k\rangle \otimes \frac{1}{2^n} \sum_{x\in\mathbb{Z}_2^n} (-1)^{f(x)+x\cdot y}|y\rangle \tag{6.130}$$

$$= \sum_{y\in\mathbb{Z}_2^n} \big(\alpha_0|0\rangle + \alpha_1(-1)^{s\cdot y}|1\rangle\big)\hat{F}(y)|y\rangle. \tag{6.131}$$

Next, conditioned on the first register being in state $|0\rangle$, we apply a transformation on the second register. It acts as $\sum_{y\in\mathbb{Z}_2^n} \hat{F}(y)|y\rangle \mapsto |w\rangle$ where $w \in \mathbb{Z}_2^n$ is the desired bit string for which we want to determine the parity $w \cdot s$. This gives us state

$$\alpha_0|0\rangle|w\rangle + \alpha_1|1\rangle \sum_{y\in\mathbb{Z}_2^n} \hat{F}(y)(-1)^{s\cdot y}|y\rangle. \tag{6.132}$$

If we measure the second register, we get outcome w with probability

$$p_1 := \big\|\alpha_0|0\rangle + \alpha_1(-1)^{s\cdot w}\hat{F}(w)|1\rangle\big\|_2^2 = |\alpha_0|^2 + |\alpha_1|^2\hat{F}(w)^2, \tag{6.133}$$

and a post-measurement state that depends on the value of $w \cdot s$:

$$|\psi_{w\cdot s}\rangle := \frac{\alpha_0|0\rangle + \alpha_1(-1)^{s\cdot w}\hat{F}(w)|1\rangle}{\sqrt{|\alpha_0|^2 + |\alpha_1|^2\hat{F}(w)^2}}. \tag{6.134}$$

To determine the value of the bit $w \cdot s$, we use an unambiguous discrimination procedure for states $|\psi_0\rangle$ and $|\psi_1\rangle$. The best such procedure gives a conclusive outcome with probability $1 - |\langle\psi_0|\psi_1\rangle|$. In our case this is equal to

$$p_2 := 1 - \frac{\big||\alpha_0|^2 - |\alpha_1|^2\hat{F}(w)^2\big|}{|\alpha_0|^2 + |\alpha_1|^2\hat{F}(w)^2}, \tag{6.135}$$

so the total success probability is

$$p := p_1 p_2 = \begin{cases} 2|\alpha_1|^2 \hat{F}(w)^2 & \text{if } |\alpha_0|^2 \geq |\alpha_1|^2 \hat{F}(w)^2, \\ 2|\alpha_0|^2 & \text{otherwise.} \end{cases} \tag{6.136}$$

The optimal choice of amplitudes α_0 and α_1 in the initial state must satisfy

$$|\alpha_0|^2 = |\alpha_1|^2 \hat{F}(w)^2 \qquad \text{and} \qquad |\alpha_0|^2 + |\alpha_1|^2 = 1. \tag{6.137}$$

By solving these equations we get $|\alpha_1| = 1/\sqrt{1+\hat{F}(w)^2}$. This gives us the desired success probability in Eq. (6.126). □

If we would not optimize the amplitudes α_0 and α_1 but just choose $\alpha_0 = \alpha_1 = 1/\sqrt{2}$, the success probability would be equal to $\hat{F}(w)^2$. This is the same as the probability of getting outcome w (and thus learning $w \cdot s$) when measuring the state $|\Psi^1(s)\rangle$ in each iteration of the "Simon-like" algorithm in Sect. 6.4.3. Notice that for implementing the "Simon-like" algorithm we do not need any knowledge about the function f, whereas the algorithm discussed above requires knowledge of the Fourier spectrum of f to implement the step for preparing the state in Eq. (6.132). This illustrates that the knowledge of the function can be beneficial and how it can be used to increase the success probability. On the other hand, if the "Simon-like" algorithm fails to produce outcome w, it at least returns $w' \cdot s$ for some $w' \neq w$. This output still contains some information about s, whereas our algorithm completely looses all information in case of failure. Intuitively, we are gambling on "all or nothing".

6.5.2 Verification algorithms

Recall from Sect. 6.4.2.3 that procedure for verifying if the oracle is hiding some specific shift can be used to boost the success probability of a quantum algorithm that solves the hidden shift problem. More formally, we consider the following problem: given a candidate shift v and oracle access to the shifted function f_s for an unknown value of s, determine if $s = v$. Note that using an oracle for f_s we can always simulate an oracle for f_{s+v}, so without loss of generality we can consider the following simplified version of the problem: given an oracle for f_s, decide if $s = 0$.

6.5.2.1 Verification using SWAP-test

To verify classically that the oracle is hiding the unshifted function f_0, a simple strategy is to query the oracle on several inputs uniformly at random and see if all answers agree with f_0. This strategy has one-sides success probability equal to

$$\min_{w \in \mathbb{Z}_2^n \setminus \{0\}} \Pr_x \big[f(x) \neq f(x+w) \big] = \mathrm{I}_f, \tag{6.138}$$

the influence of f (see Sect. 6.2.4). Using $O(1/\mathrm{I}_f)$ repetitions of this procedure, we can boost its success probability arbitrarily close to one.

Let us show that on a quantum computer one can check a candidate shift with only $O(1/\sqrt{\mathrm{I}_f})$ oracle calls. Our quantum procedure is a natural quantum generalization of the above classical strategy and is based on a version of the SWAP-test [Wat00, BCWdW01].

Proposition 6.19 (SWAP-test)**.** *Given access to quantum circuits for preparing some unknown states $|\psi\rangle$ and $|\phi\rangle$, a quantum algorithm can tell if $|\psi\rangle = |\phi\rangle$ with one-sided success probability $\frac{1}{2}\big(1 - \mathrm{Re}\big[\langle\psi|\varphi\rangle\big]\big)$ by using each circuit once.*

Notice that the error probability is sensitive to the global phase of the states and it approaches one as the states approach each other.

Proof. Let V and W be unitaries corresponding to the quantum circuits for preparing states $|\psi\rangle$ and $|\varphi\rangle$, respectively: $V|0\rangle^{\otimes n} = |\psi\rangle$ and $W|0\rangle^{\otimes n} = |\varphi\rangle$. We use the controlled versions of these unitaries to apply the circuit shown in Fig. 6.3 and measure the first qubit. If the outcome is 0, we accept the two states as equal, otherwise we reject them as being different.

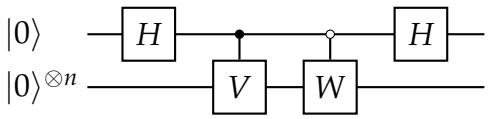

Figure 6.3: One-sided test for checking if states prepared by unitaries V and W are equal.

Let us analyze the success probability of this procedure. The final state at the end of this circuit is given by

$$\frac{1}{2}\Big[|0\rangle\big(|\psi\rangle + |\varphi\rangle\big) + |1\rangle\big(|\psi\rangle - |\varphi\rangle\big)\Big]. \tag{6.139}$$

If the first register is measured in the standard basis, we get outcome 0 ("accept") with probability

$$\frac{1}{4}\||\psi\rangle + |\varphi\rangle\|_2^2 = \frac{1}{2}\big(1 + \mathrm{Re}\big[\langle\psi|\varphi\rangle\big]\big). \tag{6.140}$$

If $|\psi\rangle = |\phi\rangle$, we always get outcome 0 and accept the states as equal. Otherwise, we get the correct outcome 1 with probability $\frac{1}{2}\big(1 - \mathrm{Re}\big[\langle\psi|\varphi\rangle\big]\big)$ and reject the states. □

Lemma 6.20. *Let f be a known Boolean function. Using one query to the oracle O_{f_s}, a quantum algorithm can verify if the hidden string is $s = 0$ with one-sided success probability I_f, the influence of f (see Sect. 6.2.4).*

Proof. We are given access to oracle O_{f_s} for some unknown value of $s \in \mathbb{Z}_2^n$ and we want to verify if $s = 0$. By applying the oracle to a uniform superposition over all strings we can prepare the state

$$|\phi_f(s)\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\mathbb{Z}_2^n} (-1)^{f(x+s)}|x\rangle. \tag{6.141}$$

Since we know the function f, we can prepare the state $|\phi_f(0)\rangle$ without using any queries, since we know the action of the oracle O_{f_0}. The inner product between these two states is

$$\begin{aligned}\langle\phi_f(s)|\phi_f(0)\rangle &= \frac{1}{2^n}\sum_{x\in\mathbb{Z}_2^n}(-1)^{f(x+s)+f(x)} && (6.142)\\ &= (F * F)(s) && (6.143)\\ &= 1 - 2\mathrm{I}_f(s) && (6.144)\end{aligned}$$

where the last equality follows from Eq. (6.40).

If we use the circuits for preparing states $|\phi_f(s)\rangle$ and $|\phi_f(0)\rangle$ in the SWAP-test described in Prop. 6.19, we can decide if $s = 0$ with one-sided success probability

$$\frac{1 - \langle\phi_f(s)|\phi_f(0)\rangle}{2} = \mathrm{I}_f(s). \tag{6.145}$$

That is, we always accept if $s = 0$, and reject with probability at least I_f otherwise. □

According to Prop. 6.15, we can boost the success probability of the quantum algorithm from Sect. 6.4.2.1 (see Theorem 5.17) using the imperfect checking procedure from Lemma 6.20.

Corollary 6.21. *Let f be a Boolean function and $\hat{F}$ be its Fourier transform. For any value of parameter $p \in [p_{\min}, p_{\max}]$, a quantum algorithm can solve* BFHSP_f *with arbitrary high success probability and complexity*

$$O\left(\frac{1}{\sqrt{p}}\left(\frac{1}{\|\boldsymbol{\varepsilon}^{p}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}\|_2}+\frac{1}{\sqrt{\mathrm{I}_f}}\right)\right), \tag{6.146}$$

where $p_{\min}$, $p_{\max}$, and $\boldsymbol{\varepsilon}^{p}_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$ are given in Definition 5.8, and components of $\boldsymbol{\pi}$ and $\boldsymbol{\sigma}$ are given by $\pi_w = |\hat{F}(w)|$ and $\sigma_w = 1/\sqrt{2^n}$ for $w \in \mathbb{Z}_2^n$.

6.5.2.2 Verification using SWAP-test with arbitrary weights

In this section we consider a slight generalization of the verification procedure from the previous section, where we use an arbitrary superposition as the initial state instead of the uniform superposition.

The classical equivalent of this procedure is as follows. We choose a probability distribution over $\mathbb{Z}_2^n$ and query the oracle O_{f_s} several times according to this distribution. If no disagreement between f and f_s is found on these inputs, we conclude that $s = 0$.

In the quantum case we start with an arbitrary normalized state $\sum_{x\in\mathbb{Z}_2^n} \alpha_x|x\rangle$ with amplitudes $\alpha_x \in \mathbb{C}$ that we will optimize over later. We use one query to O_{f_s} and the knowledge of f to prepare states $|\phi_f(s)\rangle$ and $|\phi_f(0)\rangle$, respectively, where

$$|\phi_f(s)\rangle := \sum_{x\in\mathbb{Z}_2^n} \alpha_x(-1)^{f(x+s)}|x\rangle. \tag{6.147}$$

Notice that the inner product between these two states is

$$\langle\phi_f(0)|\phi_f(s)\rangle = \sum_{x\in\mathbb{Z}_2^n} |\alpha_x|^2(-1)^{f(x)+f(x+s)}. \tag{6.148}$$

If we use the SWAP-test from Prop. 6.19 to compare these states, the probability of con-

cluding that $s \neq 0$ is

$$p(s) := \frac{1}{2}\big(1 - \mathrm{Re}\big[\langle \phi_f(0)|\phi_f(s)\rangle\big]\big) \tag{6.149}$$

$$= \frac{1}{2}\Big(1 - \sum_{x\in\mathbb{Z}_2^n} |\alpha_x|^2 (-1)^{f(x)+f(x+s)}\Big) \tag{6.150}$$

$$= \sum_{x\in\mathbb{Z}_2^n} |\alpha_x|^2 \frac{1}{2}\Big(1 - (-1)^{f(x)+f(x+s)}\Big) \tag{6.151}$$

$$= \sum_{x\in\mathbb{Z}_2^n} |\alpha_x|^2 \big[f(x) \oplus f(x+s)\big]. \tag{6.152}$$

This probability is zero if $s = 0$, so we never make a mistake in this case. However, if $s \neq 0$ we conclude so with probability $p(s)$. To optimize the worst case success probability, we have to maximize the minimal value of $p(s)$ for $s \neq 0$. Note that it suffices to perform the optimization over a probability distribution $p_x := |\alpha_x|^2$. If we define a Boolean matrix

$$M_{sx} := f(x) \oplus f(x+s) \tag{6.153}$$

and denote the worst case success probability by P_f, we obtain the following linear optimization problem:

$$P_f := \max_{p\in\mathbb{R}^{2^n}} q \quad \text{s.t.} \quad \sum_{x\in\mathbb{Z}_2^n} M_{sx} p_x \geq q \ (\forall s \neq 0), \quad \sum_{x\in\mathbb{Z}_2^n} p_x = 1,\ p_x \geq 0. \tag{6.154}$$

In general, $P_f \geq \mathrm{I}_f$ and there are examples where this inequality is strict. Thus, we can strengthen Lemma 6.20 and Corollary 6.21 by replacing the influence I_f with the quantity P_f defined above.

Interestingly, the same linear program also characterizes the optimal success probability of the classical strategy described at the beginning of this section, where each input x is queried independently with probability p_x. The difference between the quantum and the classical case is that in the classical case we need $O(1/P_f)$ repetitions to boost the success probability arbitrarily close to one, whereas in the quantum case only $O(1/\sqrt{P_f})$ repetitions suffice using amplitude amplification [BHMT00].

6.5.2.3 Greedy classical verification procedure

Here is a simple greedy adaptive classical strategy for deciding if the hidden shift is 0. Observe that a query x to the oracle eliminates those shifts s for which $f(x+s) \neq f(x)$. This is equivalent to $M_{sx} = 1$, where M is the Boolean matrix defined in Eq. (6.153). We can use a greedy strategy and choose the first query $x_1 \in \mathbb{Z}_2^n$ so that it eliminates as many values of s as possible. This is equivalent to choosing a column x_1 of matrix M that contains the most number of ones. Next, we consider a submatrix of M where we remove column x_1 and those rows s where $M_{sx_1} = 1$, and repeat the same procedure. When all rows apart from $s = 0$ have been eliminated, we are done, and the obtained sequence $x_1, \ldots, x_m$ is a certificate for verifying that $s = 0$ using m queries.

6.6 Zeroes in the Fourier spectrum

Recall from Sect. 6.4.2 that the success probability of the quantum rejection sampling approach for solving the Boolean hidden shift problem is limited when the function f has zero Fourier coefficients. In this section we study the structure of zero Fourier coefficients of Boolean functions and provide a way to resolve this problem. First, we show that a Boolean function can have many zero Fourier coefficients if it is degenerate (in the sense that it is invariant under some shifts). Next, we argue that there even for non-degenerate Boolean functions it is possible to have Fourier spectrum that is equal to zero almost everywhere. Finally, we show that for any non-degenerate Boolean function f we can choose a sufficiently large value of t, the number of parallel queries, so that the t-fold Fourier spectrum of f is guaranteed to be non-zero everywhere. Throughout this section we consider only the Boolean case $d = 2$.

6.6.1 Undetectable shifts and anti-shifts

In some cases the Boolean hidden shift problem cannot be solved in principle, since the function f is invariant under some shift, so the hidden shift cannot be uniquely determined (an extreme case of this is a constant function which is invariant under all shifts). In this section we consider functions with such property and analyze their Fourier spectrum.

Definition 6.22. Let $b \in \mathbb{Z}_2$. We say that s is a *b-shift* for function f, if f has the following property:

$$\forall x \in \mathbb{Z}_d^n : f(x+s) = f(x) + b. \tag{6.155}$$

We refer to 0-shifts as *undetectable shifts*, since they cannot be distinguished from the trivial shift $s = 0$. We also refer to 1-shifts as *anti-shifts*, since they negate the truth table.

The following result provides an alternative characterization of b-shifts. It relates the maximal and minimal autocorrelation value of F to undetectable shifts and anti-shifts of f, respectively (recall the definition of convolution from Sect. 6.2.3).

Proposition 6.23. *String $s \in \mathbb{Z}_2^n$ is a b-shift for function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ if and only if*

$$(F * F)(s) = (-1)^b, \tag{6.156}$$

where $F(x) := (-1)^{f(x)}/\sqrt{2^n}$ for all $x \in \mathbb{Z}_2^n$.

Proof. Let s be a b-shift of f. Then

$$(F * F)(s) = \sum_{x \in \mathbb{Z}_2^n} F(x)F(x+s) \tag{6.157}$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{f(x)}(-1)^{f(x)+b} \tag{6.158}$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{b} \tag{6.159}$$

$$= (-1)^b. \tag{6.160}$$

To prove the converse, note that all terms on the right hand side of Eq. (6.157) have absolute value equal to $1/2^n$. Since in total there are 2^n terms, $|(F * F)(s)| \leq 1$. If this inequality is saturated, then all terms in Eq. (6.157) must have the same phase. Thus, s is a b-shift for some $b \in \mathbb{Z}_2$. □

If s' and s'' are undetectable shifts of f then so is $s' + s''$, since $f(x + s' + s'') = f(x + s') = f(x)$ for any x. Hence, all undetectable shifts form a linear subspace of $\mathbb{Z}_2^n$. Also, if a' and a'' are anti-shifts, then $a' + a''$ is an undetectable shift. In particular, a Boolean function with no undetectable shifts can have at most one anti-shift.

If we want to solve the hidden shift problem for function f that has an undetectable shift s, we can apply an invertible linear transformation A on the input variables such that $A \cdot 0 \ldots 01 = s$. In this way we simulate the oracle for function $f'(x) := f(A \cdot x)$ such that $f'(x + 0 \ldots 01) = f'(x)$. Notice that f' effectively is an $(n-1)$-argument function, since it does not depend on the last argument. Similarly, if f has a k-dimensional

subspace of undetectable shifts, it is effectively an $(n-k)$-argument function. Solving the hidden shift problem for such function is equivalent to solving it for the reduced $(n-k)$-argument function f' and picking arbitrary values for the remaining k arguments. In this sense, Boolean functions with undetectable shifts are degenerate and we can without loss of generality consider only functions with no undetectable shifts.

Similarly, if f has an anti-shift, we can use the same construction to show that it is equivalent to a function f' such that

$$f'(x_1, \ldots, x_{n-1}, x_n) = f''(x_1, \ldots, x_{n-1}) \oplus x_n \tag{6.161}$$

where f'' is an $(n-1)$-argument function. To solve the hidden shift problem for f', we first solve it for f'' and then learn the value of the remaining argument x_n via a single query. In this sense, Boolean functions with anti-shifts are also degenerate. Thus, without loss of generality we can consider the hidden shift problem only for *non-degenerate* functions, i.e., ones that have no b-shifts for any $b \in \mathbb{Z}_2$.

Finally, let us show that such degenerate Boolean functions with b-shifts have at least half of their Fourier coefficients equal to zero. Let $\mathcal{S}$ be an $(n-1)$-dimensional subspace of $\mathbb{Z}_2^n$, and let us denote the two cosets of $\mathcal{S}$ in $\mathbb{Z}_2^n$ by $\mathcal{S}_b := \mathcal{S} + br$, where $b \in \mathbb{Z}_2$ and $r \in \mathbb{Z}_2^n \setminus \mathcal{S}$ is any representative of the coset for $b = 1$. The following result relates the property of having a b-shift to the property of having zero Fourier coefficients with special structure.

Lemma 6.24. *Boolean function f has a b-shift if and only if there is an $(n-1)$-dimensional subspace $\mathcal{S} \subset \mathbb{Z}_2^n$ such that $\hat{F}(w) = 0$ when $w \notin \mathcal{S}_b$.*

Proof. Assume that s is a b-shift of f. Then

$$\hat{F}(w) = \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{w \cdot x + f(x)} \tag{6.162}$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{w \cdot (x+s) + f(x+s)} \tag{6.163}$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{w \cdot (x+s) + f(x) + b} \tag{6.164}$$

$$= (-1)^{w \cdot s + b} \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{w \cdot x + f(x)} \tag{6.165}$$

$$= (-1)^{w \cdot s + b} \hat{F}(w). \tag{6.166}$$

Thus, $\hat{F}(w) = 0$ when $w \cdot s \neq b$. Let $\mathcal{S}$ be the $(n-1)$-dimensional subspace of $\mathbb{Z}_2^n$ orthogonal to s. Then $w \in \mathcal{S}_b \Leftrightarrow w \cdot s = b$ and thus $\hat{F}(w) = 0$ when $w \notin \mathcal{S}_b$.

For the converse, assume that $\mathcal{S}$ is an $(n-1)$-dimensional subspace of $\mathbb{Z}_2^n$ and $\hat{F}(w) = 0$ when $w \notin \mathcal{S}_b$. Let $s \in \mathbb{Z}_2^n$ be the unique non-zero vector orthogonal to $\mathcal{S}$. Then $\mathcal{S}_b = \{w \mid w \cdot s = b\}$ and we have

$$F(x+s) = \hat{\hat{F}}(x+s) \tag{6.167}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{w \in \mathbb{Z}_2^n} (-1)^{(x+s)\cdot w} \hat{F}(w) \tag{6.168}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{w \in \mathcal{S}_b} (-1)^{(x+s)\cdot w} \hat{F}(w) \tag{6.169}$$

$$= (-1)^b \frac{1}{\sqrt{2^n}} \sum_{w \in \mathcal{S}_b} (-1)^{x\cdot w} \hat{F}(w) \tag{6.170}$$

$$= (-1)^b F(x). \tag{6.171}$$

Hence, $f(x+s) = f(x) + b$ and thus s is a b-shift of f. □

6.6.2 Decision trees

In previous section we covered some degenerate cases of Boolean functions that have many zero Fourier coefficients. However, there still remains a question whether there are any *interesting* families of Boolean functions with many zero Fourier coefficients. Recall that for such functions it is hard to solve the hidden shift problem using the quantum rejection sampling approach. In this section we explain why such functions indeed exist and how to construct them, and in Sect. 6.6.3 we show how the quantum rejection sampling approach can be modified to deal with such functions.

Let us consider a specific way of defining Boolean functions using decision trees. A *decision tree* is a binary tree whose vertices are labeled by arguments of f, but leaves contain the values of f. To evaluate $f(x)$ for given string $x \in \mathbb{Z}_2^n$, we first look at the value of bit x_r where "x_r" is the label of the root node. If $x_r = 0$, we proceed to the left subtree and apply the same procedure, otherwise we evaluate the right subtree. Once we reach a leaf, its label is the value of $f(x)$. Without loss of generality we can consider only decision trees where on each path from the root to a leaf no argument appears more than once, otherwise some parts of the tree would not be reachable. The length of the longest path from the root to a leaf is the *height* of the tree.

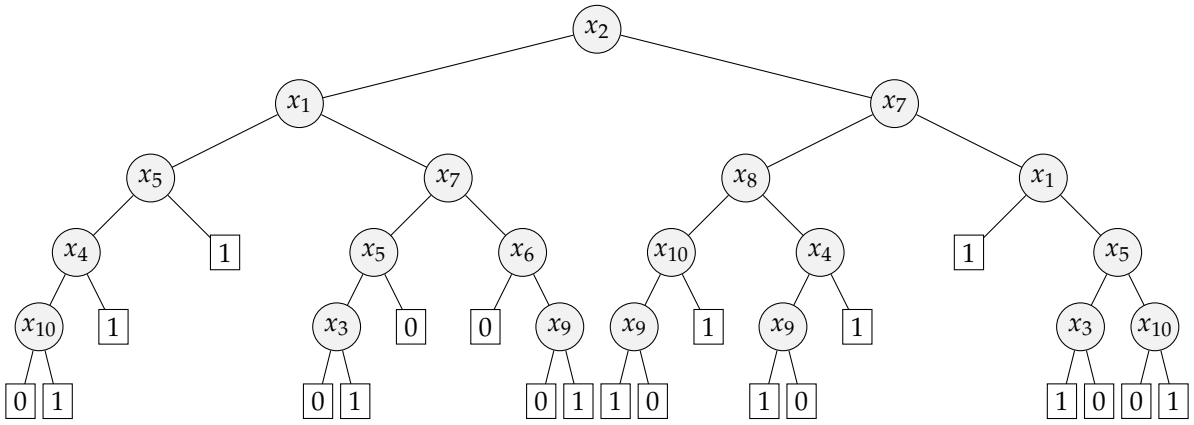

Figure 6.4: Decision tree for a 10-argument Boolean function f_{10}. To compute the value of the function for given input $x_1, \ldots, x_{10} \in \mathbb{Z}_2^n$, proceed down the tree starting from the root; move left if the corresponding argument is equal to 0 or right if it is equal to 1. Once a leaf is reached, its label is the value of the function for the given input.

An example of a decision tree is given in Fig. 6.4. From this tree we see, for example, that $f_{10}(x_1, \ldots, x_{10})$ evaluates to zero when $x_2 = x_1 = x_5 = x_4 = x_{10} = 0$, since the leftmost leave has label zero. This tree has height five.

Lemma 6.25. *If a Boolean function f is defined by a decision tree of height h, then all Fourier coefficients of f with Hamming weight larger than h are zero: $\hat{F}(w) = 0$ when $|w| > h$.*

Proof. Since the Boolean function f is given by a decision tree, let $\{P_1, \ldots, P_m\}$ be the set of all paths that start at the root of this tree and end at a parent of a leaf labeled by 1. For example, $P_1 = \{x_2, x_1, x_5, x_4, x_{10}\}$ and $P_2 = \{x_2, x_7, x_1\}$ are two such paths for the tree shown in Fig. 6.4. We can write the disjunctive normal form of f as

$$f(x) = \bigvee_{i=1}^{m} \bigwedge_{j \in P_i} (b_j^{(i)} \oplus x_j) \tag{6.172}$$

where "$\vee$" and "$\wedge$" represent logical OR and AND functions, respectively, and $b_j^{(i)} \in \mathbb{Z}_2$ is equal to one if and only if variable x_j has to be negated on path P_i. For example, x_{10} is negated on P_1, and x_2 and x_7 are negated on P_2.

To prove the desired result about the Fourier coefficients of f, we want to switch from Boolean functions to (± 1)-valued functions with (± 1)-valued variables. In particular, we want to replace $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ by a function $\tilde{F} : \{1, -1\}^n \to \{1, -1\}$ in variables

$X_i \in \{1, -1\}$ such that

$$\tilde{F}\big((-1)^x\big) = (-1)^{f(x)} \tag{6.173}$$

for all $x \in \mathbb{Z}_2^n$.

Notice that the (± 1)-valued versions of logical NOT, AND, and OR functions are given by the following polynomials:

$$\mathrm{NOT}(X) := -X, \tag{6.174}$$

$$\mathrm{AND}(X_1, \ldots, X_k) := \quad 1 - 2\prod_{i=1}^{k} \frac{1 - X_i}{2}, \tag{6.175}$$

$$\mathrm{OR}(X_1, \ldots, X_k) := -1 - 2\prod_{i=1}^{k} \frac{1 + X_i}{2}. \tag{6.176}$$

We can use these polynomials and Eq. (6.172) to write $\tilde{F}$ as

$$\tilde{F}(X) = \mathrm{OR}_{i=1}^{m} \mathrm{AND}_{j \in P_i} (-1)^{b_j^{(i)}} X_j, \tag{6.177}$$

where $\mathrm{OR}_{i=1}^{m} X_i$ stands for $\mathrm{OR}(X_1, \ldots, X_m)$ and a similar convention is used for AND.

When we determine the value of f using a decision tree, each input $x \in \mathbb{Z}_2^n$ leads to a unique leaf of the tree. Thus, when $f(x) = 1$, there is a unique value of i in Eq. (6.172) for which the corresponding term in disjunction is satisfied. With this promise we can simplify Eq. (6.176) to

$$\mathrm{OR}(X_1, \ldots, X_k) := \sum_{i=1}^{k} (X_i - 1) + 1. \tag{6.178}$$

If we plug this in Eq. (6.177), we get

$$\tilde{F}(X) = \sum_{i=1}^{m} \Big(\mathrm{AND}_{j \in P_i} (-1)^{b_j^{(i)}} X_j - 1 \Big) + 1, \tag{6.179}$$

$$= 1 - 2\sum_{i=1}^{m} \prod_{j \in P_i} \frac{1 - (-1)^{b_j^{(i)}} X_j}{2}. \tag{6.180}$$

Notice that this polynomial has degree at most $\max_i |P_i| \leq h$, the height of the tree. On the other hand, Fourier transform is self-inverse (see Prop. 6.2), so

$$(-1)^{f(x)} = \sqrt{2^n} F(x) = \sqrt{2^n} \hat{\tilde{F}}(x) = \sum_{w \in \mathbb{Z}_2^n} (-1)^{x \cdot w} \hat{F}(w). \tag{6.181}$$

The (± 1)-valued equivalent of this equation is

$$\tilde{F}(X) = \sum_{w \in \mathbb{Z}_2^n} \hat{F}(w) \prod_{i\,:\,w_i=1} X_i. \tag{6.182}$$

By comparing this with Eq. (6.180) we conclude that $\hat{F}(w) = 0$ when $|w| > h$. □

According to this lemma, we can use the following strategy to construct Boolean functions which have a large fraction of the Fourier coefficients equal to zero. We pick a random decision tree with many variables but a small height, i.e., large n and small h (notice that $n \leq 2^h - 1$). In this way we are guaranteed that the fraction of non-zero Fourier coefficients does not exceed

$$\frac{1}{2^n} \sum_{k=0}^{h} \binom{n}{k} \leq \frac{2^{H(\frac{h}{n})n}}{2^n} = \left(\frac{1}{2^n}\right)^{1-H(\frac{h}{n})} \tag{6.183}$$

where H is the binary entropy function. In particular, if $h \sim \log_2 n$ then this fraction vanishes as n goes to infinity, i.e., $\hat{F}$ is zero almost everywhere.

However, notice that when the number of zero Fourier coefficients is large, it is also more likely to pick a degenerate Boolean function (i.e., one that has a b-shift for some $b \in \mathbb{Z}_2$) which is something we would like to avoid. Recall from Lemma 6.24 that f has a b-shift if all its non-zero Fourier coefficients lie in a coset $\mathcal{S}_b$ of some $(n-1)$-dimensional subspace $\mathcal{S} \subset \mathbb{Z}_2^n$. Unfortunately, we do not know what is the probability that a random decision tree with n variables and height $\log_2 n$ corresponds to a Boolean function with this property.

On the other hand, we succeeded in using the approach described above to numerically find a Boolean function with the desired properties. In particular, we found a 10-argument Boolean function f_{10} whose decision tree is shown in Fig. 6.4. This function has no undetectable shifts or anti-shifts, while at the same time it has 928 (out of $2^{10} = 1024$) Fourier coefficients equal to zero.

Nevertheless, we also observed that by increasing the value of t we can very efficiently eliminate zeroes from the t-fold Fourier spectrum $\mathcal{F}^t$. We computed that for f_{10} the fraction of non-zero values of $\mathcal{F}^t$ for $t = 1, 2, 3, 4$ goes as 0.09, 0.61, 0.94, 1. In particular, $\mathcal{F}^4$ is non-zero everywhere. This observation is the topic of the next section, and the motivation why we constructed the t-fold quantum rejection sampling algorithm for the hidden shift problem (see Sect. 6.4.2.2).

6.6.3 Zeroes in the t-fold Fourier spectrum

Query complexity of the quantum algorithms discussed in previous sections of this chapter in one or another way depends on the Fourier spectrum of the function f considered. Moreover, query complexity of algorithms that use the t-fold Fourier states $|\Phi^t(s)\rangle$ and $|\Psi^t(s)\rangle$ for some $t > 1$ (see Sect. 6.3) depends on the generalized Fourier coefficients $\mathcal{F}^t$ (see Sect. 6.3.1.3) which we call t-fold Fourier spectrum of f:

$$\mathcal{F}^t(w) = \sqrt{\left[\hat{F}^2\right]^{*t}(w)}. \tag{6.184}$$

Note that in the limiting case $t = 1$ these are simply the absolute values of the Fourier coefficients:

$$\mathcal{F}^1(w) = |\hat{F}(w)|. \tag{6.185}$$

In this section we study the zeroes of the t-fold Fourier spectrum $\mathcal{F}^t$ of f as a function of t. The main result of this section is Lemma 6.27 which shows that unless f has an undetectable shift, $\mathcal{F}^t$ becomes non-zero everywhere when t is sufficiently large. This means that even for functions with high density of zeroes in the Fourier spectrum, one can boost the success probability of the basic quantum rejection sampling approach discussed in Sect. 6.4.2.1 by using the t-fold generalization from Sect. 6.4.2.2.

Proposition 6.26. *Let $S_t := \{w \in \mathbb{Z}_2^n : \mathcal{F}^t(w) \neq 0\}$ be the set of strings for which $\mathcal{F}^t$ is non-zero. Then $S_{t+1} = S_t + S_1$.*

Proof. Note from Eq. (6.184) that

$$\left[\mathcal{F}^{t+1}\right]^2 = \left[\mathcal{F}^t\right]^2 * \left[\mathcal{F}^1\right]^2 \tag{6.186}$$

and $\mathcal{F}^t(w) \geq 0$ for any t and w. Assume that $w_0 \in S_t$ and $w_1 \in S_1$. Then $\mathcal{F}^t(w_0) > 0$ and $\mathcal{F}^1(w_1) > 0$, hence

$$\left[\mathcal{F}^{t+1}\right]^2(w_0 + w_1) = \sum_{x \in \mathbb{Z}_d^n} \left[\mathcal{F}^t\right]^2(x) \cdot \left[\mathcal{F}^1\right]^2(w_0 + w_1 - x) \tag{6.187}$$

$$\geq \left[\mathcal{F}^t\right]^2(w_0) \cdot \left[\mathcal{F}^1\right]^2(w_0 + w_1 - w_0) > 0. \tag{6.188}$$

Thus $w_0 + w_1 \in S_{t+1}$ and hence $S_t + S_1 \subseteq S_{t+1}$. Conversely, if w cannot be written in the form $w_0 + w_1$ for some $w_0 \in S_t$ and $w_1 \in S_1$ then $\mathcal{F}^{t+1}(w) = 0$, since all terms in the sum in Eq. (6.187) vanish. □

Approach	Section	Functions:		
		delta	bent	random
PGM	Sect. 6.4.1	$O(2^n)$	$O(1)$	$O(1)$?
"Grover" ($t = 1$)	Sect. 6.4.2	$O(\sqrt{2^n})$	$O(1)$	$O(1)$?
"Simon" ($t = 1$)	Sect. 6.4.3	$O(n\sqrt{2^n})$	$O(n)$	$O(n)$
Lower bounds:		$\Omega(\sqrt{2^n})$	$\Omega(1)$	$\Omega(1)$?

Table 6.1: Summary of quantum query complexity upper bounds for the Boolean function hidden shift problem. Question marks indicate conjectured values.

From this result we can draw the following useful conclusion.

Lemma 6.27. *If a Boolean function f does not have an undetectable shift, then there exists $t \in \{1, \ldots, n\}$ such that $\mathcal{F}^t$ is non-zero everywhere.*

Proof. If S_1 spans the whole space $\mathbb{Z}_2^n$, we can inductively apply Prop. 6.26 to conclude that $S_t = \mathbb{Z}_2^n$ for some sufficiently large t. In particular, it suffices to take $t \leq n$ (say, if S_1 is the standard basis). On the other hand, if S_1 spans only a proper subspace of $\mathbb{Z}_2^n$, then it is contained in some $(n-1)$-dimensional subspace $\mathcal{S}_0$. Since $\mathcal{F}^1 = |\hat{F}|$ vanishes outside of $\mathcal{S}_0$, we conclude by Lemma 6.24 that f has an undetectable shift. □

This result together with Theorem 6.14 implies that as long as f is a Boolean function that does not have an undetectable shift, we can always use quantum rejection sampling with some $t \leq n$ to recover the hidden shift s with certainty.

Corollary 6.28. *Let f be an n-argument Boolean function. For any sufficiently large success probability p there exists $t \in \{1, \ldots, n\}$ such that $Q_{1-p}(\mathrm{BFHSP}_f) = O(t/\|\boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}\|_2)$, where $\pi_w = \mathcal{F}^t(w)$ and $\sigma_w = 1/\sqrt{2^n}$.*

6.7 Conclusions

A comparison of the quantum query complexity upper bounds for solving the Boolean function hidden shift problem for different classes of functions using the three approaches from Sect. 6.4 are given in Table 6.1. Entries with question marks indicate conjectured values.

If these conjectures hold, the "Grover-like" algorithm is optimal in all three cases. However, we know that it performs very poorly when f has lots of zero Fourier coefficients, which is the case, say, for decision trees (see Sect. 6.6.2). This suggests that the "Grover-like" approach (with $t = 1$) might not be optimal in general. Nevertheless, the $t \geq 1$ case does not have these deficiencies (see Corollary 6.28) and could potentially be optimal in general.

The "Simon-like" approach always has an overhead by a factor of order n, which is related to the fact that we need at least n linearly independent equations to solve the linear system. Finally, the PGM approach performs well in the "easy cases", i.e., bent and random functions, but fails to provide any speedup for delta functions. This can be attributed to the fact that Grover's algorithm is intrinsically sequential and cannot be parallelized (by doing all queries in parallel and then post-processing the outcomes).

We conclude that none of the three basic algorithms considered above is optimal. However, there is a hope that by combining these algorithms and possibly adding some new ideas, one might be able to obtain an algorithm that is optimal for all Boolean functions. In particular, the "Grover-like" approach with $t \geq 1$ seems to be promising.

6.7.1 Open problems

The two main open problems are the following:

- Find a query-optimal quantum algorithm for the hidden shift problem for functions of the form $f : \mathbb{Z}_d^n \to \mathbb{Z}_d$.
- Prove a matching lower bound on the quantum query complexity of this problem (say, using the adversary method).

Here are some presumably easier questions that might help to answer the harder ones:

1. Find new interesting classes of Boolean functions lying somewhere in the middle between the two extreme cases of bent and delta functions (say, the decision trees considered in Sect. 6.6.2), and characterize the quantum query complexity of the hidden shift problem for these functions.
2. Understand the quantum query complexity of the hidden shift problem for random functions (numerical evidence suggests that it is $\Theta(1)$ for both the "Grover-like" as well as the PGM approach).
3. What is the quantum query complexity of verifying a given shift? (See Sect. 6.5.2).
4. What is the quantum query complexity of extracting one bit of information about the hidden shift? (See Sect. 6.5.1).

5. What is the classical query complexity of the hidden shift problem for functions?
6. Can we say anything non-trivial about the time complexity of the hidden shift problem for $f : \mathbb{Z}_d^n \to \mathbb{Z}_d$ in the classical or quantum case?

APPENDICES

Appendix A

Water-filling vector is optimal for the SDP

Lemma 5.9. *Let $p \in [p_{\min}, p_{\max}]$, and $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$. Then, the following SDP*

$$\begin{aligned} \max_{M\succeq 0} \operatorname{Tr} M \quad \text{s.t.} \qquad & \forall k : \pi_k^2 \geq M_{kk}, \\ & \operatorname{Tr}\left[(\boldsymbol{\sigma}\cdot\boldsymbol{\sigma}^{\mathsf{T}} - pI)M\right] \geq 0. \end{aligned} \tag{5.17}$$

has optimal value $\|\boldsymbol{\varepsilon}\|_2^2$, which is achieved by the rank-1 matrix $M = \boldsymbol{\varepsilon}\cdot\boldsymbol{\varepsilon}^{\mathsf{T}}$.

Proof. We now show that the optimal value of the SDP in Eq. (5.17) can be attained by a rank-1 matrix M. Imposing the additional constraint that M can be written as $M = \boldsymbol{\varepsilon}\cdot\boldsymbol{\varepsilon}^{\mathsf{T}}$ for some $\boldsymbol{\varepsilon} \in \mathbb{R}^n$, the optimization problem in Eq. (5.17) reduces to

$$\begin{aligned} \max_{\varepsilon_k\geq 0} \|\boldsymbol{\varepsilon}\|_2^2 \quad \text{s.t.} \qquad & \forall k : \pi_k \geq \varepsilon_k \geq 0, \\ & \boldsymbol{\sigma}^{\mathsf{T}}\cdot\hat{\boldsymbol{\varepsilon}} \geq \sqrt{p}, \end{aligned} \tag{A.1}$$

where $\hat{\boldsymbol{\varepsilon}} := \boldsymbol{\varepsilon}/\|\boldsymbol{\varepsilon}\|_2$ denotes a unit vector in direction $\boldsymbol{\varepsilon}$.

We show that the optimal value is attained by $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^p_{\boldsymbol{\pi}\to\boldsymbol{\sigma}}$. Recall that according to Definition 5.8, we have $\varepsilon_k = \min\{\pi_k, \gamma\sigma_k\}$ and

$$\boldsymbol{\sigma}^{\mathsf{T}}\cdot\hat{\boldsymbol{\varepsilon}} = \sqrt{p}, \tag{A.2}$$

so that this vector satisfies the constraints of the problem in Eq. (A.1) and is therefore a feasible point. As a consequence $M = \boldsymbol{\varepsilon}\cdot\boldsymbol{\varepsilon}^{\mathsf{T}}$ is also a feasible point for the SDP in Eq. (5.17), which implies that its objective value is at least $\operatorname{Tr} M = \|\boldsymbol{\varepsilon}\|_2^2$.

We now want to find a feasible dual solution that gives the same objective value for the dual of SDP in Eq. (5.17), which can be written as [VB96]

$$\min_{\lambda_k \geq 0,\, \mu \geq 0} \sum_{k=1}^{n} \lambda_k \pi_k^2 \quad \text{s.t.} \quad \Lambda - I + \mu(pI - \boldsymbol{\sigma} \cdot \boldsymbol{\sigma}^{\mathsf{T}}) \succeq 0, \tag{A.3}$$

where $\Lambda := \operatorname{diag}(\lambda_k \mid k = 1, \ldots, n)$. Indeed, if an objective value is feasible for both the primal and the dual, it implies that this is the optimal value.

We prove that the following solution is feasible for the dual:

$$\lambda_k = \mu\left(\frac{\sigma_k}{\varepsilon_k} \sum_{l=1}^{n} \sigma_l \varepsilon_l - p\right) + 1, \tag{A.4}$$

$$\mu = \frac{1 - \|\boldsymbol{\varepsilon}\|_2^2}{p - \left(\sum_{l=1}^{n} \sigma_l \varepsilon_l\right) \cdot \left(\sum_{k=1}^{n} \frac{\sigma_k \pi_k^2}{\varepsilon_k}\right)}. \tag{A.5}$$

This choice yields $\|\boldsymbol{\varepsilon}\|_2^2$ as the dual objective value, so it remains to show that it satisfies the constraints in Eq. (A.3). Let us first prove that $\mu \geq 0$, which is equivalent to

$$\left(\sum_{l=1}^{n} \sigma_l \varepsilon_l\right) \cdot \left(\sum_{k=1}^{n} \frac{\sigma_k \pi_k^2}{\varepsilon_k}\right) \leq p. \tag{A.6}$$

Let us decompose the vector $\boldsymbol{\pi}$ into two orthogonal parts such that $\boldsymbol{\pi} = \boldsymbol{\pi}_{\leq} + \boldsymbol{\pi}_{>}$, where $\boldsymbol{\pi}_{\leq}$ corresponds to components π_k such that $\pi_k \leq \gamma \sigma_k$, and $\boldsymbol{\pi}_{>}$ to the remaining components. Decomposing $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ similarly, we have $\boldsymbol{\varepsilon} = \boldsymbol{\pi}_{\leq} + \gamma \boldsymbol{\sigma}_{>}$. The following are straightforward

$$1 = \|\boldsymbol{\pi}_{\leq}\|_2^2 + \|\boldsymbol{\pi}_{>}\|_2^2 \tag{A.7}$$

$$\|\boldsymbol{\varepsilon}\|_2^2 = \|\boldsymbol{\pi}_{\leq}\|_2^2 + \gamma^2 \|\boldsymbol{\sigma}_{>}\|_2^2 \tag{A.8}$$

$$\boldsymbol{\varepsilon}^{\mathsf{T}} \cdot \boldsymbol{\sigma} = \boldsymbol{\pi}_{\leq}^{\mathsf{T}} \cdot \boldsymbol{\sigma}_{\leq} + \gamma \|\boldsymbol{\sigma}_{>}\|_2^2. \tag{A.9}$$

Using these equalities, we obtain

$$\sum_{k=1}^{n} \frac{\sigma_k \pi_k^2}{\varepsilon_k} = \boldsymbol{\pi}_{\leq}^{\mathsf{T}} \cdot \boldsymbol{\sigma}_{\leq} + \frac{1}{\gamma} \|\boldsymbol{\pi}_{>}\|_2^2 \tag{A.10}$$

$$= \boldsymbol{\pi}_{\leq}^{\mathsf{T}} \cdot \boldsymbol{\sigma}_{\leq} + \frac{1}{\gamma}\left(1 - \|\boldsymbol{\pi}_{\leq}\|_2^2\right) \tag{A.11}$$

$$= \boldsymbol{\varepsilon}^{\mathsf{T}} \cdot \boldsymbol{\sigma} + \frac{1}{\gamma}\left(1 - \|\boldsymbol{\varepsilon}\|_2^2\right). \tag{A.12}$$

Therefore, the left hand side of Eq. (A.6) can be written as

$$\left(\sum_{l=1}^{n}\sigma_l\varepsilon_l\right)\cdot\left(\sum_{k=1}^{n}\frac{\sigma_k\pi_k^2}{\varepsilon_k}\right) = \left(\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}\right)^2 + \frac{\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}}{\gamma}\left(1-\|\boldsymbol{\varepsilon}\|_2^2\right) \tag{A.13}$$

$$= p\|\boldsymbol{\varepsilon}\|_2^2 + \frac{\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}}{\gamma}\left(1-\|\boldsymbol{\varepsilon}\|_2^2\right) \tag{A.14}$$

$$= \left(p-\frac{\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}}{\gamma}\right)\|\boldsymbol{\varepsilon}\|_2^2 + \frac{\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}}{\gamma}, \tag{A.15}$$

where we have used Eq. (A.2). Since $\varepsilon_k \leq \gamma\sigma_k$, we have $\|\boldsymbol{\varepsilon}\|_2^2 \leq \gamma\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}$, which, together with Eq. (A.2) implies that $\frac{\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}}{\gamma} \leq p$. Together with $\|\boldsymbol{\varepsilon}\|_2^2 \leq \|\boldsymbol{\pi}\|_2^2 = 1$, this implies

$$\left(p-\frac{\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}}{\gamma}\right)\|\boldsymbol{\varepsilon}\|_2^2 + \frac{\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}}{\gamma} \leq p, \tag{A.16}$$

which proves Eq. (A.6) and, in turn, $\mu \geq 0$.

We now show that $\lambda_k \geq 0$ for all $k \in [n]$. From Eqs. (A.4) and (A.5) we see that this is equivalent to showing

$$\frac{\sigma_k}{\varepsilon_k}\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma} - p \geq -\frac{p-\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}\sum_{k=1}^{n}\frac{\sigma_k\pi_k^2}{\varepsilon_k}}{1-\|\boldsymbol{\varepsilon}\|_2^2} \tag{A.17}$$

Note that $1 \geq \|\boldsymbol{\varepsilon}\|_2^2$. By multiplying out everything with $1-\|\boldsymbol{\varepsilon}\|_2^2$ and expanding, we get

$$\frac{\sigma_k}{\varepsilon_k}\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}\left(1-\|\boldsymbol{\varepsilon}\|_2^2\right) + p\|\boldsymbol{\varepsilon}\|_2^2 \geq \boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}\sum_{k=1}^{n}\frac{\sigma_k\pi_k^2}{\varepsilon_k}. \tag{A.18}$$

Note that $p\|\boldsymbol{\varepsilon}\|_2^2 = (\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma})^2$, so after rearranging terms and dividing by $\boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma}$ we get

$$\frac{\sigma_k}{\varepsilon_k}\left(1-\|\boldsymbol{\varepsilon}\|_2^2\right) + \boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma} \geq \sum_{k=1}^{n}\frac{\sigma_k\pi_k^2}{\varepsilon_k}. \tag{A.19}$$

We apply Eq. (A.12) to the right hand side and get

$$\frac{\sigma_k}{\varepsilon_k}\left(1-\|\boldsymbol{\varepsilon}\|_2^2\right) + \boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma} \geq \boldsymbol{\varepsilon}^\mathsf{T}\cdot\boldsymbol{\sigma} + \frac{1}{\gamma}\left(1-\|\boldsymbol{\varepsilon}\|_2^2\right). \tag{A.20}$$

After simplification this yields $\varepsilon_k \leq \gamma\sigma_k$, which is true by definition of $\boldsymbol{\varepsilon}$. Thus, we have $\lambda_k \geq 0$.

Finally, it remains to show that the following matrix is positive semidefinite:

$$\Lambda - I + \mu(pI - \boldsymbol{\sigma} \cdot \boldsymbol{\sigma}^{\mathsf{T}}) = \mu \left[\left(\sum_{l=1}^{n} \sigma_l \varepsilon_l \right) \cdot \operatorname{diag}(\sigma_k / \varepsilon_k) - \boldsymbol{\sigma} \cdot \boldsymbol{\sigma}^{\mathsf{T}} \right]. \tag{A.21}$$

Since $\mu \geq 0$, it is the case if and only if

$$\forall \boldsymbol{v} \in \mathbb{R}^n : \quad \left(\sum_{l=1}^{n} \sigma_l \varepsilon_l \right) \cdot \left(\sum_{k=1}^{n} \frac{v_k^2 \sigma_k}{\varepsilon_k} \right) \geq \left(\sum_{k=1}^{n} v_k \sigma_k \right)^2. \tag{A.22}$$

This follows by Cauchy–Schwarz inequality:

$$\left(\sum_{l=1}^{n} \sigma_l \varepsilon_l \right) \cdot \left(\sum_{k=1}^{n} v_k^2 \sigma_k / \varepsilon_k \right) \geq \left(\sum_{k=1}^{n} \sqrt{\sigma_k \varepsilon_k} \cdot \sqrt{v_k^2 \sigma_k / \varepsilon_k} \right)^2 \tag{A.23}$$

which completes the proof. □

References

[AA05] Scott Aaronson and Andris Ambainis. Quantum search of spatial regions. *Theory of Computing*, 1:47–79, 2005. doi:10.4086/toc.2005.v001a004. 44, 47, 48, 51

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity (CCC'09)*, pages 229–242. IEEE Computer Society, 2009. doi:10.1109/CCC.2009.42. 70

[ABN+11] Andris Ambainis, Arturs Backurs, Nikolajs Nahimovs, Raitis Ozols, and Alexander Rivosh. Search by quantum walks on two-dimensional grid without amplitude amplification. 2011. arXiv:1112.3337. 45

[AD10] Scott Aaronson and Andrew Drucker. A full characterization of quantum advice. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC'10)*, pages 131–140. ACM, 2010. arXiv:1004.0377, doi:10.1145/1806689.1806710. 70

[AKR05] A. Ambainis, J. Kempe, and A. Rivosh. Coins make quantum walks faster. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*, pages 1099–1108. SIAM, 2005. 44, 45, 46

[Amb00] Andris Ambainis. Quantum lower bounds by quantum arguments. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC'00)*, pages 636–643. ACM, 2000. arXiv:quant-ph/0002066, doi:10.1145/335305.335394. 80, 106

[Amb04] Andris Ambainis. Quantum walk algorithm for element distinctness. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS'04)*, pages 22–31. IEEE Computer Society Press, 2004. 44, 45, 46, 50

[Amb10] Andris Ambainis. Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. 2010. arXiv:1010.4458. 106

[AMRR11] Andris Ambainis, Loïck Magnin, Martin Roetteler, and Jérémie Roland. Symmetry-assisted adversaries for quantum state generation. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity (CCC'11)*, pages 167–177. IEEE Computer Society, 2011. arXiv:1012.2112, doi:10.1109/CCC.2011.24. 70, 71, 75, 76, 77, 79, 81

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in NP ∩ coNP. *Journal of the ACM*, 52(5):749–765, 2005. (Earlier version in FOCS'04). doi:10.1145/1089023.1089025. 70

[ATS03] Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC'03)*, pages 20–29. ACM, 2003. arXiv:quant-ph/0301023, doi:10.1145/780542.780546. 70

[BACS05] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2):9, 2005. arXiv:quant-ph/0508139, doi:10.1007/s00220-006-0150-x. 96

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997. arXiv:quant-ph/9701001, doi:10.1137/S0097539796300933. 85, 113

[BBHT98] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, 1998. doi:10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P. 42, 90, 92, 104

[BCvD05] Dave Bacon, Andrew M. Childs, and Wim van Dam. From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 469–478, Oct 2005. arXiv:quant-ph/0504083, doi:10.1109/SFCS.2005.38. 127

[BCvD06] Dave Bacon, Andrew M. Childs, and Wim van Dam. Optimal measurements for the dihedral hidden subgroup problem. *Chicago Journal of Theoretical Computer Science*, 2006(2), Oct 2006. URL: http://cjtcs.cs.uchicago.edu/articles/2006/2/contents.html, arXiv:quant-ph/0501044. 127

[BCWdW01] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16):167902, Sep 2001. arXiv:quant-ph/0102001, doi:10.1103/PhysRevLett.87.167902. 70, 142

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X. 76

[Bea97] Robert Beals. Quantum computation of Fourier transforms over symmetric groups. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 48–53, New York, NY, USA, 1997. ACM. doi:10.1145/258533.258548. 110

[BHMT00] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. 2000. arXiv:quant-ph/0005055. 4, 87, 89, 90, 91, 92, 94, 104, 105, 137, 138, 145

[BKS09] Sergio Boixo, Emanuel Knill, and Rolando D. Somma. Eigenpath traversal by phase randomization. *Quantum Information and Computation*, 9(9,10):833–855, 2009. arXiv:0903.1652. 70

[BL95] Dan Boneh and Richard Lipton. Quantum cryptanalysis of hidden linear functions. In *Advances in Cryptology CRYPT0 95*, volume 963 of *Lecture Notes in Computer Science*, pages 424–437. Springer, 1995. doi:10.1007/3-540-44750-4_34. 110

[BŠ06] H. Buhrman and R. Špalek. Quantum verification of matrix products. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms*, pages 880–889, 2006. 44

[BV97] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. Conference version in Proc. STOC'93, pp. 11–20. 113, 130

[CDF+02] Andrew M. Childs, Enrico Deotto, Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Andrew J. Landahl. Quantum search by measurement. *Phys. Rev. A*, 66:032314, Sep 2002. arXiv:quant-ph/0204013, doi:10.1103/PhysRevA.66.032314. 41

[CEMM98] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998. arXiv:quant-ph/9708016, doi:10.1098/rspa.1998.0164. 4, 56, 96, 99, 100

[CG04a] Andrew M. Childs and Jeffrey Goldstone. Spatial search and the Dirac equation. *Phys. Rev. A*, 70(4):042312, 2004. doi:10.1103/PhysRevA.70.042312. 44, 45

[CG04b] Andrew M. Childs and Jeffrey Goldstone. Spatial search by quantum walk. *Phys. Rev. A*, 70(2):022314, 2004. doi:10.1103/PhysRevA.70.022314. 45

[Chi08] Andrew M. Childs. On the relationship between continuous- and discrete-time quantum walk. *Communications in Mathematical Physics*, 294(2):22, 2008. arXiv:0810.0312, doi:10.1007/s00220-009-0930-1. 72, 96

[CJS10] Andrew M. Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. 2010. arXiv:1012.4019. 110

[CK11] Andrew M. Childs and Robin Kothari. Simulating sparse Hamiltonians with star decompositions. In *Theory of Quantum Computation, Communication, and Cryptography (TQC 2010)*, volume 6519 of *Lecture Notes in Computer Science*, page 11, Berlin, Heidelberg, 2011. Springer. arXiv:1003.3683, doi:10.1007/978-3-642-18073-6. 96

[CS09] Thomas W. Cusick and Pantelimon Stănică. *Cryptographic Boolean Functions and Applications*. Academic Press/Elsevier, 2009. URL: http://books.google.ca/books?id=OAkhkLSxxxMC&pg=PA73. 120

[CvD07] Andrew M. Childs and Wim van Dam. Quantum algorithm for a generalized hidden shift problem. In *Proceedings of the eighteenth annual*

ACM-SIAM symposium on Discrete algorithms, SODA '07, pages 1225–1232, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1283383.1283515, arXiv:quant-ph/0507190. 113

[CvD10] Andrew M. Childs and Wim van Dam. Quantum algorithms for algebraic problems. *Rev. Mod. Phys.*, 82:1–52, Jan 2010. arXiv:0812.0380, doi:10.1103/RevModPhys.82.1. 109, 111

[CW07] Andrew M. Childs and Pawel Wocjan. On the quantum hardness of solving isomorphism problems as nonabelian hidden shift problems. *Quantum Information and Computation*, 7(5):504–521, Jul 2007. URL: http://www.rintonpress.com/journals/qiconline.html#v7n56, arXiv:quant-ph/0510185. 111

[Dev86] Luc Devroye. *Non-uniform random variate generation*. Springer, New York, 1986. 71

[Dil72] John F. Dillon. A survey of bent functions. *The NSA technical journal*, pages 191–215, 1972. 119, 120

[Dil75] John F. Dillon. Elementary Hadamard difference sets. In *Proceedings of the 6th S-E Conference on Combinatorics, Graph Theory, and Computing*, pages 237–249. Winnipeg Utilitas Math., 1975. 119, 120

[DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992. doi:10.1098/rspa.1992.0167. 110

[Dob95] Hans Dobbertin. Construction of bent functions and balanced Boolean functions with high nonlinearity. In *Fast Software Encryption*, volume 1008 of *Lecture Notes in Computer Science*, pages 61–74, 1995. doi:10.1007/3-540-60590-8_5. 119, 120

[dW08] Ronald de Wolf. A brief introduction to Fourier analysis on the Boolean cube. *Theory of Computing Library – Graduate Surveys*, 1:1–20, 2008. doi:10.4086/toc.gs.2008.001. 115

[EH99] Mark Ettinger and Peter Høyer. A quantum observable for the graph isomorphism problem. 1999. arXiv:quant-ph/9901029. 110

[EH00] Mark Ettinger and Peter Høyer. On quantum algorithms for noncommutative hidden subgroups. *Advances in Applied Mathematics*, 25(3):239–251, 2000. arXiv:quant-ph/9807029, doi:10.1006/aama.2000.0699. 111

[FGGS00] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. 2000. arXiv:quant-ph/0001106. 10, 29

[FGH+12] Edward Farhi, David Gosset, Avinatan Hassidim, Andrew Lutomirski, and Peter Shor. Quantum money from knots. In *Proceedings of the 3rd Innovations in Theoretical Computer Science (ITCS'12) conference*, 2012. To appear. arXiv:1004.5127. 70

[FIM+02] Katalin Friedl, Gábor Ivanyos, Frédéric Magniez, Miklos Santha, and Pranab Sen. Hidden translation and orbit coset in quantum computing. In *Proceedings of the 35fth Annual ACM Symposium on Theory of Computing (STOC'03)*, pages 1–9. ACM, 2002. arXiv:quant-ph/0211091, doi:10.1145/780542.780544. 70, 111

[FRFU94] U. Feige, P. Raghavan, D. Feleg, and E. Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994. 68

[Gha11] Mirmojtaba Gharibi. The non-injective hidden shift problem. Master's thesis, University of Waterloo, Canada, 2011. URL: http://hdl.handle.net/10012/6478, arXiv:1207.4537. 112, 114, 127

[GR02] Lov K. Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. 2002. arXiv:quant-ph/0208112. 70

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC'96)*, pages 212–219, New York, 1996. ACM. arXiv:quant-ph/9605043, doi:10.1145/237814.237866. 105, 107, 113, 127

[Gro00] Lov K. Grover. Synthesis of quantum superpositions by quantum computation. *Physical Review Letters*, 85(6):1334–1337, 2000. doi:10.1103/PhysRevLett.85.1334. 72, 87

[GRR11] Dmitry Gavinsky, Martin Roetteler, and Jérémie Roland. Quantum algorithm for the boolean hidden shift problem. In *Computing and Combinatorics*, volume 6842 of *Lecture Notes in Computer Science*, pages 158–167. Springer Berlin / Heidelberg, 2011. arXiv:1103.3017, doi:10.1007/978-3-642-22685-4_14. 3, 112, 114, 138, 139

[GS97] C.M. Grinstead and J.L. Snell. *Introduction to Probability*. 2nd ed. American Mathematical Society, 1997. URL: http://books.google.ca/books?id=14oq4uWGCkwC. 7, 9, 10

[Hal07] Sean Hallgren. Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. *J. ACM*, 54(1):4:1–4:19, Mar 2007. doi:10.1145/1206035.1206039. 110

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009. arXiv:0811.3171, doi:10.1103/PhysRevLett.103.150502. 72, 75, 96, 97, 98, 99, 106

[HJ90] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990. URL: http://books.google.ca/books?id=PlYQN0ypTwEC. 8

[HLŠ07] Peter Høyer, Troy Lee, and Robert Špalek. Negative weights make adversaries stronger. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC'07)*, pages 526–535. ACM, 2007. arXiv:quant-ph/0611054, doi:10.1145/1250790.1250867. 75, 80, 81, 106

[HMdW03] Peter Høyer, Michele Mosca, and Ronald de Wolf. Quantum search on bounded-error inputs. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP'03)*, volume 2719 of *Lecture Notes in Computer Science*, pages 291–299. Springer, 2003. arXiv:quant-ph/0304052, doi:10.1007/3-540-45061-0_25. 67, 137

[Hø97] Peter Høyer. Efficient quantum transforms. 1997. arXiv:quant-ph/9702028. 110

[HW94] Paul Hausladen and William K. Wootters. A pretty good measurement for distinguishing quantum states. *Journal of Modern Optics*, 41(12):2385–2390, 1994. doi:10.1080/09500349414552221. 127

[Iva08] Gábor Ivanyos. On solving systems of random linear disequations. *Quantum Information and Computation*, 8(6&7):579–594, 2008. URL: http://www.rintonpress.com/journals/qiconline.html#v8n67, arXiv:0704.2988. 73, 111

[Joz98] Richard Jozsa. Quantum algorithms and the Fourier transform. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):323–337, 1998. arXiv:quant-ph/9707033, doi:10.1098/rspa.1998.0163. 110

[Joz01] Richard Jozsa. Quantum factoring, discrete logarithms, and the hidden subgroup problem. *Computing in Science Engineering*, 3(2):34–43, Mar/Apr 2001. arXiv:quant-ph/0012084, doi:10.1109/5992.909000. 110

[Joz03] Richard Jozsa. Quantum computation in algebraic number theory: Hallgren's efficient quantum algorithm for solving Pell's equation. *Annals of Physics*, 306(2):241–279, 2003. arXiv:quant-ph/0302134, doi:10.1016/S0003-4916(03)00067-8. 110

[JRS07] S. Jansen, M.-B. Ruskai, and R. Seiler. Bounds for the adiabatic approximation with applications to quantum computation. *J. Math. Phys.*, 48(10):102111, 2007. 37, 41

[KB06] Hari Krovi and Todd A. Brun. Hitting time for quantum walks on the hypercube. *Phys. Rev. A*, 73(3):032341, 2006. doi:10.1103/PhysRevA.73.032341. 44

[Kem05] Julia Kempe. Discrete quantum walks hit exponentially faster. *Prob. Th. Rel. Fields*, 133(2):215–235, 2005. 44

[Kit95] Alexei Kitaev. Quantum measurements and the Abelian Stabilizer Problem. 1995. arXiv:quant-ph/9511026. 4, 56, 96, 99, 100, 110

[KKR06] Julia Kempe, Alexei Kitaev, and Oded Regev. The complexity of the local Hamiltonian problem. *SIAM Journal on Computing*, 35(5):1070–1097, 2006. arXiv:quant-ph/0406180, doi:10.1137/S0097539704445226. 70

[KLM07] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An introduction to quantum computing*. Oxford University Press, 2007. URL: http://books.google.ca/books?id=gLFQAAAAMAAJ. 4, 109

[KMOR10] Hari Krovi, Frédéric Magniez, Maris Ozols, and Jérémie Roland. Finding is as easy as detecting for quantum walks. In *Automata, Languages and Programming*, volume 6198 of *Lecture Notes in Computer Science*, pages 540–551. Springer, 2010. arXiv:1002.2419, doi:10.1007/978-3-642-14165-2_46. 1, 2

[KOR10] Hari Krovi, Maris Ozols, and Jérémie Roland. Adiabatic condition and the quantum hitting time of Markov chains. *Phys. Rev. A*, 82:022333, Aug 2010. arXiv:1004.2721, doi:10.1103/PhysRevA.82.022333. 1, 2

[Kos08] Thomas Koshy. *Catalan Numbers with Applications*. Oxford scholarship online: Mathematics module. Oxford University Press, 2008. URL: http://books.google.ca/books?id=MqPLSivdBDAC. 131

[KS60] J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Undergraduate Texts in Mathematics. Springer-Verlag, 1960. URL: http://books.google.ca/books?id=0bTK5uWzbYwC. 7, 9, 10

[KS07] L.B. Koralov and Y.G. Sinai. *Theory of Probability and Random Processes*. Universitext (1979). Springer, 2007. URL: http://books.google.ca/books?id=tlWOphOFRgwC. 7, 8

[KST93] J. Köbler, U. Schöning, and J. Toran. *The Graph Isomorphism Problem: Its Structural Complexity*. Progress in Theoretical Computer Science. Birkhäuser Boston, 1993. 70

[KSV02] Alexei Yu. Kitaev, Alexander Shen, and Mikhail N. Vyalyi. *Classical and Quantum Computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, 2002. URL: http://books.google.ca/books?id=qYHTvHPvmG8C. 4, 109, 110

[Kup05] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005. arXiv:quant-ph/0302112, doi:10.1137/S0097539703436345. 110

[Kup11] Greg Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. 2011. arXiv:1112.3333. 110

[KW08] Alexei Kitaev and William A. Webb. Wavefunction preparation and resampling using a quantum computer. 2008. arXiv:0801.0342. 73

[Let75] Gérard Letac. On building random variables of a given distribution. *The Annals of Probability*, 3(2):298–306, 1975. doi:10.1214/aop/1176996400. 71

[LMR+11] Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Špalek, and Mario Szegedy. Quantum query complexity of state conversion. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS'11)*, 2011. To appear. 70, 72, 77, 79, 81, 106

[Lom04] Chris Lomont. The hidden subgroup problem - review and open problems. 2004. arXiv:quant-ph/0411037. 109

[ME99] Michele Mosca and Artur Ekert. The hidden subgroup problem and eigenvalue estimation on a quantum computer. In *Quantum Computing and Quantum Communications*, volume 1509 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 1999. arXiv:quant-ph/9903071, doi:10.1007/3-540-49208-9_15. 110

[Mes59] Albert Messiah. *Mécanique Quantique*. Dunod, Paris, 1959. 29, 37

[Mey00] C.D. Meyer. *Matrix Analysis and Applied Linear Algebra*. Number v. 1 in Miscellaneous Titles in Applied Mathematics Series. Society for Industrial and Applied Mathematics, 2000. URL: http://books.google.ca/books?id=Zg4M0iFlbGcC. 8

[MN07] Frédéric Magniez and Ashwin Nayak. Quantum complexity of testing group commutativity. *Algorithmica*, 48:221–232, 2007. arXiv:quant-ph/0506265, doi:10.1007/s00453-007-0057-8. 44

[MNRS07] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via quantum walk. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC'07)*, pages 575–584. ACM Press, 2007. doi:10.1145/1250790.1250874. 41, 44, 45, 46

[MNRS12] Frédéric Magniez, Ashwin Nayak, Peter Richter, and Miklos Santha. On the hitting times of quantum versus random walks. *Algorithmica*, 63:91–116, 2012. doi:10.1007/s00453-011-9521-6. 41, 44, 45, 46, 51, 57

[MR05] Cristopher Moore and Alexander Russell. For distinguishing conjugate hidden subgroups, the pretty good measurement is as good as it gets. 2005. arXiv:quant-ph/0501177. 127

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087, 1953. doi:10.1063/1.1699114. 71, 99

[MRRS07] Cristopher Moore, Daniel Rockmore, Alexander Russell, and Leonard J. Schulman. The power of strong Fourier sampling: Quantum algorithms for affine groups and hidden shifts. *SIAM J. Comput.*, 37(3):938–958, Jun 2007. arXiv:quant-ph/0503095, doi:10.1137/S0097539705447177. 111

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The theory of error–correcting codes*. North–Holland, Amsterdam, 1977. 119, 120

[MSS05] F. Magniez, M. Santha, and M. Szegedy. Quantum algorithms for the triangle problem. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*, pages 1109–1117. SIAM, 2005. 44, 50

[MW05] Chris Marriott and John Watrous. Quantum Arthur–Merlin games. *Computational Complexity*, 14(2):122–152, 2005. arXiv:cs/0506068, doi:10.1007/s00037-005-0194-x. 99

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010. URL: http://books.google.ca/books?id=-s4DEy7o-a0C. 4, 109

[NWZ09] Daniel Nagaj, Pawel Wocjan, and Yong Zhang. Fast amplification of QMA. *Quantum Information and Computation*, 9(11&12):1053–1068, 2009. URL: http://www.rintonpress.com/journals/qiconline.html#v9n1112, arXiv:0904.1549. 35, 99

[ORR12] Maris Ozols, Martin Roetteler, and Jérémie Roland. Quantum rejection sampling. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 290–308, New York, NY, USA, 2012. ACM. arXiv:1103.2774, doi:10.1145/2090236.2090261. 1, 2

[Reg04a] Oded Regev. Quantum computation and lattice problems. *SIAM Journal on Computing*, 33(3):738–760, 2004. arXiv:cs/0304005, doi:10.1137/S0097539703440678. 70, 110, 111

[Reg04b] Oded Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. 2004. arXiv:quant-ph/0406151. 110

[Rei09] Ben W Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS'09)*, pages 544–551. IEEE Computer Society Press, 2009. arXiv:0904.2759, doi:10.1109/FOCS.2009.55. 81, 106

[Rei11] Ben W. Reichardt. Reflections for quantum query algorithms. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'11)*, pages 560–569, 2011. arXiv:1005.1601. 81, 106

[Röt09] Martin Rötteler. Quantum algorithms to solve the hidden shift problem for quadratics and for functions of large Gowers norm. In *Proceedings of the 34st International Symposium on Mathematical Foundations of Computer Science (MFCS'09)*, volume 5734 of *Lecture Notes in Computer Science*, pages 663–674. Springer, 2009. arXiv:0911.4724, doi:10.1007/978-3-642-03816-7_56. 112, 114

[Röt10] Martin Rötteler. Quantum algorithms for highly non-linear Boolean functions. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, pages 448–457, 2010. arXiv:0811.3208. 3, 73, 105, 107, 112, 113, 130, 138

[SBBK08] Rolando D. Somma, Sergio Boixo Boixo, Howard Barnum, and Emanuel Knill. Quantum simulations of classical annealing processes. *Physical Review Letters*, 101:130504, 2008. arXiv:0804.1571, doi:10.1103/PhysRevLett.101.130504. 70

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. Preliminary version in FOCS 1994. arXiv:quant-ph/9508027, doi:10.1137/S0097539795293172. 110

[Sim94] Daniel R. Simon. On the power of quantum computation. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 116–123, Nov 1994. doi:10.1109/SFCS.1994.365701. 3, 110, 127

[SKW03] Neil Shenvi, Julia Kempe, and Birgitta K. Whaley. Quantum random-walk search algorithm. *Phys. Rev. A*, 67:052307, May 2003. arXiv:quant-ph/0210064, doi:10.1103/PhysRevA.67.052307. 44

[SMM09] Lana Sheridan, Dmitri Maslov, and Michele Mosca. Approximating fractional time quantum evolution. *Journal of Physics A*, 42(18):185302, 2009. arXiv:0810.3843, doi:10.1088/1751-8113/42/18/185302. 72

[SO10] Rolando D. Somma and Gerardo Ortiz. Quantum approach to classical thermodynamics and optimisation. In *Quantum Quenching, Annealing and Computation*, volume 802 of *Lecture Notes in Physics*, pages 1–20, Heidelberg, 2010. Springer. 30, 31, 32, 35

[STV11] Martin Schwarz, Kristan Temme, and Frank Verstraete. Contracting tensor networks and preparing PEPS on a quantum computer. 2011. arXiv:1104.1410. 99

[Sze04a] Mario Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS'04)*, pages 32–41. IEEE Computer Society Press, 2004. 13, 30, 31, 32, 35, 41, 44, 45, 46, 51, 52

[Sze04b] Mario Szegedy. Spectra of quantized walks and a $\sqrt{\delta\varepsilon}$-rule. 2004. arXiv:quant-ph/0401053. 13

[TOV+11] K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, and F. Verstraete. Quantum Metropolis sampling. *Nature*, 471(7336):87–90, 2011. arXiv:0911.3635, doi:10.1038/nature09770. 70, 73, 75, 99, 100, 101, 102, 104, 106

[Tul08] Avatar Tulsi. Faster quantum-walk algorithm for the two-dimensional spatial search. *Phys. Rev. A*, 78(1):012310, 2008. arXiv:0801.0497, doi:10.1103/PhysRevA.78.012310. 44, 45, 47, 51, 57

[Vaz98] Umesh Vazirani. On the power of quantum computation. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 356(1743):1759–1768, 1998. URL: http://www.jstor.org/stable/55010. 85

[VB96] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38(1):49, 1996. 159

[vD08] Wim van Dam. Quantum algorithms for weighing matrices and quadratic residues. *Algorithmica*, 34(4):413–428, 2008. arXiv:quant-ph/0008059, doi:10.1007/s00453-002-0975-4. 113

[vDHI03] Wim van Dam, Sean Hallgren, and Lawrence Ip. Quantum algorithms for some hidden shift problems. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*, pages 489–498, 2003. arXiv:quant-ph/0211140. 73, 111

[VKB08] Martin Varbanov, Hari Krovi, and Todd A. Brun. Hitting time for the continuous quantum walk. *Phys. Rev. A*, 78(2):022324, Aug 2008. doi:10.1103/PhysRevA.78.022324. 44

[vN51] John von Neumann. Various techniques used in connection with random digits. *National Bureau of Standards, Applied Math Series*, 12:36–38, 1951. 3, 71

[Wat00] John Watrous. Succinct quantum proofs for properties of finite groups. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS'00)*, pages 537–546. IEEE Computer Society, 2000. arXiv:cs/0009002, doi:10.1109/SFCS.2000.892141. 70, 142

[Wat01] John Watrous. Quantum algorithms for solvable groups. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC'01)*, pages 60–67. ACM, 2001. arXiv:quant-ph/0011023, doi:10.1145/380752.380759. 70

[YAG10] Man-Hong Yung and Alán Aspuru-Guzik. A quantum-quantum Metropolis algorithm. 2010. arXiv:1011.1468. 70, 106