Jakub Mareček and Georgios Korpas and
Johannes Aspman

Quantum Computing

via Randomized Algorithms

November 10, 2023

Springer

Preface

Synergies between physics and computer science have been some of the most dominant scientific and technological disciplines in recent times that aided in significant technological advances. Quantum computing is a growing field at the intersection of physics and computer science that is projected to lead to the next computational revolution based on the the theoretical and recently proven computational discovery that computers based on quantum mechanical architecture are exponentially powerful. Combining the existing expertise in both fields proves to be a nontrivial but very exciting interdisciplinary adventure that will benefit students in diverse ways.

This course aims to make this cutting-edge discipline broadly accessible to undergraduate students with a background in computer science as well as mathematics and physics. The course will introduce the students to some of the most fundamental concepts in the field, both from a theoretical point of view, so the students obtain a deep physical understanding of the underlying principles, as well as a practical one such as to be able to apply their newly acquired skills with quantum simulators or by accessing actual quantum devices on the cloud. This course provides an interdisciplinary first introduction to the emerging field of quantum computation building up from the basics of quantum mechanics to quantum computational complexity and quantum algorithms. During the course, special care is given to stress the potential quantum speedups of quantum computers against their classical counterparts.

Jakub Mareček

Acknowledgements

Use the template *acknow.tex* together with the Springer document class SVMono (monograph-type books) or SVMult (edited books) if you prefer to set your acknowledgement section as a separate chapter instead of including it as last part of your preface.

Contents

Part II Beyond the Basics

Part III Applications

Part I

The Fundamentals

[most]tcolorbox

Chapter 1

Quantum Physics 101

Quantum mechanics was developed in the beginning of the last century as a means to explain certain mystical phenomena observed in experiments involving atoms and light. This led to a revolution in physics and, more broadly, in how we look upon the nature of reality.

Nowadays, even though quantum physics still might sound mysterious and abstract, it is very much a vital part of our daily lives through its many applications in modern technologies.

1900 Max Planck solves the black body radiation problem by assuming that light comes in discrete levels of wavelengths. This gives birth to quantum mechanics.

1905 Einstein explains the photoelectric effect by using Plancks quanta of light.

1913 Bohr proposes a model of the atom based on electrons with quantized angular momentum.

TBC. Should also be some experiments and so on...

Classical physics[1] is completely deterministic. It is in theory possible to know everything about a classical system, and furthermore, once we know enough about the system, we can determine everything about its future through the basic laws of classical physics such as Newton mechanics and the theory of relativity.

One of the mysterious, or some would even say disturbing, facts about quantum mechanics is that this is no longer true. Quantum mechanics is inherently a non-deterministic, or probabilistic, theory.

In this Chapter we will give a lightning introduction to the wonderful world of quantum mechanics, with of course a special eye towards the applications into computer

[1] In this course, when we talk about classical physics we simply mean *not* quantum physics.

science. The mathematical language of quantum mechanics is mainly that of linear algebra, and much of the material will therefore be a review of concepts that you learned in linear algebra, but perhaps with a slightly different notation and language than you are familiar with.

We will also discuss the probabilistic nature of quantum mechanics and how this affects results of measurements; how quantum systems evolve with time; the quantum harmonic oscillator; and finally, we will discuss the quantum analogue of the classical bit of computer science.

1.1 Quantum states

1.1.1 States, probability and measurements in a classical world

Imagine throwing an ordinary die, or flipping a coin. The resulting outcomes will be either $\{1,2,3,4,5,6\}$ or $\{\text{Heads},\ \text{Tails}\}$, respectively. We refer to this as saying that the *state* of the die or coin is in the value of the outcome, say 5 or Heads. Obviously, it does not make sense to say that for example the coin is in a mixture of heads and tails.[2] It simply is in either the state heads or the state tails. We can summarize this by saying that, *in classical physics, a state takes values in a set.*

Furthermore, it is obvious to us that making a measurement, i.e. looking at the die or coin after it has landed, will not affect the system. If we throw the die and immediately cover it with our hand before seeing the outcome, it will still be in the state it lands on, say six, before we remove the hand, and continue to be in the state six if we cover it again. We could even imagine doing something more complicated, say that we first look only at the number on top of the die (six) then cover it then look only at the number on the side facing us, say four, then cover it again. If we now look at the number on the top we of course still assume[3] that this will still be six. This can be phrased as saying that *in classical physics measurements does not affect the system.*

Later on, we will discuss the probabilistic nature of quantum mechanics, but the notion of probability is of course something we occasionally use when describing systems in the classical world as well. After all, playing board games would perhaps be a bit less fun if we always knew exactly how the dice would land. However, this notion of probability is simply a measure of how little information we have about the system. If we had some super computer that could completely characterize the

[2] If we are very unlucky, it could of course happen that the coin manages to balance on its edge in the end, but then we would simply enlarge the set of values it can take to account for this.

[3] and correctly so,

initial state of the dice in the throwers hand, the force and angles of the hand that throw the dice, the atmospheric pressure and wind speed in the room when the dice are thrown, and so on, it could determine exactly how the dice would land. In classical physics, knowing everything about a system really means knowing everything. Using the laws of classical physics (and given a powerful enough computer) we can completely determine the future of any system once we know enough data. Or in other words, *classical physics is deterministic.*

Let us summarize what we have learned about classical physics so far:

- Classical states are elements of a set.
- Measurements does not affect the classical system.
- Classical physics is deterministic.

All of this hopefully seems rather obvious and intuitive to you and you might wonder why we are discussing such basic facts. Well, as we will see, when we step in to the quantum world, these basic things will no longer hold true and our daily life intuition about the world around us can more or less be thrown out the window.

1.1.2 Quantum states

One of the main differences between classical and quantum physics is the fact that quantum states are not just elements of a set, they are vectors in a complex-valued vector space. The strange thing is that we can give some meaning to the statement that a quantum state is in a mixture of states. If we had a quantum coin it could of course be either in the states heads or tails, but it could also be in a mixture of the two. This is called *superposition* and is one of the most fundamental concepts in quantum mechanics.

To see how this works, we first introduce some notation. We imagine that we have a system that is in some state, which we simply label by the letter ψ. This could in principle be anything we want, it is just a label for us to distinguish the state from another. For example, it could be a number corresponding to one of the classical states $\{1,2,3,4,5,6\}$ of a die, but it could also be something else, such as $\uparrow$ or $\downarrow$. The state vector is then denoted as

$$|\psi\rangle.$$

This is called a *ket vector*, or simply a *ket*.[4] The ψ is just a label that we pick for our state while the encasing $|\cdot\rangle$ is there to remind us that this is a vector. Now, superposition tells us that it could happen that the physical system is in a combination of two (or more) states, e.g., we could have something like

$$|\psi\rangle = \alpha|\psi_1\rangle + \beta|\psi_2\rangle,$$

for some states $|\psi_1\rangle$, $|\psi_2\rangle$, and some (complex) numbers α and β. The numbers α and β are usually called the probability amplitude of the states $|\psi_1\rangle$ and $|\psi_2\rangle$, respectively.[5] Due to this possibility, we directly see that we will have many more possibilities than in the classical system.

The ket vectors satisfy the ordinary axioms of a vector space. There are two operations, vector addition and scalar multiplication. Under vector addition, the vector space is closed, associative and commutative. This means that for three vectors in the space $|a\rangle$, $|b\rangle$, $|c\rangle$, we have

$$\begin{aligned}
&|a\rangle + |b\rangle = |c\rangle, \quad \text{(closed)},\\
&(|a\rangle + |b\rangle) + |c\rangle = |a\rangle + (|b\rangle + |c\rangle), \quad \text{(associative)},\\
&|a\rangle + |b\rangle = |b\rangle + |a\rangle, \quad \text{(commutative)}.
\end{aligned}$$

There is a unique identity element of vector addition, which we denote simply by 0, such that

$$|\psi\rangle + 0 = |\psi\rangle.$$

The reason why we do not use $|0\rangle$ here is because we want to reserve that notation for something completely different, as we will see later on. There is also a unique vector $(-|\psi\rangle)$ such that

$$|\psi\rangle + (-|\psi\rangle) = 0.$$

The vector space is linear and distributive under scalar multiplication. This means that for some complex numbers $z, z_1, z_2 \in \mathbb{C}$,

$$|(z_1 + z_2)\psi\rangle = z_1|\psi\rangle + z_2|\psi\rangle, \quad z(|\psi\rangle + |\varphi\rangle) = z|\psi\rangle + z|\varphi\rangle.$$

Finally, there also exists an identity element with respect to scalar multiplication, i.e., we can multiply with the number 1 and get back the same state, $1|\psi\rangle = |\psi\rangle$.

A basis of a vector space, $\{|a_1\rangle, \ldots, |a_d\rangle\}$, is a minimal set of vectors that spans the space, the number of basis vectors needed, here d, gives the dimension of the vector space. A generic state $|\psi\rangle$ in this vector space can then be expressed as a superposition of such basis vectors,

---

[4] The notation here (together with the bra vector that we will introduce shortly, is usually called either the *bra-ket* notation or the Dirac notation, after the physicist Paul Dirac who invented it.

[5] This will be discussed in more detail later on, but it is important to note that the probability amplitude is not the same as a probability. For one thing, it is a complex number.

$$|\psi\rangle = \sum_{j=1}^{d} \psi_j |a_j\rangle.$$

1.1.3 The dual space and inner product

There is also a corresponding dual vector space. The elements of this space are denoted

$$\langle\varphi|,$$

and are called *bra vectors*. The notation and their names becomes slightly more sensical when we introduce the inner product between the bra and the ket, or a bra(c)ket,[6]

$$\langle\varphi|\psi\rangle.$$

This is simply a complex number. When we have a finite-dimensional vector space together with an inner product this defines what is called a Hilbert space.[7] Two vectors are said to be orthogonal if their inner product is zero. Furthermore, it is customary to normalize quantum states such that the inner product with itself is equal to one, such vectors are called unit vectors. We will do this automatically, or in other words, we will always set

$$\langle\psi|\psi\rangle = 1.$$

Vectors that are both normalized and orthogonal are then called orthonormal. This is, for example, a very good property to demand of a set of basis vectors. The normalization of quantum states will also play a vital role when we later discuss probabilities in quantum mechanics.

Is is often useful to represent the kets as column vectors and the bras as row vectors. We then have the relation

$$|\psi\rangle = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_d \end{pmatrix} \longleftrightarrow \langle\psi| = (\psi_1^* \ldots, \psi_d^*),$$

[6] Remember that quantum physics was invented long before the invention of the meme, so this was perhaps at the time considered funny. Dirac was also a famously strange man, The strangest man.

[7] When the vector space is infinite-dimensional, some extra subtleties arise, but we will mostly be dealing with finite-dimensional vector spaces in this course, and we therefore ignore these subtleties for now.

and the inner-product (or the bracket) then simply becomes the ordinary multiplication of vectors. We further see that the elements of the corresponding vectors are related by complex conjugation and we have

$$\langle \psi | \psi \rangle = |\psi_1|^2 + \cdots + |\psi_d|^2.$$

Since the inner product between two states is just a complex number, we can ask what its complex conjugate is. This is given by

$$(\langle \varphi | \psi \rangle)^* = \langle \psi | \varphi \rangle,$$

and thus

$$|\langle \varphi | \psi \rangle|^2 = \langle \varphi | \psi \rangle \langle \psi | \varphi \rangle.$$

1.1.4 Composite systems

If we imagine that we have several quantum systems, each in some state represented by some state vector, we can combine the separate system into a larger system using the tensor product of vector spaces, $\otimes$. If we imagine that we have one system where the state is given by $|\psi\rangle$ and another where the state is given by $|\varphi\rangle$, the state of the composite system is given by

$$|\psi\rangle \otimes |\varphi\rangle.$$

States that can be written in this simple way are called *product states*. We will discuss that in more detail later on when we introduce the concept of entanglement. This way we can build complicated systems by combining several smaller systems. We will see this in action when we discuss quantum circuits. Note that the tensor product does not commute in general.

[width= colback=gray!50,title=Summary quantum states, colbacktitle=gray!20,coltitle=black]

- Quantum states are vectors in a complex vector space.
- A state is represented by the ket $|\psi\rangle$.
- The elements of the dual space are called bras and denoted $\langle \varphi |$.
- The inner product, or bracket, $\langle \varphi | \psi \rangle$, is a complex number, and its complex conjugate is given by $(\langle \varphi | \psi \rangle)^* = \langle \psi | \varphi \rangle$.
- We normalize the states such that $\langle \psi | \psi \rangle = 1$.
- Quantum states can be in a superposition of states, $|\psi\rangle = \alpha |\psi_1\rangle + \beta |\psi_2\rangle$, for some complex numbers α, β.

- More generally, we can express any quantum state in a vector space as a superposition of the basis vectors of that vector space, $|\psi\rangle = \sum_{j=1}^{d} \psi_j |a_j\rangle$, for some complex numbers ψ_j and basis vectors $|a_j\rangle$.

1.2 Measurements and probability

1.2.1 Observables

We have discussed how a quantum state is described by a state vector in a vector space. The quantum state is however not something that we can measure directly.[8] In fact, it only tells us something about the *probability* of finding some result upon performing a measurement. Note that this is in stark contrast to the classical case where the state and the outcome of a measurement is for all intents and purposes equal to each other.

We refer to the properties of a state that we can measure as *observables*. If we consider a system representing a particle in some particular state, the observables would correspond to specific properties of this particle, such as its position, its velocity or its angular momentum. Observables are described in quantum mechanics by linear operators acting on the vector space of states. We thus say that a linear operator A acts on the state $|\psi\rangle$, and denote it by

$$A|\psi\rangle.$$

The corresponding action on the bra is given by the *Hermitian conjugate* (sometimes called the adjoint) of A, which we denote by a small dagger

$$A|\psi\rangle \longleftrightarrow \langle\psi|A^\dagger.$$

Note that the operator acts on the bra from the right and on the ket from the left.

When we represent the bras and kets as vectors the operators are naturally represented by matrices. The action of the dagger is then given by complex conjugation of the elements together with transposition of the matrix. For example,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^\dagger = \begin{pmatrix} a^* & c^* \\ b^* & d^* \end{pmatrix}.$$

We can construct linear operators through the *outer product*

[8] The actual meaning of the quantum state is something that has spurred a long history of heated discussions. We will discuss some interpretations in the following sections.

$$A = |\varphi_1\rangle\langle\varphi_2|.$$

Acting with such an operator on a state $|\psi\rangle$ gives

$$A|\psi\rangle = (|\varphi_1\rangle\langle\varphi_2|)|\psi\rangle = \langle\varphi_2|\psi\rangle|\varphi_1\rangle.$$

So it transforms the state $|\psi\rangle$ into the state $|\varphi_1\rangle$ multiplied by the complex number $\langle\varphi_2|\psi\rangle$.

A very important and useful identity can be derived by considering a complete orthonormal basis $\{|v_j\rangle\}$ and expressing $|\psi\rangle = \sum_j \psi_j|v_j\rangle$, then introduce the operator $A = \sum_j |v_j\rangle\langle v_j|$. Here both sums are over the complete set of basis states. We notice that

$$A|\psi\rangle = \left(\sum_j |v_j\rangle\langle v_j|\right)|\psi\rangle = \sum_j |v_j\rangle\langle v_j|\psi\rangle = \sum_j\sum_k \psi_k|v_j\rangle\langle v_j|v_k\rangle = \sum_j \psi_j|v_j\rangle = |\psi\rangle,$$

which implies that $\sum_j |v_j\rangle\langle v_j| = \mathbb{1}$, the identity operator on the vector space. This relation is called a completeness relation, or sometimes a resolution of identity, and can be a very useful trick in many computations and proofs in quantum mechanics.

For any observable, say A, there exists a particular set of vectors called the eigenvectors, $|a_j\rangle$. They are defined through the relation

$$A|a_j\rangle = a_j|a_j\rangle,$$

where a_j is a complex number called the eigenvalue corresponding to the eigenvector $|a_j\rangle$ of A. We will typically use the above notation where the eigenvalues and eigenvectors have the same symbol, i.e., the eigenvalue of the eigenvector $|a_j\rangle$ is given by a_j. This is standard, and hopefully does not introduce too much confusion.

An especially important class of operators is the class of *Hermitian operators*. They are defined by having the property $A^\dagger = A$. One of the key consequences of this for Hermitian operators is that their eigenvalues are all real. Physical observables in quantum mechanics are always given by Hermitian operators. The reason, as we will see later, is that the result of a measurement in quantum mechanics is given by the eigenvalues of the observable we are measuring. But the results of any physical measurement should of course be a real number, so that we should impose that the observables are Hermitian operators. Another important property of Hermitian operators is that their eigenvectors form a complete set, i.e., any state can be expressed in the eigenvectors. Note however, that if the eigenvalues are the same the eigenvectors need not be orthogonal.

An operator A is called *normal* if it satisfies $A^\dagger A = AA^\dagger$. Such operators satisfy an important theorem called the spectral decomposition theorem. It states that an operator is normal if and only if it is diagonalizable with respect to some basis. This

means that we can always express a normal operator, A, as $A = \sum_j a_j |j\rangle\langle j|$, where a_j are the eigenvalues of A and $|j\rangle$ an orthonormal basis where each vector is also an eigenvector of A (with eigenvalue a_j). Obviously, Hermitian operators are always normal.

A third class of operators that will play a very important role in this course is the class of *unitary operators*. They are defined by the property $A^{-1} = A^\dagger$, or in words, that the Hermitian conjugate is equal to the inverse operator. This means that $A^\dagger A = \mathbb{1}$.

Suppose now that we have two different observables A and B and we want to know if we can express them both in terms of the same basis. Or in other words, if we can write $A = \sum_j a_j |j\rangle\langle j|$ and $B = \sum b_j |j\rangle\langle j|$. If this is possible we say that A and B are *simultaneously diagonalizable*. It turns out that this can only be done if A and B commute with each other, that is, if and only if

$$[A,B] := AB - BA = 0.$$

The notation $[A,B]$ is called the *commutator* of A and B and is a very frequently used operation in quantum mechanics.

We can again use the tensor product to build larger systems. If we have a system that is a composite system of say two different vector spaces

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle,$$

we can build composite operators acting on this tensor product as

$$A = A_1 \otimes A_2, \quad A|\psi\rangle = A_1|\psi_1\rangle \otimes A_2|\psi_2\rangle.$$

1.2.2 The wave function

Consider now a complete set of commuting observables, $A, B, C, \ldots$ together with an orthonormal basis $|a,b,c,\ldots\rangle$, where $a, b, c, \ldots$ are the corresponding eigenvalues of the observables. An arbitrary state $|\psi\rangle$ can then be expanded in this basis as

$$|\psi\rangle = \sum_{a,b,c,\ldots} \psi(a,b,c,\ldots)|a,b,c,\ldots\rangle.$$

The set of coefficients,

$$\{\psi(a,b,c,\ldots) = \langle a,b,c,\ldots|\psi\rangle\},$$

is called the *wave function* of the system in the $a,b,c,\dots$ basis. As we have mentioned before, the individual coefficients $\psi(a,b,c,\dots)$ are also called the *probability amplitude* for finding the system in the $|a,b,c,\dots\rangle$ state, or sometimes just the amplitude. It is important to note that this is not the same as the probability as we will see next. For one thing, it is in general a complex number. The actual probability of finding the eigenvalues corresponding to $|a,b,c,\dots\rangle$, is instead given by the absolute value squared of $\psi(a,b,c,\dots)$.

1.2.3 Measurements

The idea of a measurement in quantum mechanics is that we measure some observable A and the outcome will be an eigenvalue of A, where the corresponding probability of getting this result is captured by the coefficient of the state when expanded in the eigenvectors of the measured observable.

In other words, we start with an observable A that we want to measure for some system $|\psi\rangle$. We express it in its complete basis of eigenvectors $A = \sum_j a_j |a_j\rangle$. We further expand our system in this basis as $|\psi\rangle = \sum_j \psi_j |a_j\rangle$. The measurement will then return an eigenvalue of A, let us say a_j, and the probability of finding this specific result is given by $|\psi_j|^2$. Remember that we always normalize the states such that

$$1 = \langle\psi|\psi\rangle = \sum_j |\psi_j|^2,$$

so this interpretation as a probability makes sense.

After the measurement, the system has "collapsed" to the state $|a_j\rangle$ and we can measure A again to find the same result, a_j.[9]

At this point you might be wondering what all the fuss is about. We said in the beginning of this chapter that quantum mechanics is supposed to undermine our classical intuition that measurements does not affect the system. But, now we are saying that if we measure an observable A and find that the system is in, say, the state $|a_1\rangle$, then making another measurement asking if the system is in state $|a_1\rangle$ will give a positive answer with probability one. Is this not exactly what we said about the experiment with throwing a die and covering it? Can we not just say that the system was in state $|a_1\rangle$ all along?

Well, the tricky thing with quantum mechanics is that if we now measure another observable that is not commuting with A, say B, and find the result $|b_1\rangle$, and then

[9] The word "collapsed" here is a standard one used in much literature, but is definitely the subject of much debate. What exactly happens in the moment of measurement is at the core of the various interpretations of quantum mechanics that have appeared over the years, as we will briefly discuss later on.

afterwards return to measure A again, it is no longer true that we are certain to find the result a_1. We are basically back at square one and the only thing we can say is that there is a probability $|\langle b_1|a_1\rangle|^2$ to find the result a_1. This would be like throwing the die, looking at the number on top, then looking at one of the numbers on the side and then finally looking at the number on top again to find that it is no longer the same.

With the above interpretations we can define the expectation value of an observable A in the state $|\psi\rangle$ in the ordinary way. This is denoted $\langle A\rangle_\psi$ and defined by

$$\langle A\rangle_\psi := \langle\psi|A|\psi\rangle = \sum_j a_j |\langle\psi|a_j\rangle|^2,$$

where $|a_j\rangle$ is the complete set of eigenvectors of A.

Let us consider a simple example, namely that of a two-level system. This means that we have a two-dimensional vector space.[10] We introduce an orthonormal basis

$$\{|u\rangle, |d\rangle\},$$

such that we can express any state as

$$|\psi\rangle = \alpha|u\rangle + \beta|d\rangle, \quad |\alpha|^2 + |\beta|^2 = 1.$$

Next, we introduce an observable σ_z defined by

$$\sigma_z|u\rangle = |u\rangle, \quad \sigma_z|d\rangle = -|d\rangle.$$

I.e., the basis vectors are eigenvectors of σ_z with eigenvalues ± 1, respectively.

We now measure σ_z and get some result. Let us assume that this is $+1$, and the state collapses to $|u\rangle$. As said before, we can now measure σ_z again and again and every time we will get the result $+1$.

But, there is of course nothing special with the basis defined by $|u\rangle$ and $|d\rangle$, we could as easily pick another basis. For example,[11]

$$|l\rangle := \frac{1}{\sqrt{2}}(|u\rangle + |d\rangle), \quad |r\rangle := \frac{1}{\sqrt{2}}(|u\rangle - |d\rangle).$$

Related to this basis we can introduce a new observable, σ_x, that has these vectors as eigenvectors,

$$\sigma_x|l\rangle = |l\rangle, \quad \sigma_x|r\rangle = -|r\rangle,$$

[10] This kind of system will of course be the main protagonist of this course, since the qubit is a two-level system. But for now we simply think of it in slightly more abstract terms.

[11] as an exercise you can show that this is an orthonormal set of vectors,

and which does not commute with σ_z.[12] If we now measure σ_x in our system, which has collapsed to $|u\rangle$ after the first measurement, we will get the results ± 1 with probabilities

$$|\langle u|l\rangle|^2 = \frac{1}{2}|\langle u|(|u\rangle + |d\rangle)|^2 = \frac{1}{2},$$
$$|\langle u|r\rangle|^2 = \frac{1}{2}|\langle u|(|u\rangle - |d\rangle)|^2 = \frac{1}{2}.$$

Let us again assume that the result is $+1$ such that the state collapses to $|l\rangle$. Now you might start to see the problem. If we return to measure σ_z, we will no longer find $+1$ with probability one but instead we have

$$|\langle l|u\rangle|^2 = \frac{1}{2},$$
$$|\langle l|d\rangle|^2 = \frac{1}{2}.$$

The two outcomes are now equally probable. This is part of the mysterious and indeterministic nature of quantum mechanics. It is, perhaps, easy to see that, if the observables do commute we can measure them simultaneously. Since we can then diagonalize them in the same basis.

The uncertainty in measuring non-commuting observables is captured by the famous *Heisenberg's uncertainty principle*. This principle is so fundamental and important in quantum mechanics, so let us take a few lines to derive it.

When we talk about uncertainty in this setting we typically mean with respect to the standard deviation, ΔA, for some observable A. This is defined by the following equation,

$$(\Delta A)^2_\psi \coloneqq \sum (a_j - \langle A\rangle)^2_\psi |\langle \psi|a_j\rangle|^2.$$

We may simplify things and assume that the expectation value of A is zero, which implies that we have the simpler form

$$(\Delta A)^2_\psi = \langle \psi|A^2|\psi\rangle.$$

Let us now consider two observables A and B. The Cauchy-Schwartz inequality,

$$2|X||Y| \geq |\langle X|Y\rangle + \langle Y|X\rangle|,$$

applied to the combinations $|X\rangle = A|\psi\rangle$ and $|Y\rangle = iB|\psi\rangle$ gives us [13]

$$\Delta A \Delta B \geq \frac{1}{2}|\langle \psi|[A,B]|\psi\rangle|.$$

[12] This follows from the definitions.

[13] here we are assuming that the expectation values of both A and B are zero, as an exercise you can fill in the details of the derivation,

This is the uncertainty principle in its general form. In words it simply says that the product of the uncertainties in the two observables A and B, can not be smaller than the expectation value of the commutator of A and B. This is exactly what we mentioned earlier, namely that if two observables does not commute, then we can not measure them with certainty at the same time.

This is an enormously important consequence of the laws of quantum mechanics, and it has many important consequences of its own. It is for example believed to be the reason why there are galaxies and planets in the universe as well as part of the leading explanation to why the universe is expanding with an accelerating speed.

Let us finally note two important facts about quantum states. Firstly, if we have two states that only differ by an overall phase, say, $|\psi\rangle$ and $|\varphi\rangle = e^{i\gamma}|\psi\rangle$, then the statistical properties of these states are the same. This is easily seen from the fact that we have

$$|\varphi\rangle = e^{i\gamma}|\psi\rangle \implies \langle\varphi| = \langle\psi|e^{-i\gamma},$$

and we thus have

$$\langle\varphi|\varphi\rangle = \langle\psi|e^{-i\gamma}e^{i\gamma}|\psi\rangle = \langle\psi|\psi\rangle = \sum_j |\psi_j|^2.$$

For this reason, in quantum mechanics, we do not distinguish between states that differ only by an overall phase.

Secondly, we can notice is that we can only distinguish two states with complete certainty if they are orthogonal, otherwise they will have some component along the same direction and the result of the measurement has some probability of being the same for the two states.

[width= colback=gray!50,title=Summary observables and measurements, colbacktitle=gray!20,coltitle=black]

- Observables in quantum mechanics are represented as linear operators acting on the state space. They act on a ket from the left, $A|\psi\rangle$, and on a bra from the right $\langle\psi|A^\dagger$, where † denotes the Hermitian conjugate.
- Normal operators are defined by having $A^\dagger A = AA^\dagger$, and such operators satisfy the spectral decomposition theorem.
- Unitary operators are defined by having $A^\dagger = A^{-1}$.
- Hermitian operators are defined by having $A^\dagger = A$. They are normal operators and their eigenvalues are all real.
- Physical observables are described by Hermitian operators.
- The commutator between two operators A and B is denoted $[A,B] = AB - BA$. Two observables can only be simultaneously diagonalizable if they commute, i.e. if $[A,B] = 0$.

- Measurements "collapses" the quantum state into an eigenstate of the measured observable.
- Heisenberg's uncertainty principle states that we can not know two properties of a quantum system simultaneously, unless their respective operators commute.

1.3 Evolution

1.3.1 Unitary operators

An interesting question to ask at this point might be how a quantum system evolves in time? To answer this, we first consider a system that at some time t is in the state $|\psi(t)\rangle$. We then ask how this is related to the state at some other time, say $t = 0$? We encode this change in an operator that we call $U(t)$ such that we have

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle.$$

Now, to be able to say something more about this mysterious operator $U(t)$ we want to introduce some restrictions. First of all, we want to demand that it is linear. This is natural from what we have discussed before. Quantum operators are typically linear. Less trivial is the statement that we want to enforce the operator to preserve distinguishability. This means that, if we have two orthogonal states, such that they are distinguishable by a measurement, we want them to still be orthogonal after the time evolution. Furthermore, we want the probabilities to be preserved, i.e., the normalization should remain intact.

Let us see what consequences this has. If we pick two elements $|a_j\rangle$ and $|a_k\rangle$ of an orthonormal basis to represent the states at $t = 0$, we have the condition

$$\langle a_j|a_k\rangle = \delta_{jk},$$

where δ_{jk} is the Kronecker symbol.[14] But if we now let them evolve in time using $U(t)$ we want to have

$$\langle a_j|U^\dagger(t)U(t)|a_k\rangle = \delta_{jk},$$

and we see that $U^\dagger(t)U(t)$ acts as the unit operator. From this you can prove that the same is true for the action on any states. We thus need the time evolution operator to satisfy $U^\dagger(t)U(t) = \mathbb{1}$. This is exactly what we mentioned earlier as the definition of a unitary operator. So, time evolution in quantum mechanics is described by a unitary operator.

[14] $\delta_{jk} = 1$ if $j = k$ and 0 if $j \neq k$.

1.3.2 The Schrödinger equation

A slightly different view of the evolution of a system is due to thinking about a very important operator in quantum mechanics, namely the Hamilton operator, or sometimes just the Hamiltonian. This is the observable corresponding to the energy of the system.[15] The Hamiltonian determines the evolution of the system through the Schrödinger equation[16]

$$i\frac{\mathrm{d}|\psi\rangle}{\mathrm{d}t} = H|\psi\rangle. \tag{1.1}$$

More specifically, this is called the time-dependent Schrödinger equation. Here, $\hbar$ (pronounced *h-bar*) is the famous Planck's constant, $\sim 1.0546 \times 10^{-34} kg\,m^2/s$.[17] As you see it is a very small constant, and this is basically the reason why quantum physics is not part of our daily intuitions of the behaviour of things.

The Hamiltonian is Hermitian and we can expand it in its complete set of eigenvectors,

$$H = \sum_j E_j |E_j\rangle\langle E_j|.$$

These eigenstates are called the energy eigenstates and the corresponding eigenvalues are the results of a measurement of the energy of the system. Since the $|E_j\rangle$ are eigenstates of the Hamiltonian we have

$$H|E_j\rangle = E_j|E_j\rangle,$$

which is sometimes called the time-independent Schrödinger equation.

By solving Schrödinger's equation, we find the connection to the operator $U(t)$ discussed earlier. Namely,

$$U(t) = e^{-\frac{i}{\hbar}Ht}. \tag{1.2}$$

We thus see that we can consider two different pictures, one where the state itself changes with time. The change is governed by the Schrödinger equation (1.1) and the corresponding picture is aptly called the *Schrödinger picture*. In the other picture we can instead think of the states as being constant while the time dependence is all due to the operators, as in Eq. (1.2). This picture is called the *Heisenberg picture*.

---

[15] The Hamiltonian, named after William Rowan Hamilton, as a quantity describing the energy of a system is of course also important in classical physics. In classical mechanics it is however not an operator but an ordinary function.

[16] Named after its inventor, the cat-friendly Austrian Erwin Schrödinger.

[17] The German physicist Max Planck was the person who, sort of by mistake, started the whole field of quantum physics. He introduced a constant which he called h, which was later divided by 2π to give the constant $\hbar := \frac{h}{2\pi}$, which we now call Planck's constant.

1.3.3 A note on (in)determinism

As we have mentioned already in the introduction, and seen in the discussion of measurements, quantum mechanics is inherently non-deterministic. But the discussion of time evolution of the quantum state looks very deterministic, right? This is true. The time evolution of the quantum state is a deterministic process, but this does not necessarily mean that quantum mechanics is deterministic.

In classical physics, making measurements does not affect the system and the result of a measurement is equivalent to the state of the system, both before and after the measurement. This is the basis of the determinism in classical physics. By knowing the state and knowing the equations of motion, we can determine where the state came from and where it is going. As we have seen, this is no longer true in quantum physics. Time evolution of the quantum state is deterministic, but knowing the state does not tell you with certainty the result of a general measurement.

[width= colback=gray!50,title=Summary: Quantum postulates, colbacktitle=gray!20,coltitle=black]
Let us summarize what we have learned so far into four postulates of quantum mechanics.

1. States are described by unit vectors in a complex vector space (in fact a Hilbert space), and observables are described by linear Hermitian operators.

2. The possible outcomes of a measurement are given by the eigenvalues of the operator corresponding to the observable being measured.

3. If the system is in a state $|\psi\rangle$, and we measure an observable A with eigenvectors $|a_j\rangle$ and eigenvalues a_j, the probability of measuring eigenvalue a_j is given by

$$P(a_j) = |\langle a_j|\psi\rangle|^2 = \langle\psi|a_j\rangle\langle a_j|\psi\rangle.$$

4. The evolution of a quantum system is described by unitary operators.

1.4 What actually is the quantum state?

At this point, you might be asking yourself what the meaning of the quantum state is. After all, measurements tells us that eventually the state will not be in a superposition, the thing we observe is a definitive classical state, so how do we know that the state was ever in a superposition of other states? Well, such questions have given rise to a large number of debates on the interpretation of quantum mechanics. [18]

[18] See for example Wikipedia

Many early interpretations of quantum mechanics involved hidden variables, i.e., that there are some hidden variables that we do not know about which determines the measurements in a deterministic fashion. These have been essentially refuted by a number of results, such as Gleason's theorem and variations thereof. These type of results typically go under the name of Bell's theorem, and in principle they rule out almost all hidden variables theories. The experimental verification of these results was the subject of the Nobel prize in physics 2022.[19]

In contrast, the widely considered Copenhagen interpretation of quantum mechanics (with variations) is essentially Bayesian. In this interpretation, the nature of quantum mechanics is essentially non-deterministic, and we should not require one to consider the "exponential" dimension of the quantum state prior to measurement any more so than we require a person throwing a die to consider the probability distribution over the outcomes. Measurements give rise to a (practically) irreversible process in which the state is affected.

Another famous interpretation is the many-worlds interpretation due to Hugh Everett. Here, time is considered as a tree, having many branches and each branch corresponds to a possible result of a measurement. This gives rise to an uncountable number of worlds or universes. The many-worlds interpretation is thus inherently deterministic, as the universal wave function never collapses to one particular state.

Finally, let us mention the de Broglie-Bohm interpretation. This is a kind of hidden variables theory where the problems of Bell's theorem are circumvented by embracing non-locality. Locality is basically the concept that only things near to each other can affect each other. This is one of the main building blocks of Einstein's theory of special relativity. The de Broglie-Bohm interpretation is thus a deterministic theory and particles have a definite configuration at all times, even when not observed. This has gained some interest in recent years and researcher are currently working on how to align it with the ideas of special relativity.

1.5 The qubit

Let us now introduce the main protagonist of the course, the qubit. In a classical computer we use classical bits that are systems whose states takes values in the set $\{0,1\}$. The corresponding quantum system is called a *qubit* (sometimes QBit, q-bit or quantum bit). This system is described by a two-dimensional complex vector space. To make the connection to classical bits even stronger we denote a set of basis vectors in this state space as

$$\{|0\rangle, |1\rangle\}.$$

[19] Awarded to the three experimentalists Alain Aspect, John Clauser and Anton Zeilinger.

Note that, as was mentioned before, we use the notation $|0\rangle$ to denote a basis vector, not the zero vector. This basis is typically referred to as the *computational basis*. Another frequently appearing basis is given by the states

$$|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle).$$

You may recognize these bases as the u,d and l,r basis we studied earlier. The $|\pm\rangle$ basis is sometimes called the Hadamard basis. Any qubit can be expanded in either of these bases,

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \alpha_+|+\rangle + \alpha_-|-\rangle,$$

for some numbers α_j, with the extra conditions $|\alpha_0|^2 + |\alpha_1|^2 = |\alpha_+|^2 + |\alpha_-|^2 = 1$.

We will often represent the computational basis by the vectors

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Linear operators acting on a qubit will now be described by 2×2 complex matrices. Of special importance are the so called *Pauli operators*.[20] These are a set of three matrices that together with the identity matrix spans the vector space of 2×2 Hermitian matrices. In the computational basis, they read

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

As an easy, but informative and extremely useful, exercise, we can study how the Pauli operators act on our basis states. The Pauli matrices are some of the most used operation in quantum circuits, and these kinds of actions on the basis states will be used many many times throughout the course. We find[21]

$$\sigma_x \begin{Bmatrix} |0\rangle \\ |1\rangle \\ |+\rangle \\ |-\rangle \end{Bmatrix} = \begin{Bmatrix} |1\rangle \\ |0\rangle \\ |+\rangle \\ -|-\rangle \end{Bmatrix}, \quad \sigma_y \begin{Bmatrix} |0\rangle \\ |1\rangle \\ |+\rangle \\ |-\rangle \end{Bmatrix} = \begin{Bmatrix} i|1\rangle \\ -i|0\rangle \\ -i|-\rangle \\ i|+\rangle \end{Bmatrix}, \quad \sigma_z \begin{Bmatrix} |0\rangle \\ |1\rangle \\ |+\rangle \\ |-\rangle \end{Bmatrix} = \begin{Bmatrix} |0\rangle \\ -|1\rangle \\ |-\rangle \\ |+\rangle \end{Bmatrix}.$$

When discussing quantum gates, the σ_x operator is sometimes referred to as the NOT gate, since it interchanges $|0\rangle$ and $|1\rangle$.

[20] Named after the Austrian physicist Wolfgang Pauli, who is counted as one of the main inventors of quantum mechanics.

[21] perhaps you recognize some of these properties from when we studied the up/down/left/right system earlier,

1.5.1 The Bloch sphere

We know that we can express any qubit as a superposition of the two basis vectors $|0\rangle$, $|1\rangle$, and that the corresponding coefficients must satisfy $|\alpha_0|^2 + |\alpha_1|^2 = 1$. We can then use a little trigonometry to express any qubit as

$$|\psi\rangle = e^{i\gamma}\left(\cos\tfrac{\theta}{2}|0\rangle + e^{i\phi}\sin\tfrac{\theta}{2}|1\rangle\right).$$

Where γ, ϕ and θ are some real numbers. However, we also saw earlier that we can not distinguish states that only differ by an overall phase, so we can disregard the overall phase factor $e^{i\gamma}$. We can thus describe any qubit in terms of two real numbers ϕ and θ through the identification

$$|\psi\rangle = \cos\tfrac{\theta}{2}|0\rangle + e^{i\phi}\sin\tfrac{\theta}{2}|1\rangle.$$

This is simply the spherical coordinates of the unit sphere, and we have thus found that any qubit can be represented by a point on the unit sphere. This representation of the state space of a qubit as a sphere goes under the name of the *Bloch sphere*. See Figure 3.1 for an example of how we can visualize the state $|+\rangle$ on the Bloch sphere. Here we clearly see the difference between a classical bit and a qubit. A classical bit can only take the values 1 or 2 while the qubit can in principle be in any state that correspond to a point on the Bloch sphere, i.e., we have an continuum of possible states.

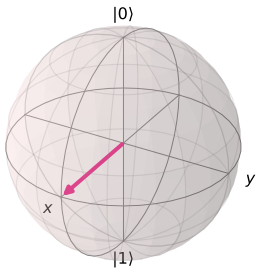


Fig. 1.1: The Bloch sphere. The vector denotes the qubit state $|\psi\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The labels x and y represent the Euclidean x and y directions.

It can be easily seen from the previous calculations that the Pauli matrices act as rotations along the different axes of the Bloch sphere. For example, acting with σ_x on $|0\rangle$ rotates the state 180°, or π radians, around the x-axis to give the state $|1\rangle$, and so on. All the standard one qubit gates can be visualized in a similar manner as their action on the Bloch sphere.

1.5.2 Several qubits

Just as before, we can combine simple systems into larger ones by using the tensor product of vector spaces. This will be vital when constructing quantum circuits, since, obviously, having just one qubit would perhaps not be all that exciting.

So, using the tensor product we can build larger systems of several qubits, for example

$$|0\rangle \otimes |0\rangle \otimes |+\rangle \otimes |1\rangle \otimes \cdots \otimes |1\rangle.$$

We will often be lazy and use the notation

$$|\psi_{n-1} \dots \psi_0\rangle \coloneqq |\psi_{n-1}\rangle \otimes |\psi_{n-2}\rangle \otimes \cdots \otimes |\psi_0\rangle.$$

For example, for the two-qubit system, given by a four-dimensional vector space, we then have the basis vectors

$$|00\rangle = |0\rangle \otimes |0\rangle, \quad |01\rangle = |0\rangle \otimes |1\rangle, \quad |10\rangle = |1\rangle \otimes |0\rangle, \quad |11\rangle = |1\rangle \otimes |1\rangle.$$

It is easy to show that these span the vector space of states. Sometimes a further simplification of notation is used for these types of combined systems where we imagine the product to indicate a binary representation of an integer, so that we write for example $|01\rangle = |1\rangle_2$ and $|11\rangle = |3\rangle_2$ and so on, where the subscript indicates how many qubits there are in the system. The subscript is of course needed because 001 and 1 are both binary representations of the number 1, while here the former would be a three qubit system and the latter a one qubit system.

1.5.3 Physical implementations

There are several physical implementations of a qubit, including:

- Superconducting qubits: These qubits are made from tiny loops of superconducting wire, which can carry electrical current without resistance. The state of a superconducting qubit can be controlled by applying electromagnetic pulses to the loop.

- Trapped-ion qubits: These qubits are made by trapping a single ion (an electrically charged atom) in a magnetic or electric field. The state of a trapped-ion qubit can be controlled by shining laser light on the ion.

- Topological qubits: These qubits are based on the properties of certain materials, such as topological insulators, that can carry electrical current on their surface while insulating inside.

- Quantum dots: These qubits are made by confining a single electron or hole (an absence of an electron) in a tiny semiconductor structure called a quantum dot.

- Nuclear Magnetic Resonance (NMR) qubits: These qubits are based on the spin of the nuclei of certain atoms.

- Photonic qubits: These qubits are based on the properties of individual photons (particles of light). For example, the polarization state of a photon can be used as a qubit, with the two possible states being horizontal and vertical polarization.

- Single-molecule spin qubits: These qubits are based on the spin of individual electrons or nuclei in a single molecule. The state of the qubit can be controlled by applying magnetic fields to the molecule. These qubits are still in the research stage and not yet commercialized.

1.6 The harmonic oscillator

Models of most physical implementations of qubits rely on various variations of the quantum harmonic oscillator. To give some intuition behind the physical qubits, as well as to illustrate the many concepts we have introduced so far, we will show how to extend the classical harmonic oscillator to the quantum harmonic oscillator, highlighting the differences.[22]

1.6.1 The classical harmonic oscillator

Classical systems follow Newton's three laws of mechanics. In particular, the second law states that the force is equal to the mass times the acceleration,

$$F = ma.$$

[22] The harmonic oscillator could very well be the single most important system in all of physics, so a basic knowledge of this system is probably a good thing to have in life.

A harmonic oscillator is a particle that undergoes harmonic motion around an equilibrium point. Think for example of a spring with a mass attached to its end such that it bounces back and forth around an equilibrium.

Let us focus on the one-dimensional case and set the equilibrium point to be $x = 0$. The system is described by a mass m and a restoring force that pushes the mass towards the equilibrium point,

$$F = -m\omega^2 x,$$

where ω is called the angular frequency. The minus sign tells us that the force is driving the spring back towards its equilibrium point. Combining this with Newton's second law we get

$$ma = m\ddot{x} = -m\omega^2 x.$$

The solution of this second order differential equation is

$$x(t) = A\cos(\omega t + \phi),$$

where A is the amplitude of the oscillations (giving the turning points of the motion) and ϕ the initial phase.

SOME PICTURES COULD DEFINITELY BE ADDED IN THE ABOVE!

The potential energy of the system is given by

$$V = \frac{1}{2}m\omega^2 x^2.$$

This gives a parabola as shown in Figure 1.2. The reason why the harmonic oscillator is so important as a physical system is that almost any smooth function can be approximated by a parabola near its minimum points.

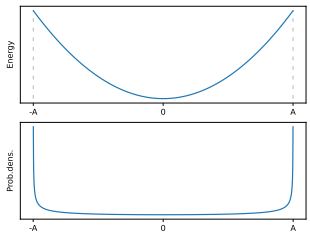


Fig. 1.2: The potential energy (top) and probability density (bottom) of the classical harmonic oscillator, with amplitude A.

Remember that the total energy of the system is given by the sum of the potential energy, V, and the kinetic energy $\frac{1}{2}mv^2$. At the turning points $x = \pm A$, the velocity, and therefore the kinetic energy, is zero, and the potential energy reaches its maximum. The total energy of the system thus simply says something about how far away from the equilibrium it can move. For example, the zero-energy harmonic oscillator simply sits still at its equilibrium. At the equilibrium point, on the other hand, the kinetic energy reaches its maximum and the potential energy is zero, this means that the particle attains the greatest velocity here. This further implies that for a classical harmonic oscillator, the probability is highest to find it close to the turning points $x = \pm A$, since this is where it moves at its slowest, and thus spends the most time. This is shown in the bottom picture of Fig. 1.2.

1.6.2 The quantum harmonic oscillator

The quantum harmonic oscillator is, as the name suggests, the quantum analogue of the classical system. As we discussed earlier, in quantum mechanics (and also in classical mechanics) an important role is played by the Hamiltonian of the system. This is simply constructed as the sum of the kinetic and potential energy. So to construct the Hamiltonian we simply take the expression for the classical kinetic and potential energy and sum them,

$$H = \frac{1}{2}mv^2 + \frac{1}{2}m\omega^2 x^2.$$

But in quantum mechanics, as we have seen, observables should be operators, so we also promote the position and momentum ($p = mv$) variables to operators.[23] This results in the expression

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2}m\omega^2\hat{x}^2,$$

where we, in this section only, adopted the very common practice of putting hats on quantum operators, to distinguish them from their classical variable analogues. Note that, in contrast to most of the rest of this course, we are here considering an infinite-dimensional Hilbert space of states, since both $\hat{x}$ and $\hat{p}$ take continuous values. This does introduce some extra subtleties that we however simply gloss over at the moment.

In quantum mechanics, the energy of the system is described by the time-independent Schrödinger equation

$$\hat{H}|\psi_E\rangle = E|\psi_E\rangle,$$

[23] There are many reasons why we use momentum instead of velocity as the go-to operator in quantum mechanics, the most important one being that momentum is a conserved quantity, while velocity is not.

where the subscript E on ψ_E is there to remind us that these are the eigenvectors of $\hat{H}$ corresponding to the energy eigenvalues E. To solve this, we express the wave function $\psi_E(x) = \langle x|\psi_E\rangle$ in the coordinate basis. In this basis we can represent the momentum operator $\hat{p}$ as a derivative $\hat{p} = -i\frac{\partial}{\partial x}$, and the equation takes the form

$$-\frac{^2}{2m}\frac{\partial^2\psi_E(x)}{\partial x^2} + \frac{1}{2}m\omega^2x^2\psi_E(x) = E\psi_E(x).$$

This does not look like something we want to explicitly solve in this course, you can take a proper course on quantum mechanics or differential equations for that.[24] Instead, we simply state that under the assumptions that the wave function is normalizable and symmetric around the equilibrium $x = 0$, we have an infinite family of solutions labeled by a level (or *quantum number*) n

$$\psi_n(x) = \frac{1}{\sqrt{2^n n!}}\left(\frac{m\omega}{\pi}\right)^{1/4} e^{-\frac{m\omega x^2}{2}} H_n\left(\sqrt{m\omega}x\right), \quad n = 0,1,2,\ldots.$$

Here, $H_n(y)$ are the so called (physicist's) Hermite polynomials, with the first few being

$$\begin{aligned} H_0(y) &= 1, \\ H_1(y) &= 2y, \\ H_2(y) &= 4y^2 - 2, \\ H_3(y) &= 8y^3 - 12y, \\ &\vdots \end{aligned}$$

The corresponding energy eigenvalues are

$$E_n = \omega(n + \tfrac{1}{2}).$$

These are the values that would be returned upon a measurement of the Hamiltonian of the quantum harmonic oscillator. Two important things to note are, first that the energies are quantized, i.e., they come in discrete steps; and secondly the lowest value is not equal to zero, but rather $E_0 = \frac{\omega}{2}$. This second point is a consequence of the uncertainty principle.

To connect with the classical system we can calculate the amplitudes, A_n, of a classical harmonic oscillator with the corresponding energies of the quantum one. We find

$$E_n = \frac{1}{2}m\omega^2A_n^2 \implies A_n = \sqrt{(2n+1)\frac{}{m\omega}}.$$

[24] Of course you are welcome to solve it yourselves. A nice trick one can use is to first guess or argue for the expression of the lowest energy state, and then use the fact that $[\hat{x},\hat{p}] = i$ together with the algebra given by introducing the *creation* and *annihilation* operators $a^{\pm} \propto \hat{p} \pm i\omega\hat{x}$ to construct the higher energy states.

Note that these increase with the quantum number n.

Figure 1.3 shows the probability amplitudes, $\psi_n(x)$, and probability densities, $|\psi_n(x)|^2$ of finding the system at the location x, for the first few energy levels in the positional basis. We note two big differences with the classical oscillator. First, there is a non-zero probability of finding the particle outside the values $x = \pm A_n$, this is not possible in the classical system. This is due to something called *quantum tunneling*. Secondly, the probability density distribution for the lowest-energy state $\psi_0(x)$, is highest at the origin $x = 0$, while for the higher values of n we see that the system starts looking more like the classical one, i.e., that it is most likely to find the system near the turning points. This is an illustration of something called the Bohr correspondence principle. Namely that quantum physics should become classical physics in the limit of large quantum numbers (or when becomes small in comparison to the energy).

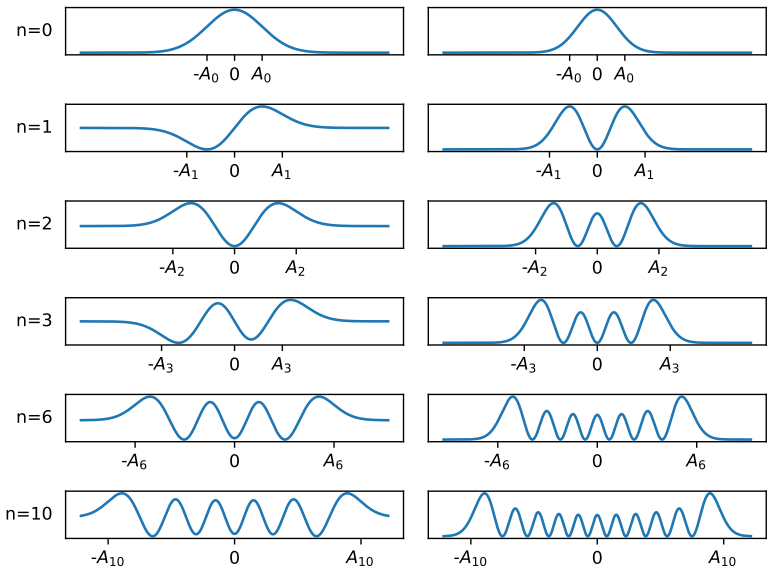


Fig. 1.3: The probability amplitudes (left) and probability densities (right) for some levels of the quantum harmonic oscillator. The classical amplitudes A_n are indicated.

2D HARMONIC OSCILLATOR??

1.7 Entanglement

Let us end this chapter by discussing one of the most mysterious concepts in quantum mechanics, namely that of *entanglement*. We will also elaborate on some of its important consequences for quantum computers.

1.7.1 Product states

We have seen that if we have two physical systems $|\psi_A\rangle$ and $|\psi_B\rangle$, we can combine them into a composite system

$$|\psi_{AB}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle.$$

Let us study this concept in more detail. For simplicity, let us consider a two-qubit system. We then have that both systems $|\psi_A\rangle$ and $|\psi_B\rangle$ can be expressed as a linear combination of the basis states $|0\rangle$ and $|1\rangle$,

$$|\psi_A\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle, \quad |\psi_B\rangle = \beta_0|0\rangle + \beta_1|1\rangle,$$

with the ordinary normalization conditions $|\alpha_0|^2 + |\alpha_1|^2 = |\beta_0|^2 + |\beta_1|^2 = 1$. The combined system looks like

$$|\psi_{AB}\rangle = (\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle) = \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle. \tag{1.3}$$

Furthermore, we have seen that we do not need to consider overall phases for the two individual systems. All in all this means that we have four real degrees of freedom in the combined system. Two coming from each qubit. But, let us now instead consider the most general two-qubit system

$$\gamma_{00}|00\rangle + \gamma_{01}|01\rangle + \gamma_{10}|10\rangle + \gamma_{11}|11\rangle,$$

with the normalization condition now being

$$|\gamma_{00}|^2 + |\gamma_{01}|^2 + |\gamma_{10}|^2 + |\gamma_{11}|^2 = 1.$$

Here, we only have one overall phase to disregard. The generic two-qubit system thus have six real degrees of freedom. Which of course is larger than the four we had before. It is easy to see that the first case is a special case of the more general second case. The extra degrees of freedom between the two cases are exactly what give rise to the mysterious concept of *entanglement*.

A state that can be written on the form (1.3), or more generally as a product

$$|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle \otimes \dots,$$

is, somewhat naturally, called a *product state*, while those that can not be written on this form are called entangled. A simple example, that will play an important role later, would be the state

$$|\psi^-\rangle \coloneqq \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$

This is called a *maximally entangled state* for reasons that will become clear later.

1.7.2 Non-locality

In 1935, Einstein, Podolsky and Rosen (EPR) published a paper called "Can quantum-mechanical description of physical reality be considered complete?" [1]. In this paper, they considered a simple thought experiment that pinpointed some of the mysteries of entanglement. EPR were interested in the question of *completeness* of a physical theory. They defined a complete theory as one where each element of *physical reality*[25] must have a counterpart in the physical theory. They proposed one simple requirement for an element of physical reality such that "if, without in any way disturbing a system, we can predict with certainty (i.e., a probability equal to unity) the value of a physical quantity, then there exists an element of physical reality corresponding to this physical quantity." [1]. They would thus say that two physical quantities corresponding to two non-commuting observables could not have a simultaneous reality, since they can not be measured simultaneously.

We will now give a simple example highlighting how quantum mechanics, or more specifically entanglement, challenges this simple idea.

Let us start with stating a simple fact. A quick calculation shows that for a generic qubit state, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, we have the result

$$\langle\sigma_x\rangle_\psi^2 + \langle\sigma_y\rangle_\psi^2 + \langle\sigma_z\rangle_\psi^2 = 1.$$

This can be interpreted as saying that there is always some direction that has eigenvalue $+1$ for the qubit. It is furthermore easy to check that this continues to hold for the product states (1.3). Let us now look at the entangled state $|\psi^-\rangle$ which we defined above. To this end, we calculate the expectation values of the operators $\sigma_{x,y,z} \otimes \mathbb{1}$ and $\mathbb{1} \otimes \sigma_{x,y,z}$.[26] The result turns out to be zero for all choices. Having a zero expectation value of course simply means that both outcomes are equally

[25] whatever that is,

[26] By the notation $\sigma_{x,y,z}$ we simply mean that we can take any of the three indices.

likely. We thus see that, even though we know the exact state the system is in, namely $|\psi^-\rangle$, we can not say anything about the individual pieces, i.e., the states of the two individual qubits.

Next, we can note that an operator such as $\sigma_z \otimes \sigma_z$ will have $|\psi^-\rangle$ as an eigenstate, more specifically, we have

$$(\sigma_z \otimes \sigma_z)|\psi^-\rangle = -|\psi^-\rangle.$$

Which of course tells us that $\langle \sigma_z \otimes \sigma_z \rangle_{\psi^-} = -1$. This is peculiar for the following reason: We can imagine having the two-qubit system $|\psi^-\rangle$ and distributing each qubit to two people, say Alice and Bob. We then imagine that Alice flies off to Mars with her qubit and Bob stays behind here on Earth. If at Mars, Alice suddenly (and randomly) decides to measure the spin along the z-axis of her qubit (i.e. measure σ_z), she will find one of the results ± 1, but since the combined eigenvalue of hers and Bob's measurement of σ_z must be equal to -1, she will immediately know what the result of Bob's measurement would be. For example, if Alice finds the result $+1$ she immediately knows that Bob must find -1 and vice versa. This is what Einstein famously called *spooky action at a distance*.

CONNECT BACK TO THE EPR PAPER!!!

1.7.3 Bell inequalities and CHSH

The EPR paper did not receive a lot of attention after its publication. By many it was mostly considered to be some philosophical detail that one need not care about when doing physics. It was not until John Bell in 1964, almost 30 years after the EPR paper, published an idea for an experiment that could make use of the entanglement introduced by EPR to make predictions about the nature of quantum mechanics.

We will discuss a variant of Bell's proposed experiment due to Clauser, Horne, Shimony and Holt (CHSH) [2].

We consider the following game. We have two players, Alice (A) and Bob (B) and one game host, Charlie (C). Charlie chooses two questions $xy \in \{00, 01, 10, 11\}$ uniformly and Alice and Bob will answer with a single bit a and b, respectively. They will win if $a \oplus b = x \wedge y$.[27] In other words, we need

$$\begin{aligned} a(0) \oplus b(0) &= 0, \\ a(0) \oplus b(1) &= 0, \\ a(1) \oplus b(0) &= 0, \\ a(1) \oplus b(1) &= 1. \end{aligned} \tag{1.4}$$

[27] Here, $\oplus$ means addition modulo 2.

If we consider classical (and deterministic) strategies, we can easily see that there are 16 possible ones. For example, one strategy would be for both Alice and Bob to always answer every question with zero. By comparing the different strategies with the winning ones of (1.4) we will see that there is no classical strategy that can win every time. The best we can do is to choose a strategy that wins $3/4$ of the times. One such example is the strategy of always answering zero to every question.[28]

The big question is now if we can do better by considering quantum strategies. For simplicity we consider the answers to be either ± 1 instead of 0 and 1. This of course makes no real difference, we can for example consider the map $a \mapsto (-1)^a$ to take us from one convention to the other.

In the quantum version we consider Alice and Bob to share an entangled state, say

$$|\varphi^+\rangle := \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle),$$

such that they have one qubit each of this state. Before answering the question they both make a measurement on their corresponding qubit, such that $a(0)$ corresponds to the measurement of σ_z, $a(1)$ of σ_x, $b(0)$ of $H = \frac{1}{\sqrt{2}}(\sigma_z + \sigma_x)$ and $b(1)$ of $\frac{1}{\sqrt{2}}(\sigma_z - \sigma_x)$. By calculating the corresponding expectation values,

$$\langle a(0)\otimes b(0)\rangle_{\varphi^+} = \langle a(0)\otimes b(1)\rangle_{\varphi^+} = \langle a(1)\otimes b(0)\rangle_{\varphi^+} = \frac{1}{\sqrt{2}}, \quad \langle a(1)\otimes b(1)\rangle_{\varphi^+} = -\frac{1}{\sqrt{2}}.$$

These expectation values measures the expectation that Alice and Bob win minus the expectation that they loose on each of the questions $\{00, 01, 10\}$ and minus this on the question $xy = 11$. Therefore, we find that the total probability of winning minus the probability of loosing is

$$\frac{1}{4}\langle\varphi^+|a(0)\oplus b(0) + a(0)\otimes b(1) + a(1)\otimes b(0) - a(1)\otimes b(1)|\varphi^+\rangle = \frac{1}{\sqrt{2}},$$

and the probability of winning is therefore

$$\frac{1}{2}\left(1 + \frac{1}{\sqrt{2}}\right) \sim 0.85.$$

This is of course better than the $3/4$ probability in the classical setting.

In physics literature, this is more often stated as the fact that the inequality, known as a Bell inequality, [29]

$$a(0)b(0) + a(0)b(1) + a(1)b(0) - a(1)b(1) \leq 2,$$

[28] As an exercise you can assure yourself that we can not do better by considering probabilistic strategies.

[29] more specifically here the CHSH inequality,

which clearly holds for classical variables $a, b \in [-1, 1]$, can be violated by considering the above quantum situation, which gives

$$\langle a(0) \otimes b(0) \rangle_{\varphi^+} + \langle a(0) \otimes b(1) \rangle_{\varphi^+} + \langle a(1) \otimes b(0) \rangle_{\varphi^+} - \langle a(1) \otimes b(1) \rangle_{\varphi^+} = 2\sqrt{2}.$$

1.7.4 The GHZ paradox

Greenberger, Horn and Zeilinger (GHZ) [3] came up with a sort of deterministic variant of the Bell inequality game which perhaps gives an even more striking paradox than the earlier discussion. Namely, it gives a problem where the quantum strategy can win with certainty every time, while the classical can not.

We now imagine a three-party game where Alice, Bob and Charlie each are asked one out of two questions x, y and z, with possible answers again given by a bit, 0 or 1. They now win if $a \oplus b \oplus c = x \vee y \vee z$. The winning strategies should thus satisfy

$$\begin{aligned} a(0) \oplus b(0) \oplus c(0) &= 0, \\ a(0) \oplus b(1) \oplus c(1) &= 1, \\ a(1) \oplus b(0) \oplus c(1) &= 1, \\ a(1) \oplus b(1) \oplus c(0) &= 1. \end{aligned}$$

It is straightforward to once again assure us that no classical strategy can do better than win 75% of the time.

For the quantum strategy, we consider the case that Alice, Bob and Charlie are each given one qubit from the entangled state

$$|GHZ\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle).$$

The measurements can then correspond to σ_x and σ_y, since we can easily check that

$$\begin{aligned} \sigma_x \otimes \sigma_x \otimes \sigma_x |GHZ\rangle &= +|GHZ\rangle, \\ \sigma_x \otimes \sigma_y \otimes \sigma_y |GHZ\rangle &= -|GHZ\rangle, \\ \sigma_y \otimes \sigma_x \otimes \sigma_y |GHZ\rangle &= -|GHZ\rangle, \\ \sigma_y \otimes \sigma_y \otimes \sigma_x |GHZ\rangle &= -|GHZ\rangle. \end{aligned}$$

Where we again considered the answers ± 1 instead of 0 and 1. By the same argument as in the CHSH game of before we can now see that the GHZ game has a strategy that wins every time.

1.7.5 Bell basis and measurements

We have seen two examples of maximally entangled two-qubit states, $|\psi^-\rangle$ and $|\varphi^+\rangle$. They are in fact two out of four basis states of maximally entangled states. This basis is called the Bell basis,

$$|\varphi^\pm\rangle := \frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle),$$
$$|\psi^\pm\rangle := \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle).$$

As we have seen, and will continue to see, they play an important role in various thought experiments involving quantum entanglement.

Due to their importance in quantum theory, it is worthwhile to consider how we can construct the Bell states out of two generic qubits. If we start from a state in the computational basis we can create a Bell state by acting with the Hadamard gate, $H = \frac{1}{\sqrt{2}}(\sigma_x + \sigma_z)$ on the first qubit and then the so called controlled NOT, or CNOT, gate, with the newly transformed first qubit as the control. This gate acts by first controlling the state of the control qubit. If this is 0 it does nothing to the other qubit, while if it is 1 it acts with σ_x on the other qubit, flipping it between 0 and 1.[30] For example, if we start from the state $|00\rangle$, we find first that

$$H \otimes \mathbb{1}|00\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle,$$

and the CNOT thus transforms this into

$$\frac{1}{\sqrt{2}}\text{CNOT}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |\varphi^+\rangle.$$

Similarly, we find that the same circuit transforms $|11\rangle$ into $|\varphi^-\rangle$, $|01\rangle$ into $|\psi^+\rangle$ and $|10\rangle$ into $|\psi^-\rangle$.

Equally important is the Bell measurement. Given two maximally entangled qubits we can perform a Bell measurement to determine which of the Bell states the entangled qubits are in, and thus entangle the information. This measurement is just the Bell creation circuit, just presented, run in the opposite order. We start by acting with the CNOT gate, followed by the Hadamard on the control qubit. This is a key ingredient in the quantum teleportation protocol, which we discuss next.

[30] These gates will be more properly introduced later as they are of course very important for the course.

1.7.6 Teleportation

The notion of entanglement is a very powerful one. To show some of its consequences, we will now discuss how quantum mechanics allows a form of teleportation of information.

We return again to our dear friends Alice and Bob. Alice was recently given a qubit

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

that she wants to send to Bob. However, they are very far away from each other and only have access to a measuring device and a telephone. So this seems hard. But perhaps there is a way? In other words, Alice needs to share some classical information over the phone such that Bob can recreate her state $|\psi\rangle$. There is a deep result in quantum mechanics called the *no-cloning theorem* that states that Bob can not simply copy Alice's state exactly.[31] Instead what we will see is that Alice will make a certain measurement changing her state but allowing her to retrieve some information that she can send to Bob such that he can rebuild the original state $|\psi\rangle$. This procedure is then what is called *quantum teleportation*.

First of all, let us note that we have seen that quantum mechanics does not allow for any direct measurement of Alice to simply get the numbers α and β such that she can communicate them to Bob. Since the measurement would change the state and she would not get the complete information of the original state. Instead we will come up with another prescription.

For this to work we imagine that besides the original qubit $|\psi\rangle$, Alice and Bob both have one qubit each from an entangled Bell pair $|\varphi_{AB}\rangle$. The steps of the procedure are easy enough and given by:

1. Alice makes a Bell measurement of her combined system of two qubits $|\psi\rangle$ and $|\varphi_A\rangle$;

2. Alice makes a measurement to decide which states $|00\rangle$, $|01\rangle$, $|10\rangle$ or $|11\rangle$ her combined system is in;

3. depending on the outcome she gives an instruction to Bob, as follows:

   - if $|00\rangle$ do nothing;
   - if $|01\rangle$ apply σ_x;
   - if $|10\rangle$, apply σ_z;

[31] You could try and derive this theorem, everything you need has been discussed in the course already.

- if $|11\rangle$, apply $\sigma_z\sigma_x$.

4. if Bob follows Alice's instructions he will now have the state $|\psi\rangle$.

So why does this work? Let us do the maths. For the Bell pair we take $|\varphi_{AB}\rangle = |\varphi^+\rangle$. The original system is then

$$|\psi\rangle \otimes |\varphi_{AB}\rangle = \frac{1}{\sqrt{2}}(\alpha|0\rangle + \beta|1\rangle) \otimes (|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle).$$

Alice then makes a Bell measurement on the first two qubits of this system. Remember that this means CNOT followed by Hadamard on the first qubit:

$$\begin{aligned}(H \otimes \mathbb{1} \otimes \mathbb{1})(\text{CNOT} \otimes \mathbb{1})(|\psi\rangle \otimes |\varphi_{AB}\rangle =& (H \otimes \mathbb{1} \otimes \mathbb{1}) \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle) \\ =& \frac{1}{2}(\alpha(|000\rangle + |011\rangle + |100\rangle + |111\rangle) + \beta(|010\rangle - |110\rangle + |001\rangle - |101\rangle)).\end{aligned}$$

This can be rearranged into

$$\frac{1}{2}(|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)).$$

Next, Alice measures her two qubits. This will give one of the results $|00\rangle$, $|01\rangle$, $|10\rangle$ or $|11\rangle$, with equal probability, projecting Bob's state to the corresponding parenthesis in the above expression. We thus see that if Alice obtains the result corresponding to $|00\rangle$ Bob's state will be $|\psi\rangle$, which is what we wanted, so no further action is needed. If Alice finds $|01\rangle$, Bob's state is $\alpha|1\rangle + \beta|0\rangle$, and acting on this with σ_x gives $|\psi\rangle$. Similarly, if Alice finds $|10\rangle$ Bob should act with σ_z and $|11\rangle$ means that he should act with $\sigma_z\sigma_x$ to get $|\psi\rangle$.

As we stated in the beginning, we also see that Alice's qubit is of course no longer in the state $|\psi\rangle$, it has collapsed to an eigenstate of her measurement. We thus say that she has teleported her state to Bob.

1.8 Further reading

Chapter 2

Theoretical Computer Science 101

Before we consider quantum computing, it is worthwhile to review classical computing. Modern computers are very complicated. People hence study many abstractions of the workings of a computer, called "models of computation". In this chapter, we will introduce three such models of computation.

2.1 Traditional Computer Science

Computer Science grew out of the work led by David Hilbert, who made significant contributions to the field of mathematics, including the development of formal axiomatic systems, which laid the foundation for the study of mathematical logic and the formalization of algorithms. His work in these areas has influenced the development of theoretical computer science, including the study of computability and complexity theory. Additionally, Hilbert's work on geometry and his development of the concept of Hilbert spaces have had an impact on the field of computer graphics as well as quantum mechanics. In the context of this chapter it is important to stress that it is Hilbert who tried to distinguish between problems that can be solved by simple methods and those which can not.

Much of computer science uses a language-inspired definition of a decision problem. One starts with an finite alphabet A. By stringing elements of the alphabet one after another, one obtains strings of finite or countably infinite length. A set of strings is called a language. A decision problem is defined by a fixed set S, which is a subset of the language U of all possible strings over the alphabet A. A particular instance of the decision problem is to decide, given an element $u \in U$, whether u is included in S.

Example 2.1 (Primality testing.)**.** For example, alphabet could be composed of binary digits $A = \{0,1\}$, U could be the set of all natural numbers encoded in binary,

and set S could be the binary encodings of prime numbers. The decision problem is the inclusion of an arbitrary binary encoding of a natural number in the set of S. $\diamondsuit$

Several models of computation were devised. Alan Turing introduced a model, where characters are stored on an infinitely long tape, with a read/write head scanning one square at any given time and having very simple rules for changing its internal state based on the symbol read and current state. Another influential model, called Lambda Calculus, has been introduced by Alonzo Church. Many of these formalisms turn out to be equivalent in computational power, *i.e.*, any computation that can be carried out with one can be carried out with any of the others. As it turns out, quantum computing may be one of the first models, where this is not the case.

2.1.1 Turing Machines

Formally, one can define a Turing machine using:

- a finite, non-empty set Q of objects, representing states
- a subset F of Q, corresponding to "accepting" states, where computation halts
- $q_0 \in Q$, the initial state
- a finite, non-empty set Γ of objects, representing the symbols to be used on a tape
- a partial function $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\}$ where for a combination of a state and symbol read from the tape, we get the next state, the symbol to write onto the tape, and an instruction to shift the tape left (-1), right (+1), or keep in its position (0).

Notice that here we assume the input is on the tape, at the beginning.

Example 2.2 (There and Back Again.)**.** Let us, for example, construct a machine, which scans over an integer encoded in binary and delimited by "blank" on the tape from left to right, and back. This is not very useful, but will be easy to understand:

- $Q = \{\text{goingright}, \text{goingleft}, \text{halt}\}$
- $F = \{\text{halt}\}$
- $q_0 = \text{goingright}$
- $\Gamma = \{0, 1, \text{“}blank''\}$
- δ given by the table below:

Current state	Scanned symbol	Print symbol	Move tape	Next state
goingright	0	0	1	goingright
goingright	1	1	1	goingright
goingright	blank	blank	-1	goingleft
goingleft	0	0	-1	goingleft
goingleft	1	1	-1	goingleft
goingleft	blank	blank	0	halt

♢

Exercise 2.3. Consider the following simulator of a Turing machine (TM):

```
def turing(code, tape, initPos = 0, initState = "1"):
  position = initPos
  state = initState
  while state != "halt":
    print f"{state} : {position} in {tape}"
    symbol = tape[position]
    (symbol, direction, state) = code[state][symbol]
    if symbol != "noWrite": tape[position] = symbol
    position += direction
```

code/ch1/turing.py

Implement a TM, which checks whether an integer, which is encoded on the tape as in binary and delimited by "blank" on both ends of the tape, is odd. If so, it should replace all symbols representing the integer with "1". Otherwise, it should replace all symbols representing the integer with "0".

Exercise 2.4. Consider the same simulator of a Turing machine (TM) as in Exercise 2.3. Implement a TM, which adds two integers, which are encoded on the tape in binary and delimited by "blank" on both ends of the tape and between the numbers. Replace both numbers with the result.

Exercise 2.5. Consider the simulator of a Turing machine (TM) of Exercise 2.3. Implement a TM, which multiplies two integers, which are encoded on the tape in unary and and delimited by "blank" on both ends and between the numbers. Do not replace the numbers, but append the result after yet another blank.

Hint: Unary encoding means the number of a ocurrences of a particular symbols (e.g., "1") is equal to the number (e.g., "11111" stands for 5).

2.1.2 Computability

Computability studies these models of computation, and asks which problems can be proven to be unsolvable by a computers. For example:

Example 2.6 (The Halting Problem)**.** Given a program and an input to the program, will the program eventually stop when given that input? ♢

A silly solution would be to just run the program with the given input, for a reasonable amount of time. If the program stops, we know the program stops. But if the program doesn't stop in a "reasonable" amount of time, we cannot conclude that it *won't* stop. Maybe we didn't wait long enough. Alan Turing proved the Halting problem to be undecidable in 1936.

2.1.3 Complexity theory

Some problems are solvable by a computer, but require such a long time to compute that solution is impractical. Here, we express the run time as a function from the dimensions of the input to the numbers of steps of a Turing machine (or similar).

Example 2.7 (Fischer-Rabin Theorem.)**.** For example, let us have a logic featuring 0, 1, the usual addition, and where the axioms are a closure of the following:

- $\neg(0 = x+1)$
- $x+1 = y+1 \Rightarrow x = y$
- $x+0 = x$
- $x+(y+1) = (x+y)+1$
- For a first-order formula $P(x)$ (i.e., with the universal and existential quantifiers) with a free variable x, $(P(0) \wedge \forall x(P(x) \Rightarrow P(x+1))) \Rightarrow \forall y P(y)$ ("induction").

This is known as the Presburger arithmetic. Fischer and Rabin proved in 1974 that every classical algorithm that decides the truth of statement of length n in Presburger arithmetic has a runtime of at least $2^{2^{cn}}$ for some constant c, because it may need to produce output of that size. Hence, this problem needs more than exponential run time. ♢

Complexity theory deals with questions concerning the time or space requirements of given problems: the *computational cost*. For algorithms working with finite strings from a finite alphabet, this is often surprisingly easy.

2.1.4 Computational Complexity of Discrete Algorithms

The term *analysis of algorithms* is used to describe general approaches to putting the study of the performance of computer programs on a scientific basis. One such

Notation	Definition	Analogy
$f(n) = O(g(n))$	see Def. 2.9	$\leq$
$f(n) = o(g(n))$	see Def.	$<$
$f(n) = \Omega(g(n))$	$g(n) = O(f(n))$	$\geq$
$f(n) = \omega(g(n))$	$g(n) = o(f(n))$	$>$
$f(n) = \Theta(g(n))$	$f(n) = O(g(n))$ and $g(n) = O(f(n))$	$=$

Table 2.1: An overview of the Bachmann–Landau notation.

approach[1] concentrates on determining the growth of the worst-case performance of the algorithm (an "upper bound"): An algorithm's "order" suggests asymptotics of the number of operations carried out by the algorithm on a particular input, as a function of the dimensions of the input.

Example 2.8. For example, we might find that a certain algorithm takes time $T(n) = 3n^2 - 2n + 6$ to complete a problem of size n. If we ignore

- constants (which makes sense because those depend on the particular hardware/virtual machine the program is run on), and
- slower growing terms such as $2n$,

we could say "$T(n)$ grows at the order of n^2". ♢

2.1.5 The Bachmann–Landau Notation

Let us introduce a formalisation of the notion of asymptotics. The formalisation known as "Big O notation" or "Bachmann–Landau notation" goes back at least to 1892 and Paul Gustav Heinrich Bachmann, according to some sources, although it was reinvented many times over. Suppose our A requires $T(n)$ operations to complete the algorithm in the longest possible case. Then we may say A is $O(g(n))$ if $|T(n)/g(n)|$ is bounded from above as $n \to \infty$. The fastest growing term in $T(n)$ dominates all the others as n gets bigger and so is the most significant measure of complexity.

Similarly to "Big O", there are 4 more notions, as summarised in Table 2.1. Formally, suppose f and g are two real-valued functions defined on some subset of $\mathbb{R}$and consider the following:

[1] Introduced by Hartmanis and Stearns in: Juris Hartmanis and Richard Stearns (1965), On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, **117**:285–306; and popularised by Aho, Hopcroft and Ullman.

Definition 2.9. We write:

$$f(x) = O(g(x)) \qquad \text{(or, to be more precise, } f(x) = O(g(x)) \text{ for } x \to \infty)$$

if and only if there exist constants N and $C > 0$ such that

$$|f(x)| \leq C|g(x)| \text{ for all } x > N \quad \text{or, equivalently, } \frac{|f(x)|}{|g(x)|} \leq C \text{ for all } x > N.$$

That is, $|f(x)/g(x)|$ is bounded from above as $x \to \infty$. Intuitively, this means that f does not grow faster than g. The letter "O" is read as "order" or just "Oh".

Definition 2.10. We also write:

$$f(x) = \Omega(g(x)) \qquad \text{(for } x \to \infty)$$

if and only if there exist constants N and $C > 0$ such that

$$|f(x)| \geq C|g(x)| \text{ for all } x > N \quad \text{or, equivalently, } \frac{|f(x)|}{|g(x)|} \geq C \text{ for all } x > N.$$

That is, $|f(x)/g(x)|$ is bounded from below by a *positive* (*i.e.*, non-zero) number as $x \to \infty$. Intuitively, this means that f does not grow more slowly than g (*i.e.*, $g(x) = O(f(x))$). The letter "Ω" is read as "omega" or just "bounded from below by".

Definition 2.11.

$$f(x) = \Theta(g(x)) \qquad \text{(for } x \to \infty)$$

if and only if there exist constants N, C and $D > 0$ such that

$$D|g(x)| \leq |f(x)| \leq C|g(x)| \text{ for all } x > N \quad \text{or, equivalently, } D \leq \frac{|f(x)|}{|g(x)|} \leq C \text{ for all } x > N.$$

That is, $|f(x)/g(x)|$ is bounded from both above and below by positive numbers as $x \to \infty$. Intuitively, this means that f grows roughly at the same rate as g.

Example 2.12. Let us consider algorithm A with parameter n and polynomial runtime $O(n^k)$. By our definition of O, the algorithm is of order $O(n^k)$ if $|T(n)/n^k|$ is bounded from above as $n \to \infty$, or — equivalently — there are real constants $a_0, a_1, \ldots, a_k$ with $a_k > 0$ so that A requires

$$a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$$

operations to complete in the worst case. Note that k is an integer constant independent of the algorithm input and independent of the parameter n. It may be that there is no such polynomial for the number of operations in terms of n. If there is such

Notation	*Name*
$O(1)$	constant
$O(\log(n))$	logarithmic
$O((\log(n))^c)$	polylogarithmic
$O(n)$	linear
$O(n^2)$	quadratic
$O(n^c)$	polynomial
$O(c^n)$	exponential
$O(n!)$	factorial

Table 2.2: Classes of functions commonly encountered in algorithm analysis

a polynomial, A is usually considered "good" as it does not require "very many" operations. ♢

This notation can also be used with multiple parameters and with other expressions on the right side of the equal sign. The notation:

$$f(n,m) = n^2 + m^3 + O(n+m)$$

represents the statement:

$$\text{there exist } C,N \text{ such that, for all } n,m > N : f(n,m) \leq n^2 + m^3 + C(n+m).$$

Similarly, $O(mn^2)$ would mean the number of operations the algorithm carries out is a polynomial in two indeterminates n and m, with the highest degree term being mn^2, *e.g.*, $2mn^2 + 4mn - 6n^2 - 2n + 7$. This is most useful if we can relate m and n (*e.g.*, in dense graphs $m = O(n^2)$, so $O(mn^2)$ would mean $O(n^4)$ there).

Let us list a number of classes of functions that are commonly encountered in the analysis of algorithms.

Here, c is some arbitrary constant positive real number. Once again, if a function $f(n)$ is a sum of functions, the fastest growing one determines the order of $f(n)$. E.g.: If $f(n) = 10\log(n) + 5(\log(n))^3 + 7n + 3n^2 + 6n^3$, then $f(n) = O(n^3)$.

One caveat here: the number of summands must be constant and may not depend on n.

Note that $O(n^c)$ and $O(c^n)$ are very different. The former is polynomial, the latter is exponential and grows much, much faster, no matter how big the constant c is. A function that grows faster than $O(n^c)$ is called *superpolynomial*. One that grows slower than $O(c^n)$ is called *subexponential*. An algorithm can require time that is both superpolynomial and subexponential.

Note, too, that $O(\log n)$ is exactly the same as $O(\log(n^c))$. The logarithms differ only by a constant factor, and the big O notation ignores that. Similarly, logs with different constant bases are equivalent.

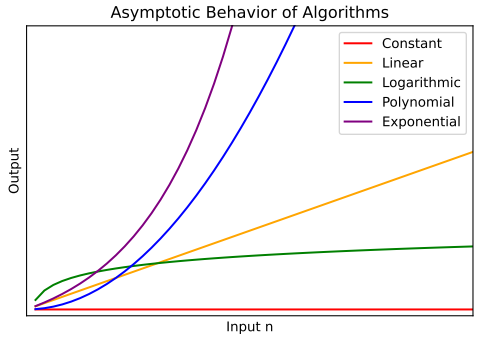


Fig. 2.1: Schematic of the asymptotic runtime of algorithms as a function of their input size n.

Exercise 2.13. Prove that any later function in the above table grows faster than any earlier function. *Hint*: you need several small proofs. Also, each function is differentiable.

2.1.6 P *and* NP

Perhaps the best known question in Computer Science asks whether it can be harder to solve a problem than to check a given solution.

In complexity theory there are two commonly used classes of (decision) problems:

- The class **P** consists of all those decision problems that can be solved on a deterministic Turing machine in an amount of time that is polynomial in the size of the input, *i.e.*, $O(n^k)$ for some constant k. Intuitively, we think of the problems in **P** as those that can be solved "reasonably fast".
- The class **NP** consists of all those decision problems whose *solutions* (called witnesses) can be verified in polynomial time on a Turing machine. That is, given a proposed solution to the problem, we can check it really *is* a solution in polynomial time.

Formally: A language $L \subset \{0,1\}^*$ is in **NP**, if there exists a deterministic Turing machine M and a polynomial p such that upon receipt of:

- an input string x, e.g., $x \in \{0,1\}^*$,

- a witness of length $p(|x|)$

M runs in time polynomial in $|x|$ and

- for all $x \in L$, there exists y such that M accepts (x,y) ("completeness"),
- for all $x \notin L$, for all y, (x,y) is rejected ("soundness").

2.2 Randomized Algorithms

It seems quite unlikely that the Turing machine can produce a truly random number. But would the availability of a source of randomness make a Turing machine more powerful? We will formalise the question using the classes of Probabilistic Polynomial Time (PP) and Bounded-Error Probabilistic Polynomial Time (BPP), where BPP $\subset$ PP. It is not known whether BPP is equal to P or NP, i.e., whether the source of randomness helps at all or whether having access to a source of randomness makes a deterministic Turing machine as powerful as a non-deterministic Turing machine, despite much attention paid to the questions over the past couple of decades. On the other hand, it is known NP $\subset$ PP and, in a somewhat different formalisation of [4], we will see that the source of randomness does render many classes of computation (LOGSPACE^A, P^A, NP^A, PP^A, and PSPACE^A) properly contained in this order, with probability 1 with respect to random oracles A.

2.2.1 Definitions

In two important definitions of randomise computation, one considers a deterministic Turing machine M, which receives:

- an input string x, e.g., $x \in \{0,1\}^*$,
- a realisation y, e.g., $y \in \{0,1\}^*$, of a random variable Y

and

- accepts the input (x,y) for all x that we would like to be accepted with a certain probability,
- rejects (x,y) for all x we would like to be rejected with a certain probability,

where the probability is with respect to Y.

PP. A language $L \subset \{0,1\}^*$ is in PP, if there exists a deterministic Turing machine M and a polynomial p such that upon receipt of:

- an input string x, e.g., $x \in \{0,1\}^*$,
- a realisation y of length $p(|x|)$, e.g., $y \in \{0,1\}^{p(|x|)}$, of a random variable Y

M runs in time polynomial in $|x|$ and

- for all $x \in L$, (x,y) is accepted with a probability strictly greater than 1/2,
- for all $x \notin L$, (x,y) is accepted with a probability less than or equal than 1/2,

where the probability is with respect to Y.

In PP, we hence ask only for some "distinguishability". The "distinguishing" can, however, take arbitrarily long. Consider, for instance, a Turing machine M of the definition, that

- for all $x \in L$, (x,y) is accepted with probability $1/2 + 1/2|x|$
- for all $x \notin L$, (x,y) is accepted with probability $1/2 - 1/2|x|$.

For any number of trials, there is an $|x|$ that makes those necessary to achieve a fixed probability of the answer being correct. Notice that the number of trials grows expoentnially with $|x|$.

Alternatively, PP is the set of languages, for which there is a variant of a non-deterministic Turing machine that stops in polynomial time with the acceptance condition being that more than one half of computational paths accept. One hence sometimes refers to PP as Majority-P. It is hence clear that NP $\subseteq$ PP.

PP is a often thought of as a counting class. Recall that the permanent of an $n \times n$ matrix $A = (a_{ij})$ is

$$\mathrm{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}. \tag{2.1}$$

[5] showed that computing permanents is at least as hard as many so-called counting problems (#P-hard), and it is hard (#P-complete) even for matrices with entries 0 or 1. The language $\{(A,k)|$ the permanent of A is at least $k\}$ is complete for PP, but it is believed to be outside of P. Alternatively, in terms of the number of accepting and rejecting paths, PP can be seen as computing the high-order bit of a #P function.

BPP. Let ε be a constant $0 < \varepsilon < 1/2$. A language $L \subset \{0,1\}^*$ is in BPP, if there exists a deterministic Turing machine M and a polynomial p such that upon receipt of:

- an input string x, e.g., $x \in \{0,1\}^*$,
- a realisation y, e.g., $y \in \{0,1\}^{p(|x|)}$, of a random variable Y in dimension $p(|x|)$

M runs in time polynomial in $|x|$ and

- for all $x \in L$, (x,y) is accepted with a probability strictly greater than $1-\varepsilon$,
- for all $x \notin L$, (x,y) is accepted with a probability less than or equal than ε,

where the probability is with respect to Y.

BPP can be seen as a subset of PP, for which there are efficient probabilistic algorithms. Because the constant ε is independent of the dimension $|x|$, we can achieve any desired probability of correctness with the number of trials independent of $|x|$ by the so-called *amplification of probability*. The majority vote of k trials will be wrong with probability:

$$\sum_{S\subseteq\{1,2,\ldots,k\},|S|\leq k/2} (1-\varepsilon)^{|S|}\varepsilon^{k-|S|} \tag{2.2}$$

$$= ((1-\varepsilon)\varepsilon)^{k/2} \sum_{S\subseteq\{1,2,\ldots,k\},|S|\leq k/2} \left(\frac{\varepsilon}{1-\varepsilon}\right)^{k/2-|S|} \tag{2.3}$$

$$< 2^k(\sqrt{(1-\varepsilon)\varepsilon})^k = \lambda^k \tag{2.4}$$

for some $\lambda = 2\sqrt{\varepsilon(1-\varepsilon)} < 1$. Cf. 4.1 in [6]. How large is this class within PP?

It turns out that BPP is a substantial subset of PP. [4] have shown that for a language $L \subset \{0,1\}^*$, the following are equivalent:

- $L \in$ BPP.
- For almost all oracles A, $L \in P^A$, wherein the almost all is with respect to a particular measure over the oracles.

It turns out that BPP has yet another definition, due to [7, Section 20.2], which is very instructive. It uses a seemingly different model of computation. There, one works with 2^N-dimensional vector $v \in [0,1]^{2^N}$, which we index with values from $\{0,1\}^N$, and which satisfies $\sum_{i\in\{0,1\}^N} v_i = 1$. This vector should be seen as a representation of a probability mass function of a random variable over $\{0,1\}^N$. One cannot access the values of v directly; rather, one obtains $i \in \{0,1\}^N$ with probability v_i, when one attempts to access v.

Let us introduce a special notation $|i\rangle$ for the representation of (so-called degenerate) distributions, where all the mass is concentrated in $v_i = 1$ for some $i \in \{0,1\}^N$. Because $|i\rangle_{i\in\{0,1\}^N}$ is a basis for $\mathbb{R}^{2^N}$, any v can be represented as $\sum_{i\in\{0,1\}^N} v_i |i\rangle$. For the example of $N = 1$, we have $v = v_0|0\rangle + v_1|1\rangle$. The only operations permitted are

linear stochastic functions $U : \mathbb{R}^{2^N} \to \mathbb{R}^{2^N}$ applied to the vector v, where linearity suggests $U(v) = \sum_{i\in\{0,1\}^N} v_i U(|i\rangle)$ and stochasticity suggests $\sum_{i\in\{0,1\}^N} U(v)_i = 1$ for all v satisfying $\sum_{i\in\{0,1\}^N} v_i = 1$. Notice that U can be represented by a matrix with non-negative entries, wherein each column sums up to 1. U can be a composition of multiple linear stochastic functions $U = U_L, U_{L-1}, \cdots U_2, U_1, U_i : \mathbb{R}^{2^N} \to \mathbb{R}^{2^N}$, where each U_i will represent the so-called gate and L will be the known as the depth of the circuit.

Let a probability threshold be a constant strictly larger than 1/2. A language $L \subset \{0,1\}^n$ is in BPP, if and only if its corresponding indicator function $F(x) : \{0,1\}^n \to \{0,1\}$ can be computed probabilistically in polynomial time such that:

1. one starts with $v \in [0,1]^{2^N}$, for some $N \geq n$ dependent on F, with an initial state $|x, 0^{N-n}\rangle$ consisting of the input padded to length N by zeros;
2. applies a linear stochastic function $U : \mathbb{R}^{2^N} \to \mathbb{R}^{2^N}$ to v, whose matrix representation can be computed in a sparse format by a Turing machine from all-ones input in time polynomial in n
3. obtains a random variable Y, wherein $F(x)$ is followed by $N-1$ arbitrary subsequent symbols with probability at least as high as the probability threshold, wherein the random variable Y has value y with probability v_y for the value v of some final register.

Exercise 2.14. Prove the equivalence. Hint: find a way of generating $N-n$ Bernoulli random variables by a suitable U.

2.3 Quantum Algorithms

Now, one can obtain the class of BQP by replacing the real-valued vectors with complex-valued vectors:

Let a probability threshold be a constant strictly larger than 1/2. A language $L \subset \{0,1\}^n$ is in BQP, if and only if its corresponding indicator function $F(x) : \{0,1\}^n \to \{0,1\}$ can be computed probabilistically such that:

1. one starts with an N-qubit register, for some $N \geq n$ dependent on F, with an initial state $|x, 0^{N-n}\rangle$ consisting of the input padded to length N by zeros;
2. applies a linear function $U : \mathbb{C}^{2^N} \to \mathbb{C}^{2^N}$ to v, whose matrix representation (a unitary matrix in $\mathbb{C}^{2^N \times 2^N}$) can be computed in a sparse format by a Turing machine from all-ones input in time polynomial in n

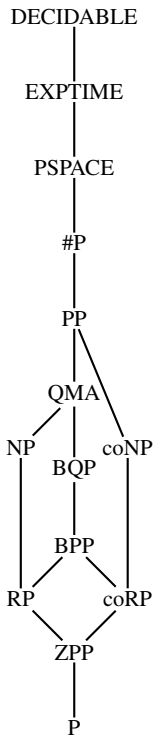


Fig. 2.2: An overview of some of the non-strict inclusions among complexity classes.

3. obtains a random variable Y, wherein $F(x)$ is followed by $N-1$ arbitrary subsequent symbols with probability at least as high as the probability threshold, wherein the random variable Y has value y with probability $|v_y|^2$ for the value v of some final register.

See Figure 2.2 for an overview if the complexity classes discussed.

Chapter 3

Quantum Computing 101

3.1 What we have seen so far?

In quantum computing, instead of symbols from finite alphabets (e.g., bits), one works with vectors in suitable complex vector spaces. An extension of BPP to this setting is known as BQP.

3.1.1 Qubits

One of the main differences between classical and quantum physics is the fact that quantum states are described by vectors in a complex vector space, rather than binary strings. Abstractly, we use the Dirac, or bra-ket, notation to denote a state vector. If the system is in some state, let us call it ψ, we denote this as

$$|\psi\rangle.$$

This is called a ket. The ψ is just a label of the state while the encasing $|\cdot\rangle$ is there to remind us that this is a vector.

The ket vectors satisfy the ordinary axioms of a vector space. Under addition, the vector space is closed, associative and commutative. There is a unique zero element, which we denote simply by 0, such that

$$|\psi\rangle + 0 = |\psi\rangle. \tag{3.1}$$

The reason why we do not use $|0\rangle$ to denote the zero vector is because we want to reserve that notation for something completely different, as we will see in a short while. There is also a unique vector $(-|\psi\rangle)$ such that

$$|\psi\rangle + (-|\psi\rangle) = 0. \tag{3.2}$$

The vector space is linear and distributive under scalar multiplication. This means that for some complex numbers $z, z_1, z_2 \in \mathbb{C}$,

$$|(z_1 + z_2)\psi\rangle = z_1|\psi\rangle + z_2|\psi\rangle, \quad z(|\psi\rangle + |\varphi\rangle) = z|\psi\rangle + z|\varphi\rangle. \tag{3.3}$$

Formally, single qubit can be seen as a two-dimensional complex space, $\mathbb{C}^2$, associated with an inner product and a basis. The standard complex inner product is ${v_i}^\dagger w_i$. The standard basis is $\{|0\rangle, |1\rangle\}$. Together with the inner product, we call $\mathbb{C}^2$ a Hilbert space, sometimes denoted $\mathscr{H}_2$. The usual representation of the state of a qubit is that of unit vector $\mathbb{R}^3$ on the so-called Bloch sphere, see Fig. 3.1, which is isomorphic to the complex projective plane $\mathbb{CP}^1$. As such, a qubit's state can be completely characterized as the unit vector on the unit sphere. A quantum state $|\psi\rangle$ and a quantum state $c|\psi\rangle$, $c \in \mathbb{C}$ are indistinguishable. Sometimes, this phase factor is called "global gauge".

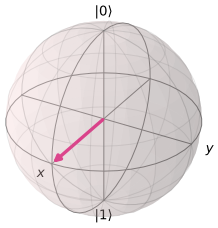


Fig. 3.1: The Bloch sphere. The vector in red denotes the qubit state $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ which is also denoted as $|+\rangle$. The labels x and y represent the Euclidean x and y directions. It is common to use the term (Pauli) z-basis for the standard basis.

3.1.2 Superposition

Formally, just as in any other vector space, we can represent vectors as combinations of basis states. Let the arbitrary state of a qubit be denoted as $|\psi\rangle \in \mathscr{H}_2$. How do we describe this state in terms of the two basis states of $\mathscr{H}_2$? Let us have two complex numbers $c_x \in \mathbb{C}$ with $x \in \{|0\rangle, |1\rangle\}$, which we will call *amplitudes*. These satisfy $\sum_{x\in\{|0\rangle,|1\rangle\}} |c_x|^2 = 1$. The general state $|\psi\rangle$ of the qubit, can be seen as:

$$|\psi\rangle = \sum_{x\in\{|0\rangle,|1\rangle\}} c_x |x\rangle\,. \tag{3.4}$$

The squares of the amplitudes can be thought as the probabilities of finding the qubit in a particular basis state.

In quantum mechanics, observable quantities always are Hermitian operators. Often, one can think of them as Hermitian matrices, that is, complex matrices H with the property $H = H^\dagger$. As a map, an observable simply corresponds to an endomorphism in the Hilbert space, $H : \mathscr{H} \to \mathscr{H}$. An observable H is measured by considering its expectation value when acting on a state $|\psi\rangle$.

$$\langle H\rangle = \langle\psi|H|\psi\rangle = \langle\psi|H\psi\rangle\,. \tag{3.5}$$

One of the most important observables in quantum mechanics is the Hamiltonian of a quantum system. When acting on a state, the Hamiltonian provides the energy of the state. The Hamiltonians play a fundamental role in many quantum simulation algorithms. However, as described above, the Hamiltonian seems to be a very physical concept. Within quantum computation, the role of the Hamiltonian can essentially be assumed by any Hermitian operator. It is customary to call operators that act on qubits as (quantum) *gates* which are usually discussed in the *circuit model of quantum computation*, see Sec. 3.5.

3.1.3 Entanglement

If we imagine that we have several quantum systems, each in some state represented by some state vector, we can combine the separate system into a combined system using the tensor product of vector spaces, $\otimes$. If we imagine that we have one system where the state is given by $|\psi\rangle$ and another where the state is given by $|\varphi\rangle$, the state of the composite system is given by

$$|\psi\rangle \otimes |\varphi\rangle. \tag{3.6}$$

States that can be written in this simple way are called *product states*. Otherwise, we call states entangled. Note that the tensor product does not commute in general.

A system of n qubits (also known as a quantum register) has a state space $\mathbb{C}^{2^n}$, which can be seen as a tensor product $\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$ of the 2-dimensional single-qubit Hilbert spaces, which we denote $(\mathbb{C}^2)^{\otimes n}$. There, each factor corresponds to one qubit. A system of n qubits is associated with the complex inner product $\langle v|w\rangle =$

$\sum_i v_i^* w_i$ and the standard basis $\{|x_1 x_2 \ldots x_n\rangle : x_j \in 0,1\}$. We denote the tensor product of N spaces $\mathbb{C}^2$, together with the inner products and the standard basis, by $\mathscr{B}^{\otimes N}$.

Entanglement is a quantum mechanical phenomenon where the properties of two or more quantum states become correlated. When entangled, the properties of the qubits are linked in such a way that the state of one qubit cannot be described independently of the other(s). Multi-qubit states that cannot be written as separable states are called *entangled states*. Measuring one qubit of an entangled state will instantaneously affect the properties of the other qubits, regardless of the distance between them. This is known as "spooky action at a distance" and is one of the most mysterious and intriguing aspects of quantum mechanics. We will see that entanglement is necessary but not sufficient for quantum speed-up.

3.1.4 BQP

Several models of quantum computation have been devised. Crucially, they do not allow for deciding any problems that are not decidable on a classical computer.

Let a probability threshold be a constant strictly larger than 1/2. A language $L \subset \{0,1\}^n$ is in BPP or BQP, respectively, if and only if its corresponding indicator function $F(x) : \{0,1\}^n \to \{0,1\}$ can be computed probabilistically in polynomial time such that:

1. one starts with register $v \in [0,1]^{2^N}$ or $\mathbb{C}^{2^N}$, for some $N \geq n$ dependent on F, with an initial state $|x, 0^{N-n}\rangle$ consisting of the input padded to length N by zeros;
2. applies a linear stochastic function $U : \mathbb{R}^{2^N} \to \mathbb{R}^{2^N}$ or $U : \mathbb{C}^{2^N} \to \mathbb{C}^{2^N}$ to v, whose matrix representation can be computed in a sparse format by a Turing machine from all-ones input in time polynomial in n
3. obtains a random variable Y, wherein $F(x)$, i.e., a single 0 or 1, is followed by $N-1$ arbitrary subsequent symbols with probability at least as high as the probability threshold, wherein the random variable Y has value y with probability v_y or with probability $|v_y|^2$, for the value v of register.

We know that P $\subseteq$ BPP $\subseteq$ BQP, although the proof is quite non-trivial: one has to establish the power of reversible (classical) circuits and then of the restriction thereof to (classical) permutations.

Exercise 3.1. To get a feel for this, notice that for any Boolean function $f(x) : \{0,1\}^n \to \{0,1\}$, we can construct $\tilde{f} : \{0,1\}^{n+1} \to \{0,1\}^{n+1}$ such that $\tilde{f}(x,y) = (x, y \oplus f(x))$, where $\oplus$ is the XOR operation. Show that this function is reversible. Show how to get the output of $f(x)$.

We know that BQP $\subseteq$ PP. The proof is rather simple. (See previous lecture) Because PP $\subseteq$ PSPACE, i.e., the class of languages that can be recognised by a (classical) Turing machine with a polynomial amount of space, we also know BQP $\subseteq$ PSPACE. Interestingly, there is no material difference between what can be done by a Turing machine with a polynomial amount of space and a quantum Turing machine with a polynomial amount of space. Unfortunately, we do not know much about the relationship between BQP and non-determinstc Turing machines (NP), other than some relativised results.

3.2 An Alternative Model of Fortnow

Following [8], we can get some more insight into the derivation of the result of Arora and Barak.

Let us consider a k-tape extension of a Turing machine:

- a finite, non-empty set Q of objects, representing states
- a subset F of Q, corresponding to "accepting" states, where computation halts
- $q_0 \in Q$, the initial state
- a finite, non-empty set Γ of objects, representing the symbols to be used on any tape
- a partial function $\delta : (Q \setminus F) \times \Gamma^k \to Q \times \Gamma^k \times \{-1,0,1\}^k$, where for a combination of a state and k symbols read from the tape, we get the next state, the symbol to write onto the k tapes, and an instruction to shift the k tapes left (-1), right (+1), or keep in its position (0).

In the "Computation as Matrix Multiplication" view of Fortnow, we consider:

- one-step binary version of the transition function: $\delta' : Q \times \Gamma^k \times Q \times \Gamma^k \to \{0,1\}$, which indicates whether the transition from a configuration c_a to c_b is permitted $c_a, c_b \in C \subseteq (Q \times \Gamma^k)$.
- one-step transition matrix T representing δ' as a $|C| \times |C|$ binary matrix.
- multi-step transition matrix T^r representing the r-step transition function as a $|C| \times |C|$ binary matrix, where $T^r(c_a, c_b) = 1$ if and only if M starting in configuration c_a will be in configuration c_b when run for r steps. $T^r(c_a, c_b)$ is the number of computation paths from c_a to c_b of length r and M accepts if and only if $T^r(c_a, c_b) \geq 1$. For polynomial-time machines, we can obtain the definition of $\#P$ this way.

One can extend this view to probabilistic machines:

- one-step $[0,1]$ version of the transition function: $\delta'' : Q \times \Gamma^k \times Q \times \Gamma^k \rightarrow [0,1]$.
- probabilistic machines use the δ'' with the additional restriction that for any initial state and symbols on the tapes, the values of δ'' for all other arguments sum up to one.
- corresponding one-step transition matrix T and multi-step transition matrices T^r are row and column stochastic.
- Entries of $T^r(c_I, c_A)$ are the probabilities of acceptance by the probabilistic machine.

Let a $0 < \varepsilon < 1/2$ be a constant. A language $L \subset \{0,1\}^n$ is in BPP, if and only if there exists a probabilistic machine as above and a polynomial p such that

- For x in L, we have $T^p(c_I, c_A) \geq 1/2 + \varepsilon$.
- For x not in L, we have $T^p(c_I, c_A) \leq 1/2 - \varepsilon$.

One can extend this view further to weird machines:

- one-step $[-1,1]$ version of the transition function: $\delta''' : Q \times \Gamma^k \times Q \times \Gamma^k \rightarrow [-1,1]$, where the negative values can be intersected with rational numbers.
- weird machines use the δ''' with the additional restriction that the corresponding one-step transition matrix T and multi-step transition matrices T^r are unitary.
- Squared entries of $T^r(c_I, c_A)$ are the probabilities of acceptance by the weird machine.

Let a $0 < \varepsilon < 1/2$ be a constant. A language $L \subset \{0,1\}^n$ is in BQP, if and only if there exists a weird machine as above and a polynomial p such that

- For x in L, we have $(T^p(c_I, c_A))^2 \geq 1/2 + \varepsilon$.
- For x not in L, we have $(T^p(c_I, c_A))^2 \leq 1/2 - \varepsilon$.

[9] have shown that not only rational numbers suffice, but only a few of those suffice.

3.3 Quantum Turing Machines

This view of Fortnow, while based on Turing Machines, is not the Quantum Turing Machine, in some sense the original definition of quantum computation:

[10] defined the quantum Turing machine with one tape for input and output and one tape for intermediate results using:

- still finite set Σ of symbols used for the inputs and outputs
- Hilbert space instead of a finite set Q of objects, representing states, with an accepting subspace
- Hilbert space instead of a finite set Γ representing symbols to be used on the intermediate result tape, with zero-vector instead of a blank symbol,
- partial function δ is now $\delta : \Sigma \times Q \otimes \Gamma \to \Sigma \times Q \otimes \Gamma \times \{L,R\}$, where each automorphism of the Hilbert space is given by a unitary matrix.

The probabilistic element comes in the form of a measurement, which translates the state to the output upon an accepting subspace is reached. Quantum Turing machines and quantum circuits were shown [11] to be equivalent in the sense that they can simulate each other in some distributional sense.

While elegant, the analogy with a Turing Machine may be somewhat confusing. It is important to stress that: There is no branching based on the intermediate results or states. Measurement required by either would collapse the intermediate result or state. The addition of a probabilistic equivalent of branching, known as post-selection, leads to a different complexity class, PostBQP = PP, as shown by [12]. There is no computation in the traditional sense. The state $|\psi(nT)\rangle$ at nth time step is simply $U^n|\psi(0)\rangle$ for some constant unitary operator U. In some sense, one hence wishes to represent all possible solutions in the initial state already. There is no notion a random access memory beyond the qubit register we work with.

3.4 Quantum Circuits

Last but not least, the standard model of quantum computing is known as the quantum circuit model of [13], and it is not too different from the alternative definition of BQP, due to Arora and Barak.

Let a probability threshold be a constant strictly larger than 1/2. Consider $F : \{0,1\}^n \to \{0,1\}^m$ and $N \geq \max\{n,m\}$. There, one:

1. starts with an initial state $|x,0^{N-n}\rangle$ padded to length N.
2. applies a unitary operator $U : \mathscr{B}^{\otimes N} \to \mathscr{B}^{\otimes N}$ (realised by a circuit), which is a composition of multiple unitary operators $U = U_L, U_{L-1}, \cdots U_2, U_1, U_i : \mathscr{B}^{\otimes N} \to \mathscr{B}^{\otimes N}$, where each U_i will be called a gate and L will be the known as the depth of the circuit.
3. obtains $F(x)$ followed by $N-m$ arbitrary subsequent symbols with probability at least as high as a probability threshold.

Let ε be a constant $0 < \varepsilon < 1/2$. A circuit U computes $F : \{0,1\}^* \to \{0,1\}^*$ if for any x we have

$$\sum_z |\langle F(x), z|U\rangle x, 0^{N-n}|^2 \geq 1 - \varepsilon. \tag{3.7}$$

The expression on the left-hand side is, indeed, the probability of getting $F(x)$ padded with with arbitrary z in the measurement of the outcome of U applied to the initial state $|x, 0^{N-n}\rangle$.

A function $\{0,1\}^* \to \{0,1\}^*$ is in BQP, if there exists a deterministic Turing machine M and a polynomial p such that M runs in time $p(|x|)$ and produces a description of a quantum circuit that computes the function.

3.4.1 Building our first quantum circuits

We are now ready to start building quantum circuits. The ingredients will be qubits and unitary operators or gates.

First of all we need to discuss where we will start, i.e., what is the initial state of the system, or the input of the circuit, and how do we prepare that? A simple choice of input vector that is most commonly used is to pick $|0\ldots0\rangle$ as the initial state vector. Given some general initial state, how do we prepare it in the $|0\ldots0\rangle$? Well, one very simple way is found by remembering that measurements will make the system collapse to a given eigenvector of the observable being measured. We can then simply make a measurement of σ_z on each qubit, which will return the results ±1 with some probabilities. If we get $+1$ we know that the qubit is in the state $|0\rangle$ as desired, while if we find -1 we know that it will be in the state $|1\rangle$. Then we simply keep the qubits that are in the $|0\rangle$ state and act with σ_x on the others, since we saw previously that $\sigma_x|1\rangle = |0\rangle$. Now we have our input vector $|\psi\rangle = |0\ldots0\rangle$.

The quantum circuit will then start with a number of qubits in the $|0\rangle$ state and act on this with some number of gates, or unitary operators. The most basic gates are :

- NOT = $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, CNOT = $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$, CCNOT = $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$ and other classical gates. When acting upon two qubits, he controlled-not, or CNOT, gate, acts in the following way:

$$\begin{aligned} |00\rangle &\to |00\rangle, \\ |01\rangle &\to |01\rangle, \\ |10\rangle &\to |1\rangle \otimes \sigma_x|0\rangle = |11\rangle, \\ |11\rangle &\to |1\rangle \otimes \sigma_x|1\rangle = |10\rangle. \end{aligned} \tag{3.8}$$

CCNOT, also known as the Toffoli gate, is a controlled-controlled-gate acting on three qubits. If the two first qubits are in the state $|1\rangle$, then it acts on the third with the NOT gate. Otherwise it does nothing. For classical circuits, the Toffoli gate is important, because similar to the NAND gate, any boolean function can be implemented by using a combination of Toffoli gates. (This property is called universality or functional completeness.)

- The Pauli matrices: σ_x, σ_y and σ_z. These are typically denoted by X, Y and Z in the circuit diagrams. On the Bloch sphere we can visualize them as a π-rotation of the qubit about the corresponding axis.

- The Hadamard gate: $H := \frac{1}{\sqrt{2}}(\sigma_x + \sigma_z)$. It changes $|0\rangle \to |+\rangle$ and $|1\rangle \to |-\rangle$. So it can be seen as a change of basis. On the Bloch sphere we can visualize it as a π-rotation about the axis $\frac{1}{\sqrt{2}}(\hat{x} + \hat{z})$. Hadamard and CNOT make it possible to entangle qubits.

- Phase shift gates changes the relative phase in the expansion in the computational basis by sending $|0\rangle \to |0\rangle$ and $|1\rangle \to e^{i\varphi}|1\rangle$. Common examples are the T gate, with $\varphi = \pi/4$,[1] and the S gate, where $\varphi = \pi/2$. On the Bloch sphere, these gates can be seen as a rotation of φ radians about the $\hat{z}$ axis.

- The controlled-U gate acts on a number of qubits and uses the first as a control. If this is $|0\rangle$ it does nothing, while if it is $|1\rangle$ it acts on the second qubit with the operator U.

One example circuit is Figures 3.2. One important thing to note is that when we read the circuits we read it from left to right, but when we write it down mathematically the gates act in the opposite order.

3.5 Looking beyond the Basics

Let us now summarize a few important results briefly, following papers of [14], [15], and [16]. These concern:

[1] the T gate is confusingly also known as the $\pi/8$ gate,

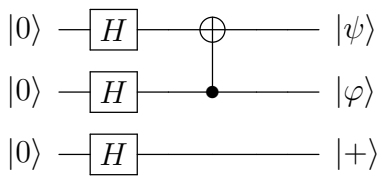


Fig. 3.2: A simple example of a quantum circuit using the H and CNOT gates.

- "role of entanglement" and "interference": are maximally-entangled states[2] sufficient and necessary?
- "universality": what gates are sufficient to implement any unitary matrix in $SU(n)$?
- "weak simulation": can we sample from the distribution on the measurement of a quantum circuit's first qubit in polynomial time using a classical computer?
- "strong simulation": can we compute the probability of measuring 1 on a quantum circuit's first qubit to any given precision in polynomial time a classical computer?

A crucial questions relate to "universality": what gates are sufficient to implement any unitary matrix in $SU(n)$? Traditionally [17], one considers all one-qubit gates plus CNOT. One often implements controlled rotations by a given angle, the phase shift gate, and CNOT, which are sufficient. [15] based on [18] defines computational universal the set of gates that can be used to simulate to within ε error any quantum circuit which uses n qubits and t gates from a strictly universal set with only polylogarithmic overhead in $(n,t,1/\varepsilon)$. Then, she shows that the set of Toffoli and Hadamard gate is computationally universal. Contrast this with the classical computation, where Toffoli on its own is universal.

We clearly need to be able to produce maximally entangled states, using CNOT, Toffoli, or similar. Let us consider the question of what gates produce maximally entangled states from some separable states. One can consider, e.g., using Hadamard and a non-local gate such as CNOT. (Non-local gate is from $SU(4) \setminus SU(2) \otimes SU(2)$.) [14] have shown that the local equivalence classes of two-qubit gates are in one-to-one correspondence with the points in a tetrahedron, except on the base. Using this tetrahedral representation of non-local gates, they have shown that exactly half the

[2] States where if you take a partial trace over one of the subsystems, the resulting state has the maximum entropy.

non-local gates are perfect entanglers. This means that the second half of the non-local gates are imperfect entanglers. While we need a perfect entangler, the role of CNOT is hence not particularly "central".

Having said that, even the role of Hadamard and CNOT is not particularly central either. Hadamard, CNOT, and one particular phase shift gate (phase shift by $\pi/2$) generate a group called the Clifford group. By a non-trivial Gottesman-Knill theorem, the Clifford gates does not make a universal gate set. In particular, the Gottesman-Knill theorem shows that a uniform family of Clifford circuit(s) acting on the computational basis state $|0\rangle^N$ followed by a computational basis measurement, can be simulated efficiently on a classical computer. (Actually, their simulation of Clifford-gate circuits belongs to the complexity class $\oplus\mathbf{L}$ ("parity-L") as classical computation with NOT and CNOT gates, which is not believed to equal to P.) This shows that while maximally entangled states are provably necessary, cf. [19] to disallow efficient classical simulation, they are not sufficient.

In a striking result, [16] shows that circuits implementing unitaries from the Clifford group (Clifford circuits), which may contain many Hadamard gates at different places in the circuit, causing rounds of constructive and destructive interference, are (efficiently) mapped to circuit that do not utilize any interference at all. In particular, to circuits where threre is one round of Hadamard gates applied to a subset of the qubits, followed by a round of "classical gates" such as Toffoli, CNOT, NOT, etc. Let $\mathscr{C}$ be an arbitrary n-qubit Clifford operation. Then there exist: (a) poly-size circuits M_1 and M_2 composed of CNOT, PHASE and CPHASE gates and (b) a tensor product of Hadamard gates and identities $\mathscr{H} = H^S \otimes I$ acting nontrivially on a subset S of the qubits, such that $\mathscr{C} \propto M_2 \mathscr{H} M_1$. Moreover, M_1, M_2 and $\mathscr{H}$ can be determined efficiently.

In real world, all quantum systems interact with the environment. We often use classical distributions over quantum states to reason about such "partially known" quantum states. Let us associate probability p_k to the event of system being in state $|\alpha_k\rangle$. Such a classical distribution is called a "mixed states", as opposed the usual "pure" state. A unitary matrix U acts on a mixture $\{p_k, |\alpha_k\rangle\}$ component-wise $\{p_k, U|\alpha_k\rangle\}$.

In a simple model of an open quantum system due to [20], one assumes:

- single qubit faults: each qubit decoheres independently, or undergoes a fault with probability η per step.
- all operations equal: no decoherence takes place inside the gates.

There, η is referred to as the *decoherence rate*. This is equivalent to a model, where at each timestep, at each qubit i, we can have a fault with a probability η_i, as long as $\sum_i \eta_i = \eta$.

[20] have shown that for models of quantum computing with gates on up to $\log(n)$ qubits, considering the noise model above introduces a delay into the simulation

by a probabilistic machine that is polynomial in the number of qubits and depth of the circuit, for any decoherence rate. [21, 22] suggested that one can correct for a substantial decoherence rate in a Clifford circuit using quantum error correcting codes, at the expense of some overhead in terms of numbers of "physical" qubits.

Chapter 4

Quantum Algorithms I

4.1 What we have seen so far?

Quantum algorithms resemble randomized algorithms, except probability amplitudes are complex. This makes it possible to work with interference. Reversibility of quantum computing (modulo noise) means one has to use the interference to pick out the solution of a functional problem. This solution needs to be represented already in some early superposition, and then see its complex probability amplitude amplified.

4.2 Introduction

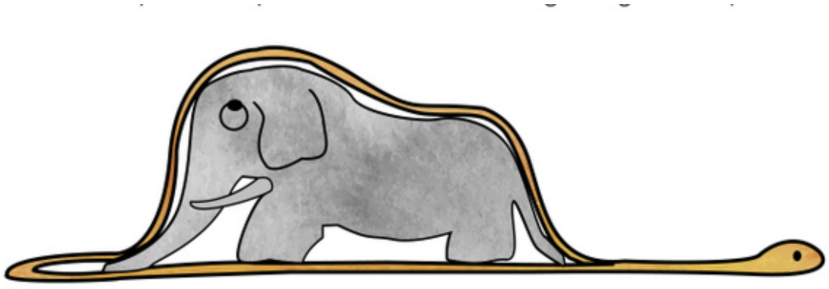


Fig. 4.1: All successful quantum algorithms resemble a boa constructor that ate an elephant.

Quantum algorithms ideally have only a modest amount of input. Typically, one uses an initial state with only a single complex probability amplitude, whose magnitude squares is 1. We can assume without loss of regularity that this corresponds to the basis state of all zeros. Often, people assume that the input is "magically made available" via oracles – but that is often "magical thinking". Often, one loads the input via controlled rotations, one scalar at a time.

Quantum algorithms then construct a maximally-entangled state on q qubits, whose representation requires a 2^q complex probability amplitudes. (In order for the state to be maximally entangled, the complex probability amplitudes have to be equal.) One can see the large entangled state as a root of a large tree, where in the leaves, one applies Hadamard gates to individual qubits, and in the non-leaf nodes, one applies CNOT. Then, applying a single-qubit gate may change all exponentially-many complex probability amplitudes in parallel, at a unit cost in terms of the depth of the quantum circuit.

Finally, the output needs to be very simple. Clearly, the measurement ends up with a basis state, depending on the corresponding complex probability amplitude. In some sense, one may wish the output were only a single complex probability amplitude whose magnitude squares is 1. This is to be seen from the sample complexity of parameter estimation in multivariate distributions: if we expect a non-trivial superposition on the output, we will need very many copies of the circuit and very many collapsing measurements to estimate the output state. Often, one employs some classical post-processing.

4.3 A View from Theoretical Computer Science

4.3.1 Definitions

First, let us introduce some more complexity classes:

In the definition of [23]: FBPP, resp. FQPP is the class of polynomially-bounded relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which there exists a polynomial-time randomized, resp. quantum algorithm A such that for all x for which there exists a y with $(x,y) \in R$ and all $\varepsilon > 0$,

$$\mathbb{P}[(x, A(x, 0^{1/\varepsilon})) \in R] > 1 - \varepsilon,$$

where the probability is over A's outputs.

In the definition of [23]: FP/rpoly, resp. FBQP/qpoly is the class of polynomially-bounded relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which there exists a polynomial-time deterministic classical algorithm A, resp. a polynomial-time quantum algorithm Q,

a polynomial $p(n,m)$, and an infinite list of advice distributions $\{\mathscr{D}_{n,m}\}_{n,m\geq 1}$, where $\mathscr{D}_{n,m}$ is supported on $\binom{}{p(n,m)}$, resp. advice states $\{|\psi_{n,m}\rangle\}_{n,m\geq 1}$, where $|\psi_{n,m}\rangle$ is on $p(n,m)$ qubits, such that for all x for which there exists a y such that $(x,y)\in R$ and all m,

$$\mathbb{P}_{r\sim\mathscr{D}_{n,m}}[(x,A(x,0^m,r))\in R] > 1-\frac{1}{m},$$

resp.

$$\mathbb{P}[(x,Q(x,0^m,|\psi_{n,m}\rangle))\in R] > 1-\frac{1}{m}.$$

If you rely on your circuit calling an oracle, you often typically cannot prove any "usual" separation between complexity classes, e.g., BPP $\neq$ BQP. You can prove only weaker "relativized" results, known as "oracle separations". The strongest results consider unstructured, random oracles. In the classical/quantum random oracle model of [24], a random function H is chosen at the beginning, anyone can classically/quantumly access H, i.e., apply a unitary $|x\rangle|y\rangle \mapsto |x\rangle|y\oplus H(x)\rangle$.

4.3.2 Results

[25] show that relative to an oracle chosen uniformly at random with probability 1 an NP-Complete problem cannot be solved on a quantum Turing machine (QTM) in time $o(2^{n/2})$. [26] show that relative to a random oracle with probability 1, there are NP search problems solvable by BQP machines but not BPP machines. We will revisit the results of Yamakawa and Zhandry later, in the chapter on security, because the same paper also shows that relative to a random oracle with probability 1, there exist functions that are one-way, and even collision resistant, against classical adversaries but are easily inverted quantumly.

From the point of view of Theoretical Computer Science, the situation in decision problems is somewhat dire: We do not know any non-relativized separation between P, BPP, and BQP. (Notice that with non-linear quantum mechanics [27], we could actually solve NP-Complete and #P-Complete problems.)

In functional problems, the situation is somewhat better. In February 2023 [23] have shown FP $\neq$ FBPP, unconditionally, and FBQP/qpoly $\neq$ FBQP/poly. Notice that the FBQP/qpoly refers to a quantum advice, not to an oracle.

4.4 Our first Quantum Algorithm: Deutsch–Jozsa

In the so-called Deutsch–Jozsa problem, we have

- a dimension n
- a black-box function $f(x) : \{0,1\}^n \to \{0,1\}$ that has a rather unusual property: either it is a constant function (there is $y \in \{0,1\}$ such that for all $x \in \{0,1\}^n$, the output is y) or balanced (for precisely 2^{n-1} inputs, the output is 0, and for precisely 2^{n-1} inputs, the output is 1)

The decision version of the problem asks whether the unknown function f is constant. It is clear that classically, one may need to perform $2^{n-1}+1$ oracle calls in the worst-case, but that there would be excellent randomized algorithms. Notice that for $n = 1$, one asks whether $f(0) + f(1) \bmod 2$ is zero.

Let us illustrate the algorithm of [28] for $n = 1$:

1. creates an initial, two-register state $|0\rangle|1\rangle$
2. apply Hadamard transform to both registers: $\frac{1}{2}\sum_{x=0}^{1}|x\rangle(|0\rangle - |1\rangle)$
3. apply the function via the oracle to obtain $\frac{1}{2}\sum_{x=0}^{1}|x\rangle(|0\oplus f(x)\rangle - |1\oplus f(x)\rangle)$
4. apply the Hadamard transform on the first register again:

$$\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}\left[\frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{x\oplus y}|y\rangle\right] = \sum_{y=0}^{1}\left[\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}(-1)^{x\oplus y}\right]|y\rangle$$

5. obtain y by measuring the first register. The probability of measuring $|0\rangle$ is $\left|\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}\right|^2$, which evaluates to which evaluates to 1 for constant functions (constructive interference) and to 0 for balanced functions (destructive interference).

We have used 1 query to the oracle. This can be generalized to any n, without increasing the number of queries!

4.5 First Few Tricks

If the algorithm seems hard to parse, do not despair. There are a few insights that will help us elucidate its workings:

- the Boolean group
- the oracle
- the Hadamard transform
- amplitude amplification.

We will also introduce the phase kickback, which we will need later.

4.5.1 Artihmetics modulo 2

First, let us consider the artihmetics modulo 2 and its relationship to the XOR operation ($\oplus$, "must have one or the other but not both"). In $n = 1$ we have seen

$$(0+1) \bmod 2 = 1 \bmod 2 = 1 = (0 \oplus 1) \text{ and}$$

$$(1+1) \bmod 2 = 2 \bmod 2 = 0 = (1 \oplus 1).$$

Beyond $n = 1$ the binary inner product $\odot$ of bitvectors $x, y \in \{0,1\}^n$ is $x_1 y_1 + \cdots + x_n y_n \bmod 2 = x_1 y_1 \oplus \cdots \oplus x_n y_n$, i.e., essentially counting ones that appear at the corresponding positions in two bitstrings, modulo 2, and thus suggesting whether the count is odd or even. One can formalize this in terms of finite field GF(2).

4.5.2 The Oracle

Next, let us consider the oracle. One assumes that for a function f, there is an oracle U_f that maps $|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$, where $\oplus$ denotes the XOR operation (or addition modulo 2). This can be simplified to $U_f |x\rangle = (-1)^{f(x)} |x\rangle$. Clearly, $|x\rangle |0\rangle \to |x\rangle |f(x)\rangle$. Either way, this is a reversible operation and can be implemented in a unitary.

In the case of Boolean function f on $n = 1$ bits, one can think of this as a CNOT gate controlled by the value of $f(x)$. In the top-left corner, you have I (the identity) for $f(0) = 0$ and σ_x (flip) for $f(0) = 1$. Similarly in the bottom-right corner, you have I (the identity) for $f(1) = 0$ and σ_x (flip) for $f(1) = 1$.

$$U_f = \begin{pmatrix} 1-f(0) & f(0) & 0 & 0 \\ f(0) & 1-f(0) & 0 & 0 \\ 0 & 0 & 1-f(1) & f(1) \\ 0 & 0 & f(1) & 1-f(1) \end{pmatrix}$$

4.5.3 Amplitude Amplification

In a way, the Deutsch–Jozsa algorithm also demonstrates the significance of allowing quantum amplitudes to take both positive and negative values. In the Qiskit Textbook and many other sources, this is illustrated starting with an interference experiment (cf. Young's double-slit interferometer, 1803): a particle can travel from the source to an array of detectors through two slits. Each detector has a probability of observing a particle that depends on the phases of the incoming waves. Same phases increase the probability (constructive interference); very different phases reduce the probability (destructive interference). One can consider 2^n possible paths x and 2^n possible detectors y, both labeled by bitstrings. The phase accumulated at detector x along a path y equals $C(-1)^{f(x)+x\cdot y}$, where $x\cdot y$ is the binary inner product and C is a normalizing constant. The probability of a particle at detector y is $\mathbb{P}(y) = |C\sum_x(-1)^{f(x)+x\cdot y}|^2$ with $C = 2^{-n}$.

Now let us consider the probability of observing an all-zero string y, which is

$$|2^{-n}\sum_x(-1)^{f(x)+x\cdot y}|^2 = |2^{-n}\sum_x(-1)^{f(x)+0}|^2,$$

in the two cases of the promise problem:

- if the $f(x) = c$, then the probability is $|2^{-n}\sum_x(-1)^c|^2 = 1$
- if the $f(x)$ is balanced, then the probability is zero $|2^{-n}\sum_x(-1)^{f(x)}|^2 = 0$, because the alternating sign will lead to a cancellation of the terms.

Exercise 4.1. Plot a diagram of the double-slit experiment and the 2^n detectors and the inference pattern for some $n \geq 8$.

4.5.4 The Hadamard Transform

We have seen the Hadamard gate:

$$H(|0\rangle) = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle =: |+\rangle \tag{4.1}$$

$$H(|1\rangle) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle =: |-\rangle \tag{4.2}$$

$$H(|+\rangle) = H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}\Big(|0\rangle + |1\rangle\Big) + \frac{1}{2}\Big(|0\rangle - |1\rangle\Big) = |0\rangle \tag{4.3}$$

$$H(|-\rangle) = H\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}\Big(|0\rangle + |1\rangle\Big) - \frac{1}{2}\Big(|0\rangle - |1\rangle\Big) = |1\rangle \tag{4.4}$$

without really understanding it.

First, notice that for an n-qubit state $|k\rangle$, the application of Hadamards qubit-wise yields:

$$H^{\otimes n}|k\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (-1)^{k\odot j} |j\rangle ,$$

where $j \odot k = j_1k_1 \oplus j_2k_2 \oplus \cdots \oplus j_nk_n$ and $\oplus$ is XOR as above. Second, this is rooted in a non-trivial fact that the Hadamard transform is the Fourier transform on the Boolean group $(\mathbb{Z}/2\mathbb{Z})^n$. (If this sounds difficult, notice that GF(2)is isomorphic to the quotient ring of the ring of integers $\mathbb{Z}$ by the ideal $2\mathbb{Z}$ of all even numbers, GF(2) = $\mathbb{Z}/2\mathbb{Z}$.)

In the example of the second application of Hadamard in Deutsch–Jozsa for $n = 1$, we obtain:

$$\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}\left[\frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{x\oplus y}|y\rangle\right] = \sum_{y=0}^{1}\left[\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}(-1)^{x\oplus y}\right]|y\rangle .$$

More broadly, the Hadamard maps $|x\rangle$ to $2^{-n/2}\sum_y(-1)^{x\odot y}|y\rangle$. For our state $2^{-n/2}\sum_x(-1)^{f(x)}|x\rangle$, this will amount to $2^{-n}\sum_x(-1)^{f(x)+x\odot y}|y\rangle$, just as in the interference experiment.

Exercise 4.2. Write down the Hadamard gate on $n = 3$.

4.5.5 Phase Kickback

In the previous chapter, we have seen the phase factor ("global phase", "global gauge"), whereby quantum states $|\psi\rangle$ and $e^{i\alpha}|\psi\rangle$ are indistinguishable by measurement with any linear operator ϕ in the sense of $|\langle\phi\rangle\psi|^2 = |e^{i\alpha}\langle\phi\rangle\psi|^2 = |\langle\phi\rangle\psi|^2$. (A phase-shift gate $P(\alpha) = e^{i\alpha}I$ multiplies any state by a global phase α.) If we apply a control-U gate to $|\psi\rangle$, where $|\psi\rangle$ is an eigenstate of U, then

- $|\psi\rangle$ is unchanged
- the phase $|0\rangle\langle 0| + e^{i\alpha}|1\rangle\langle 1|$ is transferred to the input state $|\psi\rangle$ of the control qubit.

Let us see the *phase kickback* in action. Let us consider a single-qubit gate U and its eigenstate $|\psi\rangle$:

$$U|\psi\rangle = e^{i\phi}|\psi\rangle .$$

We wish to estimate ϕ up to the period (2π). This is possible with an extra ("ancilla") qubit, one Hadamard gate on the ancilla qubit, and a CNOT gate. Notably, after

applying the CNOT controlled by the ancilla qubit, the phase of the ancilla will be ϕ:

$$XH \otimes I\,|0\rangle\,|\psi\rangle = \frac{|0\rangle + e^{i\phi}\,|1\rangle}{\sqrt{2}}\,|\psi\rangle\,,$$

where we have used the fact that CNOT = $[I0; 0X]$.

In the two-qubit Deutsch algorithm above, the first qubit acts as an ancilla qubit, and the controlled qubit is in the eigenstate of the NOT gate with eigenvalue -1. $\phi = \pi$. Thus, we get $\frac{1}{\sqrt{2}}|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle$. The phase kick-back is either 0 or π, which can be distinguished by measuring σ_x, or by applying Hadamard gate again and then measuring in the computational basis. This will be important in the following discussion of Simon's algorithm.

4.6 The Proof (Sketch) of our first Oracle Separation

Let us illustrate the first, historically, and most commonly taught "oracle separation" between BPP and BQP on Simon's problem. This is not a problem, which would be useful on its own, more akin a "guessing game". It uses a highly structured, "periodic" oracle. We will see, however, that the crucial concept of "period finding" underlies the famous Shor factoring algorithm. (Daniel Simon actually recalls[1] that Shor developed his factoring algorithm having seen a preprint of his.)

In the so-called Simon's problem, we have

- a dimension n
- a black-box function $f(x) : \{0,1\}^n \to \{0,1\}^n$ that has a rather unusual property: for all $x, y \in \{0,1\}^n$, $f(x) = f(y)$ if any only if $x \oplus y \in \{0^n, s\}$ for some unknown secret $s \in \{0,1\}^n$,

where $\oplus$ denotes the elementwise XOR operation. Notice that $x \oplus y = 0^n$, if and only if $a = b$. Thus,

- either the secret is $s = 0^n$ and the function is a bijection ("one-to-one", invertible)
- or the secret is $s \neq 0^n$ and the function is not a bijection, but rather "two-to-one".

The decision version of the problem asks whether the unknown function f is a bijection (and thus whether $s = 0^n$).

Exercise 4.3. To get a feel for this, pick an $f(x) : \{0,1\}^3 \to \{0,1\}^3$. Ask your neighbour to guess whether it's a bijection. How many queries did he need?

[1] https://aws.amazon.com/blogs/quantum-computing/simons-algorithm/

Notice that there is *no input*. Hence, it is impossible to reason about the description complexity of the input. Let us measure the complexity of (a classical or quantum algorithm) by the number of evaluations of f at distinct values (x) it requires. (This is also known as the number of oracle queries or oracle complexity.) In a deterministic Turing machine, one may try 2^n inputs one by one until one obtains two inputs producing the same output, or decides that no two match. In a probabilistic machine, one may try to sample the 2^n inputs randomly, and as long as no two match, suggest that the function is a bijection, with an ever higher probability.

The quantum algorithm for solving the problem is similar to the randomized algorithm. Repeatedly, for each sample, we perform the following steps:

1. creates an initial, two-register state $|0\rangle^{\otimes n}|0\rangle^{\otimes n}$
2. apply Hadamard transform on the first register: $\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle|0\rangle^{\otimes n}$
3. apply the function via the oracle to obtain $\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle|f(k)\rangle$
4. apply the Hadamard transform on the first register again:

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}\left[\frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}(-1)^{j\odot k}|j\rangle\right]|f(k)\rangle=\sum_{j=0}^{2^n-1}|j\rangle\left[\frac{1}{2^n}\sum_{k=0}^{2^n-1}(-1)^{j\odot k}|f(k)\rangle\right]$$

5. obtain y by measuring the first register. The probability of measuring $|j\rangle$ is $\left|\frac{1}{2^n}\sum_{k=0}^{2^n-1}(-1)^{j\odot k}|f(k)\rangle\right|^2$.

Then, we classically solve a system of equations given by the samples to obtain s. (Each sample satisfies $ys=0$.) If $s=0^n$, return YES.

4.7 Going beyond our first Oracle Separation

Shor's factoring algorihm uses a non-trivial initial preprocessing, but then we perform the following steps:

1. creates an initial, Q-qubit state $|0\rangle^{\otimes Q}$
2. apply Hadamard transform on it: $\frac{1}{\sqrt{Q}}\sum_{k=0}^{Q-1}|x\rangle$
3. apply the function $f(x)=a^x \bmod N$ using $U_f|x,0^n\rangle=|x,f(x)\rangle$ to obtain

$$U_f\frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,0^n\rangle=\frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,f(x)\rangle$$

such that the value we are looking for is in the phase of

4. apply the quantum Fourier transform: $\frac{1}{Q}\sum_{x=0}^{Q-1}\sum_{y=0}^{Q-1}\omega^{xy}|y, f(x)\rangle$
5. obtain y by measuring the first register. The probability of measuring $|y, z\rangle$ is

$$\frac{1}{Q^2}\frac{\sin^2(\frac{\pi mry}{Q})}{\sin^2(\frac{\pi ry}{Q})}$$

.

Then, we apply classical post-processing.

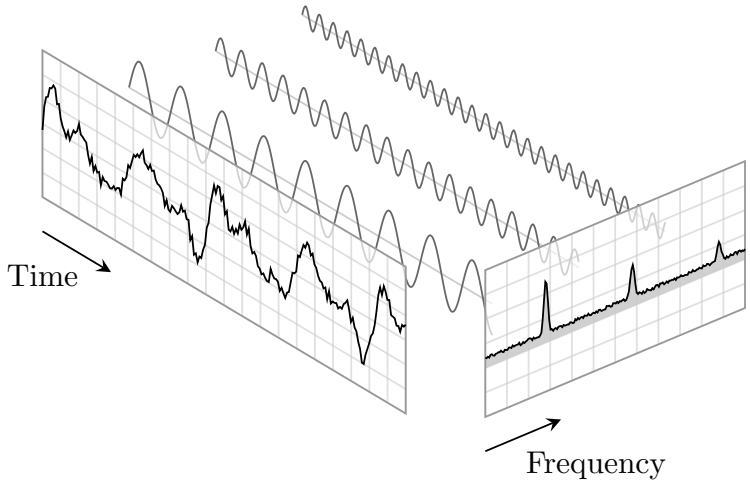
If you felt that there is common pattern across these algorithms, you are right. The quantum Fourier transform is, in some sense, just a more efficient way of measuring the phase. There are, actually, very few "paradigms" in the design of quantum algorithms, and some of the steps (equal superposition, some operation thereupon) are necessary. One may consider, for example [29], amplitude amplification (algorithms above and Grover) with or without quantum Fourier transform, Harrow-Hassidim-Lloyd (HHL), and quantum signal processing (QSP).

Chapter 5

Harmonic Analysis 101

In this lecture, we introduce the quantum Fourier transform, which is $O((\log N)^2)$, i.e., exponentially faster than the classical fast Fourier transform in $O(N \log N)$. As is perhaps familiar, the simple intuition for the classical Fourier transform is basically as a change of basis, or perhaps a duality transformation. Typically we think of it as taking us from the original function domain to its corresponding frequency domain, where certain properties, such as which frequencies are present in a signal, are easier to analyze. This is directly translated to the quantum Fourier transform, which is a change of basis from the computational basis to the Fourier basis. For example, as we will see below, in the simplest case of one qubit the quantum Fourier transform is simply the Hadamard gate. Which we have already seen is a change of basis from the computational basis to the Hadamard basis $|\pm\rangle$. Before we get into the quantum

Fourier transform, we will begin with the classical case, and in particular try to explain some harmonic analysis in general.



Harmonic analysis as we need it requires discrete samples and finite fields. The corresponding, perhaps seemingly obscure parts of harmonic analysis have also led to classical breakthroughs, such as multiplication of n-bit integers in time $O(n \log n)$ [30] and multiplication of polynomials over finite fields [31] in the same time. The idea is that we can think of the Fourier transform in a very general way as a function on a group or over a finite field. The only difference between things like the full continuous Fourier transform, the discrete-time Fourier transform and the discrete Fourier transform are then simply the choice of group. For a very nice introduction to Harmonic analysis on finite groups, see [32].

5.1 Discrete Fourier Transform

The discrete Fourier transform (DFT) maps an N-vector $\mathbf{x}$ of complex numbers to an N-vector $\mathbf{X}$ of complex numbers:

$$X_k = \sum_{j=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}}, \tag{5.1}$$

up to a normalization $\frac{1}{\sqrt{N}}$. (This is sometimes called the analysis formula.) Let us assume $N = 2^n$ throughout, where n is a constant.

One way to think of the N-vector $\mathbf{x}$ is to see those as samples of a periodic function with period T, i.e., $f(t) = f(t+T)$. In particular, one would sample f uniformly at points $j\Delta t$, where $\Delta T = T/N$ and $j = 0, 1, ..., N-1$.

Alternatively, one could see discrete Fourier transform as a function on a finite cyclic group $G_q = \{1, g, g^2, \ldots, g^{q-1}\} \cong \mathbb{Z}/q\mathbb{Z}$. A simple representation of the elements of this group is as q^{th} roots of unity, $g^n = \exp\left(\frac{2\pi i n}{q}\right)$, with $n = 0, \ldots, q-1$, where the group multiplication is simply ordinary complex multiplication. By visualizing this group action on the unit circle we can see that it is the rotational symmetry group of the q-polygon.

For any group G, and especially for cyclic groups $\mathbb{Z}_q$, it may be tempting to identify the group with its elements $g \in G$ and consider $f(g)$ only, or to identify the cyclic groups $\mathbb{Z}_q$ with the set $\{0, \ldots, q-1\}$ and modulo q addition. One could do much better, however, if one considers the group's symmetries.

Alternatively, one could see the analysis formula (5.1) as a matrix equation $\mathbf{X} = \mathbf{F}\mathbf{x}$. Thus, a discrete Fourier transform can be expressed as a so-called Vandermonde matrix (Sylvester, 1867),

$$\mathbf{F} = \frac{1}{\sqrt{N}} \begin{bmatrix} \omega_N^{0\cdot 0} & \omega_N^{0\cdot 1} & \cdots & \omega_N^{0\cdot (N-1)} \\ \omega_N^{1\cdot 0} & \omega_N^{1\cdot 1} & \cdots & \omega_N^{1\cdot (N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_N^{(N-1)\cdot 0} & \omega_N^{(N-1)\cdot 1} & \cdots & \omega_N^{(N-1)\cdot (N-1)} \end{bmatrix} \tag{5.2}$$

where $\omega_N^{m\cdot n} = e^{-i2\pi mn/N}$ and the mn is the usual product of the integers.

Notice that:

- because ω depends only on the product of frequency m, and position n, the DFT $\mathbf{F}$ is symmetric. Notice that it is also unitary: $\mathbf{F}^{-1} = \mathbf{F}^*$ and $|\det(\mathbf{F})| = 1$.
- $\mathbf{X}$ is the inner product of $\mathbf{x}$ with the m-th row of $\mathbf{F}$. Conversely, $\mathbf{f}$ is a linear combination of the columns of $\mathbf{F}$, where the mth column is weighted by X_m.
- the vectors $u_m = \left[e^{\frac{i2\pi mn}{N}} \;\middle|\; n = 0, 1, \ldots, N-1 \right]^{\mathsf{T}}$ form an orthogonal basis over the set of N-dimensional complex vectors.
- $\mathbf{F}^2$ reverses the input, while $\mathbf{F}^4 = \mathbf{I}$. The eigenvalues satisfy: $\lambda^4 = 1$ and thus are the fourth roots of unity: $+1, -1, +i, or -i$.

5.1.1 The Hadamard Transform

We can define the 1×1 Hadamard transform $H_0 = 1$ as the identity, and then define H_m for $m > 0$ by:

$$H_m = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix}.$$

Other than the normalization, the Hadamard matrices are made up of 1 and -1. Notice that Hadamard

$$H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

is a discrete Fourier transform; indeed, we have $\omega_1^{0\cdot 0} = \omega_1^{0\cdot 1} = \omega_1^{1\cdot 0} = e^0 = +1$ and $\omega_1^{1\cdot 1} = e^{-i\pi} = -1$. As a further example, the next Hadamard matrix is

$$H_2 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

Note that this is not a DFT as defined by (5.2).

Notice that classically, we can compute the fast Hadamard transform algorithm in $O(n \log n)$ while performing only sign-flips. Quantumly, the Hadamard transform can be computed in time $O(1)$, in many commonly used gate sets.

5.1.2 The z-Transform

If you know the z-transform, notice that the X_k can also be seen as evaluation of the z-transform $X(z) = \sum_{j=0}^{N-1} x_j z^{-j}$ at points ω_N^{-j}, i.e., $X_j = X(z)_{z=\omega_N^{-j}}$.

5.1.3 Examples of Discrete Fourier Transform

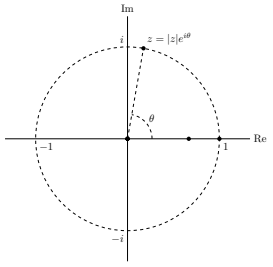
Let us see:

$$\mathbf{F}_0 = H_0 = 1$$

$$\mathbf{F}_1 = H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\mathbf{F}_2 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega_3^{1\cdot 1} & \omega_3^{1\cdot 2} \\ 1 & \omega_3^{2\cdot 1} & \omega_3^{2\cdot 2} \end{bmatrix}$$

While the omega notation may obscure the nature of the DFT, see that the column correspond to passes along the unit circle, clockwise, expressed in the corresponding complex number (e.g., 1, $-i$, -1, i for $\mathbf{F}_3$) at varying frequency (e.g., 0, 1, 2, 3 for $\mathbf{F}_3$): $\mathbf{F}_3 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$.



Exercise 5.1. Visualise $\mathbf{F}_3$, $\mathbf{F}_4$ on the unit circle.

Let us further consider some simple examples:

$$\frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.3}$$

$$\frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 1 \\ -i \\ -1 \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{bmatrix} \tag{5.4}$$

$$\frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \tag{5.5}$$

5.2 Fast Fourier Transform

5.2.1 The Many Fast Fourier Transforms

A straightforward implementation of DFT as a matrix-vector product requires $O(N^2)$ operations. In the so-called fast Fourier transform (Cooley and Tukey, 1965), one requires only $O(N \log_2 N) = O(2^n n)$ operations. There are a number of variants, all based on the divide-and-conquer approach. As we assume $N = 2^n$, we will present a variant known as the radix-2 decimation in time (DIT) algorithm.

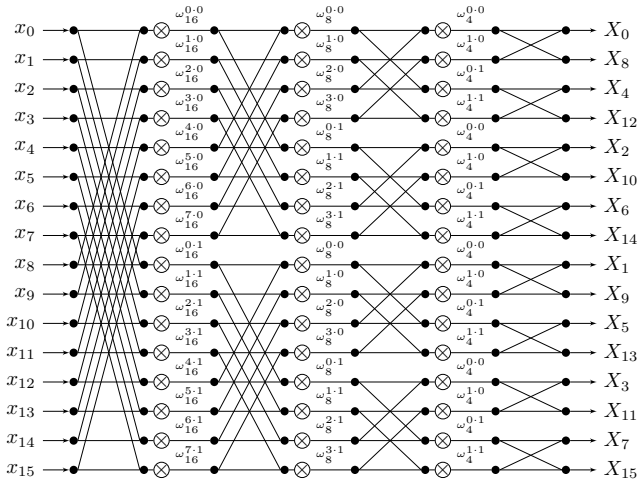
This speedup is achieved by this variant of the divide-and-conquer approach, where we consider subsets of the initial sequence, take the DFT of these subsequences, and reconstruct the DFT of the original sequence from the results on the subsequences. One option is based on the following insight:

$$X_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}} \tag{5.6}$$

$$= \frac{1}{\sqrt{N}} \left(\sum_{\text{even } j} x_j \cdot e^{\frac{2\pi jki}{N}} + \sum_{\text{odd } j} x_j \cdot e^{\frac{2\pi (j-1)ki}{N}} \right) \tag{5.7}$$

$$= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{N/2}} \sum_{\text{even } j} x_j \cdot e^{\frac{2\pi (j/2)ki}{N/2}} \right) + \left(\frac{1}{\sqrt{N/2}} \sum_{\text{odd } j} x_j \cdot e^{\frac{2\pi (j-1)/2ki}{N/2}} \right) \tag{5.8}$$

The divide-and-conquer approach can be illustrated on $N = 16$ with the following cartoon.



If you know the z-transform, you should see that $X(z) = \sum_{j=0}^{N-1} x_j z^{-j} = \sum_{l=0}^{r-1} \sum_{j \in \mathbf{I}_l} x_j z^{-j}$ for some partition $\mathbf{I}$ of $\{0,1,\ldots,N-1\}$ into r subsets, and that one can also normalise the terms. This way, one can define a variety of recursions similar to the one above, as long as the subset are chosen to be similar to the initial sequence in terms of their periodicity. This is very nicely explained in [33].

5.2.2 Fast Fourier Transform as a Factorization

Alternatively, [34] sees the Fast Fourier Transform as a certain matrix factorization. This is both important to understand FFT, but also to understand the QFT later. In particular, the $2^n \times 2^n$ DFT matrix $\mathbf{F}_n$ can be factored as:

$$\mathbf{F} = \mathbf{P}_n \mathbf{A}_n^{(0)} \mathbf{A}_n^{(1)} \cdots \mathbf{A}_n^{(n-1)}, \tag{5.9}$$

where,

- $\mathbf{P}_n$ is some permutation matrix
- $\mathbf{A}_n^{(k)} = \mathbf{I}_{n-k-1} \otimes \mathbf{B}_{k+1}$,
- $\mathbf{B}_{k+1} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{I}_k & \mathbf{I}_k \\ \blacksquare_k & -\blacksquare_k \end{bmatrix}$ with $\mathbf{I}_k$ the $k \times k$ identity matrix
- $\blacksquare_k := \blacksquare_{2^k}$ is a $2^k \times 2^k$ diagonal matrix:

$$\blacksquare_{2^k} \coloneqq \begin{bmatrix} \omega_{2^{k+1}}^0 & & & \\ & \omega_{2^{k+1}}^1 & & \\ & & \ddots & \\ & & & \omega_{2^{k+1}}^{2^k-1} \end{bmatrix}, \tag{5.10}$$

where $\omega_{2^{k+1}}$ is $e^{\frac{-2\pi i}{2^{k+1}}}$ as before.

Notice that each matrix $\mathbf{A}_n^{(i)}$ has two non-zero elements on every row. Consequently, the matrix-vector product $\mathbf{A}_n^{(i)}\mathbf{x}$ can be computed in $O(2^n)$ operations, resulting in $O(2^n n)$ operations, when one includes the permutation.

5.3 Quantum Fourier Transform

In general, $\mathbf{F}$ is an $N \times N$ unitary matrix, and thus we can implement it on a quantum computer as an n-qubit unitary for $N = 2^n$. As such, it maps an N-dimensional vector of amplitudes to an N-dimensional vector of amplitudes. This is called the quantum Fourier transform (QFT). [35, 36] presented the first polynomial, $O(n^2)$ quantum algorithms for QFT over certain finite fields and arbitrary finite Abelian groups, respectively. This is exponentially faster than the classical fast Fourier transform, which takes $O(N \log N)$ steps.

Recall that the DFT is:

$$X_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}}.$$

In contrast, the QFT on an orthonormal basis $|0\rangle, |1\rangle, \ldots, |N-1\rangle$ is a linear operator:

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi jki}{N}} |k\rangle.$$

An alternative representation of the QFT utilizes the *product form*:

$$|j_1, j_2, \ldots, j_n\rangle \to \frac{\left(|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle\right)\left(|0\rangle + e^{2\pi i\, 0.j_{n-1} j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i\, 0.j_1 j_2 \cdots j_n} |1\rangle\right)}{2^{n/2}},$$

where $|j_1, j_2, \ldots, j_n\rangle$ is a binary representation of a basis state j and $0.j_1 j_2 \cdots j_n$ is a notation for binary fraction $j_1/2 + j_2/4 \cdots j_n/2^{n+1}$. This is actually easy enough to derive:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}} |k\rangle \tag{5.11}$$

$$\frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi jki}{2^n}} |k\rangle \tag{5.12}$$

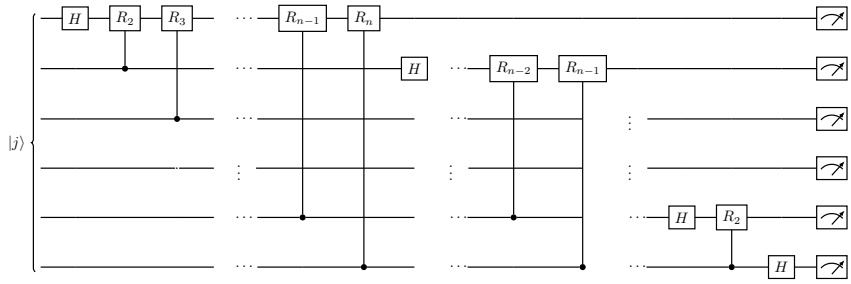
$$\frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi j(\sum_{l=1}^{n} k_l 2^{-l})i} |k_1 k_2 \ldots k_n\rangle \tag{5.13}$$

$$\frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi j k_l 2^{-l} i} |k_l\rangle \tag{5.14}$$

$$\frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi j k_l 2^{-l} i} |k_l\rangle \right] \tag{5.15}$$

$$\frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi j 2^{-l} i} |1\rangle \right] \tag{5.16}$$

A simplistic illustration of the quantum circuit for QFT, omitting swaps at the end



and normalization:
Its derivation is very nicely given in [34], using the framework of matrix decompositions above. Instead of a proper derivation, let us consider the workings of the circuit step by step. The key to its understanding is the phase kickback, which we have seen earlier.

Applying the Hadamard gate to the first qubit of the input state $|j_1 \ldots j_n\rangle$ gives

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i \, 0.j_1} |1\rangle \right) |j_2 \ldots j_n\rangle \tag{5.17}$$

since $e^{2\pi i \, 0.j_1}$ equals $+1$ when $j_1 = 0$ and equals -1 when $j_1 = 1$. We define a unitary gate R_k as

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} \tag{5.18}$$

The controlled-R_2 gate applied on the first qubit, conditional on j_2, now gives

$$\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2}|1\rangle\right)|j_2 \dots j_n\rangle \tag{5.19}$$

Applying further the controlled-R_3, R_4 ... R_n gates, conditional on j_3, j_4 etc., we get

$$\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2 \dots j_n}|1\rangle\right)|j_2 \dots j_n\rangle \tag{5.20}$$

Next we perform a similar procedure onto the second qubit. The Hadamard gate produces the state

$$\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i\ 0.j_2}|1\rangle\right)|j_3 \dots j_n\rangle \tag{5.21}$$

and the controlled-R_2 through R_{n-1} gates yield the state

$$\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i\ 0.j_2 \dots j_n}|1\rangle\right)|j_3 \dots j_n\rangle \tag{5.22}$$

We continue this procedure for each qubit, obtaining a final state

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i\ 0.j_2 \dots j_n}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i\ 0.j_n}|1\rangle\right) \tag{5.23}$$

Eventually, we use the *SWAP* operations to reverse the order of the qubits to obtain the state in the desired product form

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i\ 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i\ 0.j_{n-1} j_n}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2 \dots j_n}|1\rangle\right) \tag{5.24}$$

5.3.1 Even Faster QFT

Above, we have clearly used at most $O(n^2)$ gates. [37], [38], and others improved this to $O(n \log n)$ depth, if one allows for some error. This is based on the realization that R_s for $s \ll \log n$ are very close to the identity and can be omitted.

Part II
Beyond the Basics

Chapter 5
Grover Search and Dynamic Programming

So far, we have seen examples of quantum algorithms with an exponential speed-up, but only for problems that are *not* NP-Hard. For NP-Hard problems, we know only algorithms with quadratic speed-up so far, and even that is disputed [39, 40, 41]. In this chapter, we will explain these in a very general framework of [42].

5.1 Grover Algorithm

In the problem of [43], we have

- a dimension n
- a black-box function $f(x) : \{0,1\}^n \to \{0,1\}$ parametrized by a secret n-bit string j, which returns 1 if $x = j$ and 0 otherwise.

The functional version of the problem asks what is the unknown w. It is clear that classically, one may need to perform 2^n oracle calls in the worst-case, and that randomized algorithms would not help much. Notice that $N = 2^n$ is sometimes referred to as the "library size" we are searching.

The black-box function is usually thought of as an *oracle operator* U_w such that for states $|j\rangle$ in the computational basis

$$U_w |j\rangle = (-1)^{f(j)} |j\rangle = \mathbf{I} - 2 |w\rangle \langle w| = \begin{cases} -|j\rangle, & \text{if } j = w \\ +|j\rangle, & \text{if } j \neq w \end{cases} \tag{5.1}$$

This is sometimes known as the $\pm$-oracle or phase oracle. One can generalize this to the situation where there are multiple secrets.

We will also use a generalization, which is known as the *diffusion operator* or inversion for an arbitrary state $|s\rangle$:

$$U_s = 2\,|s\rangle\,\langle s| - \mathbf{I} \tag{5.2}$$

Let us have a bit of a geometric detour: any state $|\phi\rangle$ can be uniquely expressed as $|\phi\rangle = \alpha\,|\psi\rangle + \beta\,|\psi^{\perp}\rangle$, where $|\psi^{\perp}\rangle$ is orthogonal to $|\psi\rangle$. Then:

$$U_s\,|\phi\rangle = -\alpha\,|\psi\rangle + \beta\,|\psi^{\perp}\rangle \tag{5.3}$$

that is, amplitudes of basis states orthogonal to $|\psi\rangle$ are left unchanged, while signs of amplitudes of the basis state $|\psi\rangle$ are flipped. Furthermore, for any state ϕ, U_ψ, preserves the subspace spanned by $|\phi\rangle$ and $|\psi\rangle$.

Grover's algorithm performs the following steps:

1. creates an initial, n-qubit state $|0\rangle^{\otimes n}$
2. apply Hadamard transform on it to obtain the uniform superposition $\frac{1}{\sqrt{n}}\sum_{k=0}^{n-1}|x\rangle$
3. apply the function oracle operator U_w and the diffusion operator U_s repeatedly, q times.
4. obtain $\hat{w}$ by measuring the n-qubit register. With probability $\sin^2((q+\frac{1}{2})\theta)$ for some θ depending on $\frac{1}{\sqrt{N}}$, estimate $\hat{w}$ will be the correct $f(\hat{w}) = 1$. Otherwise, we repeat.

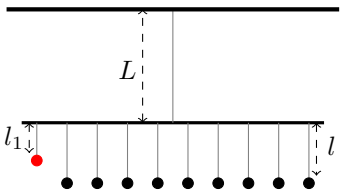
Ideally [44], one considers $q \approx \frac{\pi}{4}2^{n/2}$. If the Grover iteration U_sU_w could be implemented in unit time (a big if!), this would correspond to $O(2^{n/2}) = O(\sqrt{2^n}) = O(\sqrt{N})$ algorithm and quadratic speed-up compared to the linear search in time $O(2^n) = O(N)$.

The Grover iteration has a number of appealing interpretations: Perhaps the most physical is due to [45]. Recall the discussion of the oscillators from the second lecture. The oscillator could describe a weight (or bob) suspended from a pivot on a (massless) cord such that the bob can swing freely. Now, consider N oscillators, one of which has a slightly shorter cord, and hence a different frequency. We seek to find the one with the shorter cord. We could check the frequency of the N oscillators one by one. Alternatively, we can consider a compound pendulum.

To this end, we consider a system where the N oscillators are suspended from a *support pendulum*. We use the following notation:

- The length, mass and displacement coordinate for the support pendulum are denoted L, M, X;
- the pendulum we aim to identify has length, mass and displacement $l_1, \frac{m_1}{N}, x_1$;
- the remaining $N-1$ oscillators have length, mass and displacements $l, \frac{m}{N}, x_j$ for $j = 2, \ldots, N$.

The setup thus looks something like the following:



The Lagrangian (kinetic energy *minus* potential energy)[1] is then:

$$\begin{aligned}\frac{1}{2}[M\dot{X}^2 - KX^2 + \frac{1}{N}(m_1\dot{x_1}^2 - k_1(x_1 - X)^2) + \frac{1}{N}\sum_{j=2}^{N}(m\dot{x_j}^2 - k(x_j - X)^2)] \\ K \equiv (M + \frac{m}{N})\frac{g}{L}, \quad k_j \equiv m_j\frac{g}{l_j},\end{aligned} \tag{5.4}$$

where

- g is the acceleration due to gravity;
- K, k_1 and k are the spring, or stiffness, constants, of the corresponding oscillators. For a simple, uncoupled, harmonic oscillator with mass m, this is related to the frequencies ω through $\omega = \sqrt{\frac{k}{m}}$.

Through a simple change of variables, one obtains:[2]

$$L_{red} \approx \frac{1}{2}[M\dot{X}^2 - KX^2 + m_1\dot{\xi}^2 - k_1(\xi - \frac{1}{\sqrt{N}}X)^2 + m\dot{\bar{x}}^2 - k(\bar{x} - X)^2]. \tag{5.5}$$

Note that this has 3 degrees of freedom, two that are strongly coupled X and $\bar{x}$, while the third, ξ, is weakly coupled due to the $1/\sqrt{N}$ factor. Solving first the $X, \bar{x}$ system gives us two modes with frequencies ω_a and ω_b. The natural frequency of the ξ degree of freedom that corresponds to the special pendulum is approximately $\omega_1 = \sqrt{\frac{k_1}{m_1}}$. If ω_1 is close to either ω_a or ω_b, there will be resonant transfer of energy between the two weakly coupled systems. In $O(\sqrt{N})$ cycles, one should be able to identify the correct pendulum by having amplified its energy. If we instead had n shorter cords, it would take $O(\sqrt{N/n})$ cycles.

Imagine that one starts by a single push to the support pendulum and can change parameters of any pendulum and then observe their frequency with a finite precision that is independent of N. By bisection, we can adjust the cords of 1/2 of the pendula,

[1] remember that the Hamiltonian is the kinetic energy + potential energy

[2] Essentially, the change of variables is to the center-of-mass frame for the $j = 2, \ldots, N$ pendulums, and $\xi \propto x_1$ is simply a normalization. Finally, we ignore some $\mathcal{O}(1/N)$ terms.

1/4 of the pendula, etc., until we identify the one pendulum. This would have a runtime of $O(\sqrt{N}\log N)$.

The diffusion operator should be viewed as a quantum amplitude amplification procedure, with the aim to increase the probability amplitude of the target state. Following [46, 47], one could consider $|\phi\rangle = A|0^n\rangle = \sum_i \alpha_i |i\rangle$ and some partition:

$$|\phi\rangle = \sum_{i\in\text{good}} \alpha_i |i\rangle + \sum_{i\in\text{bad}} \alpha_i |i\rangle ,$$

with $\mathbb{P}(\text{good}) = \sum_{i\in\text{good}} |\alpha_i|^2$. Then,

$$|\phi\rangle = \sqrt{\mathbb{P}(\text{good})}\,|\phi_{\text{good}}\rangle + \sqrt{1-\mathbb{P}(\text{good})}\,|\phi_{\text{bad}}\rangle = \sin(\theta)\,|\phi_{\text{good}}\rangle + \cos(\theta)\,|\phi_{\text{bad}}\rangle$$

with $\sin^2(\theta) = \mathbb{P}(\text{good})$ and $\sin^2(\theta)+\cos^2(\theta) = 1$. The state $|\phi\rangle$ is thus orthogonal to $|\phi^\perp\rangle = \cos(\theta)\,|\phi_{\text{good}}\rangle - \sin(\theta)\,|\phi_{\text{bad}}\rangle$. $\{|\phi_{\text{good}}\rangle, |\phi_{\text{bad}}\rangle\}$ and $\{|\phi\rangle, |\phi^\perp\rangle\}$ are thus two orthonormal bases in a 2-dimensional subspace. One obtains

$$U_w\left(\sin(\theta)\,|\phi_{\text{good}}\rangle + \cos(\theta)\,|\phi_{\text{bad}}\rangle\right) = -\sin(\theta)\,|\phi_{\text{good}}\rangle + \cos(\theta)\,|\phi_{\text{bad}}\rangle \tag{5.6}$$

$$U_{\phi^\perp}\left(\sin(\theta)\,|\phi\rangle + \cos(\theta)\,|\phi^\perp\rangle\right) = \sin(\theta)\,||\phi\rangle\rangle - \cos(\theta)\,|\phi^\perp\rangle \tag{5.7}$$

$$U_{\phi^\perp}U_w\,|\phi\rangle = \cos(2\theta)\,|\phi\rangle + \sin(2\theta)\,|\phi^\perp\rangle \tag{5.8}$$

$$= \sin(3\theta)\,|\phi_{\text{good}}\rangle + \cos(3\theta)\,|\phi_{\text{bad}}\rangle . \tag{5.9}$$

We will see this view in the following lecture.

This amplitude amplification also has a geometric interpretation: one should see U_w and U_s as Householder reflections. Grover's algorithm stays in a subspace spanned by $(|s\rangle, |w\rangle)$. The two operators are reflections with respect to the hyper-planes perpendicular to w and s. It is an elementary fact of Euclidean geometry that when M_1 and M_2 are two lines in the plane intersecting at point O with intersection angle α, the operation of reflection with respect to M_1, followed by reflection with respect to M_2, is rotation by angle 2α around O. Then, the product U_sU_w is a rotation in the $(|s\rangle, |w\rangle)$ plane (for the first $\approx \pi\sqrt{N}/4$ iterations from $|s\rangle$ to $|w\rangle$) by $\theta = 2\arcsin\frac{1}{\sqrt{N}}$. This view is beautifully elaborated by [46].

Without giving a complete derivation here, let us consider $|x_0^\perp\rangle$ and $|\phi_0\rangle$ at an angle β. Then $U_{|\phi_0^\perp\rangle}U_{|x_0\rangle}$ is a rotation around the origin by angle 2β.Starting with a state $|\phi_0\rangle = \sin(\beta)\,|x_0\rangle + \cos(\beta)\,|x_0^\perp\rangle$, after q Grover iterations, we obtain: $|\phi_k\rangle = \sin((2q+1)\beta)\,|x_0\rangle + \cos((2q+1)\beta)\,|x_0^\perp\rangle$. We thus wish to pick q such that $\sin((2q+1)\beta)$ is as close as possible to 1.

[41] suggests that U_w should be seen as:

$$\mathbf{U}_w = \begin{bmatrix}1 & 1\end{bmatrix}\left(\prod_{i=1}^{n}\mathbf{M}_i\right)\begin{bmatrix}1\\-2\end{bmatrix} \tag{5.10}$$

with

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{I}_i & 0 & 0 & \dots \\ 0 & |w_i^1\rangle\langle w_i^1| & 0 & \dots \\ 0 & 0 & |w_i^2\rangle\langle w_i^2| & \dots \\ \dots & \dots & \dots & \dots \\ \dots & 0 & 0 & |w_i^S\rangle\langle w_i^S| \end{bmatrix} \tag{5.11}$$

where $\mathbf{I}_i$ is the 2×2 identity matrix acting on qubit i and $|w_i^\alpha\rangle\langle w_i^\alpha|$ projects α on the bitstring i.

The diffusion operator U_s is similar, except for the replacement of M_i by

$$\mathbf{M}_i' = \begin{bmatrix} \mathbf{I}_i & 0 \\ 0 & |+\rangle\langle +| \end{bmatrix} \tag{5.12}$$

in (5.10).

As we have mentioned at the beginning, there is also a fair amount of controversy, which centers around three issues:

- [39, 40]: one needs to be able to run the oracle with an error that scales with $N^{-1/4} = 1/2^{n-4}$. This is a very exacting standard which may be difficult to obtain for non-trivial n.
- quantumly, one needs to be able to implement the oracle in unit amount of time, but not to be able to implement the product of the Grover iteration $U_s U_w$ in unit amount of time, and not to be able to implement *many* things classically.
- [41]: the tensor-analytic view (5.10) suggests that one uses rank-2 matrix product operation, which are classically simulable in polytime. Then, one efficiently simulates the product of the Grover iterations as well.

5.2 Dynamic Programming

Let us now consider two NP-Hard functional (optimization) problems. In the TRAVELLING SALESMAN PROBLEM (TSP), we seek the shortest simple cycle that visits each vertex in a weighted graph G once (*Hamiltonian circuit*). In the MINIMUM SET COVER, we seek the minimum cardinality subset $\mathscr{S}' \subseteq \mathscr{S}$ such that

$$\bigcup_{S \in \mathscr{S}'} S = \mathscr{U}$$

for some given $\mathscr{S} \subset \mathscr{U}$, with the cardinality of the ground set $|\mathscr{U}| = n$ and $|\mathscr{S}| = m$.

A naive classical approach to either problem would construct a dynamic programming tableau, where in each row r in the tableau, we would have the lengths of Hamiltonian circuits in r-vertex subgraphs. Following [42], let $f(S,u,v)$ denote the length of the shortest path in the graph induced by a subset of vertices S that starts in $u \in S$, ends in $v \in S$ and visits all vertices in S exactly once. Then:

$$f(S,u,v) = \min_{\substack{t \in N(u) \cap S \\ t \neq v}} \{w(u,t) + f(S \setminus \{u\}, t, v)\}, \qquad f(\{v\}, v, v) = 0. \tag{5.13}$$

where $N(u)$ is the neighbourhood of u in G. For $k \in [2, |S|-1]$ fixed,

$$f(S,u,v) = \min_{\substack{X \subset S, |X| = k \\ u \in X, v \notin X}} \min_{\substack{t \in X \\ t \neq u}} \{f(X,u,t) + f((S \setminus X) \cup \{t\}, t, v)\}. \tag{5.14}$$

The algorithm of [42] picks some $\alpha \in (0, 1/2]$ and classically precomputes $f(S,u,v)$ for all $|S| \leq (1-\alpha)n/4$ using dynamic programming (5.13). That is, it computes the bottom rows of the tableau classically, in time exponential in n. Quatumly, it obtains

$$\min_{\substack{S \subset V \\ |S| = n/2}} \min_{\substack{u,v \in S \\ u \neq v}} \{f(S,u,v) + f((V \setminus S) \cup \{u,v\}, v, u)\}$$

over all subsets $S \subset V$ such that $|S| = n/2$ by taking the following steps:

1. Run Grover on (5.14) with $k = \alpha n/4$ to calculate $f(S,u,v)$ for $|S| = n/4$ starting with the rows of the tableau obtained classically.
2. Run Grover on (5.14) with $k = n/4$ to calculate $f(S,u,v)$ for $|S| = n/2$.

Under very strong assumptions about storing the data in quantum RAM, [42] claim a speed-up as suggested in Table 5.1. Notice that much of the controversy surrounding the original Grover applies to this setting as well, compounded by the QRAM assumptions. Under more plausible assumptions, [48] study dynamic programming

	Classical (best known)	Quantum (of [42])
Vertex Ordering Problems	$O^*(2^n)$	$O^*(1.817^n)$
Travelling Salesman Problem	$O(n^2 2^n)$	$O^*(1.728^n)$
Minimum Set Cover	$O(nm2^n)$	$O(\text{poly}(m,n)1.728^n)$

Table 5.1: Summary of the results of [42].

with convex value functions.

[most]tcolorbox listings hyperref float Theorem

Chapter 6

Quantum Walks and Quantum Replacements of Monte Carlo Sampling

In what follows we denote the imaginary unit as $\mathrm{i} = \sqrt{-1}$ and the $n \times n$ unit matrix as $\mathbf{1}_n$ (we skip the subscript if implied). Any comments, corrections and suggestions are most welcome. Send me an e-mail at korpageo@fel.cvut.cz.

6.1 Quantum Walks

Quantum (Random) Walks serve as a fundamental concept in the realm of quantum computing, offering a distinct perspective on random processes compared to their classical counterparts. Quantum walks, and algorithms that utilize them, have several important features that we aim to address in this section. Most notably quantum walks often show quadratic speedups [49] (similar to Grover's algorithm), sometimes show exponential speedups [50] (for example, in the Hidden Flat Problem we describe in Sec. 6.1.6) and, of equal importance, form a model of universal (quantum) computation [51, 52] allowing them to be on the same foot with the quantum Turing machine or the quantum circuit model of computation.

Here we will first introduce discrete quantum walks, then continuous quantum walks, and finally motivate their universality. A good, comprehensive introduction to quantum walks is [53] as well as the textbook [54].

6.1.1 Basics of Quantum Walks

The first quantum algorithms were built on the foundation of Fourier sampling (famously Shor's algorithm [55]), but a new category of algorithms emerged with the

introduction of the quantum walk [56, 53]—a quantum version of the classical random walk.

A quantum walk is a quantum process on a graph $G = (V,E)$, where $V = V(G)$ is the set of vertices and $E = E(G)$ the set of edges, with basis states $|x\rangle, x \in V$. For simplicity, let $V = \mathbb{Z}$ in what follows. Denote the corresponding Hilbert space as $\mathscr{H}_G$. At each time step, a quantum walk corresponds to a unitary map $U \in U(N)$ such that

$$\begin{aligned} U : \mathscr{H}_G &\to \mathscr{H}_G \\ |x\rangle &\mapsto a|x-1\rangle + b|x\rangle + c|x+1\rangle \end{aligned} \tag{6.1}$$

which conveys the information for the potential that $|x\rangle$

1. moves left with some amplitude $a \in \mathbb{C}$,
2. stays at the same place with amplitude $b \in \mathbb{C}$,
3. moves right with amplitude $c \in \mathbb{C}$.

In addition, our goal is for the process to exhibit consistent behavior across all vertices. That is, a, b and c should be independent of $x \in V$ (similarly to how the probabilities of moving left/right are independent of x in the classical random walk). Unfortunately, this definition does not work.

Theorem 6.1. *Transformation U defined by Eq. (1) is unitary if and only if one of the following three conditions is true:*

1. $|a| = 1, b = c = 0$,
2. $|b| = 1, a = c = 0$,
3. $|c| = 1, a = b = 0$.

The reason is that the only possible transformations are the trivial ones (ones that at each step either always move left or always stay in place or always move right). The same problem also appears when defining quantum walks on many other graphs.

This problem can be solved by introducing an additional "coin" state tensored to $|x\rangle$. We consider the state space consisting of states $|i,x\rangle$ for $i \in \{0,1\}$, $x \in \mathbb{Z}$, with Hilbert spaces $\mathscr{H}_{\mathrm{C}} = \mathbb{C}^2$, $\mathscr{H}_{\mathrm{W}} = (\mathbb{C}^2)^{\otimes K}$, $K \in \mathbb{Z}_{>0}$, respectively. At each step, we perform two unitary operations:

(1) A *coin flip* operation $C : \mathscr{H}_{\mathrm{C}} \to \mathscr{H}_{\mathrm{C}}$ which "puts" the walker in superposition, so it walks all possible paths simultaneously.

(2) This is followed by a *shift* operation $S : \mathscr{H}_{\mathrm{W}} \to \mathscr{H}_{\mathrm{W}}$ the operator responsible for the actual walk on G.

These operators act as:

$$C|i,x\rangle = \begin{cases} a|0,x\rangle + b|1,x\rangle & \text{if } i = 0, \\ c|0,x\rangle + d|1,x\rangle & \text{if } i = 1. \end{cases} \tag{6.2}$$

$$S|i,x\rangle = \begin{cases} |0,x+1\rangle & \text{if } i = 0, \\ |1,x-1\rangle & \text{if } i = 1. \end{cases} \tag{6.3}$$

In fact, C can be any element of $U(2)$. Very often the Hadamard operator is chosen (giving the walker the name "Hadamard walker"), that is

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{6.4}$$

while S can be explicitly described as follows:

$$S = \left(|0\rangle\langle 0| \otimes \sum_{x=-\infty}^{\infty} |x+1\rangle\langle x| \right) + \left(|1\rangle\langle 1| \otimes \sum_{x=-\infty}^{\infty} |x-1\rangle\langle x| \right). \tag{6.5}$$

Remark. We can equally exchange the order of the Hilbert spaces. In this convention $C \equiv (\mathbf{1}_{|V|} \otimes C)$ and $S \equiv (S \otimes \mathbf{1}_2)$.

A *step of a quantum walk* amounts to the unitary $U = SC$. This operator is termed a "coin" operator because its action on $|i,x\rangle$, $i \in \{0,1\}$, is to put it in the superposition state $\sqrt{p_0}|0,x\rangle + \sqrt{p_1}|1,x\rangle$ and it will be measured with probability p_0 in $|0,x\rangle$ and with probability p_1 in $|1,x\rangle$. If $C = H$, then $p_0 = p_1 = 1/2$, thus the coin analogy.

Following Eqs. (6.3) and (206), in Fig. 6.1 we can see the probability distribution we obtain after performing a quantum walk with 100 steps. There seems to be an inherit bias towards the right (center at $x = 50$).

Remark on Bias. The quantum walker's initial state is the product of the coin state and the position state. The former state controls the direction in which the walker moves. Therefore, the choice of coin operator leads to vastly different constructive and destructive interference patterns.

In the case of Fig. 6.1, the initial coin state and coin operator are chosen such that the quantum amplitudes add up constructively in one direction and destructively in the other, and the walker is more likely to move preferentially in the direction where constructive interference occurs.

This behavior is in stark contrast to a classical random walk, where the walker has equal probability of moving left or right at each step, and there is no preference or bias for either direction. The bias in a quantum walk is a unique characteristic of the underlying physics.

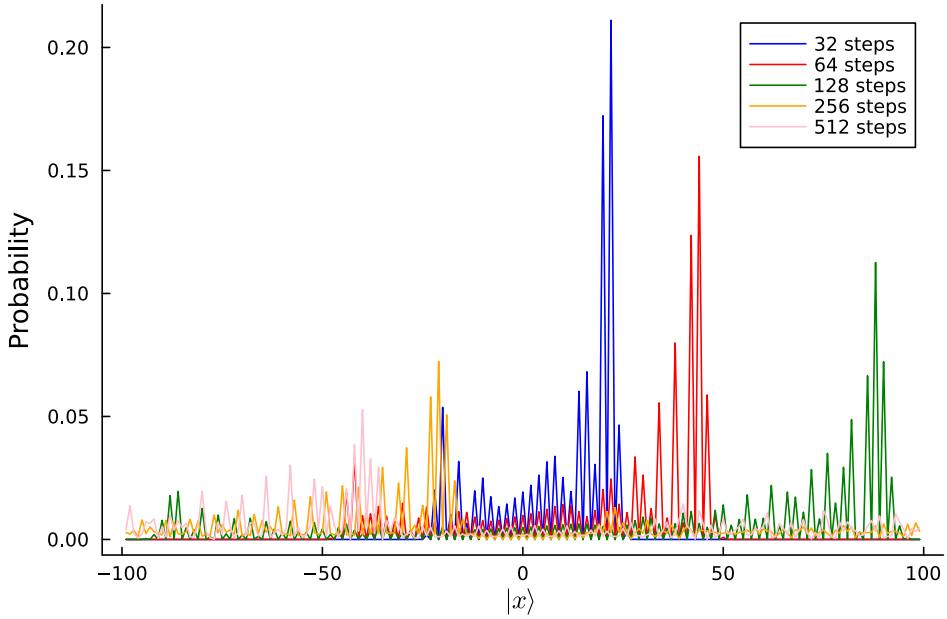


Fig. 6.1: Probability distribution of quantum walk, starting at $|-,0\rangle$, after different numbers of steps.

6.1.2 *Quantum walk on a subset of* $\mathbb{Z}$

Let us see how this works with an example on a bounded subset of the integer line with $C = H$. It is common to assume that the walker starts at position $x = 0$ with the coin state being the $|0\rangle$ or $|1\rangle$ state.

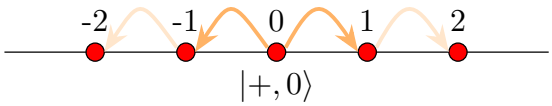


Fig. 6.2: Beginning a quantum walk, after the coin operator has been applied, at $|+,0\rangle$, by applying $C = H$ on $|0,0\rangle$, on the $\mathbb{Z}$-line.

For ease of notation, we denote the r-th application of the quantum walk operator U by $U^{(r)}|\psi_{r-1}\rangle$. Following the previous discussion, the quantum walk amounts to the following set of operations:

```
Select coin operator C = H
```

```
Initialize the state (position of the walker):
   |0⟩ = |0⟩_C ⊗ |0⟩_W = |0,0⟩ (or |1,0⟩)

for r ∈ ℕ repeat U^r|0⟩ as:
    Apply the coin operator: C|0⟩
    Apply the shift operator: S(C|0⟩)

Measure U^r|0⟩
```

Listing 6.1: Quantum Walk

Therefore, the initial state is $|\mathbf{0}\rangle \equiv |\psi_0\rangle$ and we obtain

$$|\psi_1\rangle = \frac{|0,-1\rangle + |0,1\rangle}{\sqrt{2}} \tag{6.6}$$

$$|\psi_2\rangle = \frac{|0,-2\rangle + |1,0\rangle + |0,0\rangle - |1,2\rangle}{2} \tag{6.7}$$

$$|\psi_3\rangle = \frac{|1,-3\rangle - |0,-1\rangle + 2(|0\rangle + |1\rangle)|1\rangle + |0,3\rangle}{2\sqrt{2}} \tag{6.8}$$

This state is not symmetric around the origin and the probability distributions will not be centered at the origin. This is clear from Fig. 6.1. As a matter of fact the standard deviation of the walker, after r iterations of U is [57]:

$$\sigma(r) \approx 0.54r, \tag{6.9}$$

see Fig. 6.3. This implies that the standard deviation in a coined quantum walk increases linearly over r, in contrast to the classical case where it grows with the square root in r.

In a classical random walk, the walker moves randomly through the graph, and its position becomes more uncertain over time. The standard deviation of its position typically increases linearly with the number of steps taken. This linear increase signifies a diffusive spread of the walker. On the other hand, a quantum walk displays *ballistic behavior*, which means that it spreads faster than a classical random walk. In a Hadamard quantum walk, the walker's position uncertainty (as measured by the standard deviation) increases roughly quadratically faster with the number of steps taken, which is a more efficient spreading of the walker over the graph.

6.1.3 Quantum Walk on a Complete Graph

Quantum walks can be studied on more generic graphs. In this section, we will study quantum walks on a symmetric (complete) graph in order to attain more intuition.

Let us pick an easy-to-work-with graph, the complete graph K_4 with 4 vertices and 6 edges and perform such *search*.

Classical Random Walks on K_4

Let us commence with a classical random walk on K_4 wherein we are looking to "find" the marked vertex #2 (but we do not know it). In Fig. 6.5 we display the success probability after 1 and 2 steps.

Overall, the trend for the success probability continues, and we observe the behavior of the walker in Fig. 6.6.

Then, for large N, the success probability of 1/2 is reached after $\mathcal{O}(N)$ steps.

Quantum Grover Walks on K_4

Moving on to quantum walks, we have to implement the coin and shift operators. At each vertex, we have two pieces of information: the position and the direction, just like in the case of the $\mathbb{Z}$-walker. Diagrammatically at step 0 we are back at the left of Fig. 6.5. In total we have 12 amplitudes to consider; see Fig. 6.7. Initially, we have $a_{ij} = \frac{1}{\sqrt{12}}$ for all i, j.

Then, the coin flip operator C, which here is taken to be Grover's diffusion operator, amounts to marking the state we look for, assigning a negative sign to the corresponding amplitudes. The marking is done by assuming access to an oracle O (essentially the same oracle found in Grover's operator) that is able to perform this operation. Then, it changes the direction of adjacent red-blue pair vertices, see Fig. 6.7. Then S reverses the amplitude values along their mean at each vertex. For example, the mean of the vertex #1 after application of C is

$$\mu_{12} = \frac{a_{21} + a_{13} + a_{14}}{3}. \tag{6.10}$$

Therefore, S amounts to a map $S : a_{ij} \mapsto a'_{ij} = \mu_{12} - a_{ij}$, for the three pairs $\{21, 13, 14\}$. Of course, this is applied to all amplitudes for all vertices. In the second step, we already get the amplitude asymmetry resulting from the oracle flipping the signs of the marked vertex followed by C and then S. As a result, one observes that:

$$\text{probability of success at step 1} = \frac{11}{108} \approx 0.1 \tag{6.11}$$

$$\text{probability of success at step 2} = \frac{25}{36} \approx 0.7 \tag{6.12}$$

Overall, for a large number of vertices N, the probability that the walker lands on the marked vertex is 1/2 is given after $\pi\sqrt{N}$ steps and therefore the run-time is $\mathcal{O}(\sqrt{N})$. This marks another example in which quantum walks portray a quadratic speedup over classical random walks.

6.1.4 Szegedy Walks

Consider an undirected and unweighted graph G. Szegedy's quantum walk occurs on the edges of the bipartite double cover of the original graph. If the original graph is G, then its bipartite double cover is the graph tensor product $G \times K_2$ which duplicates the vertices into two partite sets X and Y. A vertex in X is connected to a vertex in Y if and only if they are connected in the original graph; see Fig. 6.9.

The Hilbert space of a Szegedy walk, therefore, is $\mathbb{C}^{2|E|}$. Let us denote a walker on the edge connecting $x \in X$ with $y \in Y$ as $|x,y\rangle$. Then the computational basis is:

$$|x,y\rangle, \qquad x \in X, y \in Y, x \sim y \tag{6.13}$$

where $x \sim y$ denotes that the vertices x and y are adjacent. Szegedy's walk is defined by repeated applications of the unitary

$$U = R_2 R_1, \tag{6.14}$$

where

$$R_1 = 2\sum_{x \in X} |\phi_x\rangle \langle \phi_x| - \mathbf{1} \tag{6.15}$$

$$R_2 = 2\sum_{y \in Y} |\psi_y\rangle \langle \psi_y| - \mathbf{1}, \tag{6.16}$$

are reflection operators and

$$|\phi_x\rangle = \frac{1}{\sqrt{\deg(x)}} \sum_{y \sim x} |x,y\rangle \tag{6.17}$$

$$|\psi_y\rangle = \frac{1}{\sqrt{\deg(y)}} \sum_{x \sim y} |x,y\rangle. \tag{6.18}$$

Here, $\deg(x)$ is the degree of vertex x and $y \sim x$ denotes the sums over the neighbors of x. Observe that $|\phi_x\rangle$ is the equal superposition of edges incident to $x \in X$, and $|\psi_y\rangle$ is the equal superposition of edges incident to $y \in Y$. Here, there is an equivalent of the "inversion about the mean" operation of Grover's algorithm, which we also saw previously in the context of walks over K_4. The reflection R_1 goes through

each vertex in X and reflects the amplitude of its incident edges about their average amplitude, and R_2 similarly does this for the vertices in Y.

Classically, to search for a marked vertex on G with a classical random walk, one randomly walks until a marked vertex is found, and then the walker stays at the marked vertex.

Quantumly, Szegedy's quantum walk searches by quantizing this random walk with absorbing vertices and the resulting bipartite double cover. Search is performed by repeatedly applying the unitary

$$\widetilde{U} = \widetilde{R}_2 \widetilde{R}_1, \tag{6.19}$$

where the tilde distinguishes in that we are searching for absorbing vertices. At unmarked vertices they act as $\widetilde{R}_j = R_j$ simply by inverting the amplitudes of the edges around their average at each vertex. At the marked vertices, similarly to the K_4 case, they act by flipping the signs of the amplitudes of all incident edges. A similar search can be performed using Grover's diffusion operator.

6.1.5 Continuous-time Quantum Walks

Let us define the quantum analog of continuous-time random walks that will allow us later to understand the universality of quantum walks.

Classical continuous-time random walks.

The continuous-time random walk on a graph $G = (V,E)$ with adjacency matrix A defined as:

$$A_{i,j} = \begin{cases} 1, & (i,j) \in E \\ 0, & (i,j) \notin E \end{cases} \tag{6.20}$$

for every pair $i, j \in V$. In this definition we do not allow self-loops therefore the diagonal of A is zero. There is another matrix associated with G that is of equal importance, the Laplacian of G defined as:

$$L_{i,j} = \begin{cases} -\deg(i), & i = j \\ 1, & (i,j) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{6.21}$$

Here, $\deg(i)$ denotes the degree of vertex i. Let $p_i(t)$ denote the probability associated with the vertex i at time t. The continuous-time random walk on G is defined as the solution of the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}p_i(t) = \sum_{j\in V} L_{jk}p_j(t). \tag{6.22}$$

This can be viewed as a discrete analog of the diffusion equation. Observe that

$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_{j\in V} p_j(t) = \sum_{j,k\in V} L_{jk}p_k(t) = 0 \tag{6.23}$$

This shows that an initially normalized distribution remains normalized; the evolution of the continuous-time random walk for any time t is a stochastic process. The solution of the differential equation can be given in closed form as:

$$p(t) = e^{Lt}p(0). \tag{6.24}$$

Continuous-time quantum walks. Eq. (6.23) is very similar to the Schrödinger equation

$$\mathrm{i}\frac{\mathrm{d}}{\mathrm{d}t}|\psi\rangle = H|\psi\rangle, \tag{6.25}$$

Instead of probabilities of Eq. (6.23) we can insert the amplitudes $q_j(t) = \langle j \mid \psi(t)\rangle$ where $\{|j\rangle : j \in V\}$ is an orthonormal basis for the Hilbert space. Then, we obtain the equation:

$$\mathrm{i}\frac{\mathrm{d}}{\mathrm{d}t}q_j(t) = \sum_{k\in V} L_{jk}q_k(t), \tag{6.26}$$

where the Hamiltonian is given by the Laplacian L. Since the Laplacian is a Hermitian operator, these dynamics preserve normalization in the sense that $\frac{\mathrm{d}}{\mathrm{d}t}\sum_{j\in V}\left|q_j(t)\right|^2 = 0$. The solution of reads:

$$U(t) = e^{-\mathrm{i}Ht} = e^{-\mathrm{i}Lt}, \tag{6.27}$$

and the evolution of an initial state from $t = 0$ to some arbitrary time t is given by:

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle. \tag{6.28}$$

Quantum Walk on the Hypercube. This is another example where the difference between random and quantum walks becomes tremendous. Consider the Boolean hypercube, that is, the graph with vertex set $V = \{0,1\}^n$ and edge set $E = \{(x,y) \in V \times V \,|\, \Delta(x,y) = 1\}$, where $\Delta(x,y)$ denotes the Hamming distance between strings x and y. When $n = 1$, the hypercube is simply an edge, with adjacency matrix

$$\sigma_x := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{6.29}$$

For general n, the graph is the Cartesian product of this graph with itself n times, and the adjacency matrix is

$$A = \sum_{j=1}^{n} \sigma_x^{(j)}, \tag{6.30}$$

where $\sigma_x^{(i)}$ denotes the operator acting as σ_x on the i^{th} bit, and as the identity on every other bit. Consider the quantum walk with the Hamiltonian given by A. Since the terms in the above expression for the adjacency matrix commute, the unitary operator that describes the evolution of this walk is simply

$$\begin{aligned} e^{-\mathrm{i}At} &= \prod_{i=1}^{n} e^{-\mathrm{i}\sigma_x^{(i)}t} \\ &= \bigotimes_{i=1}^{n} \begin{pmatrix} \cos t & -\mathrm{i}\sin t \\ -\mathrm{i}\sin t & \cos t \end{pmatrix} \\ &\equiv U(t). \end{aligned} \tag{6.31}$$

Note that $U(\pi/2)$ flips every bit of the state (up to an overall phase), resulting in a mapping of any input state $|x\rangle$ to the state $|\bar{x}\rangle$ corresponding to the opposite vertex of the hypercube.

In contrast, consider the continuous-time (or discrete-time) random walk starting from the vertex x. The probability of reaching the opposite vertex $\bar{x}$ is exponentially suppressed at any time, since the walk rapidly reaches the uniform distribution over all 2^n vertices of the hypercube.

6.1.6 Exponential speedups using Quantum Walks

In this section we will briefly introduce the *Hidden Flat Problem* (HFP) and how quantum walks offer an exponential speedup. This is an algorithm that aims to find hidden nonlinear structures over Galois fields[1] $\mathbb{F}_p$, for p prime.

You have already heard about Schor's algorithm and its successes:

1. Factoring (see Lecture 9 for the implications thereof).
2. Discrete log.

[1] Galois fields over primes are also called prime fields. For each prime number p, the prime field $\mathbb{F}_p$ of order p is constructed as the integers modulo p, that is $\mathbb{Z}/p\mathbb{Z}$. See Chapter 5, Sec. 5.1.

In the former, the hidden structure here amounts to period finding over $\mathbb{Z}$ that is, a hidden linear structure in one dimension, while for the latter it amounts to finding a hidden line in $\mathbb{Z}_p \times \mathbb{Z}_p$.

In the HFP the goal is to determine a flat (e.g. a line) for spheres of radius $r = 1$, given a uniform superposition over points in $\mathbb{F}_q^d$. In this context, we are promised that the centers of the spheres lie on an unknown flat H, and the goal is to determine this flat using oracular access.

Problem Details

For that, we need to first introduce some weird notation. Let $\mathbb{S}_r^t(\mathbb{F}_q^d)$ denote the sphere of radius r with center t over $\mathbb{F}_q^d$. Additionally, for a finite set S, we denote by

$$|S\rangle := \frac{1}{\sqrt{|S|}} \sum_{s \in S} |s\rangle \tag{6.32}$$

the normalized uniform superposition over elements of S. Using two oracles[2] f_1, f_{-1} – let us assume they exist indeed; they are concretely defined in the context of the Hidden Radius Problem [58]– it is possible to construct the state

$$\rho_r := \frac{1}{q^d} \sum_{t \in \mathbb{F}_q} |\mathbb{S}_r + t\rangle \langle \mathbb{S}_r + t| . \tag{6.33}$$

The flat we are looking for is such a discrete set $H \subseteq \mathbb{F}_q$ allowing us to construct

$$\rho_1 := \frac{1}{|H|} \sum_{h \in H} |\mathbb{S}_1 + h\rangle \langle \mathbb{S}_1 + h| \tag{6.34}$$

The goal is to determine H by making measurements on this state. To accomplish this, a quantum walk is implemented that moves the amplitude from $|\mathbb{S}_1 + h\rangle$ to $|h\rangle$. If a sufficiently large fraction of the amplitude is moved, then the hidden flat can be determined by (classically) solving a noisy linear algebra problem.

To move amplitude from unit spheres to their centers, we will use a continuous-time quantum walk on the Winnie-Li graph.

This graph has vertex set $\mathbb{F}_q^d$, and edges between points $x, x' \in \mathbb{F}_q^d$ with $\Delta(x - x') = 1$. Thus its adjacency matrix (that serves as a Hamiltonian) is

$$A := \sum_{x \in \mathbb{F}_q^d} \sum_{s \in \mathbb{S}_1} |x + s\rangle \langle x| \tag{6.35}$$

The continuous-time quantum walk for time t is simply the unitary operator $U(t) = e^{-\mathrm{i}At}$. This unitary operator can be efficiently implemented on a quantum computer

[2] C.f. Lecture 4, Sec. 5.2 “The Oracle”.

provided that we can efficiently transform into the eigenbasis of A, and can efficiently compute the eigenvalue corresponding to a given eigenvector.

The adjacency matrix (6.35) has eigenvectors

$$|\tilde{k}\rangle := \frac{1}{\sqrt{q^d}} \sum_{x \in \mathbb{F}_q^d} \omega_p^{k \cdot x} |x\rangle, \tag{6.36}$$

for $k \in \mathbb{F}_q^d$. Therefore, by using the Fourier transform of

$$U := \frac{1}{\sqrt{q^d}} \sum_{x,k \in \mathbb{F}_q^d} \omega_p^{k \cdot x} |k\rangle\langle x| \tag{6.37}$$

we can transform to the eigenbasis of A where the corresponding eigenvalues are given by the Fourier transform of a unit sphere λ_k (whose precise form is computable). Almost all of these eigenvalues can be computed with complexity $\mathcal{O}\left(\sqrt{q^{d-1}}\right)$.

Then, the main result is the following algorithm:

```
Require ρ_H

for t = 1/√(q^{d-1} log q):
    Perform a continuous-time quantum walk with U = e^{-iAt}

Measure in the computational basis
```

Listing 6.2: Quantum Flat Problem using Quantum Walks

Each point in H occurs with probability $|H|^{-1}\left(1/\log q + \mathcal{O}\left(1/\log^{3/2} q\right)\right)$, and any point not on H occurs with probability $\mathcal{O}\left(q^{-d}\right)$.

With the above in mind, and assuming $d = \mathcal{O}(1)$ and odd, there is a quantum algorithm to determine the hidden flat of centers in time $\text{poly}(\log q)$. This provides an exponential speedup over classical algorithms.

While this algorithm is not the most trivial to follow, it is a remarkable example on the exponential speedup that quantum walks provide for certain problems.

Further quantum algorithms for algebraic problems are given in[59], an excellent survey.

6.1.7 Universality of Quantum Walks

In earlier lectures you have seen that quantum computation with time-independent Hamiltonians provides a universal model of computation. In this section, we will argue that quantum walks form a universal model of computation. Childs [51] showed that even a restricted version of this model, the "universal computation graph," forms a universal model for quantum computation. This means that any problem that can be solved by a common gate-based quantum computer can also be solved by such a quantum walk (similarly to programable quantum gate arrays or to adiabatic quantum computing, as we discuss in the Chapter 7).

This result shows the computational power of the quantum walk and that, at least in principle, any quantum algorithm we have seen previously can be recast as a quantum walk algorithm. Further improvements, in terms of complexity theoretic issues, were made in [52] using multi-particle walks.

To understand universality, we consider a (continuous) walker on $\mathbb{Z}$, like in Sec. 6.1.2, where the basis states are $|x\rangle$. The eigenstates of the adjacency matrix are the (normalized) momentum states $|k\rangle$, that is, the states that satisfy

$$\langle x|k\rangle = e^{-\mathrm{i}kx}, \tag{6.38}$$

with $\langle k|k'\rangle \sim \delta(k-k')$. The reason for this is deeply routed in physics (we will not go into details here). The point is that, $|k\rangle$ are the momentum eigenstates which are used to understand how scattering (particle interactions) works in quantum mechanics (and quantum field theory). In momentum space, with orthogonal states $|\phi_k\rangle \equiv |k\rangle$, we know that

$$|k\rangle = \sum_{x\in\mathbb{Z}} e^{-\mathrm{i}kx}|x\rangle. \tag{6.39}$$

These are also referred to as *momentum states* however, they are not normalizable (instead, we can think of them as maps $E(G) \to \mathbb{C}$. Using the adjacency matrix as the Hamiltonian H, it follows that

$$H|k\rangle = 2\cos(k)|k\rangle. \tag{6.40}$$

Next, let us consider a finite graph G and create out of it an infinite graph with adjacency matrix H by attaching semi-infinite lines to M of its vertices.

The states living on the j-th line are labeled as $|x,j\rangle$ where $|0,j\rangle$ corresponds to the state in G and where x is allowed to walk along the j-th line. The adjacency matrix of this graph is denoted by H and each of its eigenstates must be a superposition of the form of Eq. (6.39) with momenta k taking any of the values:

- $\pm k$ with eigenvalues $2\cos(k)$,

- $k = \pm i\kappa$ and eigenvalue $2\cosh(\kappa)$,
- $k = \pm i\kappa + \pi$ and eigenvalue $-2\cosh(\kappa)$.

Here $\kappa \in \mathbb{R}_{\geq 0}$. We can truncate $|k\rangle$ such that it has support over a finite number of vertices. Denote the truncated state supported over L vertices as

$$|k\rangle_L := \frac{1}{\sqrt{L}} \sum_{x=1}^{L} e^{-\mathrm{i}kx} |x\rangle. \tag{6.41}$$

In the physics literature, such states are called *wave packets* (this is just terminology originating from physics; there is no physical wave of any form or size propagating through any physical medium here) and the sign of the exponential denotes the direction of the wave; see Fig. 6.14. The infinite line in Fig. 6.14 becomes a universal computation graph by inserting a finite graph G at, say, vertex 0. As seen in Fig. 6.15. In principle, one can prepare a wave packet as the one with momentum k and let it propagate.

This amounts to a dynamic scattering process. Let us denote this incoming (to G) wave packet as

$$|w(k)\rangle_{\mathrm{L}} \quad \text{if the wave packet comes from the left,} \tag{6.42}$$
$$|w(k)\rangle_{\mathrm{R}} \quad \text{if the wave packet comes from the right.} \tag{6.43}$$

The dynamics correspond to the following equations:

$$\langle x_{\mathrm{L}} | w_{\mathrm{L}}(k) \rangle = e^{-\mathrm{i}kx} + R_{\mathrm{L}}(k) e^{\mathrm{i}kx} \tag{6.44}$$
$$\langle x_{\mathrm{R}} | w_{\mathrm{L}}(k) \rangle = T_{\mathrm{L}}(k) e^{\mathrm{i}kx} \tag{6.45}$$
$$H|w(k)\rangle = 2\cos(k)|w(k)\rangle, \tag{6.46}$$

where R_L is a reflection coefficient and T_L is the transfer coefficient. Similarly, we can write down the equations for right-coming wave packets.

For every scattering process, as the one above, there is a scattering matrix S. In this case,

$$S = \begin{pmatrix} R_{\mathrm{L}} & T_{\mathrm{L}} \\ R_{\mathrm{R}} & T_{\mathrm{R}} \end{pmatrix}, \tag{6.47}$$

and it is an element of $U(2)$. More generally, an arbitrary number of semi-infinite lines can be considered as in Fig. 6.13 with an arbitrary graph G. If there are N semi-infinite lines, then $S \in U(N)$.

We are now in a position to understand why **quantum walks form a universal model of quantum computation**. It is possible to encode a qubit state by considering two universal computation diagrams in one dimension as in Fig. 6.17.

As before, we can insert a graph G with 4 semi-infinite lines as in Fig. 6.18.

Then, a unitary is implemented by inserting a graph G such that its corresponding S-matrix[3] has the structure

$$S = \begin{pmatrix} 0 & U^\dagger \\ U & 0 \end{pmatrix}, \tag{6.48}$$

where $U \in U(2)$. Therefore, a unitary U is implemented by the scattering process of quantum walkers, through a graph G that encodes it. Childs [51] showed that with the above process, it is possible to implement the unitaries

$$U_{\pi/4} = \begin{pmatrix} e^{-\mathrm{i}\pi/4} & 0 \\ 0 & 1 \end{pmatrix}, \quad U_{\mathrm{b}} = -\frac{\mathrm{i}}{\sqrt{2}} \begin{pmatrix} 1 & -\mathrm{i} \\ -\mathrm{i} & 1 \end{pmatrix}, \tag{6.49}$$

which form a universal gate set for one-qubit operations; up to a certain precision ε, any single-qubit gate can be implemented by a string of these two unitaries.

This construction was further generalized to n-qubit gates proving that quantum walks form a universal model of computation.

By considering a finite graph G and attaching $N/2 = n$ pairs of semi-infinite paths, we are able to encode n qubits. Eventually, it is possible to encode any n-qubit unitary to a graph G to obtain a quantum walk equivalent of any arbitrary circuit.

Later, [52] showed that continuous-time multi-particle quantum walks on such graphs are also universal. They too, are generated by a time-independent Hamiltonian with a term corresponding to a single-particle quantum walk for each particle, along with an interaction term. Interestingly, the authors suggest that multi-particle quantum walks can be used, in principle, to build a scalable quantum computer with no need for time-dependent control (e.g. for pulse scheduling).

[3] The S-matrix relates the initial and final (asymptotic) states of a quantum system involved in scattering processes. Essentially, it encodes the probability amplitudes for different scattering channels or processes, which can be used to calculate various observables such as cross-sections and decay rates in particle physics.

6.2 Quantum Amplitude Estimation and Monte Carlo Sampling

Quantum Amplitude Amplification (QAE) was discovered by Gilles Brassard, Peter Hoyer, Michele Mosca and Alain Tapp in [47] and generalizes Grover's algorithm, as we will describe below. In what follows, we proceed to explain QAE directly through the lens of an algorithm candidate to replace Monte Carlo sampling techniques following closely Montanaro's work [60].

The reason lies in the speedup provided by Quantum Phase Estimation.

Classical Monte Carlo Sampling. For simplicity, let us consider a one-dimensional random variable X and a function $f : \mathbb{R} \to [0,1]$. Assume that the mean $\mu = \mathbb{E}[f(X)] < \infty$ and the standard deviation $\sigma^2 = \mathbb{V}[f(X)] < \infty$ are well defined. The Central Limit Theorem ensures that, given an i.i.d. collection of random variables $(X_1, \ldots, X_N)$, following the same distribution as X, for $N \to \infty$, the quantity $\sqrt{N}\frac{\hat{\mu}-\mu}{\sigma}$ converges to a mean-zero Gaussian with unit variance $\mathcal{N}(0,1)$. Here, $\hat{\mu}$ refers to the empirical mean. This implies that for any $\varepsilon > 0$ we estimate that

$$\lim_{N\to\infty} \mathbb{P}(|\hat{\mu} - \mu| \leq \varepsilon) = \lim_{N\to\infty} \mathbb{P}\left(|\mathcal{N}(0,1)| \leq \frac{\varepsilon\sqrt{N}}{\sigma}\right). \tag{6.50}$$

In turn, this implies that for any $z > 0$ and $\delta \in (0,1)$, in order to obtain an estimate of the form $\mathbb{P}(|\hat{\mu} - \mu| \leq \varepsilon)$, $N = \mathcal{O}(1/\varepsilon^2)$ samples are required.

QAE Replacement of Monte Carlo Sampling.

Consider a unitary operator $\mathscr{A}$ that acts on an n-qubit register as follows:

$$\mathscr{A}|0\rangle^{\otimes n} = \sum_{x\in\{0,1\}^k} a_x |\psi_x\rangle|x\rangle, \tag{6.51}$$

for $k < n$, where $|\psi_x\rangle$ is a quantum state consisting of $n-k$ qubits and $|x\rangle$ is a state consisting of k qubits. We are interested in $\mathscr{A}$ because it will allow us to prepare a specific quantum state that encodes a distribution of interest, with encoded data in the states $|x\rangle$, for which we want to estimate certain properties, such as the mean or other moments.

Furthermore, the states $\{|\psi\rangle_x\}_{x\in\{0,1\}^k}$ are assumed to be orthogonal. Next, assume that there is a unitary $\mathscr{W}$ acting as follows:

$$\mathscr{W}|x\rangle|0\rangle = |x\rangle\left(\sqrt{1-f(x)}|0\rangle + \sqrt{f(x)}|1\rangle\right). \tag{6.52}$$

This unitary is introduced to create a quantum state that encodes the function $f(x)$, which represents the property or condition of interest. The quantum state that it creates captures the information about the properties pf $f(x)$ in the amplitudes of the ancilla qubits $|0\rangle$ and $|1\rangle$.

Something quite interesting happens when one combines the two operators in the following way:

$$\mathscr{G} := (\mathbf{1}_{n-k} \otimes \mathscr{W})(\mathscr{A} \otimes \mathbf{1}_k). \tag{6.53}$$

Applying $\mathscr{G}$ to a $|0\rangle^{\otimes(n+1)}$ qubit register yields the following state:

$$|\psi\rangle = \mathscr{G}|0\rangle^{\otimes(n+1)} \tag{6.54}$$

$$= \sum_{x\in\{0,1\}^k} a_x |\psi_x\rangle|x\rangle\Big(\sqrt{1-f(x)}|0\rangle + \sqrt{f(x)}|1\rangle\Big), \tag{6.55}$$

It is customary to refer to these two states as the "bad state":

$$|\psi_{\text{bad}}\rangle := \sum_{x\in\{0,1\}^k} a_x \sqrt{1-f(x)}|\psi_x\rangle|x\rangle, \tag{6.56}$$

and the "good state":

$$|\psi_{\text{good}}\rangle := \sum_{x\in\{0,1\}^k} a_x \sqrt{f(x)}|\psi_x\rangle|x\rangle. \tag{6.57}$$

By considering the projection operator

$$\mathscr{P} := \mathbf{1}_n \otimes |1\rangle\langle 1|, \tag{6.58}$$

we can measure the probability that the last state is the $|1\rangle$ state,

$$\langle\psi|\mathscr{P}^\dagger\mathscr{P}|\psi\rangle = |\psi_{\text{good}}|^2. \tag{6.59}$$

From the definition of a good state, we can further see that

$$|\psi_{\text{good}}|^2 = \sum_{x\in\{0,1\}^k} |a_x|^2 f(x), \tag{6.60}$$

which corresponds, precisely, to the mean $\mu = \mathbb{E}(f(X))$ (note that the random variable X is discretized, as is common with Monte Carlo sampling, to fit the discrete probability of X being in x).

The whole process of estimating μ for a distribution f, therefore, amounts to running the circuit that represents $\mathscr{G}$, measuring the output on the computational basis (this step requires $\text{QFT}^\dagger$) and determining the probability of observing the state $|1\rangle$.

Quadratic Speedup of Monte Carlo Sampling

The speedup arises from [47, Theorem 12]. Concretely, assume access to a unitary

$$U|0\rangle = \sqrt{1-\mu}|\psi_{\text{bad}}\rangle + \sqrt{\mu}|\psi_{\text{good}}\rangle. \tag{6.61}$$

Then, for any $N \in \mathbb{Z}_{\geq 0}$, the QAE algorithm outputs the estimate $\hat{\mu}$ such that

$$|\hat{\mu} - \mu| \leq 2\pi \frac{\sqrt{\mu(1-\mu)}}{N} + \frac{\pi^2}{N^2}, \tag{6.62}$$

with probability at least $8/\pi^2$ by quering the algorithm exactly N times.

By using the so-called "Powering Lemma" which states (approximately) that for any $\delta \in (0,1)$, it is sufficent to iterate with U approximately $\mathscr{O}(\log(1/\delta))$ times to obtain

$$\mathbb{P}(|\hat{\mu} - \mu| \leq \varepsilon) \geq 1 - \delta. \tag{6.63}$$

Putting everything together, we realize that it is required to iterate $\mathscr{G}$ approximately $\mathscr{O}(N \log(1/\delta))$ times to obtain the guarantee of Eq. (6.63), where

$$\varepsilon = 2\pi \frac{\sqrt{\mu(1-\mu)}}{N}. \tag{6.64}$$

That is, for fixed δ the computational cost to obtain (6.63) is $\mathscr{O}(1/\varepsilon)$ which is quadratically better than the $N = \mathscr{O}(1/\varepsilon^2)$ samples required by classical Monte Carlo.

```
Require a probability distribution, a moment f,
    samples x.
Require a unitary 𝒜 that acts on n qubits, such that
    0 ≥ μ ≥ 1, t ∈ ℤ, δ ∈ ℝ_{>0}.
Require a unitary 𝒲 that acts on k+1 qubits

for t iterations repeat the QAE unitary:
    𝒢^t |0⟩^{⊗(n+1)} = [(1_{n-k} ⊗ 𝒲)(𝒜 ⊗ 1_k)]^t |0⟩^{⊗(n+1)}

Perform QFT†

Measure in the computational basis the probability the
    last qubit is |1⟩
```

Listing 6.3: QAE for Monte Carlo sampling

graphicx,epsfig hyperref algorithm algpseudocode hyperref float color Theorem Definition Claim

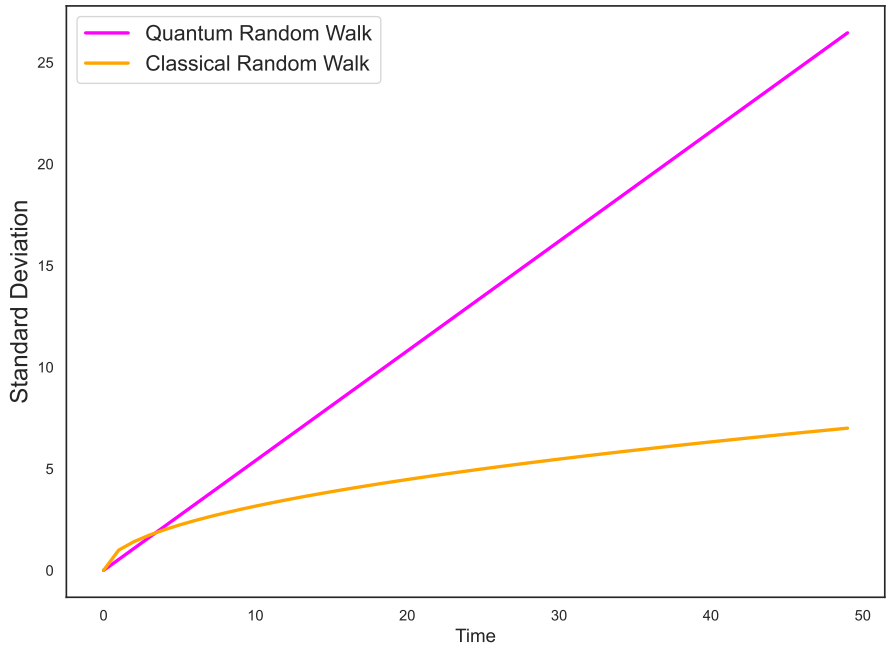


Fig. 6.3: The standard deviation of a classical versus quantum walk as a function of the steps.

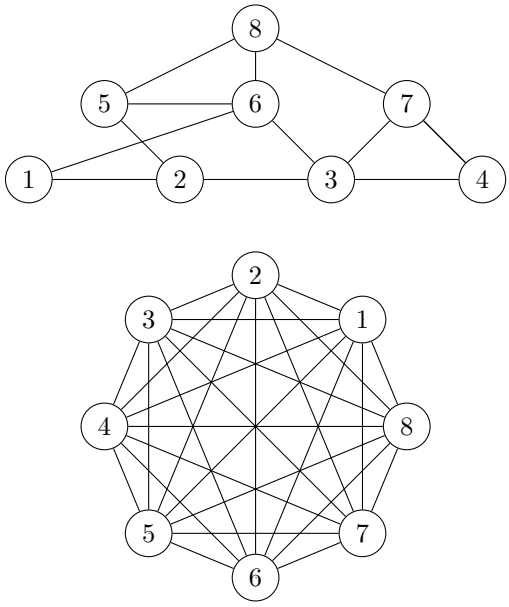


Fig. 6.4: An asymmetric non-complete graph $G = (8, 10)$ and its symmetric completion $\overline{G} = K_8$.

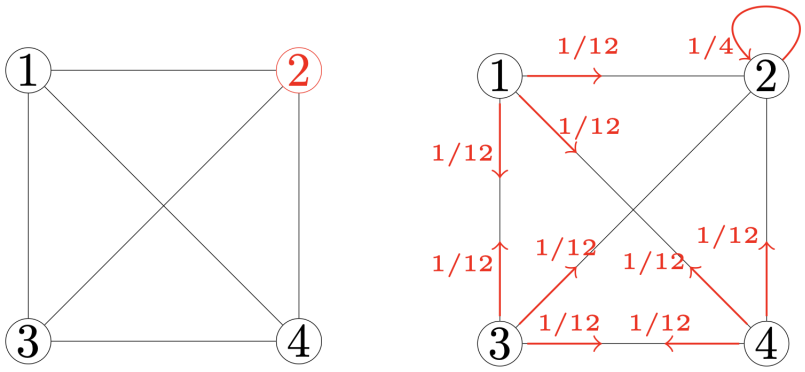


Fig. 6.5: Left: At step 1 the probability that the walker "lands" on vertex #2 is 1/4. Right: At step 2 the probability that the walker "lands" on vertex #2 is 1/2. The loop in vertex #2 denotes that this vertex is a trap: it allows us to know the walker landed on the marked vertex and the walker is not allowed to attain any other state.

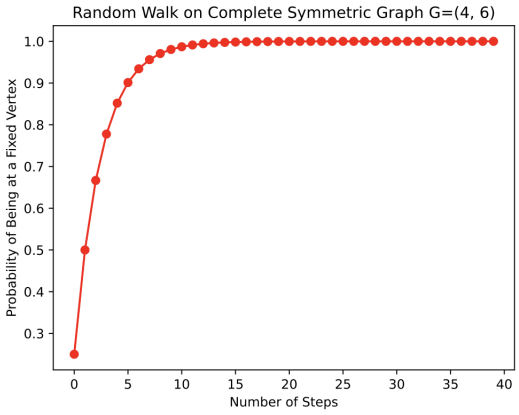


Fig. 6.6: The success probability of a classical random walk on symmetric $G = K_4$.

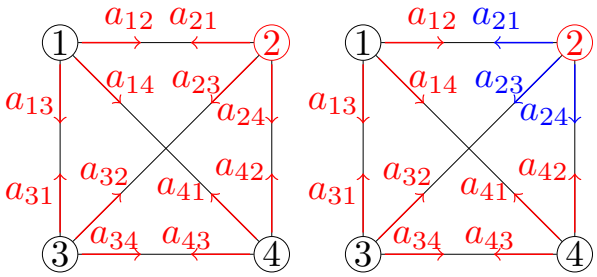


Fig. 6.7: Left: the state of the quantum walk is a superposition of the amplitudes $a_{ij} \in \mathbb{C}$, for all $i, j \in V(K_4)$. Once the oracle is applied the marked state's amplitudes obtain a negative sign (marked with blue and in analogy with Grover's operator).

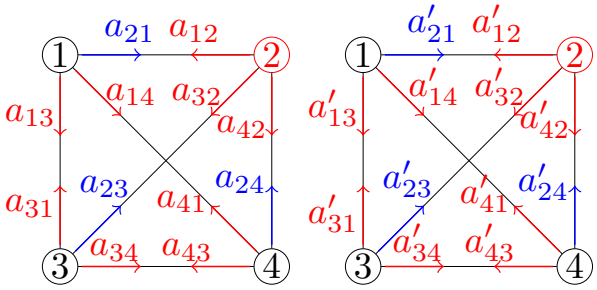


Fig. 6.8: Left: Coin operator is applied and reverses the relevant amplitudes. The shift operator reverses these amplitudes along their means.

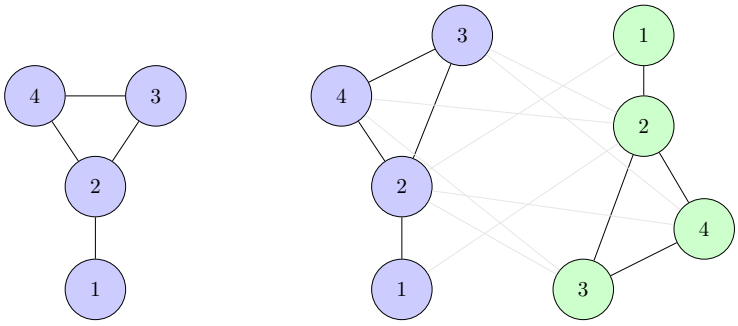


Fig. 6.9: Left: A graph G. Right: The bipartite double cover of G. The double cover contains double the number of edges.

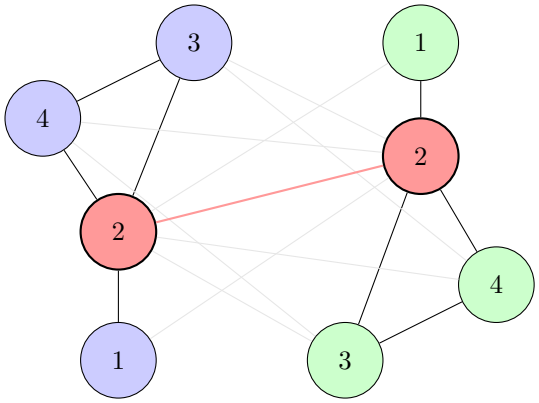


Fig. 6.10: The marked state corresponds to vertex #2 which is an absorbing vertex: $\langle 2_Y|2_X\rangle = \langle 2_X|2_Y\rangle = 0$.

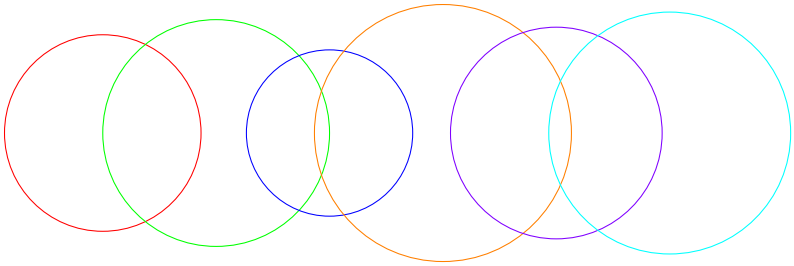


Fig. 6.11: Equidistant circles of various radii over $\mathbb{F}_q^2$ that lie on an unknown flat H on which the radii sit at. Note that the density of points in each sphere is approximately the same since they live on a Galois field.

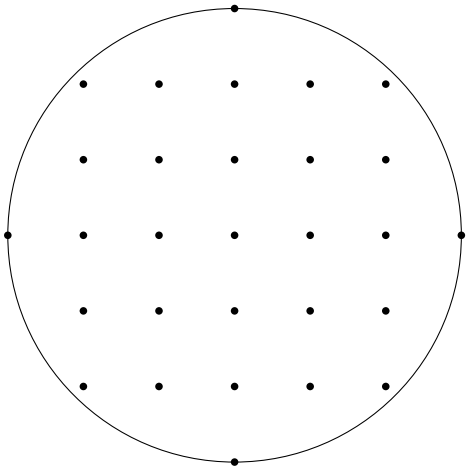


Fig. 6.12: A Winnie-Lie graph over $\mathbb{F}_p^2$ centered at $x = 0$. The edges are not shown.

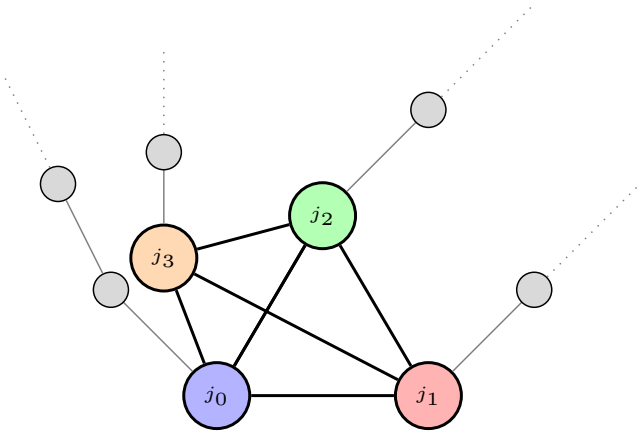


Fig. 6.13: The original graph G (thick) corresponds to the one with colored vertices $\{j_0, j_1, j_2, j_3\}$ and corresponding edges. By attaching semi-infinite lines (vertices with edges) to $M = 4$ vertices of G we construct a new infinite graph. The state of vertex j_ℓ is $|0, \ell\rangle$ with each subsequent edge on the same line having a corresponding state $|x, \ell\rangle$. We call the expanded graph a *universal computation graph*.

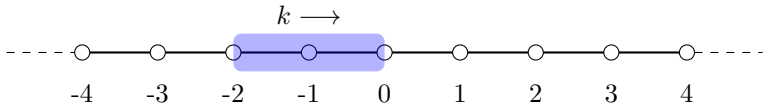


Fig. 6.14: A wave packet supported over 2 vertices moving coming from the (far) left.

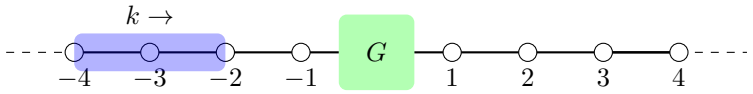


Fig. 6.15: Inserting a finite graph G into the integer line, yields a one-dimensional universal computation graph.

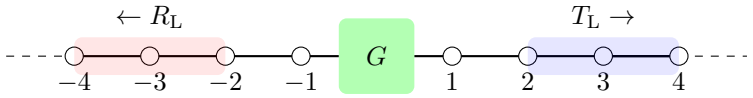


Fig. 6.16: Part of the wave packet will be reflected and part will be transfered through G. The coefficients $R_{\mathrm{L,R}}, T_{\mathrm{L,R}}$ are called reflection and transfer coefficients.

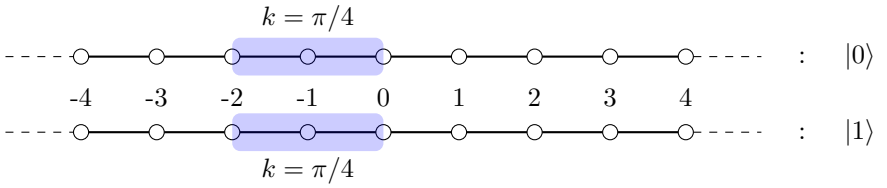


Fig. 6.17: A single qubit can be represented by two infinite lines. Crucially the momentum must be equal to $\pi/4$. The qubit is in the $|0\rangle$ state if the wavepacket propagates in the top line and in the $|1\rangle$ state if at the bottom.

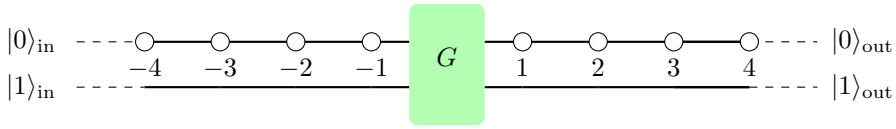


Fig. 6.18: A two-qubit unitary U can be encoded through G to be implemented as a quantum walk.

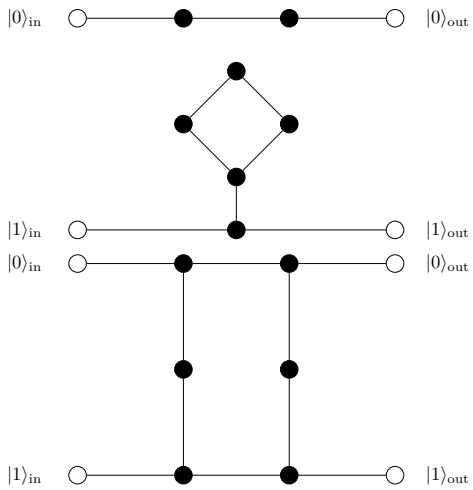


Fig. 6.19: The graphs encoding $U_{\pi/4}$ and U_{b} [51].

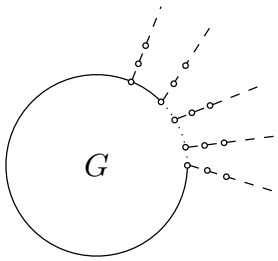


Fig. 6.20: The graph G obtained by attaching N semi-infinite paths to a graph G.

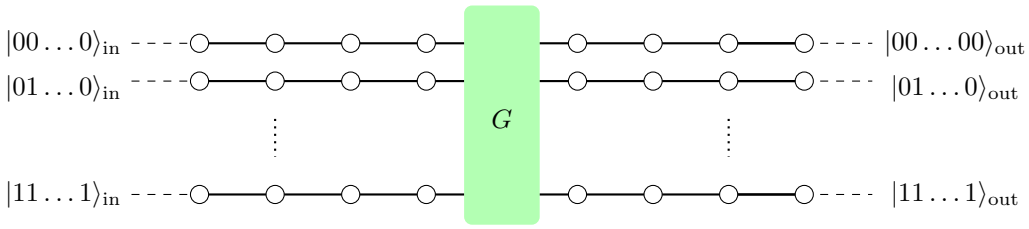


Fig. 6.21: If G is chosen to encode a desired unitary $U \in U(n)$ the circuit can be implemented by a quantum walk.

Chapter 7

Adiabatic Quantum Computing and Quantum Replacements of Optimization Algorithms

Updates

Use this https://arxiv.org/pdf/1611.04471.pdf for more info on adiabatic quantum computing and detailed computations.

7.1 Adiabatic Quantum Computation is Universal

In this lecture we would like to explain the following beautiful result by [61]:

Theorem 7.1. *The model of adiabatic computation is polynomially equivalent to the standard model of quantum computation.*

More specifically, Theorem 7.1 can be also described as:

Theorem 7.2. *The model of adiabatic computation with explicit sparse Hamiltonians is polynomially equivalent to the standard model of quantum computation.*

Even more interestingly, in [61] it is explicitly shown that:

Theorem 7.3. *Any quantum computation can be efficiently simulated by an adiabatic computation with 2-local nearest neighbor Hamiltonians operating on six-state particles set on a two dimensional grid.*

We will discuss how one begins to even show these theorems above.

7.1.1 Motivation

Let us provide some motivation. The study of adiabatic quantum computation (AQC) was initiated several years ago by Farhi, Goldstone, Gutmann and Sipser [62], who suggested a novel quantum algorithm for solving classical optimization problems such as Satisfiability (SAT).

Their algorithm, that for what follows will abbreviated as AQC (abusing notation) and will explicitly describe later on, is based on a celebrated theorem in quantum mechanics known as the *adiabatic theorem*.

The exact worst-case behavior of AQC is not known. On one the positive side, several simulations on random instances of up to 20 quantum bits led to various optimistic speculations. Fill in modern details.

On the negative side, there is some evidence that AQC takes exponential time in the worst-case for NP-complete problems.

Nevertheless, adiabatic computation was since shown to be promising in other less ambitious directions: it possesses several interesting algorithmic capabilities, as we will soon review, and in addition, it exhibits inherent robustness against certain types of quantum errors.

7.1.2 Adiabatic Quantum Computation

Let us re-introduce AQC for another time (sorry for being repetitive): A computation in this model is specified by two Hamiltonians named H_{init} and H_{final}.

The ground state[1] of H_{init} is required to be an easy to prepare state (it can be done efficiently) and serves as the input of the computation.

The output of AQC is the ground state of the final Hamiltonian H_{final}. Hence, we choose an H_{final} whose ground state represents the solution to our problem.

Additionally, we require the Hamiltonians to be **local**[2].

This, in particular, makes sure that the Hamiltonians have a short classical description (simply by listing the matrix entries of each local term).

---

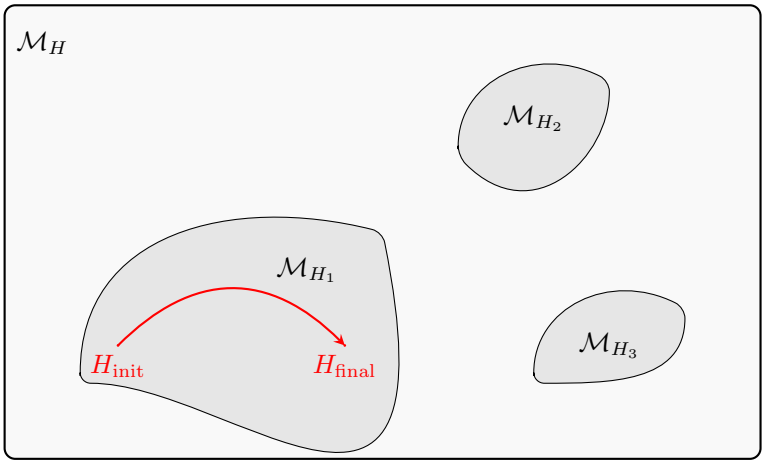
[1] Recall that this refers to the eigenvector with smallest eigenvalue of a Hamiltonian.

[2] We require them to only involve interactions between a constant number of particles (this can be seen as the equivalent of allowing gates operating on a constant number of qubits in the gate model)

The running time of the adiabatic computation is determined by the minimal spectral gap[3] of all the path connected Hamiltonians along the curve:

$$\begin{aligned} s : [0,1] &\to \mathscr{M}_H \\ H_{\text{init}} &\mapsto H_{\text{final}} \end{aligned}$$

Below we see a schematic description of the space of Hamiltonians. Note that within each disconnected component we only consider Hamiltonians related by homotopy equivalent paths.



Concretely for any $s \in [0,1]$ we have and infinite family of path parametrized Hamiltonians:

$$H(s) = (1-s)H_{\text{init}} + sH_{\text{final}} \tag{7.1}$$

and of course we are interested in reachign $s = 1$ to obtain H_{final}.

If this is done "slowly enough" we say we perform adiabatic computation and it is polynomial time if the corresponding minimal spectral gap is at least inverse polynomial.

Let us provide further motivation from a physics point of view:

- Recall that H corresponds to the energy of the quantum system.

[3] The difference between the ground state eigenenergy and the first excited state eigenenergy.

- To be physically realistic and implementable it must be local.
- Ground state of H is the state of lowest energy.
- We can set up a quantum system in the ground state of H_{init} (which is supposed to be easy to generate) and apply the Hamiltonian H_{init} to the system. We then slowly modify the Hamiltonian along the path from H_{init} towards $H_{\text{final.}}$.

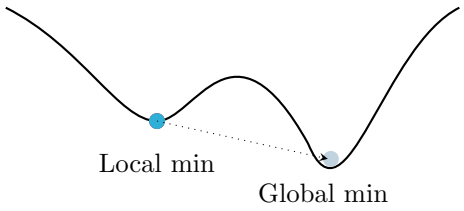
From the **adiabatic theorem** it follows that if this transformation is performed slowly enough (determined by the minimal spectral gap), the final state of the system will be in the ground state of H_{final} , as required.

What about **Computational Power** [63]? What is the computational power of this model? In order to refer to the adiabatic model as a computational model that computes classical functions (rather than quantum states), we consider the result of the adiabatic computation to be the outcome of a measurement of one or more of the qubits, performed on the final ground state. AQC is performed on qubits similar to the ones of the gate-based computers.

It is known from the very early 2000's that adiabatic computation can be efficiently simulated by standard quantum computers [64, 65]. It follows that, its computational power it cannot be greater than that of standard (gate-based) quantum computers.

Several positive results are known, regarding the paower of AQC. For example Grover search can be realized as adiabatic quantum computation [64, 66]

Moreover, AQC can "tunnel" through wide energy barriers and thus outperform simulated annealing [67].



However, whether adiabatic computation can achieve the full power of quantum computation was not known. Even whether adiabatic computation can simulate general classical computations efficiently was unknown prior to [61].

The interest for optimization problems: What was known is the potential of AQC on a restricted class of adiabatic algorithms that can be referred to as adiabatic optimization algorithms.

In these algorithms, H_{final} is chosen to be a diagonal matrix, corresponding to a combinatorial optimization problem. Being diagonal implies that the ground state

of H_{final} (the output of the computation) is a classical state, (a state in the computational basis). Nevertheless, we want to show something more powerful. We only will assume that the Hamiltonians involved are local.

For that let us discuss ***n*-qubit systems**: An n-qubit is described by a state in Hilbert space of dimension 2^n, the tensor product of 2-dimensional Hilbert spaces $\mathscr{H} = \mathbb{C}$, that is:

$$|\psi\rangle \in \mathbb{C}^{\otimes n}. \tag{7.2}$$

In terms of the individual qubits:

$$|\psi\rangle = |i_1\rangle \otimes \ldots |i_n\rangle = |i_1 \ldots i_n\rangle, \tag{7.3}$$

where $i_j \in \{0,1\}$.

Evolution. In the standard model of quantum computation, the state of n qubits evolves in discrete time steps by unitary operations. Of course, the underlying physical description of this evolution is continuous, and is governed by Schrödinger's equation:

$$\mathrm{i}\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = H|\psi(t)\rangle \tag{7.4}$$

where H is the system's Hamiltonian and $|\psi(t)\rangle$ is the state of the n qubits at time t.

We have already seen that the solution of Schrödinger equation is given as:

$$|\psi(t)\rangle = \exp(-\mathrm{i}Ht)|\psi(0)\rangle. \tag{7.5}$$

Given that the state of the system at time $t = 0$ is equal to $|\psi(0)\rangle$, one can in principle solve Schrödinger's equation with this initial condition, to get $|\psi(T)\rangle$, the state of the system at a later (terminal) time $t = T$.

Spectral Gap. We define $\Delta(H)$, the **spectral gap** of a Hamiltonian H, to be the difference between the lowest eigenvalue of H and its second lowest eigenvalue. Note that $\Delta(H) = 0$ if the lowest eigenvalue is degenerate.

What is **Locality**? One cannot efficiently apply any arbitrary Hamiltonian on a n-qubit system (just describing it requires roughly 2^{2n} space). For this we restrict to k-local Hamiltonians. We say that a Hamiltonian H is k-local if H can be written as $\sum_A H^A$ where A runs over all subsets of k particles. Notice that for any constant k, a k-local Hamiltonian on n-qubits can be described by $2^{2k}n^k = \text{poly}(n)$ numbers. We say that H is local if H is k-local for some constant k. (Commonly $k = 2$ in NISQ devices.)

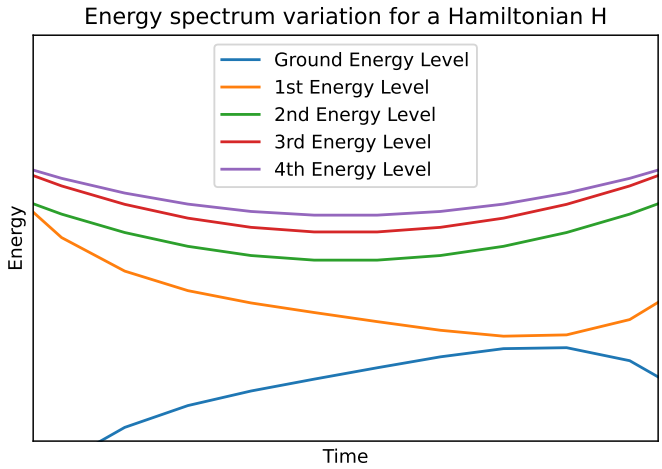
7.1.3 Adiabatic Theorem

The cornerstone of the adiabatic model of computation is the celebrated adiabatic theorem [68]. Consider a time-dependent Hamiltonian $H(s)$, and a system initialized at time $t = 0$ in the ground state of $H(0)$ (here and in the following we assume that for all $H(s)$ has a unique ground state for all s).

We let the system evolve according to the Hamiltonian $H(s)$, for $s := t/T$, from time $t = 0$ to the terminal time $t = T$. As said before, The adiabatic theorem affirms that for large enough T the final state of the system is very close to the ground state of $H(1)$.

How large T should be for this to happen is determined by the spectral gap of the Hamiltonians $\Delta(H(s))$ [61].

In the figure below we see the change of the few lowest eigenenergies for a certain evolution. It is crucial that the spectral gap does not change sign: the lowest blue eigenenergy nowhere crosses with the orange one.

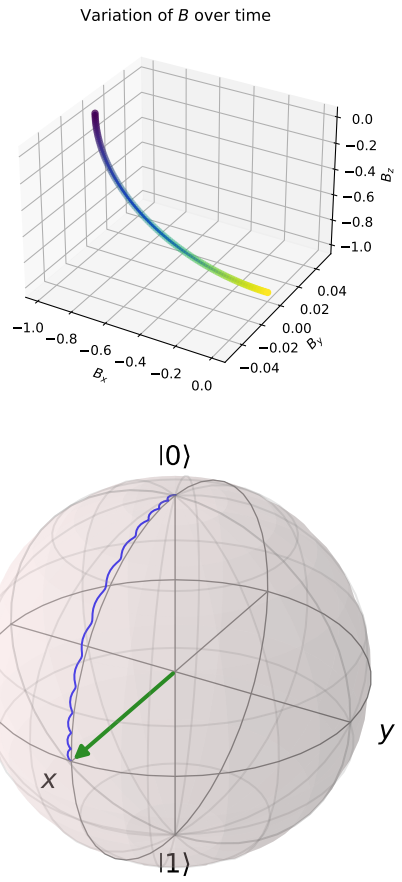


Physical intuition: Consider a spin particle (e.g. an electron) in a magnetic field B which rotates from the x direction to the z direction in a total time T. The dynamics of the particle are described by the Hamiltonian:

$$H(t) = -\cos\left(\frac{\pi t}{2T}\right)\sigma_x - \sin\left(\frac{\pi t}{2T}\right)\sigma_z. \tag{7.6}$$

Suppose that initially, the particle points in the x direction: $|\psi(0)\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, the ground state of $H(0)$. As the magnetic field is slowly rotated toward the z direc-

tion, see Fig. below, the particle's spin begins to precess about the new direction of the field, moving it toward the z axis.



Note that this produces a small wiggling component out of the xz-plane. **Adiabaticity** is seen by allowing T to be made larger and larger, so that the rotation of the field direction happens more and more slowly (as compared to the speed of precession). Then, the state will precess in a tighter and tighter orbit about the field direction (aligning completely with the geodesic). In the limit of arbitrarily slow rotation of the field, the state simply tracks the field, remaining in the instantaneous ground state of $H(t)$.

Full statement (at last): More generally, let $H(s)$ be a Hermitian operator that varies smoothly as a function of $s := t/T$. Then for T arbitrarily large, $H(t)$ varies arbitrarily slowly as a function of t. An initial quantum state $|\psi(0)\rangle$ evolves according to the Schrödinger equation (7.4), or, equivalently:

$$\mathrm{i}\frac{\mathrm{d}}{\mathrm{d}s}|\psi(s)\rangle = TH|\psi(s)\rangle. \tag{7.7}$$

Now suppose that $|\psi(0)\rangle$ is an eigenstate of $H(0)$, which we assume for simplicity is the ground state, and is nondegenerate. Furthermore, suppose that the ground state of $H(s)$ is nondegenerate for all s.

Theorem 7.4 (Adiabatic Theorem). *Given the above, in the limit $T \to \infty$, $|\psi(T)\rangle$ will be the ground state of $H(1)$.*

Remark: The proof of the adiabatic theorem is a very interesting exercise.

7.2 Sketch Proofs for the Universality of AQC

Let us repeat ousrselves (again) but a bit more formally:

Theorem 7.5 (Adiabatic Theorem (Proper)). *Let H_{init} and H_{final} be two Hamiltonians acting on a quantum system and consider the time-dependent Hamiltonian $H(s) := (1-s)\,H_{init} + sH_{final.}$. Assume that for all $s, H(s)$ has a unique ground state. Then for any fixed $\delta > 0$, if*

$$T \geq \Omega\left(\frac{\left\|H_{final} - H_{init}\right\|^{1+\delta}}{\varepsilon^{\delta}\min_{s\in[0,1]}\left\{\Delta^{2+\delta}(H(s))\right\}}\right) \tag{7.8}$$

then the final state of an adiabatic evolution according to H for time T (with an appropriate setting of global phase) is ε-close in ℓ_2-norm to the ground state of H_{final}.

The matrix norm is the spectral norm $\|H\| := \max_w \|Hw\|/\|w\|$.

The AQC model (again): Let us now describe the model of adiabatic computation. The adiabatic circuit is determined by H_{init} and H_{final} and the output of the computation is (close to) the ground state of H_{final}.

Definition 7.1. A k-local AQC $(n, d, H_{\text{init}}, H_{\text{final}}, \varepsilon)$ is specified by two k-local Hamiltonians, H_{init} and H_{final} acting on n d-dimensional particles, such that both Hamiltonians have unique ground states. The ground state of H_{init} is a tensor product state. The output is a state that is ε-close in ℓ_2-norm to the ground state of H_{final}. Let T be the smallest time such that the final state of an adiabatic evolution according to $H(s) := (1-s)H_{\text{init}} + sH_{\text{final}}$ for time T is ε-close in ℓ_2-norm to the ground state of H_{final}. The running time of the adiabatic algorithm is defined to be $T \cdot \max_s \|H(s)\|$.

7.2.1 Proof Sketch of the Equivalence 7.1

Gates to AQC

Theorem 7.1 can be proved by simulating a quantum circuit with L (two-qubit) gates on n qubits by an adiabatic computation on $n+L$ qubits.

Note that the opposite direction can also be shown [62].

We will show this by considering 5-qubit interactions. However, it is possible to reduce it to three. (Note that the practical implementation of 5-qubit interactions is still not easy.)

A Theorem:

Theorem 7.6. *Given a quantum circuit on n qubits with L two-qubit gates implementing a unitary U and $\varepsilon > 0$, there exists a 5-local adiabatic computation $(n+2, 2, H_{\text{init}}, H_{\text{final}}, \varepsilon)$ whose running time is $\text{poly}(L, 1/\varepsilon)$ and whose output is ε-close to $U|0\rangle^n = U|0\rangle^{\otimes n}$. Additionally, H_{init} and H_{final} can be computed by a polynomial time Turing machine.*

The Hamiltonian: The Hamiltonian we need is defined in [69]. We begin by defining a state

$$|\gamma_\ell\rangle := |\alpha(\ell)\rangle \otimes |1^\ell 0^{L-\ell}\rangle^c. \tag{7.9}$$

Here $|\alpha(\ell)\rangle$ denotes the state of the circuit after the application of the ℓ-th gate (and the superscript c denotes the clock qubits required for the proof of the theorem). The notation $|1^\ell 0^{L-\ell}\rangle$ means that there are ℓ qubits in the state $|1\rangle$ followed by $(L-\ell)$ qubits in the state $|0\rangle$.

We now define the Hamiltonian H_{init} with ground state $|\gamma_0\rangle = |0^n\rangle \otimes |0^L\rangle^c$,

and the local Hamiltonian H_{final} with ground state $|\eta\rangle = \frac{1}{\sqrt{L+1}} \sum_{\ell=0}^{L} |\gamma_\ell\rangle$.

We know what our initial and final eigenvectors need be. It turns out that the way to do it is:

$$\begin{aligned} H_{\text{init}} &:= H_{\text{clock init}} + H_{\text{input}} + H_{\text{clock}} \\ H_{\text{final}} &:= \frac{1}{2} \sum_{\ell=1}^{L} H_\ell + H_{\text{input}} + H_{\text{clock}} \end{aligned} \tag{7.10}$$

Note that the terms in the two Hamiltonians are defined such that the only state whose energy is 0 is the desired ground state.

This is done by assigning an energy penalty to any state that does not satisfy the required properties of the ground state. The different terms, which correspond to different properties of the ground states, are described in the following paragraphs.

Adiabatic Evolution: The adiabatic evolution then follows the time-dependent Hamiltonian

$$H(s) = (1-s)H_{\text{init}} + sH_{\text{final}} \tag{7.11}$$

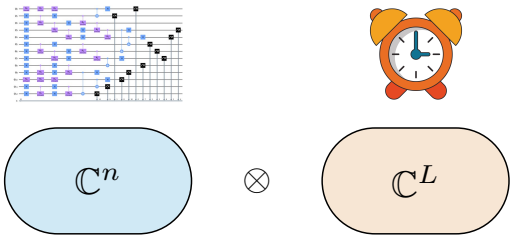
Notice that as s goes from 0 to 1, $H_{\text{clock init}}$ is slowly replaced by $\frac{1}{2}\sum_{\ell=1}^{L} H_\ell$ while H_{input} and H_{clock} are held constant.

The Hamiltonians Explained: H_{clock} First, H_{clock} checks that the clock's state is of the form $|1^\ell 0^{L-\ell}\rangle^c$ for some $0 \leq \ell \leq L$ (thus "clock").

To do this we give a penalty to any state (of the clock register) that contain a sequence 01, that is:

$$H_{\text{clock}} := \sum_{\ell=1}^{L-1} |01\rangle\langle 01|^c_{\ell,\ell+1}. \tag{7.12}$$

Below, we present a schematic of the Hilbert space of our system. Note that there is a typo in the figure, it should read $\mathbb{C}^{2n}$ and $\mathbb{C}^{2L}$.



The subscript indicates which clock qubits the projection operates on. The term $|01\rangle\langle 01|^c_{\ell,\ell+1}$ has an energy penalty (eigenvalue 1) when the clock qubits at positions ℓ and $\ell+1$ are in the state $|01\rangle$, and no energy penalty (eigenvalue 0) otherwise. By summing these terms over all pairs of adjacent clock qubits (from $\ell = 1$ to $L-1$), the clock Hamiltonian enforces a penalty whenever there is an out-of-order pair of clock qubits.

This means that the lowest energy states of the clock Hamiltonian correspond to the correct progression of the computation, where the clock qubits represent a valid encoding of the computation stages. In this way, H_{clock} helps control the adiabatic evolution of the quantum system and ensures that it follows the desired gate sequence.

The Hamiltonians Explained: H_{init} H_{input} checks that if the clock is at $|0\rangle^{\otimes L}$ (we ommited the c-clock index here, clearly referring to $\mathscr{H}_{\text{clock}}$) then the computation qubits must be in the state $|0\rangle^{\otimes n}$. This is given by:

$$H_{\text{init}} := \sum_{i=1}^{n} |1\rangle\langle 1| \otimes |0\rangle\langle 0|. \tag{7.13}$$

Let us discuss $H_{\text{clock init}}$. The goal of $H_{\text{clock init}}$ is to check that the clock's state is $|0\rangle^{\otimes L}$:

$$H_{\text{clock init}} := |1\rangle\langle 1|. \tag{7.14}$$

Finally, we have the term

$$\frac{1}{2}\sum_{\ell=1}^{L} H_\ell \tag{7.15}$$

which is the term representing the gate-based Hamiltonian and it is only apparent in the end of the AQC (in principle it is unknown).

Summary: $H_{\text{clock init}}$ and H_{clock}: These terms are related to the clock qubits. $H_{\text{clock init}}$ sets the initial state of the clock qubits and ensures that the computation starts with all clock qubits in the state $|1\rangle^c$. H_{clock} penalizes out-of-order transitions and enforces a step-by-step progression through the circuit.

H_{input}: This term sets the initial state of the quantum circuit. It essentially encodes the input data of the problem you want to solve.

$\frac{1}{2}\sum_{\ell=1}^{L} H_\ell$: This term is present only in the final Hamiltonian, H_{final}. It represents the quantum gates in the circuit. The factor $\frac{1}{2}$ ensures that the spectrum of the Hamiltonian is non-negative, which is a requirement for the adiabatic theorem to hold.

The final Hamiltonian: We now proceed to the first term in H_{final} . The Hamiltonian H_ℓ checks that the propagation from step $\ell-1$ to ℓ is correct. It checks that it corresponds to the application of the gate U_ℓ.

For $1 < \ell < L$, it is defined as:

$$\begin{aligned} H_\ell :=& \mathbf{1} \otimes |100\rangle\langle 100|^c_{\ell-1,\ell,\ell+1} - U_\ell \otimes |110\rangle\langle 100|^c_{\ell-1,\ell,\ell+1} \\ & - U_\ell^\dagger |100\rangle\langle 110|^c_{\ell-1,\ell,\ell+1} + \mathbf{1} \otimes |110\rangle\langle 110|^c_{\ell-1,\ell,\ell+1}. \end{aligned} \tag{7.16}$$

Intuitively, the three-qubit terms above move the state of the clock one step forward, one step backward, or leave it unchanged (this reminds us a quantum walk). The accompanying matrices $U_\ell, U_\ell^\dagger$ describe the associated time evolution. We have two

boundary cases $\ell = 1, L$ (initial and terminal time) for which we omit one clock qubit from these terms and define

$$H_1 := \mathbf{1} \otimes |00\rangle\langle 00|_{1,2} - U_1 \otimes |10\rangle\langle 00|_{1,2} - U_1^\dagger \otimes |00\rangle\langle 10|_{1,2} + \mathbf{1} \otimes |10\rangle\langle 10|_{1,2} \tag{7.17}$$

$$H_L := = \mathbf{1} \otimes |10\rangle\langle 10|_{L-1,L} - U_L \otimes |11\rangle\langle 10|_{L-1,L} - U_L^\dagger \otimes |10\rangle\langle 11|_{L-1,L} + \mathbf{1} \otimes |11\rangle\langle 11|. \tag{7.18}$$

Spectral gap inverse in L

We have now seen what are the Hamiltonians needed to transform a gate-based problem to an AQC.

We need to understand the spectral gap now.

Recall the state given by Eq. (7.9):

$$|\gamma_\ell\rangle := |\alpha(\ell)\rangle \otimes |1^\ell 0^{L-\ell}\rangle^c.$$

Spectral gap inverse in L: $s > 1/3$

Let S_0 a subspace of $\mathbb{C}^n \otimes \mathbb{C}^L$ spanned by

$$\{|\gamma_0\rangle, \ldots, |\gamma_1\rangle\} \tag{7.19}$$

which are equivariant states (w.r.t. the action of Hamiltonians on S). In other words, we have some form of symmetry.

Theorem 7.7. *The spectral gap of the restriction of $H(s)$ to S_0 satisfies:*

$$\Delta(H_{S_0}(s)) = \Omega(L^{-2}), \tag{7.20}$$

for all $s \in [0,1]$.

Interestingly, the proof uses a continuous-time quantum walk. The proof is technical (not very hard) but we omit it here. The important thing is to understand the need for the Hamiltonians H_{init} and H_{final} in Eq. (7.10).

With the proof on the (inverse in L) polynomial runtime, we claim the following.

The Equivalence Statement: Given a quantum circuit on n qubits with L gates, the quantum adiabatic algorithm with H_{init} and H_{final} as defined in the previous slides, with $T = \mathcal{O}(\varepsilon^{-\delta} L^{4+2\delta})$, for fixed $\delta > 0$, outputs a final state $|\eta\rangle$ that is within ℓ_2 distance ε of the history state of the circuit. The running time of the AQC algorithm is $\mathcal{O}(TL)$.

Recall, that already from 2000 it was known that gate-based algorithms can be encoded as AQC. With the proof of Theorem 7.1, the universality of AQC is also proven. A detailed introduction with many complexity theoretical aspects is [70].

7.3 Practicalities of Adiabatic Quantum Computing

We have discussed that the solution of computational problem can be encoded into the ground state of a time-dependent quantum Hamiltonian $H(s)$ which evolves following the paradigm of AQC.

Quantum annealing (QA) is a framework that incorporates algorithms [71, 72, 73] and hardware designed to solve computational problems by quantum evolution towards the ground states of final Hamiltonians that encode classical (optimization) problems.

Note that Quantum Annealers are real:

This is the D-Wave 2000Q system. This is a system that performs quantum annealing using superconducting qubits. The qubits live in the very end of a dilution refridgerator cooled at approximately -273.5 degrees Celcius.

7.3.1 Stoquasticity

QA therefore, moves between the idealized assumptions of universal AQC and the unavoidable experimental compromises.

Perhaps the most significant of these compromises has been the design of stoquastic quantum annealers.

Definition 7.2 (Stoquastic Hamiltonian). A Hamiltonian H is called stoquastic, with respect to a basis B, if and only if H has real nonpositive off-diagonal matrix elements in the basis B.

For example, a Hamiltonian is stoquastic if and only

$$\langle i|H|j\rangle \leq 0, \quad \forall i,j \in \{0,1\}^n, \quad i \neq j. \tag{7.21}$$

This means the ground state of H can be expressed as a classical probability distribution.

AQC with Stoquastic Hamiltonians:

Definition 7.3. Stoquastic adiabatic quantum computation (StoqAQC) is the special case of AQC restricted to k-local (k fixed) stoquastic Hamiltonians.

Essentially, Quantum Annealing (QA) refers to StoqAQC when considered in (realistic) open quantum systems. For what follows, these two terms are identical.

No Universality: The computational power of stoquastic Hamiltonians has been carefully studied, and is suspected to be limited in the ground-state AQC setting [70].

In other words, **it is quite unlikely that ground-state StoqAQC is universal** [74].

7.3.2 Quantum Annealing is very similar to AQC

QA follows the same idea of AQC. We still have the same tools:

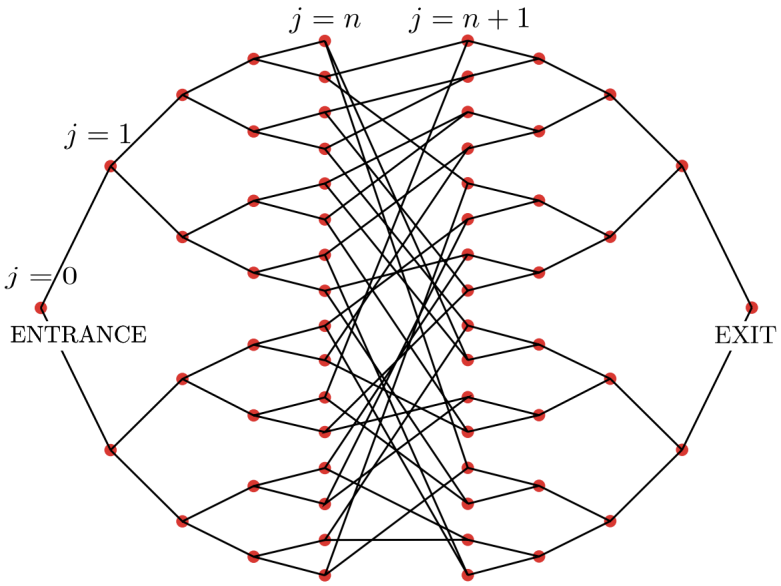
- An initial, easy-to-prepare state and a Hamiltonian H_{init},
- A problem of interest whose solution is encoded into the ground state of a Hamiltonian H_{final},
- Adiabatic evolution using Eq. (7.1):

$$H(s) = (1-s)H_{\text{init}} + sH_{\text{final}} \tag{7.22}$$

Exponential Speedups with QA It turns out that QA can be used to obtain exponential speedups!

Somma, Nagaj, and Kieferovaá [75] showed that, similarly to the case of quantum walks, utilizing QA on the glued-trees problem one obtains an exponential speedup.

Glued-Trees Problem



In this problem we are given an oracle O_A that concists of the adjacency matrix A of two binary trees that are randomly glued. There are $\mathcal{O}(2^n)$ vertices named with randomly chosen $2n$-strings.

The oracle O_A outputs the names of the adjacent vertices on any given input vertex name.

There are two special vertices:

- ENTRANCE
- EXIT

which are the roots of the binary trees. They can be identified because they are the only vertices of degree two in the graph.

Glued-Trees Problem: Given an oracle O_A for the graph and the name x of the ENTRANCE, find the name y of the EXIT.

An efficient method based on quantum walks can solve this problem with constant probability, while no classical algorithm that uses less than a subexponential (in n) number of oracles exists.

7.3.3 Optimization

An **optimization problem** is a problem to minimize or maximize a real single-valued function of multivariables called the cost function.

If the problem is to maximize the cost function f, it suffices to minimize $-f$.

Additional constraints can be imposed on the objective function:

$$\min_{x,y} \quad f(x,y) \tag{7.23}$$

$$\text{s.t.} \quad g(x) \geq 0 \tag{7.24}$$

$$x \in \mathbb{R}^m, y \in \mathbb{Z}^n \tag{7.25}$$

(Just a reminder) Optimization problems are classified roughly into two types, easy and hard ones. Loosely speaking, easy problems are those for which we have algorithms to solve in steps(=time) polynomial in the system size (polynomial complexity). In contrast, for hard problems, all known algorithms take exponentially many steps to reach the exact solution (exponential complexity). A potential solution is offered by Quantum Annealing.

Suppose we can solve such problems with QA. Does it converge?

Consider the k-th eigenstate state of the Hamiltonian:

$$H(s)|k\rangle = \lambda_k(s)|k\rangle \tag{7.26}$$

with $|0(0)\rangle$ being the ground state of H_{init} and generically $|0(s)\rangle$ the ground state of $H(s)$.

If $|0(s)\rangle$ is non-degenerate and if initial ground state is $|0(0)\rangle$ then the final state vector, at large T, take the form:

$$|\psi(s)\rangle = \sum_{\kappa} c_\kappa(s) e^{-\mathrm{i}T\phi_\kappa(s)} |\kappa(s)\rangle \tag{7.27}$$

with

$$\phi_\kappa(s) = \int_0^s \lambda_\kappa(s')\mathrm{d}s'.$$

Maybe some homework on the topic? It turns out:

$$c_0(s) \approx 1 + \mathcal{O}(T^{-2}), \tag{7.28}$$

$$c_{\kappa\neq 0}(s) \approx \frac{\mathrm{i}}{T}\left[A_\kappa(0) - \mathrm{e}^{\mathrm{i}T[\phi_\kappa(s)-\phi_0(s)]}A_\kappa(s)\right] + \mathcal{O}(T^{-2}) \tag{7.29}$$

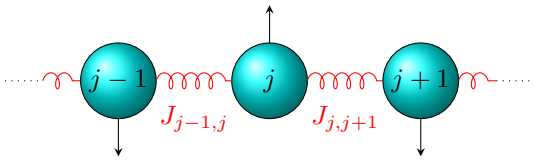
The adiabaticity condition becomes:

$$\frac{1}{\Delta_\kappa(t)^2}\left|\left\langle \kappa(t)\left|\frac{\mathrm{d}H(t)}{\mathrm{d}t}\right|0(t)\right\rangle\right| = \delta \ll 1. \tag{7.30}$$

Convergence via Ising model: Suppose that the optimization (7.23) problem we wish to solve can be represented as the ground-state search of an Ising model of general form

$$H_{\mathrm{Ising}} \equiv -\sum_{i=1}^{N} J_i \sigma_i^z - \sum_{i,j=1}^{N} J_{ij}\sigma_i^z\sigma_j^z + \mathscr{O}(\sigma^3). \tag{7.31}$$

Here, $\sigma_i^\alpha(\alpha = x,y,z)$ are the Pauli matrices.



Recall, the eigenvalue of σ_i^z is $+1$ or -1, which corresponds the classical Ising spin.

Most combinatorial optimization problems can be written in this form by, for example, mapping binary variables $\{0,1\}$ to spin variables $\{\pm 1\}$. Another important assumption is that the Hamiltonian (7.31) is proportional to the number of spins N for large N.

Transverse Field To realize QA, a (kinetic) energy term is introduced typically by the so-called time-dependent transverse field:

$$H_{\mathrm{TF}}(t) \equiv -\Gamma(t)\sum_{i=1}^{N}\sigma_i^x \tag{7.32}$$

which results in a variety of possible quantum mechanical effects to the chain:spin flips, quantum fluctuations or quantum tunneling, between the two states $\sigma_i^z = 1$ and $\sigma_i^z = -1$.

Essentially this allows a quantum search of the phase space of the system.

Initially the strength of the transverse field $\Gamma(t)$ is chosen to be very large, and the total Hamiltonian

$$H(t) = H_{\mathrm{Ising}} + H_{\mathrm{TF}}(t) \tag{7.33}$$

is dominated by the second kinetic term. (If you know about Simulated Annealing (SA), this is the quantum analogue of the high-temperature limit.)

The evolution of the TF Ising Model: The coefficient $\Gamma(t)$ is then gradually and monotonically decreased toward 0, leaving eventually only the potential term H_{Ising} .

Accordingly the state vector $|\psi(t)\rangle$, which follows the real-time Schrödinger equation, is expected to evolve from the trivial initial ground state of the transverse-field (7.33) to the non-trivial ground state of (7.31), which is the solution of the optimization problem.

An important issue is *how slowly we should decrease* $\Gamma(t)$ *to keep the state vector arbitrarily close to the instantaneous ground state of the total Hamiltonian*.

The following Theorem provides a solution to this problem as a sufficient condition.

Theorem 7.8. *The adiabaticity* (7.30) *for the transverse-field Ising model* (7.31) *yields the time dependence of* $\Gamma(t)$ *as*

$$\Gamma(t) = a(\delta t + c)^{-1/(2N-1)} \tag{7.34}$$

for $t > t_0$ *(for given* $t_0 > 0$*) as a sufficient condition of convergence of QA. Here* a, c *are small constants* $\mathcal{O}(1)$ *and* δ *is a small parameter that controls adiabaticity.*

Point is: **The power decay above satisfies the adiabaticity condition** (7.30) **which guarantees convergence to the ground state of** H_{Ising} **as** $t \to \infty$.

QA in Practice: Optimization

In practical situations QA is used as heuristic optimization method.

Due to hardware constructions, at the moment only Quadratic Binary Optimization (QUBO) problems can be implemented.

A QUBO problem reads

$$\min_{x \in \{0,1\}^N} Q(x) \tag{7.35}$$

where the objective function Q is defined as:

$$Q(x) := \sum_{i,j=1}^{N} Q_{ij} x_i x_j + \sum_{i=1}^{N} c_i x_i. \tag{7.36}$$

The problem to be optimized is then fully specified by Q_{ij} and c_i.

A broad class of paradigmatic optimization problems from Vertex Cover to the Traveling Salesperson problem have been mapped to QUBO form.

What happens if $k \geq 3$? (That is, if we have problem with interactions of degree 3 or higher.) If the problem of interest has a cost function of high-order interactions, than the quadratic, one has to encode this information in ancilla qubits.

For example, assume a problem encoding involves the 3-local expression

$$xyz, \quad x, y, z \in \mathbb{R}.$$

This has to be mapped to the expression

$$xw,$$

where $w := yz$ and impose the additional constraint

$$3w + yz - 2yw - 2zw.$$

The solution is (zero penalization) $w = yz$.

Example: The Knapsack Problem:

We are given a set of weights $w \in \mathbb{Z}^n_{\geq 0}$ and their corresponding values $v \in \mathbb{Z}^n_{\geq 0}$, and the objective is to maximize the total value of the items that can be packed into a knapsack subject to a given weight limit W.

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} v_i x_i, \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i x_i \leq W, \end{aligned} \tag{7.37}$$

where W is the maximum weight limit (threshold) of the knapsack and x_i is the binary variable representing whether the i-th item is to be placed in the knapsack.

MILP to QUBO: In converting MILPs to QUBOs we introduce a slack variable S for each linear inequality and transform it into an equivalent linear equality. We add to the objective a penalty term:

$$\lambda_0 \left(\sum_{i=1}^{n} w_i x_i - W + S \right)^2 \tag{7.38}$$

where the purpose of the auxiliary slack variable S is to reduce this term to 0 once the constraint has been satisfied, $0 \leq S \leq \max_x \sum_i^n w_i x_i - W$.

Note that in practice, S is decomposed into binary representation using variables $s_k \in \{0, 1\}$ as follows:

$$S = \sum_{k=1}^{N_s} 2^{k-1} s_k. \tag{7.39}$$

The parameter N_s corresponds to the number of binary variables required to represent the maximum value that can be assigned to the slack variable, and in the case of Knapsack, $N_s = \lceil \log_2(W) \rceil$, where $\lceil x \rceil$ is the ceiling function. This is often called "log-encoding".

The QUBO formulation: the Knapsack problem can be formulated then as:

$$\max \quad \sum_i^n v_i x_i - \lambda_0 \left(\sum_i^n w_i x_i - W + \sum_{k=1}^N 2^{k-1} s_k \right)^2, \tag{7.40}$$

Maping to the Ising model:

$$\min \quad -\left(\sum_{i=1}^n \sum_{j=1}^n J_{ij} s_i s_j + \sum_{i=1}^n h_i s_i + c \right) \tag{7.41}$$

where

$$J_{ij} = \lambda_0 2^{k-1} w_i \delta_{ij}, \tag{7.42}$$

$$h_i = \frac{v_i}{2} - \lambda_0 w_i W, \tag{7.43}$$

$$c = \sum_{i=1}^n \frac{v_i}{2} + \lambda_0 \left(\sum_{i=1}^n \frac{w_i^2}{4} + \sum_{k=1}^N 2^{2k-2} \right). \tag{7.44}$$

Does QA Fail?

Adiabatic Quantum Optimization Fails to Solve the Knapsack Problem

Lauren Pusey-Nazzaro
Department of Physics
Washington University
St. Louis, MO
lauren.p@wustl.edu

Prasanna Date
Computer Science and Mathematics
Oak Ridge National Laboratory
Oak Ridge, TN
datepa@ornl.gov

Can QA Succeed?

Article | Published: 19 April 2023

Quantum critical dynamics in a 5,000-qubit programmable spin glass

Andrew D. King, Jack Raymond, Trevor Lanting, Richard Harris, Alex Zucca, Fabio Altomare, Andrew J. Berkley, Kelly Boothby, Sara Ejtemaee, Colin Enderud, Emile Hoskinson, Shuiyuan Huang, Eric Ladizinsky, Allison J. R. MacDonald, Gaelen Marsden, Reza Molavi, Travis Oh, Gabriel Poulin-Lamarre, Mauricio Reis, Chris Rich, Yuki Sato, Nicholas Tsai, Mark Volkmann, Jed D. Whittaker, ... Mohammad H. Amin + Show authors

7.4 Variational Quantum Algorithms

Two fundamental references on VQAs are the original paper on the Variational Quantum Eigensolver [76] and the subsequent paper on the Quantum Approximate Optimization Algorithm [77]. A beautiful survey is given by [78].

Parametrized Quantum Circuits (PQCs): Variational Quantum Algorithms (VQAs) provide a general framework that can be used to solve a variety of problems.

For that we first need the idea of a parametrized quantum circuit.

Definition 7.4. A parametrized quantum circuit (PQC) is a continuous function $U : \mathbb{R}^L \to U(N)$ mapping any real parameter vector $\vartheta \in \mathbb{R}^L$ to a unitary $U(\vartheta)$.

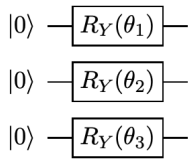
In practice such a quantum circuit is a sequence of universal quantum gates' compositions and/or tensor products.

Consider, for a moment, the following optimization problem (and keep it in mind):

$$\min_{x\in\{0,1\}^n} \quad f(x). \tag{7.45}$$

A VQA is, essentially, a (quantum) continuous relaxation of this problem.

Below we see a PQC example:

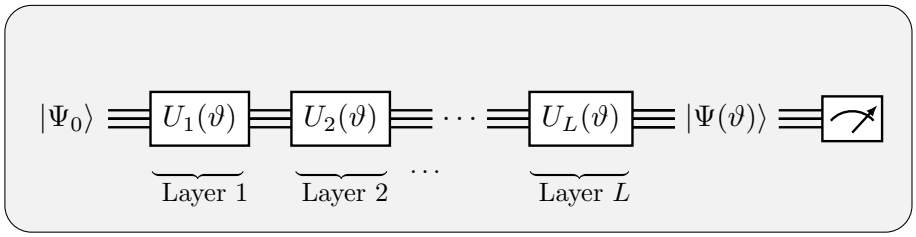


Here $U(\boldsymbol{\theta})$ is given as:

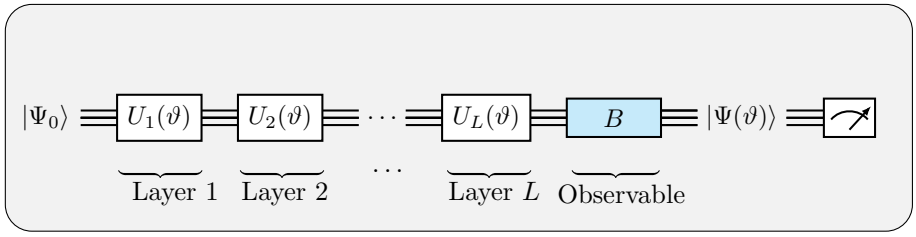
$$
\begin{aligned}
U(\theta) &= R_Y(\theta_1) \otimes R_Y(\theta_2) \otimes R_Y(\theta_3) \\
&= \begin{pmatrix} \cos\frac{\theta_1}{2} & -\sin\frac{\theta_1}{2} \\ \sin\frac{\theta_1}{2} & \cos\frac{\theta_1}{2} \end{pmatrix} \otimes \begin{pmatrix} \cos\frac{\theta_2}{2} & -\sin\frac{\theta_2}{2} \\ \sin\frac{\theta_2}{2} & \cos\frac{\theta_2}{2} \end{pmatrix} \otimes \begin{pmatrix} \cos\frac{\theta_3}{2} & -\sin\frac{\theta_3}{2} \\ \sin\frac{\theta_3}{2} & \cos\frac{\theta_3}{2} \end{pmatrix} \\
&= \begin{pmatrix}
c_1c_2c_3 & -c_1c_2s_3 & -c_1s_2c_3 & c_1s_2s_3 & -s_1c_2c_3 & s_1c_2s_3 & s_1s_2c_3 & -s_1s_2s_3 \\
c_1c_2s_3 & c_1c_2c_3 & -c_1s_2s_3 & -c_1s_2c_3 & -s_1c_2s_3 & -s_1c_2c_3 & s_1s_2s_3 & s_1s_2c_3 \\
c_1s_2c_3 & -c_1s_2s_3 & c_1c_2c_3 & -c_1c_2s_3 & -s_1s_2c_3 & s_1s_2s_3 & -s_1c_2c_3 & s_1c_2s_3 \\
c_1s_2s_3 & c_1s_2c_3 & c_1c_2s_3 & c_1c_2c_3 & -s_1s_2s_3 & -s_1s_2c_3 & -s_1c_2s_3 & -s_1c_2c_3 \\
s_1c_2c_3 & -s_1c_2s_3 & -s_1s_2c_3 & s_1s_2s_3 & c_1c_2c_3 & -c_1c_2s_3 & -c_1s_2c_3 & c_1s_2s_3 \\
s_1c_2s_3 & s_1c_2c_3 & -s_1s_2s_3 & -s_1s_2c_3 & c_1c_2s_3 & c_1c_2c_3 & -c_1s_2s_3 & -c_1s_2c_3 \\
s_1s_2c_3 & -s_1s_2s_3 & s_1c_2c_3 & -s_1c_2s_3 & c_1s_2c_3 & -c_1s_2s_3 & c_1c_2c_3 & -c_1c_2s_3 \\
s_1s_2s_3 & s_1s_2c_3 & s_1c_2s_3 & s_1c_2c_3 & c_1s_2s_3 & c_1s_2c_3 & c_1c_2s_3 & c_1c_2c_3
\end{pmatrix}
\end{aligned}
$$

for $\theta = (\theta_1, \theta_2, \theta_3) \in \mathbb{R}^3$, where $c_i = \cos\frac{\theta_i}{2}$ and $s_i = \sin\frac{\theta_i}{2}$ for $i = 1,2,3$.

Generically: The quantum part of a VQA has the following form:



More precisely, we can explicitly include the observable we want to measure:



VQAs: The Quantum Part Given a PQC with $\vartheta \in \mathbb{R}^L$ we can define a cost function

$$B(\vartheta) = f\Big(\{|\Psi\rangle_0\}, \{B_k\}, U(\vartheta)\Big). \tag{7.46}$$

Essentially, this is a cost function that involves (some) obsevable quantity - Hermitian operators $\{O_k\}$ given input states $\{|\Psi\rangle_0\}$ and the PQC $U(\vartheta)$. Here $k \in I$, where I is some indexed set.

Let $\rho_{\text{in}} := |\Psi\rangle_0\langle\Psi|_0$ (assume norm 1). A common choice is (using Born's rule) to define the "observable" function

$$B(\vartheta) = \sum_{k\in I} \text{Tr}\Big(B_k U(\vartheta)\rho_{\text{in}} U^\dagger(\vartheta)\Big), \tag{7.47}$$

or more generically

$$B(\vartheta) = \sum_{k\in I} f_k\left(\text{Tr}\Big(B_k U(\vartheta)\rho_{\text{in}} U^\dagger(\vartheta)\Big)\right), \tag{7.48}$$

for some functions f_k.

VQAs: Measurements: The observable function is one that we end up measuring (several times) in order to construct an empirical estimate of its expectation value $\langle B\rangle_\vartheta$ of the observable:

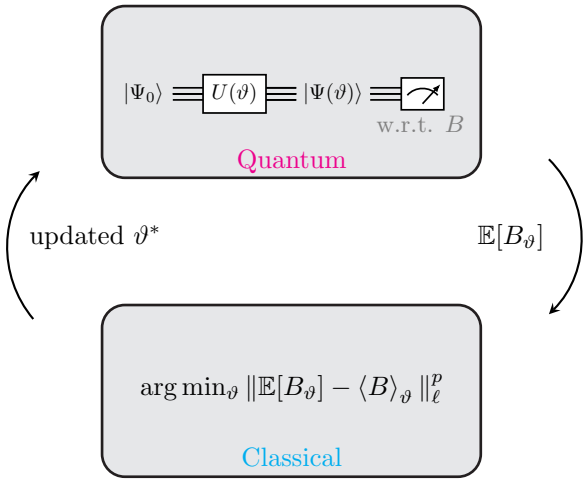
$$\langle B\rangle_\vartheta := \langle\Psi(\vartheta)|B|\Psi(\vartheta)\rangle, \tag{7.49}$$

where $|\Psi(\vartheta)\rangle := U(\vartheta)|\Psi_0\rangle$. The empirical estimate

$$\mathbb{E}[B_\vartheta]. \tag{7.50}$$

This is constructed by measuring the same circuit repeatedly. Out of this we construct a cost function we would like to minimize:

$$\vartheta^* := \arg\min_\vartheta \|\mathbb{E}[B_\vartheta] - \langle B\rangle_\vartheta\|_\ell^p \tag{7.51}$$



VQA: The Classical Part: During the optimization, one uses a finite statistic estimator of the cost or its gradients.

Essentially we are "training" the VQA by learning the parameters ϑ.

It is known that for many optimization tasks using information in the cost function gradient can help in speeding up and guaranteeing the convergence of the optimizer. One of the main advantages of many VQAs is that often one can analytically evaluate the cost function gradient.

Parameter Shift Rule: Consider a cost function as in Eq. (7.48):

$$B(\vartheta) = \mathrm{Tr}\Big(BU(\vartheta)\rho_{\mathrm{in}}U^{\dagger}(\vartheta)\Big), \tag{7.52}$$

($f_k = \mathrm{Id}$, $k = 1$). Furthermore, let the unitaries read:

$$U(\vartheta_j) = e^{\mathrm{i}\vartheta_j \sigma_j^a}. \tag{7.53}$$

Then:

$$\frac{\partial B(\vartheta)}{\partial \vartheta_j} \sim \frac{1}{\sin\alpha}\big(\mathrm{Tr}(BU^{\dagger}(\vartheta_+)\rho U(\vartheta_+)) - \mathrm{Tr}(BU^{\dagger}(\vartheta_-)\rho U(\vartheta_-)\big) \tag{7.54}$$

where $\vartheta_{\pm} = \vartheta \pm \alpha e$. Here e_j is a vector having 1 as its j-th element and 0 otherwise. Thus, one can evaluate the gradient by shifting the l-th parameter by some amount α.

Note that the accuracy of the evaluation depends on the coefficient $1/(2\sin\alpha)$ since each of the $\pm\alpha$ terms are evaluated by sampling B. This accuracy is maximized at $\alpha = \pi/4$. So, the parameter-shift rule looks like some form of finite difference. However, it evaluates the analytic gradient of the parameter by virtue of the coefficient $1/(2\sin\alpha)$. Note that several variations of the parameter shift rule exist [79, Fig. 1].

It's hard to train VQAs

Quantum Physics

[Submitted on 18 Jan 2021 (v1), last revised 14 Apr 2022 (this version, v2)]

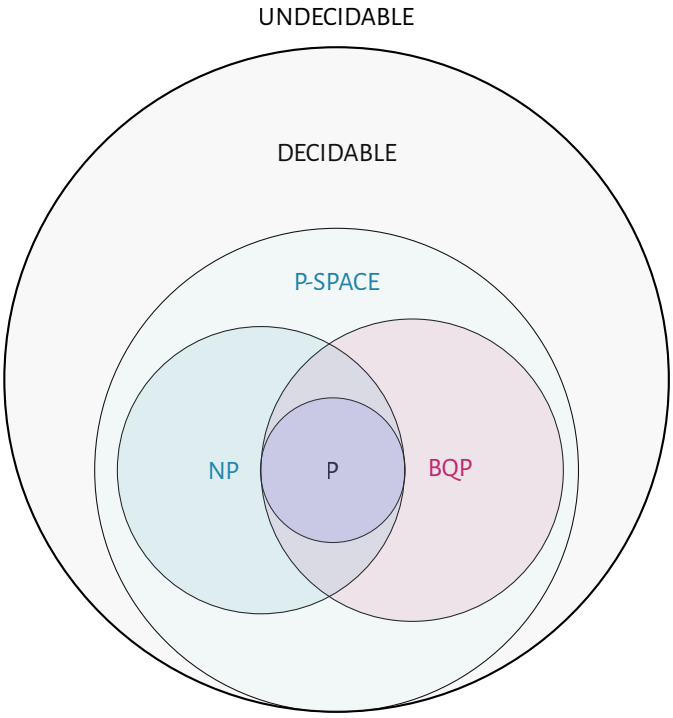
Training variational quantum algorithms is NP-hard

Lennart Bittel, Martin Kliesch

Variational quantum algorithms are proposed to solve relevant computational problems on near term quantum devices. Popular versions are variational quantum eigensolvers and quantum ap- proximate optimization algorithms that solve ground state problems from quantum chemistry and binary optimization problems, respectively. They are based on the idea of using a classical computer to train a parameterized quantum circuit. We show that the corresponding classical optimization problems are NP-hard. Moreover, the hardness is robust in the sense that, for every polynomial time algorithm, there are instances for which the relative error resulting from the classical optimization problem can be arbitrarily large assuming P ≠ NP. Even for classically tractable systems composed of only logarithmically many qubits or free fermions, we show the optimization to be NP-hard. This elucidates that the classical optimization is intrinsically hard and does not merely inherit the hardness from the ground state problem. Our analysis shows that the training landscape can have many far from optimal persistent local minima. This means that gradient and higher order descent algorithms will generally converge to far from optimal solutions.

Undecidability conjecture

I conjecture that actually the situation is worse. VQAs are undecidable.



Training VQAs: The success of a VQA depends on the efficiency and reliability of the optimization method used.

As we saw the training can be NP-Hard. Training a VQA one can encounter new challenges:

- huge number of local minima
- barren plateaus
- stochastic environment due to the finite budget for measurements
- hardware noise affecting $\mathbb{E}[B_{\vartheta}]$
- restricted qubit connectivity
- statepreparation-and-measurement (SPAM) errors
- ...

This has led to the development of many quantum hardware-aware optimizers, with the optimal choice still being an active topic of debate. A common choice is the family of SGD (e.g. SPSA).

7.5 QAOA

Quantum Approximation Optimization Algorithm (QAOA) can be implemented in NISQ devices.

QAOA is an approximation algorithm: it does not deliver the "best" result, but only the "good enough" result, which is characterized by a lower bound of the approximation ratio.

Interestingly QAOA can be applied to the MaxCut problem via a traverse Ising filed model.

Trotterization: Recall that in the case of AQC we have:

$$H(s) = (1-s)H_{\text{init}} + sH_{\text{final}}. \tag{7.55}$$

Time evolution under this *time-dependent* Hamiltonian involves is hard:

$$U(T) \sim \exp\left(-\mathrm{i}\int_0^t H(w)\mathrm{d}w\right). \tag{7.56}$$

We can mitigate this with the Trotterization technique: discretize $U(T) \equiv U(T,0)$ into intervals Δt (in total $T = L\Delta t$) small enough that the Hamiltonian is approximately constant over each interval. Then:

$$U(T,0) = U(T,T-\Delta t)U(T-\Delta t,T-2\Delta t)\ldots U(\Delta t,0) \tag{7.57}$$

$$= \prod_{j=0}^{L-1} U((L-j)\Delta T,(L-j-1)\Delta t) \tag{7.58}$$

$$=_{L\to\infty} \prod_{j=0}^{L-1} e^{-\mathrm{i}H[(L-j)\Delta t]\Delta t} \tag{7.59}$$

Using the identity

$$e^{\mathrm{i}(A+B)x} = e^{\mathrm{i}(A)x}e^{\mathrm{i}(B)x} + \mathcal{O}(x^2) \tag{7.60}$$

we deduce that

$$U(T,0) \approx \prod_{j=1}^{p} e^{\{-\mathrm{i}(1-s(j\Delta t))H_{\text{init}}\Delta t\}}e^{\{-\mathrm{i}s(j\Delta t)H_{\text{final}}\Delta t\}}. \tag{7.61}$$

Thus we can approximate AQC by repeatedly letting the system evolve under H_{final} for $s(j\Delta t)$ and then under H_{init} for $(1-s(j\Delta t))$. In this way we can construct arbitrary unitaries.

QAOA: Combinatorial Optimization: Recall that a combinatorial optimization problem amounts to finding the n-bit string z that (approximately) satisfies the maximal amount of m constraints C_α, each of which takes the form

$$C_\alpha(z) = \begin{cases} 1 \text{ if } z \text{ satisfies the constraint} \\ 0 \text{ otherwise.} \end{cases} \tag{7.62}$$

We wish to find a string z that approximately maximizes the objective function

$$C(z) = \sum_{\alpha=1}^{m} C_\alpha(z) \tag{7.63}$$

Quantum Analogue: For the quantum analogue of the previous problem we define a diagonal operator: H_C acting on the 2^n-dimensional Hilbert space where each bitstring z is a basis vector $|z\rangle$.

H_C acts on $|z\rangle$ as follows:

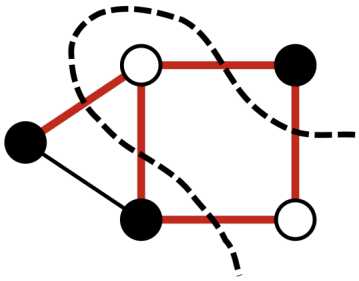
$$H_C|z\rangle = C(z)|z\rangle \tag{7.64}$$

and since $C(z)$ is scalar valued, we can see that each $|z\rangle$ is an eigenstate of H_C.

Let us view $\hat{C}$ as a Hamiltonian and the highest energy eigenstate $|z\rangle$ is the solution to the combinatorial optimization problem, as it gives the highest value of $C(z)$.

Max-Cut: In the case of Max-Cut we have:

$$C(z) = \frac{1}{2} \sum_{(i,j)\in E(G)} z_i z_j \tag{7.65}$$



QAOA at last: QAOA leverages approximate adiabatic quantum computation via Trotterization. We use two Hamiltonians: The first one is the **problem Hamiltonian** H_C which just by looking at Eq. (7.65) you should suspect its the Ising Hamiltonian.

The other one is called **mixer Hamiltonian** which is

$$H_B = \sum_{j=1}^{n} \sigma_j^x \tag{7.66}$$

The corresponding unitaries we need are:

$$U_C = e^{-i\gamma H_C} \tag{7.67}$$

$$U_B = e^{-i\beta H_B} \tag{7.68}$$

QAOA: Optimization: The goal is to maximize the expression

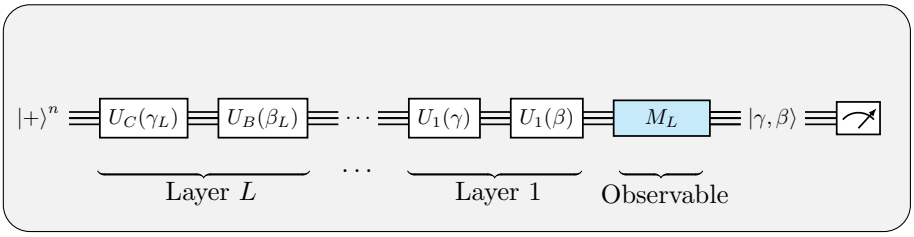
$$M_L(\gamma,\beta) \coloneqq \langle \gamma,\beta | M_L | \gamma,\beta \rangle \tag{7.69}$$

$\gamma \in [0,2\pi]^L$, $\beta \in [0,\pi]^L$.

and

$$|\gamma,\beta\rangle = U_C(\gamma_L)U_B(\beta_L)\ldots U_C(\gamma_1)U_B(\beta_1)|+\rangle^n. \tag{7.70}$$

Compare with Eq. (7.57). Its basically the same.



QAOA: Intuition: We begin in an eigenstate of H_B and then repeatedly let the system evolve under H_C and H_B, alternating between the two.

The approximation increase as $L \to \infty$.

We are trying to find

$$(\gamma^*,\beta^*) = \arg\max_{\gamma,\beta} \|\mathbb{E}[M_L] - \langle M_L \rangle\|_\ell^p \tag{7.71}$$

In the end we measure $|\gamma,\beta\rangle$ in the computational basis to get some bitstring z, and evaluate $C(z)$.

We repeat the above steps $\mathcal{O}(m \log m)$ (m number of constraints) such that we bound $C(z)$ with high probability.

Key result: QAOA with $L = 1$ achieves an approximation ratio of $r_c = C(z)/C_{\max} = 0.6924$ when performing Max-Cut on 3-regular graphs.

Part III
Applications

hyperref graphicx,epsfig hyperref algorithm algpseudocode hyperref mathrsfs multirow float color Theorem Definition Claim

Chapter 9

Applications in Financial Services

In this chapter, we want to make a few remarks on the practical aspects of use of what we have seen, esp. in Chapters 7 and 8, in the financial services industry. Therein, one needs to deal with many more vendors (i.e., salesmen) than universities (i.e., researchers).

9.1 Practical aspects of quantum annealers

Outside of many reputable vendors of gate-based quantum computers, there are also numerous vendors of so-called quantum annealers. While there are several quantum annealers across the world in academic environments, the most well-known vendor is D-Wave Systems. Other vendors that develop superconducting quantum annealers are Qilimanjaro and Avaqus. Recall, QA is a type of analog quantum computation based on the concept of adiabatic quantum computation (AQC). As such, it is possible to devise systems that perform AQC with stoquastic Hamiltonians but are not necessarily based on superconducting qubits. Such examples include Pasqal and QuEra that use arrays of Rydberg atoms which are highly excited atoms with a large distance between the electron and the nucleus. Finally, several companies manufacture specialized classical hardware (e.g., based on FPGAs) that simulates quantum annealing, for example Fujitsu.

What is common among the above is that the most obvious use cases and candidate problems to be attacked by these machines are hard optimization problems. Therefore, similar to companies offering classical solvers, such as Gurobi, understanding of optimization is crucial for a career in this area.

Note that the applications go beyond optimization, e.g. to machine learning, and we discuss below an example: QBoost.

9.1.1 Focus: D-Wave

D-Wave being the first company to file for a patent. D-Wave gained a lot of notice once multi-qubit quantum tunneling effects were observed experimental and showed the computational potential it may have.

Currently, the most advanced D-Wave machine is the 5,760-qubit Advantage machine with which the study [80] was performed.

How is performance measured? It is commong to use a metric known as Time-To-Solution (TTS) when performing benchmarking studies. There, data collected from multiple runs of the QA are used to compute the probability of finding a ground state solution for the given configuration of (adjustable) parameters. This probability is:

$$p_{\text{TTS}} \coloneqq \frac{\text{\# of ground state solutions}}{\text{\# of total QA runs}}. \tag{9.1}$$

The TTS proper is defined as the expected time to obtain the ground state solution at least once with success probability α and it is computed as:

$$\text{TTS} = t_{\text{run}} \frac{1 - \log \alpha}{1 - \log p_{\text{TTS}}}. \tag{9.2}$$

Here t_{run} is the annealing time for a single run of the QA and $\alpha = 0.99$ by default. Scheduling is a NP-Hard problem and you should expect that TTS scales exponentially with the size of the input N.

In the context of adiabatic quantum optimisation algorithms (Ising model mapping) as well as the quantum adiabatic theorem, for a problem of size N , quantum optimisers (are hoped to) solve NP-Hard combinatorial optimisation problems in time proportional to $\exp(\beta N \gamma)$ as $N \to \infty$, for positive coefficients β (scaling exponent) and γ.

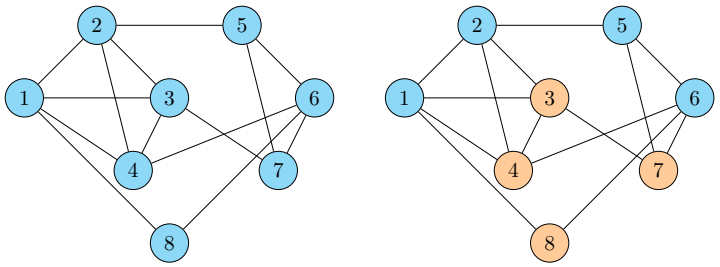
It should expected that, since scheduling problems are NP-Hard, TTS should scale exponentially with the problem size N in the asymptotic limit for $\gamma = 1$. The value of the β parameter that turns out to fit the experimental results $\text{TTS} = T_0 \exp \beta N$, for some constant $T_0 > 0$, ranges between 1.01 and 1.17 depending on the D-Wave machine.

9.2 More on QUBO

In this section we aim to discuss a few more QUBO formulations of interesting hard problems.

9.2.1 Graph Partitioning

Consider an undirected graph $G = (V, E)$. The task is to partition the set of vertices V into two subsets of equal size $N/2$, such that the number of edges connecting the two subsets is minimized.



We can directly assign spin variables represented by the graph vertices where $x = +1$ values mean the blue class and $x = -1$ values mean the orange class. The problem is solved by considering the following cost function:

$$L(x) = L_A(x) + L_B(x), \tag{9.3}$$

where

$$L_A(x) = \alpha \sum_{i=1}^{N} x_i, \tag{9.4}$$

a term that provides a penalty term if the number of elements in the blue set is not equal to the number of elements in the orange set, and

$$L_B(x) = \beta \sum_{(u,v)\in E(G)} \frac{1 - x_u x_v}{2}, \tag{9.5}$$

a term that provides a penalty each time an edge connects vertices from different subsets. If $\beta > 0$, then we wish to minimize the number of edges between the two subsets while if $\beta < 0$ we want to maximize this number. If $\beta < 0$ is chosen, then it must be small enough so that it is never favorable to violate the other constraint L_A.

9.2.2 Binary Integer Linear Programming

Consider the binary vector $x = (x_1 \ldots x_N) \in \{0,1\}^N$. Binary integer linear programming (BILP) amounts to the following problem:

$$\begin{aligned} \max_{x\in\{0,1\}^N} \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & A \in \mathbb{R}^{M\times N} \\ & b \in \mathbb{R}^M \end{aligned} \tag{9.6}$$

A variety of problems can be formed as BILPs (for example in the context of banking revenue maximization subject to regulating constraints). The cost function $L(x)$ corresponding to the QUBO formulation is

$$L(x) = L_A(x) + L_B(x), \tag{9.7}$$

where

$$L_A(x) = \alpha \sum_{j=1}^{m} \left(b_j - \sum_{i=1}^{N} A_{ij} x_i \right)^2, \tag{9.8}$$

where α is a constant. Note that $L_A(x) = 0$ enforces the constraint $Ax = b$. When this is not met, we get an overall penalty to the objective function. Furthermore,

$$L_B(x) = -\beta \sum_{i=1}^{N} c_i x_i, \tag{9.9}$$

for $\beta < \alpha$ another constant. Essentially, the constants α and β are tuning parameter that determines the relative importance of maximizing the objective function compared to satisfying the constraints. The condition $\beta < \alpha$ ensures that the constraints take precedence over the objective function, which is usually the case in constrained optimization problems. The reason for the minus sign is there since in QUBOs (naturally) we have a minimization problem.

9.2.3 Portfolio optimization

One of the fundamental problems in quantitative finance is portfolio optimization which is part of modern portfolio theory (MPT). A typical portfolio optimization is formulated as follows. Let N be the number of assets (things you can buy or sell in a market), μ_i the expected return of asset $i \in [N]$, σ_{ij} the covariance between the returns of asset i and asset j and R the target portfolio return. The decision variables are the weights $w_i \in \mathbb{R}$ associated to asset i for all $i \in [N]$ with.

The standard approach here is the *Markowitz mean-variance* approach. This amounts to the following quadratic program:

$$\begin{aligned} \min_{w\in\mathbb{R}^N} \quad & \sum_{i,j=1}^{N} w_i w_j \sigma_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i = 1, \\ & \sum_{i=1}^{N} w_i \mu_i = R. \end{aligned} \tag{9.10}$$

Intuition. Essentially, this problem amounts to the construction of an optimal portfolio from the set of all possible assets with known characteristics such as their returns, volatilities, and pairwise correlations. Realistically, we would expect to be able to select $M \leq N$ assets from the set of available N assets that should be the best possible choice according to the criteria set by the constraints.

Quadratic programs of this form are efficiently solvable using a number of quadratic programming solvers efficiently. However, consider the case where where weights w are discrete; this situation starts resembling like a NP-Complete problem.

In such a situation Prob. (9.10) can be mapped to a QUBO suitable for QA. This is done as follows. We define the QUBO objective as:

$$L(s) = \sum_{i=1}^{N} a_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} s_i s_j. \tag{9.11}$$

In this context

$$s_i = \begin{cases} 1 & \text{means asset } i \text{ is selected,} \\ 0 & \text{means asset } i \text{ is not selected.} \end{cases} \tag{9.12}$$

Then, given the N asset set $s = \{s_1, \ldots, s_N\}$ find the binary configuration that minimizes the $L(s)$ subject to the cardinality constraint that can be added via a penalty term $L_{\text{pen}}(s)$. Specifically, the requirement that $\sum_{i=1}^{N} s_i = M$ is encoded via:

$$L_{\text{pen}}(s) = P\left(M - \sum_{i=1}^{N} s_i\right)^2. \tag{9.13}$$

Furthermore, in Eq. (9.12), the coefficients a_i, reflect the asset attractiveness as a standalone (think user defined hyperparameter). Assets with large expected risk-adjusted returns are commonly rewarded with negative values for a_i while assets with small expected risk-adjusted returns should be penalised with positive values of a_i. Finally, the coefficients b_{ij} reflect the pairwise diversification penalties (if positive) and rewards (if negative) and are derived from the asset pairwise correlations. For all purposes of this course, assume a_i and b_{ij} as given.

The total QUBO to be solved is:

$$\min_{s\in\{0,1\}^N} \quad L_{\text{total}}(s) \coloneqq L(s) + L_{\text{pen}}(s). \tag{9.14}$$

The minimization of this QUBO optimizes for the risk-adjusted returns by using the so-called Sharpe ration which is computed as $(r - r_0)/\sigma$ where r is the expected annualised asset return, r_0 is the applicable risk-free interest rate and σ is the asset volatility.

The higher the Sharpe ratio the better returns relative to the risk taken. Volatility is usually estimated as the historical annualized standard deviation if the net asset value returns. Finally, expected returns can be either estimated as the historical returns or derived independently using e.g. Monte Carlo simulations.

9.3 Quantum Boost

Let us discuss how QBoost is used in the context of Machine Learning (ML). First, let us set up some notation:

Object	Definition
$x_t \in \mathbb{R}^N$	vector of N features
$y_t \in \{0,1\}$	binary classification label
$\{x_t, y_t\}_{t\in[M]}$	training set
$c_i(x_t) = \pm\frac{1}{N}$	value of weak classifier i on event t
$q \coloneqq (q_1, \ldots, q_N)$	vector of binary weights associated with each weak classifier

Table 9.1: Notation

The classification error for sample t is given by the square error

$$\left(\sum_{i=1}^{N} c_i(x_t) q_i - y_t\right)^2. \tag{9.15}$$

The total cost function to minimize is the sum of squared errors across the training data:

$$L(s) = \sum_{t=1}^{M} \left(\sum_{i=1}^{N} c_i(x_t) s_i - y_t\right)^2 \tag{9.16}$$

Expanding this out yields a term y_t^2 that does not depend on s and does not influence the minimization of L (can be absorbed as a constant energy shift). Overfitting can be done by adding a penalty $\lambda > 0$. The objective to minimize in the QBoost agorithm is:

$$\tilde{L}(s) = \sum_{t=1}^{M} \left(\sum_{i=1}^{N} c_i(x_t) q_i \sum_{j=1}^{N} c_j(x_t) s_j - 2y_t \sum_{i=1}^{N} c_i(x_t) s_i \right) + \lambda \sum_{i=1}^{N} s_i \tag{9.17}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} q_i q_j + \sum_{i=1}^{N} (\lambda - 2C_i) s_i, \tag{9.18}$$

where

$$C_{ij} \coloneqq \sum_{t=1}^{M} c_i(x_t) c_j(x_t), \qquad C_i \coloneqq \sum_{i=1}^{M} c_i(x_t) y_t.$$

Remark: the penalty term added here is analogous to LASSO regression method with L_1 penalty. This is sort of ubiquitous in the ML literature. Note that ridge regression with L_2 penalty could be chosen instead.

Next, we need to map the problem to an Ising model. To do so we consider σ to be spin variables by defining

$$\sigma = 2s - 1. \tag{9.19}$$

The Ising Hamiltonian is then written as:

$$H = \frac{1}{4} \sum_{i,j=1}^{N} C_{ij} \sigma_i \sigma_j + \frac{1}{2} \sum_{i,j=1}^{N} C_i \sigma_i + \sum_{i=1}^{N} (\lambda' - C_i) \sigma_i, \tag{9.20}$$

where $\lambda' \coloneqq \frac{1}{2}\lambda$ is a rescaled penalty coefficient. QA aims to solve the problem to minimize H and compute the ground state spin configuration bit-string $|s\rangle$, with $s \in \{-1, 1\}^N$. Then, for each new sample x, the classifier is given as

$$R(x) = \sum_{i=1}^{N} s_i c_i(x) \quad \in [-1, 1]. \tag{9.21}$$

Application: QBoost

QA for ML applications has been gaining a lot of popularity (and serves as a business model for a number of quantum computing startups). It is claimed to have demonstrated performance advantage in compaerison with algorithms such as binary decision tree-based Extreme Gradient Boosting (XGBoost) and DNN classifiers on small datasets.

A very interesting application is that of forecasting credit card client defaults. For that one can utilize a publicly available dataset available from the UCI Machine Learning Repository [307,308]. This dataset consists of 30,000 samples with binary classifications:

- a client does not default - class 0
- a client does default - class 1

There are $N = 23$ features $(F_1, \ldots, F_{23})$ that are available to extract predictive power from:

- F_1: amount of given credit (continuous)
- F_2: gender (binary)
- F_3: education (discrete)
- F_4: marital status (discrete)
- F_5: age (discrete)
- F_6: repayment status of previous month (discrete)
- F_7: repayment status of two months ago (discrete)
- F_8-F_{11}: similar (discrete)
- F_{12}: bill amount past month (continuous)
- F_{13}: bill amount two months ago (continuous)
- F_{14}-F_{17}: similar (continuous)
- F_{18}: amount of previous month payment (continuous)
- F_{19}: amount of payment two months ago (continuous)
- F_{20}-F_{23}: similar (continuous)

Having these features, the next step is to construct the weak classifiers. For that each feature can be used separately as an input into a logistic regression classifier with the goal to make a binary prediction: $-1/N$ for class 0 and $+1/N$ for class 1. That, of course, would require splitting the data to a training and validation test (e.g., 0.7 training and 0.3 validation). A rule for determining the output can be set to be a majority vote: the prediction of the (strong) classifier is given by the sum of the of the weak classifiers with values in $\{-1, +1\}$ (as required by QBoost).

It can be argued that QBoost provides an improvement on this approach by finding an optimal configuration of the weak classifiers.

Such a simple implementation can lead to results such as the ones in Fig. 9.2 (you are more than welcome to try this on your own) where QBoost shows what some people call "performance" advantage. While several, much more sophisticated classical classifiers exist, there is no fundamental reason that the same argument cannot be applied for quantum classifiers as well.

	Accuracy	Precision	Recall
GradBoost	0.83	0.69	0.35
MLP	0.83	0.69	0.35
QBoost	0.83	0.71	0.33

Table 9.2: Caption

9.4 Warm-starting QAOA

Recall, that we have discussed the Quantum Approximate Optimization Algorithm (QAOA) [77] last time. We discussed that QAOA This algorithm encodes a combinatorial optimization problem in a Hamiltonian H_C whose ground state is the optimum solution. The QAOA first creates an initial state which is the ground state of a mixer Hamiltonian H_M where a common choice is

$$H_M = -\sum_{i=1}^{N} \sigma_i^x, \tag{9.22}$$

with ground state being $|+\rangle^{\otimes n}$. Then, recall, for depth-L QAOA, we apply L times the unitary $U_{\text{QAOA}} = U_C(\gamma)U_B(\beta)$ defined as:

$$U_C(\gamma) \coloneqq e^{-\mathrm{i}\gamma H_C} \tag{9.23}$$

$$U_B(\beta) \coloneqq e^{-\mathrm{i}\beta H_M}. \tag{9.24}$$

The result is:

$$U_{\text{QAOA}}|+\rangle^{\otimes n} = |\gamma,\beta\rangle. \tag{9.25}$$

A classical optimizer (e.g. SPSA) then seeks the optimal values of β and γ to create a trial state which minimizes the energy of the problem Hamiltonian H_C.

While a very promising algorithm, initially it lacked any theoretical guarantees on its performance ratio and for certain problem instances of interest (e.g. Max-Cut) it cannot, for constant L, outperform the classical Goemans-Williamson randomized rounding approximation (which for MAXCUT finds cuts whose expected value is an α fraction of the global optimum, for $0.87856 < \alpha < 0.87857$, with the expectation over the randomization in the rounding procedure).

While several improvements of the QAOA have been developed in the literature, we will focus here on warm-starting QAOA [81].

Relaxations. QUBOs have already been discussed a lot. A common formulation is:

$$\min_{x\in\{0,1\}^n} x^T Q x + \mu^T x. \tag{9.26}$$

where x is a vector of n binary decision variables, $Q \in \mathbb{R}^{n\times n}$ a symmetric matrix, and $\mu \in \mathbb{R}^n$ a vector. Since for binary variables $x_i^2 = x_i, \mu$ can be added to the diagonal of Σ, and in the following, we only add μ when it simplifies the notation in the given context. Note that, as discussed, practically any mixed-integer linear program can be encoded in a QUBO it is automatically NP-Hard.

If Q is positive semidefinite, there exists a trivial continuous relaxation of the QUBO above:

$$\min_{x\in[0,1]^n} x^T Q x \tag{9.27}$$

is a convex quadratic program and the optimal solution c^* of the continuous relaxation is easily obtainable with classical optimizers. However, if Q is not positive semidefinite (and in many applications of interest there is no reason to be) one can obtain another continuous relaxation, the semidefinite program (SDP):

$$\begin{aligned} \max_{Y\in\mathbb{S}^n} & \operatorname{tr}(QY) \\ \operatorname{diag}(Y) &= \mathbf{1} \\ Y &\succeq 0, \end{aligned} \tag{9.28}$$

where $\mathbb{S}^{n\times n}$ denotes the set of $n \times n$ symmetric matrices, $\mathbf{1}$ is an n-vector of ones, and $Y \succeq 0$ denotes that Y must be positive semidefinite. Given the optimal solution Y^* to the SDP above, there exist several approaches to generating solutions[1] of the corresponding (QUBO), often with approximation guarantees. Furthermore, quantum computers offer the prospect of some speed-ups in solving SDPs (not discussed in these lectures).

Warm-starting QAOA. The solutions of either continuous-valued relaxation discussed above can be used to initialize VQAs: this is known as warmstarting them [82]. Let us focus on how to warm-start the QAOA [77].

In QAOA, each decision variable x_i of the discrete optimization problem corresponds to a qubit by the substitution $x_i = (1 - s_i)/2$. Each s_i is replaced by a spin operator σ_i to transform the cost function to a cost Hamiltonian H_C. Then, after the procedure described initially, that is utilizing the unitary U_{QAOA}, one performs the final measurement which can be considered as a randomized rounding. Warm-starting amounts to replacing the initial equal superposition state $|+\rangle^{\otimes n}$ with a state

$$|\phi^*\rangle = \bigotimes_{i=0}^{n-1} \hat{R}_y(\theta_i)\,|0\rangle_n \tag{9.29}$$

[1] As a matter of facr, a classical laptop can solve instances of (SDP) relaxations of QUBO, where Q has 10^{13}.

which corresponds to the solution c^* of the relaxed Problem (9.27). Here, $\hat{R}_Y(\theta_i)$ is a θ_i-parametrized rotation around the y-axis of the qubit (see Fig. 9.1) and $\theta_i := 2\arcsin(\sqrt{c_i^*})$ for c_i^* given as the solution of QUBO (9.27).

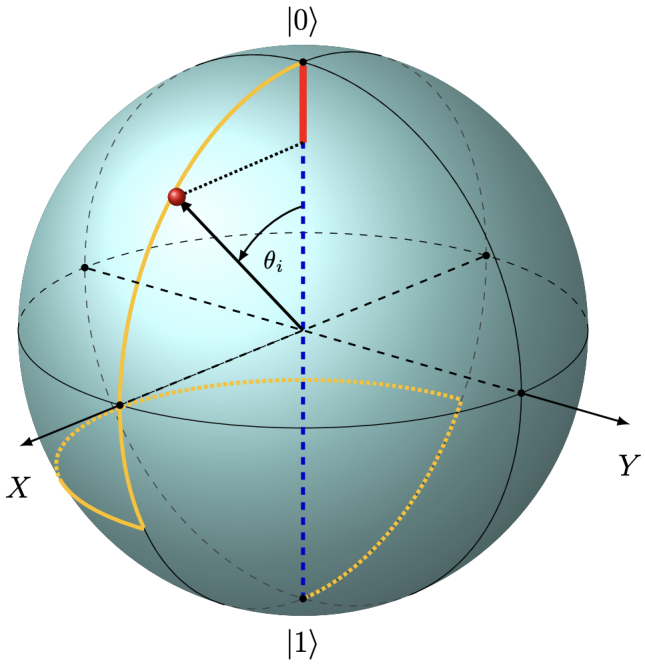


Fig. 9.1: The initial state is given by the red segment. The yellow path shows the evolution of the quantum state starting at $|0\rangle$ to $R_y(-\theta_i)R_z(-\beta)R_y(2\theta_i)$ for $\beta = \pi/2$.

Additionally, the mixer Hamiltonian also is replaced. A choice for the warm-starting mixer Hamiltonian is

$$H_M^{\mathrm{WS}} = \sum_{i=1}^{n} H_{M,i}^{\mathrm{WS}} \tag{9.30}$$

where

$$H_{M,i}^{\mathrm{WS}} = \begin{pmatrix} 2c_i^* - 1 & -2\sqrt{c_i^*(1-c_i^*)} \\ -2\sqrt{c_i^*(1-c_i^*)} & 1 - 2c_i^* \end{pmatrix} \tag{9.31}$$

which has $R_y(\theta_i)|0\rangle$ as ground state. One can show that the ground state of H_M^{WS} is $|\phi^*\rangle$ with energy $-n$. Therefore, WS-QAOA applies at layer k a mixing gate which is given by the time-evolved mixing Hamiltonian $U_M(\beta) = e^{-\mathrm{i}\beta H_M^{\mathrm{WS}}}$.

For technical reasons [81] one has to actually modify the definition of θ_i as

$$\begin{aligned}\theta_i &= 2\arcsin\left(\sqrt{c_i^*}\right) && \text{if} \quad c_i^* \in [\varepsilon, 1-\varepsilon] \\ \theta_i &= 2\arcsin(\sqrt{\varepsilon}) && \text{if} \quad c_i^* \leq \varepsilon \\ \theta_i &= 2\arcsin(\sqrt{1-\varepsilon}) && \text{if} \quad c_i^* \geq 1-\varepsilon.\end{aligned}$$

where $\varepsilon \in [0, 0.5]$ and the mixer Hamiltonian H_M is adjusted accordingly. The parameter ε provides a continuous mapping between WS-QAOA and standard QAOA since at $\varepsilon = 0.5$ the initial state is the equal superposition state and the mixer Hamiltonian is the X operator. If all $c_i^* \in (0,1)$ or $\varepsilon > 0$, WS-QAOA converges to the optimal solution of (QUBO) as the depth L approaches infinity as does standard QAOA.

This directly follows from (surprise) the adiabatic theorem and the fact that we start in the ground state of the mixer which overlaps with all computational basis states including the optimal solution.

For large enough L, WS-QAOA therefore the adiabatic evolution transforming the ground state of the mixer into the ground state of H_C as expected. The speed of the adiabatic evolution is limited by the spectral gap of the intermediate Hamiltonians as we discussed in the previous lecture.

The speed of the evolution can be related to the depth L, where a slow evolution (larger terminal time T) implies a larger L. The idea of WS-QAOA is to speed-up this evolution by optimizing the parameters instead of following a fixed annealing schedule.

Several other variations of WS-QAOA have been studied, for example "rounded warm-started" QAOA. Such a technique is very appealing for application on NISQ devices for combinatorial optimization problems.

Below we quote a nice experimental demonstration from [81].

There, the authors investigate the role of the warm-start mixer operator $\hat{H}_M^{(ws)}$ by replacing it with the standard mixer $-\sum_{i=0}^{n-1} \hat{X}_i$ while using the initial state given by the continuous solution c^*. Under these conditions the energy of the optimized state does not converge to the minimum energy, see blue triangles in Fig. 9.2(b). The probability of sampling the optimal discrete solution is between warm-start and standard QAOA but depends heavily on the initial point given to COBYLA, see Fig. 9.2(a). These results further justify the use of the modified mixer in WS-QAOA.

They even proceed to further illustrate the advantage of a warm-starting QAOA at low depth by solving 250 random portfolio instances with warm-started and standard QAOA, both at depth $L = 1$. There the standard QAOA produces variational states that poorly approximate the ground state, see the histogram of E^*_{cold} in Fig. 9.3(a). However, WS-QAOA produces optimized variational states that are much closer to the minimum energy of each problem Hamiltonian. Furthermore, we find

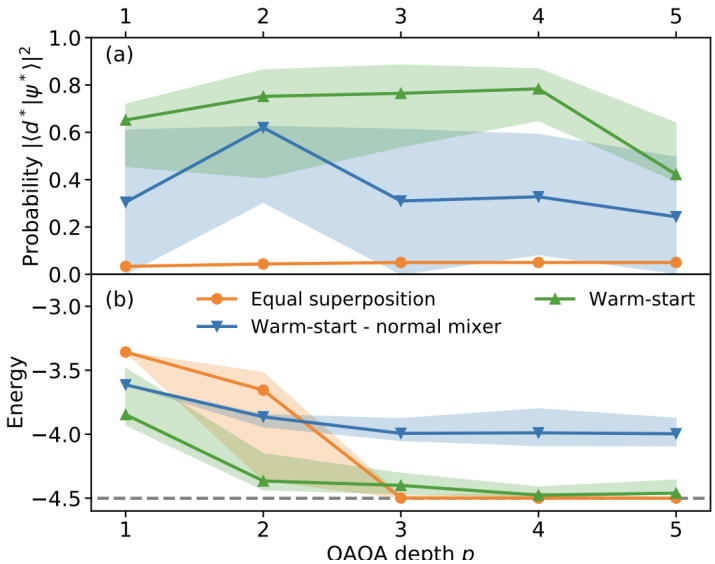


Fig. 9.2: (a) Probability to sample the optimal state $|d^*\rangle$ from the optimized trial state $|\psi^*\rangle$ and (b) energy of $|\psi^*\rangle$ for warm-start and standard QAOA at different depths for $n = 6$ assets and $q = 2$. The optimal discrete and continuous solutions are $d^* = (0,0,1,1,1,0)$ and $c^* \simeq (0.17,0,0.97,0.73,1.0,0.14)$, respectively. QAOA is run ten times with different initial random guesses for (β,γ) chosen uniformly from $\pm 2\pi$. The thick lines show the median of the ten runs while the shaded areas indicate the 25% and 75% quantiles. The gray dashed line shows E_0.

that WS-QAOA tends to produce better solutions when the overlap $d^{*T}c^*/B$ between the optimal solutions to the discrete and relaxed problems is closer to 1.

9.5 Asset Management and Monte Carlo Simulations

Derivative pricing using a quantum computer

What are derivatives and why one would be interested in using a quantum computer?

Short answer: derivatives are financial instruments that make some people rich and quantum computers can do things (in principle) quadratically faster. Use this notebook.

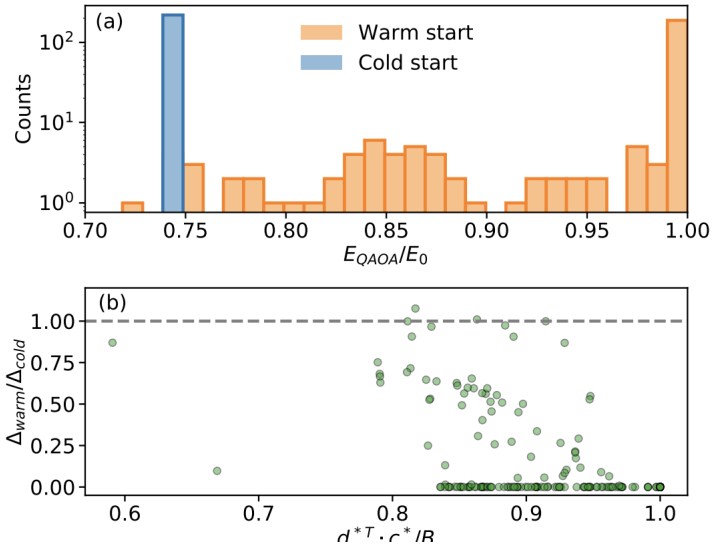


Fig. 9.3: Improvement of depth-one WS-QAOA over standard QAOA for 250 random portfolio instances with $q = 2$ (q controls the risk-return trade-off).

Chapter 10

Applications in Security

The question of many people's minds is whether and when "quantum computers would kill RSA"? We will review some recent work in security applications:

- generating random strings
- quantum key distribution
- Shor factoring
- Grover-based factoring
- variational factoring.

While the first two happily live in "vendor-land", the latter three are more involved.

10.1 Generating random strings

US authorities now recommend using random strings only from quantum effects, rather than pseudorandom generators. In some cases [83, 84], quantum random number generators (RNG) come with strong guarantees, but often, it seems an overkill to utilize a quantum computer to generate random numbers. There are now purpose-built devices [85] that can generate random strings at 17 Gbps, exceeding what can be done with near-term quantum computers. The purpose-built devices can be bought, e.g., from ID Quantique. This is hence one example of quantum technologies being essential to security.

10.2 Quantum key distribution

An important quantum technology in security is quantum key distribution, which makes it possible to certify that the communication has not been intercepted. There are two approaches:

- *Prepare-and-measure*: measuring an unknown quantum state changes it.
- *Entanglement-based*: measuring one of two entangled quantum systems affects the other.

Either way, one can calculate the amount of information that has been intercepted by measurement. ID Quantiq showcased quantum key distribution at 307 km, and sells related devices. Toshiba demonstrated QKD at 100 km of fiber in 2004 and the first with a continuous key rate exceeding 10 Mbit/second in 2017. (CTU has purchased such devices from both ID Quantiq and Toshiba.) This is hence an example of quantum technologies being readily available to improve security.

10.3 Factoring integers

Much of modern cryptoprimitives are built on factoring of large integers. A textbook version of public-key cryptography, here cited from [86] in verbatim, is as follows:

1. Select two large prime numbers, p and q.
2. Compute the product $n = pq$.
3. Select at random a small odd integer, e, that is relatively prime to $\phi(n) = (p-1)(q-1)$.
4. Compute d, the multiplicative inverse of e, modulo $\phi(n)$.
5. The RSA public key is the pair $P = (e,n)$. The RSA secret key is the pair $S = (d,n)$.

The encryption of message M on $\log n$ bits involves $M^e \mod n$ to obtain $E(M)$, while decryption requires $E(M)^d \mod n$.

What is the complexity of factoring n to p and q?

- poly(log(n)) is the runtime of factoring algorithms on a BSS machine. Testing whether an integer is a prime is in P, but does not provide the factors, when the number is not prime.

- $O(n^{1/4})$ is the runtime of the best deterministic factoring algorithms for factoring an integer n with $\log n$ bits in length.
- $O(\exp(c(\log n)^{1/3}(\log\log n)^{2/3}))$ is the runtime of the best randomized algorithms, for some constant c and integer n. The runtime is thus subexponential, but not polynomial time: $O(\exp(\sqrt{(\log n)(\log n)}) = O(n)$. It is thus unlikely that factoring is NP-Complete. The elliptic curve method (ECM, [87]) is the fastest known algorithm for small numbers, e.g. within 100 digits. The the number field sieve (NFS, [88]) is the best classical algorithm for large numbers, and has been used to factor a 240-digit (795-bit) number in 900 core-years.
- $O((\log n)^2(\log\log n)(\log\log\log n))$ is the runtime of a quantum algorithm introduced by Peter Shor [35], along with a polynomial-time (in $\log n$) classical post-processing algorithm.

10.3.1 Shor factoring

Peter Shor introduced an algorithm for factoring integers [35], which based on two facts of number theory, makes it possible to reduce factoring to order finding, i.e., determining r in $f(x+r) = f(x)$ for $f(x) = a^x$. When one receives a composite number n, it uses $O(\log^3 n)$ order-finding operations to produce a non-trivial factor of n with a constant probability.

This is based on the following facts:

Theorem 5.2 in [86]: Suppose that n is an L-bit composite number, and x is a non-trivial solution to the equation $x^2 = 1 \mod n$ in the range $1 \leq x \leq n$, i.e., neither $x = 1 \mod N$ nor $x = n.-1 = -1 \mod n$. Then at least $\gcd(x-1,n)$ and $\gcd(x+1,n)$ is a non-trivial factor of n can be computed using $O(L^3)$ operations.

Theorem 5.3 in [86]: Suppose that $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_m^{\alpha_m}$ is the prime factorization of an odd composite positive integer. Let x be an integer chosen uniformly at random, subject to the requirements that $1 \leq x \leq n-1$ and x is co-prime to n. Let r be the order of $x \mod n$. Then the probability r is even and $x^{r/2} \neq -1 \mod n$ is greater or equal to $1 - \frac{1}{2^m}$.

With these facts, we can formulate the Shor factoring algorithm:

1. If n is even, return 2.
2. If $n = a^b$ for $a \geq 1$ and $b \geq 2$, return a.
3. Choose x in [1, n-1]. If $\gcd(x,n) > 1$, return $\gcd(x,n)$.

4. Use order-finding to find the order r of x modulo n. If r is even and $x^{r/2} \neq -1 \mod n$ and either of $\gcd(x^{r/2}-1,n)$ and $\gcd(x^{r/2}+1,n)$ is non-trivial, return the non-trivial factor.
5. Repeat from 3 otherwise.

Shor's order-finding works as follows:

1. creates an initial, Q-qubit state $|0\rangle^{\otimes Q}$
2. apply Hadamard transform on it: $\frac{1}{\sqrt{Q}}\sum_{k=0}^{Q-1}|k\rangle$
3. apply the function $f(x) = a^x \bmod N$ using $U_f|x,0^n\rangle = |x,f(x)\rangle$ to obtain

$$U_f \frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,0^n\rangle = \frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,f(x)\rangle$$

   such that the value we are looking for is in the phase
4. apply the quantum Fourier transform: $\frac{1}{Q}\sum_{x=0}^{Q-1}\sum_{y=0}^{Q-1}\omega^{xy}|y,f(x)\rangle$
5. obtain y by measuring the first register. The probability of measuring $|y,z\rangle$ is

$$\frac{1}{Q^2}\frac{\sin^2(\frac{\pi mry}{Q})}{\sin^2(\frac{\pi ry}{Q})}$$

   .

Let us consider the example of $n = 15$

1. Let us consider $n = 15$ and a random number x coprime (having no non-trivial common factors) with n, e.g., $x = 7$.
2. Compute the order r of x modulo n, as follows: apply Hadamard transform to the first register of $|0\rangle|0\rangle$. Compute $f(k) = x^k \mod n$ in the second register

$$\frac{1}{\sqrt{2^t}}[|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + |6\rangle|4\rangle + \cdot].$$

   When inverse Fourier transform is applied to the first register (seen as $2^t = 2048$ frequencies) and the second register is measured, one obtains one of 1, 7, 4, or 13. Eventually, we obtain $r = 4$ as the order of $x = 7$.
3. Classically, we see r is even, and $x^{r/2} \mod n = 7^2 \mod 15 = 4 \neq -1 \mod 15$. Again classically, we run $\gcd(x^2-1,15) = 3$ and $\gcd(x^2+1,15) = 5$ to obtain two factors.

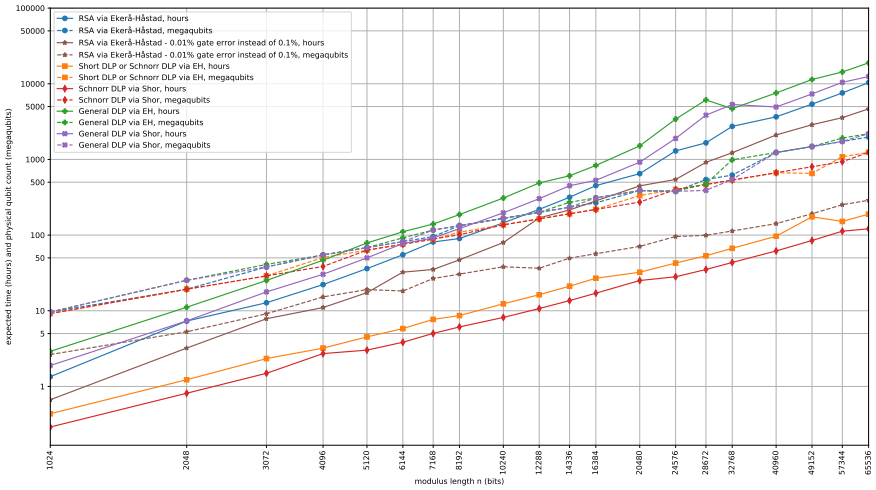


Fig. 10.1: Scaling of Shor's Factoring: Log-log plot of estimated space (in the millions of qubits) and expected-time costs (days required) with the number of bits of RSA keys, cited from [89] in verbatim.

Shor's factoring has been demonstrated for the number of 15 more than two decades ago, and the scalability beyond is still very much a subject of lively discussion. A Google team [89] estimates that one could perform factoring of 2048-bit RSA integers in 8 hours using 20 million noisy qubits. (See Figure 10.1.) The assumptions of a planar grid of qubits with nearest-neighbor connectivity, physical gate error rate of 10^{-3}, a surface code cycle time of 1 microsecond, and the use of surface codes are all quite realistic. Surface codes are textbook material [86, Chapter 10], although not covered by this course. A French team [90] suggested that one could perform factoring of 2048-bit RSA integers in 177 days with 13436 qubits, without being very explicit about the requirement of 430 million memory qubits. Likewise, the use of 3D gauge color codes is out of reach in current qubit technologies. Otherwise, the assumptions of physical gate error rate of 10^{-3}, a processor cycle time of 1 microsecond are quite realistic.

It is very important to stress that these estimates rely crucially on the assumptions on the overhead of commonly used quantum error correcting codes. For (perhaps difficult to decode, but otherwise viable) codes with lower overhead, these estimates of the numbers of qubits required would be radically lower. The best lower bounds for the space overhead [91] of 2D codes are of the order of $\Omega(\sqrt{\log(1/\delta)})$ for δ error rate, and the bounds can be even lower for 3D codes.

10.3.2 Grover-based factoring

[92] introduced another quantum algorithm for factoring, which they call GEECM (Grover plus Edwards Elliptic Curve Method). To gain some intuition, consider the trial division, where we would generate a small primes and perform Grover search for those that divide the n. It reduces the number of operations of Edwards Elliptic Curve Method from $L^{\sqrt{2}+o(1)}$ to $L^{1+o(1)}$ for $L = \exp(\sqrt{\log\sqrt{n}\log\log\sqrt{n}})$.

10.3.3 Variational factoring

In principle, you can use drastically fewer qubits in some cases, but with lesser hopes of speed-up. Notably, the explicit, "schoolbook" binary multiplication of p and q yields equations that have to be satisfied by bits p_i and q_i and carry bits $z_{i,j}$. One can formulate a "least-squares version" of the problem, which would minimise the sum of residuals squared, across the equations (bits). Clearly, this would be a QUBO, as in the previous lecture, and approached with, e.g., QAOA without any guarantees of finding the solution. On the flipside, one can get lucky. For instance, [93] report factoring 1099551473989, 3127, and 6557 with 3, 4, and 5 qubits, respectively, using a QAOA.

In a very similar spirit, a Chinese team [94] got to the frontpages of many newspapers announcing that 2048-bit semi-prime number can be factored on a NISQ level computer with 372 physical qubits and a gate depth in the thousands. The same paper has shown that a 48-bit number can be factored using the Schnorr factoring and QAOA. Unfortunately, they did not analyze how many runs of the circuit this would require in general. Our analyses [79] show would scale much worse than the runtime of the Shor factoring.

10.3.4 A Rejoinder

In the US, Congress passed Quantum Computing Cybersecurity Preparedness Act[1] in December 2022, which bars federal authorities from using cryptoprimitives based on factoring. It is unlikely that this is based on the discovery of a new factoring algorithm, but rather based on the risk of there being one. In many information security standards, you need to be sure that if you encrypt today, no one will be able to decrypt without knowing the key for the next 20+ years. In "Store Now,

[1] https://www.congress.gov/bill/117th-congress/house-bill/7535/text

Decrypt Later" attacks, nation states already gain access to large troves of encrypted information, in the hope that they would be able to decrypt it in the near future. Notice that for digital signatures (e.g., certificates on the web), the risk is much less: you can wait until a new factoring algorithm appears.

References

[1] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47:777–780, May 1935.

[2] John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt. Proposed experiment to test local hidden-variable theories. *Phys. Rev. Lett.*, 23:880–884, Oct 1969.

[3] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. Going beyond bell's theorem. *Bell's theorem, quantum theory and conceptions of the universe*, pages 69–72, 1989.

[4] C. Bennett and J. Gill. Relative to a random oracle a, $\mathbf{p}^a \neq \mathbf{np}^a \neq \text{co-}\mathbf{np}^a$ with probability 1. *SIAM J. Comput.*, 10(1):96–113, 1981.

[5] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189 – 201, 1979.

[6] A Yu Kitaev, AH Shen, and MN Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, Providence, RI, 2002. Graduate Studies in Mathematics vol. 47).

[7] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[8] Lance Fortnow. One complexity theorist's view of quantum computing. *Theoretical Computer Science*, 292(3):597–610, 2003. Algorithms in Quantum Information Prcoessing.

[9] Leonard M Adleman, Jonathan Demarrais, and Ming-Deh A Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997.

[10] David Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[11] A Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361. IEEE, 1993.

[12] Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 461(2063):3473–3482, 2005.

[13] David Elieser Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 1989.

[14] Jun Zhang, Jiri Vala, Shankar Sastry, and K Birgitta Whaley. Geometric theory of nonlocal two-qubit operations. *Physical Review A*, 67(4):042313, 2003.

[15] Dorit Aharonov. A simple proof that toffoli and hadamard are quantum universal. *arXiv preprint quant-ph/0301040*, 2003.

[16] Maarten Van Den Nes. Classical simulation of quantum computation, the gottesman-knill theorem, and slightly beyond. *Quantum Info. Comput.*, 10(3):258–271, mar 2010.

[17] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical review A*, 52(5):3457, 1995.

[18] Yaoyun Shi. Both toffoli and controlled-not need little help to do universal quantum computing. *Quantum Information & Computation*, 3(1):84–92, 2003.

[19] Richard Jozsa and Noah Linden. On the role of entanglement in quantum-computational speed-up. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 459(2036):2011–2032, 2003.

[20] Dorit Aharonov and Michael Ben-Or. Polynomial simulations of decohered quantum computers. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 46–55. IEEE, 1996.

[21] Daniel Gottesman. *Stabilizer codes and quantum error correction*. PhD thesis, California Institute of Technology, 1997.

[22] Emanuel Knill. Quantum computing with realistically noisy devices. *Nature*, 434(7029):39–44, 2005.

[23] Scott Aaronson, Harry Buhrman, and William Kretschmer. A qubit, a coin, and an advice string walk into a relational problem, 2023.

[24] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*, pages 41–69. Springer, 2011.

[25] 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.

[26] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 69–74. IEEE, 2022.

[27] Daniel S Abrams and Seth Lloyd. Nonlinear quantum mechanics implies polynomial-time solution for np-complete and #p problems. *Physical Review Letters*, 81(18):3992, 1998.

[28] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[29] J. Abhijith, Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Alexander Malyzhenkov, David

Mascarenas, Susan Mniszewski, Balu Nadiga, Daniel O'malley, Diane Oyen, Scott Pakin, Lakshman Prasad, Randy Roberts, Phillip Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter J. Swart, James G. Wendelberger, Boram Yoon, Richard Zamora, Wei Zhu, Stephan Eidenbenz, Andreas Bärtschi, Patrick J. Coles, Marc Vuffray, and Andrey Y. Lokhov. Quantum algorithm implementations for beginners. *ACM Transactions on Quantum Computing*, 3(4), jul 2022.

[30] David Harvey and Joris Van Der Hoeven. Integer multiplication in time $o(n \log n)$. *Annals of Mathematics*, 193(2):563–617, 2021.

[31] David Harvey and Joris Van Der Hoeven. Polynomial multiplication over finite fields in time $o(n \log n)$. *Journal of the ACM (JACM)*, 69(2):1–40, 2022.

[32] Gabriel Peyré. The discrete algebra of the Fourier transform. 2020. A draft textbook at https://mathematical-tours.github.io/daft-sources/DAFT-EN.pdf.

[33] Pierre Duhamel and Martin Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Signal processing*, 19(4):259–299, 1990.

[34] Daan Camps, Roel Van Beeumen, and Chao Yang. Quantum fourier transform revisited. *Numerical Linear Algebra with Applications*, 28(1):e2331, 2021.

[35] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[36] A. Yu. Kitaev. Quantum measurements and the abelian stabilizer problem, 1995.

[37] R. Cleve and J. Watrous. Fast parallel circuits for the quantum Fourier transform. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 526–536, 2000.

[38] Lisa Hales and Sean Hallgren. An improved quantum Fourier transform algorithm and applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 515–525. IEEE, 2000.

[39] Neil Shenvi, Kenneth R Brown, and K Birgitta Whaley. Effects of a random noisy oracle on search algorithm complexity. *Physical Review A*, 68(5):052313, 2003.

[40] Daniel Shapira, Shay Mozes, and Ofer Biham. Effect of unitary noise on grover's quantum search algorithm. *Physical Review A*, 67(4):042301, 2003.

[41] EM Stoudenmire and Xavier Waintal. Grover's algorithm offers no quantum advantage. *arXiv preprint arXiv:2303.11317*, 2023.

[42] Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1783–1793. SIAM, 2019.

[43] Lov K Grover. Quantum search on structured problems. In *Quantum Computing and Quantum Communications: First NASA International Conference,*

QCQC'98 Palm Springs, California, USA February 17–20, 1998 Selected Papers, pages 126–139. Springer, 1999.

[44] Christof Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60(4):2746, 1999.

[45] Lov K Grover and Anirvan M Sengupta. Classical analog of quantum search. *Physical Review A*, 65(3):032319, 2002.

[46] Jozef Gruska. *Quantum computing*, volume 2005. McGraw-Hill, London, 1999.

[47] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[48] David Sutter, Giacomo Nannicini, Tobias Sutter, and Stefan Woerner. Quantum speedups for convex dynamic programming. *arXiv preprint arXiv:2011.11654*, 2020.

[49] Andrew M. Childs and Jeffrey Goldstone. Spatial search by quantum walk. *Phys. Rev. A*, 70:022314, Aug 2004.

[50] Andrew M Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 59–68, 2003.

[51] Andrew M Childs. Universal computation by quantum walk. *Physical review letters*, 102(18):180501, 2009.

[52] Andrew M Childs, David Gosset, and Zak Webb. Universal computation by multiparticle quantum walk. *Science*, 339(6121):791–794, 2013.

[53] Julia Kempe. Quantum random walks: an introductory overview. *Contemporary Physics*, 44(4):307–327, 2003.

[54] Renato Portugal. *Quantum walks and search algorithms*, volume 19. Springer, 2013.

[55] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[56] Y. Aharonov, L. Davidovich, and N. Zagury. Quantum random walks. *Phys. Rev. A*, 48:1687–1690, Aug 1993.

[57] Andrew M Childs, Edward Farhi, and Sam Gutmann. An example of the difference between quantum and classical random walks. *Quantum Information Processing*, 1:35–43, 2002.

[58] Andrew M Childs, Leonard J Schulman, and Umesh V Vazirani. Quantum algorithms for hidden nonlinear structures. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 395–404. IEEE, 2007.

[59] Andrew M Childs and Wim Van Dam. Quantum algorithms for algebraic problems. *Reviews of Modern Physics*, 82(1):1, 2010.

[60] Ashley Montanaro. Quantum speedup of monte carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301, 2015.

[61] Dorit Aharonov, Wim Van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review*, 50(4):755–787, 2008.

[62] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[63] W. van Dam, M. Mosca, and U. Vazirani. How powerful is adiabatic quantum computation? In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001.

[64] Wim Van Dam, Michele Mosca, and Umesh Vazirani. How powerful is adiabatic quantum computation? In *Proceedings 42nd IEEE symposium on foundations of computer science*, pages 279–287. IEEE, 2001.

[65] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001.

[66] Jérémie Roland and Nicolas J Cerf. Quantum search by local adiabatic evolution. *Physical Review A*, 65(4):042308, 2002.

[67] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. Quantum adiabatic evolution algorithms versus simulated annealing. *arXiv preprint quant-ph/0201031*, 2002.

[68] Tosio Kato. On the adiabatic theorem of quantum mechanics. *Journal of the Physical Society of Japan*, 5(6):435–439, November 1950.

[69] A. Kitaev, A. Shen, and M. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, May 2002.

[70] Tameem Albash and Daniel A Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002, 2018.

[71] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

[72] Giuseppe E Santoro, Roman Martonák, Erio Tosatti, and Roberto Car. Theory of quantum annealing of an ising spin glass. *Science*, 295(5564):2427–2430, 2002.

[73] Arnab Das and Bikas K. Chakrabarti. Colloquium: Quantum annealing and analog quantum computation. *Rev. Mod. Phys.*, 80:1061–1081, Sep 2008.

[74] Edward Farhi and Aram W Harrow. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674*, 2016.

[75] Rolando D. Somma, Daniel Nagaj, and Mária Kieferová. Quantum speedup by quantum annealing. *Phys. Rev. Lett.*, 109:050501, Jul 2012.

[76] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):4213, 2014.

[77] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[78] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[79] Vyacheslav Kungurtsev, Georgios Korpas, Jakub Marecek, and Elton Yechao Zhu. Iteration complexity of variational quantum algorithms. *arXiv preprint arXiv:2209.10615*, 2022.

[80] Andrew D King, Jack Raymond, Trevor Lanting, Richard Harris, Alex Zucca, Fabio Altomare, Andrew J Berkley, Kelly Boothby, Sara Ejtemaee, Colin Enderud, et al. Quantum critical dynamics in a 5,000-qubit programmable spin glass. *Nature*, pages 1–6, 2023.

[81] Daniel J Egger, Jakub Mareček, and Stefan Woerner. Warm-starting quantum optimization. *Quantum*, 5:479, 2021.

[82] Jacek Gondzio. Warm start of the primal-dual method applied in the cutting-plane scheme. *Mathematical Programming*, 83(1-3):125–143, January 1998.

[83] Peter Bierhorst, Emanuel Knill, Scott Glancy, Yanbao Zhang, Alan Mink, Stephen Jordan, Andrea Rommal, Yi-Kai Liu, Bradley Christensen, Sae Woo Nam, et al. Experimentally generated randomness certified by the impossibility of superluminal signals. *Nature*, 556(7700):223–226, 2018.

[84] Yanbao Zhang, Hsin-Pin Lo, Alan Mink, Takuya Ikuta, Toshimori Honjo, Hiroki Takesue, and William J Munro. A simple low-latency real-time certifiable quantum random number generator. *Nature communications*, 12(1):1056, 2021.

[85] Marco Avesani, Davide G Marangon, Giuseppe Vallone, and Paolo Villoresi. Source-device-independent heterodyne-based quantum random number generator at 17 gbps. *Nature communications*, 9(1):5365, 2018.

[86] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2002.

[87] Hendrik W Lenstra Jr. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987.

[88] Arjen K Lenstra, Hendrik W Lenstra Jr, Mark S Manasse, and John M Pollard. The number field sieve. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 564–572, 1990.

[89] Craig Gidney and Martin Ekera. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, 2021.

[90] Élie Gouzien and Nicolas Sangouard. Factoring 2048-bit rsa integers in 177 days with 13 436 qubits and a multimode memory. *Phys. Rev. Lett.*, 127:140503, Sep 2021.

[91] Nouédyn Baspin, Omar Fawzi, and Ala Shayeghi. A lower bound on the overhead of quantum error correction in low dimensions. *arXiv preprint arXiv:2302.04317*, 2023.

[92] Daniel J Bernstein, Nadia Heninger, Paul Lou, and Luke Valenta. Post-quantum rsa. In *Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8*, pages 311–329. Springer, 2017.

[93] Amir H. Karamlou, William A. Simon, Amara Katabarwa, Travis L. Scholten, Borja Peropadre, and Yudong Cao. Analyzing the performance of variational quantum factoring on a superconducting quantum processor. *npj Quantum Information*, 7(1):156, 2021.

[94] Bao Yan, Ziqi Tan, Shijie Wei, Haocong Jiang, Weilong Wang, Hong Wang, Lan Luo, Qianheng Duan, Yiting Liu, Wenhao Shi, Yangyang Fei, Xiangdong Meng, Yu Han, Zheng Shan, Jiachen Chen, Xuhao Zhu, Chuanyu Zhang, Feitong Jin, Hekang Li, Chao Song, Zhen Wang, Zhi Ma, H. Wang, and Gui-Lu Long. Factoring integers with sublinear resources on a superconducting quantum processor, 2022.