POLITECNICO DI TORINO

Department of Mathematical Sciences

M.Sc. in Mathematical Engineering

Final Dissertation

Quantum machine learning and optimisation: approaching real-world problems with a quantum coprocessor

Supervisor:
Paolo Brandimarte

Supervisor:
Marco Gatta

Candidate:
Luca Asproni

Academic Year 2018-2019

Acknowledgements

The five years spent at Politecnico di Torino have been life-changing. Throughout this time the university has been setting up lots of challenges, constantly raising the bar on the amount of effort needed to keep up. There have been some hard times, but I can be nothing but grateful, for those let me mature both professionally and as a person. I owe a lot to my university, because it was able to punish negligence, reward the hard work and overall allowed me to experience priceless satisfaction in the achievement of my goals.

This work is the result of an intense academic journey. It perfectly merges together themes for which I have been growing interest in the past years at university with those that I have learned to love recently, in my work experience. I would like to thank my academic supervisor Professor Paolo Brandimarte and my work supervisor Marco Gatta, who have given me the opportunity to pursue this work and shared their insights throughout the writing of this dissertation. I am grateful to Dr. Marco Magagnini, for his superb guidelines have let me explore a number of fascinating topics. I would also like to express my sincere gratitude to Dr. Davide Caputo, who has overseen this whole work providing meaningful help and giving me the physical insight that I was sometimes missing. His patience and enthusiasm have motivated me the whole time.

I am indebted to my family and parents, for they have always been beside me, ready to celebrate the joyful moments but also proving particularly understanding in the hard times. I would like to thank Paolo and Federico, with whom I shared the most intense hours at Politecnico and have always been a friendly, encouraging presence. I am also grateful to Giulia, Vittorio and Paolo for their constant support and irreplaceable friendship. I owe a lot to Alessandro and Simone, for their invaluable presence over the last couple of years. This five-year journey was only possible thank to the people I was surrounded by. To them and everyone who has enriched this experience I could not be more thankful.

Abstract

This work investigates the application of quantum computing to data science, reformulating machine learning and optimisation problems using a hybrid quantum-classical formalism. The fundamentals of quantum mechanics are presented, limiting the study to the area of concern of quantum computing, but setting a rigorous mathematical framework. Then, an overview of current and expected near-term devices, along with a description of the issues related to noise and the reduced number of available qubits in modern quantum devices, motivate the need for a hybrid approach to solve complex tasks, allowing classical and quantum hardware to work synergistically.

One of the main aims of this work is to present how supervised and reinforcement learning tasks, as well as combinatorial optimisation problems, are modeled in such a way that the computationally expensive parts of the algorithms needed to solve them can be run on a Quantum Computer. Both benefits and drawbacks are discussed: the former are related to a whole new computational paradigm, which is expected to execute classically intractable operations and to exploit the natural speed of quantum systems to evolve towards equilibrium; the latter mainly derive from physical and engineering limitations that currently do not allow to efficiently control large-scale quantum systems.

Four case studies are reported and investigated in this work: an image classification task for a self-constructed dataset, using a Support Vector Machine algorithm exploiting functions which map input data to a quantum state space; a reinforcement learning problem, formulated as a Quadratic Unconstrained Binary Optimisation (QUBO) problem, with the objective of finding a winning strategy for the game of Blackjack by using self-simulated data; the Number Partitioning Problem (NPP), formulated as a QUBO, which can be considered one of the most complex scenarios for quantum computers in terms of qubits connectivity and couplers precision; finally, the limited-assets Markowitz portfolio optimisation problem, mapped to a QUBO model, with the goal of overcoming the complexity of this type of optimisation by testing a quantum-inspired heuristics as the input size is still limited but with a potential to scale up.

Throughout this work both the approach of universal gate-model and adiabatic quantum computing have been investigated, but with a particular focus given to the latter in most of the case studies. For one of the cases studied here, the Number Partitioning Problem, a remote access to a quantum annealer was exploited, which allowed a thorough study of the capabilities of one of the devices that are able to solve the most complex problems among modern quantum computers.

Finally, by posing a particular emphasis on the accuracy and solution quality of the problems of interest, this study allows a comparison between quantum and classical techniques, highlighting the differences and stressing what benefits a quantum or hybrid quantum-classical strategy may bring to approach machine learning and optimisation problems.

Sommario

Questo lavoro esamina l'applicazione del quantum computing alla data science, formulando problemi di ottimizzazione e machine learning attraverso un formalismo ibrido quantistico-classico. Vengono presentati i concetti fondamentali della meccanica quantistica, limitando lo studio all'area di interesse del quantum computing ma allo stesso tempo impostando il contesto matematico in modo rigoroso. Dopodiché, attraverso una panoramica sia dell'hardware presente ad oggi sia di quello atteso per il futuro prossimo, e anche grazie all'esaminazione dei problemi legati al rumore e al ridotto numero di qubit disponibili nei dispositivi quantistici moderni, si motiva il bisogno di un approccio ibrido per risolvere problemi complessi, che permetta ad hardware classico e quantistico di lavorare sinergicamente.

Uno degli scopi principali di questo lavoro è quello di presentare delle strategie per modellare problemi di ottimizzazione combinatoria e supervised e reinforcement learning in modo da far lavorare un Quantum Computer sulle parti degli algoritmi computazionalmente più costose. Vengono discussi sia i benefici sia gli svantaggi: i primi sono strettamente legati alla scelta di un modello di computazione nuovo, dal quale ci si aspetta l'esecuzione di operazioni classicamente intrattabili e lo sfruttamento della velocità intrinseca dei sistemi quantistici di evolvere verso uno stato di equilibrio; i secondi derivano principalmente dalle attuali limitazioni fisiche e ingegneristiche che non permettono un controllo efficiente di sistemi quantistici a larga scala.

Vengono esaminati quattro casi di studio: un problema di classificazione di immagini auto-costruite attraverso un algoritmo di Support Vector Machine, il quale sfrutta funzioni che portano i dati dallo spazio di input allo spazio degli stati quantistici; un problema di reinforcement learning, formulato come Quadratic Unconstrained Binary Optimisation (QUBO), con lo scopo di trovare una strategia vincente per il gioco del Blackjack usando dati auto-simulati; il Number Partitioning Problem (NPP), formulato come QUBO, che può essere considerato uno degli scenari più complessi dal punto di vista della connettività dei qubit e della precisione numerica dei connettori fra coppie di qubit; infine, il problema del limited-assets Markowitz portfolio optimisation, scritto come QUBO, al fine di implementare un'euristica quantum-inspired che possa sormontare la complessità del problema al crescere della dimensione dell'input.

Nel corso di questo lavoro sono stati investigati sia l'approccio di computazione universale gate-model, sia quello di adiabatic quantum computing, con particolare attenzione verso quest'ultimo dato nella maggior parte dei casi di studio. Per uno dei casi studiati, il Number Partitioning Problem, è stato sfruttato l'accesso da remoto a un quantum annealer, che ha permesso uno studio completo delle capacità di tale macchina, in grado di risolvere tra i problemi più complessi che i quantum computer moderni possano affrontare.

Infine, ponendo particolare enfasi all'accuratezza e qualità delle soluzioni dei problemi presentati, questo studio propone un paragone tra tecniche classiche e quantistiche, evidenziandone le differenze ed esaminando i benefici che una strategia quantistica o ibrida quantistico-classica possa portare nell'approccio a

problemi di machine learning e ottimizzazione.

Contents

Chapter 1

Introduction

Quantum computing is a field at the intersection of physics, computer science and mathematics that exploits the capabilities of a new kind of computer based on quantum mechanics.

Quantum computers revolutionise the very foundations of how computations are carried on by machines. While classical computers are built on the concept of bits, i.e. logical values that can be either 0 or 1 and are implemented by devices that exist in one of two possible states, like electrical switches that can either be on or off, quantum computers rely on the idea of quantum bits, or qubits, i.e. a two-state quantum-mechanical system which is simultaneously, with certain probabilities, in both states. This leads to the need for a new approach to computation and information processing, with the expectation that by exploiting the properties of quantum mechanics it is possible, in very specific tasks, to obtain remarkable speedup and boost in performance.

It is very important to stress that quantum computers are not meant to outperform the classical ones in every task [1]. Rather, their purpose is to solve problems that are computationally hard, like integer factorisation, for which Peter Shor, in [2], presented one of the most famous quantum algorithms, database search, supported by an algorithm by Lov Kumar Grover shown in [3], simulation of quantum systems [4], and many other applications in the fields of optimisation [5] and machine learning [6] that are being investigated in the latest years. Hence, it is safe to say that quantum computers are not likely to completely replace their classical counterpart, but rather, they might take on some hard tasks in scientific computing for which the request of computational power is very high.

Theory of algorithms that can be run on such computers has already been developed. However, they usually require qubits to be ideal, i.e. not affected by the intrinsic problems that arise when trying to control a quantum system such as, for example, interactions between the system itself and the environment. Moreover, quantum algorithms usually need a number of qubits that is very high and unlikely to be reached by any device in the near future. For these reasons, while from a hardware viewpoint the quest remains to scale the number of qubits

as much as possible and improve their reliability, from a software viewpoint the main focus of researchers has shifted to finding how current or near-term quantum devices can be used as co-processors, meaning that only some parts of new or existing algorithms are processed by a quantum machine and the overall computation is performed through the interaction between both quantum and classical computers.

Although the latest years have seen an increase in the attention given to quantum computing both by researchers and technology companies [7], quantum supremacy has not been proved yet. In other words, it is yet to be found a well-defined computational task that a quantum computer is able to solve and is intractable by any classical processor. With the hardware that has been developed so far, for real-world problems there is no tangible advantage in using a quantum device over a classical one. However, this is believed to change in the near future and, in the meantime, it is already possible to carry on research using existing small machines, in order to have working algorithms and expertise in the field for when more complex hardware is developed.

Even though some still do not believe that there will ever be a real quantum advantage, the majority keeps contributing to research. The main reason why quantum computers are believed to be able to quickly provide efficient solutions to many scientific problems, especially for those instances in which classical hardware does not have enough computational power, is because qubits exploit the properties of quantum mechanics. The feature of being deterministically neither in state 0 nor 1 until observed is called *superposition*, and we say that a qubit is in a superposition of the two states, meaning that it has a probability of collapsing either into 0 or 1. The second great feature that quantum computers exploit is *entanglement*. When two or more qubits are entangled, measuring one of them is equivalent to observing all the entangled ones. In other words, when a qubit is measured, the state of all the others in entanglement is found deterministically.

These features are at the core of quantum computing and, theoretically, allow extremely fast computations. Specifically, with qubits, we expect an exponential speedup in representing all possible states that bits can be in. For example, if we want to represent the two classical states 0 and 1 on a quantum computer, we only need a single qubit as all information is contained in its superposition state. If we have a two-bits system, we have four possible states: 00, 01, 10 and 11. If we want to represent them on a quantum computer, we need two qubits, which are allowed, as a system, to be in a superposition state of all four states 00, 01, 10, 11. In general, with N qubits we are able to represent 2^N states. The expected speedup in computations becomes clearer when we need to explore all those states; while on a classical computer we have to actually change the state of bits, on a quantum computer all the information related to different states is already contained in the superposition state of qubits. In other words, N qubits represent 2^N states *at the same time*.

Nevertheless, this does not mean that using N qubits for computations is equivalent to having 2^N bits. Qubits live in a quantum state as long as they are not measured: once they are observed, they collapse into one of the 2^N states

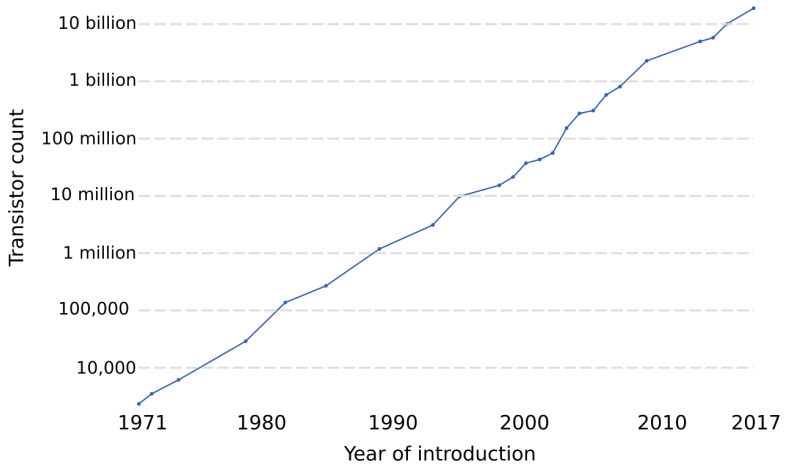


Figure 1.1: Moore's Law describing the number of transistors in a dense integrated circuit over the years. Data provided in [8]

and act as classical bits ever since.

It is clear that, in order to use quantum hardware properly, one needs to understand the basics of quantum-mechanical systems, how they work, what are the main properties that can be used to fully exploit their computational power and what are the problems and challenges in programming such systems, which usually arise from nowadays engineering limitations.

The marvelous promises of speedup boosts and solvability of problems that would otherwise be intractable, such as, for example, simulating complex chemical properties and reactions, which are described by quantum mechanics and are hard to simulate on a classical computer, are not the only reasons why quantum computing has gained so much attention in recent years. Indeed, there are also some concerns regarding progress in development of classical computers. Throughout the years, engineers have managed to build smaller and smaller transistors so that, by being able to collect more of them in a limited space, it has been possible to reach the great computational power that we have today. The number of transistors has been increasing following Moore's law, which states that such number approximately doubles every two years. The trend over nearly the past 50 years is shown in Fig. 1.1. At this pace we have been able to reach a transistors' size in the order of a few nanometers. Some say that soon we will reach a point where it will be impossible to build smaller transistors. Furthermore, as they approach atoms' size, it is believed that classical properties will make space to the rules of quantum mechanics.

Hence, there is plenty of reasons to investigate how quantum computers work, what are their capabilities, how different machines can be compared between one another and how current engineering limitations affect computations. In this work we go through a brief overview of the basics of quantum mechanics

as applied to quantum computing, the mathematical formulation of key topics in this field and how to represent data in a quantum computer. There exist two fairly different computational models, gate-based and adiabatic quantum computing, which means that quantum hardware can be divided in two families that support different kinds of computations, the fundamentals are in common. We are going to investigate which are the differences and how these relate to computational tasks. Next, we have a quick overview of the current stage of progress in the development of these machines.

All of this aims at providing the knowledge needed to understand currently existing algorithms and how to program a quantum computer. As a matter of fact, we are going to study a few recently developed algorithms that are used in a wide range of applications, and we will mainly focus on tasks ranging from optimisation to machine learning.

The main scope of this work is to present how to model both simple and complex data science problems in the quantum computing formalism. We will focus on understanding which techniques are most suitable in different situations and what aspects of the algorithms can be improved. Ultimately we will analyse the performances. This is done by exploring four cases studies, either solved using a quantum computer or a simulator, using classical and quantum hardware in what is called a hybrid approach. We will work on image classification with a self-constructed dataset; on reinforcement learning applied to the search for a strategy to play the game of Blackjack; on the number partitioning problem, which represents a complex task for current quantum computers; finally, we will solve a limited-assets markowitz portofolio optimisation problem formulated as quadratic unconstrained binary models.

Chapter 2

Fundamentals of quantum computing

2.1 Qubits as elements of a Hilbert space

Quantum computers differ from their classical counterpart in that the hardware used for computations is actually a quantum system. As such, their behaviour is intrinsically probabilistic and the outcome of measurements of such systems is described by probability theory. Quantum information theory studies how to formalise the behaviour and make statements about the result of an observation of a quantum system. We shall restrict the rigorous mathematical formulation to finite-dimensional Hilbert spaces, which are of interest in quantum computing.

In the quantum computing framework, the smallest system is composed by only a qubit. In order to describe it, we define its state as an element of a Hilbert space also known as *quantum state space*, which is usually $\mathbb{C}^N$ in quantum information theory, that contains all the information needed to study its probabilistic nature. In our one-qubit system, the Hilbert space is $\mathbb{C}^2$. In such framework, we define the standard basis using quantum mechanics' notation. Let $|\psi\rangle$ be a column vector $\begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \in \mathbb{C}^2$, which we refer to as *ket*, then a basis for $\mathbb{C}^2$ over the field $\mathbb{C}$ is given by:

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

The vectors $|0\rangle$ and $|1\rangle$ are associated with the two possible outcomes 0 and 1 that we can obtain if we observe our qubit, and will be referred to as computational basis. Then, we can write:

$$|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle, \tag{2.1}$$

$\alpha_1, \alpha_2 \in \mathbb{C}$. This formulation highlights why we say that a qubit can be in both states 0 and 1 at the same time, as in Fig. 2.1.

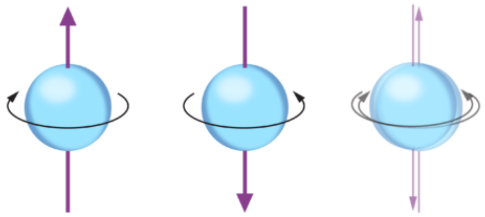


Figure 2.1: From left to right: states $|0\rangle$ and $|1\rangle$ as up and down spins (arrow) of an electron, followed by a superposition of the two states [9]

Now let us define the set of possible events $E = \{e_1, e_2\}$, where event e_i refers to obtaining the $i-th$ outcome from measuring our qubit. The result of the measurement can be described as a random variable X that takes values on the set {'outcome state is 0', 'outcome state is 1'} with support on the space $S = \{0, 1\}$ and associates with each event i a probability p_i that such event might happen. When we go back to our state vector $|\psi\rangle$ and impose the normalisation condition $|||\psi\rangle|| = 1$, we strongly relate $\alpha = (\alpha_1, \alpha_2)$ with $p = (p_1, p_2)$. As a matter of fact, $|||\psi\rangle|| = 1$ implies that

$$\sum_{i=1}^{N=2} |\alpha_i|^2 = 1,$$

or, in other words, $|\alpha_i|^2$ represents the probability of $i-th$ event happening. The entries of vector α are in fact called *probability amplitudes*. From this characterisation stems the superposition property.

When we consider a more complex quantum system, for example one composed by two qubits, the dimension of the Hilbert space increases and, as a consequence, we need to find a new basis. In order to do so, we shall first introduce a compact notation for tensor product.

Let $|\psi\rangle = \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix}$ and $|\phi\rangle = \begin{pmatrix}\phi_1\\ \phi_2\end{pmatrix}$, then we have that the tensor product of the two states is $|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix}\psi_1\phi_1\\ \psi_1\phi_2\\ \psi_2\phi_1\\ \psi_2\phi_2\end{pmatrix}$. It is common to denote the resulting state as $|\psi\rangle \otimes |\phi\rangle = |\psi\phi\rangle$. Bearing this in mind, it is straightforward to define the new computational basis as the set

$$\mathcal{B} = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\},$$

where $|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$, $|01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$, $|10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}$, $|11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$. Then, the vector

notation for a general state of the two-qubit quantum system can be written as follows:

$$|\psi\rangle = \alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle,$$

where $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4) \in \mathbb{C}^4$.

The generalisation to N-dimensional quantum systems is quite straightforward: the Hilbert space becomes $\mathbb{C}^N$, its basis can be written as the set of orthonormal vectors $\{|k_i\rangle\}_{i=1,\cdots,N}, \quad k_1, \cdots, k_N \in \mathbb{C}^N$ and the state of the system takes the form $\sum_{i=1}^{N} \alpha_i |k_i\rangle, \quad \alpha_i \in \mathbb{C}, \quad i = 1, \cdots, N$.

2.2 Operations on qubits

Besides ket, another compact form for elements of the Hilbert space is commonly defined: the *bra*. A bra is the conjugate transpose of a ket:

$$\langle\psi| = |\psi\rangle^\dagger.$$

and, in a finite-dimensional real or complex space it is a row vector.

It is then possible to write the inner product of two quantum states $|\psi_1\rangle$, $|\psi_2\rangle$ as

$$\langle\psi_1|\psi_2\rangle,$$

the *overlap*:

$$|\langle\psi_1|\psi_2\rangle|^2$$

and the outer product as

$$|\psi_1\rangle\langle\psi_2|.$$

Given a Hermitian or self-adjoint operator, i.e. equal to its conjugate transpose, which is identified by a matrix H, its expectation value in state $|\psi\rangle$ is given by

$$\langle\psi|H|\psi\rangle.$$

The outer product is useful for the definition of *density matrices*, which are alternative representation of quantum states. Let $|\psi\rangle$ be the state of a system, then the density matrix of such system is given by

$$\rho = |\psi\rangle\langle\psi|$$

and we refer to $|\psi\rangle$ as a *pure state*. This name stems from the idea that by only knowing $|\psi\rangle$ we can represent a system in which all its particles are in the same physical configuration. For example, take a single-qubit system and the pure state $|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle$, then its density matrix is written as

$$|\psi\rangle\langle\psi| = \begin{bmatrix} |\alpha_1|^2 & \alpha_1\alpha_2^* \\ \alpha_1^*\alpha_2 & |\alpha_2|^2 \end{bmatrix}$$

and if $|\alpha_1|^2 > 0$, $|\alpha_2|^2 > 0$, with $|\alpha_1|^2 + |\alpha_2|^2 = 1$, then our system is in a superposition state of $|0\rangle$ and $|1\rangle$. We have important information contained

in the complex coefficients α_1, α_2, i.e. we know phase and amplitudes of the corresponding states $|0\rangle$ and $|1\rangle$. On the other hand, there exist *mixed states*, which only give information about probabilities of finding the system in one state or another. The density matrix of a mixed state is defined as the sum of density matrices of pure states, weighted by probabilities:

$$\rho_{mixed} = \sum_{i=1}^{\#\text{ pure states}} p_i |\psi_i\rangle\langle\psi_i|.$$

This is the most general formula for a density matrix. It also allows to see that density matrices have unit trace, are positive semi-definite and Hermitian. Specifically, the eigenvalues are p_i and the eigenvectors, also called *eigenstates*, are $|\psi_i\rangle$. Since p_i's are probabilities, they are non-negative real values and sum up to one, from which stem the positive semi-definiteness and unit-trace property. Then, $|\psi_i\rangle\langle\psi_i|$ is hermitian because it holds

$$(|\psi_i\rangle\langle\psi_i|)^\dagger = \langle\psi_i|^\dagger |\psi_i\rangle^\dagger = |\psi_i\rangle\langle\psi_i|,$$

and since density matrices are convex combinations of hermitian matrices, with coefficients being real values, they are hermitian too.

In our single-qubit system, the density matrix of a mixed state takes the form

$$\rho_{mixed} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix},$$

where $a + b = 1$, $a, b \in \mathbb{R}_+$. The mathematical representation of the absence of information about phase and amplitude in a mixed state is given by the off-diagonal elements of ρ_{mixed} being equal to 0. Mixed and pure states can be represented geometrically in a very specific form, which will be discussed further in section 2.8, and is very important in determining whether there is control over the quantum system or not. Problems connected with mixed states will be presented in section 2.10.

2.3 Measurement of quantum states

Having defined density matrices, we can now focus on how to make measurements of quantum systems. The quantity that can be measured, or *observable*, in a physical system composed of qubits depends on what kind of hardware is used to represent the quantum bits. For example, it can be the spin of atoms or the polarization of light. This topic will be covered in greater detail in section 2.11.

Let $|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$. In order to find the probability to measure our qubit and find it in state 0, we simply need to project $|\psi\rangle$ onto $|0\rangle$, which in turn means that we must construct the density matrix $|0\rangle\langle 0|$, compute

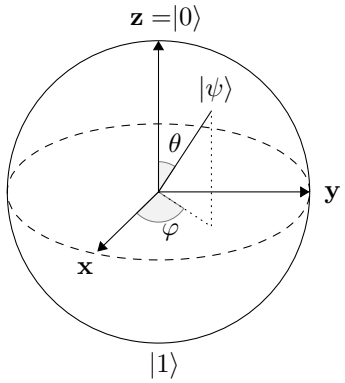
$$|0\rangle\langle 0|\psi\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \alpha_1|0\rangle$$

and finally take the $\ell - 2$ norm of the result, which is equal to $|\alpha_1|^2$ since, by construction, $|0\rangle$ has unit norm. In other words, we projected $|\psi\rangle$ onto $|0\rangle$ and computed the length of the projection. The reason why such density matrix is needed is that it leads to calculating $\langle 0|\psi\rangle$, which is a complex scalar measure of similarity to state $|0\rangle$.

Also, it is easy to verify that the matrix $P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ truly identifies an orthogonal projection operator $P_{|0\rangle} : \mathbb{C}^2 \to \mathbb{C}^2$ onto state $|0\rangle$: $\mathbb{C}^2$ is a Hilbert space and as such is endowed with a scalar product $(\cdot, \cdot)_{\mathbb{C}^2}$. Then, it is immediate to see that for any two vectors $x, y \in \mathbb{C}^2$ it holds $(P_0 x, y) = (x, P_0 y)$. The same can be verified for $P_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ and generalised to higher dimension.

2.4 Geometrical representation of qubits

A measurement of a qubit system is simply a projection of its state onto the computational basis. Fortunately, there is a common representation of qubits' states that helps understanding what happens when we make measurements or modify a state: the Bloch sphere, shown below.



The Bloch sphere is a three-dimensional representation of qubit state space, with $|\psi\rangle$ being a general state and $|0\rangle$ and $|1\rangle$ the computational basis.

By construction, the state of a qubit is a unit norm vector in $\mathbb{C}^2$. Hence, to visualise it, we should represent a four-dimensional real space. However, the formula 2.1 that describes any general (pure) state $|\psi\rangle$ is commonly rewritten by exploiting the constraint $\langle\psi|\psi\rangle = ||\psi||^2 = 1$ as

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right), \tag{2.2}$$

where γ is a real number and $0 \leq \theta \leq \pi$, $0 \leq \phi \leq 2\pi$. The global phase $e^{i\gamma}$ has no observable effect and is therefore omitted from now on. As a consequence, $|\psi\rangle$ can be represented on a unit sphere in $\mathbb{R}^3$, called Bloch sphere, and is defined

Notation	$\boldsymbol{\theta}$	$\boldsymbol{\varphi}$	$\boldsymbol{\lvert\psi\rangle}$
$\lvert 0\rangle$	0	$\varphi \in [0, 2\pi]$	$\lvert 0\rangle$
$\lvert 1\rangle$	π	$\varphi \in [0, 2\pi]$	$\lvert 1\rangle$
$\lvert +\rangle$	$\pi/2$	0	$\frac{\lvert 0\rangle + \lvert 1\rangle}{\sqrt{2}}$
$\lvert -\rangle$	$\pi/2$	π	$\frac{\lvert 0\rangle - \lvert 1\rangle}{\sqrt{2}}$
$\lvert i\rangle$	$\pi/2$	$\pi/2$	$\frac{\lvert 0\rangle + i\lvert 1\rangle}{\sqrt{2}}$
$\lvert -i\rangle$	$\pi/2$	$3\pi/2$	$\frac{\lvert 0\rangle - i\lvert 1\rangle}{\sqrt{2}}$

Table 2.1: Fundamental quantum states. Recall equation 2.2, omitting the global phase, for the relation between the two angles θ and φ and the formulation used in $|\psi\rangle$

by the two angles θ and ϕ. This representation holds for a single qubit; for more complex systems composed of multiple qubits, the space dimension increases and thus there is no suitable visual depiction.

Being an internal point or lying on the surface of the sphere has a strong physical meaning: superficial points correspond to pure states, while the internal ones indicate mixed states. To show this, for simplicity, we shall consider the two-dimensional case. Let ρ be a density matrix describing any single-qubit system. Then, it can be written as

$$\rho = \frac{1}{2}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + \frac{a_x}{2}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + \frac{a_y}{2}\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix} + \frac{a_z}{2}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix},$$

where $\vec{a} \in \mathbb{R}^3$ contains the coordinates of the state of the system and a_i corresponds to its component along axis i. Eigenvalues of ρ are $\lambda_1 = \frac{1}{2}\left(1 + ||\vec{a}||^2\right)$ and $\lambda_2 = \frac{1}{2}\left(1 - ||\vec{a}||^2\right)$, but since density matrices are positive semi-definite, it must hold that $||\vec{a}||^2 \leq 1$. For pure states, the following relation must hold:

$$tr(\rho^2) = tr(\rho)^2$$

while, in general, it is true that $tr(\rho^2) \leq tr(\rho)^2$. A thorough proof of this is provided in [10]. Recalling that the trace of a density matrix is 1, we have that

$$tr(\rho^2) = \frac{1}{2}(1 + ||\vec{a}||^2) = 1 \Leftrightarrow ||\vec{a}||^2 = 1.$$

In other words, the vector $\vec{a}$ that represents the point in $\mathbb{R}^3$ corresponding to the state of our system lies on the surface of the unit ball if and only if such state is pure, otherwise it represents a mixed state and is an interior point of the sphere.

Among pure states, it is possible to spot some fundamental ones that come in handy in many quantum computing algorithms. Table 2.1 summarizes them. Exception made for $|0\rangle$ and $|1\rangle$, they are all superposition states with equal

Gate	Matrix form	Starting state	Result
X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$\|0\rangle$	$\|1\rangle$
Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	$\|0\rangle$	$i\|1\rangle$
Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\|1\rangle$	$-\|1\rangle$
R_ϕ	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$	$\|1\rangle$	$e^{i\phi}\|1\rangle$
H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	$\|0\rangle$	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
CNOT	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	$\|10\rangle$	$\|11\rangle$

Table 2.2: Common quantum gates and the result of their operation performed on some states chosen for example purposes. The choice of the starting state is motivated by the effect of each gate. Here are taken into account states that change when the presented gates are applied. Such statement is not true for every couple state-gate. For example, applying Z gate to $|0\rangle$ would still yield state $|0\rangle$ and is not worth showing here

probability of being in 0 or 1. It is common to prepare qubits in one of such states at the beginning of an algorithm so that it starts from an unbiased scenario, where bias here refers to the probability of collapsing in one of the two states being higher than its complementary. How to change the state of a qubit will be discussed in further detail in the next section.

2.5 Quantum gates

In order to control qubits' state, similarly to how it works with classical computers, we need gates. The operation that a gate performs on a qubit can be visualised as a rotation on the Bloch sphere of a vector $|\psi\rangle$ representing the qubit's state. We assume from now on that the starting state on which a gate is applied is $|0\rangle$.

Table 2.2 summarizes fundamental gates that are commonly used. Gates X, Y and Z are often denoted as σ_x, σ_y and σ_z and in their matrix representation are called *Pauli matrices*. They operate rotations around, respectively, axes x, y and z. X gate is also known as bit-flip gate, as it turns $|0\rangle$ into $|1\rangle$ and $|1\rangle$ into $|0\rangle$. Z gate and R_ϕ gates operate shifts on qubit's phase, hence leaving unchanged its probability of collapsing into 0 or 1. The former is also called

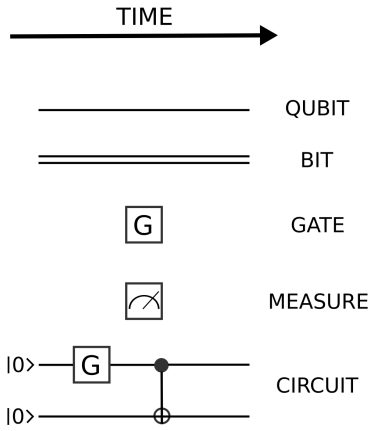


Figure 2.2: Elements of a quantum circuit. For single-qubit gates G, symbols in circuits are the same as the gate name shown in table 2.2. The other single-qubit operation is the measurement, whose symbol is shown here. Bottom panel shows a simple circuit of two qubits; a general gate G is applied to the first one and then a CNOT entangles qubits. The order of application of gates goes from left to right, as shown on top by the time direction.

phase-flip gate and can be reconstructed through the latter.

One of the most important gates is H: the Hadamard gate. If performed on a qubit in a state that is in the computational basis $\{|0\rangle, |1\rangle\}$, it puts the qubit in a superposition state with equal probabilities, respectively $|+\rangle$ or $|-\rangle$. If applied again, i.e. computing $H|+\rangle$ or $H|-\rangle$, the result will be again, respectively, $|0\rangle$ or $|1\rangle$.

Finally, the CNOT gate, or CONTROLLED-NOT gate, is one of the most common tool to entangle qubits. In the example shown in table 2.2, given a system in state $|10\rangle$, applying the CNOT gate changes its state to $|11\rangle$, i.e. the controlled qubit, if it is in $|0\rangle$ and the other one is in $|1\rangle$, is flipped to $|1\rangle$. In other words, the whole system does not behave as an ensemble of independent qubits, but instead the state of its components are strongly correlated one with another and the behaviour of one qubit is deterministically determined by the other one. More details on entanglement are presented in section 2.6. Fig 2.2 is a graphical representation of the elements that compose circuits. They are typically drawn as a collection of single solid lines, one per each qubit, on which are orderly applied gates. At the end of computation, a measurement is performed and therefore qubits collapse and turn into classical bits. An example of a basic circuit is shown in bottom panel of Fig. 2.2, implementing a general gate G and a CNOT gate in order to build an entangled two-qubit system. Usually all qubits are initialised in state $|0\rangle$. Quantum computer must be able to easily prepare qubits in an initial state, before starting any computation. More on the basic principles on which quantum machines are built will be covered in 2.10. Another common approach, depending on the circumstances, is to apply a Hadamard

gate before other computations, so that qubits are put in a superposition state of equal probabilities.

Mathematically, gates are represented by matrix that are applied to quantum states in vector notation. Nevertheless, not all matrices are eligible to represent gates, but some conditions must be met. Specifically, they must be unitary matrices. Let U be a complex-valued matrix, then it is unitary if it holds

$$U^\dagger U = I.$$

Such matrices preserve the intrinsic reversibility of quantum mechanics, which means that modifying twice a quantum state according to the information contained in U will result in the initial state again. In quantum computing, the alteration of a state consists of applying a unitary that, by construction and in order to maintain the properties of quantum systems, is a reversible operation.

Furthermore, unitary matrices preserve norms. In quantum computing, the Hilbert space in which qubits states live is $\mathbb{C}^N$, therefore the inner product of such space induces the usual Euclidean norm. We exploit it to show that unitary operators preserve norms:

$$||U|\psi\rangle||^2 = \langle\psi|U^\dagger U|\psi\rangle = \langle\psi|\psi\rangle = |||\psi\rangle||^2,$$

where $|\psi\rangle \in \mathbb{C}^N$. As a consequence of this property, applying gates to qubits does not change probability amplitudes, which is fundamental because it is through ordered set of gates, or *circuits*, that we are able to build and run most quantum algorithms, resembling how computations are carried on on classical computers, and do not need to care whether the application of gates affect the probability of an outcome.

The gates studied in this section are the main building blocks of most quantum algorithms. However, the questions of what kind of computations they allow and whether we need more gates to perform all possible algorithms arise. The former question was answered in [11], in which it is shown that any operation on qubits in order to change its state can be decomposed into a combination of a finite number of gates. The collection of such gates is called *universal gate set*, where universality refers to the possibility of performing any arbitrary operation.

Further studies show that the universal gate set is not unique. As a matter of fact, it has been proved that we do not need rotation gates around all axis to be able to perform any single-qubit operation, as first researches in the field pointed out. Instead, it is possible to decompose any arbitrary unitary using only the following gates [12]:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, \quad R_{\pi/4}\begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}, \quad CNOT = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix},$$

where the first two gates are single-qubit operations, while the last one is a two-qubit unitary.

One of the main problems related to reducing the number of gates in the universal set is that, in order to approximate a complex unitary, we need to apply many times the fundamental gates and therefore create longer circuits. This creates many issues on current hardware: the application of each gate nowadays is carried successfully only up to a certain precision, or *fidelity*, and it leads to new sources of noise, forcing the system to interact with the environment and, as a consequence, gradually lose quantum properties. Engineers' challenge is to improve the system quality, which means, overall, to build hardware that approaches as much as possible the behaviour of a closed-system. More details on these issues are described in section 2.10.

Finally, a machine that exploits the action of gates over qubits in order to control their state and perform computation is called, for the reasons described in this section, *universal* or *gate-model quantum computer*. On the other hand, there exist machines that work in a very different way: they exploit quantum properties but do not control the state of their qubits at each step of the computation. This approach is called *Adiabatic quantum computing* and will be discussed in greater detail in section 2.9.

2.6 Entanglement

One of the main differences between classical and quantum systems is entanglement. It has been shown that such property is necessary to provide remarkable speedup in some quantum algorithms [13].

Let $|\psi\rangle$, $|\phi\rangle$ be two quantum states in the space spanned by the computational basis $\{|0\rangle, |1\rangle\}$. As described in section 2.1, we can write

$$|\psi\phi\rangle = \alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle,$$

with $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \mathbb{C}^4$. In such scenario, the state $|\psi\phi\rangle$ is simply given by the product of $|\psi\rangle$ and $|\phi\rangle$, therefore the two are unentangled. Entangled states can not be written as a linear combination of $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. For example, the well known Bell states form an orthonormal basis for the space of all states given by the entanglement of two qubits:

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}, \frac{|00\rangle - |11\rangle}{\sqrt{2}}, \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \frac{|01\rangle - |10\rangle}{\sqrt{2}}.$$

Suppose any of the Bell states, for example the first one, can be written as the product of $|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle$ with $|\phi\rangle = \alpha_3|0\rangle + \alpha_4|1\rangle$. Then we would have

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} = \alpha_1\alpha_3|00\rangle + \alpha_1\alpha_4|01\rangle + \alpha_2\alpha_3|10\rangle + \alpha_2\alpha_4|11\rangle,$$

which has no solution.

Entangled states show very interesting properties that stand out more clearly when measuring them.

Take for example the first Bell state $|\psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Suppose we want to make a measurement on one of the two qubits that compose the system and then investigate what happens to the second one. By convention, the measurement is performed on the right-most qubit. This means that, in order to measure it onto state $|0\rangle$, we need to apply to $|\psi\rangle$ a projection matrix of this form:

$$\mathbb{1} \otimes |0\rangle\langle 0| = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

from which stems clearly that the first qubit is left unobserved since we apply the identity matrix to it, while the second one is measured on state $|0\rangle$. Then we get

$$(\mathbb{1} \otimes |0\rangle\langle 0|)|\psi\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}|00\rangle.$$

From this we can compute the probability of the system being in state 00, which is $||\frac{1}{\sqrt{2}}|00\rangle||^2 = \frac{1}{2}$. But most importantly, at this point we have made a measurement on our system, therefore the right-most qubit now has to be treated as a classical bit as it collapsed onto a classical state. Likewise, since the two qubits were entangled, also the left-most qubit has collapsed onto a classical state. This can be seen by making another measurement on the resulting state $\frac{1}{\sqrt{2}}|00\rangle$. If we measure the left-most qubit onto state $|1\rangle$, we get

$$(|1\rangle\langle 1| \otimes \mathbb{1})\frac{1}{\sqrt{2}}|00\rangle = \frac{1}{\sqrt{2}} \left(\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \end{bmatrix} = \vec{0}.$$

The probability of the left-most qubit collapsing onto state 1 in such scenario is equal to 0. Hence, when the first qubit collapses onto the classical state 0, so does the second one, *deterministically*.

Computations thus far confirm what intuition might say when taking a closer look at the Bell state we chose. $|\psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ is actually describing a superposition state of $|00\rangle$ and $|11\rangle$, each with equal probability $\frac{1}{2}$. It follows that when one of the two qubits is in classical state 0, the other one collapses onto 0 as well. Likewise, if one is in state 1, then so is the other one. This kind of reasoning can be made for other Bell states too and, in principle, even for more complex systems, even though intuition can't help as much when the number of qubits grow.

2.7 State evolution

The evolution over time of a quantum mechanical system is described by the Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}|\psi\rangle = H|\psi\rangle, \tag{2.3}$$

where $\hbar$ is the Planck constant, $|\psi\rangle$ is the state of the system and H is the Hamiltonian operator. The Hamiltonian is a hermitian operator that carries information about the energy of the system and its formulation changes depending on the situation.

The study of Hamiltonians, their eigenvalues and eigenvectors, also called eigenstates, plays a central role in quantum computing. The smallest eigenvalue is called ground state energy and the corresponding eigenstate is the state in which the system reaches the lowest value of energy. Any other state for which the corresponding eigenvalue is greater is called *excited state*. Adiabatic quantum computing relies on the study of Hamiltonians, the investigation of how to encode objective functions of optimisation problems as the Hamiltonian of a quantum system and how to reach the ground state energy, which corresponds to the minimum in the optimisation problem. More on this will be covered in section 2.9.

Nevertheless, the study of Hamiltonians is a key topic in some algorithms designed for gate-model quantum computers, too. It is possible to prove [14] that, given an eigenstate $|\psi_0(0)\rangle$ at time $t = 0$ of a Hamiltonian H, its evolution over time can be written in terms of the eigenvalue E_0 of H:

$$e^{-iH_0t/\hbar}|\psi_0(0)\rangle = e^{-iE_0t/\hbar}|\psi_0(0)\rangle,$$

yielding the solution of the Schrödinger equation as a superposition of the eigenstates of H:

$$|\psi(t)\rangle = \sum_j c_j e^{-iE_jt/\hbar}|\psi_j(0)\rangle,$$

where the time evolution operator $e^{-iE_jt/\hbar}$ is unitary and hence preserves the $\ell - 2$ norm of the state.

This is a fundamental result in the quantum computing field as it allows to study the evolution of a quantum system through the operator $U(H,t) = e^{-iHt/\hbar}$, which is unitary and, as discussed in section 2.5, can be reproduced as a combination of fundamental gates.

Algorithms developed in recent years strongly use the knowledge about Hamiltonians that derive from specific problems in order to build circuits that approximate the behaviour of quantum systems in the energy landscape described by such Hamiltonians. This practice is useful in many fields, from quantum systems simulation to machine learning. A more detailed description of some of these algorithms is presented in chapter 3.

2.8 Information encoding

In order to understand how to use quantum computers to solve complex problems, especially those arising from optimisation and machine learning fields, we need a way to map real-world data into qubits. A number of techniques have been developed, allowing us to represent any kind of data in a quantum computer. Thorough descriptions of many of them are presented in [15]. Here we cover the most common, and probably most straightforward, ones.

2.8.1 Basis Encoding

The most obvious technique to encode data in a quantum computer is to replace a bit $b \in \{0, 1\}$ with a qubit $|b\rangle$. In $n-$qubit systems an $n-$dimensional bit string $b_1 \cdots b_n$ takes the form $|b_1 \cdots b_n\rangle$. This technique has both advantages and disadvantages.

The greatest advantage consists in the ease of preparing qubits' state, which means that before any computation starts it is not hard to manipulate qubits in such a way that their state is either $|0\rangle$ or $|1\rangle$. Real values can be easily represented by expanding them in binary encoding, saving one qubit to keep information about the sign of such values.

Even though this technique is rather straightforward and easy to implement in principle, it requires an incredibly large amount of qubits, in the order of the number of bits in classical computers. Current hardware only provides a limited amount of qubits, which makes it nearly impossible to solve large problems using such a technique. Moreover, basis encoding is best suited for input data that are already binary by nature.

2.8.2 Amplitude Encoding

In order to overcome problems related to the number of currently available qubits, a common approach consists of encoding data into amplitudes. Let $\vec{x} \in \mathbb{R}^2$, then it is possible to represent this vector with only one qubit as:

$$|x\rangle = x_1|0\rangle + x_2|1\rangle,$$

where x_1, x_2 are the two components of vector $\vec{x}$ and make up the amplitudes of qubit $|x\rangle$. This technique is called *amplitude encoding*.

The advantage with respect to other encoding rules is that, in order to represent an $n-$dimensional vector, we need only $\log_2(n)$ qubits. This approach is valid for any $n \in \mathbb{N}$ and does not strictly require it to be a value such that $\log_2(n)$ is still an integer. As a matter of fact, one way to generalise this is presented in [15]: let $\vec{x} \in \mathbb{R}^3$, then it is always possible to use a padding technique in order to augment its dimensionality up to an arbitrary value. In our case, we would transform $\vec{x}$ into a vector of the form $(x_1, x_2, x_3, 0)$ that can be encoded in a two-qubit system as follows:

$$|q_1\rangle = x_1|0\rangle + x_2|1\rangle$$

$$|q_2\rangle = x_3|0\rangle + 0|1\rangle.$$

This way we are able to represent any kind of data, at the cost of paying particular attention to the amplitude of state $|1\rangle$ of $|q_2\rangle$: we must remember that it is an auxiliary value that should not influence computations.

Even though amplitude encoding seems appealing, it does pose some complex challenges. The main problem that arises when using this kind of encoding is that state preparation is not trivial. Indeed, custom routines have to be investigated in order to adjust qubits state such that their amplitude is exactly equal to the value of data that needs to be encoded. Furthermore, it may be required to apply a high number of gates in order to reach the desired state, which not only can be hard to find, but also introduce noise in the system.

2.8.3 Hamiltonian encoding

A completely different approach from the ones presented previously consists of encoding data in the Hamiltonian of a quantum system. The rationale is that this way we do not have to modify amplitudes of qubits in order to match our input, but rather we need to be able to find a way to represent data in the form of a Hamiltonian matrix.

In gate-model quantum computing, the procedure of solving a problem consists of approximating the Hamiltonian with a series of unitaries that can be implemented as gates. In general, this is not a trivial task and needs careful study of gates fidelity and approximation error.

On the other hand, for some problems especially in optimisation and machine learning fields, there exist straightforward mapping to well-know Hamiltonians. These are the cases for which an adiabatic computing approach is most suitable as hardware is built in such a way that qubits *naturally* follow the evolution over time described by 2.3 with the information contained in a specific Hamiltonian.

2.9 Adiabatic quantum computing

In this section we are going to study in greater details an approach to quantum computing that is very different from the gate-model one: adiabatic quantum computing.

While universal gate-model quantum computers focus on controlling a system of qubits in every step of its state's evolution, applying gates that operate unitary transformations and reproduce, in principle, any kind of state evolution, adiabatic quantum computers' goal is to encode specific Hamiltonians and let qubits naturally end up in the lowest energy state, following Schrödinger equation 2.3.

In Fig. 2.3 is shown a sketch of an energy landscape drawn as a function of the configuration of the system. The function that describes the energy of the system is its Hamiltonian. In order to use adiabatic quantum computers, also called *quantum annealers*, to solve optimisation problems, the objective function

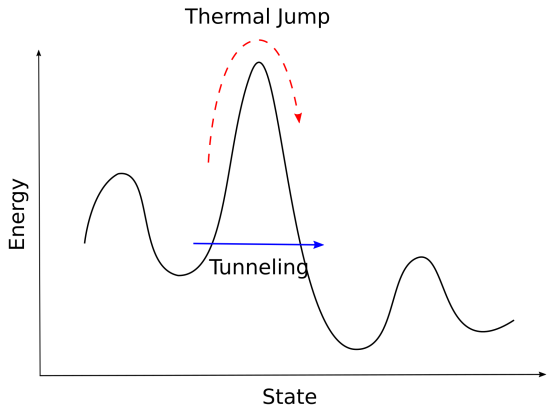


Figure 2.3: Sketch of an energy landscape described by a Hamiltonian and the comparison between quantum annealing, which exploits the quantum tunneling property (blue), and simulated annealing, which works via thermal jumps (red) to escape local minima.

is mapped to a Hamiltonian, hence the minimisation of the latter implies that also the former is minimised.

In order to find the configuration that minimises the system's Hamiltonian, qubits undergo a process called *quantum annealing*, which motivates the choice of the name for this kind of computers. Quantum annealing works in a very similar way to its classical counterpart Simulated annealing: qubits start in an initial configuration to which is associated a certain value of energy (value of the optimisation problem's objective function); then, throughout the annealing, the system tends to go in lower-energy states (i.e. minimises the objective); the goodness of the system's configuration is determined by energy (in the classical approach the search of solution is dependent on a temperature parameter); finally, the system reaches a state that minimises, at least locally, its energy (converges to a local or even optimal minimum).

The main difference between the two approaches is a property of quantum systems, called *tunneling*, to jump on different energy levels without having to climb uphills in the Hamiltonian, as shown in blue in Fig. 2.3. On the other hand, simulated annealing is forced to accept, with a probability that depends on temperature, solutions that are worse with respect to the current one, otherwise it would not be able to escape local minima. This is also called *thermal jump* and is shown in red in Fig. 2.3.

Since quantum annealing is a process that strongly relies on properties of physical systems that naturally try to reach the ground state configuration, and as a consequence does not provide a full control over the system, it has been possible to build hardware that provide an incredibly higher number of qubits, if compared to gate-model computers.

However, the strong limitation of quantum annealers, as already briefly men-

tioned, is that they are not universal computers, meaning that they cannot perform *any* operation as their counterpart do. In mathematical terms, this means that quantum annealers support only a single type of Hamiltonian, given by the Ising model. Let σ_i denote the spin of $i-th$ qubit in an $n-$qubit system, then the Ising Hamiltonian takes the form:

$$H_{Ising} = \sum_{i=1}^{n} h_i \sigma_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} J_{i,j} \sigma_i \sigma_j, \tag{2.4}$$

where, of course $\sigma_i \in \{-1, 1\}\ \forall i = 1, \cdots, n$ and h_i and $J_{i,j}$ are coefficients called *biases* and *couplers*.

The strict limitations of being able to solve only a subset of optimisation problems, and not constantly controlling qubits' state, have been the cause of many arguments on whether quantum annealers were in fact quantum computers. Nevertheless, constant progress in their development has recently lead to quite well performing machines. Researchers and technology companies now use them to investigate their capability in a number of large-scale tasks [16, 5].

Such a success in latest years is due to the fact that the range of real-world problems that quantum annealers can solve is quite wide. As a matter of fact, it is trivial to show that it is possible to transform the task of minimising H_{Ising} into minimising a quadratic function.

Consider the following change of variables for any $i = 1, \cdots, n$:

$$x_i = \frac{\sigma_i + 1}{2}.$$

Then

$$\begin{cases} x_i = 0 & \text{if } \sigma_i = -1 \\ x_i = 1 & \text{if } \sigma_i = +1 \end{cases}.$$

Exploiting this formulation and the relation $x_i = x_i^2\ \forall i = 1, \cdots, n$, it follows that

$$\min_{\sigma} H_{Ising} \iff \min_{x} x^T Q x, \tag{2.5}$$

where σ is a vector of all spin variables, x contains all the binary ones and Q is a matrix whose diagonal entries depend on biases h_i, while the off-diagonal ones depend on couplers $J_{i,j}$.

Hence, quantum annealers appear to be a suitable choice for solving Quadratic Unconstrained Binary Optimisation (QUBO) problems. Moreover, with a Lagrangian formulation of constrained models it is possible to extend the range of tasks for which annealers apply to quadratic constrained problems. More on this will be covered on case studies in chapter 5.

In order to obtain remarkable performances from the use of quantum annealers, a lot of study both from a hardware and a software point of view must be done. Before showing why, we shall explain some more details about quantum annealing.

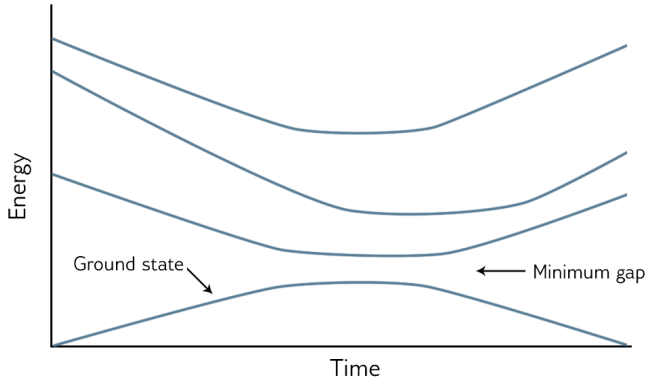


Figure 2.4: Representation of energy levels of Ising Hamiltonian drawn as a function of annealing time. Bottom curve indicates how the lowest energy configuration varies throughout the annealing. Curves in higher parts of the picture represent energy levels of excited states. It is also shown the point of minimum distance (gap) between ground state and the first excited state. Image from [17]

Fig. 2.4 shows the evolution over time of energy levels associated with an Ising Hamiltonian. The configuration of a system that is subject to such Hamiltonian can be in either the ground state or excited state. Throughout a process that modifies the energy landscape, a system can jump on different levels because of many reasons like vibrations caused by external noise, therefore it is possible that, from the ground state, our system goes into an excited state, which no longer minimises the Hamiltonian.

Quantum annealing is a process that modifies the Hamiltonian of a qubits' system. A simple example of this is shown in Fig. 2.5 As a consequence, along with the introduction of such modifications, energy levels higher than the one corresponding to the ground state emerge and get closer and closer to the ground state energy, up to a point of minimum distance, called *minimum gap*. Then, once this critical point is passed, as shown in Fig. 2.4, energy levels drift apart.

The reason why quantum annealing modifies the energy landscape is related to how qubits' state must be prepared in an quantum annealer. First of all, we must define a simple Hamiltonian for which the ground state is known. Typically it is given by all qubits being in a superposition state. Then, starting from the lowest energy state, we must gradually introduce the Hamiltonian term that derives from our own specific QUBO problem, so that eventually the Hamiltonian of the system will be described only by the problem Hamiltonian and the initial one will be negligible. During this process, however, new energy levels are introduced and with them the probability of jumping out of the ground state. In other words, we can describe the overall Hamiltonian of the system as

$$H(t) = H_{Initial}(t) + H_{Problem}(t),$$

where $H_{Initial}(0) \gg H_{Problem}(0)$ and $H_{Initial}(T) \ll H_{Problem}(T)$, being

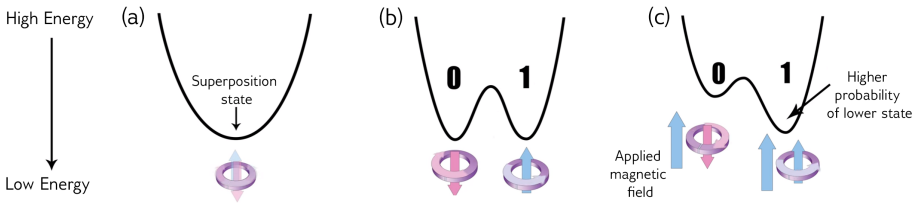


Figure 2.5: Picture illustrating the stages of quantum annealing. The ground state of the starting energy landscape is well known and typically consists of a superposition state (a), then a problem Hamiltonian is applied gradually (b) until it completely takes over the initial one (c), modifying the landscape and hence the state that minimises the energy. Image courtesy of D-Wave [17]

T the final instant of time of the whole process. If the annealing is too fast, meaning that the problem Hamiltonian takes over the initial one very quickly, the system is more likely to jump from the ground state to an excited one.

Fig. 2.6 is a qualitative sketch of the energy as the problem Hamiltonian is introduced. Moreover, it shows a comparison between a simple annealing and one that, at some point, has been paused and then restored. This is a feature that some annealers provide and it has been empirically proved that doing so can improve performances [18, 19]. The reason behind this, as presented in [18] is that, with a pause, qubits are allowed to enlarge the solution space and hence less likely end up in local minima.

In order to truly obtain improvement in performances, we also need to choose carefully the instant of time in which pausing the annealing [18]. If this occurs much before the minimum gap point or far past it, there would be no effect on the probability of staying in the ground state. However, if it is performed right before the minimum gap point, qubits will likely stay in the lowest energy state and end up in the global minimum.

Fig. 2.6 is also an example of the application of a technique called Reverse Annealing. It consists of stopping an annealing process and starting a new one, where the initial configuration is given by the final one of the previous process. This approach is very common in classical algorithms and reverse annealing is its quantum version.

Fig. 2.6 also provides the order of magnitude of annealing time. Even though it is a qualitative representation of the annealing process, a whole cycle performed by an annealer is usually in the order of dozens of microseconds, which is what makes them an interesting heuristics worth investigating.

2.10 Computation in the NISQ era

Quantum computing theory has been investigated for decades, but only in recent years quantum hardware has been developed. This creates a gap between theory and practice.

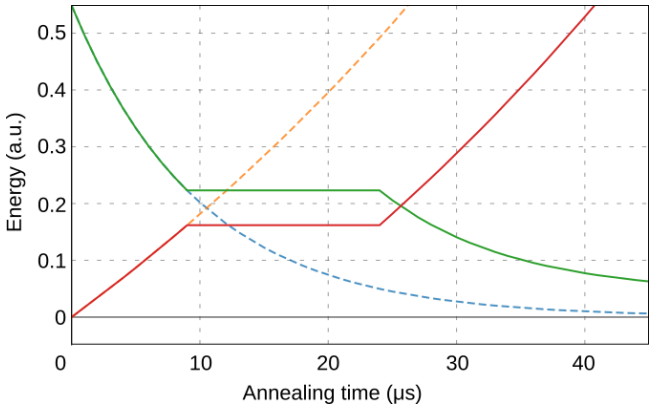


Figure 2.6: Change in energy during the annealing process as the problem Hamiltonian takes over the initial one (respectively, orange and blue). Moreover, the same evolution is qualitatively drawn by allowing for an annealing pause (respectively, red and green). Image taken as part of the work in [19]

In the past, a lot of work carried on by researchers focused on building algorithms that would be able to run on somewhat ideal quantum computers. This means that the basic assumptions comprised qubits being unaffected by any source of noise, i.e. they were supposed to compose a closed quantum system, and also available in a huge number. It is very hard at this point in time to tell whether such computers will ever exist. What is certain is that they are neither available now nor in the near future.

Some might say that the development of quantum computers, at this stage, is comparable to that of classical computers in $1950's$. Current machines have low number of qubits, engineers struggle at isolating their systems and improvements on gates fidelity is in constant progress.

Nevertheless, current quantum computers form an interesting playground for what might be able to come next. Even if we do not reach the hardware complexity required by most well-known algorithms, the ones that promise polynomial or even exponential speedup with respect to classical approaches, it is believed that the day will come when noisy, small quantum computers will be able to accomplish some tasks so well that classical computers will be outperformed on them.

In this section we are going to investigate what are the main challenges that affect today's computers development. We are going to focus on the mathematical formulation of some problems and their implications on data science applications.

Quantum annealers aside, we might say that this is an era in which computers provide fairly few and noisy qubits. J. Preskill defined it as 'Noisy Intermediate-Scale Quantum' era for the first time in [20], and soon this term caught on. It mainly refers to universal quantum computers and essentially

sums up modern engineering challenges. "Intermediate-scale" points out that the number of qubits available in a NISQ computer is at least equal to 50.

In the NISQ era, some fundamental principles lead the construction of quantum computers: DiVincenzo's criteria, thoroughly described in [21]. Essentially, in order to build a proper - according to DiVincenzo - quantum computer, we must be able to:

- Build a scalable system with well characterised qubits,
- Easily initialise the qubits' state to a simple state, like $|000...0\rangle$,
- Allow for the decoherence time to be long enough, in particular longer than gates operation time,
- Identify and implement a universal gate set,
- Measure qubits in an easy way.

In order to get a grasp of the idea behind quantum computers' development, the above points are quite self-explaining. A key topic is decoherence time and is worth describing in greater detail.

Recall that given a state $|\psi\rangle$, an equivalent representation is the density matrix $\rho = |\psi\rangle\langle\psi|$. In general, we have

$$|\psi\rangle\langle\psi| = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

Off-diagonal entries b and c are called coherences. When $b, c > 0$, ρ describes a pure state.

Modern quantum systems do not behave as closed systems. Instead, they still interact with the environment and therefore are affected by external sources of noise. This means that as time goes by and the interactions intensify, the pure quantum state *decoheres* and transforms into a mixed state. This means that the density matrix eventually takes the form

$$\rho = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}.$$

As we have seen in section 2.2, a mixed state no longer carries information about phase and amplitude. Hence it is not possible to control such state and modify probability amplitudes arbitrarily. Instead, its behaviour follows classical probabilistic theory.

Decoherence is a real curse for quantum computing: once the system's coherences vanish from its density matrix, we are no longer able to make computations. It follows that all operations must be performed *before* the decoherence time.

Moreover, a great issue regarding the control of a system is one that gate-model computers have to face: energy relaxation. If qubits are given enough time, and they are in an excited state, they will eventually end up in the ground state, hence losing information on previous states. This time must be long enough so that gates application is feasible and computations are not spoiled.

Furthermore, external noise can corrupt qubits state. When this happens, such qubits are no longer reliable and therefore cannot be used for computations. In order to overcome this problem, a number of techniques have been proposed. One of the first has been presented by Shor in [22], paving the way for future works, some of which have been comprehensively summarised in [23], that focus on the idea of using multiple qubits in order to correct any single-qubit error. The intuition that motivates this approach is that information carried by one qubit that is affected by noise can be stored in other qubits through suitable circuits. This way, even if a state gets corrupted, its information is not completely lost. Thorough explanation of how error correction is performed is beyond the scope of this work.

However, it is fundamental that, in order to perform the operations that are being studied in quantum computing theory, more qubits are needed. From this stems the difference between *logical* and *physical* qubits. In one word, these two kinds of qubit are different in that the latter are noisy, while the former are not. Ongoing research focuses on improving quality of qubits and isolation of the systems, with the goal of exploiting less and less physical qubits to compose a logical one.

2.11 How to build a quantum computer: hardware overlook

In this section we investigate how quantum computers are built and compare the different technologies that are being developed. It is important to stress that we are not going into great details as it is beyond the scope of this work. However, a brief overlook is worth studying in order to better understand where we are now, what are the most important factors to take into account when studying quantum computing hardware and what we can do with current machines, what are their limitations and which ones are more promising.

2.11.1 Superconducting circuits

The greatest challenges in the development of better quantum computers consist of keeping a system isolated and improving the quality and time of gates' application.

One strategy of implementing qubits is by using superconducting circuits. They are made of superconducting loops connected by Josephson junctions, i.e. devices made up of two superconducting electrodes separated by a barrier, that can be tuned by applying a magnetic field in order to create a superposition state of up and down spins. Fig. 2.7 shows one way in which they are interconnected

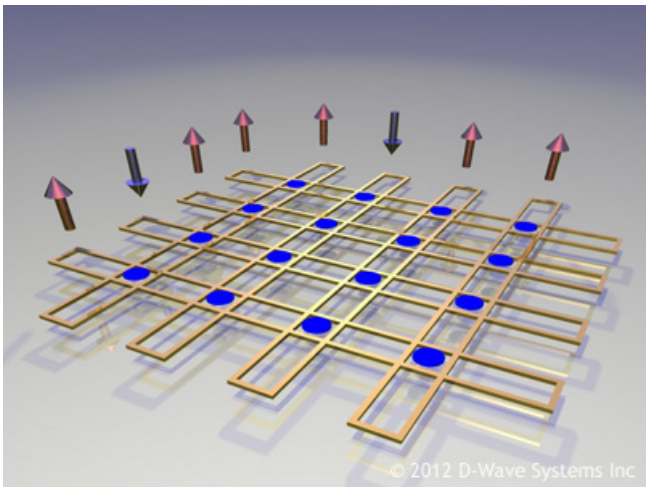


Figure 2.7: Layer of superconducting qubits, courtesy of D-Wave Systems Inc. Blue points indicate couplers that let interactions between qubits and arrows represent up or down spin. Further details can be found in [24]

in order to form a lattice. Such choice is due to the fact that two-state (spins) systems naturally encode the two levels 0 and 1 of a qubit.

These are also called *transmon qubits* and, in order to control them, they must be put in a state where temperature is close to absolute zero. Fig. 2.8 shows the structure of an IBM quantum computer: it contains a refrigerator that reaches lower and lower temperatures as layers go from top to bottom, where the chip containing superconducting circuits is placed.

Qubits are physically implemented in 2D lattices. As a consequence, connectivity between them becomes a central topic in the study of quantum systems. Being disposed in a 2 dimensional layer, it is not possible to connect each qubit with all others. This means that, for example, especially in gate-model computing we cannot entangle any pair of qubits directly. In general, if we have some complex task that can be described by a graph in which nodes are qubits, we may be unable to embed such graph into the architecture of computers. This is very often the case because qubits connectivity in modern hardware is too trivial to be able to represent complex graph that usually arise from real-world problems. Fig. 2.9 shows the architecture of two IBM machines that are accessible online via cloud. It also includes information on each qubit about rates of corruption of quantum states due to the application of gates.

D-Wave Systems computers are very similar on the outside to Fig. 2.8, but in fact rely on different circuit architectures. Fig. 2.10 shows how a D-Wave machine looks like on the outside: a cube that contains the isolated and refrigerated computer, which can be programmed with the least (so far) interactions with the environment.

D-Wave's quantum annealers are built upon a Chimera architecture, whose

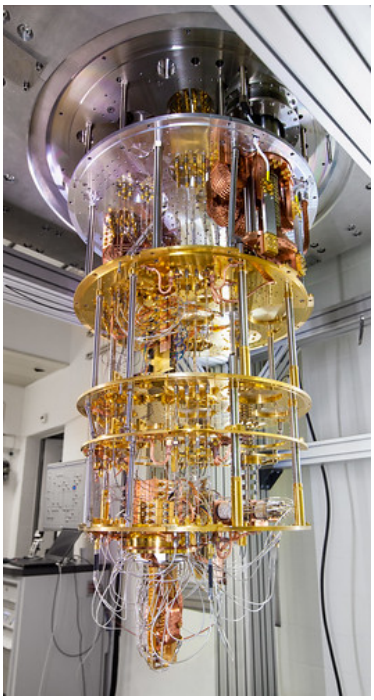


Figure 2.8: IBM's quantum computer. Refrigerator that cools down qubits, with the actual chip containing superconducting loops at the very bottom of the machine

subset is shown in Fig. 2.11. The number of qubits is incredibly higher than those present in gate-model machines. Latest D-Wave computers provide more than 2000 qubits, all disposed in the Chimera architecture. As already mentioned, this is possible because quantum annealers do not aim at constantly controlling qubits' state, but rather build up the energy landscape and let them end up naturally in the ground state. This allows for a use of a much greater amount of qubits and the possibility to scale up even more. Ongoing researches work on structures featuring more than 5000 qubits [25] and supporting a graph architecture called *Pegasus*, presented in [26].

With the topology allowed by quantum annealers, it is possible to embed much more complex graphs with respect to those that can be studied with gate-model computers. Even though they are able to solve only quadratic unconstrained optimisation problems, their topology makes them much more suitable for such tasks, at least at the moment, than universal machines.

Quantum annealing attempts to close the gap between research and industry, for which gate-model computers are still struggling because of the restricted amount of qubits that they provide.

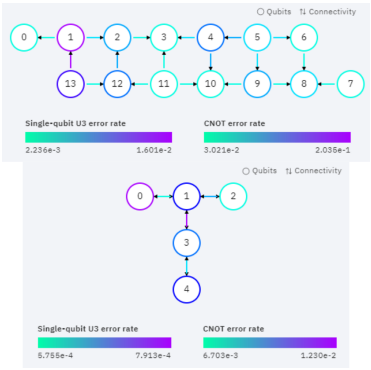


Figure 2.9: Top panel shows ibmq_melbourne featuring 14 qubits, while the bottom one is ibmq_ourense providing 5 qubits. Both pictures reveal information about connectivity of qubits supported by the corresponding machine and about rates of corruption of quantum states due to application of gates. $U3$ is a compact notation for a rotation of a qubit configuration of both angles θ and φ, according to notation in section 2.4

Figure 2.10: D-Wave's 2000Q quantum computer in an environment that can be programmed from the outside

2.11.2 Trapped ions and beyond-NISQ technologies

While superconducting qubits are physical realisation of classical hardware that is brought to exhibit quantum properties, there are some other ways to build qubits that actually use particles.

Some, like IonQ, have based their technology on the use of ions. Specifically, they *trap* ions, which are the actual qubits, in a system that works at environmental temperature. This allows to overcome the problem of bringing the whole

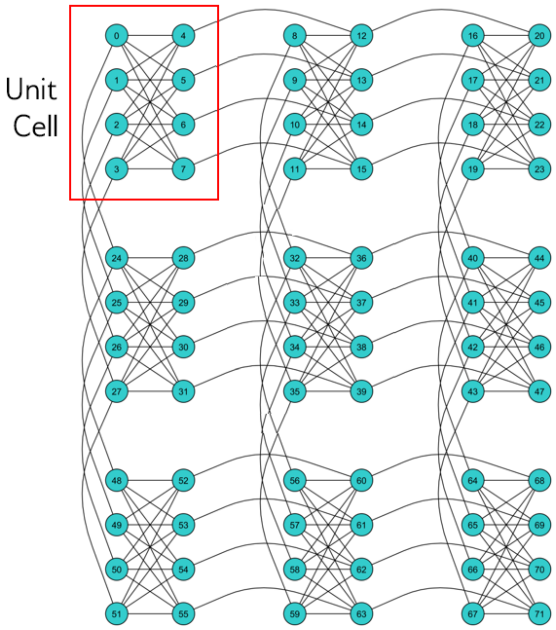


Figure 2.11: Illustrative example of a Chimera graph, as presented by D-Wave in [27]. Each unit cell is a bipartite graph where each node is connected with another unit cell

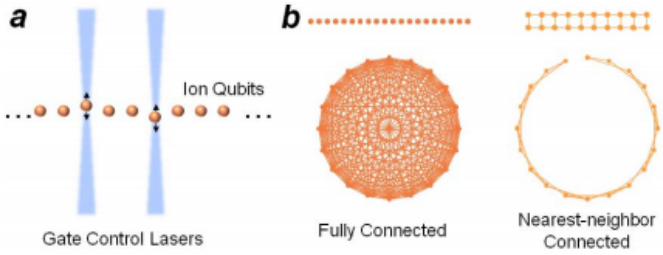


Figure 2.12: Picture from [28] representing **a** the control of ions through laser beams and **b** two examples of arbitrary topology that trapped ions can support

machine near absolute 0 temperature, which is very costly and fairly difficult to implement. Moreover, such trapped ions systems are believed to be able to scale up easily.

These computers allow qubits to be embedded on a topology of arbitrary complexity, up to that of a complete graph. Fig. 2.12 shows a couple of examples for the fully-connected and nearest-neighbour-connected cases.

It has been possible to host up to 160 trapped ions, but better control of

operations over qubits must be investigated. Latest news show that there have been experimented simple operations on 79 qubits and more complex quantum algorithms on 11 qubits [29]. Nevertheless, one of the most important achievements reached with this technology has been the ability to successfully simulate a water molecule [30]. This result paves the way for further improvements towards simulation of quantum systems, which can be of critical help in a number of fields, such as chemistry and health sciences.

Besides superconducting qubits and trapped ions, new technologies are being developed and if successfully realised, they immediately go beyond the NISQ era. One of these is composed by *topological qubits*.

In this framework qubits are realised on top of the concept of Majorana fermions. They are both particle and their own antiparticle, but their study goes beyond the scope of this work and will thus not be presented here any further. Topological qubits are implemented through electrons and holes, which recall the nature of Majorana fermions. Specifically, this kind of quantum computing exploits the phenomenon of electrons being able to be delocalised at the ends of superconductors.

Delocalisation can be recreated in laboratories and allow electrons to avoid any interaction with the environment. On one hand this seems appealing, as we would overcome any problem related to noise and create logical qubits right away without the need for error correction, but on the other hand this property prevents topological qubits from being controlled deterministically in a successful way. This is a great challenge that, if overcome, will allow the development of already-beyond NISQ era quantum computers.

Finally, another very promising technology consists of *photonic* quantum computing. As the name suggests, the units of computations are photons.

Photonic quantum computing can be useful for representing both binary and continuous variables. The latter approach is based on the notion of *qumodes* instead of qubits. In other words, thanks to photons and all the knowledge about quantum optics, it is possible to encode information in such a way that measurements of qumodes yield real values. In such framework, gates are no longer in the form described so far, because they now need to operate on completely different units, but their investigation is beyond the scope of this work.

Regardless of whether photons are used to encode binary or continuous information, which mathematically translates to measuring different observables, like polarisation or position, their advantages consist of being naturally resistant to environmental noise and unaffected by problems related to heat, as it happens for superconducting circuits. A disadvantage is that they are not as easy to manipulate.

Some researchers have been developing photonic quantum computing led by the idea that it would be most suitable for machine learning tasks, given its continuous nature. There have been a number of studies around this possibility, most of them published in [31].

Not only photonic quantum computing paves the way for interesting future works, but it can also be approached via a Python 3 framework, developed by Xanadu, that makes it easier to program such computers. In its white paper

[32], in which are presented approaches valid both for qubits and qumodes, is also shown a way to merge quantum with classical computations in order to be able to perform a backpropagation algorithm on neural networks composed of both classical and quantum nodes.

Although this technology has been tailor-made for machine learning tasks, and therefore seems appealing for the scope of this work, it has not been investigated thoroughly. The reason is that it is not mature yet, meaning that no working quantum hardware can be accessed, therefore no experiments can be conducted and only theoretical results can be studied.

2.12 Conclusions

In this chapter we have set the mathematical framework of quantum computing, exploring the main properties of qubits and showing what operations can be applied on them. In the context of gate-model computing, we have shown the main gates that allow to change their states and entangle them, in order to arbitrarily alter their probabilities of collapsing into one or the other classical state.

Some strategies to encode classical data into qubits were analysed, building the bridge from practical problems to quantum computation. One of these techniques, the Hamiltonian encoding, was shown to be closely related to the second well-known computational model: adiabatic quantum computing. We have thus examined it, highlighting the differences from the gate-model approach and showing what class of problems can be solved in such a framework.

Finally, an overview of the progress of current hardware was presented. We have gone through the different technologies that allow to build quantum computers, their advantages and disadvantages and introduced the NISQ era concept, which summarises current engineering issues that still limit the scale and power of quantum computers.

In the next chapter we will examine some well-known algorithms that are useful in tackling optimisation and machine learning tasks. We will see how gate-model computers can be used to analyse data in a hybrid framework, where quantum and classical processors work synergistically. The techniques that we will study are at the core of data science and simulation problems solving, as they are both widely used and the concepts on which they are built inspire most other modern algorithms.

Chapter 3

Hybrid quantum-classical computing

Throughout the years a number of algorithms have been developed in anticipation of the realisation of quantum computers that would be suitable to execute them on large scale. These machines are called fault-tolerant, indicating that they do not suffer from noise and all other issues explained in section 2.10.

At the time being no such device exists. In the meantime lots of researches have been carried on with the goal of obtaining a quantum advantage already at this early stage. This led to working within a hybrid environment, in which CPU's and QPU's, Quantum Processing Units, are able to make computation synergistically. Fig. 3.1 shows a sketch of this paradigm.

The key idea behind hybrid approaches consists of revising already existing algorithms or creating new ones in such a way that quantum hardware is exploited for very specific tasks which are computationally heavy and might be executed more quickly on a QPU. In such framework, we do not expect as much speedup as promised in the algorithms working in fault-tolerant computers.

In this and the following chapters we will focus on optimisation and learning tasks. In such situations we are given a loss function that is variable in a number of parameters and we need to find the best configuration of these variables such that the loss, or cost, function is minimised. Maximisation problems easily translate to minimisation ones by changing the sign of the loss function.

A schematic representation of a hybrid approach in these specific fields is given in Fig. 3.2, which shows the general framework in which the loss function is dependent on the results of quantum computation and they do not necessarily coincide, although this is frequent in many applications. A common technique consists of running iterative algorithms in which each iteration may be computationally heavy or can somehow be improved. This way, qubits' state can be encoded in some way dependent on the iteration's parameters, a quantum routine is executed and the final state measured. Measurements are then used to update parameters according to some rule, which for example can be a gradient

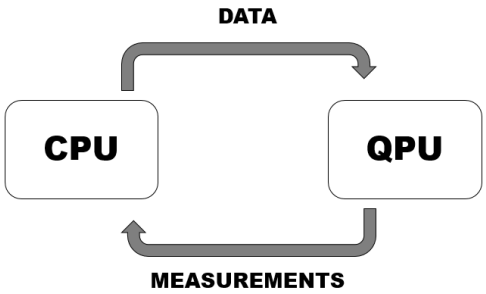


Figure 3.1: Illustrative picture of a hybrid paradigm. Classical data is being processed by classical computers, encoded in a QPU and measurements are stored back in CPU's. Then the evaluation of results from quantum devices are commonly used to update instructions given to the QPU

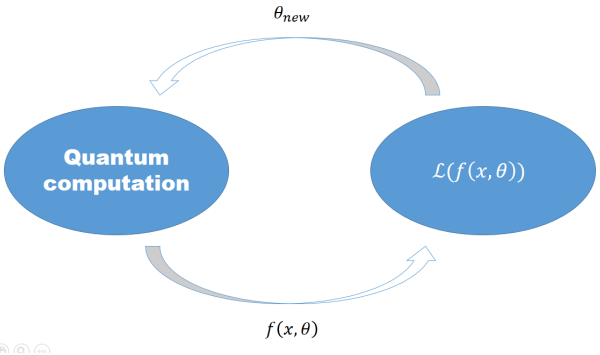


Figure 3.2: Hybrid workflow for learning tasks. A quantum device executes some task dependent on parameters θ; results are sent to a CPU that evaluates a function f; classical computer implements a routine to update parameters based on the new value of a loss function $\mathcal{L}$ evaluated at f. The cycle usually goes on until the loss function converges to some desired value

descent heuristics [33], and keep on with the iterations until a minimum of the loss function is reached. Given the heuristic nature of this approach, convergence to a global minimum is typically not guaranteed and some work must be carried on in order to escape local optima.

The way in which classical computers interact with QPU's changes from one quantum device to another. With gate-model computers, CPU's either are used to prepare circuits parameters in a way that is dependent on input data and iteration that is being processed, or start computation and directly use results from the quantum machine for some other operations. When interacting with annealers, instead, classical computers are mainly used to prepare the

Hamiltonian matrix and to find some suitable way to embed complex models into the available connectivity structure provided by the quantum machine.

In the following sections we are going to cover some well-known algorithms that can be used to solve optimisation and machine learning tasks. We will assume to work with a universal gate-model quantum computer, but will also ultimately present how hybrid computations can be performed using an annealer. Moreover, we suppose the quantum system to work within the context of the Ising model. It supports at most pairwise interactions between qubits, as given by the quadratic term in 2.4, but the following algorithms can be generalised to Hamiltonians with any order of interactions, at the cost of introducing non-trivial complexity in the computations.

The algorithms that we are going to investigate share a common ground with other hybrid routines and therefore build the fundations needed to understand how a general quantum-classical approach can be used to solve real-world tasks, and to build quantum-assisted solvers for optimisation, classification or regression problems.

3.1 Variational Quantum Eigensolver

A very common approach to hybrid quantum-classical computations consists of implementing circuits that are dependent on some parameters $\vec{\theta}$. Qubits are typically initialised to $|0\rangle$ and their state is modified by a number of entanglement and rotation gates. The angles used in the latter are given as a function of variables $\vec{\theta}$. When angles coincide with the values of the parameters, rotation gates take the form $R_{\vec{\theta}}$ and parameters are supplied by a CPU as input to the QPU, as schematically shown in Fig. 3.3.

In an optimisation framework, let $\mathcal{L}(\vec{x})$ be the objective function of a binary minimisation problem. We aim at solving

$$\min_{\vec{x}\in\{0,1\}^n} \quad \mathcal{L}(\vec{x}). \tag{3.1}$$

In order to work with quantum devices, we need to map $\mathcal{L}$ to an appropriate Hamiltonian, namely the Ising Hamiltonian 2.4. Recall that a mapping from binary variables to those taking values in $\{-1, +1\}$ is given by

$$x_i = \frac{1 - \sigma_i}{2},$$

where $x_i \in \{0, 1\}$ and $\sigma_i \in \{-1, +1\}$ $\forall i = 1, \cdots, n$.

Having the Hamiltonian H_{Ising} we can start computations on the QPU. We assume qubits are initialised to state $|0\rangle$, as is common in quantum computing frameworks with no loss of generality, and the set of parameters $\vec{\theta}$ are assigned some real values.

Then, we build an *ansatz*, i.e. a well-defined collection of ordered gates acting on qubits with both rotation operators and entangling ones. The circuit representation is provided in Fig. 3.3. To do so, it is common to identify a unitary

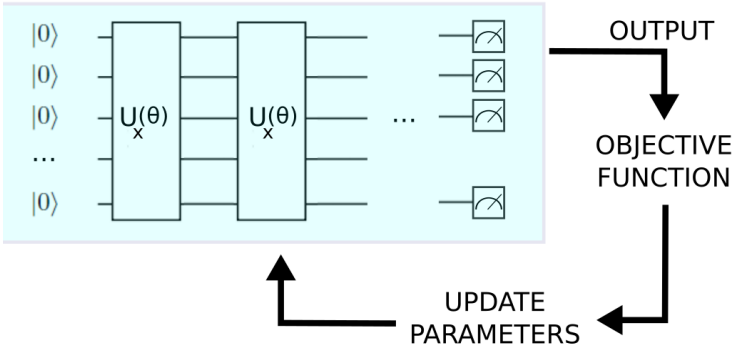


Figure 3.3: Ansatz in a general form in a hybrid computation framework. The highlighted part is a circuit that repeatedly applies a set $U_x(\theta)$ of gates to qubits, which can rotate them by angles θ and entangle them. The value of such angles, in general, may depend on some data x as usual in learning tasks. Measures are computed at the end of the ansatz, passed on to a CPU and used to compute some objective function, and finally the results of current iteration are used to update circuit parameters θ

U composed by single-qubit rotations and entangling gates, indexed by parameters $\vec{\theta}$, and repeat it multiple times. Usually, each unitary starts by applying a Hadamard gate on each qubit in order to immediately switch to superposition states. Stacking multiple U layers leads to an overall circuit $\mathcal{U}$. It is important to stress that this is a common approach, but no golden standard for an ansatz that works efficiently for any problem is available. Intuition helps identifying some choices that are more likely to lead to bad performances, like stacking an incredibly high number of gates and hence exceeding qubits decoherence time, but in general the quality of ansatz is problem-dependent.

Having constructed the circuit $\mathcal{U}$, we exploit the QPU to build the state

$$|\psi(\vec{\theta})\rangle = \mathcal{U}(\vec{\theta})|0\rangle$$

Then we measure the expected value of H_{Ising} in state $|\psi(\vec{\theta})\rangle$, i.e.

$$\langle H_{Ising}\rangle_{|\psi(\vec{\theta})\rangle} = \langle\psi(\vec{\theta})|H|\psi(\vec{\theta})\rangle, \tag{3.2}$$

by sampling the outcome of the circuit multiple times.

From quantum mechanics theory, specifically from the variational theorem, we know that $\langle H_{Ising}\rangle_{|\psi(\vec{\theta})\rangle} \geq E_{gs}$, where E_{gs} is the energy associated with the ground state of H_{Ising}, and the equality holds if and only if $|\psi(\vec{\theta})\rangle$ is the ground state. Therefore, encoding $\mathcal{L}$ as a Hamiltonian implies that the procedure for minimising the latter also minimises the objective function, and viceversa. The

goal is then to find the ground state of an Ising Hamiltonian so that it can be used to recover information on the minimum of the initial objective function $\mathcal{L}$.

Having evaluated the expected value of the Hamiltonian, we switch to the CPU and use some classical heuristics to find a new set of parameters based on the value of the previous ones and on the measurements obtained with the quantum device, aiming at minimising $\mathcal{L}$. We iterate until convergence to a, in general, suboptimal set of parameters $\vec{\theta^*}$, which we can use to retrieve the solution of the initial problem by measuring $|\psi(\vec{\theta^*})\rangle$. The outcome will be a bitstring representing the solution of 3.1. This routine is called Variational Quantum Eigensolver (VQE).

An important remark is that the original problem must be formulated in terms of binary variables. This is strictly related to the nature of the quantum device that we are exploiting. Qubits are two-level systems and therefore the results of their measurements will be either 0 or 1. This mathematically translates to projecting a state using the Z Pauli operator

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

i.e. we project the state onto Z axis of the representation given by the Bloch sphere. This can be explained by writing the frequency of obtaining state 0 as

$$f_0 =< \psi(\vec{\theta})|0\rangle\langle 0|\psi(\vec{\theta}) >$$

and equivalently for state 1:

$$f_1 =< \psi(\vec{\theta})|1\rangle\langle 1|\psi(\vec{\theta}) >$$

from which follows:

$$f_0 - f_1 = \langle\psi(\vec{\theta})|\sigma_z|\psi(\vec{\theta})\rangle.$$

Hence, we are restricted to operating with Hamiltonians that describe two-level systems, or, in other words, we can only encode binary variables.

Regarding the classical optimiser that is in charge of updating parameters $\vec{\theta}$, a common choice is the Simultaneous Perturbation Stochastic Approximation (SPSA) method. The key idea is that $\vec{\theta}$ is updated following the rule

$$\vec{\theta}_{n+1} = \vec{\theta}_n - c_n\hat{g}_n(\vec{\theta}_n),$$

which is the general relation used for updating parameters in an iterative algorithm. $\vec{\theta}_{n+1}$ is the value in current iteration, n indicates the number of iteration, $c_n \in \mathbb{R}$ is a scalar usually called *learning rate* in machine learning applications and $\hat{g}_n$ is an approximation of the gradient of the objective function that is computed by using finite differences. A thorough description of this algorithm is provided in [34]. The great advantage consists of being able to merge it with quantum computations, as $\hat{g}$ reduces to only a couple of evaluations of slight modifications of 3.2.

Finally, the Variational Quantum Eigensolver algorithm can be used to find the lowest eigenvalue of the Hamiltonian matrix. As seen in section 2.7, the value of the ground state energy E_{gs} of H_{Ising} corresponds to the lowest eigenvalue of such operator. This in turn means that when we find the optimal configuration $\vec{\theta}^*$, we can compute the expected value $\langle H \rangle_{|\psi(\vec{\theta}^*)\rangle}$ and thus obtain the eigenvalue of interest. However, it must be noted that VQE does not necessarily find a global solution, which may happen if the matrix associated with H_{Ising} is defined by a non-convex operator. The result can be the eigenvalue corresponding to an excited state.

3.2 Quantum Approximate Optimisation Algorithm

Another hybrid algorithm following the approach shown in Fig. 3.3, which uses a CPU to iteratively update parameters to feed to a quantum device, is the Quantum Approximate Optimisation Algorithm (QAOA)[35].

Its name stems from the idea of letting the system follow an adiabatic pathway, starting from a simple initial Hamiltonian and ending up in a problem Hamiltonian, much like the quantum annealing procedure. The difference from the latter is that QAOA works for gate-model computers, meaning that the adiabatic pathway is approximated by a set of unitaries.

Similarly to VQE, the starting point of this algorithm consists of encoding the loss function, expressed in terms of binary variables, into the problem Hamiltonian H_P and preparing an initial Hamiltonian H_I for which the ground state is known.

Then, our goal is to approximate the system's Hamiltonian, whose evolution over time can be written as:

$$H(t) = (1-t)H_I + tH_P,$$

so that at time $t = 0$, H_I alone defines the energy landscape of the system, while at $t = 1$, H_P takes over the other term.

Recall that the time evolution operator of a quantum system with Hamilonian H is given by the unitary

$$U(H,t) = e^{-iH(t)/\hbar},$$

therefore the state at any time t reads

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

Let us consider the Suzuki-Trotter approximation of such unitary with M discretisation points of δ_t-long time intervals:

$$U(H(t)) \approx \prod_{m=1}^{M} e^{-i[\delta_t(1-m\delta_t)H_I + \delta_t(m\delta_t)H_P]}$$

we can exploit this formulation to rewrite the adiabatic pathway that we want our system to follow as

$$U(H(t)) \approx U(H_P, \gamma_M)U(H_I, \beta_M) * \cdots * U(H_P, \gamma_0)U(H_I, \beta_0),$$

where $\gamma_i, \beta_i \in \mathbb{R}_+ \ \forall i = 0, \cdots, M$ are parameters to be optimised that represent for how long the evolution under each Hamiltonian is applied to the system, and $U(H_P, \gamma_i) = e^{-iH_P\gamma_i}, U(H_I, \beta_i) = e^{-iH_I\beta_i}$.

In principle, the higher is M the better is the approximation, however NISQ devices must take into account decoherence times, which poses the non-trivial question of what is the optimal M.

QAOA algorithm hence exploits the Suzuki-Trotter expansion of the time evolution operator of a general Hamiltonian H, which describes a two-level system, in order to approximate with a finite number of unitaries, each implemented by the QPU as a collection of gates, the behaviour of the system under a specific energy landscape dictated by a minimisation or learning problem.

The paradigm adopted in QAOA is the same as VQE: parameters $\vec{\gamma}, \vec{\beta}$ are initialised, fed to the QPU which implements the approximation strategy described above and computes

$$\langle \vec{\gamma}, \vec{\beta} | H | \vec{\gamma}, \vec{\beta} \rangle$$

by repeatedly sampling, then a classical optimiser updates the parameters. This whole process is iterated until convergence to some, in general, local minimum.

3.3 Conclusions

Hybrid quantum-classical techniques are at the core of modern computational models involving quantum computing. In this chapter we presented how CPU's and gate-based QPU's are commonly used together to solve optimisation and learning tasks.

First, we have analysed the Variational Quantum Eigensolver, a powerful algorithm that relies on circuits whose gates learn how to modify the state of qubits in such a way that a loss function is minimised. Variational circuits are widely used in a number of applications, but still suffer from a limited knowledge on which ansatz might prove more efficient.

Then, we have presented the Quantum Approximate Optimisation Algorithm, which is similar to the previous technique in that it requires to learn certain parameters of the gates, but the choice of ansatz does not represent an issue anymore. QAOA is a clever way of simulating, in principle, two-level Hamiltonians of any degree, but strongly suffers from current engineering limitations.

In the next chapter we will dive into quantum machine learning. We will present strategies to solve machine learning problems using the formalisms of both quantum computational models, the gate-based one and adiabatic computing. In order to do so, we briefly recall how the Support Vector Machine algorithm works and apply it in a quantum-enhanced way to a self-constructed

case study on image classification. Then, we will investigate a quantum-inspired strategy, with the formalism of adiabatic computing, for solving a reinforcement learning problem: finding a winning strategy for the game of Blackjack. In this case too the dataset is self-constructed and a comparison with state-of-the-art classical algorithms will be presented.

Chapter 4

Quantum machine learning

Recent availability of quantum computers has intensified researches on which benefits they might bring to a number of fields. They are expected to provide speedup in some computationally hard tasks and to solve problems that are intractable for classical computers, such as reproducing the physics of a complex quantum system. Since they rely on physical systems that obey the rules of quantum mechanics, they pave the way to a number of strategies for solving complex problems that would otherwise be impossible to explore with classical hardware.

Machine learning is a field that strongly depends on hard computations. Either if one seeks to find patterns or wants to make inference and better understand the behaviour of some data, in order to obtain meaningful results, very complex computations must be done.

Quantum machine learning merges these two disciplines. The core idea is to solve typical problems that arise when we need to make an algorithm learn from data, and rely on a QPU as co-processor to help with computations. It is clear that such approach is intrisically hybrid, also because state of the art quantum processors are incapable of solving alone the large-scale challenges posed by machine learning.

Since we are now in the NISQ era, quantum computers are still noisy and provide a fairly low number of qubits, consequently an actual speedup or boost in performance is not guaranteed. As already mentioned, quantum supremacy is yet to be proved and it follows that the role of modern research in quantum machine learning is to investigate the capabilities of hybrid approaches, often rewriting already existing algorithms which are referred to as *quantum-enhanced* or *quantum-assisted*.

As of today, a fairly low number of companies own a quantum computer. Even less are able to build one. However, some of them provide cloud access to their machines so that it is possible to carry on researches reporting experimental results. In this chapter we are going to apply quantum subroutines for solving machine learning tasks. Experiments have been conducted through cloud access to IBM's gate-model simulators and exploiting the D-Wave's tool *QBSolv*, a

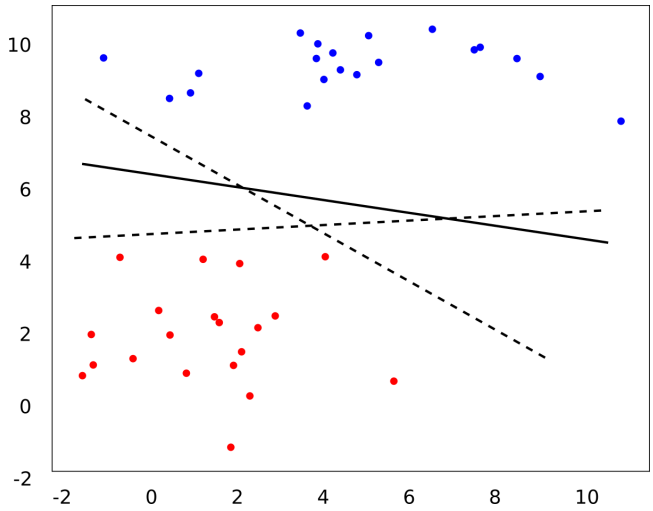


Figure 4.1: Sketch of $d = 2$ dimensional data coloured by label. Dotted lines represent general separating hyperplanes, while solid line shows a possible result of SVM

classical decomposing solver that finds the minimum of large QUBO problems by splitting them into subinstances, which might then be embedded into the quantum annealers' structure, and solving them via classical heuristics, as we will do in this chapter, or by running the QPU's. There are a number of obstacles that do not make it possible to obtain speedup with today's machines, whether they are gate-based or annealers, namely the time required to gain access to a quantum machine and, for the latter, to decompose the original problem in order to match QPU's architecture. This work focuses on evaluating the accuracy of the proposed methods rather than computational time, highlighting both benefits and possible improvements of a hybrid approach.

All experiments have been carried on by exploiting Python 3 frameworks provided by the quantum computing companies themselves. Indeed, they offer facilities to ease programming their machines by providing both low-level languages or libraries that allow to directly control quantum gates and high-level interfaces that contain built-in algorithms such as VQE (see section 3.1), QAOA (see section 3.2) and others.

4.1 Support Vector Machine

Before we dive into the investigation of quantum-assisted algorithms, we shall briefly describe the classical implementation of one of the most famous classification algorithms that are widely used at the moment: the Support Vector Machine (SVM).

Let $\{(\vec{x}_i, y_i)\}_{i=1}^{n}$ be a set of n $d-$dimensional samples $\vec{x}_i \in \mathbb{R}^d$ which are

associated with a known label y_i. It is called *Training set*. In a general learning task, such label can either be a real value, an integer or a binary value. From now on we will assume $y_i \in \{-1, +1\}\ \forall i = 1, \cdots, n$, meaning that the problem we consider is a binary classification one.

The support vector machine algorithm consists of finding a hyperplane that separates the $d-$dimensional space in two subspaces, so that all samples with label $y_i = -1$ belong to one subspace and those with $y_i = +1$ belong to the other one.

As we can see in a simplified example on Fig. 4.1, we could, in principle, find an infinite number of hyperplanes that achieve our goal. Support Vector Machines ask for an additional requirement: the distance of data from the hyperplane must be maximised. This is a reasonable condition: it helps maintaining a certain level of accuracy when the algorithm is applied to new data for which no label is given, also called *test data*. This is clear if we assume to find a general hyperplane that separates well our training data. When we want to classify a new sample, there is a probability that we assign it the wrong label. Intuitively, such probability is lower when the hyperplane that has been found previously is the one furthest away from the data, as the result of its classification takes better into account the variability of data, whilst a badly chosen hyperplane might classify wrongly a data point that belongs to a certain class but is far away from the training samples in the same class. For this reason, SVM maximises the margin of the separating hyperplane, i.e. its distance from data points.

We recall that when we substitute the coordinates of a point into the equation of a hyperplane, the result will be 0 if such point lies on the plane, it will always have positive sign if it is on one side of the plane (i.e. one of the two subspaces) and negative sign if it lies on the other side. With this we are able to write the optimisation problem for finding the SVM hyperplane:

$$\begin{aligned} \max_{\vec{w},b} \quad & \frac{1}{||w||_2} \\ \text{s.t.} \quad & y_i\left[\langle w, x_i\rangle + b\right] \geq 1 \quad \forall i = 1, \cdots, n, \end{aligned} \tag{4.1}$$

where the equation of the hyperplane is given by $h(x) = \langle \vec{w}, \vec{x}\rangle + b$ with $\vec{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$ and $\langle \cdot, \cdot \rangle$ denotes the euclidean inner product. The constraint indicates that we require all data points (x_i, y_i), $\forall i = 1, \cdots, n$ to be classified correctly. If this condition is met, we say that our data is *linearly separable*. However, this is hardly the case in complex datasets; a simple example of non-linearly separable data is provided in Fig. 4.2.

Moreover, there might be other cases in which data is almost linearly separable, meaning that only it is possible to find a separating hyperplane which incorrectly classifies only a few points. In this situation it would be convenient to allow the algorithm to make some mistakes, so that we could get an overall good solution even if not perfect.

In order to overcome these two issues, we shall reformulate the maximisation problem in such a way that the inner product in the constraint takes

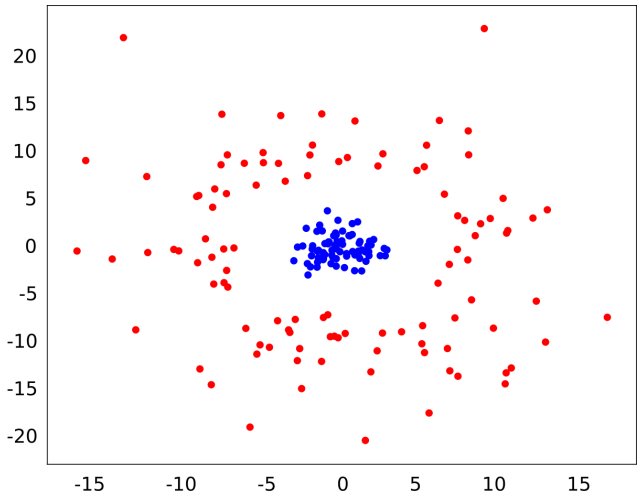


Figure 4.2: Qualitative illustration of 2−dimensional randomly generated data, coloured by class label, that cannot be separated correctly by a hyperplane.

as arguments every pair of data points, and the rationale behind this is that we can then substitute it with some more complex function solving the non-linearly-separability problem and at the same time allow for some misclassifications. Hence, we switch to the Lagrangian form of the problem, exploiting the Karush-Kuhn-Tucker conditions (KKT) for optimality. This leads to the following problem:

$$\begin{aligned} \max_{\vec{\alpha}} \quad & -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_i \alpha_i \\ \text{s.t.} \quad & \sum_i \alpha_i y_i = 0 \\ & \alpha_i \in [0, C] \quad \forall i = 1, \cdots, n \end{aligned} \tag{4.2}$$

where $\vec{\alpha}$ is the vector of Lagrange multipliers, by KKT conditions we have $\vec{w} = \sum_i \alpha_i y_i \vec{x}_i$, which is substituted in the problem above to yield the double sum, and C is the multiplier related to slacks variables that represent how much error we allow and vanish in the final formulation.

By tuning the parameter C, which is usually done via some validation techniques such as Cross Validation [36], we overcome the second problematic scenario described before, allowing for the search of a hyperplane that is able to classify data with an overall good accuracy, even though samples are not linearly separable. On the other hand, with this formulation we are able to replace the inner product $\langle x_i, x_j\rangle$ with a non-linear function $K(x_i, x_j)$, called *kernel*. Specifically, if K satisfies certain conditions [37], we are able to compute an inner product $\langle \phi(x_i), \phi(x_j)\rangle$ in some space that has a dimension higher than

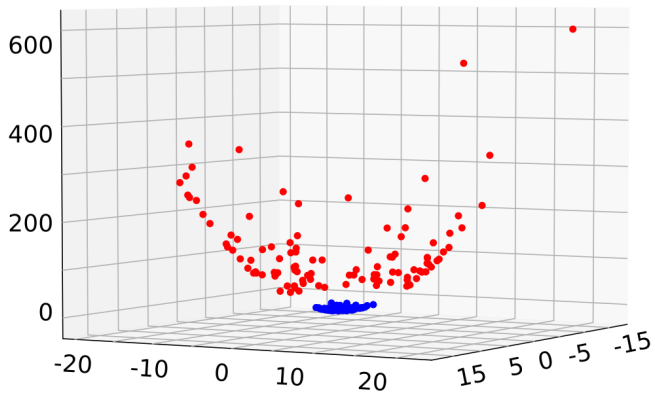


Figure 4.3: Data from Fig. 4.2 mapped onto a higher dimensional space (third coordinate is given by the sum of the first two squared). Here data become linearly separable, motivating the search for a hyperplane in the dimensionality-augmented space.

d, using only information given by $\langle x_i, x_j \rangle$ and without the need to know the analytic form of ϕ. This means that we map our input data into a more complex space, called *feature space*, through a non-linear function ϕ, then we compute the inner product in such space not suffering from the burden of making computations in a high dimensional space, because all we need to know is $\langle x_i, x_j \rangle$. The rationale behind this approach is that if data are not linearly separable in the original space, it may become so in the feature space, as we can see in Fig. 4.3; in such space we find a hyperplane which defines a decision boundary, i.e. the set of points that separate the space in two subspaces. When it is projected back into the original space, it is no longer linear. The sketch of a boundary that could be given by this approach is shown in Fig. 4.4. This procedure is called Kernel Method.

4.2 Support Vector Machines with quantum enhanced features space: a case study on image classification

In this section we are going to apply a quantum-assisted SVM algorithm in order to classify a dataset of images.

As covered in section 4.1, it is possible to build a classifier that linearly separates some training data $\{(\vec{x}_i, y_i)\}_{i=1}^n$, where $\vec{x}_i \in \mathbb{R}^d$, in some space with dimension higher than d so that the decision boundary projected back on $\mathbb{R}^d$ is no longer linear. This is achieved by selecting a suitable kernel K: different choices of K yield different decision boundaries. At the state of the art, no kernel is universally regarded as the best one to adopt, instead its choice strongly

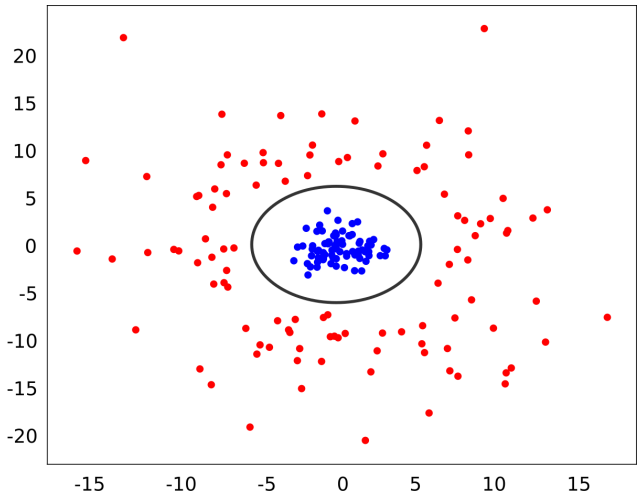


Figure 4.4: Data from Fig. 4.2 with the boundary defined by a linear separator in the space shown in Fig. 4.3 and projected onto original space

depends on the dataset under study [38].

The idea behind a quantum-enhanced SVM, as proposed in [39], is to use a gate-model quantum computer to map input data into the higher dimensional space that is given by the quantum state space. A great difference from the classical kernel method [38] is that now the map is known and well-defined.

Before we dive into the experiment, we shall briefly recall a well-known procedure to reduce the dimensionality of data: Principal Component Analysis (PCA). We will use it as part of the preprocessing phase in order to cut down to a small enough data matrix so that it can match the architecture of IBM's quantum simulator.

Given a collection of observations $\{\vec{x}_i\}_{i=1}^n$, with $\vec{x}_i \in \mathbb{R}^d \ \forall i = 1, \cdots, n$, the goal is to find a new set of features that explain enough information about the data, i.e. are able to faithfully reproduce their variance. PCA consists of finding a number $k \leq \min\{n, d\}$ of mutually orthogonal directions, called *principal components*, in $\mathbb{R}^d$ that maximise the variance in our data. Since each direction is able to explain, in general, less variability than the whole original space, the number k must be sought by taking into account the trade-off between variance explained and the complexity of the resulting space $\mathbb{R}^k$; PCA reads as follows: first, we look for a direction that maximises the variance of data contained in the matrix $X \in \mathbb{R}^{nxd}$ by solving:

$$\max_{\vec{w}} ||X\vec{w}||^2 = \max_{\vec{w}} \vec{w}^T X^T X \vec{w},$$

where $\vec{w}$ is usually asked to have unit norm for computational purposes; then, we calculate a second principal component by solving the same problem as before, where the space is now orthogonal to the first direction found and thus takes

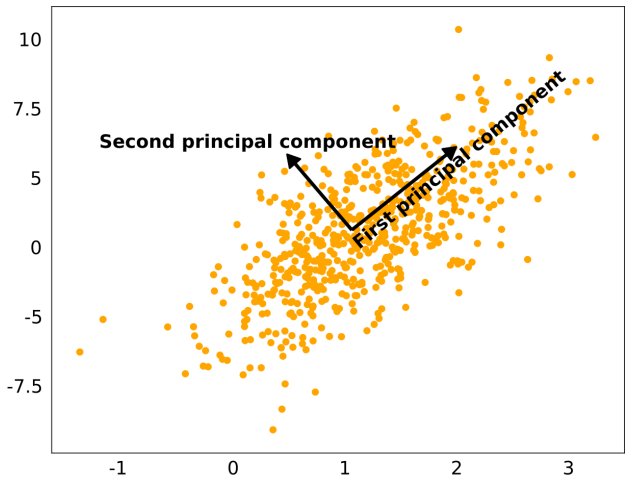


Figure 4.5: Example of application of PCA on 2−dimensional sample data. The first principal component maximises variability in the data; the second one does the same with the additional constraint of being orthogonal to the first one.

the form:

$$X_{new} = X - X\vec{w}\vec{w}^T.$$

A visual example of this process for 2−dimensional data is given in Fig. 4.5. In general, we can calculate the $k-$th component by updating the space with the following:

$$X_k = X - \sum_{j=1}^{k-1} X\vec{w}_j\vec{w}_j^T.$$

In conclusion, Principal Component Analysis is an algorithm that finds the basis of a space of dimension $k \leq \min n, d$, which composes the new set of features that maximise the variance in the data, meaning that they carry as much information contained in the orginal data as possible. Clearly, when $k = \min n, d$ we have the same space as before, but it is expressed in terms of new features.

Dimensionality reduction will be a very important step in our experiment. Having recalled the basic ideas, we are now able to proceed with our analysis. Let us, then, consider a dataset composed by self-constructed and self-labeled images, shown in Fig. 4.6. We have 24 grey-scale images with resolution $20x30$, i.e. $\{(\vec{x}_i, y_i)\}_{i=1}^{24}$ where $\vec{x_i} \in ([0, 255] \cap \mathbb{N})^{20x30}$ and $y_i \in \{-1, 1\}$ $\forall i = 1, \cdots, 24$, where the label indicates whether an image represents a happy or sad smile. Since the goal of this experiment was to investigate what kind of decision boundary would result from the mapping of input data to a quantum state space, and the relative performance, and since these images do not differ in a remarkable way one from another, there was no need to produce a bigger dataset. Moreover,

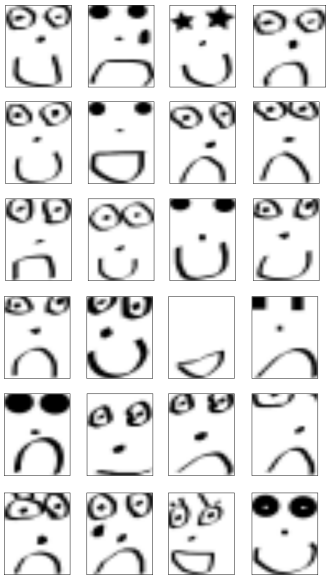


Figure 4.6: Self-constructed and self-labeled dataset of $20x30$ images either belonging to class 'happy' or 'sad'

knowing that the algorithm would be tested on the IBM quantum simulator, which classically reproduces the connectivity and structure of real quantum devices and provided up to 5 qubits when this experiment was conducted, a much higher resolution would have prevented a meaningful embedding into the architecture. Further explanations will be provided when we discuss the relation between input data dimensionality and number of qubits.

The first step consists of generating the data matrix $X \in ([0, 255] \cap \mathbb{N})^{24x600}$, where each row represents an image and each column represents a pixel. Then, we split the dataset in training and test set. The former is composed by 70% of the total number of images, chosen at random, while the remaining ones make up the latter. We consider standardised data, meaning that we make each feature (pixel) have zero mean and unit variance.

After a few simple preprocessing steps, we perform PCA on X in order to reduce its dimensionality. Having operated a standardisation on data allows PCA to weight the importance of each feature correctly, i.e. not being affected by the range of values that they take. Unfortunately, we are forced to work with a number of features as limited as 2, otherwise we cannot feed our data to the simulator.

Having extracted the first 2 principal components, we can analyse the amount of variance explained by each of them. As we can see from panel (a) in Fig. 4.7, neither of them contributes in a very significant way to reproducing the variability of the original data. However, this comes as no surprise given the nature of our dataset: images expressed through 600 pixels are difficult to be

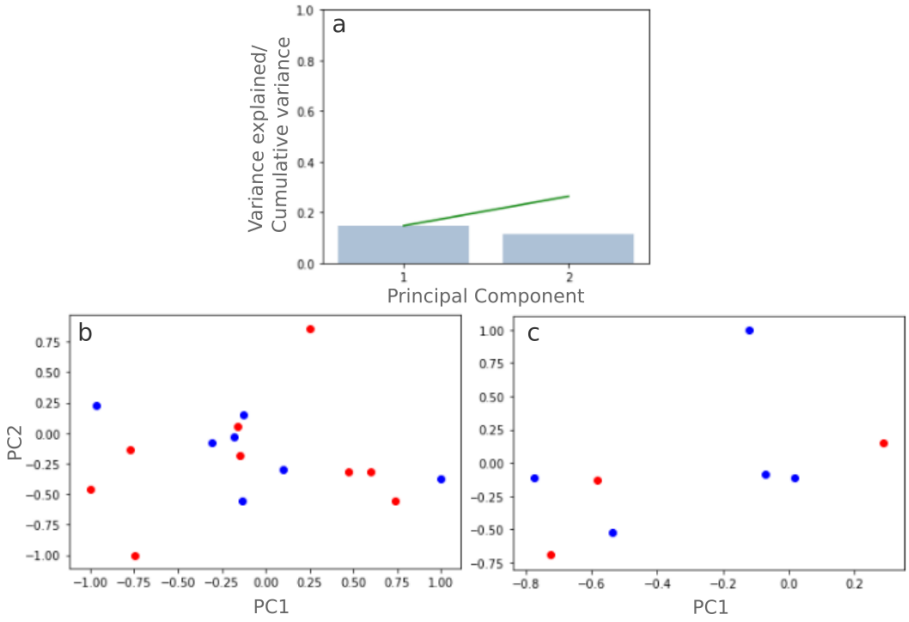


Figure 4.7: a. Variance explained by each principal component shown as a bar graph. Cumulative variance shown by line chart. b-c. plot of data projected onto the first and second principal components, colored by class, respectively of training and test set, where label happy is represented in blue and sad in red

thoroughly represented via only a couple of combinations of the features. As a result, we do not expect to correctly classify all samples in the test set.

Then, before diving into the classification part of the computation, we rescale the two new features given by the principal components to the range $[-1, 1]$. This is motivated by the fact that SVM strongly relies on computing distances, hence similarly to PCA, in general, we want to avoid giving more importance to some features with respect to others, namely the ones that take values in a wider range. In Fig. 4.7, panels (b-c), are shown the projected data into the plane spanned by the two principal components, coloured by class. We can see clearly that such dataset is not linearly separable, hence a kernel trick should come in handy.

At this point, we have all is necessary to begin with an SVM algorithm. We ponder over two similar ways of implementing a hybrid quantum-classical classifier, as proposed in [39].

The first one consists of preparing a set of qubits to state $|0\rangle$, applying a circuit whose gates act in a way that is dependent on the value of data points $\vec{x}_i$, then we build a second circuit parametrised by a vector $\vec{\theta}$ that has to be optimised and finally we make multiple measurements to obtain the probabilities of each sample belonging to the two classes. We iterate this process and use

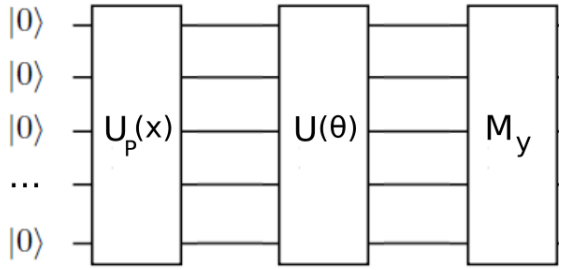


Figure 4.8: General form of a variational circuit. Qubits are initialised at state $|0\rangle$; a unitary U_P applies rotation and entangling gates by quantities that depend on input data $\mathbf{x}$; a set of gates parametrised by optimisation variables $\boldsymbol{\theta}$ modifies the previous states; finally, a measurement on the resulting configuration is applied in order to obtain a label y

some classical optimiser, such as the SPSA discussed in 3.1, to update the value of vector $\vec{\theta}$ until we reach convergence. This approach resembles the Variational Quantum Eigensolver in that we use a variational circuit, i.e. a set of gates indexed by parameters to be optimised, but it also includes a state-preparation part to stress the dependence on input data. A visual representation is given by Fig. 4.8.

In this experiment we build a quantum-enhanced Support Vector Machine following a different approach, even though somewhat similar to the previous one. The reason why the latter was not picked is that it was computationally much heavier. This is unlikely to happen on real quantum hardware, however, we had access to a simulator and therefore could not afford to make too hard operations. The second approach is more balanced in terms of computations performed by CPU and QPU, thus the simulator had to reproduce a less complex dynamics of the quantum system.

The core idea is to exploit the formulation of the SVM optimisation problem in Lagrangian form 4.2. The kernel function computes the scalar product of input data in a higher dimensional space; we are going to map our data into a quantum state space via very specific functions and then compute the squared modulus of the scalar product, also called overlap. Then, after having constructed the matrix of pairwise overlaps, we use it as a new kernel function to feed to a CPU and solve the optimisation problem classically. The algorithm with all necessary steps is provided in Algorithm 1.

Choosing a quantum-enhanced kernel reduces to selecting a set of gates that apply transformations to states in a data-dependent way. This is not trivial and a lot of investigation can still be done in order to understand which are proper circuits. A common idea is to choose some non-linear function of the input data, namely our dimensionality-reduced images $\vec{x}_i \in \mathbb{R}^2$, $\forall i = 1, \cdots, 24$, to obtain suitable values for angles as output. Then, we should feed these values to rotation gates and stack different layers, including entanling gates, in order

Algorithm 1 Quantum-enhanced features space: SVM learning phase

Input: Unlabeled training samples $\{\vec{x}_i\}_{i=1}^{n}$. Number of measurement repetitions R.

Require: Depending on the circuit, number of qubits may derive from their relation with number of features

Initialise all qubits to state $|0\rangle$

Choose a circuit $\mathcal{U}_{\phi(\vec{x})}$ of input-dependent gates

for all pairs $(\vec{x}_i, \vec{x}_j)$ **do**

for r in $1...R$ **do**

Compute $|\phi(\vec{x}_i)\rangle = \mathcal{U}_{\phi(\vec{x}_i)}|0\rangle$

Compute $|\phi(\vec{x}_j)\rangle = \mathcal{U}_{\phi(\vec{x}_j)}|0\rangle$

Compute $o_r = |\langle\phi(\vec{x}_i)|\phi(\vec{x}_j)\rangle|^2$

end for

Store $K(\vec{x}_i, \vec{x}_j) = \frac{1}{R}\sum_{r=1}^{R} o_r$

end for

Output: Kernel matrix $K \in \mathbb{R}^{nxn}$

to obtain a data-dependent circuit.

Which should be the non-linear maps and what gates, and in which order, compose the circuit is still an open issue [39]. In this work we rely on the choices proposed in [39]; such circuit seems reasonable as it is considered to be hard to classically simulate for bigger systems, which should be the driving idea if one wants to ultimately gain a quantum advantage with respect to usual kernels, but when operating on a number of qubits as small as ours, a quantum simulator is able to support the computations.

The circuit of interest was already shown in Fig.3.3, where the unitary is given by Z-phase and entangling gates, yielding the following:

$$U_{\phi(\vec{x})}|\psi\rangle = CNOT R_{\phi_{1,2}(\vec{x})} CNOT (R_{\phi_1(\vec{x})} \otimes R_{\phi_2(\vec{x})})|\psi\rangle,$$

where $|\psi\rangle$ is a general state, i.e. $|{+}{+}\rangle$ before applying the unitary for the first time and $(H \otimes H)U_{\phi(\vec{x})}|{+}{+}\rangle$ for the second time, $\phi_k(\vec{x}) = x_k, k = 1, 2$ and $\phi_{1,2}(\vec{x}) = (\pi - x_1)(\pi - x_2)$. Clearly the number of qubits is strictly related to the dimension of input data, the choice of the function ϕ and hence to the circuit. In this framework, to compute the overlap of two quantum-enhanced data points, for each of them, we need as many qubits as their dimension. Finally, we choose a number $R = 1024$ of measurements to estimate the expected value of the overlap.

Once the learning is finished, the testing phase comes natural by considering the formulation of the SVM classifier:

$$\hat{f}(\vec{x}_{test}) = sign\left(\sum_i \alpha_i y_i K(\vec{x}_i, \vec{x}_{test}) + b\right),$$

where b is the bias term to be estimated separately. All terms are known, except

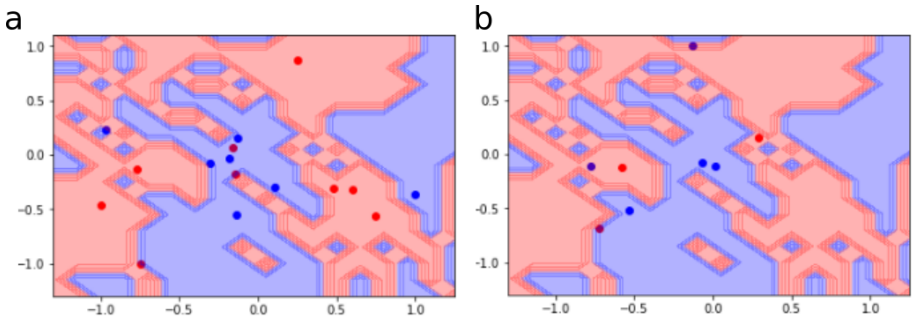


Figure 4.9: Samples and predicted decision boundaries coloured by class. Panel a. reports results on training set, while panel b. on test set. In both cases, data are plotted on the plane spanned by the two principal components and labels happy are drawn in blue, while sad in red

for $K(\vec{x}_i, \vec{x}_{test})$, which we can compute by mapping $\vec{x}_{test}$ to the quantum state space using the same circuit as before.

Following this approach, we have found remarkable results. Fig. 4.9 shows samples and decision boundaries of both training and test set, projected into the plane spanned by the two principal components. As we can see, the result is a non-trivial decision boundary, which confirms that a quantum-enhanced kernel is capable of modeling some high level of complexity in the data. For obvious reasons we do not report accuracy performance of classical algorithms on this dataset: the data have been oversimplified to be fitted on the quantum simulator, hence it is easy to find a classical routine that outperforms our quantum-enhanced SVM. We can see that our classifier makes a couple of mistakes out of 8 test samples, yielding an accuracy of 75% correctly classified data. However, this result can be investigated further, especially for what concerns the circuits that implement the map to the quantum state space; moreover, it anticipates that with more qubits we will be able to analyse larger and maybe more complex datasets, opening the possibility to increase classification performances for data that require highly non-linear kernels.

4.3 Reinforcement learning as a QUBO model: a case study on the game of blackjack

In this section we investigate a way to formulate a reinforcement learning (RL) task as a QUBO model, which is the formalism supported by quantum annealers, as the use of such devices has recently been drawing attention to RL applications [40]. Testing the performance of modern annealers is not the scope of this investigation, hence we will solve our problem using classical heuristics provided with the QBSolv tool; however, this work examines a way to construct the QUBO model for RL tasks which can then be fed to a real quantum processor.

Figure 4.10: Reinforcement learning routine: a player performs an action in a specific environment; the consequences are rewards, or penalties, which the player takes into account for pondering the next action; the process iterates until the player optimises their policy so that rewards are maximised

In this framework an agent, or player, as shown in Fig. 4.10, is set in an environment and takes actions following a policy; then, agents iteratively update their behaviour by analysing the response of the environment so that some criterion is met, such as the maximisation of a profit or minimisation of a loss. Typically the environment is simulated multiple times so that the player is forced to face as many different situations as possible and hence learn how to react to a wide range of circumstances.

Since the early study of reinforcement learning, some have tried - and succeeded - to teach machines how to play a variety of games, ranging from chess to complex videogames [41]. Remarkable results in the gaming industry indicate that it is possible to tackle much complex problems also in other non-ludic fields, because games might provide a number of different situations that is comparable to that of some real-world problems.

In this work we focus on teaching an agent to play a variant of the game of blackjack. In order to do so, we need to introduce a bit of notation:

The game simulations have been performed by following the rules presented in Algorithm 2. Changing them does not influence the performance analysis of the QBSolv in charge of optimising the policy of the player, which is the focus of this experiment. As a matter of fact, a different choice of rules simply leads to a different assignment of rewards.

- The index of current match simulation is m;
- The number of total simulations is M;
- Each match is composed by a sequence of T_m plays, variable depending on the match;

Algorithm 2 Blackjack games simulation: learning phase

Input: Player's initial policy; Dealer's policy; Number of matches M

for m in 1...M **do**

Draw two cards for each player

while the sum of cards of all participants does not exceed 21 & at least one of them chooses 'Hit' **do**

Player chooses an action a_t according to their policy: pick uniformly at random between 'Hit' and 'Stick'

Dealer chooses an action a_t according to their policy: hit if sum is smaller than 17, otherwise stick

Store player's pair (state, action)

Compare sums of cards and check if match is finished

end while

Set reward R_m for each pair equal to: 1 the sum of player is greater than that of the dealer; -1 in the opposite scenario; 0 in a tie

Similarly, set reward R_m for each couple of the (state,action) pairs occurred in the current match

end for

Output: Average rewards associated with each (state,action) pair $R_{(state_action)}$ and each couple of (state,action) pairs $R_{(state,action)_1,(state,action)_2}$

- The state of the player at each time $t = 1, \cdots, T_m$ is s_t;
- The set of all possible states is $\mathcal{S}$;
- The action taken by the player at time $t = 1, \cdots, T_m$ is a_t;
- The set of all possible actions is $\mathcal{A}$;
- A reward R_m is given after each match;
- The set of all possible rewards is $\mathcal{R}$.

We assume the action space to be $\mathcal{A} = \{\text{Hit}, \text{Stick}\}$ and the state space $\mathcal{S} = \{2, 3, 4, 5, 6, ..., 21\}$, i.e. all possible sums of cards that players can have throughout the game, assuming that ace, for simplicity, counts as 1 and there are enough decks to keep the distribution of cards uniform during the whole match. Then, for the learning phase we establish an environment in which the dealer one player are sitting at the game table, the former following a predefined policy and the latter always choosing a random action. We simulate $M = 100000$ matches.

The overall workflow of this experiment, besides the validation task, is shown in Fig. 4.11. The next step after preparing reward data, in order to find the optimal policy, we need to formulate our task as a quadratic problem with binary variables. Moreover, since quantum annealers are able to work with Ising Hamiltonians, we must find a formulation of our problem that does not

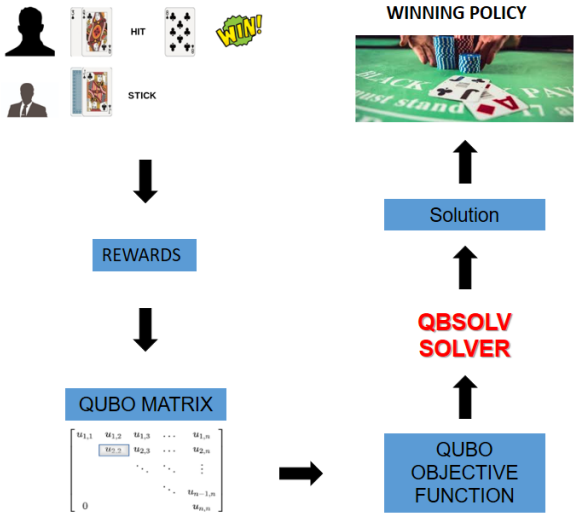


Figure 4.11: Workflow of the analysis, from data preparation to problem solution, before validation. Steps include simulating the matches, assigning rewards, formulating the problem as a QUBO model, feed it to the solver and retrieving the solution

allow for external constraints, i.e. we must transform it into a QUBO model (see ref. 2.9); to do so we rewrite the objective function of our problem by accounting for possible constraints.

Thanks to the relation seen in 2.9 between Ising Hamiltonians and quadratic functions of the form $\vec{x}^T Q \vec{x}$, $\vec{x}$ being the vector of binary variables, what we need to provide to the QBSolv or a quantum annealer is simply the matrix Q. In this work we exploit a high level Python's library provided by D-Wave that allows us to use a solver called *QBSolv*, which decomposes large problems into smaller instances as is needed when working with real quantum devices, and solves them with classical heuristics. The reason why we do not connect to the actual quantum device is that our problem is not large enough to motivate a thorough investigation of the QPU. As a consequence, the computational time required to obtain a solution in this context does not provide any information on the speed of quantum hardware, and is not reported in the following.

Going back to our task, we need to build a matrix Q that contains information on the rewards given by each pair (state, action), so that our player learns which action is optimal in as many situations as possible. Since we must rely on binary variables, we encode them as follows:

$$x_i = \begin{cases} 0 & \text{if i-th pair } (state, action)_i \text{ is not performed} \\ 1 & \text{if i-th pair } (state, action)_i \text{ is performed} \end{cases} .$$

The entries of the QUBO matrix reflect the importance of performing a cer-

tain action given the current sum of cards (diagonal), as well as the interaction between couples of (state,action) (off-diagonal), i.e. what reward, on average, can be obtained by ending up in the two states and performing the corresponding two actions in the same match. In principle, with this approach we could explore only a subset of said pairs, however, setting the number of simulations as high as 100000 we avoid this situation. As a consequence, a first formulation for the Q matrix is of the following form:

$$Q_1 = \begin{bmatrix} R_{(s,a)_1} & R_{(s,a)_1,(s,a)_2} & R_{(s,a)_1,(s,a)_3} & ... & R_{(s,a)_1,(s,a)_p} \\ 0 & R_{(s,a)_2} & R_{(s,a)_2,(s,a)_3} & ... & R_{(s,a)_2,(s,a)_p} \\ 0 & 0 & R_{(s,a)_3} & ... & R_{(s,a)_3,(s,a)_p} \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & R_{(s,a)_p} \end{bmatrix} \tag{4.3}$$

where (s, a) stands for (state,action), p is the total number of pairwise combinations of (state,action) and the lower-triangular part, since it is clear from this formulation that Q is symmetric, is set to 0 because it brings no additional information with respect to the upper-triangular one.

Because we are forced to use binary variables, we need to include in the model an additional term: in each state the player must perform only one of the two actions. This is a constraint that would not otherwise be taken into account in Q_1, therefore solving the problem in such case would yield an ambiguous solution where, for example, we could have

$$x^*_{(state_1,action_1)} = x^*_{(state_1,action_2)} = 1$$

or

$$x^*_{(state_1,action_1)} = x^*_{(state_1,action_2)} = 0,$$

where the apex indicates that x^* is a solution to the problem. As a consequence we could have cases in which our algorithm suggests, for the same sum of cards, both hitting and sticking or neither of them.

We solve this problem by introducing the constraint

$$x_{(state_i,\text{'Hit'})} + x_{(state_i,\text{'Stick'})} = 1 \quad \forall i = 1, \cdots, N, \tag{4.4}$$

where N is the total number of (state,action) pairs. In order to include it in our model, we need to map it to the quadratic form

$$\begin{aligned} &\left(x_{(state_i,\text{'Hit'})} + x_{(state_i,\text{'Stick'})} - 1\right)^2 = \\ &= x^2_{(state_i,\text{'Hit'})} + x^2_{(state_i,\text{'Stick'})} + 1 + \\ &+2x_{(state_i,\text{'Hit'})}x_{(state_i,\text{'Stick'})} - 2x_{(state_i,\text{'Hit'})} - 2x_{(state_i,\text{'Stick'})}. \end{aligned}$$

Ignoring the constant, which would yield no effect in the optimisation, we summarise the contribute of this constraint for $state_i$ with the matrix of coefficients

$$\begin{bmatrix} 1-2 & 2 \\ 0 & 1-2 \end{bmatrix} = \begin{bmatrix} -1 & 2 \\ 0 & -1 \end{bmatrix}. \tag{4.5}$$

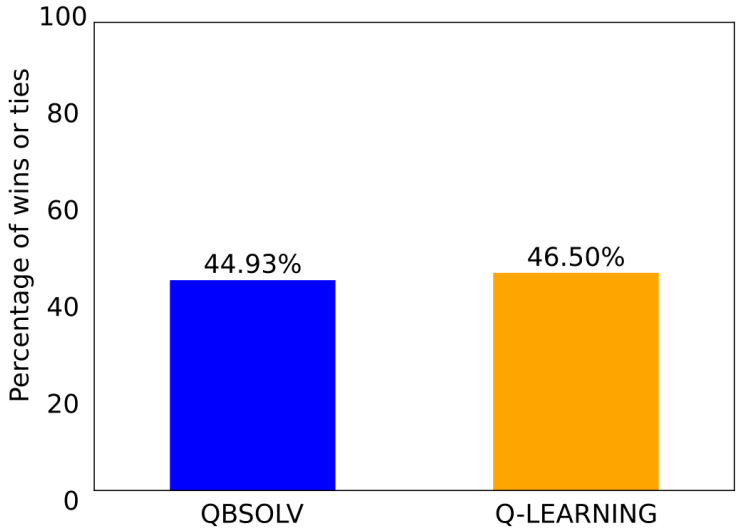


Figure 4.12: Percentages of wins or ties over total number of games $M = 10000$ given by QBSolv and state-of-the-art Q-Learning algorithm

Now we are able to write a square matrix Q_2 containing the contributions of constraint 4.4 for all states $i = 1, \cdots, N$, each of which takes the form of 4.5.

The final formulation of the QUBO matrix is thus given by

$$Q = -Q_1 + \lambda Q_2,$$

where Q_1 has negative sign, indicating that we want to *maximise* the rewards, and $\lambda \in \mathbb{R}$ is a tunable coefficient related to the constraint. Specifically, setting low values for λ results in giving low importance to the constraint, and thus increasing the probability of finding an infeasible solution; on the other hand, too high values for λ prevent classical heuristics and quantum annealing from finding the optimal solution and instead settle for a local minimum.

To solve the problem it is simply needed to feed the QUBO matrix Q to QBSolv, which uses a Tabu Search heuristics to find a solution. In order to evaluate the performance, we set up a framework in which a dealer and two different players sit at the game table: the first one follows the same policy as before; one of the two players acts accordingly to a pre-defined arbitrary set of rules, i.e., in this work, always hit until game is finished; the last player is our intelligent agent which chooses their action based on QBSolv's solution.

We simulate $M = 10000$ games similarly to how is described in Algorithm 2 and record the number of times our agent wins or ties. Before presenting our results, we use a classical approach to solve the same problem, so that a

comparison between the two agents - the one taught by QBSolv and the other following a policy dictated by a classical algorithm - can be made.

OpenAI provides the Python's toolkit Gym [42] to develop reinforcement learning algorithms. We exploit this to build a state-of-the-art Q-learning strategy for learning to play blackjack as explained in Algorithm 2. Q-learning is a technique that consists of initialising a set of rewards, called *Q-table*, associated with (state,action) pairs; as the simulations are performed, such table is updated following the rule

$$Q^m((state, action)_i) = Q^{m-1}((state, action)_i)$$
$$+\eta \left(R_{(s,a)_i} + \gamma Q^*((state, action)_{i+1}) - Q^{m-1}((state, action)_i) \right),$$

where m indicates the number of simulation, η is a learning rate, γ is a discount rate and $Q^*((state, action)_{i+1}) = \max_{a \in \mathcal{A}} Q^{m-1}((state_i, a))$. Refer to [43] for further investigation on rates, choices of Q-table initialisation and more on Q-learning.

Having trained the second agent with a classical approach, we are able to make a comparison on performances. We let each trained player alone in a series of games versus the dealer and a dummy player who always hits. Fig. 4.12 shows the percentage of wins and ties achieved by each intelligent agent. They do not perform much differently one from another: even though there is no clear sign of a quantum advantage, this result opens up the possibility of QUBO model being a good formulation of the problem, hence makes it worth investigating this approach further for more complex problems when proper quantum computers become available.

4.4 Conclusions

The application of quantum computing to machine learning tasks is still an open point in ongoing research. With the quantum version of Support Vector Machines we have investigated what benefits gate-model computers may bring to binary classification problems, presenting a case study on a self-constructed and self-labeled dataset of images. We have seen that the limited number of available qubits in modern devices poses a great restriction to the dimensionality of input data, but at the same time we observed that mapping input features to a quantum state space, and thus building a quantum-enhanced kernel, is a technique capable of yielding very complex decision boundaries. Finally, we stressed that the choice for an ansatz and the map to quantum states is central in the study of the classifier's performances, and it is not clear yet which strategies work best and under what circumstances.

In the second part of this chapter we posed a reinforcement learning problem: finding a strategy to win at blackjack. We simulated the games and reformulated our task as a QUBO model, which was then solved via the D-Wave's classical tool QBSolv. We found that a solution given by a quantum-inspired formulation of the problem yielded performances very similar to the state-of-the-art Q-learning technique.

In the next chapter we will focus on the application of quantum computing to optimisation tasks. We will analyse two case studies: the Number Partitioning Problem, which will allow a thorough investigation of the capabilities of modern quantum annealers, and the Limited-Assets Markowitz Portfolio Optimisation problem, a common task in the finance field for which we provide a QUBO formulation and the relative performance analysis.

Chapter 5

Quantum optimisation

One of the main fields of application for quantum computing is optimisation. A number of heuristics and meta-heuristics have been developed to solve large-scale problems, but when complexity increases there is no known algorithm able to always outperform the others. The choice of an optimisation strategy strongly depends on the nature of the problem at hand and on the parameters related to such strategy. Typically, we cannot rely on exact methods, which are able to find the global minimum of a problem; instead, we usually exploit algorithms that are likely to output a suboptimal solution. This is due to the fact that, when the problem size increases along with its complexity, it is impossible to explore the whole solution space or run algorithms that ensure a global optimum in a reasonable amount of time; exact methods require tons of operations and the number scales too quickly as the problem complexity increases.

Quantum computing offers alternative heuristics which sometimes may lead to solutions of the same level of suboptimality as those yielded by classical algorithms, but with an expected speedup, or even better solutions. The reason is that quantum optimisation comes with brand new algorithms which, by exploiting the properties of quantum-mechanical systems, offer approaches to complex problems that would be hardly implementable on classical computers.

Of course, as for all considerations about quantum computing in the NISQ era, it is not sure yet that quantum computers will be able to outperform classical ones; however, results obtained with noisy QPU's seem promising and it is worth investigating the capabilities of quantum-related heuristics.

In order to understand where quantum computing fits and what tasks it is interesting to test it on, we shall briefly summarise the classes of optimisation problems, qualitatively represented in Fig. 5.1. They are divided by of computational complexity; even though both a space (in memory) and time analysis is usually conducted, we hereby focus only on the latter as most of the times it is the prevailing issue.

Time computational complexity refers to the number of operations that an algorithm requires as a function of the problem, sometimes called input, size. The simplest optimisation models fall in the category of P, which stands for

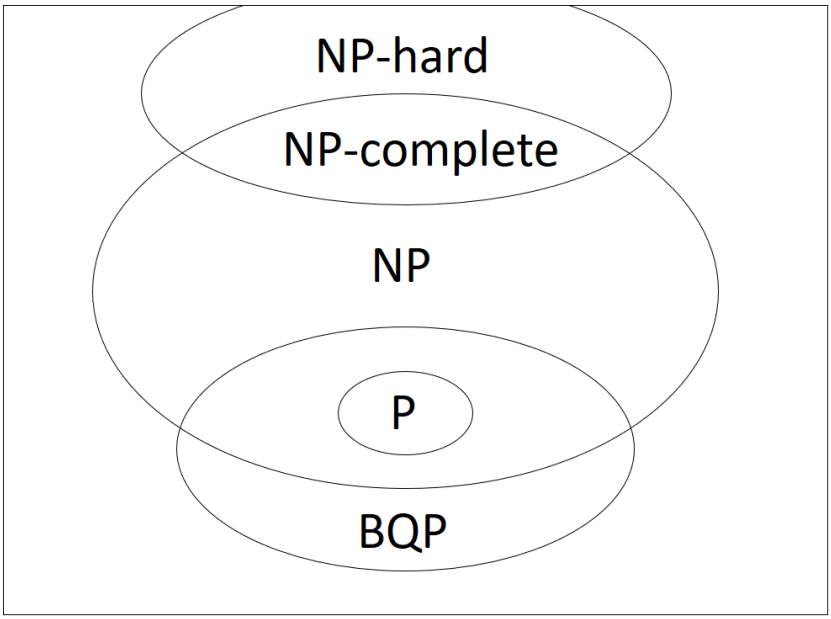


Figure 5.1: Venn's diagram of optimisation problems classes of complexity

polynomial time. Problems of this class can be solved in the worst case scenario by an algorithm that require a number of operations which is polynomial in the input size. For example, if we wanted to find the highest value in an array of n integers, we could read each element and finally output the result: this approach requires n operations; similarly, for nxn matrices, this method would require n^2 computations. The time complexity is thus polynomial in the input size.

A wider range of problems is NP: for these ones, if we know the answer, or solution, to a problem, there exists an algorithm that verifies it in polynomial time. The question of whether $P = NP$ is still an open issue; the definition of computational complexity is relative to the existence of proper algorithms, thus if it will be possible to prove that there always exists a way to check in polynomial time an answer to NP problems, the relation $P = NP$ will hold.

A subset of NP problems are $NP - complete$: this is a class of questions for which, natively, it might not be possible to check an answer in polynomial time; however, they can be reduced to problems for which this statement holds true.

Harder problems are those represented as $NP - Hard$, for which we know that it is neither possible to check an answer in polynomial time nor reduce them to NP models. Take for example purposes the Max-Cut problem: it essentially consists of finding two subsets S_1, S_2 of a bigger set S such that the total conflicts, i.e. penalties relative to each element belonging to the same set as others, between S_1 and S_2 is minimal. The $NP-complete$ formulation of this would be the question of whether there exist two such subsets; the $NP - hard$

version would be the quest of actually finding S_1 and S_2.

Finally, BQP stands for *bounded-error quantum polynomial time*: it is the set of decision problems, i.e. those that can be phrased as a yes/no question, that can be solved by an algorithm running on an error-corrected quantum computer which is able to solve the problem with high probability, usually considered as at least 2/3.

In quantum computing research, a lot of attention has been drawn to BQP problems for obvious reasons. However, in order to solve complex problems and provide proper alternative to current classical heuristics, and keeping in mind that we are still in the NISQ era, a lot of studies, as well as this work, have been focusing on solving $NP - hard$ problems via a hybrid quantum-classical approach. The quest is thus to check the performance and scalability of current hybrid algorithms and quantum hardware, so that even problems whose complexity scales quicker than polynomially, for example exponentially, in the input size can, or will, be tackled efficiently.

A number of techniques can be developed to use the quantum computing formalism for solving optimisation problems. Gate-based computers usually require to find problem-specific formulations and ansatz, which brings the benefit of universality of computations, meaning that in principle they can be used to solve any kind of task, but also have the downside of requiring complex investigations. Moreover, the number of qubits provided by modern QPU's are so low that no large-scale problem can be solved through universal quantum computers.

Quantum annealers offer a much higher number of qubits than their quantum counterpart. This comes at the cost of not being able to control them slavishly, which also makes annealers more suitable at optimisation tasks. Indeed, they can be used for supervised learning problems, as we have studied in section 4.3, and other fields, but the choice of problems typical of operations research seems more natural. Section 2.9 introduced the limitation of quantum annealers to be able to solve only Quadratic (Unconstrained) Binary Optimisation (QUBO) problems and section 4.3 showed an example of constrained model which, with the suitable reformulation consisting in including constraints in the objective function, was solved with a quantum annealing approach. This workaround allows to enlarge the set of problems that can be tackled by annealers, which then covers a wide range of optimisation tasks.

In this chapter we are going to investigate how complex real-world models can be formulated as QUBO and what are the capabilities of real quantum devices provided by D-Wave. We are going to study a well-known optimisation problem, the Number Partitioning Problem, and motivate this choice by explaining why it pushes annealers to the limit of their capabilities; then, we will analyse the QUBO formalism for a financial application in portfolio optimisation and compare the results with classical exact methods for small instances of the problem.

5.1 A case study on the Number Partitioning Problem

We have seen in section 2.10 that D-Wave's annealers support a structure known as Chimera graph (ref. Fig. 2.11); logical qubits are encoded as groups of physical qubits, which are represented as nodes of the graph, and as a consequence put up with the same connectivity. In terms of optimisation models, variables are represented as logical qubits, hence they must satisfy connectivity limitations given by the Chimera graph. In order to solve large-scale problems that do not naturally fit the annealer's structure, a problem decomposition into smaller instances is needed, as well as a routine to merge partial solutions together.

In this section we discuss which techniques can be used to overcome these issues, investigate the accuracy and the capability of the D-Wave 2000Q quantum annealer to solve problems with a significantly large input. To perform this study, we use one well-known NP-Hard model: the number partitioning problem (NPP) [44]. Thanks to the simplicity of this problem, it is easy to generate artificial instances of any size for which the optimal solution is known. Consequently, the measurement of the quality of the solution provided by the quantum annealer, along with the classical implementation of the tabu-search algorithm for a problem decomposition, will be possible even for large datasets. The results of this case study have been published in [19]. For this work, we acknowledge the support of the Universities Space Research Association, Quantum AI Lab Research Opportunity Program. Also, we thank Davide Venturelli for fruitful discussions.

The number partitioning problem (NPP) is defined as the task of discriminating if a given set S of positive integer numbers can be divided (partitioned) into two subsets S_1 and S_2 where the total sum of the elements in S_1 equals the total sum of the elements in S_2. Although the NPP is an NP-complete problem, the optimisation version is considered NP-Hard and can be formulated in the following way: given a list of N positive integers $\{a_1, a_2, ..., a_N\}$, the solution consists in finding a subset $A \subset \{a_1, a_2, ..., a_N\}$ such that the difference:

$$D(A) = \left| \sum_{i \in A} a_i - \sum_{i \notin A} a_i \right|, \tag{5.1}$$

is minimised. Throughout this work, we will refer to this difference as the *delta* between the two subsets A and $S \setminus A$. This problem is of both practical and theoretical importance: possible real applications span from multiprocessor pipeline scheduling [45], where balancing and partitioning different resources can be crucial, to cryptography [46] and all those problems requiring load balancing for I/O capacities, e.g. during databases processing [47].

Given a physical system composed of qubits, it is possible to define its Hamiltonian and initialise it in such a way that the lowest-energy state corresponds to all qubits being in a superposition state of 0 and 1. Then, as the annealing proceeds, the problem Hamiltonian deriving from the problem's specifications is introduced, as described in section 2.9. We recall that D-Wave's quantum

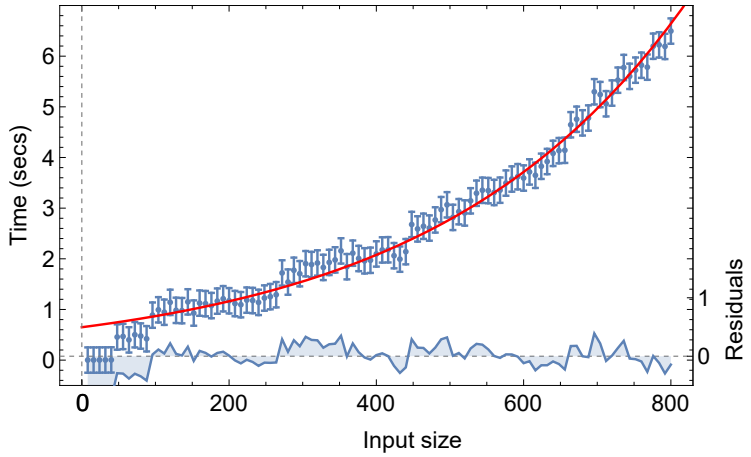


Figure 5.2: Execution time of tabu-search for increasing input size. Classical partitioning of a set with the classical embedded tabu-search as backend. The red line is the exponential fitting $t = Ae^{x/B}$ where B=340 a.u. and x is the size of the input. Blue points represent measured data. The blue line in the bottom part is the deviation of the experimental point from the value of the fitting

annealer is able to solve problems expressed in the form of an Ising glass, with a Hamiltonian written in the form 2.4.

A complete formulation of the NPP as Ising spin glass has been provided in Ref. [48]. The Hamiltonian for this type of problem can be defined by assuming an increase in the energy when the total of amplitudes associated with positive spin states is different from that of amplitudes with negative spins. According to this formulation, it is possible to use the following relation:

$$H = \left(\sum_{i \in N} a_i S_i \right)^2 \tag{5.2}$$

with $S_i = \pm 1$ the spin values indicating the subset to which the i-th element belongs and a_i the element of the set A. It follows that if the ground state has $H > 0$ there is no exact solution of the specific problem and the ground state is the one minimising the mismatch between the two subsets.

In order to formulate the problem as a QUBO model, we first have to convert our $S_i = \pm 1$ into binary variables of the form $q_i \in \{0, 1\}$ (see relation 2.5). Now, the original Ising problem can be mapped into the QUBO form:

$$\min \sum_{i,j} Q_{ij} x_i x_j \tag{5.3}$$

where x represents a binary variable and Q is the so-called QUBO matrix containing the weights of qubits (h_i in Eq.2.4) in the diagonal and the couplers

coefficients (c_{ij} in Eq.2.4) in the (i, j) elements. Similarly to what we discussed in section 4.3, this matrix will be symmetric ($c_i c_j = c_j c_i$).

Having the QUBO matrix, it is possible to submit it to the QPU and retrieve a solution of the optimisation problem. However, the connectivity between qubits required by the NPP is that of a complete graph, which is yet to be supported by any modern quantum annealer that provides a fairly high number of qubits. To overcome this and similar problems, the D-Wave device operates a minor-embedding of the problem onto its Chimera architecture. Specifically, one can either run the built-in tabu-search heuristics provided by the D-Wave Hybrid tool to optimally decompose the problem into subproblems, as we do in this work, or choose a custom minor-embedding strategy. The subproblems will then be mapped onto the Chimera graph, for which the QPU will start the quantum annealing.

In Fig. 5.2 the time required to solve the NPP on classical hardware by using the D-Wave Qbsolv is reported as a function of the input set size. The elaboration time increases exponentially while a structured procedure is applied in order to find the minimum: a number of subproblems are generated, handled and finally merged into a global solution of the NPP. The exponential increase in the execution time confirms the NP-Hardness of the problem when approached with classical hardware and formulations. When the problem is submitted to the QPU, the execution time changes and paves the way for a wide range of investigations of the D-Wave Hybrid tool. Moreover, this peculiar model allows us to study what happens in one of the worst case scenarios from the perspective of the qubits connectivity: a fully connected graph, where the number of couplers and weights precision play a central role [49, 5].

In order to investigate the capabilities of the D-Wave hybrid tool, we solve multiple NPP examples of increasing size. For each fixed problem size we use 10 different datasets and collect statistics of the results. For experimental purposes, we choose the data in such a way that the ground state of the corresponding Ising models is $H = 0$, i.e. there is a single partition of the set of numbers.

For our studies, we first construct the QUBO matrix for each problem, and then we define the tabu-search heuristics as the algorithm that splits the original problem into the subproblems, preparing them to be embedded on the Chimera graph.

Fig. 5.3a shows the QUBO matrix defining the connectivity of qubits required by the specific NPP instance and with regular patterns related to the number amplitudes in the dataset. With the problem being formulated as an Ising model, all variables are coupled in pairs, resulting in a dense (upper-triangular) QUBO matrix. Such connectivity is the most complex to handle and can thus be an issue for current quantum hardwares, making it interesting to investigate the quantum annealer performance.

The distribution of partition deltas for each different problem size is summarised in Fig. 5.3b. We produced 10 different datasets to be partitioned for every problem size and we computed the value of delta for all these instances. For each problem size we have built a boxplot of deltas coming from the solution of the NPP.

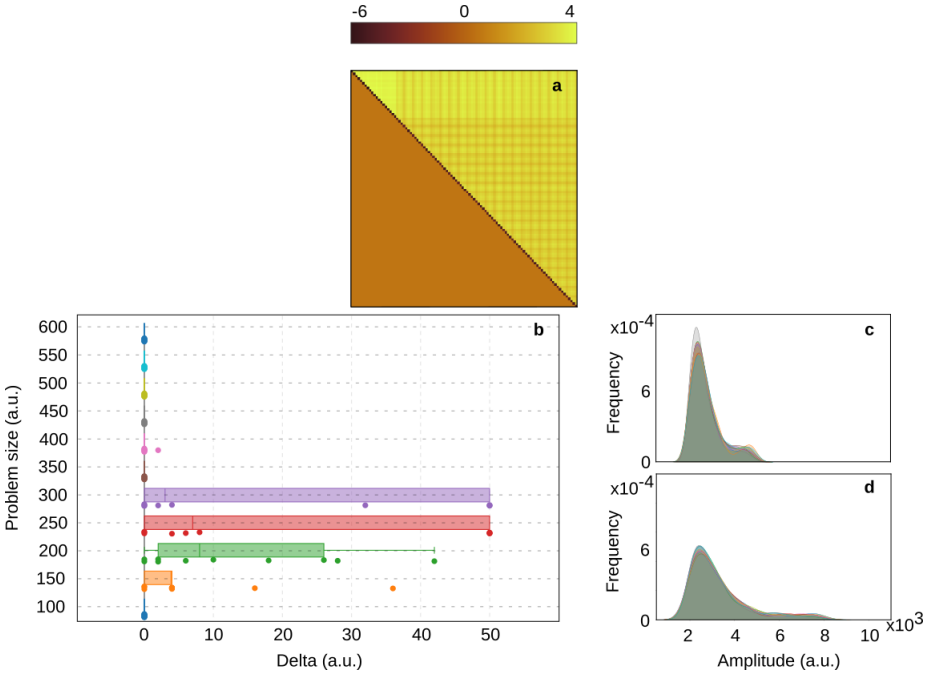


Figure 5.3: QUBO matrix and delta distributions over multiple datasets. **a.** QUBO matrix of one instance of data with problem size equal to 100. The entries are scaled and the intensity of colors is used as a means to summarize the main characteristics of the plot: the diagonal is made up of negative values, the lower triangular part is 0 and the upper one has no null entries. **b.** Boxplots of deltas for different input problem sizes computed over 10 datasets for each size with dots representing the values of the delta in each instance. These values were saturated to 50, therefore such numerical value is to be interpreted as the result of a bad solution. **c-d.** Kernel density estimation of the distribution of input data for problems with, respectively, 200 and 500 variables, showing the data from all 10 instances in each plot

The combination of quantum annealing with the classical minor-embedding heuristics is able to find the optimal solution in most cases. This is achieved especially when the problem is very small (and, as a consequence, computationally easy) or when its size is significantly higher. In fact, for our smallest problem and for those with input size greater than 450 binary variables, we are able to optimally solve the 10 different NPP instances. On the other hand, for middle-sized problems, not all distributions of data allow qubits to reach the ground state. As a result, we obtain the optimal solutions only for a subset of the given problems.

Figs. 5.3c-d report the density distribution of each of the 10 datasets used for 2 different problem sizes (200 and 500 variables). As explained above, the

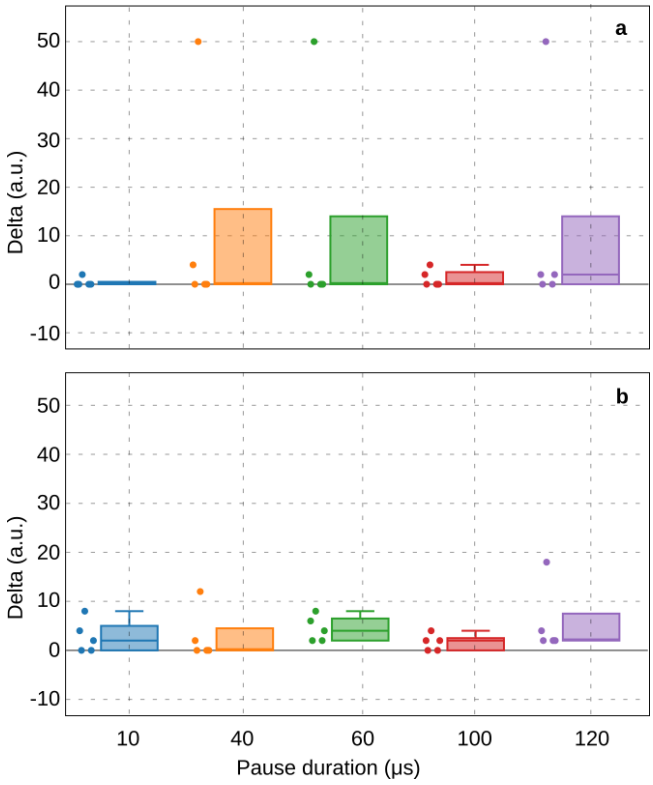


Figure 5.4: Annealing cycle and boxplots of deltas with pause. Boxplots of deltas found over multiple runs of the annealing with the same pause starting point, same value of persistent current but different pause duration times. The dots represent the values of (eventually saturated) deltas for every run. **a** shows boxplots for one of the problems with 300 variables yielding a very bad solution, while **b** shows the same for one of the problem with 200 variables

quality of the results on the bigger model exceeds the one on intermediate sizes. Comparing both density distributions we can conclude that this behavior is fundamentally related to the fact that a shift of the distribution curve to lower values leads to a dataset containing more solution degeneracy for lower energy states and consequently simpler to solve even when the problem is bigger.

An effective method to enhance the exploration of the solution space is the direct manipulation of the annealing schedule, as we have anticipated in section 2.9. This distinctive technique can be used to improve the quality of the solution in the cases described before in which we could not reach the ground state. Indeed, in contrast to what we did with the first approach, where the annealing has been used without interfering with the spontaneous process, we exploit now the capability of the D-Wave solver API to manipulate directly the scheduling of the cycle. To accomplish this, we define the time instant at which the cycle

has to be stopped and resumed, as well as the value of the persistent current powering the adiabatic relaxation. This procedure is the annealing pause (see section 2.9).

Fig. 5.4 shows the results of the analysis on two problems, one of size 200 and the other of size 300, for which the uncontrolled annealing performed worst. For each problem we have paused the annealing after 10 μs, let the system rest for 10, 40, 60, 100 and 120 μs respectively halfway through the flow of current and finally let the annealing end. This whole process was repeated 5 times for each problem.

The best energy configuration in terms of distance from the ground state for the two problems analysed here were not achieved with the same parameters settings. In fact, every instance requires different values of the pause starting point, duration and persistent current. Nevertheless, all of our choices greatly improved the results previously obtained with the uncontrolled annealing, even though not all of them led to the optimal solution. We were able to record considerable results multiple times, proving that the introduction of the pause can increase the accuracy of the annealing.

This improvement in the quality of the results is due to the effect of the pause on the search region of the solution space: by pausing the flow of the persistent current, and hence the annealing, we are able to widen the exploitation of the energy landscape and, as a consequence, the probability of finding the global minimum, as discussed in section 2.9.

In conclusion, we have investigated the behaviour of the quantum annealer on a level of complexity which is potentially that of real-life problems. One interesting result was found: a discontinuous accuracy with the problem size. While high-quality results were found at small problems, there is a counter-intuitive behaviour as the problem dimension increases: a dip in the accuracy for medium-sized problems and a recovery as size increases. This effect was explained by the value distribution within the dataset: lower values in the input allow higher accuracy of the result, even when the size of the problem is rising.

The medium-sized problems were studied in more detail by applying pauses during the annealing cycle, allowing the system to explore the solution space with a modified equilibrium. Our results prove that with the correct parameters tuning it is possible to improve dramatically the accuracy of the solution, obtaining optimal results in cases that had proven to be troublesome in a non-altered context.

5.2 A case study on a Limited-Assets Markowitz Porfolio Optimisation problem

In this section we present a case study on how to implement a common optimisation problem deriving from the financial field in a hybrid computing framework: the Markowitz Mean-Variance Portfolio Optimisation problem with limited assets, or *cardinality constrained*. We are going to use the QBSolv solver by D-

Wave in order to simulate the problem decomposition and quantum annealing as if it was run on the QPU; computational time will not be recorded, whilst a thorough study of the solution quality will be performed. This will be done by solving the same problem via a CPLEX exact solver and therefore knowing the global minimum for two instances of the problem. In order to do so, we restrict our cases to feature a relatively small amount of variables (in the order of $1 * 10^3$ and $1.2 * 10^3$) and highlight which are the upsides and downsides of a quantum annealing heuristics with respect to an exact method.

The Markowitz Mean-Variance Portfolio optimisation problem is the task of finding the amount of investment to assign to each of n available assets, knowing their means μ and covariances Σ, in order to minimise the total variability of such investment while assuring a threshold level of return. As the name points out, the model's variability is expressed as the total variance of investments; this need not be necessarily the case, as alternative problems feature different measures of risk. Namely, we need to solve the following:

$$\begin{aligned} \min_{\vec{x}} \quad & \vec{x}^T \Sigma \vec{x} = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{i,j} x_i x_j \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mu_i x_i = \rho \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \quad \forall i = 1, \cdots, n \end{aligned} \tag{5.4}$$

where index i indicates the asset, x_i is the fraction of investment on the corresponding asset, Σ is the variance-covariance matrix of the assets and ρ is the threshold return that we ask to cover. From this formulation stems that the maximum value for ρ is equal to the biggest mean return of the assets, which is obtained when investing the whole available amount in the corresponding asset, hence setting up a higher ρ would inevitably result in an empty solution space for problem 5.4.

From the objective function and the first constraint in 5.4, we can see that the choice for ρ and the analysis of solution quality must be performed by taking into account an important trade-off: asking for high values of ρ ensure bigger overall returns, but will likely increase the amount of variance (risk); adopting a conservative policy and minimising risks result in low returns. A common approach to mean-variance portfolio optimisation consists of analysing a number of results for different values of ρ, so that one can find the value that best satisfies the mean-variance trade-off. In this work we focus on formulating the problem in a QUBO model and compare solutions using some benchmarks, hence we simplify the analysis by fixing a single value for ρ.

The limited-assets markowitz (LAM) problem consists of increasing the complexity of 5.4 by setting some bounds both on the number of assets chosen amongst all the available and on the amount of investment on each asset. Specif-

ically, if an asset i is taken, then the amount x_i invested on it must lie in some pre-defined interval; moreover, the number of chosen assets cannot exceed some bound $K < n$. The LAM problem reads:

$$\begin{aligned}
\min_{\vec{x}} \quad & \vec{x}^T \Sigma \vec{x} = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{i,j} x_i x_j \\
\text{s.t.} \quad & \sum_{i=1}^{n} \mu_i x_i = \rho \\
& \sum_{i=1}^{n} x_i = 1 \\
& x_i \leq M_i y_i \quad \forall i = 1, \cdots, n \\
& \sum_{i=1}^{n} y_i \leq K \\
& y_i \in \{0, 1\} \quad \forall i = 1, \cdots, n \\
& x_i \in \{0\} \cup [l_i, u_i] \quad \forall i = 1, \cdots, n
\end{aligned} \tag{5.5}$$

where M_i's are the so-called *big-M*'s, which can easily be quantified as values greater than or equal to 1, since clearly $l_i > 0$ and $u_i \leq 1$; K is the maximum number of assets to invest on; $y_i \ \forall i = 1, \cdots, n$ are supplementary binary variables that take value 1 if the corresponding x_i is strictly positive, so that counting the assets chosen by a solution boils down to summing over all such binary variables.

This problem has drawn attention to research because of its much greater complexity with respect to the simple mean-variance problem 5.4. As a matter of fact, the LAM model falls into the category of NP-hard problems [50], especially because of the constraint on the cardinality of assets chosen. This limitation poses some great challenges to real-world applications, thus motivating the need for some efficient heuristics able to scale up with the problem size.

For this experiment we partly follow the approach of [51] and take two open datasets, available in Beasley's OR Library [52], of anonymised assets for which returns and pairwise correlations are provided. The first dataset consists of 31 assets, while the second one is made up of 85. From a classical optimisation point of view, it is possible to formulate 5.5 as a Mixed Integer Quadratic Program with as many variables as twice the number of assets [50], therefore with these data exact methods are still able to quickly solve the problem. However, in order to formulate it as a QUBO, the number of variables increases further, making these instances an interesting playground to test quantum annealing.

The first step to construct our QUBO matrix consists of formulating 5.5 via binary variables. In order to do so, we discretise the set $\{0\} \cup [l_i, u_i]$: the more discretisation points we set, the more faithful our formulation will be to the original problem and thus the closer our solution will be to the global minimum. As a consequence, we merge the cardinality constraints into one that requires only the number of variables for which it is chosen a discretisation point greater

than 0 to be at most equal to K. Then, we must add the additional condition of choosing exactly one discretisation value for each variable.

In this set-up we also need to reformulate the objective function and the two remaining constraints so that they account for the discretisation. Specifically, for every time we consider variable x_i for the $i-th$ asset in the continuous variable model, we now have to consider a relation of the form $x_i \rightarrow d_m z_i^m$ where z_i^m is the binary variable associated to the $m-th$ discretisation point for asset i and d_m is the value of discretisation.

We introduced a slight simplification of the model by allowing for the threshold return constraint to consider the overall return feasible if it is least equal to $4/5-th$ of the actual value of ρ chosen; this way we ease the search for a solution whose feasibility criteria are close to the ones in the original problem, hence reducing the drawback of using a discretisation technique. We expect to find solutions that do hardly satisfy the equality constraint with a threshold equal to ρ; instead, since they aim at minimising the overall variance, and because of the trade-off discussed previously between variance and return, they will likely find the minimum feasible value of expected return.

Finally, before including all constraints into the objective function, we turn the cardinality inequality constraint into an equality one by adding slack variables. Following model 5.5, only one slack ζ is needed, however, it should be an integer variable ranging from 0 to K, where $\zeta = 0$ indicates that we are taking an amount of assets exactly equal to K, while $\zeta = K$ means that we are investing on no asset. The latter situation should never be verified as it would mean that our solution does not satisfy the constraint about a threshold level of return, nevertheless we consider K a proper upper bound. The reformulation as an equality constraint cannot be avoided: in order to be consistent with the QUBO formulation, we must have only quadratic terms in the objective function, meaning that when we include constraints in it, if not already quadratic, they must be squared, which may lead the annealing to find a good - but infeasible - solution that takes more than K assets. Furthermore, by squaring linear terms, we contribute to the off-diagonal entries of the QUBO matrix, hence exploit the correlation and connectivity, i.e. entanglement, between qubits.

In order to include ζ in our model, we need to discretise it, too. In this case the discretisation is trivial and it boils down to adding $K+1$ binary variables, each one related to one of the integers from 0 to K.

With a little change of notation, the final model takes the following form:

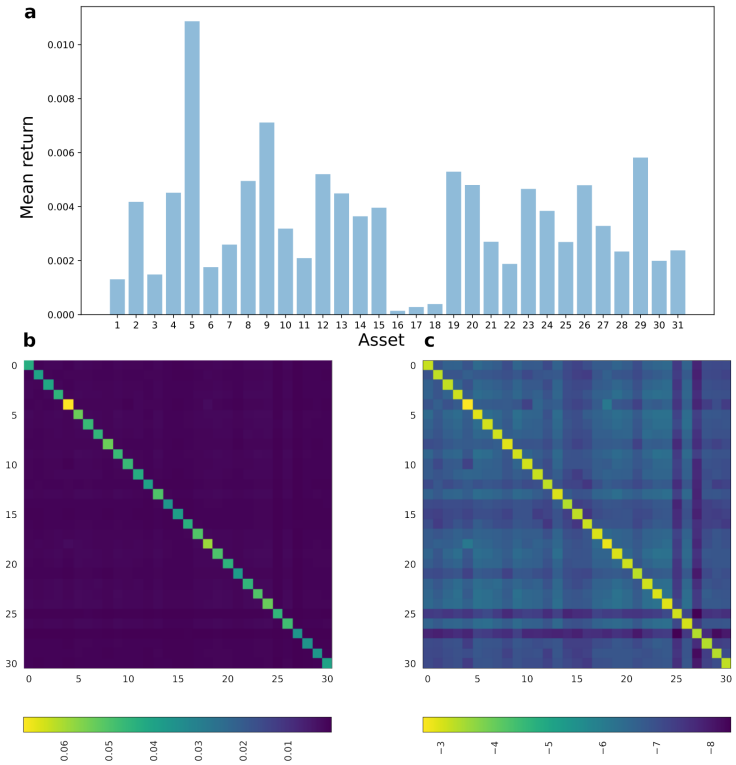


Figure 5.5: Data for first instance of the LAM model. The mean return of 31 assets is shown in panel (a). The covariance matrix in linear and logarithmic scale is shown, respectively, in panels (b-c): assets are not much correlated one with another, as (b) highlights; however, covariances are not exactly equal to 0, as shown in (c) by the log scale

$$
\begin{aligned}
\min \quad & \sum_{m=1}^{M}\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{i,j}d_{i,m}x_{i,m}d_{j,m}x_{j,m} \\
\text{s.t.} \quad & \sum_{m=1}^{M}\sum_{i=1}^{n}\mu_i d_{i,m}x_{i,m} = \rho \\
& \sum_{m=1}^{M}\sum_{i=1}^{n} d_{i,m}x_{i,m} = 1 \\
& \sum_{m=1}^{M} x_{i,m} = 1 \quad \forall i = 1,\cdots,n \\
& \sum_{m=2}^{M} x_{i,m} + \sum_{k=0}^{K} k y_k = K \\
& \sum_{k=0}^{K} y_k = 1 \\
& x_{i,m} = \{0,1\} \quad \forall i = 1,\cdots,n \quad \forall m = 1,\cdots,M \\
& y_k \in \{0,1\} \quad \forall k = 0,\cdots,K
\end{aligned}
\tag{5.6}
$$

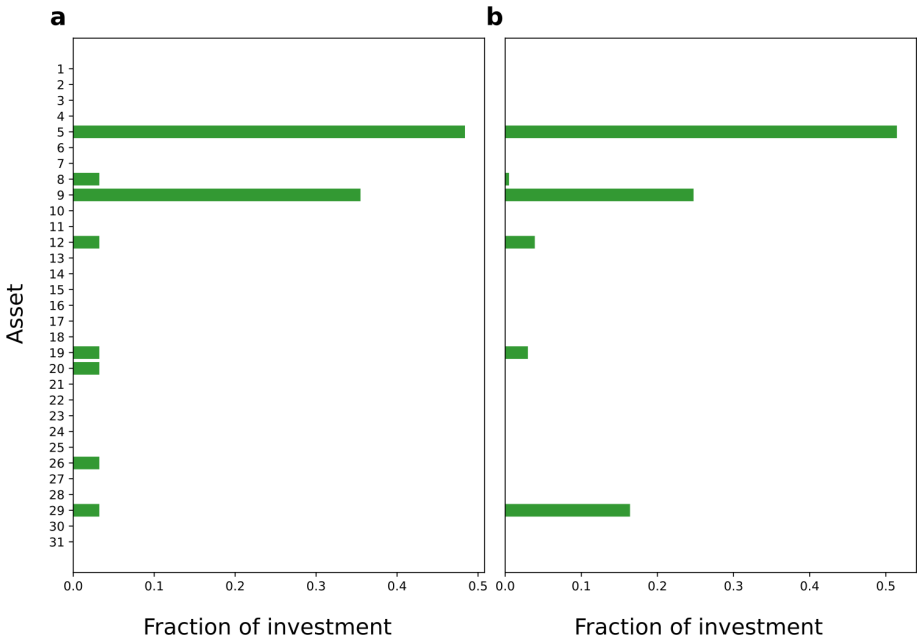


Figure 5.6: Distribution of investments by QBSolv solver (panel a) and CPLEX (panel b) for the first instance of the problem

where $d_{i,m} \in \{0\} \cup [l_i, u_i]$ with $d_{i,1} = 0 \ \forall i = 1, \cdots, n$, $x_{i,m}$ denotes the binary variable related to asset i and discretisation point m and y_k is related to the slack variable for the $k-th$ integer.

We shall analyse the two instances of the problem separately. The first one comprises 31 assets whose means $\vec{\mu}$ are shown in Fig. 5.5 (a). We compute the covariance matrix with the well-known relation

$$Cov(X, Y) = \text{corr}_{X,Y} \cdot \sigma_X \cdot \sigma_Y,$$

where X, Y are assets and σ_X, σ_Y their standard deviation provided in [52] along with their correlation $\text{corr}_{X,Y}$, and show it in Fig. 5.5 (b-c).

We choose the values $\rho = \max \vec{\mu}$, $l_i = \min_{m=1,\cdots,M} d_m$ and $u_i = \max_{m=1,\cdots,M} d_m$ $\forall i = 1, \cdots, n$, where $d_{i,m} = d_m \ \forall i = 1, \cdots, n \ \forall m = 1, \cdots, M$. We set the number of discretisation points M in such a way that the total number of binary variables are $\sim 1000 + K$ with $K = 10$, which is one of the common choices for this kind of problem [51]. In this instance, such discretisation approximates the interval $[0, 1]$ with steps of length ~ 0.03.

We use the QBSolv tool to solve the problem. After a tuning analysis of the coefficients associated to each constraint in the QUBO matrix, we obtain results shown in Fig. 5.6(a). Out of $n = 31$ assets, with cardinality thresholds set on $K = 10$, we select 8. The best solution we find through QBSolv yields

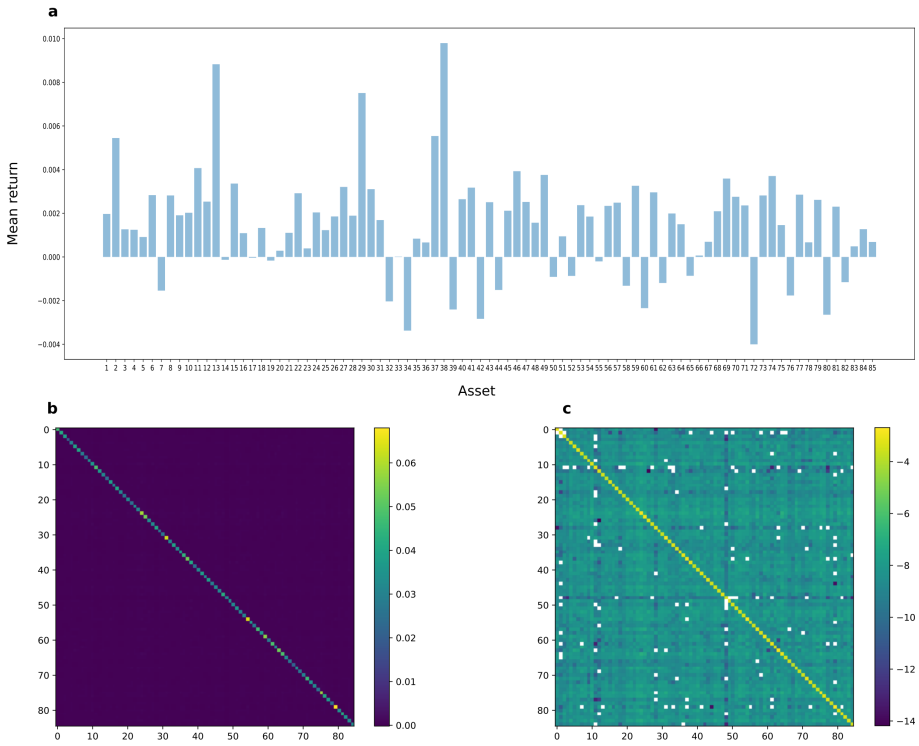


Figure 5.7: Data for the second instance of the LAM model, showing both mean returns of 85 assets (panel a) and their covariance matrix, in linear (panel b) and logarithmic scale (panel c). The main difference in means from the first instance is that now some assets are not fruitful (negative mean). White dots in (c) represent highlight negative covariances. Once again assets are quite uncorrelated

a total amount of investment equal to approximately 1.03, which violates the equality constraint that requires the total wealth invested to be at most 1, i.e. 100%. This is due to the choice of parameters we tuned: in order to find a solution that solves best our problem in terms of risk minimisation, we allow for a little exceed in the investment. Clearly this solution is infeasible, but a simple reschedule allows us to make it feasible and, at the same time, obtain a result that is very close to the global minimum. As a matter of fact, we show in Fig. 5.6(b) the output of CPLEX exact methods. We summarise numerical results in table 5.1.

We shall now analyse the second instance of the problem, made up of 85 assets, with means $\vec{\mu}$ shown in Fig. 5.7 (a) and covariance matrix, which is computed as the previous one, in Fig. 5.7 (b-c).

We follow the same reasoning as before to set the values for ρ, $d_{i,m}$ $\forall i = 1, \cdots, n$ $\forall m = 1, \cdots, M$ and K. On the other hand, since the number of assets is

Solver	$\boldsymbol{\rho}$	**Total variance**	**Return**	**Sum invested**	**# Assets**
QBSolv	0.0108	0.0240	0.0087	1.0322	8
CPLEX	0.0108	0.0233	0.0087	0.9999	6

Table 5.1: Table of numerical results by QBSolv and CPLEX for the first instance of the problem

Solver	$\boldsymbol{\rho}$	**Total variance**	**Return**	**Sum invested**	**# Assets**
QBSolv	0.0098	0.0099	0.0078	1.0769	8
CPLEX	0.0098	0.0091	0.0078	0.9999	9

Table 5.2: Table of numerical results by QBSolv and CPLEX for the second instance of the problem

almost three times the previous one but we cannot afford to have too large scale problem mapped to the QPU simulator, we choose the number of discretisation points M by increasing the total amount of variables to only $\sim 1200 + K$. As a result, the discretisation step's length is ~ 0.07.

We run again the QBSolv solver and show the results in Fig. 5.8(a). Similarly to the former case, we tune parameters in such a way that our solution is close to the global minimum in terms of variance minimisation. However, the concept of closeness becomes more and more hard to achieve as the length of discretisation steps increase. As for the previous scenario, a little workaround can make QUBO's solution feasible, at the cost of increasing the total variance. In Fig. 5.8(b) we show results obtained by CPLEX. We summarise numerical results in table 5.2.

In conclusion, we have run a quantum annealing simulator for two instances of increasing size of the limited-assets Markowitz portfolio optimisation problem. Even though the clear drawback of this approach lies in the necessity of implementing some sort of discretisation, which eventually leads to a dramatic increase in the number of variables, we were able to bound such number and still obtain a suboptimal solution quite close to the global minimum. Accessing the QPU might be a first step towards overcoming this issue, but the most impactful change would be given by increasing the number of qubits in the annealer and improving their connectivity.

We found interesting results in terms of solution quality related to scalability: while large-scale instances are intractable by exact methods, quantum annealing seems to be a heuristics that scales more efficiently and able to find good suboptimal, if not optimal, solutions. A thorough investigation is worth carrying on with larger and better-performing QPU's for more complex instances: the approach seen in this experiment might be a valid alternative to existing classical algorithms. Moreover, from a computational time viewpoint, even if quantum annealing might not find the global minimum, it could yield solutions of quality comparable to those given by classical algorithms more quickly. As a matter of fact, the fewer decomposition operations are required in a hybrid computing framework, the faster is the overall annealing.

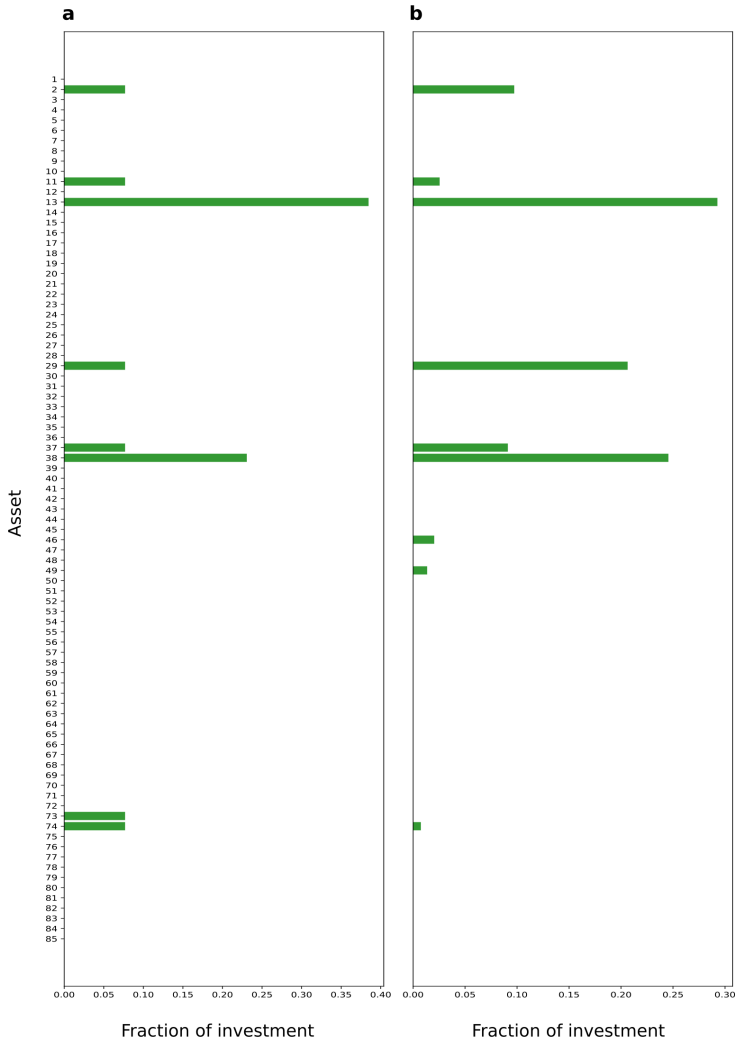


Figure 5.8: Distribution of investments by QBSolv solver (panel a) and CPLEX (panel b) for the second instance of the problem

5.3 Conclusions

In this chapter we examined the performances of a QUBO formulations of two NP-Hard problems. For the first one, the Number Partitioning Problem, we have accessed one of D-Wave's quantum annealers via cloud and thoroughly evaluated its capabilities. We have generated a number of instances for which the optimal solution was known, with growing input size in order to approach the complexity of real-world problems and obtained remarkable counterintuitive

results: for small and large scale instances we were able to find the global minimum with a high level of accuracy, while for middle-sized problems the annealer was not able to yield high quality solutions as often. In such cases, stopping the annealing cycle proved to be an effective strategy that allowed to ultimately reach the global minimum. The connectivity required by the NPP problem is the most complex to manage for quantum computers and thus allowed a detailed investigation of modern annealers' capabilities. This kind of study, under the described circumstances, was not performed before and the results that we obtained were published in [19].

In the second part of the chapter we proposed a QUBO formulation of the limited-assets Markowitz Portfolio optimisation problem. We analysed the performances on two small instances so that a comparison with classical exact methods was possible. On one hand, we found high quality solutions that did not differ much from global optima; on the other hand, the need for a discretisation of continuous variables lead to a significant increase of their number. This prevented us from adopting a thick grid and ultimately led to a slight decrease in solution quality. However, since the results obtained with the QUBO approach proved to depend on the choice of the discretisation grid, we can conclude that with progress in hardware development it might be possible to tackle larger-scale problems that, currently, pose great challenges to classical optimisation techniques.

Chapter 6

Conclusions

In the first part of this work we introduced the mathematical framework of quantum computing. We started describing what a qubit is and which properties make it more powerful than classical bits, motivating the search for a quantum speedup in scientific tasks such as molecule simulations, machine learning and optimisation. We therefore presented what a superposition state is and its relation with the probabilities of collapsing into one of the deterministic states; we discussed, from a mathematical point of view, the act of observing, i.e. measuring, a quantum-mechanical system composed by qubits; we studied the entanglement property, one of the key characteristics that can make quantum hardware much faster than their classical counterpart, and how measuring part of an entangled system inevitably leads the other to deterministically collapse on a certain state. Then, we presented quantum states of qubits as elements of the Bloch sphere, which is a geometrical representation at the basis of understanding how the first wide class of quantum hardware works: universal gate-model computers. Qubits state are rotated over the Bloch sphere, hence change their phase and amplitudes, via the application of gates, which are mathematically represented by unitary matrices. We described a few basic gates that are commonly used to reproduce any possible rotation on the sphere, hence allow to cover the whole space of (pure) quantum states.

Furthermore, we discussed some common ways to encode classical data in qubits states, highlighting the upsides and downsides of each method, both in term of the complexity required to find the proper sequence of gates and from the viewpoint of number of qubits needed for each approach.

We briefly studied the dynamics of quantum systems and introduced the concept of Hamiltonian operator. This is a central topic especially for, but not restricted to, the second class of quantum computers: quantum annealers or adiabatic quantum computers. Their alternative approach does not allow to have control over the qubits state at any time instant throughout the computation; rather, they set up the energy landscape of the quantum system and let it naturally reach ground state. We have seen that quantum annealers support a specific Hamiltonian function, the Ising Hamiltonian, and how from this fol-

lows that such computers, from an operations research viewpoint, are only able to solve quadratic unconstrained binary problems, preventing them from being universal computers.

To conclude the introduction, we discussed what modern obstacles are in quantum computing and which advancements may lead to competitive performance. We presented the concept of NISQ era, the limitations due to reduced number of qubits and the problem of noise, caused by interactions between the quantum system composing the hardware and environment, leading to short decoherence times. Finally, we investigated which are current technologies employed to build a system that obeys the rules of quantum mechanics, spanning from superconducting circuits cooled down at very low temperature, to real particles such as atoms and photons.

The scope of this work was to present a number of techniques to formulate data science problems using the quantum computing formalism, understand the limitations of current and near-term devices as well as the benefits that already emerge from quantum computation and finally analyse performances. This was done in the second part of the work, in which we first briefly described some well-known hybrid algorithms that are being currently used and then investigated four different important case studies, two in the field of machine learning and two optimisation tasks. The need for hybrid quantum-classical computation is strictly related to being in the NISQ era, in which current quantum hardware is not capable of solving large-scale complex tasks alone because of noise and the reduced number of qubits.

The first algorithm that we discussed, the Variational Quantum Eigensolver, is a technique that can be used both for optimisation and supervised learning tasks. It is based on the idea of parametrising a quantum circuit in such a way that an optimal set of gates is found according to some rule, such as the minimisation of a loss function. Similarly, the second routine, the Quantum Approximate Optimisation Algorithm, simulates the Hamiltonian of a system by applying a set of gates, each for a certain amount of time that has to be optimised according to some rule. Not only these two techniques represent the perfect synergy between quantum and classical hardware in solving complex tasks, but they also build the foundations for understanding common approaches to hybrid computing, such as those investigated through the data science case studies.

The first experiment that we conducted focused on the implementation of a Support Vector Machine algorithm for image classification, with a quantum-enhanced features space. The goal was to study the effect of a new kind of kernel, one that maps input data onto a quantum state space, on the decision boundaries of the Support Vector Machine classifier. In order to perform such analysis, we built and labeled a dataset composed by both sad and happy smiles, with a very limited number of pixels. The reason behind this choice was that we had access to a gate-model hardware simulator to test our algorithm, but it only provided a few qubits; moreover, in order to implement a quantum-enhanced kernel, we had to represent each feature of our dataset with a qubit. After applying the Principal Component Analysis dimensionality reduction technique

and a circuit describing the state evolution of qubits considered hard to simulate classically, we computed our kernel in the quantum state space and fed that information to a classical computer in charge of finding the parameters of our classifier. Such choice for the circuit was relevant to understand what might be the effect of new, classically intractable kernels; at the same time, working in a small-scale scenario has let us make all computations with a classical device. The results we obtained were interesting: no clear quantum advantage was shown, mostly because of all the restriction required to work with gate-model quantum hardware, as well as no incredibly high quality performances; however, the decision boundary obtained was very complex, paving the way for further investigation on different datasets and circuits ansatz in order to understand whether a quantum approach might be the key to correctly classify extremely complex datasets, where classical kernels may fail.

The second case study focused on modeling a reinforcement learning problem in a QUBO formalism, so that it could be solved with a quantum annealer. We simulated multiple times the game of Blackjack in a situation where an agent plays alone against the dealer. Both follow a very specific policy: the latter never sticks below 17, while the player always chooses a random action between hit and stick, in every situation. This allowed us to collect enough data on each stage of the game and record the corresponding reward, meaning if the player lost the game, won or there was a tie. Having constructed our dataset, we were able to formulate and solve the (quadratic) problem of finding what action to take in any stage of the game in order to maximise the expected reward. Writing such problem in the QUBO formalism has let us feed it to a quantum annealing simulator and retrieve results. We tested the newly-learned policy in a situation where our intelligent agent had to face another player and the dealer together; we did the same with an agent trained through the state of the art Q-learning algorithm in order to be able to compare performances. Even though there was no interest in recording the computational time on a simulator, we did register a measure of quality for the classical and quantum-inspired solutions: they scored a win or tie approximately the same number of times.

The third case study focused on investigating the capabilities of a real quantum annealer on one of the hardest problems from a qubits connectivity viewpoint: the Number Partitioning Problem. We presented the task and stressed how it was able to pose great challenges to quantum annealers because of its full-connectivity requirement. As we have seen, quantum annealers only support a very specific architecture which cannot be that of a complete graph, hence a minor-embedding strategy had to be performed in order to solve such problem. We have studied how the annealing process works and highlighted the intrinsic hybridity of working with a NISQ device, which forced us to consider a model decomposition when the input size grew. We built a number of instances in such a way that the optimal solution would be always known, so that a comparison of the quality of quantum annealing solution with the global minimum could be performed. We obtained interesting results: for small and large size instances we were able to find high quality results, while for the middle-sized ones we recorded a dip in the accuracy. In order to overcome this, we investigated further the

techniques to control the annealing. We experimentally confirmed that pausing the annealing cycle, letting it rest and resuming it later, when done properly, may lead to an increase in performance with respect to the non-altered context. Given the complexity of the Number Partitioning Problem in terms of qubits connectivity and the difficulty deriving from large input sizes, we were able to the test quantum annealing heuristics and the adiabatic quantum computer on problems that modeled complexity of real-world applications. For these reasons we published our results in [19].

The last case study consisted of the application of quantum annealing to a financial task: the limited-assets markowitz portfolio optimisation problem. The goal was to model such complex task in a QUBO formalism. The main issue we encountered was the discretisation of continuous variables: since quantum annealers only work with binary variables, we had to find a suitable model that would take into account the trade-off between setting a thick discretisation, hence approximating well original variables, and bounding the number of variables in the QUBO model, which increase with the number of discretisation points. This and the fact that a lot of constraints had to be formulated as quadratic terms and inserted in the formula for the objective function, with the consequent complexity of tuning the corresponding parameters, caused the solution found by the annealer simulator to exceed the constraint of investing an amount for a total equal to 100%. However, with a thorough investigation of how to tune the parameters related to constraints, better solutions might have been found. Nevertheless, we tested the QUBO formulation in two small-sized instances of the problem and obtained results similar to those yielded by exact methods. As a consequence, we were able to consider quantum annealing as a valid alternative heuristics to classical methods, given its robustness to input size scaling and the quality of solutions that it is able to find.

To conclude, the case studies analysed in this work showed some techniques to model complex data science tasks using a quantum and hybrid quantum-classical formalism. They highlighted limitations of current and near-term quantum devices especially due to the reduced number of available qubits. However, they also stressed the importance of continuing research in this field: by improving the quality of quantum hardware we are able to solve more and more complex tasks from a wide range of fields, and by exploring all new opportunities opened up by the use of a brand new kind of hardware obeying the rules of quantum mechanics, we can approach scientific tasks in a different way, which can yield an increase in performance both in terms of problem solutions quality and of speed in computations.

Bibliography

[1] Cristian S Calude and Elena Calude. The road to quantum computational supremacy. *arXiv preprint arXiv:1712.01356*, 2017.

[2] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[3] Lov K Grover. Quantum computers can search arbitrarily large databases by a single query. *Physical review letters*, 79(23):4709, 1997.

[4] Bruce M Boghosian and Washington Taylor IV. Simulating quantum mechanics on a quantum computer. *Physica D: Nonlinear Phenomena*, 120(1-2):30–42, 1998.

[5] Davide Venturelli, Salvatore Mandrà, Sergey Knysh, Bryan O'Gorman, Rupak Biswas, and Vadim Smelyanskiy. Quantum optimization of fully connected spin glasses. *Phys. Rev. X*, 5:031040, Sep 2015.

[6] Peter Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. 08 2014.

[7] Max Riedel, Matyas Kovacs, Peter Zoller, JÃŒrgen Mlynek, and Tommaso Calarco. Europe's quantum flagship initiative. *Quantum Science and Technology*, 4(2):020501, feb 2019.

[8] Our World in Data. Moore's law: Transistors per microprocessor, 2019. Available at `https://ourworldindata.org/search?q=Moore`.

[9] IFS Labs BAS DE VOS. A quantum leap in computing power, 2019. Available at `https://blog.ifsworld.com/2019/01/a-quantum-leap-in-computing-power/`.

[10] Karl Blum. *Density matrix theory and applications*, volume 64. Springer Science & Business Media, 2012.

[11] 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.

[12] Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information. *Phys. Today*, 54:60–2, 2001.

[13] 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.

[14] Peter Wittek. *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.

[15] Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*, volume 17. Springer, 2018.

[16] Tobias Stollenwerk, Bryan O'Gorman, Davide Venturelli, Salvatore Mandrà, Olga Rodionova, Hokkwan Ng, Banavar Sridhar, Eleanor Gilbert Rieffel, and Rupak Biswas. Quantum annealing applied to de-conflicting optimal trajectories for air traffic management. *IEEE transactions on intelligent transportation systems*, 2019.

[17] D-Wave Systems Inc. Introduction to quantum annealing, 2019. Available at `https://docs.dwavesys.com/docs/latest/c_gs_2.html`.

[18] Daniele Ottaviani and Alfonso Amendola. Low rank non-negative matrix factorization with d-wave 2000q. *arXiv preprint arXiv:1808.08721*, 2018.

[19] Luca Asproni, Davide Caputo, Blanca Silva, Giovanni Fazzi, and Marco Magagnini. Accuracy and minor embedding in subqubo decomposition with fully connected large problems: a case study about the number partitioning problem. *arXiv preprint arXiv:1907.01892*, 2019.

[20] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[21] David P DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik: Progress of Physics*, 48(9-11):771–783, 2000.

[22] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, Oct 1995.

[23] Simon J Devitt, William J Munro, and Kae Nemoto. Quantum error correction for beginners. *Reports on Progress in Physics*, 76(7):076001, 2013.

[24] D-Wave Systems Inc. Introduction to the d-wave quantum hardware, 2019. Available at `https://www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware`.

[25] D-Wave Systems Inc. D-wave previews next-generation quantum computing platform, 2019. Available at `https://www.dwavesys.com/press-releases/d-wave-previews-next-generation-quantum-computing-platform`.

[26] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. Next-generation topology of d-wave quantum processors. Technical report, Technical report, 2019.

[27] D-Wave Systems Inc. D-wave qpu architecture: Chimera, 2019. Available at `https://docs.dwavesys.com/docs/latest/c_gs_4.html`.

[28] Dmitri Maslov, Yunseong Nam, and Jungsang Kim. An outlook for quantum computing. *Proceedings of the IEEE*, 107:5–10, 01 2019.

[29] Philip Ball. Ion-based commercial quantum computer is a first, 2019. Available at `https://physicsworld.com/a/ion-based-commercial-quantum-computer-is-a-first/`.

[30] Yunseong Nam, Jwo-Sy Chen, Neal C Pisenti, Kenneth Wright, Conor Delaney, Dmitri Maslov, Kenneth R Brown, Stewart Allen, Jason M Amini, Joel Apisdorf, et al. Ground-state energy estimation of the water molecule on a trapped ion quantum computer. *arXiv preprint arXiv:1902.10171*, 2019.

[31] Xanadu. Research, 2019. Available at `https://www.xanadu.ai/research/`.

[32] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[33] Augustin Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[34] James C Spall. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins apl technical digest*, 19(4):482–492, 1998.

[35] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[36] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of database systems*, pages 532–538, 2009.

[37] James Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446, 1909.

[38] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[39] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209, 2019.

[40] Florian Neukart, David Von Dollen, Christian Seidel, and Gabriele Compostella. Quantum-enhanced reinforcement learning for finite-episode games with discrete state spaces. *Frontiers in Physics*, 5:71, 2018.

[41] The AlphaStar team. Alphastar: Mastering the real-time strategy game starcraft ii, 2019. Available at `https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii`.

[42] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[43] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[44] Stephan Mertens. The easiest hard problem: Number partitioning. *Computational Complexity and Statistical Physics*, 125(2):125–139, 2006.

[45] A. H. G. Rinnooy Kan and A. van Vliet. Probabilistic analysis of packing and partitioning algorithms (e. g. coffman, jr. and george s. lueker). *SIAM Review*, 35(1):153–154, 1993.

[46] Carlo Harpes, Gerhard G. Kramer, and James L. Massey. A generalization of linear cryptanalysis and the applicability of matsui's piling-up lemma. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology — EUROCRYPT '95*, pages 24–38, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[47] Mark Lewis, Gary Kochenberger, and Bahram Alidaee. A new modeling and solution approach for the set-partitioning problem. *Comput. Oper. Res.*, 35(3):807–813, March 2008.

[48] Andrew Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2:5, 2014.

[49] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite-range tunneling? *Phys. Rev. X*, 6:031015, Aug 2016.

[50] Daniel Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical programming*, 74(2):121–140, 1996.

[51] F Cesarone, A Scozzari, and F Tardella. Efficient algorithms for mean-variance portfolio optimization with hard real-world constraints. *Giornale dell'Istituto Italiano degli Attuari*, 72:37–56, 2009.

[52] John E Beasley. Or-library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072, 1990.