Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN
DATA SCIENCE AND COMPUTATION
35° ciclo

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

UNCONVENTIONAL COMPUTING PARADIGM METHODS WITH APPLICATION TO COMPUTATIONAL CHEMISTRY

Presentata da : Marco Maronese

Coordinatore Dottorato:
Prof. Daniele Bonacorsi

Supervisore:
Prof. Andrea Cavalli
Co-Supervisore:
Dott. Sergio Decherchi

Esame finale anno 2024

Acknowledgements

I would like to thank all my colleagues and collaborators for their work and above all I want to thank Lorenzo Rocutto for these 4 years of mutual support.

I want to thank Professor Andrea Cavalli for the opportunities to work closely with various entities in my research field, enabling me to grow as a researcher.

Sergio Decherchi Ph.D, I extend my gratitude for his unwavering support throughout these years of collaboration and for the valuable teachings he provided. I then extend all my thanks to all the staff and colleagues of the Italian Institute of Technology in Genoa.

I also wish to express my thanks to Fabio Traversa Ph.D, the CTO of Memcomputing Inc., for the guidance and knowledge he shared with me in the realms of research and startups, along with the outstanding work during my internship at their headquarters.

My thanks go to Ivano Tavernelli Ph.D and Francesco Tacchino Ph.D of IBM Zurich Labs for their assistance during our collaboration and for the remarkable opportunity to undertake an internship in their laboratories.

I also thank Carlo Cavazzoni Ph.D and Daniele Dragoni Ph.D for their collaboration within the IIT-Leonardo Joint-Lab.

I would like to thank CINECA for providing access to computational time on D-Wave Systems quantum computers through the ISCRA-C project FENSIA. We are also grateful for the additional time we won via a second ISCRA-C project.

Lastly, I'd like to express my gratitude to Professor Enrico Prati from UNIMI (formerly CNR-IFN) and the QuantumTeam of CNR-IFN for the opportunities and collaboration over these years, as well as for their support throughout my research journey.

Contents

List of Figures

List of Tables

Abstract

Over the past two decades, the landscape of applied chemistry has undergone a notable transformation, with theory and modeling emerging as integral components of this scientific discipline. Alongside traditional analytical and synthetic chemistry, research in the field of chemistry has seen a profound change, mainly due to substantial advances in methodology, numerical techniques, and the exponential growth of computer software and hardware capabilities. From the beginning of the new millennium, parallel Graphics Processing Units (GPUs) started to be actively used for general-purpose computing on GPU and later found their way into fields of material science, computational chemistry, and quantum chemistry. Therefore, contemporary High-Performance Computing (HPC) clusters often provide, in addition to the so-called regular compute nodes, the GPU nodes where computations can be run on both CPU and GPU cores. Thanks to advanced HPC structures and the power of parallel processing offered by GPUs, simulations are increasingly replacing the need for dangerous and costly experiments with precise calculations. Physics-based methods such as Molecular Dynamics (DM) and Monte Carlo (MC) techniques have benefited enormously from the increasingly marked development of parallel computing. However, the modern computing paradigm starts to show limitations. These limitations have multiple natures. On the one hand, the Von Neumann architecture, which enables today's flexible general-purpose computing, suffers from the limitations of a Turing-equivalent approach. In fact, to date there are no suitable methods for the classical computer paradigm, capable of effectively solving many problems such as in many cases of combinatorial optimization (the problem to find the optimal solution) and other problems of the class called NP-hard which still remain intractable today. Furthermore, hardware development no longer guarantees an exponential increase in performance due to the engineering problems of developing ever smaller transistors. This fundamental limitation, as well as physical constraints brought on by Moore's law scaling of transistors, and a growing set of unreachably difficult optimization problems, have together spurred the interest in unconventional computing architectures.

Many bottlenecks of classical computation are still very limiting for a massive use of simulation techniques for computational chemistry. The calculation

of expectation values for thermodynamic observables, useful for studying chemical reactions, passes from the problem of estimating multidimensional integrals. Sampling problems, where you want to generate configurations of a system congruent to a probability distribution, is a huge obstacle to studying complex systems and reactions where statistically very long and expensive simulations are required. Problems, however, such as solving the Shrödinger equation or in general finding the structure of a crystal are all limited in part by the capabilities of combinatorial optimization problem solvers. The problems of sampling and optimization are therefore the two points for which the need for new computational alternatives is more evident.

In this thesis, three different promising alternatives are analyzed to address these problems that seem prohibitive for classical computation. These approaches can be divided into those that exploit the phenomenon of classical mechanics and those that instead exploit the phenomenology of quantum mechanics. Quantum computers represent the second category. In the early 1980s, quantum computers were proposed as an alternative paradigm for solving computational problems based on the exploitation of the postulates of quantum mechanics. In 1995, Shor defined an algorithm based on the concepts of quantum information and quantum computation to tackle the problem of integer factorization, which requires a polynomial increase in resources rather than an exponential one as in a Turing-based approach. Therefore, from a theoretical perspective, quantum computing surpasses the limits of classical computation for certain computational problems. However, a digital and general-purpose quantum computer as it was conceived between the 1980s and 1990s is not the only paradigm that exploits quantum phenomena. An example of this are adiabatic quantum computers (AQCs) which are instead an analog and non-Turing alternative. The third paradigm considered in this work instead represents a digital paradigm that exploits classical mechanics but is non-Turing. Memcomputing is the name given to such an emerging computational paradigm. Memcomputing exploits the evolution of a circuit based on memristors to perform computations. Memcomputing is a non-Turing paradigm that does not exploit the Von Neumann architecture. It does not have a dedicated memory component but rather exploits the evolution of a physical system. Even if these paradigms have a very strong theoretical basis, however, the challenges of building machines capable of processing these calculations remain a significant hurdle today. For this reason, it is useful to evaluate methods based on these paradigms net of the current capabilities of the machines or simulators. The first chapter presents the main limitations of today's Molecular Dynamics

and Monte Carlo methods which boil down to the two fundamental problems of optimization and sampling. In the second and third chapters, the elements relating to the tested computational paradigms are presented. The second chapter introduces quantum computing on a general level with a distinction between the universal quantum computer (or digital, circuit, or general-purpose) and the analog quantum computer, specifically the adiabatic quantum computer. In the third chapter, the Memcomputing paradigm is introduced. For all three there are descriptions of the physical implementation of the various types of quantum computers and the memcomputing circuit. This is to provide a deeper understanding of today's limitations in hardware implementation for these alternative paradigms. The fourth, fifth, and sixth chapters present 4 benchmarking works between classical solutions and methods based on quantum computers and memcomputing. In the fourth chapter, alternative quantum methods to Monte Carlo techniques for estimating integrals are studied. The performance of quantum amplitude estimation methods adapted to be implemented on modern quantum hardware was evaluated. The tests were performed on an 11-qubit trapped ion computer. In the fifth chapter, a model of water molecules arranged in a lattice was developed to test quantum optimization methods in finding the crystal structure of the lattice with the presence of ions. In the sixth chapter, the performances of non-turing paradigms (memcomputing and AQC) on NP-hard optimization problems are analyzed. Initially, AQC and memcomputing were evaluated together on 3 NP-hard problems: the Semiprime Factorization problem (FP), the Hard-Assignment Gromov-Wasserstein problem (GWP), and the Capacitated Helicopter Routing Problem (CHRP). Finally, the AQCs were tested on a machine learning application for feature extraction from satellite images where the quantum computer is used to accelerate matrix decomposition. All the tests result in good development prospects for these paradigms from quantum computing to memcomputing, however, showing the limits present in current machines.

Chapter 1

Open problems in computational chemistry

1.1 Molecular simulations

Computational studies play an increasingly important role in chemistry and biophysics, mainly thanks to improvements in hardware and algorithms. The ability to properly sample configurational and conformational properties and to subsequently describe at the atomic level the dynamical evolution of complex macromolecular systems has wide application. This research is of paramount importance in the study of macromolecular stability of mutant proteins [1], molecular recognition, ions, and small molecule transportation of the influenza M2 channel [2], protein association, the role of protein flexibility for influenza A RNA binding [3], folding and hydration, influenza neuraminidase inhibitor [4], drug resistance [5], enzymatic reactions, folding transitions [6], screening [7], accessibility assessment, and hemagglutinin fusion peptide [8]. One should also mention multivalent binding mode [9], docking [10], drug (e.g., Oseltamivir and Zanamivir) efficiency against mutants [11], structural biochemistry [12], biophysics, molecular biology, influenza multiple dynamics interactions [9], enzymology, pharmaceutical chemistry [13], biotechnology, rational epitope design [14], computation vaccinology [15], binding [16], and free energy [17].
For instance, one may wish to calculate the free energy to assess the strength and the stability of the bond in between a monoclonal antibody (mAb) and an antigen, such as the viral hemagglutinin, to quantify the efficiency of the neutralization process. In the last two decades theory and modeling turned to become one of the major topics of applied chemistry along with analytic, synthetic, and other chemistry fields. This made possible because of significant improvements in methodology, numerical methods, and computer software and hardware. Much experimental research started to include computational

modeling. The role of computer simulation in modern chemistry cannot be overestimated and the use of effective modeling and simulation plays a critical role in practical applications by providing insights into experiments and helping in system optimization. Specifically, simulations are more and more often used to substitute dangerous and expensive experiments with calculations. At the same time, the impressive progress of modern experimental research in material science and biology necessitates further developments and continuous extension of the applicability and accuracy of nowadays computational chemistry methods. The fast but accurate qualitative and quantitative modeling of large biological molecules, nanoparticles, and interfaces becomes the main focus of the research which requires significant computational efforts and is not always achievable at the current technology level. For example, most of the computational chemistry problems are about solving the Schrödinger equation for electrons in molecules or the Newton equations of motion for a system of classical particles which require large computational infrastructures such as High-Performance Computing (HPC) clusters often provide, in addition to the so-called regular compute nodes, the Graphics Processing Units (GPUs) nodes where computations can be run on both CPU and GPU cores. In fact, simulating macromolecular systems

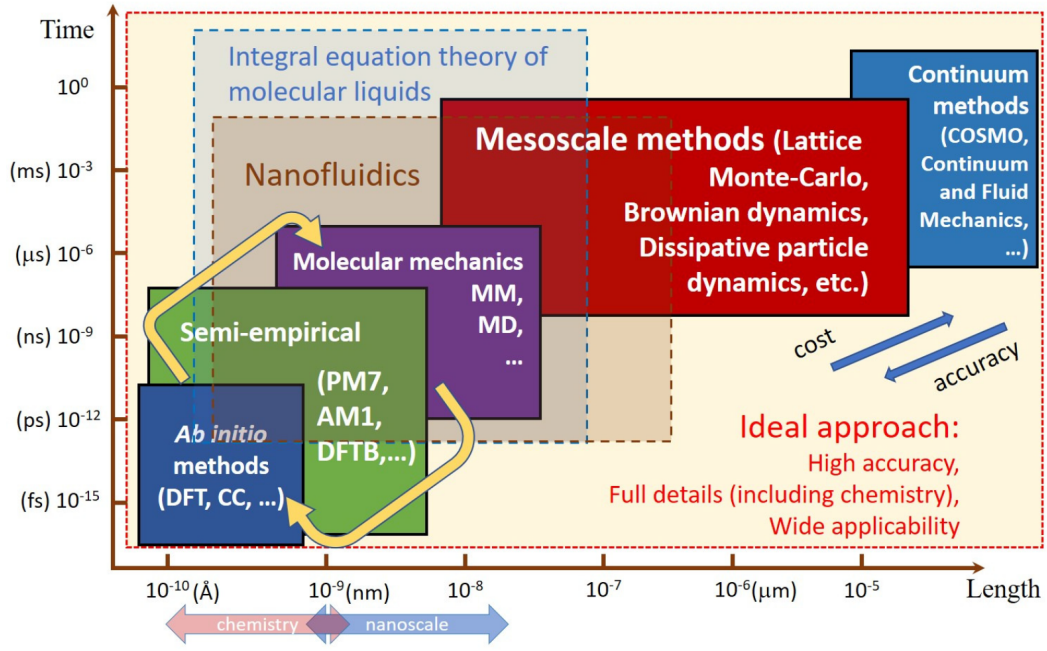


FIGURE 1.1: Length-Time scales diagram [18].

is possible today subject to a compromise between the accuracy of the simulation and the size of the system to be simulated. For this reason, there are many methodologies that are positioned as the most suitable solutions based

on the size of the system and the time scale of the dynamics that you want to simulate as shown in Figure 1.1. Consequently, mathematics should play a central role in the development of new computational methodologies and new computational paradigms. The ultimate goal is to develop numerical alternatives with a computational complexity that scales better with the size of the system in order to guarantee a better compromise between the accuracy and speed of the simulation. An important example that shows the limits of the modern computational approach in the simulation of large molecular systems concerns the study of complexes consisting of a drug and its target in solution. Such computational predictions are challenging because a comprehensive description must cover a range of time scales, from the femtosecond period of molecular vibrations to the slow diffusion rate of all species in solution, up to the millisecond and beyond to follow the binding and unbinding of drugs and targets. Such systems require massive computations for adequate statistics and a robust estimation of thermodynamic observables. thermodynamic observables are quantitatively related to free energy. For instance, the Gibbs binding free energy is directly related to the equilibrium concentration of bound ([PL]) and unbound ligand ([L]) and protein ([P]) complexes, according to

$$\Delta G_{bind} = RT \ln \frac{K_D}{C_0} \tag{1.1}$$

where T is the temperature, R is the gas constant, and C_0 is the standard state concentration of $1 mol/L$. K_D is the dissociation constant and is defined by

$$K_D = \frac{[L]\,[P]}{[LP]} \tag{1.2}$$

In terms of equilibrium thermodynamics, the ergodic theorem then provides a suitable theoretical framework for linking the chemical world to the physical observables used to assess drug potency and efficacy. In particular, for closed systems, the time average of their properties is equal to the average over the entire space. This provides the statistical properties of a system in thermodynamic equilibrium. Molecular simulation can thus merge the microscopic and macroscopic worlds by estimating the time that the system spends in a certain microscopic state. If the simulations are sufficiently extensive, they can also estimate the probability of that state. This is becoming ever more feasible thanks to modern algorithms and efficient hardware architectures.

1.2 Molecular dynamics and Monte Carlo methods

1.2.1 Monte Carlo technique in computational chemistry

The objective of Monte-Carlo (MC) simulations is to generate an ensemble of representative configurations under specific thermodynamics conditions for a complex macro-molecular system [19]. Applying random perturbations to the system generates these configurations. To properly sample the representative space, the perturbations must be sufficiently large, energetically feasible and highly probable. Monte Carlo simulations do not provide information about time evolution. Rather, they provide an ensemble of representative configurations, and, consequently, conformations from which probabilities and relevant thermodynamic observables, such as the free energy may be calculated. The calculation of free energy, as well as other thermodynamic observables, corresponds to estimates of multidimensional integrals. The ensemble average of an observable O (such as the enthalpy) is obtained by weighting the various realizations of the observable by their corresponding probability

$$\langle O \rangle = \int dp^N dr^N O(r^N, p^N) P(r^N, p^N) \tag{1.3}$$

where, for example, in an NVT ensemble The probability that the macromolecule is in a state characterized by atomic positions r^N, and atomic momenta p^N is given by

$$P(r^N, p^N) dp^N dr^N = \frac{\exp\{-\beta H(r^N, p^N)\} dp^N dr^N}{Z_{NVT}} \tag{1.4}$$

which is called Boltzmann distribution. In the last equation, H is the Hamiltonian of the system while Z is called the partition function that, in the canonical ensemble (NVT), is

$$Z \equiv \int dp^N dr^N \exp\{-\beta H(r^N, p^N)\} \tag{1.5}$$

The multidimensional integrals associated with the probability and the partition function may be efficiently calculated with a procedure called Monte Carlo integration. In this approach the integration space is sampled according to a Markovian process and the integral is approximated by the average of the corresponding sampled states. Such an approach is efficient if the sampled states have a high probability of occurrence. A sufficient, but not necessary,

condition for such an efficient sampling to hold is called detailed balance:

$$P(\mathcal{S})T(\mathcal{S} \to \mathcal{S}')A(\mathcal{S} \to \mathcal{S}') = P(\mathcal{S}')T(\mathcal{S}' \to \mathcal{S})A(\mathcal{S}' \to \mathcal{S}) \tag{1.6}$$

where $P(\mathcal{S})$ is the probability (emission probability) that the system is in the state $\mathcal{S} \equiv (r^N, p^N)$, $T(\mathcal{S} \to \mathcal{S}')$ is the transition probability from state $\mathcal{S}$ to the state $\mathcal{S}'$, and $A(\mathcal{S} \to \mathcal{S}')$ is the acceptance probability of such a transition. If we assume that the transition probability is symmetrical

$$T(\mathcal{S} \to \mathcal{S}') = T(\mathcal{S}' \to \mathcal{S}) \tag{1.7}$$

then the detailed balance equation reduces to

$$\frac{A(\mathcal{S} \to \mathcal{S}')}{A(\mathcal{S}' \to \mathcal{S})} = \frac{P(\mathcal{S})}{P(\mathcal{S}')} = \exp\{\beta\left(H(\mathcal{S}') - H(\mathcal{S})\right)\} \tag{1.8}$$

For the so-called Metropolis algorithm [20].

$$A(\mathcal{S} \to \mathcal{S}') = \min\{1, \exp\{\beta\left(H(\mathcal{S}') - H(\mathcal{S})\right)\}\} \tag{1.9}$$

Consequently, each state is defined from the previous one (Markovian process). A transition to a lower energy is always accepted, while a transition to a higher energy is accepted with probability

$$\exp\{\beta\left(H(\mathcal{S}') - H(\mathcal{S})\right)\} \tag{1.10}$$

1.2.2 Molecular dynamics simulations

Molecular dynamics studies the temporal evolution of the coordinates and the momenta (the state) of a given macromolecular structure. Such an evolution is called a trajectory. A typical trajectory is obtained by solving Newton's equations. The trajectory is important in assessing numerous time-dependent observables [21] such as the accessibility of a given molecular surface [22], the interaction in between a small molecule (e.g., a drug) and the hemagglutinin or the neuraminidase of a given influenza strain, the interaction epitope-paratope in between an antigen (e.g., hemagglutinin) and an antibody (e.g., CR8020), the appearance and disappearance of a particular channel or cavity, and the fusion of the hemagglutinin with a cell membrane (fusion peptide), amongst others. From an MD trajectory, it is possible to compute a temporal average of

an observable by averaging this observable over time along the trajectory:

$$\overline{O} = \lim_{t\to\infty} \frac{1}{t} \int d\tau O(r^N(\tau), p^N(\tau)) \tag{1.11}$$

Although it has never been formally proven (and that it is not always applicable: for instance, when the trajectory is periodic or when the phase space is constituted of disconnected regions), the ergodicity principle is often invoked [23]. The ergodicity principle states that the average over periods of time along a given trajectory of an observable is, at the limit, identical to the ensemble average of this observable as obtained, for instance, from Monte Carlo simulations:

$$\overline{O} \approx \langle O \rangle \tag{1.12}$$

Ergodicity is instrumental in performing MD simulations in the canonical and isobaric-isothermal ensemble. MD simulations are generally considered to suffer from three main limitations:

- the accuracy of the interaction model or force field (MM, QM/MM, semi-empirical, ab initio, etc.) may not enable the desired insights.
- The simulation output (the trajectory) is high-dimensional, noisy, and can be difficult to interpret and describe using a meaningful and relevant lower-dimension level of description.
- Given the limitation on the timestep, which needs to be small enough for integration to be stable and accurate, the timescales that can be sampled are often shorter than the process of interest to the researcher.

The choice of a proper potential is of the utmost importance in obtaining accurate molecular dynamics simulations [24]. The potential must be physically sound as well as computationally tractable. An approximate potential may be calculated from quantum mechanics and from the Born-Oppenheimer approximation in which only the positions of the atomic nucleus bonding are considered [24]. The potentials may be divided into bonding potentials and long-range potentials. The bonding potentials involve interaction with two atoms (bound lengths), three atoms (bound angles), and four atoms (dihedral angles). Long-range interactions are associated with the Lennard-Jones potential (van der Waal) and the Columbic potential. The harmonic approximation is utilized for the bonding potentials, which means that solely small displacements are

accurately represented. The general form of the potential is

$$\begin{aligned}\mathcal{U}(r^N) = &\sum_d k_d(d-d_0)^2 + \sum_S k_S(S-S_0)^2 + \sum_\theta k_\theta(\theta-\theta_0)^2 + \\ &+ \sum_\chi k_\chi(1+\cos\nu\chi-\delta) + \sum_\phi k_\phi(\phi-\phi_0)^2 + \\ &+ \sum_{i,j}\epsilon_{ij}\left(\left(\frac{r^0_{ij}}{r_{ij}}\right)^1 2 + \left(\frac{r^0_{ij}}{r_{ij}}\right)^6 + \frac{q_iq_j}{\epsilon_l r_{ij}}\right)\end{aligned} \tag{1.13}$$

where d is the bound length, S is the Urey-Bradley bound length, θ is the bound angle, χ is the dihedral angle, ϕ is the improper dihedral angle, r_{ij} is the distance in between atom i and j while k_d, k_S, k_θ, k_χ, and k_ϕ are constants, d_0, S_0, θ_0, ϕ_0, and r^0_{ij} are equilibrium positions, ϵ_{ij} is related to the Lennard-Jones well depth, and ϵ_l is the effective dielectric constant [25]. Finally, q_i is the partial atomic charge associated with atom i the partial charge comes from the asymmetrical distribution of the electrons in the chemical bounds. The first term on the last line is the van der Waal interaction (or Lennard-Jones potential), and the last term on the last line is the Columbic interaction. The parameters of the model are determined experimentally and from quantum mechanics. Among the most popular potentials are CHARMM and AMBER [24]. The two differ mostly in the manner in which the parameters are estimated. These potentials may model proteins, lipids, ethers, and carbohydrates, as well as small molecules (e.g., drugs). The number of interactions involved in long-range interactions rapidly becomes prohibitive. For instance, for the Columbic potential, there are potentially $N!/2!(N-2)!$ interactions (this is because it is the number of ways we can choose pairs of charges from a set of N charges), which correspond to approximately a quarter of a billion interactions for an influenza hemagglutinin. To reduce the computational burden, their action range is truncated. The truncation should be performed in such a way as not to introduce artificial discontinuities, which may result in computational artifacts.

1.3 Bottleneck in classical computational chemistry

It has already been mentioned previously that molecular dynamics suffers from the problem of choosing an accurate force-field. This problem can ideally be solved by calculating the solution of the Schrödinger equation of the molecular system in question. This problem, which today is solved approximately

for a few particles, can be reformulated as an optimization problem. In computational chemistry, a problem such as minimizing the energy of a system to determine its crystalline structure is also an optimization problem for which there are Monte Carlo methods particularly suitable for solving it but always with a limited number of particles. Finding a solver to solve an optimization problem with better scaling than existing methods today would therefore allow us to increase the simulation capabilities of molecular systems. In the previous sections, however, the problem of sampling configurations of a molecular system to estimate its thermodynamic properties has already been introduced. It therefore appears that, from a computational science point of view, finding new solutions for sampling and optimization problems are extremely preparatory for computational chemistry.

1.3.1 Sampling problem

Unlike molecular dynamics simulations, Monte Carlo simulations are free from the restrictions of solving Newton's equations of motion. This freedom allows for cleverness in the proposal of moves that generate trial configurations within the statistical mechanic's ensemble of choice. The common problem between MC methods and molecular dynamics remains the difficulty of exploring separate regions of phase space. Systems with high entropy are characterized by multimodal probability distributions and high energy barriers which lead to less exploratory sampling and therefore less accurate estimation of the integrals. The replica exchange method provides an initial alternative to overcome this problem. Many biomolecular processes involve an activation process in which a high-energy barrier exists between the initial and the final state [26]. In order to efficiently sample the macromolecular states, this type of barrier must be overcome. An efficient, although computationally expensive, approach to overcome such a barrier is called replica exchange (refer to [26] and, in the same spirit, [27]). These methods involve a certain number of non-interacting simulations, called replicas, which are performed in parallel. Each simulation is characterized by its own temperature: low temperature simulations tend to explore local minima, while high-temperature simulations may overcome energy barriers and consequently move in between local minima. To favour a better exploration of the macromolecular states, the replicas are periodically exchanged (swapped) according to the following acceptance probability:

$$A_R(\mathcal{S} \to \mathcal{S}') = \min\left[1, \exp\left\{-(\beta_{\mathcal{S}'} - \beta_{\mathcal{S}})\left(H(\mathcal{S}')\Big|_{T_{\mathcal{S}'}} - H(\mathcal{S})\Big|_{T_{\mathcal{S}}}\right)\right\}\right] \quad (1.14)$$

This acceptance probability is similar to the ones introduced before, except for the fact that each state is characterized by its own temperature. Once the exchange is completed, the simulations resume normally until another exchange is performed. The whole procedure allows for a better sampling of the macromolecular states. Various enhanced sampling methods exist to increase the ability to explore regions of phase space that are less entropic (characterized by fewer microstates that identify a given macrostate) [28]. In addition to the ability linked to the method with which to sample states of the system, there remains an intrinsic limitation linked to the estimation of the relative thermodynamic observables. In fact, from the central limit theorem, it turns out that the number of samples to be collected to obtain an estimate within a certain error ϵ, scales as

$$\mathcal{O}\left(\frac{1}{\epsilon^2}\right) \tag{1.15}$$

1.3.2 Optimization problems for crystalline structure search

The position of the constituent atoms of a macromolecular structure is usually determined either through X-ray crystallography for the larger structure or through nuclear magnetic resonance (NMR) for the smaller molecules. If only the amino acid sequence of a protein is available, the three-dimensional structure may be inferred either from methods based on homology, such as threading or from ab initio methods, which predict the structure from the sequence alone [29]. Among the larger structures associated with influenza are the hemagglutinin and the neuraminidase. Because a protein has to be crystallized to apply X-ray crystallography, the position of its constituent atoms may be distorted from their natural positions by the crystallization process. Consequently, bond lengths and bond angles may be distorted and steric clashes in between atoms may occur. Therefore, it is recommended to minimize the potential energy of the macromolecular structure to remediate this deficiency and to create a more realistic structure [30].

The global optimization of nonlinear functions, such as the potential, is a notoriously difficult problem because of the complexity of the energy landscape and the profusion of local minima [31]. Usually, only local optimization is performed. Such a minimization may be achieved through various algorithms [31] such as the steepest descent algorithm, the conjugate gradient algorithm, and the Newton-Raphson method. The first two are based on the gradient, while the latter is based on the Hessian. In most cases, local optimization is sufficient to refine the structure. If a global optimization is suited or required,

an approach such as simulated annealing could be utilized [32]. Simulated annealing is an MC method. The position of the atoms is subjected to small random displacements. The acceptance probability of such a displacement is given by

$$A_R(\mathcal{S} \to \mathcal{S}') = \min\left[1, \exp\left\{-\beta_k \left(H(\mathcal{S}')\Big|_{T_k} - H(\mathcal{S})\Big|_{T_k}\right)\right\}\right] \tag{1.16}$$

where

$$T_k \in \{T_1, T_2, \ldots, T_K\}, \;\; T_{k+1} < T_k \tag{1.17}$$

This means that the temperature acts as a control parameter. Initially, the temperature is high, which implies that transitions from lower to higher energy are allowed with a nonnegligible probability in being able to escape local minima. Subsequently, the temperature is gradually reduced (cooling) to decrease the occurrence of such a transition. Transitions to lower energy are always accepted. With a proper choice of temperatures, a global optimization may be achieved. The position of the global minimum associated with the energy landscape may be further refined with local optimization.

Chapter 2

Quantum computing

2.1 Introduction

To introduce the motivations that led to the conception of quantum computing, we must start with the key concepts of classical computational sciences and their limitations. Let's start from a key concept in computer science: Algorithms. An algorithm is a collection of simple instructions for carrying out some task [33]. The concept of algorithm has a long tradition: only in the 1930s did the fundamental principles of modern algorithm theory and calculation emerge. Alonzo Church, Alan Turing, and other pioneers of the computer age introduced these ideas, responding to a significant challenge posed by mathematician David Hilbert in the early 20th century. Hilbert's question revolved around whether there was an algorithm that could potentially solve all mathematical problems, a question sometimes called the "entscheidungsproblem". The response to Hilbert's challenge was negative: there is no algorithm capable of solving all mathematical problems. To establish this, Church and Turing faced the profound task of formally defining what we mean when we refer to the intuitive concept of algorithm in mathematical terms. In doing so, they laid the foundation for modern algorithm theory and, consequently, for the modern field of computer science. First, Turing defined a class of machines, now known as Turing machines [34], to capture the notion of an algorithm to execute a computational task. The second approach is via the computational circuit model [35]. Although these computational models appear different, it turns out that this is the case they are equivalent [36]. The limitations of the above approaches are assessed based on the resources required to solve a certain computational problem. Firstly, the study of computational complexity theory allows us to classify the difficulty of various computational problems. For instance, prime factorization is considered one of the problems that are prohibitively challenging for a classical, Turing-based computational approach which shares the same

complexity classes as a Turing machine [37]. In the early 1980s, Feynman proposed an alternative paradigm for solving computational problems based on the exploitation of the postulates of quantum mechanics [38]. In 1995, Shor defined an algorithm based on the concepts of quantum information and quantum computation to tackle the problem of integer factorization [39], which requires a polynomial increase in resources rather than an exponential one as in a Turing-based approach. Therefore, from a theoretical perspective, quantum computing surpasses the limits of classical computation for certain computational problems. However, the challenges of building machines capable of processing these calculations remain a significant hurdle today. This chapter is therefore divided as follows: firstly, we introduce the formalism of the Turing machine and the circuit model, and then we focus on their limitations. Subsequently, we introduce the concepts related to quantum computation, starting from the postulates of quantum mechanics to the elements of various quantum computing paradigms, quantum hardware implementation, and today's limitations.

2.2 Quantum mechanics for quantum computing

2.2.1 Quantum state and qubits

The first concept to introduce into quantum mechanics is related to the first postulate

Postulate 1: Associated to any isolated physical system is a complex vector space with an inner product (that is, a Hilbert space $\mathcal{H}$) known as the state space of the system. The system is completely described by its state vector, which is a unit vector in the system's state space.

Let us consider the case of a finite-dimensional Hilbert space. For an N-dimensional Hilbert space $\mathcal{H}$, we can define a basis set of N vectors $\{|i\rangle\,; i = 0, \ldots, N-1\}$ (using Dirac's notation for vectors). Therefore, each element $|\psi\rangle \in \mathcal{H}$ can be writable as

$$|\psi\rangle = \sum_{i=0}^{N-1} c_i \, |i\rangle \,, \tag{2.1}$$

where $c_i \in \mathbb{C}\ \forall i$, and therefore $\mathbb{C}^N \subseteq \mathcal{H}$. Quantum states are localized on the unit sphere centered on the origin of space $\mathcal{H}$, it follows that

$$\langle\psi|\psi\rangle = \sum_{i=0}^{N-1} |c_i|^2 = 1 \tag{2.2}$$

In Dirac's notation: the *ket* is a column vector while *bra* is the transpose-conjugate of the *ket* vector. The inner product $\langle\psi|\psi\rangle$ is so the dot product between $\langle\psi|$ and $|\psi\rangle$. In quantum computing, these definitions are useful to define the building block of quantum computation: The qubit. In principle, any quantum mechanical system that can be modeled by a two-dimensional complex vector space can be viewed as a qubit. Contrary to the bit b, defined as a minimal unit of information in classical theory and which can only take on one of two values which are generally labeled as 0 and 1, in quantum information theory, the qubit is defined as a linear combination of two vector states labeled as $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.3}$$

Without loss of generality, one can define the vector states $|0\rangle$ and $|1\rangle$ as an orthonormal basis for the 2-dimensional Hilbert space, isomorphic to $\mathbb{C}^2$, that can be written in vector representation as

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad\qquad |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.4}$$

Such a basis, as defined above, is called computational basis. From such definition it results that any qubit vector state can be written as follow:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle := \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \in \mathbb{C}^2 \tag{2.5}$$

From Equation 2.2 follow that $|\alpha|^2 + |\beta|^2 = 1$ and since Since α and β are complex numbers, a qubit state is parameterizable as

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\,|1\rangle\right) \tag{2.6}$$

where θ, φ and γ are real number. The global phase factor $e^{i\gamma}$ because it has no observable effects, qubit states corresponding to different values of γ are

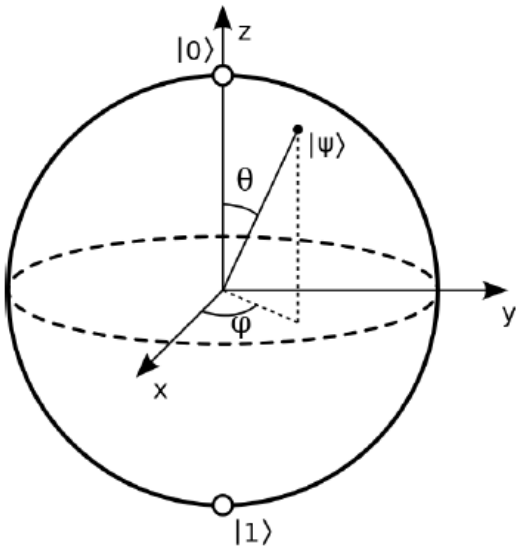


FIGURE 2.1: Graphical representation of a qubit using the Bloch sphere. Besides the classically possible states $|0\rangle$ and $|1\rangle$ superposed $|\Psi\rangle$ states are also possible.

indistinguishable [40]. A qubit state is effectively writable as

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle \tag{2.7}$$

The numbers $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi)$ define a point on the unit three-dimensional sphere as shown in the Figure 2.1.
This sphere is called the Bloch sphere. It provides a useful means of visualizing the state of a single qubit. Many of the operations on a single qubits which we describe later in this Chapter are neatly described within the Bloch sphere picture. The Cartesian coordinates of the point (θ, φ) on the Bloch sphere are given by $(\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)$.
The next step is to define a multi-qubit state: a state $|\psi\rangle$ of n qubits is an element of a Hilbert space $\mathcal{H} = \mathcal{H}_0 \otimes \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_{n-1}$. The Hilbert spaces $\mathcal{H}_i$ are 2-dimensional so we define a computational basis $(|0\rangle\,; |1\rangle)$ over any space $\mathcal{H}_i$. Following such a definition, we can assign a computational basis for the space $\mathcal{H}$ too. The computational basis of $\mathcal{H}$ is given by the tensor product between the elements of the computational basis of the spaces $\mathcal{H}_i$. Therefore we write that the computational basis of $\mathcal{H}$ is

$$\{|k_0\rangle \otimes |k_1\rangle \otimes \cdots \otimes |k_{n-1}\rangle \,|\, k_i \in \{0, 1\} \forall i\} \tag{2.8}$$

but we can simplify the notation by writing $\{|s\rangle \, | s \in \{0,1\}^n\}$. The set $\{0,1\}^n$ is the set of all binary strings of n bit. we can find a vector representation also for such computational basis of a multi-qubit Hilbert space. For example, if we consider the state $|0\rangle \otimes |1\rangle$ then

$$|01\rangle = |0\rangle \otimes |1\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{2.9}$$

A n-qubit Hilbert $\mathcal{H}^n$ space is a 2^n vector space, indeed the number of elements of the computational basis is 2^n, so a generic quantum state $|\psi\rangle \in \mathcal{H}^n$

$$|\psi\rangle = \sum_{s \in \{0,1\}^n} \alpha_s \, |s\rangle \tag{2.10}$$

can be written as a vector of a space $\mathbb{C}^{2^n}$. Also in such case the state $|\psi\rangle$ must respect the normalization condition:

$$\sum_{s \in \{0,1\}^n} |\alpha_s|^2 = 1 \tag{2.11}$$

We define the following notation: given a set of qubits, if we want to define the number of qubits of the set we will use the lowercase Latin letters (such as $n, m, l, \dots$) as we have already done. When we specify the number of dimensions of the correspondent multi-qubit Hilbert space (such as $2^n, 2^m, 2^l, \dots$) we will use the correspondent uppercase Latin letters (then $N \equiv 2^n, M \equiv 2^m, L \equiv 2^l, \dots$). After such a definition we can introduce another way to write the computational basis of a multi-qubit Hilbert space in order to simplify the notation: we can replace the binary string s of the element $|s\rangle$ of the computational basis with the correspondent decimal number. In such way the computational basis of a n-qubits Hilbert space can be written as $\{|0\rangle, |1\rangle, |2\rangle, \dots, |N-1\rangle\}$ Therefore the state $|\psi\rangle$ in Eq. 2.11 can be written as

$$|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i \, |i\rangle \quad , \quad \sum_{i=0}^{N-1} |\alpha_i|^2 = 1 \tag{2.12}$$

A complex coefficient α_i is sometimes called an *amplitude*.

2.2.2 Transformations of quantum states

To define the transformation that brings a quantum state $|\psi\rangle$ to a new state $\left|\psi'\right\rangle$, The following postulate gives a prescription for the description of such state changes.

Postulate 2: The evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $\left|\psi'\right\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2.

Since a qubit (or multi-qubit) state can be thought of as an element of the unitary radius sphere of a Hilbert space, the transformations that we perform over $|\psi\rangle$ must leave invariant the square module of $\langle\psi|\psi\rangle$. The operators that satisfy this criterion are operators $\hat{U}$ such that

$$\left\langle \hat{U}\psi \middle| \hat{U}\psi \right\rangle = \langle\psi|\psi\rangle \tag{2.13}$$

The transformations that can be performed on qubits are mathematically defined by unitary operators i.e. operators such that

$$\hat{U}\hat{U}^{\dagger} = \hat{U}^{\dagger}\hat{U} = I \tag{2.14}$$

A that acts on a one qubit state can be represented as a two-by-two matrix and then an element of the group $U\left(2\right)$.

A general element of $U \in U(2)$ is writable as

$$U = u_0 I - iu_1\sigma_1 - iu_2\sigma_2 - iu_3\sigma_3 = \begin{pmatrix} u_0 - iu_3 & -(u_2 + iu_1) \\ u_2 - iu_1 & u_0 + iu_3 \end{pmatrix} \tag{2.15}$$

where $u_i \in \mathbb{R}\ \forall i$ and the three unitary matrices $\sigma_1, \sigma_2, \sigma_3$ (preferably written like as $\sigma_x, \sigma_y, \sigma_z$ or X, Y, Z) are called Pauli matrices

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.16}$$

Therefore, the group $U(2)$ is isomorphic to a real values vector space, consequently it is easily parameterizable. Since a n-qubit state is defined on a N-dimensional Hilbert space, where $N = 2^n$, the operators that act on that are elements of the unitary group $U(N)$. For a natural number $N \in \mathbb{N}$, the set of all $N \times N$ unitary matrices, that are isometries of the N-dimensional Hilbert

space $\mathcal{H}$, forms the a group called the unitary group $U(N)$. To define a parameterization of a general $U(N)$ group, it is necessary to define the Lie group and the Lie algebra. In a nutshell every matrix $U \in U(N)$ can be writable as

$$U = e^{-iH} \text{ where } H \in H(N) \tag{2.17}$$

The set $H(N)$ is the set of all $N \times N$ hermitian matrices, i.e., the matrices such that $\hat{H} = \hat{H}^\dagger$. The $H(N)$ form a Lie algebra for the unitary group, which means that $H(N)$ is a vector space and

$$U = e^{-i\sum_i a_i H_i} \tag{2.18}$$

indeed

$$H_i = i\frac{\partial U}{\partial a_i}\bigg|_{a=0} \tag{2.19}$$

which means that $H(N)$ is the vector space tangent to $U(N)$. The dimension of the Lie algebra vector space is equal to the number of the independent free parameters of an element of $U(N)$ which is N^2. Therefore an operator on n-qubit is completely defined by 4^n real parameters. A simple way to parameterize $U(N)$ is therefore by choosing a basis for the space $H(N)$. Let's define the following base $\{\mathbb{I}, X, Y, Z\}^{\otimes n}$ where n is defined such that $N = 2^n$, which is the set of all possible tensor products between n elements of $H(2)$ basis $\{\mathbb{I}, X, Y, Z\}$. Lie algebra is defined with a noncommuting vector basis. For the algebra of $U(2)$, for example, the three elements, which together with the identity, constitute the basis of the algebra follow the following commutation relation

$$[\sigma_i, \sigma_j] = 2i\epsilon_{ijk}\sigma_k \tag{2.20}$$

for this reason, the exponential in Equation 2.18 cannot be decomposed into a succession of independent transformations in a trivial way, in fact the Baker–Campbell–Hausdorff formula is valid [41]

$$e^{tA}e^{tB} = e^{tZ}, \quad Z = A + B + \frac{1}{2}[A, B] + \frac{1}{12}[A, [A, B]] - \frac{1}{12}[B, [A, B]] + \dots \tag{2.21}$$

where A and B are quadratic matrices and t is a scalar. Since the elements of the basis of the algebra of $U(N)$ do not commute between each other then, from the Equation 2.18 which

$$U = e^{-i\sum_i a_i H_i} \neq \prod_i e^{-ia_i H_i} \tag{2.22}$$

However, there is an approximation that allows the Equation 2.18 to be written as a product of unitary operators with an error of $\mathcal{O}(t^2)$ [42]. The Trotter-Suzuki formula [36] said that

$$e^{it(A+B)} = \lim_{t\to\infty} \left(e^{i\frac{t}{m}A} e^{i\frac{t}{m}B}\right)^m \tag{2.23}$$

where A and B are hermitian operators. Another way to define operators on a Hilbert space is with the Dirac notation. A one-qubit operator can be written as

$$U = U_{00}\,|0\rangle\langle 0| + U_{01}\,|0\rangle\langle 1| + U_{10}\,|1\rangle\langle 0| + U_{11}\,|1\rangle\langle 1| \tag{2.24}$$

and since the projection operator $|i\rangle\langle j|$ act on an element $|k\rangle$ of computational basis as $|i\rangle\langle j|\,|k\rangle = \delta_{jk}\,|i\rangle$, then we can represent the action of the one-qubit operator U on a one-qubit state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ as a matrix applied to a vector:

$$\begin{aligned} U\,|\psi\rangle &\equiv \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha U_{00} + \beta U_{01} \\ \alpha U_{10} + \beta U_{11} \end{pmatrix} \\ \Rightarrow\ U\,|\psi\rangle &= \Big(\alpha U_{00} + \beta U_{01}\Big)\,|0\rangle + \Big(\alpha U_{10} + \beta U_{11}\Big)\,|1\rangle \end{aligned} \tag{2.25}$$

In general an unitary operator that acts over an n-qubits Hilbert space can be written as

$$\hat{U} = \sum_{i,j=0}^{N-1} U_{ij}\,|i\rangle\langle j| \tag{2.26}$$

which can be represented as the $N \times N$ matrix $(U_{ij}) \in U(N)$.

2.2.3 Dynamics of closed quantum systems

A more refined version of the second postulate can be given which describes the time evolution of a quantum system, by defining the dynamic equation of a closed quantum system, called Schrödinger equation

$$i\hbar \frac{d\,|\psi\rangle}{dt} = H\,|\psi\rangle \tag{2.27}$$

Where'H' is a Hermitian operator representing the system's Hamiltonian, which contains information regarding the system's available energy levels. Consequently, it enables the description of all dynamic properties of the system. A property of the Hermitian matrices is that there exists a spectral decomposition

and all the eigenvalues are real. Therefore, in Dirac notation

$$H = \sum_E E \, |E\rangle\langle E| \tag{2.28}$$

where E is the eigenvalue, which represents the accessible energy levels of the system, and $|E\rangle$ are the eigenvectors, or eigenstates, and therefore $|E\rangle\langle E|$ is the projector on the eigenstates $|E\rangle$. The discussion about the meaning of eigenvalue and eigenstates of hermitian operators in quantum mechanics will be held in the next section. From the solution of the Schrödinger equation the second postulate is retrieved

$$|\psi(t_1)\rangle = \exp\left\{-i\frac{H\,(t_1 - t_0)}{\hbar}\right\}|\psi(t_0)\rangle = U(t_1, t_0)\,|\psi(t_0)\rangle \tag{2.29}$$

where the unitary operator $U(t_1, t_0)$ is defined as time evolution operator

$$U(t_1, t_0) \equiv \exp\left\{-i\frac{H\,(t_1 - t_0)}{\hbar}\right\} \tag{2.30}$$

The eigenstates $|E\rangle$ of the Hamiltonian operator are often called stationary states, indeed $U(t_1, t_0)$ share the same eigenstate of H

$$\exp\left\{-i\frac{H\,(t_1 - t_0)}{\hbar}\right\}|E\rangle = |E\rangle \exp\left\{-i\frac{E\,(t_1 - t_0)}{\hbar}\right\} \tag{2.31}$$

indeed the global phase has no observable effects.

2.2.4 Observables and Measurement

In quantum mechanics, there are two protagonists: the system and the observer. An interaction between the observer and the system is called a measurement. Properties of the system that can be measured are called observables. Examples are position, momentum, angular momentum, energy, etc. Each observable corresponds to a hermitian operator O. To retrieve information from a quantum system the observer has to interact with the system and then perform a measurement. The second postulate says that a closed quantum system evolves with a unitary transformation but a measurement operation makes the system not closed anymore. TO define a measurement it is necessary to define an observable operator O that, since is hermitian, there exists a spectral decomposition

$$O = \sum_i O_i \, |i\rangle\langle i| \tag{2.32}$$

where $|i\rangle$ are the eigenstates and O_i are the eigenvalues. The projector operators $|i\rangle\langle i|$ are called measurement operators and it projects a quantum state $|\psi\rangle$ on the eigenstate $|i\rangle$ with eigenvalue O_i which is the outcome of the measurement. The meaning of these elements comes from the third postulate, which provides a description of the effects of measurements on quantum systems.

Postulate 3: Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement then the probability that result m occurs is given by

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{2.33}$$

and the state of the system after the measurement is

$$|\psi_m\rangle = \frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}} \tag{2.34}$$

The measurement operators satisfy the *completeness equation*,

$$\sum_m M_m^\dagger M_m = I \tag{2.35}$$

This equation being satisfied for all $|\psi\rangle$ is equivalent to the completeness equation. However, the completeness equation is much easier to check directly, so that's why it appears in the statement of the postulate. A simple but important example of a measurement is the measurement of a qubit in the computational basis. This is a measurement on a single qubit with two outcomes defined by the two measurement operators $M_0 \equiv |0\rangle\langle 0|$, $M_1 \equiv |1\rangle\langle 1|$. Observe that each measurement operator is Hermitian, and that $M_0^2 = M_0$, $M_1^2 = M_1$. Thus the completeness relation is obeyed, $\mathbb{1} = M_0^\dagger M_0 + M_1^\dagger M_1 = M_0 + M_1$. Suppose the state being measured is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Then the probability of obtaining measurement outcome 0 is

$$p(0) = \langle\psi|M_0^\dagger M_0|\psi\rangle = \langle|\psi\rangle|M_0||\psi\rangle\rangle = |\alpha|^2 \tag{2.36}$$

Similarly, the probability of obtaining the measurement outcome 1 is $p(1) = |\beta|^2$. The state after measurement in the two cases is therefore

$$\begin{aligned} \frac{M_0 |\psi\rangle}{|\alpha|} &= \frac{\alpha}{|\alpha|} |0\rangle \\ \frac{M_1 |\psi\rangle}{|\beta|} &= \frac{\beta}{|\beta|} |1\rangle \end{aligned} \tag{2.37}$$

but the value $\frac{\alpha}{|\alpha|}$ and $\frac{\beta}{|\beta|}$ are global phase so they can be ignored. For a n-qubit system, we could measure just a subset of the qubits. Let's write a multi-qubits state as in equation 2.10

$$|\psi\rangle = \sum_{s \in \{0,1\}^n} \alpha_s |s\rangle \tag{2.38}$$

Measuring the first qubit, labelled with q_0, alone gives 0 with probability

$$p_{q_0}(0) = \sum_{s' \in \{0,1\}^{n-1}} |\alpha_{s'0}|^2 \tag{2.39}$$

leaving the post-measurement state

$$|\psi\rangle = \sum_{s' \in \{0,1\}^{n-1}} \frac{\alpha_{s'0}}{\sqrt{p_0(0)}} |s'\rangle \tag{2.40}$$

Let's take as an example the Hamiltonian operator H, the observable energy, from the Equation 2.28. By splitting the state $|\psi\rangle$ into the eigenstate basis of H

$$|\psi\rangle = \sum_E \langle E|\psi\rangle |E\rangle \tag{2.41}$$

The quantum state $|\psi\rangle$ is, therefore, in superposition between the state $|E\rangle$ which means the outcome of the energy measurement is E with a probability $|\langle E|\psi\rangle|^2$. It is possible to compute the expectation value of the energy measurement

$$\langle\psi|H|\psi\rangle = \langle\psi| \sum_E E |E\rangle\langle E| |\psi\rangle = \sum_E E |\langle E|\psi\rangle|^2 \tag{2.42}$$

There are two important characteristics of measure operators: In general they do not commute and they are not invertible The first characteristic leads to the order of two consecutive measurements do not bring to the same state. The Heisenberg principle is related to such property.

Let's consider two observatories that do not commute and therefore do not share a set of eigenstates. After a measurement with the observable O_1, the measurement value O_{1i} is obtained, and the state is projected with $|O_{1i}\rangle\langle O_{1i}|$

into the eigenstate $|O_{1i}\rangle$. However, $|O_{1i}\rangle$ is not eigenstate O_2, this results in an uncertainty in the measurement of O_2 since

$$|O_{1i}\rangle = \sum_j \langle O_{2j}|O_{1i}\rangle \, |O_{2j}\rangle \tag{2.43}$$

On the other hand, if a measurement with O_2 is performed first, the state will be projected onto a superposition state on the O_1 eigenstates. This uncertainty is the basis of the Heisenberg uncertainty principle whereby I cannot measure with arbitrary precision two observables that do not commute. In quantum computing, it is conventional to write the n-qubit quantum states in a superposition between states of the computational basis which are eigenstates of the operator $Z^{\otimes n}$, which is the tensor product of the Pauli Z operator on n qubits. Therefore the states of the qubits are, by convention, measured via the observable $Z^{\otimes n}$. From a more general point of view, to consider a system evolved through the measurement operation, we must consider the system as an open system and to do so we need to consider the formalism of density matrices. The density operator formalism provides a description of systems whose state is not completely known. More precisely, suppose a quantum system is in one of a number of states $|\psi_i\rangle$, with a probability p_i. The set $\{|\psi_i\rangle\}$ is called an ensemble of pure states. A quantum system whose state $|\psi\rangle$ is known exactly is said to be in a pure state. The density operator for a system defined in an ensemble $\{|\psi_i\rangle\}$ is defined as

$$\rho \equiv \sum_i p_i \, |\psi_i\rangle\langle\psi_i| \tag{2.44}$$

It is said that the system, described by a density operator ρ, is in a mixed state if if there is no unitary vector $|\psi\rangle$ on a Hilbert space such that $\rho = |\psi\rangle\langle\psi|$ but it can be written only as an ensemble of pure states. In an open system, a quantum state is no longer considered as a unit vector on a Hilbert station but it is necessary to consider an enabler of quantum states, in fact, in general, an open system can be described as a mixed state. One criterion for understanding whether a system is in a mixed or pure state is to evaluate the trace of the density operator associated with the square. If $\mathrm{Tr}\{\rho^2\} = 1$ that ρ is a pure state, otherwise, if $\mathrm{Tr}\{\rho^2\} < 1$ then it is a mixed state. The three postulates of quantum mechanics can be rewritten with the density operator formalism. The evolution of a closed quantum system described by ρ is

$$\rho = \sum_i p_i \, |\psi_i\rangle\langle\psi_i| \xrightarrow{U} \sum_i p_i U \, |\psi_i\rangle\langle\psi_i| \, U^\dagger = U\rho U^\dagger \tag{2.45}$$

where U is a unitary operator. Instead, the measurements are described in the density operator formalism in the following way: Given measurements operators M_m, it is useful to consider that from a state $|\psi_i\rangle$, the probability to measure m is

$$p(m|i) = \langle\psi_i|M_m^\dagger M_m|\psi_i\rangle = \mathrm{Tr}\{M_m^\dagger M_m\,|\psi_i\rangle\langle\psi_i|\} \tag{2.46}$$

Then, the probability to obtain the results m from ρ is

$$p(m) = \sum_i p(m|i)p_i = \sum_i p_i\,\mathrm{Tr}\{M_m^\dagger M_m\,|\psi_i\rangle\langle\psi_i|\} = \mathrm{Tr}\{M_m^\dagger M_m\rho\} \tag{2.47}$$

and, after the measurement transforms ρ in

$$\rho_m = \sum_i p_i \frac{M_m\,|\psi_i\rangle\langle\psi_i|\,M_m^\dagger}{\mathrm{Tr}\Big\{M_m^\dagger M_m\,|\psi_i\rangle\langle\psi_i|\Big\}} = \frac{M_m\rho M_m^\dagger}{\mathrm{Tr}\Big\{M_m^\dagger M_m\rho\Big\}} \tag{2.48}$$

The region where it is important to define the formalism of density operators in quantum computing is to describe subsystems that make up a quantum system. Given two systems A and B, the whole system is described by the density operator ρ^{AB} while the reduced density operator for the system A is

$$\rho^A \equiv \mathrm{Tr}_B\left(\rho^{AB}\right) \tag{2.49}$$

where Tr_B is known as partial trace over the system B. Given a density operator $|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|$, where $|a_1\rangle, |a_2\rangle \in \mathcal{H}_A$ and $|b_1\rangle, |b_2\rangle \in \mathcal{H}_B$, the partial trace is defined as

$$\mathrm{Tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) = |a_1\rangle\langle a_2|\,\mathrm{Tr}\{|b_1\rangle\langle b_2|\} = |a_1\rangle\langle a_2|\,\langle b_1|b_2\rangle \tag{2.50}$$

In general $\rho^A \equiv \mathrm{Tr}_B\left(\rho^{AB}\right)$ is a mixed state. Let's consider now a general (n_1+n_2)-qubit state, defined in the Hilbert space $\mathcal{H}_2^{\otimes(n_1+n_2)} = \mathcal{H}_2^{\otimes n_1} \otimes \mathcal{H}_2^{\otimes n_2}$ that, for semplicity, it is writable $\mathcal{H}^{AB} = \mathcal{H}^A \otimes \mathcal{H}^B$

$$|\psi\rangle = \sum_i \sum_j c_{ij}\,|i\rangle \otimes |j\rangle \tag{2.51}$$

The partial trace over the space $\mathcal{H}_2^{\otimes n_2}$ of $\rho = |\psi\rangle\langle\psi|$ is

$$\rho_A = \mathrm{Tr}_B\left(\rho\right) = \mathrm{Tr}_B\left(\sum_{ijkl} c_{ij}\bar{c}_{kl}\,|i\rangle\langle k| \otimes |j\rangle\langle l|\right) = \sum_{m=0}^{N_2-1}\sum_{ik} c_{im}\bar{c}_{km}\,|i\rangle\langle k| \tag{2.52}$$

That is the resulting mixed state after measurements with projections $|m\rangle\langle m|$.

The reason for doing this is because the partial trace operation is the unique operation that gives rise to the correct description of observable quantities for a subsystem of composite systems [36].

Entanglement

Operations over more then one qubit leads to particular cases. Let's consider two qubits state of two different Hilbert space: an n-qubits state $|\psi\rangle \in \mathcal{H}^n$ and an m-qubits state $|\psi'\rangle \in \mathcal{H}^m$. Let's consider now a gate A which acts on $\in \mathcal{H}^n$ and B which acts on $\in \mathcal{H}^m$ then we define an operator $A \otimes B$ such that

$$\Big(A \otimes B\Big)\big(|\psi\rangle \otimes |\psi'\rangle\big) \equiv A\,|\psi\rangle \otimes B\,|\psi'\rangle \tag{2.53}$$

Such gate $A \otimes B$ can be easily written as a $NM \times NM$ matrix in the following way:

$$A \otimes B \equiv \begin{pmatrix} A_{00}B & A_{01}B & \dots & A_{0(N-1)}B \\ A_{10}B & A_{11}B & & \vdots \\ \vdots & & \ddots & \\ A_{(N-1)0}B & \dots & & A_{(N-1)(N-1)}B \end{pmatrix} \in U(NM) \tag{2.54}$$

The operators $A \otimes B$ acts over the space $\mathcal{H}^{n+m} \equiv \mathcal{H}^n \otimes \mathcal{H}^m$ and it doesn't create entaglement between the first set of n qubits with the set of m qubits. Instead if we define an operator U which acts over the space $\mathcal{H}^{n+m}$ then it is not always possible to write it as a tensor product between two operators that act one over $\mathcal{H}^n$ and the other one over $\mathcal{H}^m$.
Such operators are called *not separable* and they are able to transform two separate states into an entangled state. Entanglement is the second fundamental difference between quantum bits and classical ones after the superposition over different bitstrings. In order to explain the notion of entangled states we need to define what is a separable state (Ref. [43]). A n-qubits quantum state is said to be *separable* if it can be written as the tensor product of the states of single-qubit state:

$$|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle \otimes \cdots \otimes |\psi_{n-1}\rangle \tag{2.55}$$

A n-qubits quantum state that is not separable is called an entangled state. Entangled states are quantum states that cannot be described only by looking at individual single-qubit system. Entangled states are easier to understand with an example. Commonly used quantum state in the literature are the Bell

states

$$\begin{aligned} \left|\Phi^+\right\rangle &= \frac{1}{\sqrt{2}}\Big(|00\rangle + |11\rangle\Big) \qquad \left|\Phi^-\right\rangle = \frac{1}{\sqrt{2}}\Big(|00\rangle - |11\rangle\Big) \\ \left|\Psi^+\right\rangle &= \frac{1}{\sqrt{2}}\Big(|01\rangle + |10\rangle\Big) \qquad \left|\Psi^-\right\rangle = \frac{1}{\sqrt{2}}\Big(|01\rangle - |10\rangle\Big) \end{aligned} \tag{2.56}$$

Let's take, for example, the Bell state $|\Phi^+\rangle$: Let's rewrite $|\Phi^+\rangle$ as

$$\left|\Phi^+\right\rangle = \frac{|0\rangle_{q_1} \otimes |0\rangle_{q_0} + |1\rangle_{q_1} \otimes |1\rangle_{q_0}}{\sqrt{2}} \tag{2.57}$$

If we consider only the state of the qubit q_0 it comes out that the qubit has an equal probability to be in the state $|0\rangle$ or $|1\rangle$ after measurement. The same observation holds for the other qubit. But these individual descriptions are not sufficient to fully describe $|\Phi^+\rangle$. For example, a state like

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)_{q_1} \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)_{q_0} \tag{2.58}$$

satisfies the 2 individual descriptions, but is not equal to $|\Phi^+\rangle$. In order to fully describe $|\Phi^+\rangle$ we should add another condition that will link the qubits q_0 and q_1 together. In the case of $|\Phi^+\rangle$ measuring qubit q_0 will necessarily collapse the state of qubit q_0 to either $|0\rangle$ or $|1\rangle$, but as qubits q_0 and q_1 are entangled, the state of qubit q_1 will also be impacted by the measurement and depending on the value measured for q_0, the final state after measuring qubit q_0 is either $|0\rangle \otimes |0\rangle$ or $|1\rangle \otimes |1\rangle$. To summarise, 2 or more qubits are entangled if their states are linked and cannot be fully described individually.

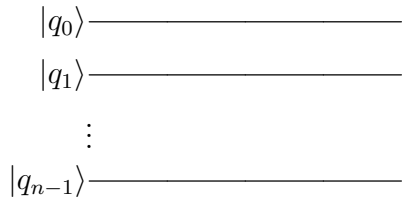
2.3 Elements of circuital quantum computing

2.3.1 Quantum circuits

Having defined the elements of quantum mechanics, it is possible to introduce the quantum equivalent of classical circuital computing where the gates are not boolean operations but unitary operators acting on qubit vector states, The sequence of operations represents the computation on a circuital quantum computer called a quantum circuit. A quantum circuit is a sequence of gates (unitary operators) applied on sets of qubits called *quantum register*. Such a quantum circuit is similar to the one used in classical computing: a quantum circuit is composed of quantum gates applied sequentially to a given number of qubits. Thanks to the similarity between the classical and the quantum models,

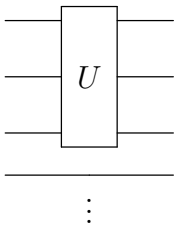
many mathematical tools and formalisms from classical computing can be recast in the quantum world.
A quantum circuit is identified by a graph composed of the following elements:

- Each qubit of the registers is represented by a horizontal line.



  The qubit identifier can be written at the extremities of the line. The time evolution of the qubits goes from left to right. Generally, the qubits of a register are initialized in the state $|0\rangle$.
  A register qubit is ordered as follows: the first qubit q_0 is the first from the right in the binary string s which identifies the element $|s\rangle$ of the computational basis i.e. it is the last qubit in the string s and for such reason, it is called *the less significant qubit*.

- Each gate U has a unique representation: a tile of square shape on the qubits line on which the gate U is applied



- At the end to the computation, to extract information from the a qubit state, we apply an operation called measurement (we will talk about such operation in the follow section) represented on the circuit with the following icon

  The measurement operation breaks the superposition of states of the measured qubit and it returns a value which can be stored in a classical bit represented by a doubled horizontal line.

A quantum circuit without measurement is reversible since all unitary operators (gates) are reversible.
This is an important property of the quantum circuit that a classical version has not. If we want to extract information we break the reversible property of the circuit because the measurement operation is not unitary and, in particular, not invertible. It is interesting to underline that this formalism is based on the second postulate to define operations on qubits as unitary transformations. The second postulate, however, is valid only if it is considered an isolated system and this leads to complications in the creation of devices capable of carrying out such calculations.

2.3.2 Elementary gates

The following Section contains references from [44] and [45].

One-qubit gates

The simplest gate that we can define on a single qubit is the identity operator:

$$I = |0\rangle\langle 0| + |1\rangle\langle 1| \tag{2.59}$$

We can also write it as a matrix:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.60}$$

Then a set of important gates corresponds to the Pauli matrices, called Pauli gates:

$$\begin{aligned} X &= |1\rangle\langle 0| + |0\rangle\langle 1| \quad , \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ Y &= i\,|1\rangle\langle 0| - i\,|0\rangle\langle 1| \quad , \quad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \\ Z &= |0\rangle\langle 0| - |1\rangle\langle 1| \quad , \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{aligned} \tag{2.61}$$

The following equivalences with the Pauli matrices hold: $\sigma_1 = X$, $\sigma_2 = Y$, $\sigma_3 = Z$, where σ_n is the n-th Pauli matrix.
It's worth noting that we can obtain each one of the gates X, Y, and Z from the other two in the following way: $X = -iYZ$, $Y = -iXZ$ and $Z = iXY$.
The gate X is the equivalent of the NOT operation in the classical computation indeed it trasforms $|0\rangle \to |1\rangle$ and $|1\rangle \to |0\rangle$ and it is also called *bitflip* gate while

the gate Z is called *phaseflip* gate since it acts on the computational basis in the following way: $|0\rangle \to |0\rangle$ and $|1\rangle \to -|1\rangle$ and it has not an equivalent gate with the classical computation.
Such single qubit gate, $\mathbb{1}, X, Y$ and Z are a basis for the space $U(2)$. Indeed the set of the 2×2 unitary matrix has a vector space structure and a matrix $U \in U(2)$ can be written as

$$U = u_0 I + u_1 X + u_2 Y + u_3 Z \tag{2.62}$$

where the u_0, u_1, u_2 are u_3 are real numbers. Every transformation of a single-qubit state can be seen as a rotation of the Bloch sphere. Let's introduce the gates which realize the rotations of angle θ of the Bloch sphere on the axis $\hat{x}$, $\hat{y}$ and $\hat{z}$ and they are called respectively R_x, R_y and R_z

$$\begin{aligned}
R_x(\theta) &= e^{-i\frac{\theta}{2}X} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} X = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \\
R_y(\theta) &= e^{-i\frac{\theta}{2}Y} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} Y = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \\
R_z(\theta) &= e^{-i\frac{\theta}{2}Z} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} Z = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}
\end{aligned} \tag{2.63}$$

The gate R_z is called *phase gate* since it introduces a relative phase $e^{i\theta}$ on a superposition state. We can also define a rotation of θ of the Bloch sphere on an axis identified by an arbitrary 3-dimensional real vector $\vec{n} = (n_x, n_y, n_z)$ in the following way

$$R_{\vec{n}}(\theta) = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} \left(n_x X + n_y Y + n_z Z\right) \tag{2.64}$$

If $n_x = 0$ we obtain that $X R_{\vec{n}}(\theta) X = R_{\vec{n}}(-\theta)$ that because $XYX = -Y$ and $XZX = -Z$. Another relevant operation is the Hadamard gate:

$$Had = \frac{1}{\sqrt{2}} \left(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|\right) \tag{2.65}$$

which can be written as:

$$Had = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.66}$$

This gate corresponds to a rotation of $\frac{\pi}{2}$ about the axis $\hat{y}$, and maps the basis states to an equally probable superposition of the basis states:

$$\begin{aligned} Had\,|0\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \equiv |+\rangle \\ Had\,|1\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \equiv |-\rangle \end{aligned} \tag{2.67}$$

An important property of the gate Had is that $Had = Had^{\dagger}$ (i.e. the Hadamard gate is hermitian) and then $H^2 = \mathbb{1}$ since H is also unitary. The gates X, Z and clearly $\mathbb{1}$ have such property too.
Remembering that two matrices A and B are called similar if it exists an invertible matrix M such that $A = M^{-1}BM$, we now have the following similarity relations: $HadXHad = Z$ ($HadZHad = X$) and $HadYHad = -Y$. The following theorem (Ref. [44]) shows how we can decompose a general single-qubit gate U in a sequence of gates known to us.

Theorem 2.3.1 (Z-Y decomposition for a single qubit)**.** *Suppose U is a unitary operation on a single qubit. Then there exist real numbers α, β, γ and δ such that*

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta) \tag{2.68}$$

The utility of such a theorem lies in the following corollary

Corollary 2.3.1.1. *Suppose U is a unitary gate on a single qubit. Then there exist unitary operators A, B and C on a single qubit such that $ABC = \mathbb{1}$ and $U = e^{i\alpha} AXBXC$ where α is some overall phase factor.*

It is important to specify that in such case $e^{i\alpha} \equiv e^{i\alpha}\mathbb{1}$.
There are other special gates that are important for a reason which will be shown in the next Section because they lead to the universality. Such gates are

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad\qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \tag{2.69}$$

We can see that $T = \sqrt{S}$.
Moreover, there is a set of three gates that are important for the quantum computers that will be used in such a discussion. The most important of such a set of three gates is

$$U_3(\theta, \phi, \lambda) = \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\lambda+\phi)}\cos\frac{\theta}{2} \end{pmatrix} \tag{2.70}$$

Because if we apply it on a single qubit state $|0\rangle$ we obtain the most general single qubit state

$$U_3(\theta, \phi, \lambda)\,|0\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle \tag{2.71}$$

as it is written the equation 2.7. The other two gates are

$$\begin{aligned} U_1(\lambda) = U_3(0,0,\lambda) &= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix} \\ U_2(\phi,\lambda) = U_3(\frac{\pi}{2},\phi,\lambda) &= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i(\lambda+\phi)} \end{pmatrix} \end{aligned} \tag{2.72}$$

Multi-qubit gates

Let's see the most important multi-qubit gates. The section above shows that an important property that the quantum computation offers is the quantum entanglement. In order to create entangled states we need to define a gate whose action can not be written as a tensorial product of single-qubit operations. The most important example of such a gate is the CX gate also called *Controlled-NOT* gate. Such gate acts on two qubits, and it applies the X gate on the second qubit only if the first one (called control qubit) is in the state $|1\rangle$. It can be defined by the following sum of tensorial products:

$$\begin{aligned} CX &= |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes X \\ &= |0\rangle\langle 0| \otimes (|0\rangle\langle 0| + |1\rangle\langle 1|) + |1\rangle\langle 1| \otimes (|1\rangle\langle 0| + |0\rangle\langle 1|) \\ &= |00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 10| + |10\rangle\langle 11| \end{aligned} \tag{2.73}$$

The correspondent matrix representation of the CX gate will be

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.74}$$

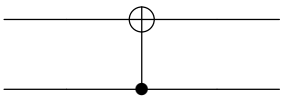
One can easily check with the following example that this gate creates entanglement starting from an original factorized state: let's build a separable two-qubit state

$$\frac{1}{\sqrt{2}}\,(|0\rangle + |1\rangle) \otimes |0\rangle \tag{2.75}$$

and let's apply to it a CX gate controlled by the qubit in superposition.

$$\begin{aligned} & CX\left(\frac{1}{\sqrt{2}}\left(|0\rangle+|1\rangle\right)\otimes|0\rangle\right) \\ &= CX\left(\frac{1}{\sqrt{2}}\left(|00\rangle+|10\rangle\right)\right) \\ &= \frac{1}{\sqrt{2}}\Big(|00\rangle+|11\rangle\Big) \end{aligned} \tag{2.76}$$

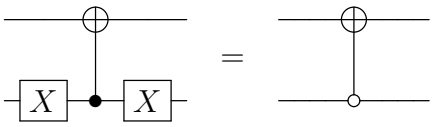
The output state is the Bell state $|\Phi^+\rangle$ defied in 2.56 and as we know it is not expressible by a tensor product between two single-qubit states. Since CX is its own inverse, it can also take an entangled state to a factorized one. CX gate is represented in the graph of the quantum circuits as follows:



The symbol $\bigoplus$ identifies the line of the target qubit, while the black dot corresponds to the control qubit.
There exists another version of the *Controlled-NOT* such that the gate X is applied on the target qubit if the control one is in $|0\rangle$ and not in $|1\rangle$. Since

$$\begin{aligned} & (X\otimes\mathbb{1})\left(|0\rangle\langle 0|\otimes\mathbb{1}+|1\rangle\langle 1|\otimes X\right)(X\otimes\mathbb{1}) \\ =& X|0\rangle\langle 0|X\otimes\mathbb{1}+X|1\rangle\langle 1|X\otimes X \\ =& |1\rangle\langle 1|\otimes\mathbb{1}+|0\rangle\langle 0|\otimes X \end{aligned} \tag{2.77}$$

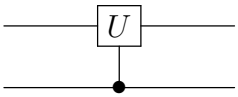
that corresponds to the following circuit equality



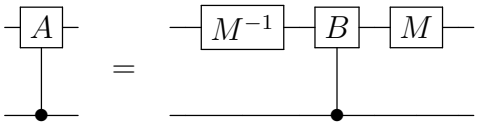
the resulting gate act such that The CX gate is only a particular case of a family of gates where a generic gate U is applied on a target qubit when the control qubit is in state $|1\rangle$. These gates are called CU or *Controlled-U* gates, and they are written as:

$$CU = |0\rangle\langle 0|\otimes\mathbb{1}+|1\rangle\langle 1|\otimes U \tag{2.78}$$

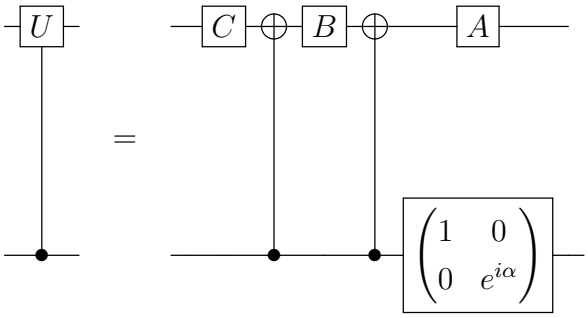
The graphic representation of the *Controlled-U* gate in a quantum circuit is the following:



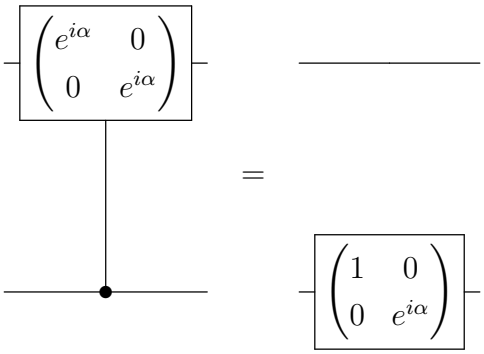
If we consider two Controlled-U gates A and B such that $A = M^{-1}BM$, then the following equality holds:



Since $U = e^{i\alpha}AXBXC$ from the Corollary **1** we obtain that

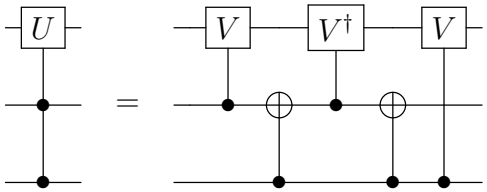


That because



Let's consider now the three-qubit gates. The first gate of such type that must be introduced is the CCX gate or *Toffoli* gate. The *Toffoli* gate can be decompose in CX and single-qubit gate, in particular H, T, $T^\dagger$ and S (see [46]). The *Toffoli* gate is not necessary to compute a general CCU gate. The following theorem shows how to decompose a general CCU without *Toffoli* gates

Theorem 2.3.2. *Suppose U is a unitary gate on a single qubit. Than*



Where V is any unitary operator satisfying $V^2 = U$.

2.3.3 The universal quantum computer

The definition of universality for quantum computers is summarized as follows:

Definition 1. *The universality for quantum computers is the ability to achieve any desired unitary operator on any arbitrary number of qubits [47].*

David P. DiVincenzo in the paper [48] defined five requirements for the realization of an universal quantum computer:

- *A scalable physical system with well characterized qubits*
- *The ability to initialize the state of the qubits to a simple fiducial state, such as* $|000\ldots\rangle$
- *Long relevant decoherence times, much longer than the gate operation time*
- *A universal set of quantum gates*
- *A qubit-specific measurement capability*

The meaning of 'characterized qubits' can be summarize as follows: a qubit is a quantum two-level system but if the qubit has additional energy levels, the control system of the quantum computer should be designed so that the its probability to go in those states is sufficiently small to allow the computation. The ability to initialize the state of the qubits is a computing requirement that the registers should be initialized to a known value before to start the computation.
We will talk about the third requirement when we will introduce the decoherence concept. Instead, the meaning of the last requirement can be explained as follows [48]. If a qubit's density matrix is

$$\rho = p\,|0\rangle\langle 0| + (1-p)\,|1\rangle\langle 1| + \alpha\,|0\rangle\langle 1| + \alpha^*\,|1\rangle\langle 0| \tag{2.79}$$

the measurement should give outcome $|0\rangle$ with probability p and $|1\rangle$ with probability $1-p$ independent of α and of any other parameters of the system, including the state of nearby qubits, and without changing the state of the rest of the quantum computer. Let's now introduce the concept of universal set of basis gates. Before that we must specify some definitions. The first one is the definition of a *dense subgroup* of the unitary group $U(N)$ [49]

Definition 2 (Dense subgroup of the unitary group $U(N)$)**.** *A sub group $G \subset U(N)$ is everywhere dense if for every $u \in U(N)$ and for every $\varepsilon > 0$ there exists a $g_\varepsilon \in G$ such that the distance between g_ε and u is less than ε.*

There are several ways to define the distance on an operator group, but for the purposes of this definition, they are equivalent. The distance most often used in quantum computing literature is the *trace distance* and is defined on $U(N)$ as

$$dist(U,V) = \sqrt{\frac{N - |Tr(UV^\dagger)|}{N}} \qquad U,V \in U(N) \tag{2.80}$$

Now we can define a universal set of basis gates as follows [49]

Definition 3. *A finite set of quantum gates forms a pure universal quantum basis in n-qubit space if they generate an everywhere dense subgroup of $U(2^n)$.*

Let's now consider a specific realization such as that of the quantum computer hardware used in this Thesis. A set of basis gates is given by the gates $U_3(\theta,\phi,\lambda)$ and CX:

$$U_3(\theta,\phi,\lambda) = \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\lambda+\phi)}\cos\frac{\theta}{2} \end{pmatrix} \qquad CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.81}$$

The universality of such a set of gates is given by the following Theorem [49]:

Theorem 2.3.3. *The circuit group generated by CX and all single-qubit unitary operators is purely universal in the multi-qubit space.*

Indeed the set of all operators of kind $U_3(\theta,\phi,\lambda)$ is a dense subgroup of $U(2)$. Compiling unitary generative using a set of single-qubit and 2-qubit quantum gates is a problem addressed primarily by Kitaev and Solovay which led to a theorem that bears their name. The Solovay-Kitaev theorem [50] is a crucial result in quantum computation and a breakthrough in the quantum compilation problem. Robert M. Solovay first announced the results in 1995,

but they were formalized and published independently a few years later by Alexei Y. Kitaev in 1997 on a review paper, including an algorithm to quickly approximate quantum gates [51]. The theorem roughly states that if we consider any quantum algorithm i.e. a unitary transformation $U \in SU(d)$, it is possible to find very quickly an approximation of the sequence of gates as long as they belong to a suitable set B. Such a set needs to satisfy some requirements to be exploited:

- All the gates in B need to be a unitary matrix with determinant 1;
- $\forall A_j \in B$, the inverse operation $A_j^\dagger \in B$;
- B must be a universal set, i.e. it is possible to approximate any unitary operation as a finite sequence of gates from the set.

It is worth highlighting that the second request is not strictly necessary to approximate unitary transformations, but it is a condition required to proof the theorem only [52]. However, if the requirements are met, the theorem holds for every desired accuracy ϵ, and the length of the resulting sequence scales efficiently.

Theorem 2.3.4. ***(Solovay-Kitaev)*** *Let be $\epsilon > 0$ a desired fixed accuracy, then $\forall U \in SU(d)$ exists a finite circuit C of gates driven by a set B such that the distance $d(U, C) < \epsilon$. The sequence S of gates has a length $\mathcal{O}(log^c(\frac{1}{\epsilon}))$, where c is a constant value. The circuit depth returned by the theorem and the execution time of its implementations scales polylogarithmically, even if distinct formulations and proofs*

achieve different values of the constants [53], [54]. For instance, the Dawson-Nielsen formulation [50] provides sequences of length $\mathcal{O}(log^{3.97}(1/\epsilon))$ in a time of $\mathcal{O}(log^{2.71}(1/\epsilon))$, while in [55] those quantities scales as $\mathcal{O}(log^{3+\gamma}(1/\epsilon))$ and $\mathcal{O}(log^{3+\gamma}(1/\epsilon))$ respectively, where γ is a positive constant which can be set at will. Additional boosts in performance can be gained by introducing some constraints such as restricting B a to a diffusive set of gates [52] the space of unitary matrices efficiently or by employing ancilla qubits [55]. Geometrical proof [54] shows that despite of the strategy considered, no algorithm can return sequence using less than $\mathcal{O}(log(1/\epsilon))$ gates. Moreover, one critical issue that the theorem does not address is finding a universal set of gates, but fortunately, it can be proved that almost all sets of gates have such propriety [56], [57]. The Solovay-Kitaev theorem provides an elegant and efficient classical algorithm for compiling an arbitrary unitary transformation into a circuit of quantum gates,

balancing the sequence length, the pre-compilation time, and the execution time. However, algorithms based on the theorem do not represent the only potential strategies to approach the quantum compiling problem. An optimal quantum circuit for a general two-qubit gate requires at most 3 CNOT gates and 15 elementary one-qubit gates. In the case of a purely real unitary two-qubit gate transformation, the construction requires at most 2 CNOTs and 12 one-qubit gates [58]. Such a method, known as KAK decomposition, is used for instance by the IBM QX architectures to address two-qubit random $SU(4)$ transformations [59]. For instance, the Quantum Fast Circuit Optimizer (Qfactor) optimizes the distance between a sequence of unitary gates, and a target unitary matrix, using an analytic method based on the SVD operation. In the following, modern approaches are outlined

2.4 Elements of adiabatic quantum computing

Adiabatic Quantum Computing (AQC) employs the adiabatic theorem to guide entangled qubits through a time-dependent Hamiltonian, steering them toward a configuration representing a superposition of the global minima for a given classical optimization problem. Adiabatic quantum computation is a universal quantum computational paradigm based on the adiabatic theorem and equivalent to gate-based quantum computation [60]. In the text, the computers developed by the company D-Wave are referred to as Adiabatic Quantum Computers (AQC) for simplicity. However, in reality, the D-Wave hardware does not strictly adhere to the conditions of adiabaticity, and the system's Hamiltonian does not allow for universal quantum computation. D-Wave hardware is, therefore, an approximation of AQC and is special-purpose, naturally implementing the algorithm known as quantum annealing.
AQC facilitates Quantum Annealing (QA), akin to Simulated Annealing (SA), where a Quantum Annealing (QA) algorithm controls a quantum system by gradually reducing quantum fluctuations' amplitude. This approach aims to maintain the system close to its instantaneous ground state, much like the quasi-equilibrium state in SA, but using quantum tunneling instead of thermal hopping. The key elements for introducing adiabatic quantum computation are the spin-glass Hamiltonian and the adiabatic theorem.

2.4.1 Spin-glass Hamiltonian for AQC

This Section introduces the classical and quantum formulation of an Ising spin-glass model, which stands at the basis of AQC's functioning.

Consider a d-dimensional lattice Λ, where each site i is occupied by a spin variable s_i. The term spin variable refers to a two-state system. The Ising model is a statistical system whose behavior depends on the following Hamiltonian:

$$H = J \sum_{\langle ij \rangle} s_i s_j + h \sum_i s_i \ , \tag{2.82}$$

where s_i is a random spin variable that assumes values ± 1 on the N sites of the lattice, and the first summation runs on sites i and j such that the two sites are nearest neighbours. The Ising model was named after Ernst Ising, which was the first to study and characterize it in 1925 [61].

The Ising model can represent several different physical systems. For instance, consider a system of magnetic dipoles aligned along the same axis, with only two allowed orientations. In this case, the first term in Eq. 2.82 is a two-sites interaction which can produce an ordered ferromagnetic state (if $J < 0$). The second term represents the paramagnetic effect of a uniform external field. We may also consider a binary alloy of type AB. In this case, the spin variables indicate whether a certain site on the crystalline lattice is occupied by an atom of either type A or type B. Neighbors of the same type will contribute to the total energy with a term $+J$ due to reciprocal repulsion, while neighbors of different types will contribute with $-J$. Lastly, one could use the spin values to represent the presence $(+1)$ or absence (-1) of a molecule in a certain cell of a *lattice gas* (a useful model for modelling the critical behavior of a fluid system).

The Ising model can be generalized by allowing each spin-pair appearing in the first summation of Eq. 2.82 to interact with different values of J_{ij}. In the magnetic analogy, it means abandoning the hypothesis that spins are equally distributed across the lattice. Thus, certain couples will be more closely packed, interacting with higher values of the coupling J_{ij}. The model obtained is known as *spin-glass*.

We may also remove the finite range of the interaction, allowing distant spins to interact with each other. It means that the summation over the nearest neighbors appearing in Eq. 2.82 is now performed over any spin pair. If the quadratic couplings are distributed according to a Gaussian probability density, the system we have obtained is then called the *Sherrington and Kirkpatrick model of spin glasses* [62].

We can push the generalization further by considering a scenario where the external magnetic field is not uniform and can vary from site to site. This leads to the following Hamiltonian:

$$H = \sum_{i,j,i\neq j} J_{ij} s_i s_j + \sum_i h_i s_i \ , \tag{2.83}$$

where J and h now depend on the lattice site considered, and the summation is performed over any pair (i,j) such that $i \neq j$.

We can now abandon the idea that the spin-glass model is defined over a lattice. Since interactions have infinite range, there is no need to think of spin variables as bounded in a certain spatial location. It is far more useful to think of the model as a graph, where each spin variable possesses connections with many (potentially all) other spin variables. A coupling $J_{ij} = 0$ corresponds to a missing link in the graph. Nonetheless, the expression *site* will still be used in the present work, meaning a particular position in the graph.

We will call *spin configuration* any function

$$S : \Lambda \to \{\pm 1\} \tag{2.84}$$

which assigns spin up (+1) or down (-1) to each site in Λ. Let Ω be the set of all possible spin configurations. The cardinality of the set is $|\Omega| = 2^N$.

2.4.2 Quantum spin-glass model

A quantum analog of the spin-glass model can be defined by mapping each spin variable s_i to a two-state quantum system q_i:

$$\begin{aligned} s_i = +1 &\to |\psi\rangle_i = |+1\rangle_i \\ s_i = -1 &\to |\psi\rangle_i = |-1\rangle_i \end{aligned} \tag{2.85}$$

The system Hamiltonian can be redefined as follows:

$$H = \sum_{i\neq j} J_{ij} \sigma_i^z \sigma_j^z + \sum_i h_i \sigma_i^z \ , \tag{2.86}$$

where σ_i^z are Pauli-z matrices such that.

$$\begin{aligned} \sigma_i^z \, |\psi\rangle_{j\neq i} &= 0 \\ \sigma_i^z \, |+1\rangle_i &= |+1\rangle_i \\ \sigma_i^z \, |-1\rangle_i &= -\,|-1\rangle_i \ . \end{aligned} \tag{2.87}$$

In the Hamiltonian in Eq. 2.86, all the interaction terms act along the z-axis. For this reason, the eigenstates of the Hamiltonian correspond to the set Ω of the classical states of the classical spin-glass model. This means that if the system is initialized with a configuration in Ω, it does not evolve as time passes, i.e. there is no chance for the spins to flip. If we want to build a true analog of the classical spin-glass model, we should allow the configurations in Ω to relax and become a superposition of multiple classical states. In the classical spin-glass model spin-flip events are possible thanks to thermal effects. In the quantum system, one may introduce a new interaction term that acts along a different direction, like a transverse magnetic field that acts along the x-axis:

$$H = \sum_{i,j,i\neq j} J_{ij}\sigma_i^z\sigma_j^z + \sum_i h_i\sigma_i^z + g\sum_i \sigma_i^x \,, \tag{2.88}$$

with $g \in \mathbb{R}$. The last term in the above expression allows the mixing of the eigenstates of the Hamiltonian in Eq. 2.86. In a suggestive classical analogy, the quantum superposition of different configurations may corresponds to the statistical ensemble, while the transverse magnetic field emulates the presence of a non-zero temperature.

The effect of the transverse field depends on the magnitude of g. If $g \to 0$, the effect vanishes and the eigenstates become increasingly similar to the classical states. If $g \to \infty$, the effect amplifies and the total Hamiltonian H in Eq. 2.88 becomes $H \sim g\sum_i \sigma_i^x$. In this case, the eigenstates of the Hamiltonian correspond to the eigenstates of σ^x, which means each spin has an equal probability to be in the $+1$ or -1 state.

2.4.3 Adiabatic theorem and convergence conditions of Quantum Annealing

In this Section, the principles of QA are introduced in mathematical terms. The main references for this Section are [63] and [64]. First, the proof of the adiabatic theorem is reviewed. Then, the spin-glass model with a transverse field is introduced as an example of QA implementation. Finally, the convergence condition for QA is presented.

Adiabatic theorem

Let's consider a general Hamiltonian which depends on time t only through the dimensionless time $s = t/\tau$, where τ is a characteristic time scale of the system

$$H(t) = \widetilde{H}\left(\frac{t}{\tau}\right) \equiv \widetilde{H}(s) \ . \tag{2.89}$$

The parameter τ is introduced to control the rate of change in the Hamiltonian. By varying the dimensionless time s, we can analyze how the system evolves and responds to the changes in the Hamiltonian in a more manageable way. In quantum systems the state vector $|\phi(t)\rangle$ follows the real-time Schrödinger equation,

$$i\frac{d}{dt}|\phi(t)\rangle = H(t)\,|\phi(t)\rangle \ , \tag{2.90}$$

where we set $\hbar = 1$. In terms of the dimensionless time we get:

$$i\frac{d}{ds}\left|\widetilde{\phi}(s)\right\rangle = \tau\widetilde{H}(s)\left|\widetilde{\phi}(s)\right\rangle \ . \tag{2.91}$$

Assuming the initial state of the system at $s = 0$ as the ground state of the initial Hamiltonian $\widetilde{H}(0) = H(0)$ and that the ground state of $\widetilde{H}(s)$ is not degenerate for $s \geq 0$. The objective is to obtain a sufficient condition that allows to evolve the $\widetilde{H}(s)$ in time in such a way that, at any s, the state of the system corresponds to the instantaneous eigenstate $|0(s)\rangle$ corresponding to the lowest eigenvalue at that time. This is usually called adiabatic evolution. We will now present the adiabatic theorem, which provides a sufficient condition to adiabatically evolve the system. To keep track of the distance between the state vector and the ground state, it is natural to expand the state vector by the instantaneous eigenstates of $\widetilde{H}(s)$. First, we derive useful formulas for the eigenstates. The kth instantaneous eigenstate of $\widetilde{H}(s)$ is denoted as $|k(s)\rangle$:

$$\widetilde{H}(s)\,|k(s)\rangle = \varepsilon_k(s)\,|k(s)\rangle \ , \tag{2.92}$$

where $\varepsilon_k(s)$ is the eigenvalue of the state $|k(s)\rangle$ with respect to the Hamiltonian $\widetilde{H}(s)$. Let's assume that $|0(s)\rangle$ is the ground state of $\widetilde{H}(s)$ and that the eigenstates are orthonormal, which means that $\langle j(s)|k(s)\rangle = \delta_{jk}$. Let us differentiate Eq. 2.92 with respect to s, and then project the obtained expression onto the state $|j(s)\rangle$. We obtain

$$\left\langle j(s)\left|\frac{d}{ds}\right|k(s)\right\rangle = \frac{-1}{\varepsilon_j(s) - \varepsilon_k(s)}\left\langle j(s)\left|\frac{d\widetilde{H}(s)}{ds}\right|k(s)\right\rangle \ , \tag{2.93}$$

if $j \neq k$. In the case $j = k$ we can impose the following condition:

$$\left\langle k(s) \middle| \frac{d}{ds} \middle| k(s) \right\rangle = 0 \, . \tag{2.94}$$

Note that, if $j = k$, the calculation that brings from Eq. 2.92 to Eq. 2.93 does not produce any condition on the term appearing in Eq. 2.94. Nonetheless, we can demonstrate that the condition in Eq. 2.94 is always achievable by a time-dependent phase shift. Indeed, if we define $\left|\widetilde{k}(s)\right\rangle = e^{i\theta(s)} \left|k(s)\right\rangle$, we find

$$\left\langle \widetilde{k}(s) \middle| \frac{d}{ds} \middle| \widetilde{k}(s) \right\rangle = i\frac{d\theta}{ds} + \left\langle k(s) \middle| \frac{d}{ds} \middle| k(s) \right\rangle \, . \tag{2.95}$$

The second term on the right hand side is purely imaginary, because

$$\left\langle k(s) \middle| \frac{d}{ds} \middle| k(s) \right\rangle^* + \left\langle k(s) \middle| \frac{d}{ds} \middle| k(s) \right\rangle = \frac{d}{ds} \left\langle k(s) | k(s) \right\rangle = 0 \, . \tag{2.96}$$

Then, a proper tuning of the phase factor $\theta(s)$ suffices at making the sum of the right-handed terms of Eq. 2.95 equal to zero. Thus, condition Eq. 2.94 holds for the phase-tuned eigenstate $\left|\widetilde{k}(s)\right\rangle$ even if the original eigenstate $\left|k(s)\right\rangle$ does not satisfy it.

The following theorem holds [63]:

Theorem 2.4.1. *If the instantaneous ground state of the Hamiltonian $\widetilde{H}(s)$ is not degenerate for $s \geq 0$ and the initial state is the ground state at $s = 0$, i.e. $\left|\widetilde{\psi}(0)\right\rangle = \left|0(0)\right\rangle$, the state vector $\left|\widetilde{\psi}(s)\right\rangle$ has the asymptotic form in the limit of large τ as*

$$\left|\widetilde{\psi}(s)\right\rangle = \sum_j c_j(s) e^{-i\tau\phi_j(s)} \left|j(s)\right\rangle \, , \tag{2.97}$$

where

$$c_0(s) \approx 1 + \mathcal{O}(\tau^{-2}) \, , \tag{2.98}$$

and

$$c_{j\neq 0}(s) \approx \frac{i}{\tau}\left[A_j(0) - e^{i\tau\left[\phi_j(s) - \phi_0(s)\right]} A_j(s)\right] + \mathcal{O}(\tau^{-2}) \, , \tag{2.99}$$

where $\phi_j(s) \equiv \int_0^s ds' \varepsilon_j(s')$, $\Delta_j(s) \equiv \varepsilon_j(s) - \varepsilon_0(s)$, and

$$A_j(s) \equiv \frac{1}{\Delta_j(s)^2} \left\langle j(s) \middle| \frac{d\widetilde{H}(s)}{ds} \middle| 0(s) \right\rangle \tag{2.100}$$

We can conclude that the system evolves adiabatically if the right-hand side

of 2.99 is much smaller than unity. In such case, at any time s, the system has a low probability to occupy states different from the istantaneous ground state $|0(s)\rangle$. This condition can be rewritten as

$$\tau \gg |A_j(s)| \, . \tag{2.101}$$

Expression 2.101 implies that Adiabatic evolution is possible when τ is large, which means $\widetilde{H}(s)$ changes slowly in time. Using the original time variable t, the adiabaticity condition is further rewritten as

$$\frac{1}{\Delta_j(t)^2} \left| \left\langle j(t) \left| \frac{dH(t)}{dt} \right| 0(t) \right\rangle \right| = \delta \ll 1 \, , \tag{2.102}$$

which must hold for all times. This is the usual expression of the sufficient condition for adiabatic evolution.

Convergence conditions of quantum annealing

We now derive a condition which guarantees the convergence of QA. The problem consists of finding an anneal schedule (i.e. the time dependence of the control parameters) such that the adiabaticity condition (Eq. 2.102) is satisfied. We consider the transverse-field spin-glass model introduced in Section 2.4.2 since modern QA devices implement this Hamiltonian. Suppose one wants to solve an optimization problem that can be represented as the ground-state search of a spin-glass model of the general form

$$H_{\text{glass}} \equiv -\sum_{i=1}^{N} J_i \sigma_i^z - \sum_{ij} J_{ij} \sigma_i^z \sigma_j^z - \sum_{ijk} J_{ijk} \sigma_i^z \sigma_j^z \sigma_k^z - \ldots \, , \tag{2.103}$$

where the σ_i^z are the Pauli matrices that act along the z-direction. Many combinatorial optimization problems can be written in this form, by mapping binary variables to spin variables.

An important assumption is that Hamiltonian 2.103 is extensive, i.e. proportional to the number of spins N for large N. To realize QA, fictitious kinetic energy is typically introduced by the time-dependent transverse field

$$H_T \equiv -\sum_{i=1}^{N} \sigma_i^x \, . \tag{2.104}$$

The Pauli operators σ_i^x enable spin flips, quantum fluctuations, or quantum tunneling between the states that possess eigenvalues +1 and -1 with respect

to σ_i^z. Such effects allow a quantum search of the phase space. The total Hamiltonian takes the expression

$$\begin{aligned} H(t) &= -F(t)\left(\sum_{i,j} J_{ij}\sigma_i^z\sigma_j^z + \sum_i h_i\sigma_i^z\right) - G(t)\sum_i \sigma_i^x \\ &\equiv F(t)H_P + G(t)H_T \ , \end{aligned} \tag{2.105}$$

where t is the physical time, $F(t)$ and $G(t)$ are positive real numbers, H_P is the Hamiltonian whose ground state correspond to the solution of the optimization problem, and $H(t)$ is the transverse field Hamiltonian. The *problem Hamiltonian* H_P in Eq. 2.105 is a simplified version of the more general H_{glass}. The reason for this restriction comes from the fact that modern quantum annealers can only implement Hamiltonians with interaction terms that are at most quadratic. Nonetheless, the following deductions about convergence for QA algorithms hold for a generic H_{glass}.

Each eigenstate of $H(\tau)$ is a set of N binary values $S = \{s_1, s_2 ... s_N\}$, where N is the total number of spin degrees of freedom of the system. Each spin variable s_i can assume two different states, $+1$ and -1. There are 2^N different eigenstates of H_P, one for each possible combination of the N spin variables. A generic state of the system can be expressed as:

$$\begin{aligned} |\phi\rangle &= \sum_{\{s_1,...,s_N\}\in\Omega} \alpha\big(\{s_1,...,s_N\}\big)\,|s_1,...,s_N\rangle \ , \\ \text{with} \quad &\sum_{\{s_1,...,s_N\}\in\Omega} |\alpha\big(\{s_1,...,s_N\}\big)|^2 = 1 \end{aligned} \tag{2.106}$$

where the sums are over each possible configuration of the N spins.

Suppose we are interested in finding which of the eigenstates corresponds to the minimum energy of H_P. By choosing a correct annealing schedule for $F(t)$ and $G(t)$ (i.e. choosing their dependence on time), we can encourage the system to converge to the global minimum. At $t = 0$ we set $G(0) \gg F(0)$. This way, the initial ground state will be an eigenstate of Hamiltonian H_T, which means an equally probable superposition of all the classical states in the phase space Ω. As t grows, the system must be gradually forced into states that are a mixture of low-energy configurations with respect to Hamiltonian H_P, so we must raise $F(t)$ and lower $G(t)$. If the annealing schedule is sufficiently slow, the adiabatic theorem assures that the system will remain close to the lowest energy eigenstate of the instantaneous Hamiltonian $H(t)$. For $t \to \infty$, we impose $G(t) \ll F(t)$, so that the system finds itself in a superposition

dominated by the state corresponding to the spin configuration that minimizes H_P.

An important issue is how slowly we should modify $F(t)$ and $G(t)$ to keep the state vector arbitrarily close to the instantaneous ground state of total Hamiltonian 2.105. For simplicity, we suppose to fix $F(t)$ to a positive constant k for every time t. Expression 2.105 can then be rewritten as

$$H'(t) = H_P + \Gamma(t) H_T \tag{2.107}$$

where $H' = H/k$ and $\Gamma(t) = G(t)/k$. Note that modern quantum computers allows to control the time dependency of both F and G terms. Nonetheless, the evolution of the ratio $G(t)/F(t)$ is the driving force of quantum annealing. For this reason, what can be learned for $F(t)$ fixed is instructive and can then be extended to more general contexts. The following theorem provides a sufficient condition for convergence.

Theorem 2.4.2. *Imposing the adiabaticity condition in Eq. 2.102 on the transverse field spin-glass model in Eq. 2.107 yields the following sufficient condition of convergence for QA:*

$$\Gamma(t) = a(t\delta + c)^{-1/(2N-1)} \tag{2.108}$$

for $t > 0$. Here a and c are constants of order $\mathcal{O}(N^0)$ and δ is a parameter sufficiently small such that the adiabaticity condition in Eq. 2.102 holds.

A proof for this theorem can be found in reference [63].

Computational complexity

The power-law dependence on t in Eq. 2.108, sufficient to ensure convergence for QA, is much faster than the log-inverse law typical of SA, $T(t) = pN/\log(\alpha t + 1)$. However, we can not conclude that QA provides an algorithm to solve NP problems in polynomial time. Indeed, suppose we want $\Gamma(t)$ to reach a certain small value ε so that the system is close to the ground state of H_P. The time required to satisfy such condition is estimated from Eq. 2.108 as

$$t_f \approx \frac{1}{\delta}\left(\frac{1}{\varepsilon}\right)^{2N-1} . \tag{2.109}$$

This relation shows that the QA algorithm requires a time exponential in N to converge.

2.5 Hardware realization

2.5.1 Superconductive qubits

Superconducting circuits are the basis of superconducting qubits, as they are built using superconducting elements that behave quantum mechanically.
First of all let's introduce a basic quantum electronic circuit. The LC oscillator is the simplest example of a quantum integrated circuit. LC circuits consist of an inductor L connected to a capacitor C. For a circuit to behave quantum mechanically there must be no dissipation, that is, the circuit must have zero resistance at the operating temperature. This is essential to preserve quantum coherence.
The LC circuit obeys the equations of motion of the linear harmonic oscillator where the flux Φ in the inductor is analogous to the position coordinate, and the charge Q on the capacitor is analogous to the conjugate momentum. Therefore we define two operators $\hat{\Phi}$ and $\hat{Q}$ such that

$$\left[\hat{\Phi}, \hat{Q}\right] = i\hbar \tag{2.110}$$

Even if the circuit has a huge amount of electrons, the degrees of freedom have been reduced to one, the Cooper-pair fluid moving back and forth in the circuit [65]. The Hamiltonian of the circuit is

$$\hat{H} = \frac{\hat{\Phi}^2}{2L} + \frac{\hat{Q}^2}{2C} \tag{2.111}$$

The inductance L of the system has the role of the mass and the inverse of the capacitance $1/C$ of the spring constant. Being $\omega = 1/\sqrt{LC}$ the resonance frequency of the circuit then we can write

$$\hat{H} = \hbar\omega\left(\hat{a}^\dagger\hat{a} + \frac{1}{2}\right) \tag{2.112}$$

where

$$\begin{aligned} \hat{a}^\dagger &= -i\frac{1}{\sqrt{2C\hbar\omega}}\hat{Q} + \frac{1}{\sqrt{2L\hbar\omega}}\hat{\Phi} \\ \hat{a} &= i\frac{1}{\sqrt{2C\hbar\omega}}\hat{Q} + \frac{1}{\sqrt{2L\hbar\omega}}\hat{\Phi} \end{aligned} \tag{2.113}$$

In an harmonic oscillator the difference between two consecutive energy levels is always $\Delta E = \hbar\omega$. Therefore it is impossible to approximate it to a two level system using the lowest two levels.

We want to approximate our system to a two level qubit. The property of a system to be approximated in a two level system is often called 'non-linearity'. In a non linear system the energy level cannot be uniformly spaced. The only element which is both non-dissipative and non-linear is a Josephson junction. Josephson junctions is the essential element of superconducting qubits.

Josephson junction

Josephson junctions consist of two superconductors separated by an insulating barrier. The width of the barrier is of the order of nanometres. The origin of the non-linearity of the Josephson element is associated to the discreteness of charge that tunnels across the thin insulating barrier. Hence the junction is characterized by only one degree of freedom $N(t)$, the number of Cooper pairs having tunnelled across the barrier.
The variable that gives the number of Cooper-pairs across the junction N should be treated as a discrete operator, as we have said that the charge tunnelling across the barrier is discrete

$$\hat{N} = \sum_N N \left|N\right\rangle\!\left\langle N\right| \tag{2.114}$$

The two superconducting electrodes form a capacitor, but we will ignore the Coulomb energy that builds up as Cooper pairs tunnel from one side to the other. The tunnelling of electrons through the barrier couples the $|N\rangle$ states. The coupling energy is given by the Hamiltonian

$$\hat{H}_J = -\frac{E_J}{2} \sum_N \left[\left|N\right\rangle\!\left\langle N-1\right| + \left|N-1\right\rangle\!\left\langle N\right| \right] \tag{2.115}$$

Where E_J is a constant. The Hamiltonian can be written in terms of the phase difference of the Cooper pairs wave functions on the two sides of the junction. We introduce new basis states

$$|\varphi\rangle = \sum_N e^{iN\varphi} |N\rangle \tag{2.116}$$

Where $\varphi \to \varphi + 2\pi$ leaves the $|\phi\rangle$ unchanged. The txo operator $\hat{\varphi}$ and $\hat{N}$ are conjugate variables with

$$\left[\hat{\varphi}, \hat{N}\right] = i \tag{2.117}$$

Conversely, the $|N\rangle$ state is given by the Fourier transformation of the phase state

$$|N\rangle = \frac{1}{2\pi}\int_0^{2\pi} d\varphi e^{iN\varphi}\,|\varphi\rangle \tag{2.118}$$

We introduce the operator

$$e^{i\hat{\varphi}} = \frac{1}{2\pi}\int_0^{2\pi} d\varphi e^{iN\varphi}\,|\varphi\rangle\langle\varphi| \tag{2.119}$$

Which acts on $|N\rangle$ as

$$e^{i\hat{\varphi}}\,|N\rangle = |N-1\rangle \tag{2.120}$$

We can obtain the expression for the coupling Hamiltonian 2.115 in the basis of $|\varphi\rangle$.

$$\hat{H}_J = -\frac{E_J}{2}\left(e^{i\hat{\varphi}} + e^{-i\hat{\varphi}}\right) = -E_J\cos\hat{\varphi} \tag{2.121}$$

Charge qubit

The charge qubit, known as Cooper-pair box, consists of a small superconducting island located between the insulating barrier of a Josephson junction and one of the plates of a capacitor (Fig). In this type of qubit the charge is the controlled variable, controlled by the voltage source, inducing a charge difference between both sides of the Josephson junction.

The Hamiltonian of the circuit is

$$\hat{H} = E_C\left(\hat{N} - N_g\right)^2 - E_J\cos\hat{\varphi} \tag{2.122}$$

The first term is the electrostatic energy (Coulomb energy) operator

$$\hat{U} = E_C\left(\hat{N} - N_g\right)^2 = E_C\sum_N (N - N_g)^2\,|N\rangle\langle N| \tag{2.123}$$

where $N_g = C_g\frac{V_g}{2e}$ is the polarization charge induced by the voltage V_g on the gate capacitor of capacitance C_g while $E_C = \frac{(2e)^2}{2(C_J+C_g)}$ is the electrostatic Coulomb energy, the energy used to store a Cooper pair in the capacitor.
The charge qubit is in the charge regime, $E_C \gg E_J$. $\hat{N}$ is a discreet variable representing the quantity of Cooper pairs on the island which can only take integer values. The offset charge N_g is, on the other side, a continuous variable. The charge states are given by the excess of Cooper pairs that have tunnelled to the island. For a qubit, all the states over $|2\rangle$ can be ignored when the offset charge N_g is about the same charge as an electron, $N_g = 1/2$, because of a great energy difference between the first and second states. Thus, we can make a two

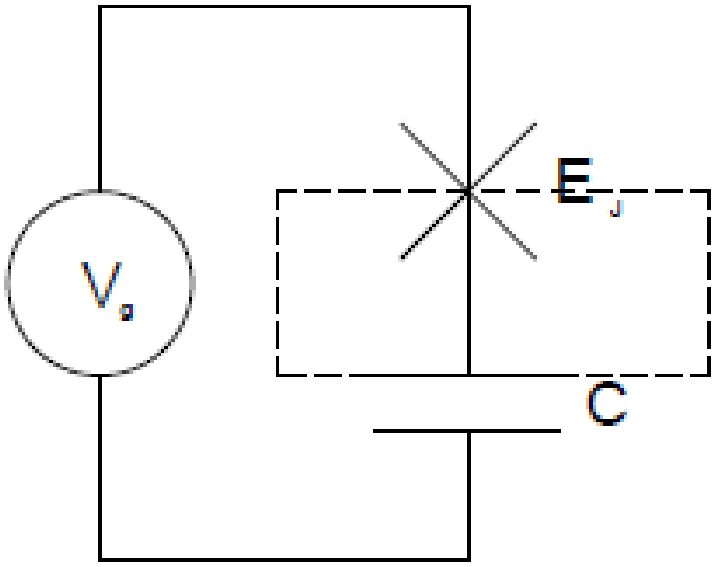


FIGURE 2.2: Circuit representation of a charge qubit [66]. The region which lies inside the dashed lines is the Cooper-pair box.

level approximation.

The qubit basis states are $|0\rangle$ and $|1\rangle$, the first corresponding to the lack of Cooper pairs in the superconducting island and the second corresponding to the presence of a single Cooper pair. When $N_g = 1/2$, the energy is the same for both charge states $|0\rangle$ and $|1\rangle$ leading to degeneracy. For that reason an energy splitting occurs in that region and the energy eigenstates become linear combinations of the charge states.

$$|\pm\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle \pm |1\rangle\Big) \tag{2.124}$$

Making the two levels approximation, we can replace the Hamiltonian in the Equation 2.122 in the qubit reduced Hamiltonian

$$\hat{H}_{qubit} = E_C \begin{pmatrix} N_g^2 & 0 \\ 0 & (1-N_g)^2 \end{pmatrix} - \frac{E_J}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.125}$$

If we change the zero of energy of the system to $E_0 = E_C(1/2 - N_g)^2 + 1/4$ (Ref. [67]) then

$$\hat{H}_{qubit} = -\frac{1}{2}\Big(E\hat{\sigma}_z + E_J\hat{\sigma}_x\Big) \tag{2.126}$$

where $E = E_C(1 - 2N_g)$. One can thus make a correspondence between the Cooper pair box and a spin 1/2 in a magnetic field using the Pauli spin matrices:

$$\hat{\sigma}_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \hat{\sigma}_y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix} \qquad \hat{\sigma}_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.127}$$

The Hamiltonian in the Equation 2.126 takes the form

$$\hat{H}_{qubit} = -\vec{\hat{s}} \cdot \vec{h} \tag{2.128}$$

where $\vec{\hat{s}} = \frac{1}{2}\vec{\hat{\sigma}} = \frac{1}{2}(\hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z)$ is the spin operator and $\vec{h}$ is an effective magnetic field whose components in the bass (x, y, z) are $(E, 0, E_J)$.

The qubit in the IBM's quantum computer are a kind of charge qubit called

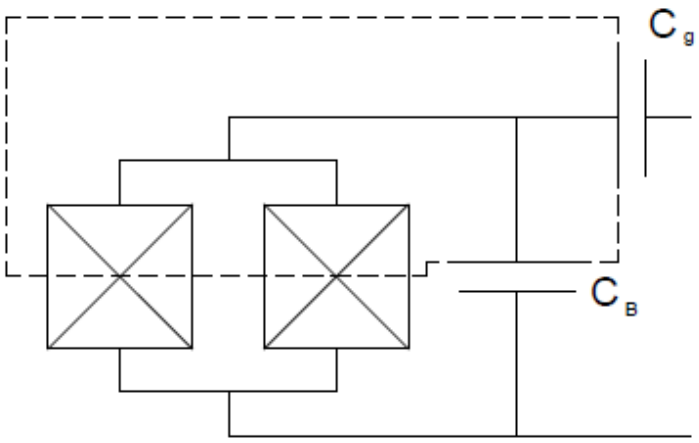


FIGURE 2.3: Circuit representation of a transmon qubit [66]. The region which lies inside the dashed lines is the Cooper-pair box

transmon.

The transmon consists of two superconducting islands coupled through two Josephson junctions which operates in the flux regime $E_J \gg E_C$, normally $E_J/E_C \sim 50$.

The difference between a transmon as from Figure 2.3 and the charge qubit in the Figure 2.2 is that transmons have an additional capacitance, meaning that $E_C = (2e)^2/(C_J + C_B + C_g)$. Such architecture is characterized by a higher decoherence times. The decoherence times will be introduced in the next Section

2.5.2 Trapped-ion qubits

Even an atom can be identifiable as a qubit, obviously under the necessary conditions. Charged atoms (ions) can be contained (that is, trapped) in one precise location using electric fields. It is possible to contain neutral atoms using optical tweezers, but our focus is on ions, which can be contained using an electromagnetic trap. In 1953 Wolfgang Paul proposed his now-called Paul trap and, together with Dehmelt, they awarded the 1989 Physics Nobel Prize, Current trapped ion quantum computers extensively use the Paul trap. Since, using the laws of electrostatics, one can prove that it is impossible to create a confining potential with only static electric fields, Paul and Dehmelt propose time-dependent saddle-shaped potentials. In detail, the Paul trap is a potential composed of a combination of an oscillating potential (referred to as radio-frequency RF) and a static one (DC potential) [68]. The potential Φ is

$$\Phi(\vec{r}, t) = \Phi_{DC}(\vec{r}) + \Phi_{RF}(\vec{r}, t) \tag{2.129}$$

where the DC potential

$$\Phi_{DC}(\vec{r}) = \frac{1}{2}\left(u_x x^2 + u_y y^2 + u_z z^2\right) \tag{2.130}$$

and the RF potential is

$$\Phi_{RF}(\vec{r}, t) = \frac{1}{2}\left(v_x x^2 + v_y y^2 + v_z z^2\right)\cos\left(\Omega_{RF} t + \phi\right) \tag{2.131}$$

The vectors u_i, v_i, ϕ and the frequency Ω_{RF} need to be adjusted to the charge and mass of the ion. The technology to trap many ions and put them close together in a one-dimensional array is called an ion chain. To make a quantum computer with ions it is necessary to place the ions in a sufficiently spaced one-dimensional array and cool them all down to the point where their motion in space is quantized. In this scenario, photons that would bring the ion to their excited states will not cause any relative motion. Instead, all ions will recoil together. This phenomenon is called the Mossbauer effect [36]. However, only a select few isotopes meet the requirements, such as the effect described above, to be able to make qubits. The reason is that our qubit basis states are the ground and excited states of an electron in the atom, and we need to be able to transition between them using laser light. There is a need for ions with an excited state that is long-lived, and also one that can be manipulated using frequencies that lasers can produce. The best ions for our purposes are single-charged ions in Group II of the periodic table, such as Calcium-40, Beryllium-9,

and Barium-138, commonly used in university laboratories [69]. The rare earth Ytterbium-171 is used by IonQ and Honeywell. These elements have two valence electrons, but their ionized version only has one. The valence electron is not so tightly bound to the atom, so it is the one whose state is used to represent a qubit.

Qubit encoding on a trapped ion

To simplify, we consider the energy levels of the hydrogen atom which has quantum states identifiable with 3 quantum numbers $|n, l, m\rangle$ where n indicates the energy level while the other quantitative numbers indicate the degenerate states with total angular momentum l and angular momentum along a preferred direction (z) indicator with m. Given a value l we have that $m \in \{-l, -l+1, \ldots, l-1, l\}$. However, we need to broaden this discussion to introduce the spin of the electron. For an electron the total spin is always $1/2$ so the quantum states will be $|n, l, m, s = \pm 1/2\rangle$ given that s, the component along z of the spin, has the same behavior as m. For a general atom, different values of l do not correspond to degenerate states but to different energy levels. For this reason, at a given point, the energy levels are ordered based on the value of l and therefore we have, for example, the states, called orbitals, S, P, D with angular moment $l = 0, 1, 2$ respectively. The states can be rewritten so as to consider the angular momentum and the spin as a single observable, the total angular momentum J. In this way we will have the states $S_{1/2}$, $P_{1/2}$, $P_{3/2}$, $D_{3/2}$ and $D_{5/2}$. If one considers the calcium isotope ion $^{40}Ca^{+}$, the Hilber subspaces generated by the basic elements are considered which, in energy level order, are $S_{1/2}$, $D_{3/2}$, $D_{5/2}$, $P_{1/2}$ and $P_{3/2}$ where however the D orbitals correspond to $n-1$. The procedure for encoding a qubit consists of identifying a ground state $|g\rangle$ and an excited state $|e\rangle$ with energies that are E_0 and E_1 respectively. After many simplifications involving some approximations, we find that the Hamiltonian that describes the two energy levels of the ion resonant to the laser light is given by the operator

$$H = E_0 |g\rangle\langle g| + E_1 |e\rangle\langle e| + \frac{\hbar\Omega}{2}\left(ae^{i\varphi} + a^\dagger e^{-i\varphi}\right) + H_1(t) \tag{2.132}$$

where

$$a = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \qquad a^\dagger = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{2.133}$$

Where Ω is called Rabi frequency and it depends on the applied magnetic field due to the laser, and by the magnetic moment of the ion. The Rabi

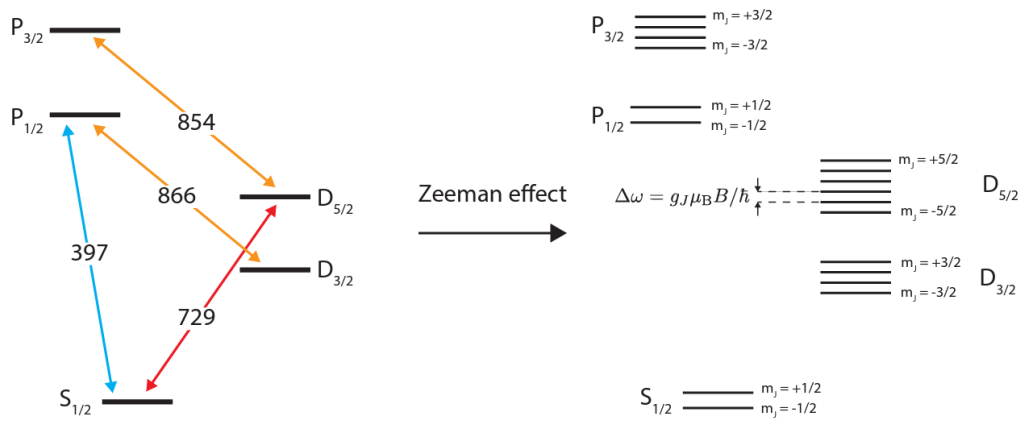


FIGURE 2.4: $^{40}Ca^{+}$ energy levels involved to engineer a left-hand optical qubit. Meter on the right are shown the energy levels in the presence of an external magnetic field (Zeeman effect) where it is possible to implement a hyperfine qubit.

oscillations drive the exchange between the state $|g\rangle$ and $|e\rangle$. The Hamiltonian $H_1(t)$ is due to the potential in Equation 2.129 and it can be ignored in such discussion. Now there are two ways to encode a qubit. The first one is called, the optical qubit. In such case, always taking the ion $^{40}Ca^{+}$ into consideration, an infrared laser couples the two levels $|g\rangle = \left|S_{1/2}, m = -1/2\right\rangle$ and $|e\rangle = \left|D_{5/2}, m = -1/2\right\rangle$ that correspond, in computational base, to $|0\rangle$ and $|1\rangle$. The exited state $\left|D_{5/2}, m = -1/2\right\rangle$ respect the channel $\left|S_{1/2}, m = -1/2\right\rangle \leftrightarrow \left|D_{5/2}, m = -1/2\right\rangle$ has a lifetime of 1.17×10^9ns $\sim$ 1s which means that in $\sim$ 1s a spontaneous emission of a photon will bring the electron back to $\left|S_{1/2}, m = -1/2\right\rangle$ starting from $\left|D_{5/2}, m = -1/2\right\rangle$. In Figure 2.4 the energy levels of the orbitals involved are graphically represented. Another, more common, type of qubit encoded on atomic states is called hyperfine qubit. In this case, the Zeeman effect is exploited. This effect concerns the splitting of atomic levels due to the presence of an external magnetic field. The external magic field introduces a Hamiltonian term equal to

$$H_B = -\vec{\mu} \cdot \vec{B} \quad \text{where } \vec{\mu} = -\frac{\mu_b}{\hbar}\left(g_L \vec{L} + g_S \vec{S}\right) \tag{2.134}$$

The terms g_L and g_S are 1 and ~ 2 respectively. The term H_B produces a splitting of the energy levels since it depends, approximately, on the total angular momentum $\vec{J}$ as shown in Figure 2.4. Therefore the states $\left|S_{1/2}, m = -1/2\right\rangle$ and $\left|S_{1/2}, m = +1/2\right\rangle$ will be separated by an energy difference equal to $\Delta E = g\mu_B B$ where is the Lande factor and it depends by the angular momentum and the spin of the electron orbital. Therefore the hyperfine qubit is built on the states

$\left|S_{1/2}, m = -1/2\right\rangle$ and $\left|S_{1/2}, m = +1/2\right\rangle$ that are separated by a small energy difference if the magnetic field in small. The exited state $\left|S_{1/2}, m = +1/2\right\rangle$ is called hyperfine state and it is a metastable state with a lifetime of almost infinite. In a nutshell, the selection rules for a certain type of atomic level transitions said that transitions of $\Delta l = 0$ are not permitted. Therefore the two states in the hyperfine qubit are coupled via an auxiliary state (generally with a short lifetime) with a higher energy. So to decay into the ground state, these types of qubits must transition from an auxiliary state and such a transition is generally unlikely at low temperatures. This makes these types of qubits very stable. However, this structure leads to a complication when introducing unitary transformations of the qubit which in fact must be very precise.

Gates on Trapped ion qubits

According to Schrödinger's equation, the Rabi oscillations defined in 2.132, produce a time evolution on the ground state $|g\rangle$

$$|\psi(t)\rangle = \exp\left\{-it\frac{\Omega}{2}\left(ae^{i\varphi} + a^{\dagger}e^{-i\varphi}\right)\right\}|g\rangle \equiv U(\Omega, \varphi, t)\,|g\rangle \tag{2.135}$$

therefore, since

$$U(\Omega, \varphi, t)\,|g\rangle = \cos\left(\frac{\Omega t}{2}\right)|g\rangle - i\sin\left(\frac{\Omega t}{2}\right)e^{i\varphi}\,|e\rangle \tag{2.136}$$

and

$$U(\Omega, \varphi, t)\,|e\rangle = -i\sin\left(\frac{\Omega t}{2}\right)e^{-i\varphi}\,|g\rangle + \cos\left(\frac{\Omega t}{2}\right)|e\rangle \tag{2.137}$$

than $U(\Omega, \varphi, t)$ is a parametrized unitary transformation on a single qubit and, therefore, it is possible to implement any element of $U(2)$ with the identity.

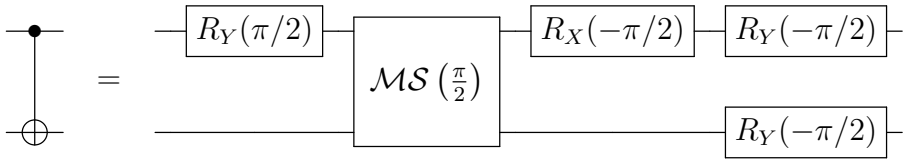
$$U(\Omega, \varphi, t) = \begin{pmatrix} \cos\left(\frac{\Omega t}{2}\right) & -i\sin\left(\frac{\Omega t}{2}\right)e^{-i\varphi} \\ -i\sin\left(\frac{\Omega t}{2}\right)e^{i\varphi} & \cos\left(\frac{\Omega t}{2}\right) \end{pmatrix} \tag{2.138}$$

As for two-qubit transformations, an ingenious gate, known as the Mølmer-Sørensen gate is often used. In this case, 2 ions are involved which are brought together mechanically and one auxiliary. There is, also in such case, a Rabi frequency Ω_{MS}, of two simultaneous laser pulses, associated with this evolution. adjusting the time and the phase of the lasers can lead to a superposition of $|g\rangle\,|g\rangle\,|n\rangle$ and $|e\rangle\,|e\rangle\,|n\rangle$ which leads to, for example, in a two-qubit state $(|g\rangle\,|g\rangle + |e\rangle\,|e\rangle)/\sqrt{2}$. Using Schrödinger's equation allows us to derive how the qubits evolve when we apply the Mølmer-Sørensen protocol for a time t. The

gate has the following form

$$U_{MS}(t) = \begin{pmatrix} \cos\left(\frac{\Omega_{MS}t}{2}\right) & 0 & 0 & -i\sin\left(\frac{\Omega_{MS}t}{2}\right) \\ 0 & \cos\left(\frac{\Omega_{MS}t}{2}\right) & -i\sin\left(\frac{\Omega_{MS}t}{2}\right) & 0 \\ 0 & -i\sin\left(\frac{\Omega_{MS}t}{2}\right) & \cos\left(\frac{\Omega_{MS}t}{2}\right) & 0 \\ -i\sin\left(\frac{\Omega_{MS}t}{2}\right) & 0 & 0 & \cos\left(\frac{\Omega_{MS}t}{2}\right) \end{pmatrix} \tag{2.139}$$

Since the CNOT gate is commonly used in quantum algorithms, let's define how to obtain it from the Mølmer-Sørensen gate. It is possible to do so by using a combination of single-qubit rotations and the Mølmer-Sørensen gate applied for a period of $t = \pi/2\Omega_{MS}$



Therefore, having defined a universal gate basis, trapped ion technology follows the DiVincenzo criteria and is suitable for performing quantum computing.

2.6 Limitations in today's quantum computers

From DiVincenzo's criteria and the postulates of quantum mechanics, it can be deduced that the qubit system in a quantum computer must be closed and therefore not interact with the outside. In general, this prerequisite cannot be met in practice. For this reason, error correction methods have been developed that aim to detect and correct the error rather than prevent it. These error correction methods consist of producing qubits called logical qubits by aggregating clusters of qubits together, in this way, with intermediate measurements on auxiliary qubits, it is possible to detect and correct errors on the qubit register. To date, these error correction methods are difficult to implement because, in addition to the need to have many qubits, the implemented quantum gates bring with them an error rate as well as the measurement operations. This leads to problems in detecting and then correcting errors using error correction algorithms. Quantum computers that implement logical qubits are called fault-tolerant while those today that do not implement it are called Noisy intermediate-scale quantum (NISQ) devices [70].

2.6.1 Decoherence times

The Schrödinger equation governs the dynamics of a closed quantum system and it leads to a time evolution of the quantum states described by a unitary operator. However, the interaction with the environment induces non-unitary dynamics. We refer to a non-unitary dynamic by calling it dissipative dynamics and this leads to the phenomenon of decoherence and therefore to the loss of information contained in the qubits. The effects of closed and open quantum systems both follow the master equation in the Lindblad form [71].

$$\partial_t \rho = \frac{1}{i\hbar}[H, \rho] + \sum_k \gamma_k \left(L_k \rho L_k^\dagger - \frac{1}{2} L_k^\dagger L_k \rho - \frac{1}{2} \rho L_k^\dagger L_k \right) \tag{2.140}$$

The dissipative (non-unitary) evolution is described by the jump operators L_k with their associated weight γ_k. The different k interactions with the environment are called channels (in general channels are the term used to refer to the operators that act on the density matrix). For typical error channels, the decoherence time scale is $t_k = 1/\gamma_k$. For simplicity, in the following description of the decoherence, only two channels are considered: $L_1 = \sigma_x$ with time T_1 and $L_2 = \sigma_x$ with time T_2. Indeed σ_x leads to qubit decay on the ground state, for example, Spontaneous emission, while σ_x leads to the qubit dephasing. Let's take a step back to give the definition of a coherent quantum state. A coherent state is a quantum state that maintains its relatively complex phase during a period of time [66]. The decoherence is the loss of coherence. Coherence is lost when the system acquires phases from the surrounding environment and no longer maintains its complex relative phase. Coherence can be described by a system coupled to a heat bath. Imagine the qubit is in a pure state, with density matrix $\rho = |g\rangle\langle g|$ in time $t = 0$. At $t > 0$ the system evolves coupled to a reservoir at temperature T. The state is not pure anymore, as it mixes with the first excited state. The stationary state of the system is given by

$$\rho = \frac{1}{1 + e^{-\beta\omega}} |g\rangle\langle g| + \frac{e^{-\beta\omega}}{1 + e^{-\beta\omega}} |e\rangle\langle e| \tag{2.141}$$

where $|e\rangle$ is the first excited state.
It is important that a qubit remains in a coherent superposition of states, that is necessary for quantum computation. The ideal qubit is unperturbed by its environment but still accessible to be controlled and measured. Quantum devices have associated decoherence times, which limit the number of quantum operations that can be performed before the results are affected by noise.
One can distinguish between two measures of decoherence times (Ref. [72]):

- T_1 is the "longitudinal coherence time" (also known as "amplitude damping")
- T_2 is the "transverse coherence time" (also known as "phase damping").

T_1 is the time required by a qubit to relax from the first excited state to the ground state, that is, the decay time. T_2 is the average time in which the energy-level splitting remains unchanged, in other words the time it takes for the phase difference between two eigenstates to become random, the dephasing time. One way to estimate T_1 is to initialize a qubit to the ground state $|0\rangle$. One applies an X gate to turn it into $|1\rangle$, and measure it in the computational basis after a time t. The probability of the qubit of staying in the $|1\rangle$ state is expected to follow an exponential decay curve e^{-t/T_1}. To experimentally determine T_2, one can initialise a qubit to the ground state $|0\rangle$, apply a Hadamard transform H to change it into $(|0\rangle + |1\rangle)/\sqrt{2}$, and wait for a time t before applying another transform H and measuring the qubit on the computational basis. The decrease of the probability of obtaining a $|0\rangle$ measurement during time should follow the expression $\frac{e^{-t/T_2}+1}{2}$. For what concerns the causes of decoherence, the decohering elements can be extrinsic and intrinsic. The extrinsic decoherence is caused by the already mentioned coupling to the environment, and the solution will be isolating our system as much as possible (as it must be still readable). Intrinsic decoherence is caused by noise coming from the superconducting circuit. The temperature at which the qubits operate is of the order of millikelvin, meaning that the temperature is not a cause of decoherence. The noise coming from the superconducting circuit is more relevant.

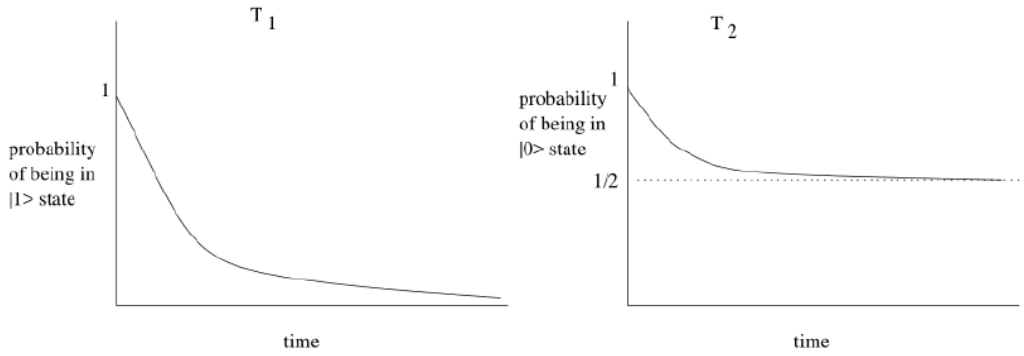


FIGURE 2.5: Expected experimental curves for T_1 and T_2, Ref. [72].

Since IBM provides values for coherence times T_1 and T_2 these times can be compared with an estimated time for the execution. In Table 2.1 the statistics

Parameter	Average	Min	Max
$T_1(\mu s)$	108.5	69.34	197.58
$T_2(\mu s)$	90.63	19.1	$248,62$
Readout error	1.18×10^{-2}	3×10^{-3}	1.418×10^{-1}
CNOT error	8.126×10^{-3}	3.913×10^{-3}	1
Gate time (ns)	451.556	298.667	625.778

TABLE 2.1: Properties of the *ibm_kolkata* quantum device

about the values of T_1 and T_2 end the other performance parameters of the 27-qubit device *ibm_kolkata* are listed. The device *ibm_kolkata* is one of the best performing superconductive devices in terms of error resilience vs number of qubits. A metric that evaluates the performance of a NISQ quantum hardware is the Quantum Volume (QV) [73]. The quantum volume evaluates the performances of hardware taking into account the error resilience and the number of qubits. The device *ibm_kolkata* has the most high QV for a superconductive device. Considering a trapped ion device, the coherence times are considerably longer. This is thanks to the hyperfine states technique described in the previous section. For example, the 11-qubit trapped ion device *harmony* device of IONQ has average times: $T_1 > 10^7 \mu s$, therefore the order of seconds, while $T_2 = 2 \times 10^5 \mu s$. The downside, however, is that gate execution times are very high. The two qubit gate has an average execution time of $210 \mu s$ which is 1000 times the execution time of the superconductive device. A new generation IONQ device has up to $T_1 = 100s$ and $T_2 = 1s$ and a quantum volume of 4 orders of magnitude with respect to the more performing superconductive device but, however, hundreds of microseconds as gate runtime.

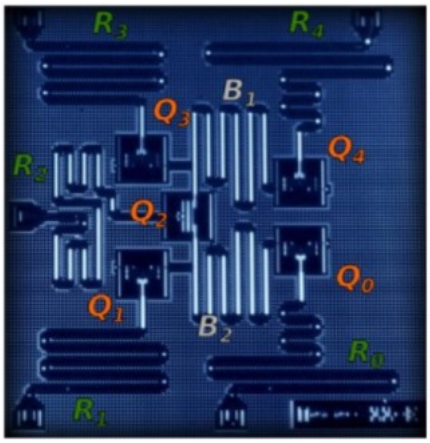


FIGURE 2.6: Immagine of the *ibmqx2* quantum processor [74]

2.6.2 Error rates and coupling map

In addition to the decoherence, each operation performed with quantum gates introduces accuracy errors in the system. According to IBM, CX gates are less accurate than single-qubit operations by approximately a factor of 10. The error rates are not fixed and depend on the calibration of the device. Each device is typically calibrated twice daily, and from each calibration, a list of qubit-specific operation error rates is provided, as well as the associated measurement error rates.

Moreover, we have that not all CX between the qubits are physically imple-

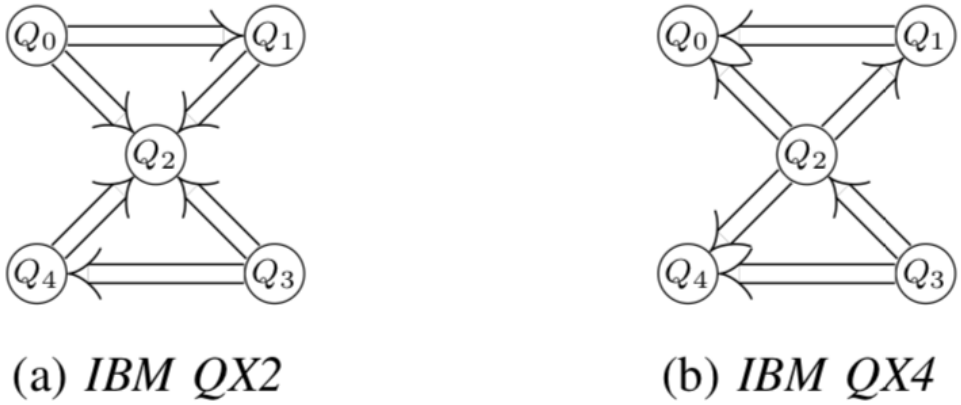
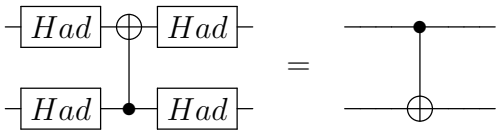


FIGURE 2.7: Coupling map of two different quantum processor of 5 qubits [59]

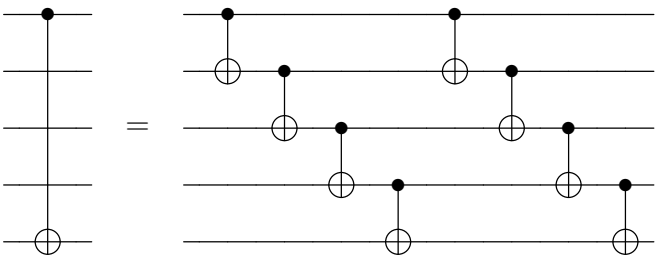
mented. In Figure 2.7 there are two examples of coupling maps i.e. the physical connections between the qubits in the processor. Such links are represented by an arrow that establishes which CX is physically implemented. The arrow starts from the control qubit to the target qubit.

There is a method to reverse the rules of the two coupled qubits using Hadamard gates [44]



If you want to apply an operation on two unconnected qubits, it is only necessary that there is a path between qubits that joins them in the lattice. In fact, if for example, we consider a circuit with five qubits connected via a linear lattice,

the first and last can be entangled with a CX gate using the following circuit



Therefore if the path in the lattice involves n qubit, $2(n-1)$ gates are needed. Because of errors and decoherence times, current quantum computers can be said to perform approximate quantum computations. Together with chip scaling (number of qubits), noise and decoherence rates are the greatest obstacles to a scenario of quantum supremacy. As such, the development of quantum error correction schemes is an active area of research. Different hardware providers develop quantum devices with different topologies for the qubit lattice. Google develops devices with a quadratic lattice, IBM with a hexagonal lattice, and Rigetti with an octagonal lattice instead. Trapped ion devices are, instead, fully connected and this is an important feature that makes trapped ion devices suitable for implementing error correction already today.

2.7 Error correction

An approach to building logical qubits is with the surface codes based on the stabilizer formalism. The surface codes are, in general, the most promising class of error correction techniques. They are an evolution of the toric code developed by Kitaev. Surface codes are the planar version of the toric code. One of the significant advantages of surface codes is their relative tolerance to local errors.

Stabilizer formalism

Stabilizer codes are very useful to work with. The general formalism applies broadly and there exists general rules to construct preparation circuits, correction circuits and fault-tolerant logical gate operations once the stabilizer structure of the code is known. The stabilizer formalism was first introduced by Daniel Gottesman [75] uses the Heisenberg representation for quantum mechanics. Describing quantum states in terms of operators rather than the states. A state $|\psi\rangle$ is defined to be stabilized by some operator, K if $|\psi\rangle$ is an eigenstate

of K with eigenvalue $+1$

$$K\left|\psi\right\rangle = \left|\psi\right\rangle \tag{2.142}$$

in other words, $|\psi\rangle$ is invariant under K. For example, $|0\rangle$ is stabilized by Z. An n-qubit stabilizer state is then defined by the n generators the of an Abelian (all elements commute) sub-group, $\mathcal{G}$, of the n-qubit Pauli group $\mathcal{P}_n = \{\mathbb{I}, X, Y, Z\}^{\otimes n}$

$$\mathcal{G} = \{K_i | K_i\left|\psi\right\rangle = \left|\psi\right\rangle, [K_i, K_j] = 0, \forall\,(i,j)\} \subset \mathcal{P}_n \tag{2.143}$$

Other properties of the stabilizers are that K_i is hermitian and $K_i^2 = \mathbb{I}\ \forall i$. Many extremely useful multi-qubit states are stabilizer states, including two-qubit Bell states, and Greenberger-Horne-Zeilinger (GHZ) states. Using X and Z gates it is possible to design the generators of $\mathcal{G}$. Given a GHZ state on 3 qubit $|GHZ\rangle_3 = 1/\sqrt{2}\,(|000\rangle + |111\rangle)$ the generators can be XXX, ZZI and IZZ. The link be- tween stabilizer codes and stabilizer states come about by dfining a relevant coding subspace within the larger Hilbert space of a multi-qubit system. To illustrate this reduction let us examine a simple two qubit example. A 2-qubit system has a Hilbert space dimension of four, however, if we require that these two qubit state is stabilized by the XX operator, then there are only two orthogonal basis states which satifies this condition,

$$\left|0\right\rangle_L \equiv \frac{1}{\sqrt{2}}\left(\left|01\right\rangle + \left|10\right\rangle\right) \qquad \left|1\right\rangle_R \equiv \frac{1}{\sqrt{2}}\left(\left|00\right\rangle + \left|11\right\rangle\right) \tag{2.144}$$

which can be used to define an effective logical qubit. Hence by using stabilizers, we can reduce the size of the Hilbert space for a multi-qubit system to an effective single qubit system. In the context of QEC, the stabilizers that are used to define the logical subspace are utilized to detect and correct errors. A stabilizer code is therefore a subspace defined via stabilizer operators for a multi-qubit system. The physical qubits, in a logical one, are therefore initialized in $\left|0\right\rangle_L$ put in entanglement the physical qubits. Operators that act in the subspace generated by $\{\left|0\right\rangle_L, \left|1\right\rangle_L\}$ must be defined. In the meanwhile, the stabilizers are used to measure the errors. As errors occur from the random and unpredictable appearance of X and Z operations, they must be detected by repeatedly measuring each qubit, which can be done with combined X and Z measurements. However, since $[X, Z] \neq 0$, sequential measurements of the same qubit conflict with one another, causing random projections of the qubit state onto these operators' respective eigenstates, completely destroying the quantum state. Instead, stabilizers are defined such that $[K_i, K_j] = 0\ \forall\,(i,j)$.

Chapter 3

Memcomputing

In the landscape of modern computing, three foundational concepts stand as pillars, shaping the very essence of computation and technological progress. Alan Turing's visionary creation, the Turing machine, set forth the abstract blueprint for computation, while John von Neumann's architectural innovation revolutionized the way computers operate. Parallelly, Gordon Moore's empirical observation of the pace of technological advancement, known as Moore's Law, propelled the digital realm into a trajectory of exponential growth. Computing architectures have not changed in their essential features since the 1950s, but many parts of the engineering of such systems have improved, resulting in handheld smartphones more capable than room-sized supercomputers of the past. But this framework is now showing its age. The Von Neumann architecture allows for flexible general purpose computation but also introduces what is called the von Neumann bottleneck. Performance in these systems is limited by the constant need to communicate between CPU and memory, to obtain data as well as program instructions. This fundamental organizational limitation, as well as physical constraints brought on by Moore's law scaling of transistors, and a growing set of unreachably difficult optimization problems, have together spurred the interest in unconventional computing architectures. Among these alternatives to the Turing-Von Neumann approach, we find the Memcomputing paradigm. Memcomputing is the name given to an emerging computational paradigm that exploits the evolution of a circuit based on memristors to perform computations. It should not be confused with the in-memory or near-memory computing paradigm which was conceived to avoid most of the costs of moving data by processing directly within the memory subsystem [76]. In fact, Memcomputing is a non-Turing paradigm that does not exploit the Von Neumann architecture. It does not have a dedicated memory component but rather exploits the evolution of a physical system. The dynamical evolution of this system, therefore, plays the role of a fictitious memory. Memcomputing devices perform computations harnessing the nonlinear dynamics of a physical system.

Such concept was pioneered by Chua [77]. The Chua's approach allows to solve nonlinear optimization problems [77] through possibly such nonlinear dynamics. One of the key differences between the Chua's approach and Memcomputing is that while the former is a fully analogical method, the latter exploits logical gates, hence the dynamical evolution is used to support a fully bit-based solver. Another computational device based on the dynamic of nonlinear systems has been recently devised . This is the Simulated Bifurcation Machine (SBM) developed by Toshiba [78] which is inspired by quantum principles. The SBM simulates a classical system of nonlinear oscillators that produces bifurcations in the phase space, enabling the simulation of a spin-glass system. As SBM is a fully digital solution this system is currently the most closely related to Memcomputing.

3.1 Non-Turing computation with dynamical systems

To understand memcomputing it is necessary to introduce a dynamical system as a computing paradigm first [79]. The first consideration is that the computational process itself is a dynamical system, regardless of the object of computation. Problems are formulated such that a physical system evolves from a start state, which represents the problem instance, to an end state which represents the solution. For example, consider a conventional digital computer. It can be represented as a long Boolean vector combined with a state transition function, which is in fact the logic of the machine. Normally we think of the machine as moving in discrete steps. While a dynamical system may legitimately operate in a discrete state space, note that a physically-realized the digital computer actually moves rapidly through a continuous space of values, such as voltage levels, settling within well-defined and well-separated ranges that represent 0 and 1. An analog computer is an even more natural candidate for computing as a dynamical system. In the following, we will assume a machine that can represent variables with more than 1 bit of precision. This description is general enough to fit either digital or analog machines. In the digital case, multiple bits are grouped together to represent a single variable, and dynamics are expressed as a set of numerical operations implemented on top of digital logic.

3.1.1 Dynamical systems

First of all, the term dynamical system has to be explained [80]. Dynamical systems can be approached in two ways. On the one hand, on an abstract level, a dynamical system is just a difference or differential equation, the properties of which shall be studied. On the other hand, this formal mathematical description generally appears as the result of a modeling process in the analysis of a time-dependent process, for example, the course of a chemical reaction, the spread of a forest fire, the growth of a population, the spreading of an epidemic disease, or the run of an evolutionary algorithm. A modeling process aims at formalizing and quantifying the important characteristics of the system, for instance, the interactions between entities. If the modeler is interested in how the system evolves in time, the result is a dynamic system. Thus, a dynamical system is a model of a time-dependent system or process expressed in a formal mathematical way. In general, several classes of dynamical systems can be distinguished. Dynamical systems can be grouped by the nature of the change, that is, whether the changes are deterministic or stochastic. This section considers deterministic systems. For a rigorous introduction to the field of random dynamical systems, the reader is referred to Arnold [81]. Dynamical systems can further be classified into discrete time and continuous time dynamical systems. In the first case, the change of the variable, $\vec{x} \in X \subset \mathbb{R}^n$, where X is called phase space, occurs at discrete time steps, leading to an iterated map or difference equation

$$\vec{x}\,(t+1) = f\,(\vec{x}\,(t)) \tag{3.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and a starting point $\vec{x}\,(0) = \vec{x}_0$ is fixed. In general, the starting time is assumed to be $t_0 = 0$. A dynamical system in continuous time leads to a differential equation

$$\frac{d}{dt}\vec{x}(t) = F\,(\vec{x}(t)) \tag{3.2}$$

where $\vec{x} \in X \subset \mathbb{R}^n$, $\vec{x}(0) = \vec{x}_0$ and $F : \mathbb{R}^n \to \mathbb{R}^n$ is the flow vector field representing the laws of the temporal evolution of $\vec{x}$. The solutions $\{\vec{x}(t) |\, \dot{\vec{x}}(t) = F\,(\vec{x}(t))\}$ are also called trajectories in the phase space. In a more general definition, F is time-dependent therefore, $\dot{\vec{x}} = F(\vec{x}, t)$ where $F : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$, but for simplicity, only the time-invariant case will be considered $F(\vec{x}, t) \equiv F(\vec{x})$. Considering now only the continuous time case, an equivalent to the equation

3.2 is the following

$$T(t)\vec{x}(0) = \vec{x}(0) + \int_0^t F\left(\vec{x}\left(t'\right)\right) dt' \tag{3.3}$$

where $T : \mathbb{R} \times X \to X$ is called the flow field or just the flow of the dynamical system. Generally, finding a solution to Equation 3.2 is very complicated for cases where $F(\vec{x})$ is non-linear. However, one way to obtain information on the properties of the dynamic system under examination is to study the solutions in equilibrium and their stability through Liapunov theories. Equilibrium solutions are solutions of Equation 3.2 that are the rest points of the system, that is points where no more changes in the system occur. There are many synonyms for equilibrium solutions including the terms rest point, singularity, fix point, fixed point, stationary point (also solution or state), or steady state [82]. In other words, an equilibrium solution is a solution $\vec{x}(t) = \vec{x}_e$ such that

$$\frac{d\vec{x}_e}{dt} = 0 \Rightarrow F\left(\vec{x}_e\right) = 0 \tag{3.4}$$

Therefore, $\vec{x}_e$ is an equilibrium point if and only if $\vec{x}(t) = x_e$ is a trajectory. However, not all the Equilibrium solutions have the same properties. Equilibrium points can be divided into different classes following the study of their stability. A simple analysis to study the stability of an equilibrium point is the linear stability analysis [80]. Expanding the Equation 3.2 on an equilibrium point $\vec{x}_e$

$$\dot{\vec{x}} \approx J(\vec{x}_e)(\vec{x} - \vec{x}_e) \tag{3.5}$$

where J is the Jacobian matrix. By diagonalizing the Jacobian it is possible to calculate the local trajectories in the neighborhood of x_e.

$$\vec{x}(t) \approx \vec{x}_e + \sum_i \vec{u}_i e^{\lambda_i t} \tag{3.6}$$

where the sum is over the eigenvalues λ_i and the associated eigenvectors $\vec{u}_i$ of the Jacobian matrix. If $\forall\, i$ $\mathrm{Re}\,\lambda_i$ is positive, than $\vec{x}_e$ is repulsive, otherwise if $\forall\, i$ $\mathrm{Re}\,\lambda_i$ is negative than $\vec{x}_e$ is stable. A dynamical system can then have an equilibrium point with most directions being stable and a few flat ones (directions $\vec{u}_i$ where $\mathrm{Re}\,\lambda = 0$). The stability of that point would still be governed by the stable directions. The number of eigenvalues with positive real part (number of unstable directions) is called the index of the equilibrium point. However, a more sophisticated analysis of the equilibrium point stability was provided by Lyapunov.

Lyapunov stability

Assuming that $F(\vec{x})$ satisfies the standard conditions for the existence and uniqueness of solutions, F is Lipschitz continuous, an equilibrium point $\vec{x}_e$ is said locally stable, in the sense of Lyapunov, if $\forall\, \epsilon > 0$, $\exists\, \delta > 0$, such that, if $\|\vec{x}(0) - \vec{x}_e\| \leq \delta \Rightarrow \|\vec{x}(t) - \vec{x}_e\| \leq \epsilon$, $\forall\, t \geq 0$ [83]. That means all the trajectories that have an initial point near the equilibrium point $\vec{x}_e$ remain near $\vec{x}_e$ all time. More in-depth, an equilibrium point is said locally asymptotically stable if it is locally stable and δ can be chosen so that

$$\|\vec{x}(0) - \vec{x}_e\| \leq \delta \Rightarrow \vec{x}(t) \to \vec{x}_e \text{ as } t \to \infty \tag{3.7}$$

while a more strong definition is the globally asymptotically stable equilibrium point which is similar to the above condition but $\forall \delta > 0$.
To verify the stability of an equilibrium point without explicitly solving the differential equation associated with the dynamical system, an important tool is the Lyapunov function. A Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$ that maps a trajectory $\vec{x}(t)$ to a scalar $V(\vec{x}(t))$. A Lyapunov function can be thought of as a generalized energy function, and $-\dot{V}$ as the associated generalized dissipation function. Studying the Lyapunov function as a function of time allows us to infer the stability of trajectories. To characterize the Lyapunov function it is first necessary to provide the following definitions. Assuming a set D such that $0 \in D \subseteq \mathbb{R}^n$, a function $V : D \to \mathbb{R}$ is positive semidefinite (PSD) on D if $V(0) = 0$ and $V(\vec{x}) \geq 0$, $\forall x \in D$. Instead $V : D \to \mathbb{R}$ is called positive definite (PD) on D if $V(0) = 0$ and $V(\vec{x}) > 0$, $\forall x \in D \setminus \{0\}$. In the end, V is negative semidefinite (NSD), or negative definite (ND), if $-V$ is PSD, or PD. Another important definition is the Lie derivative. Lie derivative of a $\mathcal{C}^1$ function $V : \mathbb{R}^n \to \mathbb{R}$ along vector field $F : \mathbb{R}^n \to \mathbb{R}^n$ is

$$\mathcal{L}_F V(\vec{x}) \equiv \nabla V(\vec{x})^T F(\vec{x}) \tag{3.8}$$

Let $\vec{x}(t)$ be a solution to ODE $\dot{\vec{x}}(t) = F(\vec{x}(t))$, than the derivative of $V(\vec{x}(t))$ with respect to the time

$$\frac{dV}{dt} = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \frac{\partial x_i}{\partial t} = \nabla V(\vec{x})^T F(\vec{x}) = \mathcal{L}_F V(\vec{x}) \tag{3.9}$$

Therefore, the Lie derivative characterizes the time-course evolution of the value of V along the solution trajectory of $\dot{\vec{x}} = F(\vec{x})$. With these elements, it is possible to state the Lyapunov theorem as follows

Theorem 3.1.1. *Let $D \subset \mathbb{R}^n$ be a set containing an open neighborhood of the origin. If there exists a PD function $V : D \to \mathbb{R}$ such that $\mathcal{L}_F V(\vec{x})$ is NSD then the origin is stable. If in addition, $\mathcal{L}_F V(\vec{x})$ is ND then the origin is asymptotically stable.*

A Lyapunov function is a PD $\mathcal{C}^1$ function such that $\mathcal{L}_F V(\vec{x})$ is NSD or ND. If V is also radially unbounded, then, the origin is globally asymptotically stable. Then, to verify if $\vec{x}_e$ is globally asymptotically stable, the first step is to translate the space such that $\vec{x}_e = 0$ and then find a Lyapunov function with $\mathcal{L}_F V(\vec{x})$ ND on a radial region $R_A \in D$ centered on 0, such that $V(\vec{x}) \to \infty$ as $\vec{x} \to \partial R_A$.

3.1.2 Ideal dynamical system for computation

Before introducing how a dynamic system can be engineered to solve computational problems, an important classification of dynamic systems must be highlighted. Dynamical systems can be typically grouped into two categories, according to whether they conserve or not volumes in phase space. Considering at time $t = 0$ and a $(n-1)$-dimensional surface $\partial\Omega(t=0)$ that encloses a volume of the phase space $\Omega(t=0)$ Let's then take all points on this surface and let them evolve according to the Equation 3.2. At time t the surface evolves in $\partial\Omega(t)$ which encloses the volume $\Omega(t)$. The system is said conservative if $\Omega(t) = \Omega(t=0)$ and non-conservative if $\Omega(t) \neq \Omega(t=0)$. Moreover, another definition is the dissipative system where $\Omega(t) < \Omega(t=0)$. Since

$$\frac{d\Omega}{dt} = \int_{\partial\Omega(t)} F(\vec{x}(t)) \cdot \hat{n} d\sigma = \int_{\Omega(t)} \nabla \cdot F(\vec{x}(t)) dx \tag{3.10}$$

if $\nabla \cdot F(\vec{x}(t)) < 0$ then the system is dissipative. To have good properties for computation is important to study the long-time properties of dynamical systems. The question is, waiting enough time, it will be possible to determine that the system trajectory is within a finite distance from a solution point and with that distance independent of the size of the problem we are trying to solve? Therefore it is necessary to design a dissipative system due to the definition of attractor. It is called an attractor The set of all points in the phase space to which a dissipative dynamical system converges in the long-time limit. For instance, the attractor of an equilibrium point is that equilibrium point itself. Other kinds of trajectories have different attractors. There are periodic orbits i.e. solutions where there exists a constant $T_P > 0$ such that $\vec{x}_P(t) = \vec{x}_P(t + nT_P)$ where $n \in \mathbb{N}$. It is clear that such orbits 'reside' on hyper-surfaces of

multi-dimensional invariant tori. However, a dynamical system may also have quasi-periodic orbits as attractors. They never really come back exactly on themselves, like periodic orbits, rather, they densely cover the hyper-surfaces of multi-dimensional invariant tori. Finally, the Equation 3.2 may show a highly irregular pattern at long times, which, despite being deterministic, is difficult (practically impossible) to predict. In such cases, the system displays a chaotic dynamical pattern. Therefore, according to the initial conditions we choose for the Eqaution 3.2, we may end up in any of these attractors. We can then define the following important notion: It is defined as the basin of attraction the set of initial points in the phase space that evolve according to the Equation 3.2, and whose trajectories end up in a given attractor. With regard to computation, it is clear that neither (quasi-)periodic orbits, nor chaotic behavior are good features to design a physical machine to compute the solution of a particular problem. In both cases, the solution or output would still change in time, in the chaotic case, even unpredictably.
Initial conditions are extremely important when the bifurcation phenomenon occurs in the dynamical system. In dynamical systems, a bifurcation occurs when a small smooth change made to the parameter values (the bifurcation parameters) of a system causes a sudden qualitative or topological change in its behavior. There are several classes of bifurcations, an example is the saddle-node bifurcation. Assuming a dynamical system in 1-dimension, $\dot{u} = f(u, \mu)$ where μ is the bifurcation parameter. If $f(u, \mu)$ satisfying:

$$f(0,0) = 0 \quad \frac{\partial f}{\partial u}(0,0) = 0 \tag{3.11}$$

which means that $u = 0$ is an equilibrium point at $\mu = 0$. Supposing that the ODE that defines the system satisfies the hypotheses of the Cauchy-Lipschitz theorem, such that for each initial condition, there exists a unique solution and that, $f \in \mathcal{C}^k$ with $k \geq 2$ in a neighborhood of $(u, \mu) = (0, 0)$ and

$$a \equiv \frac{\partial f}{\partial \mu}(0,0) \neq 0 \quad b \equiv \frac{\partial^2 f}{\partial u^2}(0,0) \neq 0 \tag{3.12}$$

The following properties hold in neighborhood of $u = 0$ for small enough μ

- if $ab < 0$ the differential equation has no equilibrium point for $\mu < 0$, or for $\mu > 0$ if $ab > 0$
- if $ab < 0$ the differential equation possesses two equilibrium point $u_{\pm}(\epsilon)$, $\epsilon = \sqrt{\mu}$ for $\mu > 0$ (or for $\mu > 0$ if $ab > 0$), with opposite stability.

Then for the system, a saddle-node bifurcation occurs at $\mu = 0$. From the Equation 3.12, the series expansion of f around $(0, 0)$ is

$$f(u, \mu) = a\mu + bu^2 + o\left(|\mu| + u^2\right) \tag{3.13}$$

therefore, closed to $(0, 0)$

$$\mu = -\frac{b}{a}u^2 + o\left(u^2\right) \tag{3.14}$$

Consequently, if $ab\mu > 0$ the system has no equilibrium point, for $\mu = 0$ it has an equilibrium point in $U = 0$ instead, for $ab\mu < 0$ a pair of equilibrium point occurs

$$u_{\pm}(\mu) = \pm\sqrt{-\frac{a\mu}{b}} + o\left(\sqrt{|\mu|}\right) \Rightarrow \frac{\partial f}{\partial u}\left(u_{\pm}(\mu), \mu\right) = 2bu_{\pm}(\mu) + o\left(\sqrt{|\mu|}\right) \tag{3.15}$$

then the equilibrium $u_{-}(\mu)$ is attractive, asymptotically stable when $b > 0$ and repelling, unstable when $b < 0$. Whereas, the equilibrium $u_{+}(\mu)$ has opposite stability properties. This means that by changing μ, linked to the initial conditions and therefore to the statement of the computational problem to be solved, it is possible to direct the system towards one solution or another. This is therefore a good property for a dynamic system to be exploited as a computing machine. In the particular context of the memcomputing paradigm, the focal point is to design a dynamic system so that it has topological (and therefore global) characteristics that will aid in the efficient solution of combinatorial optimization problems. Such features can be summarized in the following points

1. The dynamical system has to be dissipative so that it has attractors.
2. Of all the attractors, we choose equilibrium points as representations of the solutions to a given problem. Say, we map one equilibrium point to one solution. The system has to be constrained to always reach such points.
3. This means that we do not want (quasi-)periodic orbits or chaos.
4. A part from the equilibrium points representing the solutions to the problem, to avoid local minimum traps, the system can only have, as additional critical points saddle points and no additional minima.
5. The system must admit the bifurcation phenomenon so as to control the topology of the system with the initial condition and then obtain different solutions, (equilibrium points) with different inputs (initial conditions).

3.2 Realisation of digital memcomputing machine

Electronic circuits are ideal for engineering a dynamic system for the purpose of performing calculations. Chua, in 1983, [77], created one of the most famous examples of an electronic circuit used as an analog solver of non-linear optimization problems. The task of solving a non-linear optimization problem with constraints reduces to that of finding the solution of the associated canonical circuit using a circuit simulation program, such as SPICE. This is a non-Turing but analog approach for solving, for examples, linear programming problems, quadratic programming problems, and polynomial programming problems. Memcomputing takes inspiration from the Chua circuit design and elevates it into a digital version.

3.2.1 From analog to digital

Meaningful interaction between man and calculating machine involves defining a finite set of symbols and reading the result of the calculation unambiguously by finite means Writing the input and reading the results cannot possibly require exponential resources as the size of the input increases. This requirement means that one wants to solve a problem whose input and output are digital, or at least it can be written this way (maybe as an approximation). Manipulation of digital symbols is best performed by Boolean algebra, that is the set of logical propositions that relate the truth value of two or more variables represented in a binary form, namely, whether their value is 'true' (logical 1) or 'false' (logical 0) [84]. AND and OR gates are two examples of boolean operations that act on two binary variables (two bits) and they produce a binary output. Classical digital computing is based on the decomposition of mathematical calculations in a universal basis of boolean operations. The physical realization of such operations is made by exploiting electronic circuits. Such a computational paradigm is analog to a Turing machine. Specifically, classical computation as understood by Zuse and Von Neumann is equivalent to a Turing machine having memory of the state of the binary variables in the register. Memcompuiting instead is a non-Turing paradigm and it exploits a dynamical system to simulate the behavior of logical operations thus resulting in a non-Turing but digital paradigm. The main revolution of the memcomputing paradigm is based on the creation of bi-directional or terminal-agnostic boolean gates, this implies that they will always satisfy their logical proposition, irrespective of whether the truth value is assigned at the traditional in-terminals or the traditional out-terminals and it is not a feature of Turing machines. The dynamic system that creates these

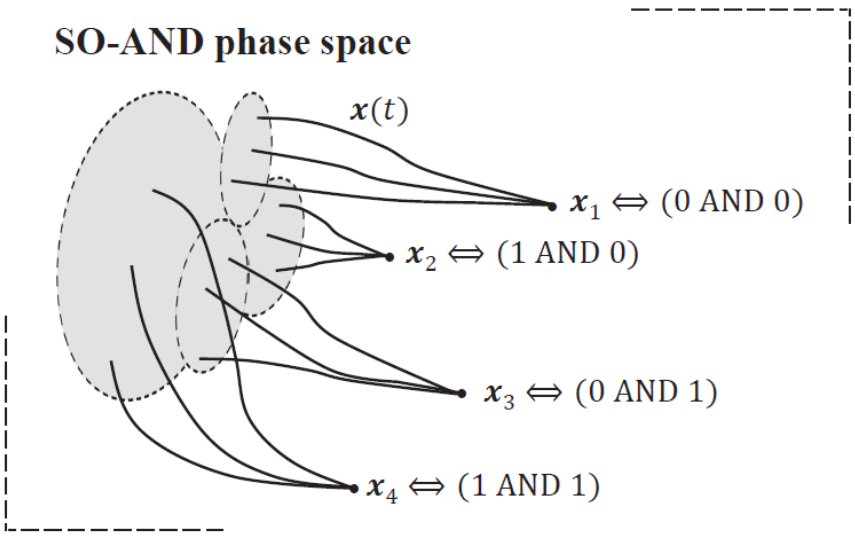


FIGURE 3.1: The phase space of a self-organizing AND (SO-AND) gate with only four equilibria, each one corresponding to a logically consistent state of an AND gate. In the absence of any other attractor, the phase space of this gate clusters into four basins of attraction (grey areas). Reference figure in [85].

terminal-agnostic gates has the property of time non-locality due to the long-range order in the system variables. Such a concept is related to a memory [85]. Time non-locality is an important feature of quantum computing and it is related to the quantum entanglement.

3.2.2 Self-Organizing Logical Gates

let's consider an AND gate. This gate has only four logically consistent states, according to the truth values assigned to the in-terminals:

$$0 \wedge 0 = 0; \;\; 0 \wedge 1 = 0; \;\; 1 \wedge 0 = 0 \;\; ; 1 \wedge 1 = 1 \tag{3.16}$$

Let's now design a physical system, following some appropriate equation of motion of the type of the Equation 3.2, which has only these four states as its equilibrium points, and no other attractor, namely no periodic orbits or chaos. This physical system will then dynamically self-organize (SO) into any one of these four states, according to the initial conditions. Such a gate is called self-Organizing Logic Gate (SOLG), in which only the final states are important, not how such states are reached during dynamics [86]. The variables of the original Boolean gate will be mapped into voltages of an actual electronic circuit that implements the dynamic system underlying the SOLGs. However, the three terminals of this new SO-AND gate are not necessarily digital, because

the voltages follow a trajectory $\vec{x}(t)$ in the phase space $X \subset \mathbb{R}^3$. It is necessary to relax the digital condition at the terminals of the gate and allow them to adapt to any bounded value they may support during dynamics.
Therefore, when the system is at equilibrium (in one of the four logically consistent states), no dynamics should occur. On the other hand, away from the logically consistent states, the system will change its state, always attempting to converge to one of the equilibrium points (Figure 3.1). In fact, at any given time during dynamics (after the initial condition has been set, but before the output is reached), the state of the system could be in any non-linear combination of 'input' and 'output' states. When the system is in a non-linear combination of states then the system is in an unstable configuration as shown in Figure 3.2.
This behavior can be realized if we introduce extra degrees of freedom, namely, the dimension of our (phase) space of dynamical variables is not three (or whatever is the number of gate terminals), but larger. The extra (memory) degrees of freedom are represented by $\tilde{\vec{x}}$. They could be due to any physical mechanism that induces time non-locality (memory) in the system [87]. To summarize the concept of the SOLG two steps have to be specified: Provide dynamics to the voltages at the terminals, $v_i = v_i(t)$ where i is the index associated with the gate terminal; Add as many extra dynamical (memory) degrees of freedom $\tilde{x}_k(t)$ as necessary to let the system evolve to the only logically consistent equilibrium states of the gate.
Let's consider first the step: by providing dynamics to the literals of the terminals of the original gate, one goes from a set of discrete states (the logically consistent solutions of the gate) to a dynamical system of the voltage variables only. The resulting system may have several types of critical points, in addition to the equilibrium points which correspond to the correct logical proposition of the gate. In particular, this (reduced) phase space of the voltages may contain local minima that could trap the system dynamics. This is where the second step of the procedure comes in handy. If memory variables are appropriately introduced, they expand the reduced phase space of the voltages, transforming any possible local minimum into saddle points. This leaves the equilibrium points representing the logically consistent solutions as the only equilibrium points. Active elements are needed to obtain appropriate extra degrees of freedom. The reason is that the dynamical system has to be able to always end up in the correct equilibrium points, which represent the logical states of the gate. In other words, the terminal voltages have to be guided toward the solution. To accomplish this, the system needs feedback so that if it strays away from the correct path in the phase space, it will immediately correct it.

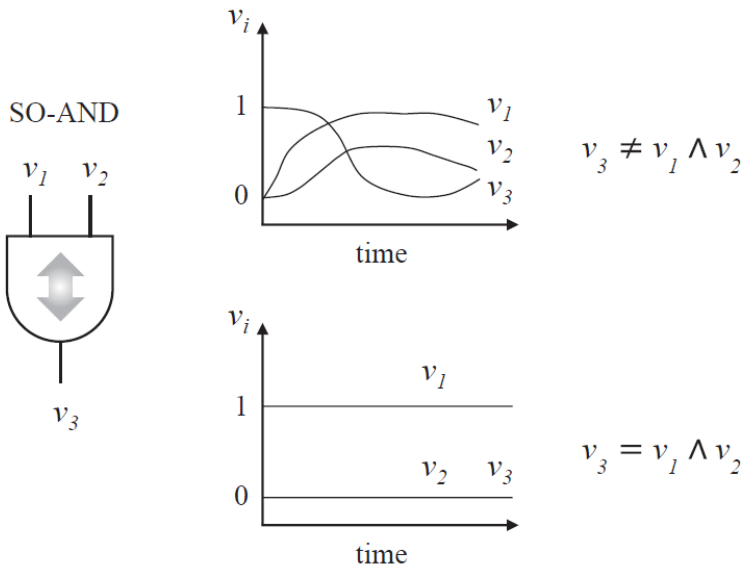


FIGURE 3.2: The SO-AND gate is in an unstable configuration if its logical relations are unsatisfied. It is in a stable configuration if one of its logical relations is satisfied. Reference figure in [85].

3.2.3 Physical realization of SOLGs

Although the specific gate realization may be different, the general idea is the same for all types of Boolean gates. The first step to describe the physical realization is to choose reference voltage v_c to associate the logical 0 to v_c (for example $v_c = 1V$) and 1 to $-v_c$. Consider, again, an AND gate with initial configuration 01 as input and 1 as output. It means that the SOLG is in an initially unstable configuration, and will attempt to dynamically change its state to reach one of the four possible equilibrium states compatible with an AND gate. A set of active devices called dynamic correction modules (DCMs) [88], attached to each terminal reads the voltages of the other terminals as well and they provide the necessary feedback to the system. DCMs are made of resistive memories, with a minimum R_{on} and a maximum R_{off}, and voltage-controlled voltage generators (VCVGs), which are active devices. In Figure 3.3 an illustration of the circuit that implements a DCM is shown. The VCVGs are linear voltage generators piloted by the voltages v_1, v_2, and v_3 at the terminals of the SOLG. The output voltage of the VCVG is given by

$$v_{VCVG} = a_1 v_1 + a_2 v_2 + a_3 v_3 + dc \tag{3.17}$$

The parameters a_1, a_2, a_3 and dc are determined to satisfy a set of constraints characteristic of the gate, namely constraints that will guide the physical system to always satisfy the gate logic. Therefore, these parameters are different for an

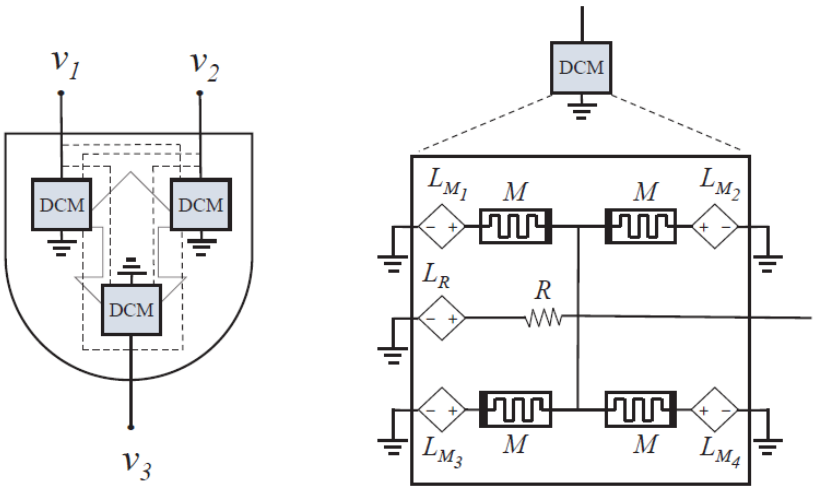


FIGURE 3.3: Self-organizing (SO) AND gate, left panel, formed by dynamic correction modules (DCMs), right panel.M indicates the resistive memories, while The linear functions L drive the voltage-controlled voltage generators (VCVG). Reference figure in [85]

AND, OR, XOR, or any other gate. Since 5 voltages are involved in DCMs, 20 parameters have to be defined. If the gate is connected to a network and the gate configuration is correct, no current flows from any terminal: the gate is in stable equilibrium (steady state). Note that one could choose a different set of parameters (or even a different design altogether for SOLGs) that satisfies the same conditions, or simply the general principles of operation of SOLGs. However, finding a suitable set of parameters to guide the system toward the equilibrium point is the main problem concerning the memcomputing paradigm.

3.3 Combinatorial optimization problems with memcomputing

3.3.1 Boolean problems and MAX-SAT

A logical proposition that relates variables is called a clause. A clause is also sometimes called a constraint. For later use, any clause can be converted into an equivalent form known as the conjunctive normal form (CNF). A logic formula is in conjunctive normal form if it is a conjunction of disjunctions of boolean variables. CNF, in a nutshell, is a series of clauses with ORs (disjunctions), that represent the constraints, connected by ANDs. This form defines a very important problem known as a satisfiability (SAT) problem or constraint satisfaction

problem, where the variables have to satisfy a set of constraints. Satisfiability (SAT) problem It is the problem of finding an assignment of the Boolean variables, such that their CNF formula evaluates to true. Equivalently, we can say that each clause should have at least one literal that is true under the assignment (for a disjunctive clause). Such a formula and all its clauses are then said to be 'satisfied'. If there is no assignment satisfying all clauses, the CNF formula is said to be 'unsatisfiable'. The particular CNF representing a SAT problem in which all clauses contain three distinct literals, of which none is a negation of the others, like

$$\varphi(\vec{x}) = (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \quad (3.18)$$

is called an instance of a 3-SAT problem.
Unlike 2-SAT problems which belong to the P class of problems that can be solved polynomially, 3-SAT problems are among the NP-complete problems. In general, such problems are Boolean problems. Let $\vec{x} \in \{0,1\}^n$ and a function $f : \{0,1\}^n \to \{0,1\}^m$, a system of Boolean statements (with n, m integers) that accept as input the Boolean variables $\vec{x} \in \{0,1\}^n$. Solving the problem defined by f and $\vec{y} \in \{0,1\}^m$ means that the machine must return a possible input $\vec{x}$ of f if it exists, that satisfies $f(\vec{x}) = \vec{y}$. Memcomputing is suitable for solving such kinds of problems.

Self-Organizing Logical Circuit

To introduce how memecomputnig can solve such problems the first consideration is that A CNF formula has a simple Boolean circuit representation [89].
Let's then assume that the problem can be written in the form of a Boolean circuit (whether in CNF or not), namely, it can be written as a set of logic gates whose input is a set of truth values (in a binary representation, a collection of 0s and 1s), and whose output is also a collection of truth values. Now that we have seen how to construct individual SOLGs, it is straightforward to assemble them to obtain self-organizing logic circuits (SOLCs). The construction of these SOLCs is indeed quite easy to do. The procedure can be summarized as follows: (i) Transform a given problem into Boolean format; (ii) Design one Boolean circuit that represents such a problem. This sets its physical topology (architecture); (iii) Replace the standard logic gates of this Boolean circuit with SOLGs; (iv) Apply the Boolean values that represent the input of the problem to the appropriate terminals of this circuit; (v) Let the circuit self-organize to the solution; vi Read the solution at the appropriate terminals. An example

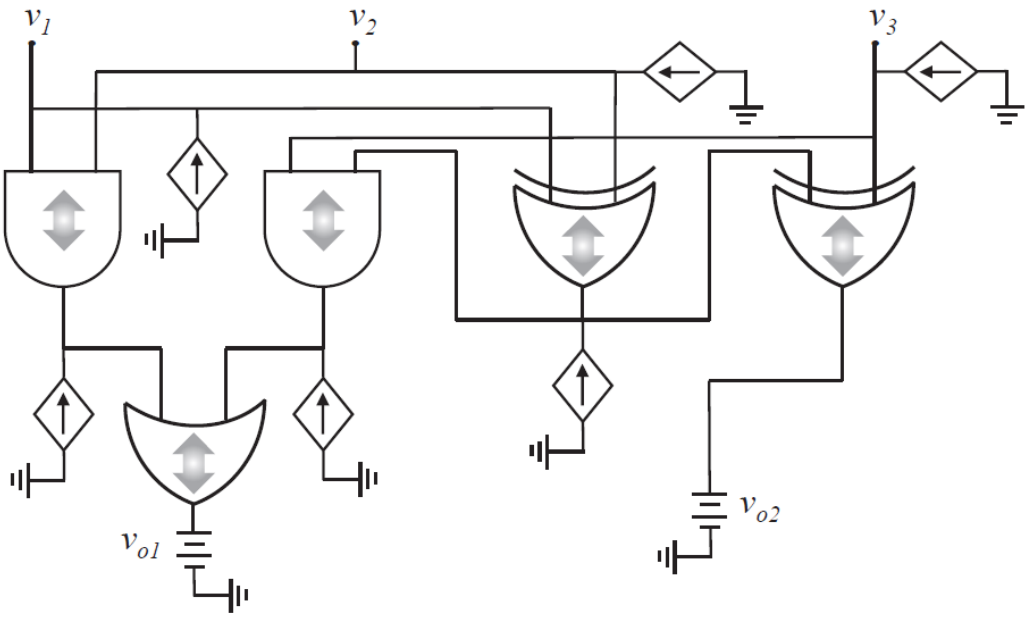


FIGURE 3.4: Sketch of a possible 3-bit sum SOLC such that, given the bits v_{o1} and v_{o2}, outputs the consistent bits v_1, v_2, and v_3. Note that in addition to SO-OR, SO-AND, and SOXOR gates, the circuit employs VCDCGs (diamonds with arrows inside) at each gate terminal, except at the terminals where the input is supplied. These VCDCGs eliminate, as possible equilibrium points, the zero-voltage state of the terminals. Reference figure in [85].

is provided by a 3-bit ADDER in the Figure 3.4. It is worth stressing once more that the previous choice of circuitry is just one possible realization. The important point to remember is that either we choose SOLG's parameters and design so that no solution other than the logically consistent one is present, or if that is difficult to do when one connects different SOLGs together to form a SOLC, we may connect a VCDCG at each terminal but the ones at which we send the inputs, as it is shown in Fig. 7.15. This guarantees the absence of any equilibria other than the logically consistent relations of the gates. However, a circuit can be understood as a single gate with a given number of input and output terminals. The elements that assemble it are autonomously self-organized but the property of these circuits is that it exploits long-range correlations to obtain large-scale order. To understand how the various SOLGs are reorganized within a SOLC, we need to think of the circuit as a many-body system with variables connected to each other. The memcomputing feature that allows solving Boolean problems (as well as combinatorial optimization problems) lies in working in a critical regime where it expresses a large-scale order. A system

that expresses order is a system whose fundamental units are arranged in an ordered fashion (imagine, for instance, the atoms in a solid). 'Long-range' means that such an order extends over the entire spatial (or temporal) dimension of the system, and even to infinity, if the system is extended to its thermodynamic limit.

Given a physical system, some of its properties (e.g., the spins of individual elements of magnetic materials; see the following example) may showcase this type of order. In memcomputing the correlations between the different degrees of freedom of the system are long-ranged. For instance, such phenomenon appears in many natural phenomena; the onset of continuous phase transitions being a typical (and important) example [90]. Consider, for instance, a ferromagnetic material in which the spin domains are all aligned in some direction. By varying some parameter, say increasing the external temperature, the material goes from one phase of matter (ferromagnet) to another (paramagnet) in which the spin domains have random orientations (Fig. 4.3). In proximity to the (critical) point in between the two phases, the correlations of the spin fluctuations decay as a power law, meaning that different spins can correlate at arbitrary distances from each other. To quantify what I just wrote we can calculate the (2-point) correlation function, $G_c(\vec{r})$, at a distance $\vec{r}$ between spins $s(\vec{r})$ (vectors with a given magnitude and orientation) at different locations, averaged over all possible spin configurations. Away from the critical point, we expect the correlations to be local/short-ranged, meaning that they typically decay exponentially

$$G_c(\vec{r}) \equiv \langle \Delta s(\vec{r}) \Delta s(0) \rangle \sim \exp\{-|\vec{r}|/\zeta\}, \quad |\vec{r}| \to \infty \tag{3.19}$$

where the fluctuations $\Delta s = s - \langle s \rangle$ ($\langle \cdot \rangle$ is the average among all the configurations), and ζ is the correlation length, namely the characteristic length over which the spin fluctuations correlate. At criticality (near the critical point), instead, we find that the correlation function behaves algebraically as

$$G_c(\vec{r}) \equiv \langle \Delta s(\vec{r}) \Delta s(0) \rangle \sim \frac{1}{|\vec{r}|^{\alpha}}, \quad |\vec{r}| \to \infty \tag{3.20}$$

with α some constant—called the critical exponent of the correlation function—that depends on the dimensionality, symmetries of the system, and range of interactions. The correlation length, ζ, instead diverges at criticality. Since the correlations decay as a power law, by simply scaling the distance $\vec{r}$ by some arbitrary factor λ, the correlation function would only change as $G_c(\lambda\vec{r}) = \lambda^{-\alpha} G_c(\vec{r})$, namely the system is self-similar at different scales: fluctuations occur at all length scales. We would the call these correlations scale-free (or

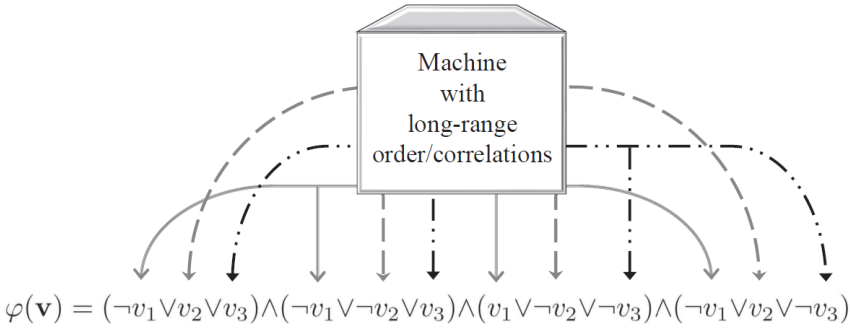


FIGURE 3.5: Illustrative representation of the long-range correlations in a circuit that realizes the conjunctive normal form in the Equation 3.18. A machine that supports long-range order/correlations is able to assign logical values to each variable of a CNF formula, in a correlated way, even though those variables appear in different clauses. This is because, the different variables can be mapped into quantities (e.g., voltages) of a physical system, with each quantity spatially separated from the others with the machine correlating such quantities. Reference figure in [85].

scale-invariant). Then one can see that if a dynamical system could be engineered so that its correlations are long-ranged, somehow driving it toward the critical state of some sort of (continuous) phase transition, its different parts could correlate strongly at long distances, making it a good candidate to solve hard problems efficiently.

This is because hard problems typically require the simultaneous (correlated) assignment of logical values of different variables anywhere in the problem specification. For instance, a conjunctive normal form (CNF) formula, as in Figure 3.5, can be viewed as a Boolean physical circuit, with the variables being physical quantities (e.g., voltages) of an actual physical system. A machine (a collection of elementary units) with long-range correlations can then assign logical values to the variables in a correlated way, even if the variables appear in multiple clauses. Therefore, computation at a continuous phase transition or, more precisely, computation at criticality seems very appealing. MemComputing machines should be some mechanism of collective dynamics. A similar idea of computing at criticality was put forward in 1990 by Langton [91], who argued, using cellular automata as a model of computing machines, that the optimal conditions for processing and storing of information are achieved in the proximity of a phase transition, in particular continuous ones that show critical behavior. The same author has called this behavior computation at the edge of chaos. A similar conclusion was subsequently reached regarding the operation of the brain, where one observes scale-free features in the firing of

neurons at distant locations of the cortex [92]. Collective dynamics then seems to be a property that MemComputing machines share with the brain. However, although these ideas are very compelling, even if correct, they do not suggest how to actually build a machine that could generate such a scale-free behavior on its own, namely without tuning any external parameters, and take full advantage of it during computation. The physical realization of a memcomputing machine suggested in the previous section indicates that this collective dynamic will necessarily depend on the choice of parameters that control the voltage of the active elements present in the DCMs internal to the SOLGs. Starting from the assumption of finding an adequate set of parameters, memcomputing is thus theoretically capable of solving NP-hard problems in polynomial time. For example, if we consider the integer factorization problem where it is required to find two primes p and q such that $n = p \times q$, we know that the circuit that creates a multiplier between two integers scales as $\mathcal{O}(N^2)$ where N is the number of bits related to n. By replacing the circuit with the equivalent SOLC then it is possible, by fixing n, to obtain the solution p and q in $\mathcal{O}(N^2)$.

MAX-SAT and combinatorial optimization

If the CNF formula cannot be satisfied by any set of literals, one could still try to solve a very important optimization problem: one could look for the variables assignment that maximizes the number of satisfied clauses. This problem is called a Maximum satisfiability (MAX-SAT) problem. In such a case, the MAX 2-SAT problem is NP-complete. Always taking the CNF in Equation 3.18 as an example of 3-SAT problem, the corresponding SOLC is shown in Figure 3.6. However, MAX-SAT aims at finding an assignment of the Boolean variables that maximizes the number of satisfied clauses (equivalently, minimizes the number of unsatisfied clauses) in a given logic formula. To define this problem via a SOLC one must take into consideration that there will be unsatisfied clauses, therefore, the dynamics do not necessarily reach an equilibrium point. In fact, in the infinite time limit, it would settle into a (quasi-)periodic orbit. Indeed, even if the dynamics reached the global optimum, we would not know if that is really the optimum, or some other 'sub-optimal' solution. Let us consider a percentage of unsatisfied clauses, and call it α. The question could be, how long does it take the solver to reach the threshold α as a function of problem size? Then adding an adder in the circuit in 3.6 on the clauses outputs and fixing the adder output with α. Iteratively, reducing α, the solution of the MAX-SAT can be obtained. Since MAX-SAT (from 3 and up at least) is an NP-complete

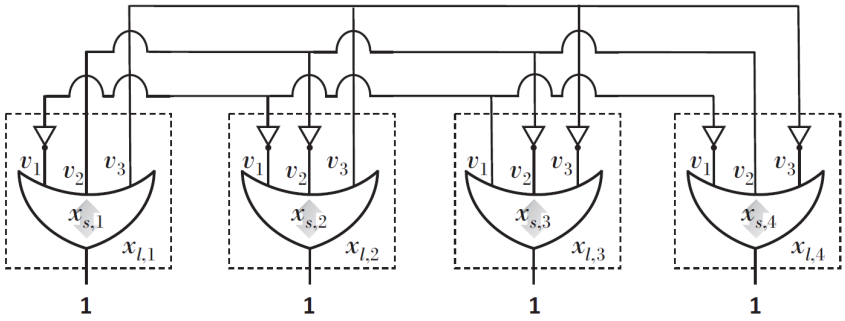


FIGURE 3.6: Example of a self-organizing circuit for a 3-SAT representing the CNF in Equation 3.18. The the output of each 3-terminal SO-OR gate is set to 1 (true), and the problem is to look for an assignment of the variables v_1, v_2 and v_3 that satisfies each clause. x_s and x_l are the memory variables. Reference figure in [85]

problem, then every combinatorial optimization problem (as well as every NP problem) can be mapped in polynomial time into a MAX-SAT problem [93].

3.3.2 Integer linear programming with memcomputing

Integer Linear Programming (ILP) is a class of combinatorial optimization problem that can be formalized as follows:

$$\begin{aligned} &\min_{\boldsymbol{x}} \sum_i f_i x_i \\ &A_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq} \\ &A_{ineq}\boldsymbol{x} \leq \boldsymbol{b}_{ineq} \end{aligned} \tag{3.21}$$

where $\boldsymbol{x} \in \mathbb{N}^n$, $f_i \in \mathbb{R}\, \forall i \in \{1, 2, \cdots, n\}$, $A_{eq} \in \mathbb{N}^{m_{eq} \times n}$, $A_{ineq} \in \mathbb{N}^{m_{ineq} \times n}$, $b_{eq} \in \mathbb{N}^{m_{eq}}$, $b_{ineq} \in \mathbb{N}^{m_{ineq}}$ and m_{eq} and m_{ineq} are the number of equalities and inequaliteies constraints respectively.
The memcomputing approach to ILP problems is based on the concept of Self-Organizing Algebraic Gates (SOAGs).

Self-Organizing Algebraic Gates

The strategy for building SOLGs can be employed also in the design of self-organizing algebraic gates (SOAGs) [94]. These gates are similar to the SOLGs in their mode of operation. However, instead of satisfying a Boolean relation, they satisfy an algebraic relation, e.g., an inequality relation between variables. SOAGs are terminal-agnostic gates that can use any terminal simultaneously

as 'input' or 'output' to satisfy an algebraic relation. In addition to time non-locality (memory), SOAGs require feedback to operate as desired, hence the need of active elements. Such gates can be used in tackling, e.g., Integer Linear Programming (ILP), an optimization problem in which one needs to find a variable assignment that satisfies also algebraic relations. The interested reader should consult for the application of SOAGs to ILP in [94].

Solving ILP problems with Memcomputing

Memcomputing deals with ILP problems through specific circuital elements, dubbed Self-Organizing Algebraic Gates. Using SOAGs, one can assemble a Self-Organizing Algebraic Circuit (SOAC). The SOAC collectively self-organizes in order to satisfy the constraints of an ILP problem 3.7. Indeed the linear equalities and inequalities of a given ILP problem can be directly mapped on a SOAC (see Figure 3.7). Instead, the cost function can be easily reformulated as an extra linear inequality with an extra bounding parameter. Iteratively, this bound is reduced forcing the SOAC to self-organize and find a new feasible solution, each time closer to the global optimum as described for MAX-SAT problem mapped on SOLC. Currently, no physical implementation of such concept exists, but Memcomputing Inc. has realized software able to simulate the circuital dynamic of Memcomputing machines on classical computers. The simulation software exploits GPUs to increase the overall performance.

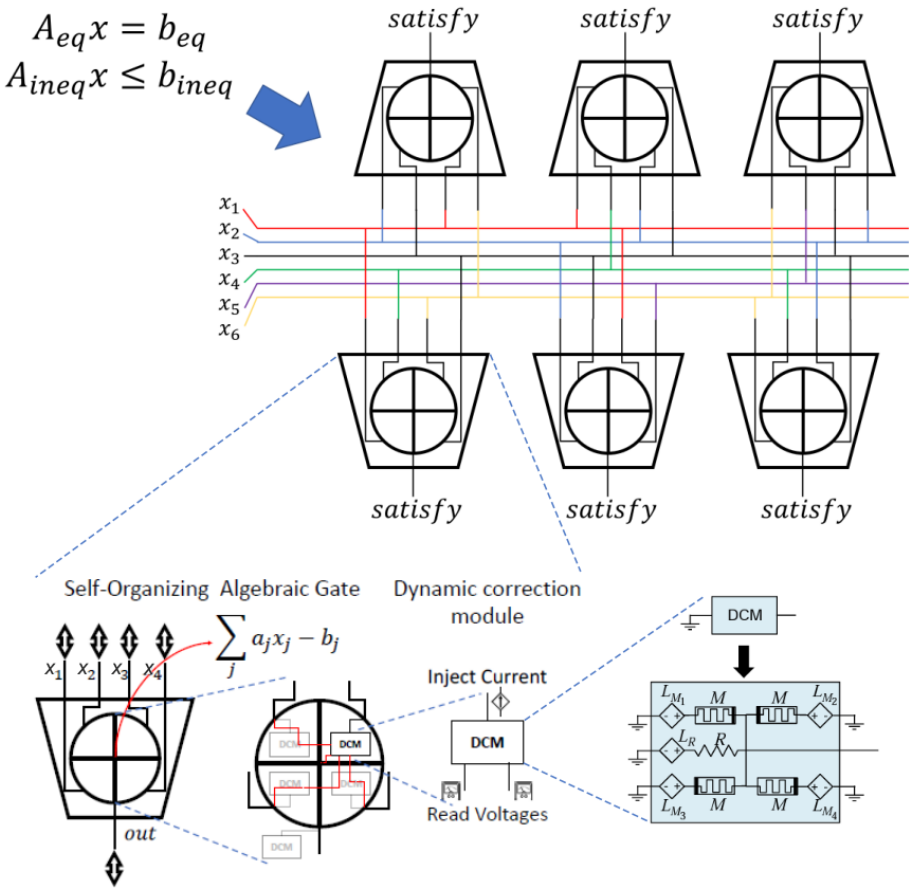


FIGURE 3.7: A Self-Organizing Algebraic Circuit (SOAC) represents an ILP problem. Each Self-Organizing Algebraic Gate (SOAG) is a linear condition that has to be satisfied when solving the ILP. The output of the SOAGs is imposed in order to obtain feasible solutions. The cost function is mapped into an additional SOAG whose inequality value is progressively reduced. The SOAGs at the circuital level are composed by dynamic correction modules (DCMs); the circuit components of a DCM are illustrated in the figure below on the right.

Chapter 4

Quantum computing for integral estimation

4.1 Chapter overview

This chapter discusses the application of quantum computing methods, specifically of methods called quantum amplitude estimation, for the estimation of integrals. The main aim is to evaluate the speedup of these techniques compared to Monte Carlo methods and their applicability on NISQ devices. Hybrid methods alternative to the standard definition of the Quantum Amplitude Estimation (QAE) algorithm were therefore studied and implemented. These algorithms are more resistant to noise than modern quantum machines. Specifically, a trapped ion quantum computer was used which guarantees better performance in terms of noise. The survey and implementation of different versions of the QAE algorithm reveal that the maximum likelihood QAE, which does not make use of phase estimation, emerges as the best candidate to demonstrate an advantage over classical Monte Carlo methods in the short term. Numerical integration is a fundamental problem in many areas of science and engineering. These techniques are in fact widely used in computational chemistry [25], [95]. Traditional numerical integration methods can be computationally expensive and require significant computational resources. Monte Carlo methods [96] are exploited for numerical integration problems, as they are flexible and able to handle high-dimensional problems. However, Monte Carlo methods are limited in terms of approximation accuracy depending on the number of samples used in the simulation. Such class of methods is statistical and it suffers from the difficulty of generating a good sampling by starting from an arbitrary probability density function. As a result, Monte Carlo methods can be computationally expensive for high-precision integration, making them less efficient for certain applications [97]. Quantum computing[98], [99] provides an alternative path, more specifically after the introduction of the Quantum Amplitude Estimation

(QAE) algorithm [100]. QAE is a quantum algorithm that can be used to estimate the amplitude of a specific state in a quantum system with quantum speed-up compared to classical algorithms. This allows QAE to be used for efficient numerical integration of complex functions. In particular, QAE has been shown to be able to beat the classical Monte Carlo methods for numerical integration, especially in high-dimensional integration problems [101]. By using QAE, it is possible to estimate the integrals with the same precision by extracting fewer samples, thus significantly reducing the computational resources required for numerical integration. Moreover, QAE can be used to solve a wide range of numerical integration problems, including those in finance [102], physics [103], [104], and machine learning [105]–[108]. It has been shown that QAE can be applied to estimate option pricing [109] in financial derivatives [110], [111], to solve differential equations [112], and to perform molecular simulations [113] and quantum simulation, among others. The QAE algorithm carries a quadratic speedup with respect to classic Monte Carlo techniques. However, such advantage is constrained by the existence of fault-tolerant quantum hardware [114], [115] that does not exist today. In currently available noisy intermediate scale quantum (NISQ) devices [70], the QAE algorithm has not yet shown an advantage in practical cases [116] due to the insufficient coherence times of the physical qubits [117] and because the error carried by the gates. Indeed, the QAE circuit involves 2^M applications of controlled quantum queries and a quantum Fourier transform. Therefore, alternative approaches have been proposed for QAE that involve a lower-depth circuit. Such quantum-classical hybrid algorithms can be divided into two categories, namely the Maximum Likelihood Amplitude Estimation (MLAE) [118] approach and the Iterative QAE [119] approach, respectively.

A major difference consists in the fact that, while the MLAE approach allows to fix the number of quantum queries, in the iterative approach such number is classically computed at each iteration from the result of the previous one. Therefore, the way errors propagate during the runs is not trivial. For this reason, it is necessary to evaluate the performance of such methods on noisy devices to determine which one best realizes its purpose. The performance analysis of such methods has been then divided in two steps. First, the analysis is carried out with the use of a quantum computing simulator. Here, the standard QAE algorithm, the MLAE, the IQAE and the classical Metropolis-Hastings Monte Carlo (MHMC) algorithm are compared. The scaling on the estimation error, the execution time and the depth of the circuits are considered, respectively. A

1D integral is used to exemplify and benchmark, providing a reasonable trade-off between being representative and carrying an addressable circuit depth. All simulations are performed on the quantum circuit simulator *aer_simulation* of Qiskit library, by involving 3 qubits. The second part of the analysis instead involves the use of a quantum hardware with the specific aim of determining the degree of resilience of the methods with respect to the noise of the hardware. For such analysis, the trapped ion device *Harmony* of IONQ is used. The trapped ion qubits are well-known for their superior coherence and fidelity [120]. Such processors have already shown that they can support quantum circuits with depth larger than with other technologies [121]. For this reason, it represents an adequate candidate for benchmarking amplitude estimation solutions. The MLQAE method appears to be a more short-term alternative to the standard QAE for estimating integrals. The Iterative method, on the other hand, is more sensitive to the noise of the quantum processing unit (QPU). Moreover, the Iterative hybrid approach has a depth of the circuits that implements it on average higher. Furthermore, it requires deeper circuits than the MLAE approach in terms of scaling with respect to the estimation error. Instead, the two approaches exhibit a comparable speed-up.

4.2 Quantum speedup over Monte Carlo techniques

Quantum Amplitude Estimation (QAE) [100] is a fundamental quantum algorithm with the potential to achieve a quadratic speedup for many applications that are classically solved through Monte Carlo (MC) simulation [104]. QAE is of particular interest for its exploitation in numerical integration.

4.2.1 Problem statement

Let's consider the following integral of a function $f : \mathbb{R}^d \to \mathbb{R}$.

$$\mu = \int_X g(x)\, dx \tag{4.1}$$

over the integration domain $X \in \mathbb{R}^d$ and let's define a probability distribution $p(x)$ such that $p(x) \neq 0$ for $x \in X$ than the integral can be rewritten as

$$\mu = \int p(x) f(x)\, dx = \mathbb{E}_p[g(x)] \tag{4.2}$$

where $g(x) \equiv f(x)/p(x)$ in X and 0 otherwise. Classically, the Monte Carlo method consists of a set of methods to estimate the integral in the Equation 4.2. Such methods start from the assumption of sampling M values $(x_0, \ldots, x_{M-1})$ from the $p(x)$ and then estimate I with the average estimator.
The basic core of Monte Carlo methods involves estimating the expected output value μ of a randomized algorithm Q that generates samples. The natural algorithm for doing so is to produce k samples, each corresponding to the output of an independent execution of Q, and then to output the average $\tilde{\mu}$ of the samples as an approximation of μ. Assuming that the variance of the random variable corresponding to the output of Q is at most σ^2, the probability that the value output by this estimator is far from the truth can be bounded using Chebyshev's inequality:

$$\Pr\left[|\tilde{\mu} - \mu| \geq \epsilon\right] \leq \frac{\sigma^2}{M\epsilon^2} \tag{4.3}$$

It is therefore sufficient to take

$$M \approx \mathcal{O}\left(\frac{\sigma^2}{\epsilon^2}\right) \tag{4.4}$$

to estimate μ up to additive error ϵ with, say, 99% success probability. This simple result is a key component in many more complex randomized approximation schemes. The estimation error bound of classical MC simulation scales as $\mathcal{O}(1/\sqrt{M})$, where M denotes the number of (classical) samples. Although this algorithm is fairly efficient, its quadratic dependence on σ/ϵ seems far from ideal: for example, if $\sigma = 1$, to estimate μ up to four decimal places, we would need to run A over 100 million times [103]. Unfortunately, it can be shown that without any further information about Q, the sample complexity of this algorithm is asymptotically optimal [122] with respect to its scaling with σ and ϵ, although it can be improved by a constant factor [123]. Here is where quantum computing comes to help. The number of uses of Q can be reduced almost quadratically beyond the classical bound. The result is based on amplitude estimation.

4.2.2 State preparation

probability distribution loading

The first step is to define a loading strategy allowing to encoding of the $p(x)$. This is the state preparedness problem that can be traced back to the article

by Lov Grover and Terry Rudolph [124]. There, the authors discussed how to efficiently create quantum states proportional to functions satisfying certain integrability conditions. Let p be a probability distribution. The state to create is

$$|\psi\rangle = P\,|0\rangle_n = \sum_{i\in\{0,1\}^n} \sqrt{p(x_i)}\,|i\rangle \tag{4.5}$$

where the value of $p(x_i)$ is obtained from discretizing the distribution p.
In a 1-dimensional case, the sample space X is discretized in $N = 2^n$ intervals so that the samples can be identified by $i \in \{0, i, \ldots, N-1\}$ and the corresponding random variable will be identified with the set $\{x_0, x_1, x_2, \ldots, x_{N-1}\}$ where $x_i = x_0 + i\Delta$ and Δ is the spacing. To create the state $|\psi\rangle$ we proceed in K steps from initial state $|0\rangle$ to a state $|\psi_K\rangle = |\psi\rangle$ that approximates $|\psi\rangle$ with $2^K = N$ discretizing intervals. Encoding the values of a probability density $p(x)$ into the amplitudes of an n qubit state is generally an operation that requires a number of gates that grow exponentially with the number of qubits. Specifically, the number of 2-qubit gates is at most $2^{n+1} - 2n$ [125]. In the case of the Equation 4.5 this number is smaller, given that all amplitudes have zero relative phases, but it is still exponential in the number of qubits. Furthermore, Monte Carlo integration is particularly efficient for high dimensional integrals, therefore it is necessary to use quantum states to encode multivariate distributions. Any quantum circuit with some n qubits may be interpreted as preparing a d-dimensional multivariate distribution where each dimension is represented by n_i qubits [101], also called wires hereafter, where i indexes the dimension, and is such that.

$$n = \sum_{i=0}^{d} n_i \tag{4.6}$$

Furthermore, for each dimension, the binary numbers supported are interpreted as real numbers starting at some position $x_l^{(i)} \in \{x_0^{(i)}, x_1^{(i)}, \ldots, x_{N_i-1}^{(i)}\}$ and with equal spacing

$$\Delta^{(i)} \equiv \left| x_l^{(i)} - x_{l-1}^{(i)} \right| \tag{4.7}$$

The final state will be, correspond to the state in the Equation 4.5 for a d-dimension sample space,

$$\begin{aligned} |\psi\rangle &= P\,|0\rangle_{n_1}\,|0\rangle_{n_2}\cdots|0\rangle_{n_d} = P_1\,|0\rangle_{n_1} \otimes P_2\,|0\rangle_{n_s} \otimes \cdots \otimes P_d\,|0\rangle_{n_d} = \\ &= \sum_{i_1=0}^{N_1-1}\sum_{i_2=0}^{N_2-1}\cdots\sum_{i_d=0}^{N_d-1} \sqrt{p_1\left(x_{i_1}^{(1)}\right)}\sqrt{p_2\left(x_{i_2}^{(2)}\right)}\cdots\sqrt{p_d\left(x_{i_d}^{(d)}\right)}\,|i_1\rangle_{n_1}\,|i_2\rangle_{n_2}\cdots|i_d\rangle_{n_d} \end{aligned} \tag{4.8}$$

Weight function loading

Instead of performing measurements from the state $|\psi\rangle$ in the Equation 4.5 to evaluate the integral, let's consider encoding $f(x)$ in the qubit register. Let's consider an operator U_f such that

$$U_f |x\rangle_n |0\rangle = |x\rangle_n \left(\sqrt{1 - \tilde{f}(x)} |0\rangle + \sqrt{\tilde{f}(x)} |1\rangle \right) \tag{4.9}$$

where U_f is an operator over $n+1$-qubit and $\tilde{f}(x)$ is a function obtained by an affine transformation applied on $f(x)$. Indeed a necessary condition is that $\tilde{f}(x) \in [0,1] \ \forall x \in X$ therefore, $\tilde{f}(x) = Af(x) + B \in [0,1]$. Later the inverse affine transform will be performed in classical post-processing to recover the mean of the actual random variable with this shifting and re-scaling since $\mathbb{E}[af(x) + b] = a\mathbb{E}[f(x)] + b$.
In such a case, the state $|x\rangle$ is another way to write the state $|i\rangle$ where i is the decimal number correspondent to the bitstring of n qubit and related to x as follow $i = (x - x_0)/\Delta$. For simplicity, to study the multidimensional case, consider now the state $\left|x^{(d)}\right\rangle$ of the d^{th} dimension and, therefore, only a function $f(x^{(d)})$. The transformation defined in the Equation 4.9 can be decomposed in two steps. In the first step, a circuit denoted R and termed the "quantum arithmetic circuit" is applied to $\left|x^{(d)}\right\rangle |0\rangle_k$ in such a way

$$R \left|x^{(d)}\right\rangle |0\rangle_k = \left|x^{(d)}\right\rangle \left| \arcsin \sqrt{\tilde{f}\left(x^{(d)}\right)} \right\rangle_k \tag{4.10}$$

Where the register of k qubit is used to encode in a bitstring, an approximation of an arcsine function. Therefore R follows the formalism of a quantum oracle

$$|x\rangle |y\rangle \xrightarrow{R} |x\rangle |y \oplus f(x)\rangle \tag{4.11}$$

In the second step, the register of k qubit is then used to control a bank of R_y rotation gates performed on an ancilla qubit (as shown in Figure 4.1)

$$\begin{aligned} &|\Omega\rangle_k R_Y \left(\arcsin \sqrt{\tilde{f}\left(x^{(d)}\right)} \right) |0\rangle = |\Omega\rangle_k \left(\sqrt{1 - \tilde{f}(x^{(d)})} |0\rangle + \sqrt{\tilde{f}(x^{(d)})} |1\rangle \right) \\ &\text{where} \quad |\Omega\rangle_k \equiv \left| \arcsin \sqrt{\tilde{f}\left(x^{(d)}\right)} \right\rangle_k \end{aligned} \tag{4.12}$$

Indeed, Ω is the bitstring that approximates the arcsine of the square root of $\tilde{f}(x^{(d)})$ which means if the j^{th} qubit of the register of k qubits is $|\Omega_j = 1\rangle$ then

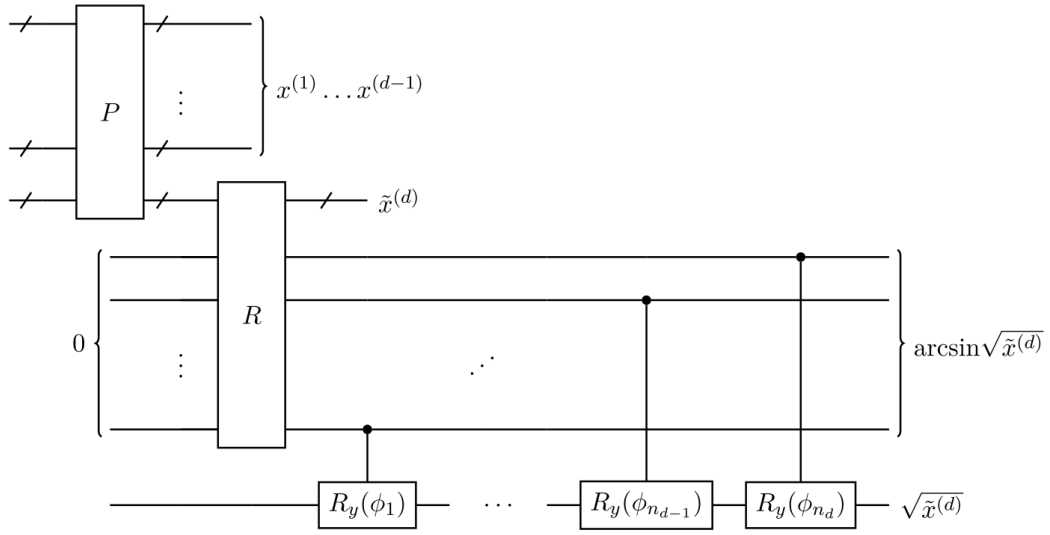


FIGURE 4.1: An illustration of how a probability distribution loading circuit, P, can be supplemented with a quantum arithmetic circuit, R, such that an expectation value of interest is encoded in the amplitude of a qubit.

a rotation of angle $\phi_j = 2^{j+1}$ is applied therefore

$$R_Y\left(\arcsin\sqrt{\tilde{f}\left(x^{(d)}\right)}\right) = \prod_{j=0}^{k-1} R_Y\left(\Omega_j 2^{j+1}\right) \tag{4.13}$$

The problem with this procedure concerns the construction of the circuit that performs the R transformation which is often very expensive in terms of gate and circuit depth. The complexity of this approach depends on f but often will be dominated by the computation of the arcsin , which can be realized using $\mathcal{O}(n^2)$ operations where n is the number of qubits that encode x [126]. Moreover, such a strategy requires ancilla qubits to encode the arcsin .
Let's consider now another strategy to encode $f(x)$ instead of introducing a quantum register to store $|\Omega\rangle$ defined in Equation 4.12. In [116] it is shown how to implement an operator

$$U_P |i\rangle_n |0\rangle = |i\rangle_n \left(\cos p^{(k)}(x_i) |0\rangle + \sin p^{(k)}(x_i) |1\rangle\right) \tag{4.14}$$

where $p^{(k)} : [0,1] \to \mathbb{R}$ is a polynomial of degree k and $\sin p^{(k)}(x_i) \approx p^{(k)}(x_i) + \mathcal{O}\left(x^3\right)$. The corresponding quantum circuits, illustrated in Figure 4.2 for $k = 2$, use polynomially many (multi-controlled) R_Y gates. In particular supposing $k \leq \lceil n/2 \rceil$, U_P requires $\mathcal{O}\left(n^d\right)$ $CNOT$ gates. Consider expanding into a polynomial

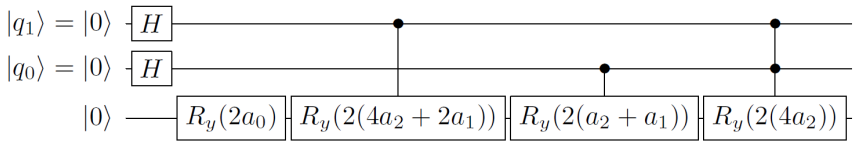


FIGURE 4.2: Circuit preparing a state with amplitudes given by the polynomial $p(x) = a_2x_2 + a_1x + a_0 = (4a_2 + 2a_1)q_1 + (a_2 + a_1)q_0 + 4a_2q_1q_0 + a_0$, for $x = 0, 1, 2, 3$, represented by two qubits. [127]

of degree k, $P^{(k)}(x)$, not of $\tilde{f}$ like in [127] but of $\sqrt{h}$.

$$\sqrt{h(x^{(d)})} \equiv C\sqrt{1 + \tilde{f}(x^{(d)})} \approx P_k(x^{(d)}) + \mathcal{O}((x)^k) \tag{4.15}$$

where $h(x^{(d)}) \in [0, 1]$. In the end, net of the approximations, the final state is obtained, also considering the loading of the probability density,

$$|\psi\rangle = \sum_{i_1,\cdots,i_d} \sqrt{p_{i_0,\ldots,i_d}}\, |x_{i_0} \cdots x_{i_d}\rangle \left(\sqrt{1 - h(x_{i_d}^{(d)})}\, |0\rangle + \sqrt{h(x_{i_d}^{(d)})}\, |1\rangle \right) \tag{4.16}$$

To simplify the notation, such a state, and the relative operators to get it can be written as follows

$$\mathcal{A}\, |0\rangle_n |0\rangle = \sqrt{1 - a}\, |\Psi_0\rangle |0\rangle + \sqrt{a}\, |\Psi_1\rangle |1\rangle \tag{4.17}$$

where $a \in [0, 1]$ is the probability to measure the ancilla qubit in $|1\rangle$ and it is unknown [100]; $|\Psi_0\rangle$ and $|\Psi_1\rangle$ are two normalized states, not necessarily orthogonal [119].

To relate the unknown value a to the integral in the Equation 4.2 one can define

$$|\Psi_1\rangle = \frac{1}{\sqrt{a}} \sum_x \sqrt{p(x)h(x)}\, |x\rangle_n \tag{4.18}$$

therefore $a \sim \mathbb{E}_p[h(x)] = \mathbb{E}_p[C^2(1 + (Af(x) + B))] = \tilde{A}\mathbb{E}_p[f(x)] + \tilde{B}$.

4.2.3 Quantum Amplitude Estimation algorithm

The Quantum Amplitude Estimation (QAE) algorithm takes as input a state as defined in Equation 4.17 prepared to leverage an operator $\mathcal{A}$ which must be designed to use the QAE algorithm. Let's define an operator $\mathcal{Q} = \mathcal{A}\mathcal{S}_0\mathcal{A}^\dagger\mathcal{S}_{\Psi_0}$ where $\mathcal{S}_0 = \mathbb{I} - 2\, |0\rangle_{n+1} \langle 0|_{n+1}$ representing reflection with respect to the state $|0\rangle$ while $\mathcal{S}_{\Psi_0} = \mathbb{I} - 2\, |\Psi_0\rangle_n \langle \Psi_0|_n \otimes |0\rangle \langle 0|$ the reflection with respect to the state $|\Psi_0\rangle_n$. Design $\mathcal{S}_{\Psi_0}$ reflection as a quantum circuit could look difficult but if $|\Psi_0\rangle$

and $|\Psi_1\rangle$ are orthogonal then it can be performed only with a Z gate applied on the ancilla qubit. Applications of $\mathcal{Q}$ are denoted as *quantum samples* or *oracle queries* [119].

The canonical QAE follows the form of quantum phase estimation (QPE): (i) it uses m ancilla qubits, initialized in equal superposition, to represent the final result, (ii) it defines the number of quantum samples as $M = 2^m$ and (iii) it applies geometrically increasing powers of $\mathcal{Q}$ controlled by the ancilla qubits. Eventually, it performs an inverse quantum Fourier transform (QFT) on the ancilla qubits before they are measured. The measurement of m ancilla qubits produces a bit string equivalent to an integer $y \in \{0, \ldots, M-1\}$ and an estimation of a can be defined as $\tilde{a} = \sin^2 \theta_a$ where $\theta_a \equiv y\pi/M$. The algorithm produces an estimation $\tilde{a}$ such that

$$|a - \tilde{a}| \leq \delta \equiv \frac{2\pi\sqrt{a(a-1)}}{M} + \frac{\pi^2}{M^2} \sim \mathcal{O}\left(\frac{1}{M}\right) \tag{4.19}$$

with a probability

$$\begin{aligned} \mathbb{P}[y||a - \tilde{a}| \leq \delta] &= \mathbb{P}[y = \lfloor M\theta_a/\pi \rfloor] + \mathbb{P}[y = \lceil M\theta_a/\pi \rceil] \\ &= \frac{\sin^2(M\Delta)}{M^2 \sin(\Delta)} + \frac{\sin^2(\pi - M\Delta)}{M^2 \sin(\pi/M - \Delta)} \geq \frac{8}{\pi^2} \end{aligned} \tag{4.20}$$

where $M \leq 2$ while Δ is the minimal distance on the unit circle between the angles θ_a and $\pi y/M$. Therefore returning a scaling complexity of $\mathcal{O}(1/M)$ for M applications of quantum samples $\mathcal{Q}$.

The success probability to obtain $\tilde{a}$ with the discrepancy shown in the Equation 4.19 can quickly be boosted to close to 100% by repeating the QAE circuit multiple times and by using the median estimate. Since the success probability of QAE is larger than $8/\pi^2$ we would in principle only need, for instance, 24 repetitions to achieve a success probability of 99.75% [116]. However, current quantum hardware introduces additional errors. In the work [116] the circuit was repeated 8192 times to get a reliable estimate of a.

4.3 Alternative Quantum Amplitude Estimation methods

As discussed, today's hardware limitations prevent one from getting the theoretical speedup. For this reason, various algorithms have been developed involving both classical and quantum devices in order to reduce the resources required

for quantum computation without renouncing a speedup against Monte Carlo methods. The alternative hybrid algorithms can be divided into two categories. The first category (I), to which we will refer with a *Maximum Likelihood Amplitude Estimation (MLAE) approach*, consists of a Grover-like circuit, and a maximum likelihood estimation on a classical processor. Such approach has been discussed by different groups: in Ref. [118] the idea was introduced for the first time, while two variants have been proposed later [128], [129]. The second category (II), under the name of *iterative approach*, consists of an alternating sequence of Grover-like circuits, where the oracle query $\mathcal{Q}$ is applied k times, and a classical algorithm to compute the value k of $\mathcal{Q}$ applications for the next quantum iteration. Such iterative procedure leads to decrease the confidence interval of the estimation $\tilde{a}$ up to the error ϵ with up to $T = \lceil \log_2(\pi/8\epsilon) \rceil$ iterations [119]. Such an algorithm in turn takes inspiration from the variant of the quantum approximation counting described by Aaronson and Rall [130].

MLAE approach

As mentioned earlier, such a quantum-classical hybrid class of algorithms involves both a Grover-like circuit and a Maximum Likelihood Estimation (MLE) [131] on a classical processor, respectively. In the following, the strategy of the algorithm is analyzed. Let $\mathcal{A}$ be an operator such that

$$\mathcal{A} \left|0\right\rangle_n \left|0\right\rangle = \cos\theta_a \left|\Psi_0\right\rangle \left|0\right\rangle + \sin\theta_a \left|\Psi_1\right\rangle \left|1\right\rangle \tag{4.21}$$

The aim of the algorithm is to estimate $a = \sin^2\theta_a$. It is possible to replace QPE with a set of Grover iterations combined with a Maximum Likelihood Estimation (MLE) [118].
Let's perform N_k sampling from the circuit

$$\begin{aligned} &\mathcal{Q}^k \mathcal{A} \left|0\right\rangle_n \left|0\right\rangle = \\ &= \cos\left((2k+1)\theta_a\right) \left|\Psi_0\right\rangle \left|0\right\rangle + \sin\left((2k+1)\theta_a\right) \left|\Psi_1\right\rangle \left|1\right\rangle \end{aligned} \tag{4.22}$$

From the Eq. 4.22 we are able to model the probability distribution of the measurements of the ancilla qubit with $\mathbb{P}[|0\rangle] = \cos^2\left((2k+1)\theta_a\right)$ and $\mathbb{P}[|1\rangle] = \sin^2\left((2k+1)\theta_a\right)$.
The likelihood of such model is

$$L_k(h_k, \theta_a) = \left[\sin^2\left((2k+1)\theta_a\right)\right]^{h_k} \left[\cos^2\left((2k+1)\theta_a\right)\right]^{N_k - h_k} \tag{4.23}$$

where h_k is the number of measurements that have returned 1 and N_k is the total number of measurements.
Next, let's perform the sampling with $M+1$ times with $k = 0, \ldots, M$ in order to compute the total likelihood

$$L(\boldsymbol{h}, \theta_a) = \prod_{k=0}^{M} L_k(h_k, \theta_a) \tag{4.24}$$

Using the maximum likelihood procedure we are able to compute the optimal θ_a such that our model approximates the true distribution as closely as possible

$$\hat{\theta}_a := \arg\max_{\theta_a} L(\boldsymbol{h}, \theta_a) = \arg\max_{\theta_a} \ln L(\boldsymbol{h}, \theta_a) \tag{4.25}$$

Now a and θ_a are uniquely related through $a = \sin^2 \theta_a$ in the range $0 \leq \theta_a \leq \pi/2$ so $\hat{a} = \sin^2 \hat{\theta}_a$, as the function is invertible in such interval. As shown in Ref. [129] a pure MLAE method as just described above brings *less* than a quadratic advantage respect to the classical Monte Carlo. There, it is described an algorithm that exploits the MLAE method, but alternated with the variational optimization step.

Iterative approach

As in the previously discussed approach, the IQAE replaces the standard QAE circuit, with the low complexity circuit defined in Equation 4.22. The difference between the two approaches is that, while in the MLAE method the k times with which $\mathcal{Q}$ is applied ranges from 0 to M, in the iterative approach k is computed by a classical routine which takes as input a confidence interval $[\theta_l, \theta_u]$ for the angle θ_a.
The algorithms described in Refs. [119], [130], [132] belong to the iterative approach class. In the following, the most representative IQAE is described in details [119]. The IQAE uses the quantum computer to approximate, with N_{shots} measure, $\mathbb{P}[|1\rangle] = \sin^2((2k+1)\theta_a)$ for the last qubit in $\mathcal{Q}^k \mathcal{A} |0\rangle_n |0\rangle$ for different powers k. The first step consists to set a confidence interval $[\theta_l, \theta_u] \subseteq [0, \pi/2]$. For convenience we set $[\theta_l, \theta_u] = [0, \pi/2]$.
After that we must define a confidence level $1 - \alpha \in (0, 1)$, a target accuracy $\epsilon > 0$ and a number of shots $N_{shots} \in \{1, \ldots, N_{max}(\epsilon, \alpha)\}$ where

$$N_{max}(\epsilon, \alpha) = \frac{32}{(1 - 2\sin(\pi/14))^2} \log\left(\frac{2}{\alpha} \log_2\left(\frac{\pi}{4\epsilon}\right)\right) \tag{4.26}$$

From the definition of the inputs $(\epsilon, \alpha, N_{shots})$ one can calculate the maximum possible error

$$L_{max}(\epsilon, \alpha, N_{shots}) = \arcsin\left(\frac{2}{N_{shots}} \log\left(\frac{2T(\epsilon)}{\alpha}\right)\right)^{1/4} \tag{4.27}$$

For each iteration i an integer k_i must be defined. In Ref. [119] it is defined a routine called FindNextK which takes the integer k_{i-1} ($k_0 = 0$) and the interval $[\theta_l, \theta_u]$ obtained from the previous iteration. In such core routine of the algorithm, the idea is to define an integer k_i such that $[(4k_i + 2)\theta_l, (4k_i + 2)\theta_u]_{mod 2\pi}$ is fully contained either in $[0, \pi]$ or $[\pi, 2\pi]$. That choice follows the aim of estimating, instead of $\sin^2((2k + 1)\theta_a)$, the quantity $\cos((4k + 2)\theta_a)$ which can be inverted only in $[0, \pi]$ or $[\pi, 2\pi]$.
If FindNextK finds an integer k_i which satisfies the constraints, then one performs the N shots (which depends of N_{shots}) of the circuit $\mathcal{Q}^k\mathcal{A}\left|0\right\rangle_n\left|0\right\rangle$ to estimate $a_i = \mathbb{P}[|1\rangle]$. Otherwise the iteration i must be performed with $k_i = k_{i-1}$.
The next step consists to define an interval $[a_i^{min}, a_i^{max}]$ using different techniques such as the Chernoff-Hoeffding method according to which $a_i^{max} = \min(1, a_i + \epsilon_{a_i})$ and $a_i^{min} = \max(0, a_i - \epsilon_{a_i})$ where

$$\epsilon_{a_i} = \sqrt{\frac{1}{2N} \log\left(\frac{2T}{\alpha}\right)} \tag{4.28}$$

The new confidence interval $[\theta_l, \theta_u]$ is than calculated from $[a_i^{min}, a_i^{max}]$ reversing the formula $a = \cos((4k + 2)\theta_a)$. Such algorithm demonstrates a quadratic speedup with respect to Monte Carlo techniques up to a factor given by N_{max}. The speedup in such case depends by the confidence level which can be arbitrarily set.

QAE methods list							
Algorithm	NISQ-readiness	Qubits	Depth	N_{shots} vs α	Speedup over MC	Ref.	Type
QAE	Low	$n+1+\log 1/\epsilon$	$d \cdot 1/\epsilon + \log\log 1/\epsilon$	$B(p = 0,81, k = 1, n = N_{shots})$ **	$1/\epsilon$	[100]	O
QAE NO-PE *	Medium	$n+1$	$d \cdot 1/\epsilon$	–	$\epsilon^{-4/3}$	[118], [133]	I
VarQAE *	High	$n+1$	$< d \cdot 1/\epsilon$	–	$> \epsilon^{-4/3}$	[129]	I
Power-law AE *	High	$n+1$	$d \cdot (1/\epsilon)^{1-\beta}$	–	–	[128]	I
IQAE	Medium	$n+1$	$d \cdot 1/\epsilon$	N_{max} Eq. 4.26	$N_{max} \cdot 1/\epsilon$	[119]	II
SQAE	Medium	$n+1$	–	–	–	[130]	II
FAE	Medium	$n+1$	$d \cdot 1/\epsilon$	–	$(1/\epsilon) \cdot \ln log(\pi/\epsilon)$	[132]	II

* Classical optimization required
** Binomial distribution

TABLE 4.1: Comparison between QAE algorithms. Here $n+1$ is the number of qubits on which the oracle query $\mathcal{Q}$ is applied, d is the depth of $\mathcal{Q}$ while ϵ is the target accuracy and α the confidence level. In the last two algorithms $\beta \in (0, 1]$, $k \leq 2$ and $q \in [1, \ldots, k-1]$. Types column indicates to which approach the algorithm belongs to, based on the classification of Section 4.3: O corresponds to the original QAE, I to the MLAE approach and II to the iterative respectively

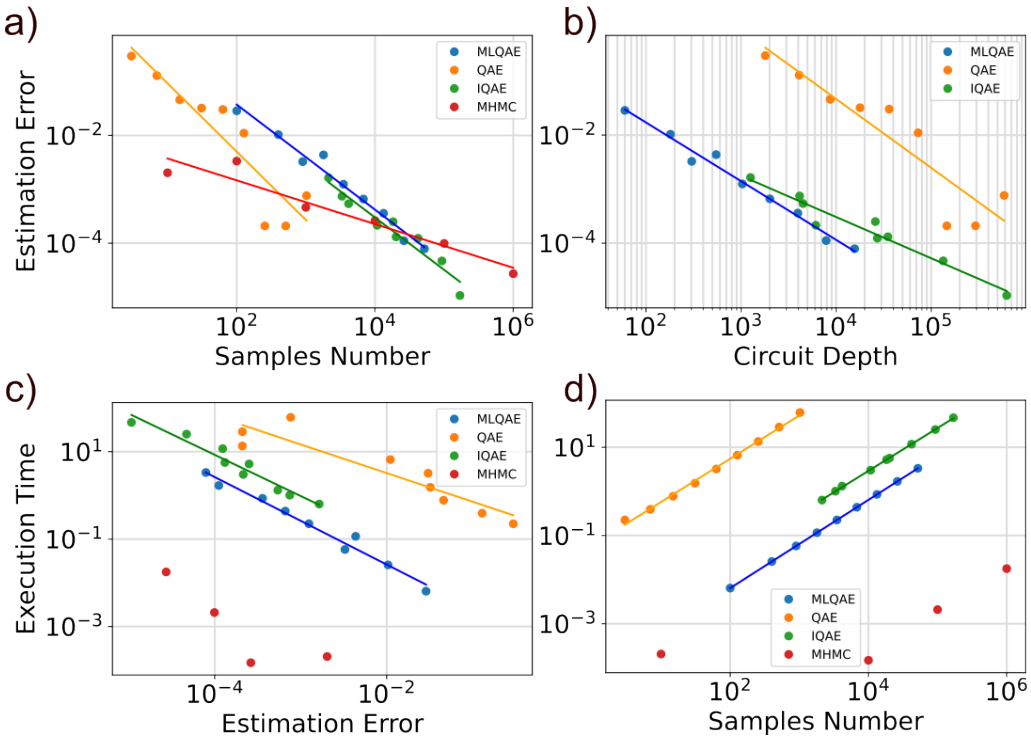


FIGURE 4.3: Performance of the amplitude estimation algorithms to estimate an 1D integral with $n = 3$ qubits. All results of the quantum algorithms were obtained with the local qiskit simulator *aer_simulation* and each plot shows the average of 10 executions with different simulator seeds. All simulations of the IQAE [119] algorithm were performed with a 90% confidence interval. **a)** The top-left plot shows how much the error on estimating the integral decreases as a function of the number of samples/oracle queries. The data are fitted with a function $x^{-\eta}$ in loglog scale. For the algorithms MLQAE [118], standard QAE, IQAE, and classical Metropolis-Hastings Monte Carlo (MHMC) the slopes are respectively of -0.974 ± 0.058, -1.267 ± 0.206, -0.971 ± 0.092 and -0.485 ± 0.051 (result obtained without considering the case with 10 samples). MLQAE and IQAE are performed with 100 shots per quantum circuit while the QAE was executed standalone. The data show quadratic speedup for the QAE (orange) and slightly less than quadratic speedup for the MLQAE and IQAE algorithms (blue and green, respectively) compared to estimation as a classic sampling Monte Carlo method (red). **b)** The top-right plot shows the estimation error as a function of the higher circuit depth (we must consider the highest one because MLQAE and IQAE required the simulation of more than one quantum circuit). The two lower plots, **c)** and **d)**, show instead the execution time (for the quantum algorithms it is the execution time of the QisKit local simulator) in function of the estimation error (left) and the number of samples/oracle queries (right).

4.4 Comparison of the algorithms by statistical analysis

4.4.1 Bench-marking the methods of quantum amplitude estimation

In order to benchmark the effectiveness of algorithms of quantum amplitude estimation, we consider an integral sufficiently representative despite a relatively shallow depth of the quantum circuit required for the implementation on a gate-model quantum computer. More specifically selected function consists of $\sin^2 x$. We therefore consider the following integral

$$\mathcal{I} = \frac{1}{\Delta} \int_0^{\Delta} \sin^2 x \; dx \tag{4.29}$$

which is approximated by the following summation of 2^n samples

$$\mathcal{D} = \sum_{x=0}^{2^n-1} \frac{1}{2^n} \sin^2 \left(\frac{(x+1/2)\,\Delta}{2^n} \right) \xrightarrow{n\to\infty} \mathcal{I}. \tag{4.30}$$

encoded by a qubit register q of n qubits.
The integral $\mathcal{D}$ can be interpreted as the expectation value of the function

$$f(x) = \sin^2 \left(\frac{(x+1/2)\,\Delta}{2^n} \right) \tag{4.31}$$

with an uniform probability distribution $p(x) = 1/2^n$.
The corresponding quantum circuits $\mathcal{P}$ to encode $p(x)$ on the $n+1$ qubit register (consisting in q plus an ancilla qubit) and $\mathcal{R}$ to encode $f(x)$ are

$$\mathcal{P} \left|0\right\rangle_n \left|0\right\rangle = \frac{1}{\sqrt{2^n}} \sum_x \left|x\right\rangle_n \left|0\right\rangle \tag{4.32}$$

$$\mathcal{R} \left|x\right\rangle_n \left|0\right\rangle = \left|x\right\rangle_n \left(\sin \left(\frac{(x+1/2)\,\Delta}{2^n} \right) \left|1\right\rangle + \cos \left(\frac{(x+1/2)\,\Delta}{2^n} \right) \left|0\right\rangle \right) \tag{4.33}$$

respectively [118]. $\mathcal{P}$ can be realized with Hadamard gates, and $\mathcal{R}$ with controlled-Y rotation gates. The methods described in the previous Section aim to reduce the number of qubits and the length of the circuits while maintaining an advantage over the classic Monte Carlo method.
In the Table 4.1 the QAE variants are classified on the basis of the relative NISQ readiness. The number of the qubits and the circuits depth are two parameters to evaluate the NISQ readiness. Standard QAE algorithm results at

low NISQ readiness. However, the algorithm scaling does not depend only on the circuits depth but also on the number of shots N_{shots} and potentially on the number of calls to the QPU. Another parameter to consider is the confidence level α, i.e. the probability of obtaining a result within the desired target error ϵ. In the original QAE the lowest α is 81%, but it can be increased by performing more parallel circuit runs, which means a higher N_{shots}. Other algorithms, like the IQAE, allow one to arbitrarily choose the confidence level therefore there is a correlation between the computational cost and α. All the algorithms described above succeed in estimating an integral with an error that scales as $\mathcal{O}(x^{-\eta})$ where x is the number of samples, η is positive and it has been experimentally evaluated, as follows.

4.5 Experimental test on a trapped-ion quantum computer

4.5.1 Trapped-ion quantum computer used for the experimental test

Trapped ion technology is a promising approach for realizing quantum computing due to its long coherence times and the ability to establish full connectivity between qubits. Among the various trapped ion platforms, IonQ has developed a scalable architecture for building quantum processors with sufficient fidelity and low error rates for the purposes of this study. The technology relies on trapping individual ytterbium ions in a linear array and using laser pulses to manipulate their quantum states. The ions are cooled and trapped in a high vacuum chamber to minimize decoherence. The qubits are encoded in the hyperfine states of the electron and nuclear spins of the ion, which have long coherence times and are immune to certain types of noise. IonQ has made their 11 qubit device called *Harmony* available on the AWS Braket platform [134]. *Harmony* has an average single-qubit gate fidelity of 99.35%, two-qubit gate of 96.02% and state preparation and measurement of 99.3% - 99.8%. The coherence times, instead, are of the order of seconds: $T_1 > 10^7 \mu s$ and $T_2 = 2 \cdot 10^5 \mu s$ The device also features reconfigurable connectivity, allowing for flexible qubit connectivity for various quantum algorithms. Such properties makes it one of the highest-performing quantum processors currently available to the public at the time of this study.

4.5.2 Assessing the performances on a trapped ion device

The Maximum Likelihood Quantum Amplitude Estimation and the Iterative Quantum Amplitude Estimation are algorithms specially designed for noisy quantum computers. Although it is possible to use a simulator for a first evaluation of the performance of these algorithms, as shown in Figure 4.3, some characteristics of these algorithms can make the solutions more or less effective when in the presence of noise. Both being hybrid solutions, the noise present in quantum devices affects the final result in a non-trivial way. Figure 4.4 shows some estimates of integrals performed with such two hybrid techniques. The performance in the presence of noise is specially executed on several cases. It consists of an integral function $F(x)$ at different values of x, as the value of the integral to be estimated affects the noise resilience of these techniques.
Such benchmarking was performed on a 11 trapped ion qubits device (*IONQ-Harmony*). The results were obtained by using 2 qubits (it is therefore a large discretization of the integral since $n = 1$) and without exploiting any error suppression or error mitigation technique. The number of qubits is necessarily reduced because in the case $n = 1$ whatever the number of oracle calls, the length of the circuit remains unchanged. With two qubits it is always possible to compile a circuit into one with a fixed depth. For this reason, such bench-marking scheme is not influenced by the length of the circuits (that are equal for each run) but only by how resilient the hybrid strategy exploited by the two algorithms is to the noise. For a good bench-marking it is necessary to chose the algorithms parameters in order to prevent overload of the device. For such reason the confidence level set for the IQAE should be less than 90%. Otherwise, it achieves an estimation of $\tilde{a} \sim 0.5$ almost always independently by Δ. However, since the focus is to evaluate effective alternative solutions to QAE, a level of confidence not lower than that of the standard QAE (81%) has been chosen. It was necessary also to increase the number of circuits shots as, even if the confident level parameter was fixed to 85% in our case, the noise in the hardware reduces the effective confidence level. If, by using a simulator with a confidence level set to 90%, $N_{shots} = 100$ was a good choice, for the runs on the trapped ion device $N_{shots} = 512$. Also for the MLQAE runs, the effective confidence level is affected by the hardware noise therefore $N_{shots} = 512$.

4.6 Discussion

Let's first consider the results from the simulator. The aim is both to evaluate the scaling of the algorithms, i.e. how the estimation error decreases with

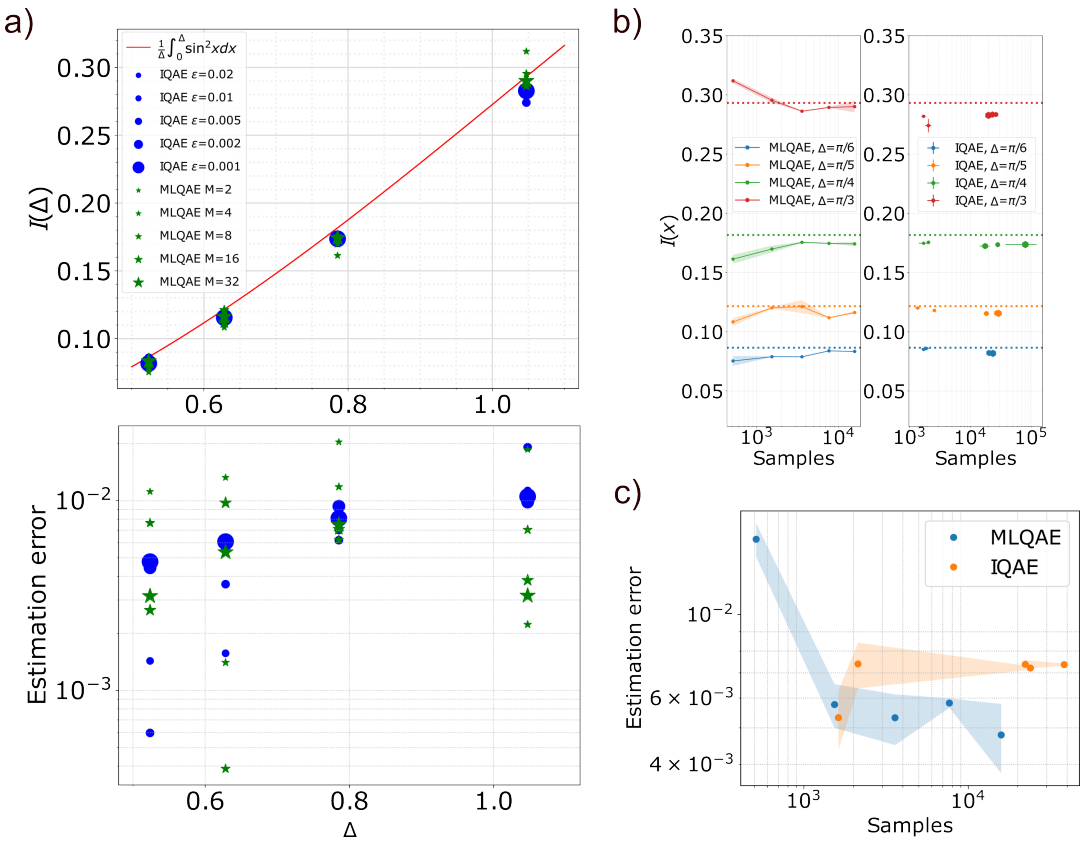


FIGURE 4.4: Performances of MLQAE and IQAE on the 11-qubits *IONQ-Harmony* device. **a)** Estimation of the integral function of Eq. 4.29 at different values of Δ respectively set at $\pi/3$, $\pi/4$, $\pi/5$, and $\pi/6$. The blue dots represent the estimations obtained using IQAE while the green stars the MLQAE. The IQAE runs were performed at confidence level of 85% and $N_{shots} = 512$. The size of the dots increases as the target error ϵ decreases. In the same way, the size of the stars increases as the M increases in the MLQAE runs. Also for the MLQAE runs each circuit was repeated 512 times. Each dot and star represents the average of 5 different runs. If in the upper plot the fit of the results on the target function, the lower plot shows the estimation errors reached by the two algorithms in different settings. **b)** The two plot in this panel shows how the estimation changes as the settings of the algorithms change for the four values of Δ. The right plot for the MLQAE and the left one for IQAE. For the MLQAE the settings correspond to the number of samples represented in the x-axis. For the IQAE case the results have an uncertainty on the number of samples and the different settings are represented by the dot size. **c)** The scaling of the estimation error with respect to the number of samples of both the algorithms. Each dot is the average of all the runs (5 runs for 4 values of Δ for a total of 20 runs). For the IQAE case, only the standard deviation of the estimation errors is shown.

the number of samples, and to evaluate how the length of the circuits scales. From the latter, it is possible to forecast when the two methods will start to provide a benefit over the Monte Carlo method. The simulations are performed by *aer_simulation* of the Qiskit library. From the fit on the data in Figure 4.3 we estimate the theoretically predicted trends within 1.3σ. For the standard QAE algorithm (theoretical $\eta = 1$), we estimate $\eta = 1.267 \pm 0.206$ while for the IQAE and MLAE algorithms we estimate a value compatible with $\eta = 1$ and thus with the QAE algorithm (a trend with $\eta < 1$ or at least slightly less performing than the standard QAE is expected). However, to evaluate the performance, one should consider, at the same time, the average circuit depth since it impacts the viability on modern quantum devices in the NISQ era.
IQAE and MLAE are classical-quantum hybrid algorithms that are therefore compositions of runs of multiple circuits interspersed with classical routines. The Figure 4.3 shows the maximum depths reached by the circuits to perform an estimation of the integral. We note how the two hybrid algorithms IQAE and MLAE - as expected - have a shallower depth than the standard QAE, but in particular the IQAE algorithm needs very long circuits to perform an estimation of the same quality as MLAE. As shown in Figure 4.3, the IQAE approach requires circuits that are also twice as long for an estimate with an error of 10^{-4}. Indeed, even though the number of samples is similar between the two algorithms at the same error ϵ, because of the structure of IQAE, interactions with multiple oracle calls (which are counted as samples) concatenated in the same circuit can happen. Such property is also reflected in the execution time of the algorithms by quantum simulator. It turns out that the MLQAE algorithm is the fastest at the same error of integral estimation ϵ. Such conclusion is general for any number of qubits. The iterative method, as well as the MLQAE, performs on the quantum processor the estimate of the probability $\mathbb{P}[|1\rangle]$ on the ancilla qubit which will depend on the unknown value of the integral and not on the number of qubits dedicated to encoding the integral. The two lower plots in Figure 4.3 allow us to extrapolate the execution times on the simulator of quantum solutions examined in this work. Generating a sample with the Monte Carlo method takes 4 orders of magnitude less time than MLQAE. Around the interval $[10^{-13}, 10^{-11}]$ we could find an advantage in terms of computational time of the MLQAE algorithm. Approximately 10^{24} classical samples are required to achieve precision in such range. Such values are lower if we consider a physical implementation of a quantum computer. If we assume execution of the quantum gates 100 times faster than its simulated counterpart,

then the advantage can be obtained around the interval $[10^{-9}, 10^{-8}]$ with a number of classical samples required around 10^{-17}. In the case of gate execution 1000 times faster than the laptop simulation then this advantage could be in the range $[10^{-7}, 10^{-6}]$ where it would take about 10^{11} classical samples. Further analysis of the resources required by QAE algorithms is reported in Ref.[104], based on a practical example of a multidimensional integral. To achieve a precision of the order of 10^{-3}, 1000 qubits are estimated to be used, combined with long coherence times and high fidelity gates.

We now turn the attention to the experimental results obtained from runs on a noisy quantum computer, in our case of a 1-dimensional integral. For this analysis we specifically chose one of the devices with the highest coherence time and the highest connectivity among those currently available. Both algorithms that should provide a NISQ-friendly alternative to the standard QAE, however turn out to have a workflow that is very susceptible to noise. In fact, all the runs were performed with the same length of the circuits, since at $n = 1$ such circuits have a very limited depth (~ 10). Therefore, the limits of the algorithms are not only linked to the length of the circuits but also to how they propagate the error. The IQAE depends on an iterative process where the number of oracle calls chained in a circuit of an iteration depends on the result of the estimation in the previous iteration. This iterative process is therefore very sensitive to noise as errors propagate from one iteration to another. Indeed, even at 2 qubits it is not possible to find a decreasing trend of the error on the estimates of the integers with the number of quantum samples. At more than two qubits, the circuits begin to grow noticeably with a trend represented in Figure 4.3. With the length of the circuits depending on the results of the previous iteration we can easily understand how even with few qubits this process leads to random results (a random result corresponds to an estimate of $\tilde{a} \sim 0.5$ which corresponds to a complete overlap of the ancilla qubit being measured). However, as from Figure 4.4, already using 2 qubits (therefore $n = 1$) it is already possible to achieve a precision of 10^{-3}. Especially with the MLQAE method we obtain a decreasing trend which leads to average estimates with an error lower than 10^{-3} exceeding 10^3 samples. In general, the MLQAE method exceeds the performance of the IQAE in the presence of noise, confirming the considerations that are followed by the *aer_simulation* runs.

4.7 Conclusions

Using a simulator, we assess the scalability of various integral estimation methods, quantum and classical, with respect to their performance improvements concerning estimation error reduction, sample size, computational time, and quantum circuit length. The execution on a NISQ device shows how such methods behave when the noise is involved. Among the Quantum Amplitude Estimation (QAE) variants, the maximum likelihood QAE, which eschews phase estimation, emerges, compared to the iterative method, as the prime contender for an advantage over classical Monte Carlo methods. Such assertion finds reinforcement in the observed trade-off between quantum circuit length and the accuracy of integral estimation, especially in the presence of noise.

Chapter 5

Quantum computing for ground-state search

5.1 Quantum computing for ground state estimation problem

5.1.1 Quantum phase estimation algorithm

In quantum mechanics, the ground state is the state of the minimum energy accessible by the system. The problem of finding the ground state of a quantum system corresponds to computing the minimum eigenvalue of the associated Hamiltonian operator and the relative eigenstate. Considering an Hilbert space n-dimensional $\mathcal{H}_n$ and a system Hamiltonian H, each quantum state can be written as

$$|\psi\rangle = \sum_n c_n |\phi_n\rangle \quad \text{where} \quad \hat{H}|\phi_n\rangle = E_k |\phi_n\rangle \tag{5.1}$$

Analytically it is possible to retrieve the spectrum of H by applying the Fourier transformation of the expectation value of $\exp\left\{-i\hat{H}t\right\}$.

$$\begin{aligned} &\int \langle\psi|e^{-i\hat{H}t}|\psi\rangle\, e^{i\omega t}dt = \int \langle\psi|\, e^{-i\hat{H}t} \sum_n |\phi_n\rangle\langle\phi_n|\, |\psi\rangle\, e^{i\omega t}dt = \\ &= \sum_n \int \langle\psi|\, e^{-i\hat{H}t}\, |\phi_n\rangle\, c_n e^{i\omega t}dt = \sum_n |c_n|^2 \int e^{-iE_n t} e^{i\omega t}dt = \\ &= \sum_n |c_n|^2 \delta\left(E_n - \omega\right) \end{aligned} \tag{5.2}$$

The final distribution has a support equal to the Hamiltonian spectra with values equal to the probability of finding $|\psi\rangle$ in $|\phi_n\rangle$. The quantum algorithm called Quantum Phase Estimation (QPE) is able to approximate the eigenvalues of a unitary operator following a strategy similar to the calculations in Equation 5.2. Such an algorithm can be used to find the ground state of a quantum

system. The elements to take into consideration to design the algorithm are three

- Implement a quantum circuit that realizes the unitary $U = \exp\{-iH\}$, which is unitary since H is hermitian.
- Prepare a state $|u\rangle \approx |\phi_0\rangle$ (in the description of the algorithm it is assumed to have an oracle capable of preparing a state that approximates $|\phi_0\rangle$)
- Define a register containing t qubits initially set to the state $|0\rangle$. The value of t is determined by two factors: the desired level of accuracy for estimating E_0 and the desired probability of success for the phase estimation procedure.

The state $|u\rangle$ has to be stored on a second register of n qubit while the register of t qubit is necessary to store the state $\left|\tilde{\varphi}^{(t)}\right\rangle = |\varphi_1 \dots \varphi_t\rangle$ where $\varphi \approx 0.\varphi_1 \dots \varphi_t$. It is necessary to specify that, by notation, the eigenvalue of H corresponds to $2\pi\varphi \operatorname{Tr} H$. The normalization term $\operatorname{Tr} H = \sum_i E_i$ allow to have $\varphi \in [0, 1]$. The QPE algorithm is therefore implemented as a circuit that performs the following transformation

$$QPE\,|0\rangle^{\otimes t}\,|u\rangle = \sum_i c_i \left|\tilde{\varphi}_i^{(t)}\right\rangle |\phi_i\rangle \tag{5.3}$$

Finally measuring the first register we obtain a collection of results that fit the distribution in Equation 5.2. If $|u\rangle \approx |\phi_0\rangle$ then $c_i \sim 0\ \forall i \neq 0$ and the QPE circuit returns $\left|\tilde{\varphi}_0^{(t)}\right\rangle |\phi_0\rangle$. Considering the simple case of $|u\rangle = |\phi_0\rangle$, the generalization is easy to obtain, the phase estimation is summarized in two stages. First, a circuit shown in Figure 5.2 is applied. The circuit begins by applying a Hadamard transform to the first register, followed by the application of controlled-U operations on the second register, with U raised to successive powers of two. The final state of the first register is easily seen to be:

$$\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle\,|u\rangle \to \bigotimes_{j=0}^{t-1} \left(|0\rangle + e^{2\pi i 2^j \varphi}\,|1\rangle\right) |u\rangle = \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^k-1} e^{2\pi i \varphi k}\,|k\rangle\,|u\rangle \tag{5.4}$$

The second stage of phase estimation is to apply the inverse quantum Fourier transform on the first register which can be done in $\mathcal{O}(t^2)$ steps.

$$\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^k-1} e^{2\pi i \varphi k}\,|k\rangle\,|u\rangle \xrightarrow{QFT^\dagger} |\tilde{\varphi}\rangle\,|u\rangle \tag{5.5}$$

The third and final stage of phase estimation is to read out the state of the first register by doing a measurement in the computational basis. The final output

allows to reach an n bit precision of the phase φ given by

$$n = t - \left\lceil \log_2 \left(2 + \frac{1}{2\alpha} \right) \right\rceil \tag{5.6}$$

where $1 - \alpha$ is the success probability. If, for example, one wants to obtain a precision equal to chemical precision of $1.6 \times 10^{-3} Ha$ with a probability of 0.5 then $n = \lceil \log_2 (1.6 \times 10^{-3}) \rceil$ and $\alpha = 0.5$, therefore, $t = 11$ qubit are necessary. The computational cost, instead, scales as $\mathcal{O}(2^t)$ (the number of the controlled-U applications) or, in other words, $\mathcal{O}(1/\epsilon)$ where ϵ is the estimation error (indeed $\epsilon \sim 2^{-n} \sim 2^{-t}$).

5.1.2 Variational quantum eigensolver

Today, the quantum computing technology is in its infancy. Quantum computers at this stage of development are called Noisy-Intermediate Scale Quantum (NISQ) devices [70], [135]. There are several indications that these NISQ devices may outperform conventional computers in the near future [136], [137]. Algorithms running on these restricted devices may require only a small number of qubits, show some degree of noise resilience, and are often cast as hybrid algorithms, where some steps are performed on a quantum device and some on a conventional computer. These algorithms, which are often called *NISQ-friendly* have characteristics such as a moderate number of operations, or quantum gates, and good scaling of qubits with respect to the size of the problem. For such reasons, well-known quantum algorithms such as Shor's algorithm for factoring prime numbers, or Grover's algorithm for unstructured search problems, are not suitable. As said before, the best-performing quantum approach to finding eigenvalues is the quantum phase estimation (QPE) algorithm. The QPE algorithm offers an exponential speedup over classical methods and requires a number of quantum operations $\mathcal{O}(1/\epsilon)$ to obtain an estimate with precision ϵ. QPE is not suitable for NISQ devices but there exists alternatives to solve the eigenvalue problem which requires a gate number polynomial with the number of qubits.

The Variational Quantum Eigensolver (VQE) was originally developed by Peruzzo et al. [138], and its theoretical framework was extended and formalized by McClean et al. [139]. The VQE is among the most promising examples of NISQ algorithms. In its most general description, it aims to compute an upper bound for the ground-state energy of a Hamiltonian. It is grounded in the variational principle (and more precisely in the Rayliegh-Ritz functional [140]), which optimizes an upper bound for the lowest possible expectation value of

an observable with respect to a trial wavefunction. Let us consider a quantum system S composed of n qubits and a Hamiltonian H of a different system Q associated with a d-dimensional Hilbert space where $d \leq 2^n$. The interest is to calculate the eigenvalues and eigenstates of H, denoted by E_i and $|\phi_i\rangle$ using S. As mentioned, the VQE method is based on the variational principle of Rayliegh-Ritz, thus considering the expectation value of H

$$\left\langle \hat{H} \right\rangle = \frac{\langle \psi | \hat{H} | \psi \rangle}{\langle \psi | \psi \rangle} \tag{5.7}$$

where is assumed a normalized wave function, $\langle \psi | \psi \rangle = 1$, however, attention should be paid to normalization in the case of leakage errors from the computational basis [139]. From the Rayliegh-Ritz principle, considering the eigenvalue ordered as $E_0 \leq E_1 \leq \cdots \leq E_d$, since

$$\langle \psi | \hat{H} | \psi \rangle \geq E_0 \tag{5.8}$$

it is worth that

$$\min_{|\psi\rangle} \langle \psi | \hat{H} | \psi \rangle = \langle \psi_0 | \hat{H} | \psi_0 \rangle = E_0 \tag{5.9}$$

the expectation value forms an upper bound for the ground-state energy. To be solved with the help of a quantum computer, the optimization problem in Equation 5.9 must be mapped onto the system S defined previously on n qubits. The class of hermitian operators must be restricted to the class of operators whose expectation value can be measured efficiently on S. A sufficient condition for this property is that operators have a decomposition into a polynomial sum of simple operators. Observables suitable for direct measurement on a quantum device are tensor products of spin operators (Pauli operators). Therefore H must be rewritten as

$$\hat{H} = \sum_{i\alpha} h^i_\alpha \hat{\sigma}^i_\alpha + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \hat{\sigma}^i_\alpha \hat{\sigma}^j_\beta + \sum_{ijk\alpha\beta\gamma} h^{ijk}_{\alpha\beta\gamma} \hat{\sigma}^i_\alpha \hat{\sigma}^j_\beta \hat{\sigma}^k_\gamma + \dots \tag{5.10}$$

where the coefficients $\{h^i_\alpha, h^{ij}_{\alpha\beta}, h^{ijk}_{\alpha\beta\gamma}, \dots\}$ are real and the Greek indices identify the Pauli operator while Roman indices identify the qubit on which the operator acts, By exploiting the linearity of quantum observables, it follows that

$$\left\langle \hat{H} \right\rangle = \sum_{i\alpha} h^i_\alpha \left\langle \hat{\sigma}^i_\alpha \right\rangle + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \left\langle \hat{\sigma}^i_\alpha \hat{\sigma}^j_\beta \right\rangle + \sum_{ijk\alpha\beta\gamma} h^{ijk}_{\alpha\beta\gamma} \left\langle \hat{\sigma}^i_\alpha \hat{\sigma}^j_\beta \hat{\sigma}^k_\gamma \right\rangle + \dots \tag{5.11}$$

otherwise the decomposition into Pauli matrices of a Hermitian operator written in Equation 5.10 can be simplified as

$$\hat{H} = \sum_{a}^{\mathcal{P}} h_a \hat{P}_a \quad P_a \in \{\hat{I}, \hat{X}, \hat{Y}, \hat{Z}\}^n \tag{5.12}$$

where the number of Pauli strings $\mathcal{P}$ grows polynomially with the number of qubits. In such a way it is possible to compute in polynomial time $\left\langle \hat{H} \right\rangle$ by measurements from the state S. Estimating an expectation value through direct sampling it therefore involves a cost equal to

$$\mathcal{O}\left(\frac{\mathcal{P}}{\epsilon^2}\right) \tag{5.13}$$

where ϵ is the estimation error of the eigenvalue. In order to translate this minimization task in Equation 5.9 into a problem that can be executed on a quantum computer, one must start by defining a so-called ansatz wavefunction that can be implemented on a quantum device as a series of quantum gates. Such a goal can be achieved using parametrized unitary operations. Therefore $|\psi\rangle$ must be obtained as the application of a parametrized unitary $U(\boldsymbol{\theta})$ to an n-qubits initial state

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta}) \, |0\rangle^{\otimes n} \tag{5.14}$$

where $\boldsymbol{\theta} \equiv (\theta_0, \theta_1, \ldots, \theta_m)$ denotes an m-tuple of real values parameters. The key aspects of the ansatz are its expressibility and trainability. The expressibility defines the ability of the ansatz to span a large class of states in the Hilbert space. That means if such operator ansatz is chosen such that $\forall \, |\psi\rangle \in \mathcal{H}^{\otimes n}$, $\exists \boldsymbol{\theta}$ such that $|\psi(\boldsymbol{\theta})\rangle = |\psi\rangle$ than the VQE problem is therefore writable as

$$E_0 = \min_{\boldsymbol{\theta}} \langle \psi(\boldsymbol{\theta}) | \hat{H} | \psi(\boldsymbol{\theta}) \rangle = \min_{\boldsymbol{\theta}} \sum_{a}^{\mathcal{P}} h_a \langle \psi(\boldsymbol{\theta}) | \hat{P}_a | \psi(\boldsymbol{\theta}) \rangle \tag{5.15}$$

More narrowly the expressibility of the ansatz can be evaluated with the ability to produce the ground state $|\phi_0\rangle$ [141], therefore, there must exist parameters $\overline{\boldsymbol{\theta}}$ such that $\left|\psi(\overline{\boldsymbol{\theta}})\right\rangle = |\phi_0\rangle$. Typically, assuming n qubits, the transformations $U(\boldsymbol{\theta})$ used for VQE are of the type

$$U(\boldsymbol{\theta}) = U_0(\boldsymbol{\theta}_0) U_{ent} U_1(\boldsymbol{\theta}_1) U_{ent} \cdots U_{p-1}(\boldsymbol{\theta}_{p-1}) U_{ent} U_p(\boldsymbol{\theta}_p) \tag{5.16}$$

where $U_l(\boldsymbol{\theta}_l)$, $l = 0, 1, \ldots, p$, corresponds to a single layer of transformations, typically composed by single qubit parameter gates, and U_{ent} is called entanglement map and it makes the n qubits entangled. Instead, the ansatz trainability describes the practical ability of the ansatz to be optimized using techniques tractable on quantum devices.
The parameters of the ansatz used need to be updated iteratively until convergence. In general, this requires sampling the expectation value of the Hamiltonian several times for a given parameter set in the ansatz in order to define an update rule for the parameters (i.e. the updated value of the parameters is a function of the expectation value measured). The choice of optimization is critical for at least three main reasons: (*i*) it directly impacts the number of measurements required to complete an optimization step, as e.g. computing the numerical gradient of a quantum circuit can require value estimation of the Hamiltonian with respect to several slightly modified wave functions (this is also generally true for gradient-free methods) [142]; (*ii*) certain optimizers have been designed to alleviate specific optimization issues, such as the barren plateau problem [143]; (*iii*) it directly impacts the number of iterations required to reach convergence (if it allows for convergence to be reached at all) [144]. The total cost of the VQE algorithm is, therefore, the cost expressed in Equation 5.13 for the expectation value estimation, plus the empirical cost of the classical parameter optimization.

5.2 Combinatorial optimization as ground state search problem

Trainability is therefore the crucial issue regarding the performance of the VQE. To understand if this solution can provide a speed-up compared to other classical solutions to solve an eigenvalue problem. In the following sections, the ability of VQE to solve optimization problems will therefore be explored. Consider a combinatorial optimization problem. Every problem of this type can be expressed as a binary optimization problem

$$\begin{aligned} \min_{\{b\}_n} \quad & C(\{b\}_n) \\ \text{s.t.} \quad & f_i(\{b\}_n) = 0 \\ & g_i(\{b\}_n) \leq 0 \end{aligned} \tag{5.17}$$

in general, a binary function like $C(\{b\}_n)$ can be expanded as

$$C(\{b\}_n) = c_0 + \sum_i c_i b_i + \sum_{ij} c_{ij} b_i b_j + \sum_{ijk} c_{ijk} b_i b_j b_k + \dots \tag{5.18}$$

The optimization problem can be converted back into an unconstrained problem by adding slack variables s_i for inequalities, to transform them into equalities, and penalty terms, with appropriate weights, for the equalities. Therefore it can be written as

$$\min_{\{b\}_n} \tilde{C}(\{b\}_n) = \min_{\{b\}_n} \left(C(\{b\}_n) + \sum_i \lambda_i (f_i(\{b\}_n))^2 + \sum_i \delta_i (g_i(\{b\}_n) + \bar{b}_i)^2 \right) \tag{5.19}$$

and $\tilde{C}(\{b\}_n)$ can also be expanded to a polynomial in binary variables as in Equation 5.18. Such a kind of optimization problem is said Higher-order Unconstrained Binary Optimization problem or HUBO problem. Evaluating the performance of the VQE on these problems allows you to focus only on trainability. Mapping a problem of this type into the VQE formalism consists of designing a Hamiltonian which is essentially classical. This means that, given an n-qubit state $|\psi\rangle$

$$\begin{aligned} \langle\psi|\hat{H}|\psi\rangle &= \sum_{b_0,\cdots,b_{n-1}} \left| c_{b_0,\cdots,b_{n-1}} \right|^2 \tilde{C}(b_0,\cdots,b_{n-1}) \\ |\psi\rangle &= \sum_{b_0,\cdots,b_{n-1}} c_{b_0,\cdots,b_{n-1}} |b_0,\cdots,b_{n-1}\rangle \end{aligned} \tag{5.20}$$

This condition corresponds to having a diagonal Hamiltonian in the computational basis and the eigenvalues of $\hat{H}$ represent all the possible values of $\tilde{C}(\{b\}_n)$. Therefore, $\langle b_0,\cdots,b_{n-1}|\hat{H}|b_0,\cdots,b_{n-1}\rangle = \tilde{C}(b_0,\cdots,b_{n-1})$. The problem is therefore reduced to finding the state of the computational basis corresponding to the bitstring that realizes the minimum value of $\tilde{C}$. The solution of the binary optimization problem is not a superposition of computational basis states and, therefore, it is not necessary to estimate $\left\langle \hat{H} \right\rangle$ to find the ground state eigenvalue of H but just sample the ground state bitstring. The goal is, therefore, to prepare a quantum state of n-qubits, and measure the qubits in the observable $\hat{\sigma}_z$. Indeed a diagonal Hamiltonian can be decomposed into a

combination of σ_z Pauli matrices.

$$\begin{aligned}\hat{H} &= \sum_{b\in\{0,1\}^n} \tilde{C}(b_0,\cdots,b_{n-1})\,|b_0,\cdots,b_{n-1}\rangle\langle b_0,\cdots,b_{n-1}| = \\ &= \sum_{b\in\{0,1\}^n} \tilde{C}(b_0,\cdots,b_{n-1}) \bigotimes_{i=0}^{n-1} \left(\hat{\sigma}_z^{(i)} + (-1)^{b_i}\,\hat{I}^{(i)}\right) = \\ &= h_0 + \sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{ij} h_{ij}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)} + \sum_{ijk} h_{ijk}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\hat{\sigma}_z^{(k)} + \dots\end{aligned} \tag{5.21}$$

Such final results have the same form as the polynomial expansions of a binary function in Equation 5.18. The correspondence between the efficient $\{c\}$ in Equation 5.18 and $\{h\}$ in Equation 5.21 is obtained considering that

$$\left\langle b_0,\cdots,b_{n-1}|\hat{\sigma}_z^{(i_1)}\hat{\sigma}_z^{(i_2)}\cdots\hat{\sigma}_z^{(i_{k\le n})}|b_0,\cdots,b_{n-1}\right\rangle = z_{i_1}z_{i_s}\cdots z_{i_{k\le n}} \tag{5.22}$$

where $z_{i_j} = 1 - 2b_{i_j}$ is called Ising variable, indeed $z \in \{-1,1\}$ that are the eigenvalues of $\hat{\sigma}_z$ for $|b\rangle \in \{|1\rangle, |0\rangle\}$. Finding the ground state of a Hamiltonian with a VQE method is suitable only if $\hat{H}$ can be decomposed in a polynomial number of terms with respect to the number of qubits. If this condition is respected then it is possible to address this problem with the VQE and evaluate its scaling as a solver of combinatorial problems. Quantum computers, in such a case, are used as a sampler with respect to a parametric probability distribution that can be prepared with a polynomial number of operations. The routine performed by a quantum computer consists of sampling qubit state measurements where the qubit state $|\psi_\theta\rangle = U(\theta)\,|0\rangle$ is prepared by a quantum parametric circuit ansatz U of parameters θ. Samples are obtained by preparing the qubit state $|\psi_\theta\rangle$ and measuring the qubits K times so as to obtain a set of samples $X = \{x_k\}_{k=1}^K$, where x_k are bitstrings.

The classical routine has to improve the probability of sampling the minimum energy bitstring. In fact, even considering an ansatz capable of creating a superposition of all the states of the computational basis, $U = Had^{\otimes n}$, it is possible to sample the ground state but with a possibility that decreases exponentially with the number of qubits, and therefore binary variables of the problem. The trainability of the VQE therefore corresponds to the ability to design an ansatz and choose an optimization technique that together are able to amplify the probability amplitude of $|\psi(\boldsymbol{\theta})\rangle$ corresponding to the ground state. The procedure to amplify the amplitude relative to the ground state remains that of

estimating the expectation value of H via samples x_k

$$\langle\psi(\boldsymbol{\theta})|\hat{H}|\psi(\boldsymbol{\theta})\rangle \approx \frac{1}{K}\sum_{k=1}^{K} H_k \tag{5.23}$$

where $H_k = \langle x_k|H|x_k\rangle$, which is the energy corresponded to the sample x_k. Parameters optimization is performed on classical processors and this is done in general using gradient-based methods. For such reason, it is useful to define the unitary $U(\boldsymbol{\theta})$ (i.e. the transformation that is applied on the qubits of the quantum processor) in order to calculate analytically the gradient of our estimate of $\left\langle \hat{H} \right\rangle$ as it is defined in Equation 5.23. The step of expectation value estimation and parameter updates are then repeated until convergence is reached.

5.2.1 Quantum approximate optimization algorithm

One type of quantum state that can be explored as a parametric ansatz is that produced by adiabatic state preparation with a variable path. In adiabatic quantum computation [145] and adiabatic state preparation [146] one makes use of the adiabatic theorem, described in Chapter 2, which states loosely that if one prepares the lowest eigenstate of an initial Hamiltonian $\hat{H}_B$, by continuously changing the Hamiltonian from $\hat{H}_M$ to a final problem Hamiltonian $\hat{H}_C$, one finishes in the lowest eigenstate of $\hat{H}_C$ if the evolution is slow enough. In adiabatic computation, slow enough is quantified relative to the minimum eigenvalue gap between the ground and first excited states along the evolution. While many developments have occurred in the area of adiabatic quantum computation and modifications to the Hamiltonian, perhaps the most commonly considered form of evolution is defined by

$$\hat{H}(s) = A(s)\hat{H}_M + B(s)\hat{H}_c \tag{5.24}$$

where $s \in [0,1]$, $A(0) = B(1) = 1$ and $A(1) = B(0) = 0$. The evolution is controlled by continuously changing the parameter s as a function of time t. Let us now consider how this idea can be translated into the formalism of the VQE algorithm. Consider the set of all paths of $A(s)$ and $B(s)$ from 0 to 1 as a function of time $t \in [0,\tau]$ and denote it $F(\tau)$, where τ is some finite time. Label one such path as $f \in F(\tau)$. In a noiseless coherent situation (quantum closed system), the unitarity of evolution dictates that the final state of the evolution is uniquely determined by the path f. In this situation, the final pure state can be written as a higher-order function of the path f, or $|\psi[f]\rangle$. Thus any

expectation values of the final state may be written as functionals of the path, $\left\langle \hat{H} \right\rangle [f]$, and by the variational principle

$$\left\langle \hat{H}_C \right\rangle [f] = \langle \psi [f] | \hat{H}_C | \psi [f] \rangle \geq E_0 \tag{5.25}$$

such that the optimal path is the path in $F(\tau)$ that minimizes the value of $\langle H \rangle [f]$. This functional minimization may be changed into a standard minimization by parameterizing the path f by a set of parameters θ and performing an optimization on the parameters θ that determine the path. As such, adiabatic state preparation may be considered as an ansatz to be used in the variational quantum eigensolver to find the ground state of a Hamiltonian $\hat{H}_C$, where the state parameters are the shape or nature of the path [147]. In a quantum digital computer, where the VQE algorithms can be run, such an eigenvalue solver based on the adiabatic theorem is called Quantum Approximate Optimization Algorithm or QAOA [148], [149]. QAOA can be seen as a form of VQE with a specific choice of the variational form that is derived from the problem Hamiltonian $\hat{H}_C$. QAOA applies an approximation of an adiabatic evolution, i.e. a quantum dynamic evolution respects the time-dependent Hamiltonian operator. Let be $\hat{H}_C$ and $\hat{H}_M$, also called problem Hamiltonian and mixer Hamiltonian, as 2^n qubit observables, the first step in the QAOA algorithm is to prepare the qubit register in the ground state of $\hat{H}_M$. A typical choice for the mixer Hamiltonian is the following

$$\hat{H}_M = \sum_{i=0}^{n-1} \sigma_x^{(i)}, \qquad \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{5.26}$$

and the ground state is

$$|\psi_M\rangle = \mathrm{Had}^{\otimes n} |0\rangle_n = |+\rangle_n \tag{5.27}$$

The next step is to perform a simulation of the time evolution with respect to the time-depended Hamiltonian

$$\hat{H}(t) = A(t)\hat{H}_M + B(t)\hat{H}_C \tag{5.28}$$

For a time-depended Hamiltonian $H(t)$, the time evolution operator is generalized as

$$U(t_0, t) = \mathcal{T} \exp\left\{ -\frac{i}{\hbar} \int_{t_0}^{t} \hat{H}(t')dt' \right\} \tag{5.29}$$

Such a kind of unitary evolution can be simulated on a quantum circuit discretizing it in p_1 time step with size Δt

$$\int_{t_0}^{t} \hat{H}(t')dt' \approx \sum_{l=0}^{p_1-1} H(l\Delta t)\Delta t \tag{5.30}$$

Since, in general, $\left[\hat{H}(l\Delta t), \hat{H}((l+1)\Delta t)\right] \neq 0$, then the time-evolution operator is applied on the qubits by the Trotter approximation

$$\exp\left\{-i\sum_{l=0}^{p_1-1}\hat{H}(l\Delta t)\Delta t\right\} = \lim_{p_2\to\infty}\left[\prod_{l=0}^{p_1-1}\exp\left\{-i\hat{H}(l\Delta t)\frac{\Delta t}{p_2}\right\}\right]^{p_2} \tag{5.31}$$

where is assumed $\hbar = 1$.
However, from the Equation 5.28, $\hat{H}(l\Delta t) = A(l\Delta t)\hat{H}_M + B(l\Delta t)\hat{H}_C$ where, in general $\left[\hat{H}_M, \hat{H}_C\right] \neq 0$. Still using the Trotter-Suzuki formula

$$\exp\left\{-i\hat{H}(l\Delta t)\frac{\Delta t}{p_2}\right\} = \lim_{p_3\to\infty}\left[\exp\left\{-i\frac{A(l\Delta t)\Delta t}{p_2 p_3}\hat{H}_M\right\}\exp\left\{-i\frac{B(l\Delta t)\Delta t}{p_3 p_2}\hat{H}_C\right\}\right]^{p_3} \tag{5.32}$$

Unificando le Equaizoni 5.30, 5.31 and 5.32 it is possible to approximize $U(t_0, t)$ as

$$\lim_{p\to\infty}\prod_{i=0}^{p-1}\left(e^{-i\beta_i\hat{H}_M}e^{-i\gamma_i\hat{H}_C}\right) \tag{5.33}$$

where $\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{R}^p$ are vectors of parameters. The variational form of QAOA is constructed with a layer of Hadamard gates, followed by two alternating unitaries

$$U_C = e^{-i\gamma\hat{H}_C}\ ,\ \ U_M = e^{-i\beta\hat{H}_M} \tag{5.34}$$

For a given depth $p \in \mathbb{N}$ the variational form is thus defined as

$$U(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \left[\prod_{i=1}^{p} U_B(\beta_i)U_C(\gamma_i)\right]\mathrm{Had}^{\otimes n} \tag{5.35}$$

This yields the trial wave function

$$|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle = U(\boldsymbol{\beta}, \boldsymbol{\gamma})\,|0\rangle^{\otimes n} \tag{5.36}$$

One can easily design a quantum circuit that re-eliminates the operator $U(\boldsymbol{\beta}, \boldsymbol{\gamma})$ if the conditions on the Hamiltonians H_M and H_C are taken into account. To implement U_M and U_C as operators acting on a qubit register, the Hamiltonians $\hat{H}_M$ and $\hat{H}_C$ must be decomposed in a sum of composed Pauli matrices. The

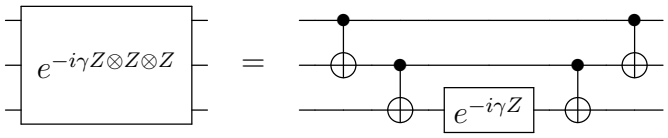
two Hamiltonians can be chosen so as not to incur further trotterization. The initial choice of the mixer Hamiltonian, shown in Equation 5.26, guarantees a direct implementation on a register of n qubits

$$U_M |+\rangle^{\otimes n} = \bigotimes_{i=0}^{n-1} R_X(\beta) |+\rangle_i \tag{5.37}$$

Regarding $\hat{H}_C$, the QAOA algorithm is mainly used to solve combinatorial optimization problems where $\hat{H}_C$ is diagonal and is therefore a sum of terms in Pauli matrices Z that commute between each other. The unitary matrix U_C is therefore diagonal itself and it adds only relative phases, in fact in this case it is also called phase operator, in contrast with the mixer operator U_M which has terms out of the diagonal and which therefore mixes elements of the computational base. If we want to understand the QAOA as a special case of the VQE, the role of U_C is exactly that of an entanglement map. It should be clarified how to implement a unitary how in a quantum circuit

$$\exp\{i\gamma \hat{\sigma}_z^{i_1} \hat{\sigma}_z^{i_2} \cdots \hat{\sigma}_z^{i_k}\} \tag{5.38}$$

using only single and two-qubit gates. The following circuit shows this compilation for a 3-qubit



in general, it requires $2(k-1)$ CNOT gates.

5.2.2 CVaR Optimization

VQE and QAOA minimize the expectation of the problem Hamiltonian for a parameterized trial quantum state. The expectation is estimated as the sample mean of a set of measurement outcomes, while the parameters of the trial state are optimized classically.

Instead of aggregating the samples to estimate the expectation value of the Hamiltonian, good results have been obtained by exploiting the Conditional Value-at-Risk as an aggregation function [150]. Formally, the CVaR of a random variable X for a confidence level $\alpha \in (0, 1]$ is defined as

$$\mathrm{CVaR}_\alpha(X) = \mathbb{E}[X | X \leq F_X^{-1}(\alpha)] \tag{5.39}$$

where F_X denotes the cumulative density function of X. In other words, CVaR is the expected value of the lower α-tail of the distribution of X.
Without loss of generality, assume that the samples $\mathcal{H}_k = \mathcal{H}(x_k)$ are sorted in non decreasing order, i.e. $\mathcal{H}_{k+1} \leq \mathcal{H}_k$. Then, the CVaR_α is defined as

$$\frac{1}{\lceil K\alpha \rceil} \sum_{k=1}^{\lceil K\alpha \rceil} \mathcal{H}_k \tag{5.40}$$

and it is a generalization of both the sample mean ($\alpha = 1$) and the best observed sample ($\alpha \to 0$). It is clear that this can be applied to both VQE and QAOA, simply by replacing the sample mean in Equation 5.23 with CVaR_α in the classical optimization algorithm. The CVaR could give a reasonable benefit. Suppose $|\phi_0\rangle$ ë is the ground state, and $|\phi_1\rangle$, $|\phi_2\rangle$ and $|\phi_3\rangle$ are first, second and third excited state. Define $|\psi_A\rangle = (|\phi_0\rangle + |\phi_3\rangle)/\sqrt{2}$ and $|\psi_B\rangle = (|\phi_1\rangle + |\phi_2\rangle)/\sqrt{2}$. Suppose the energy level are equally separated, then $\langle\psi_A|H|\psi_A\rangle = \langle\psi_B|H|\psi_B\rangle$, therefore the optimizer will encounter a gradient plateau. However for our purpose, we would like to obtain the ground state, therefore $|\psi_A\rangle$ is better than $|\psi_B\rangle$ in practice. The CVaR could help with this problem by emphasizing the distribution $|\psi_A\rangle$. The CVaR emphasizes the best-observed samples and leads to a smooth objective function without introducing local minimum [150]. Also, the implementation of CVaR is relatively straightforward. Since CVaR throws away some of the samples, the accuracy of the estimation decreases. In order to get the same accuracy, the sampling number needs to be increased. The same amount of samples should be involved in the calculation.

5.3 Optimize a water crystal lattice with quantum computing algorithms

The problem of achieving the minimum energy of a chemical system can be reformulated as a combinatorial optimization problem. Over the years, however, molecular dynamics has proven to be a more effective tool for studying molecular systems because of the computational cost involved in combinatorial computation. With quantum computers, people have begun to think that studying a molecular system combinatorially may be more advantageous instead. To date, quantum devices are not yet able to handle long calculations due to high noise but there are algorithms such as the Variational Quantum Eigensolver (VQE) and the Quantum Approximate Optimization Algorithm (QAOA) that allow to address combinatorial optimization problems through the joint use of classical

and quantum processors. In this paper we will use variations of these two quantum optimization algorithms to compute the ground state of a lattice of water molecules interacting with each other through the formation and destruction of hydrogen bonds. The system is then completed with the addition of an ion that interacts with the electric dipoles of the water molecules.

After introducing the quantum methods that will be tested, in this section the molecular system used as a benchmarking problem will be presented. The point is to map the system's Hamiltonian into an n qubit Hamiltonian. The final system that will be used to test variational quantum methods is the result of a series of complications of a basic starting system described in [151]. In summary, the system in [151] is composed of water molecules arranged as lattice particles consisting of a single oxygen atom at the center of a site and two hydrogen atoms on each side. The internal state of the system, such as the dipole moment at a site, is defined with respect to the location of the hydrogen atoms at the site depending on their role in hydrogen bonds (H bonds) being a donor or an acceptor. To achieve the minimum energy of these systems via the VQE and QAOA algorithms we have rewritten the water lattice Hamiltonian into an Ising Hamiltonian since from an Ising system it is easy to write a diagonal Hamiltonian operator with respect to the computational basis of the qubits of the device.

5.3.1 2D square lattice

Let's define a system of $N = N_r N_c$ water molecules arranged on a 2D square lattice, with N_r rows and N_c columns, and having a lattice constant a. For simplicity, we consider $a = 1$.
Each lattice site has an Oxygen atom at the center and two Hydrogen atoms. We assume the Hydrogen atoms occupy the consecutive sites only at each lattice points so that we have a non-zero dipole moment for each water molecule. Thus at each site (i, j) there is one Hydrogen along the row and one along the column denoted by superscripts x and y respectively. Each water molecule acts like a dipole given by the vector

$$\vec{\mu}_{i,j} = \frac{\mu_0}{\sqrt{2}} \left(\sigma^x_{i,j}, \sigma^y_{i,j}\right) \tag{5.41}$$

where μ_0 is the permanent dipole moment of water. The Hydrogen occupancy is defined in terms of the Ising-like states

$$\sigma_{i,j}^{x,y} = \begin{cases} +1 & \text{hydrogen atom left or top of oxygen atom at site } (i,j), \\ -1 & \text{hydrogen atom right or bottom of oxygen atom at site } (i,j) \end{cases} \tag{5.42}$$

Each water molecule interacts only with its nearest neighbor as shown in Figure 5.1. Since we consider finite number of water molecules we use the periodic boundary conditions to mimic bulk water.
The interactions are repulsive R if the two nearest sites are occupied by Hydrogen atoms and attractive (Hydrogen bond) ϵ_H if one of the two nearest sites are unoccupied. The Hydrogen bond attractions are about 3 to 4 times stronger than the repulsions between the Hydrogen sites. The dimensionless Hamiltonian H can be written as follows

$$\begin{aligned} \beta H &= \frac{\epsilon_H}{2} \sum_{(i,j)}^{N-1} (\sigma_{i+1,j}^x \sigma_{i,j}^x + 1) + \frac{R}{2} \sum_{(i,j)}^{N-1} (\sigma_{i+1,j}^x \sigma_{i,j}^x - 1) \\ &+ \frac{\epsilon_H}{2} \sum_{(i,j)}^{N-1} (\sigma_{i,j+1}^y \sigma_{i,j}^y + 1) + \frac{R}{2} \sum_{(i,j)}^{N-1} (\sigma_{i,j+1}^y \sigma_{i,j}^y - 1) \\ &= J/2 \sum_{(i,j)}^{N-1} (\sigma_{i+1,j}^x \sigma_{i,j}^x + \sigma_{i,j+1}^y \sigma_{i,j}^y) + N(\epsilon_H - R) \end{aligned} \tag{5.43}$$

where $J = \epsilon_H + R$ is the strength of the interactions. We choose a smaller value for the Hydrogen bond strength ϵ_H as and $-2k_BT$ and repulsions R as $0.5k_BT$ respectively.

Ion interaction

We introduce an ion to the system. At each water site we get an additional energy contribution given by a dipole-ion interaction

$$-\vec{\mu}_{i,j} \cdot \vec{E}_{i,j} = -\frac{\Gamma}{(r(i,j) - R_0)^2} \sigma_{i,j} \tag{5.44}$$

where $r(i,j)$ is the position of the ion at the water site (i,j) and R_0 is the position of the ion. The ion-dipole interactions are long-ranged r^{-2}, hence we need to sum over the periodic images of the ion. The dimensionless Hamiltonian

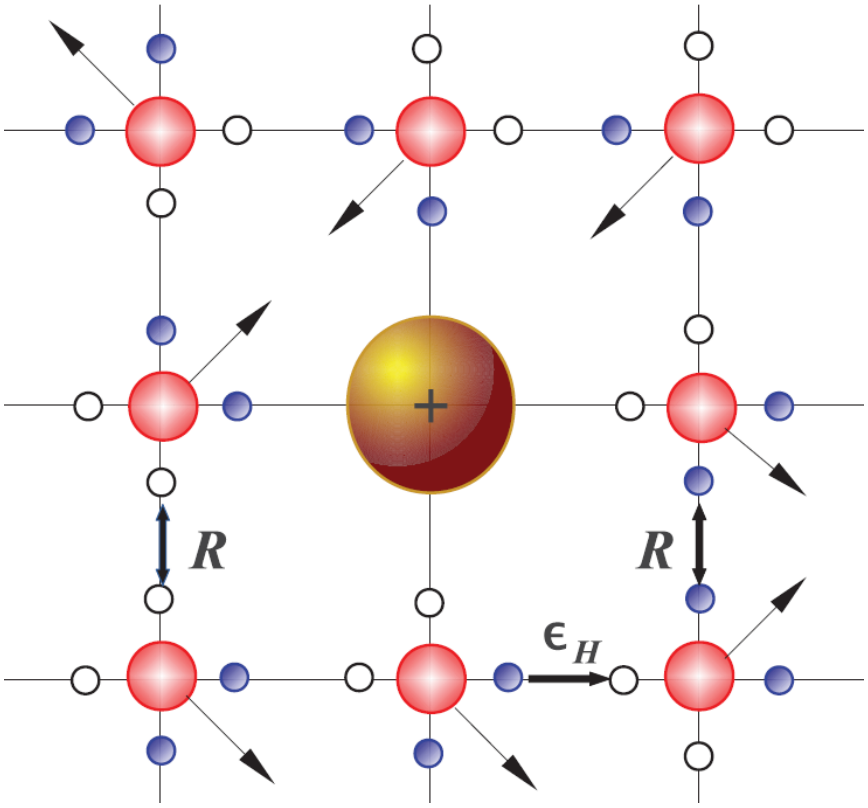


FIGURE 5.1: Illustration of 8 water molecules arranged in a 3×3 square lattice with a positive ion in the center. The oxygen atom is positioned at the center of each site of the lattice while the hydrogen atoms (in blue) occupy two out of four positions (the other two unpaired positions are in white) for a total of 4 configurations identified by the vector of the electric dipole.

of the system is

$$\begin{aligned}
\beta H =& \frac{\epsilon_H}{2} \sum_{(i,j)}^{N-1} (\sigma^x_{i+1,j}\sigma^x_{i,j} + 1) + \frac{R}{2} \sum_{(i,j)}^{N-1} (\sigma^x_{i+1,j}\sigma^x_{i,j} - 1) + \\
&+ \frac{\epsilon_H}{2} \sum_{(i,j)}^{N-1} (\sigma^y_{i,j+1}\sigma^y_{i,j} + 1) + \frac{R}{2} \sum_{(i,j)}^{N-1} (\sigma^y_{i,j+1}\sigma^y_{i,j} - 1) + \\
&- \sum_{(i,j)}^{N-1} \frac{\Gamma}{(|i-i_I|^2 + |j-j_I|^2)^{3/2}} (|i-i_I|\sigma^x_{i,j} + |j-j_I|\sigma^y_{i,j}) = \\
&= J/2 \sum_{(i,j)}^{N-1} (\sigma^x_{i+1,j}\sigma^x_{i,j} + \sigma^y_{i,j+1}\sigma^y_{i,j}) - \Gamma \sum_{(i,j)}^{N-1} (g_x(i,j)\sigma^x_{i,j} + g_y(i,j)\sigma^y_{i,j}) + \\
&+ (N-1)(\epsilon_H - R)
\end{aligned} \tag{5.45}$$

where $g_x(i,j) = |i-i_I|/(|i-i_I|^2 + |j-j_I|^2)^{3/2}$ and $g_y(i,j) = |j-j_I|/(|i-i_I|^2 + |j-j_I|^2)^{3/2}$ are the strength of the interactions.

5.3.2 Hexagonal lattice

Water molecules can be rearranged into a 2-D hexagonal lattice. A hexagonal lattice is a composition of two triangular lattices A and B. To represent the dipole of the water molecules we must define two sets of 2-D coordinates for both the lattices A and B as in it is shown in Figure 5.3 .
Unlike the case of a square lattice where, for each water molecule, the dipole μ is expressed simply in the basis $\vec{\sigma}_1 = (1, 0)$ and $\vec{\sigma}_2 = (0, 1)$, in a hexagonal lattice is better to define μ in two different sets of basis, one for the lattice A and one for B. The hydrogen atoms in the square lattice are positioned in 4 directions

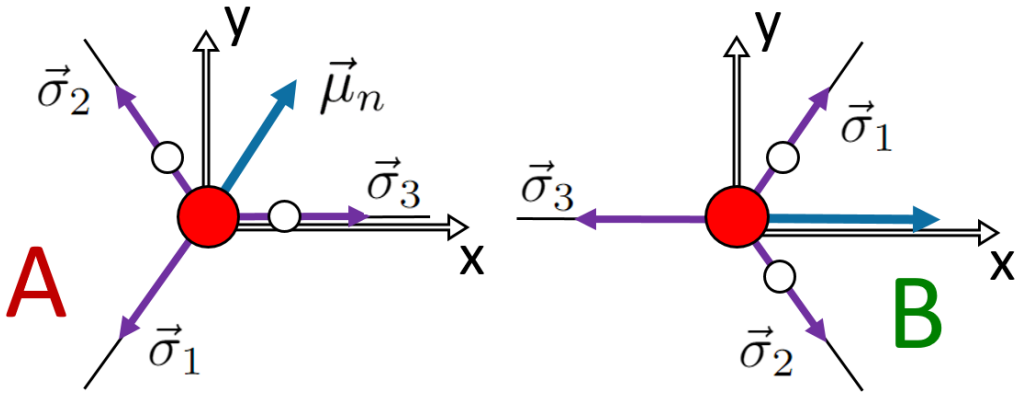


FIGURE 5.2: Illustration of the two different settings of the two lattices representing the hexagonal one. For each molecule, the Cartesian coordinates x and y are defined with axes originating at the lattice site (oxygen atom). The lattice of type A has the unit vector of the direction 3, i.e. $\vec{\sigma}_3$ along the x axis while for the lattice B it is opposite to x. Two bases $\{\vec{\sigma}_1, \vec{\sigma}_2\}$ are then defined to construct the electric dipole vector.

which are defined by the vectors $\vec{\sigma}_1$, $\vec{\sigma}_2$ and by their opposites $-\vec{\sigma}_1$, $-\vec{\sigma}_2$. In the case of the hexagonal lattice, however, the hydrogen atoms are positioned in directions that are not orthogonal to each other. These three directions are identified by the set of vectors $\{\vec{\sigma}_1, \vec{\sigma}_2, \vec{\sigma}_3\}$ associated with each molecule. For molecules in the A lattice, a new choice is the following

$$\vec{\sigma}_1 = \begin{pmatrix} -\cos(\pi/3) \\ -\sin(\pi/3) \end{pmatrix}, \quad \vec{\sigma}_2 = \begin{pmatrix} -\cos(\pi/3) \\ \sin(\pi/3) \end{pmatrix}, \quad \vec{\sigma}_3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = -(\vec{\sigma}_1 + \vec{\sigma}_2) \quad (5.46)$$

For the molecules in B the triad of vectors is obtained by mirroring the vectors of the set A along the first and the second coordinate.

$$\vec{\sigma}_1 = \begin{pmatrix} \cos(\pi/3) \\ \sin(\pi/3) \end{pmatrix}, \quad \vec{\sigma}_2 = \begin{pmatrix} \cos(\pi/3) \\ -\sin(\pi/3) \end{pmatrix}, \quad \vec{\sigma}_3 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} = -(\vec{\sigma}_1 + \vec{\sigma}_2) \quad (5.47)$$

The Figure illustrates sets of vectors $\{\vec{\sigma}_1, \vec{\sigma}_2, \vec{\sigma}_3\}$ for one molecule in the lattice A and for one in B. From this set of vectors, it is possible to define the direction of the electric dipole of all the molecules of the hexagonal lattice as a function of binary variables.

$$\vec{\mu}_n/\mu_n = a\left(\sigma_{1,n}\vec{\sigma}_1 + \sigma_{2,n}\vec{\sigma}_2 + \sigma_{3,n}\vec{\sigma}_3\right) \tag{5.48}$$

where a is a normalization factor. From Equation 5.46 and Equation 5.47, the electric dipole can be written for all molecules as

$$\vec{\mu}_n/\mu_n = -\frac{1}{2}\left(\left(2\sigma_{1,n} + \sigma_{2,n} + 1\right)\vec{\sigma}_1 + \left(2\sigma_{2,n} + \sigma_{1,n} + 1\right)\vec{\sigma}_2\right) \tag{5.49}$$

where $\sigma_{1,n}, \sigma_{2,n} \in \{-1, 1\}$ and $a = -1/2$. The presence of a hydrogen atom along the direction $\vec{\sigma}_1$ corresponds to $\sigma_1 = 1$ while a hydrogen along $\vec{\sigma}_2$ corresponds to σ_2 =1. Given that the available directions are 3 then it is not possible to have at least one hydrogen atom along one of the two directions $\vec{\sigma}_1$, $\vec{\sigma}_2$. So the three possible configurations of the two binary variables are $(-1, 1)$, $(-1, 1)$ and $(1, 1)$. The configuration $(1, 1)$ corresponds to having a electric moment aligned with $\vec{\sigma}_3$ but in the opposite direction. Now that the configurations of

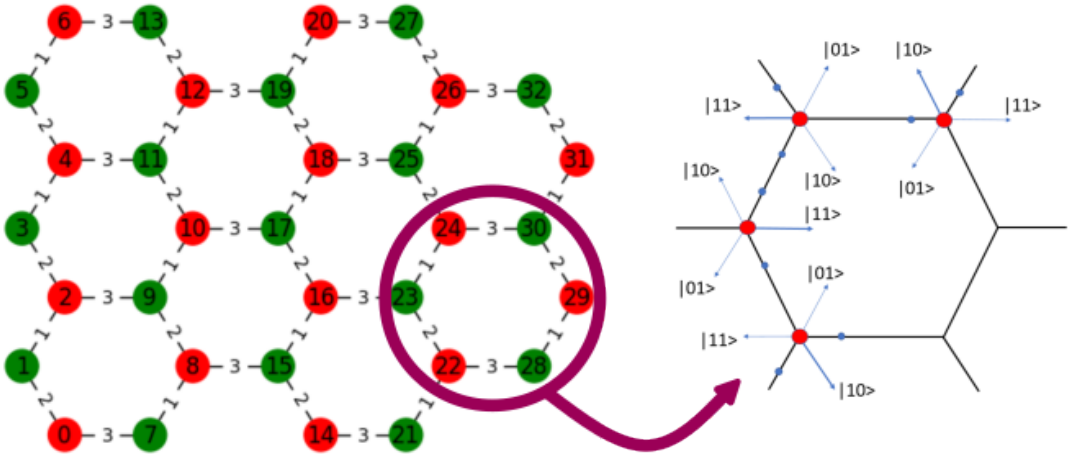


FIGURE 5.3: Example of a hexagonal lattice with 33 water molecules, left side, and configurations of molecules in a hexagon, right side. Each molecule in the A lattice, in red, has as prime neighbors only molecules in the B lattice, in green. Each edge is labeled with a number (1, 2, or 3) based on which direction the two molecules are aligned. Each molecule, having 3 possible orientations, can have a configuration which, in binary variables, can be encoded with 01, 02, and 11. This is a type of dense encoding, as opposed to "*one-hot*" encoding which would require 3 variables.

the molecules in the lattice have been mapped into binary variables, it is possible to introduce the terms of the Hamiltonian concerning the hydrogen bonds and the repulsions between the adjacent molecules in the lattice. Given two adjacent molecules with index n and m, the interaction $h_\alpha(n,m)$ can be of three types and are indicated with the index $\alpha \in \{1,2,3\}$ which represents which binary variable is involved.

$$\beta h_\alpha(n,m) = -\frac{\epsilon_H}{2}\left(\sigma_{\alpha,n}\sigma_{\alpha,m} - 1\right) + \frac{R}{2}\left(\sigma_{\alpha,n}\sigma_{\alpha,m} + 1\right) \tag{5.50}$$

where $\sigma_{1,n} \in \{-1,1\}$, $\sigma_{2,n} \in \{-1,1\}$ and

$$\sigma_{3,n} \equiv -\left(\sigma_{1,n} + \sigma_{2,n} + 1\right) \in \{-1,1\} \tag{5.51}$$

The Figure 5.3 shows an example of hexagonal lattice where each coupling is labeled by $\alpha \in \{1,2,3\}$ and compared with the Figure 5.3 it can be understood that the term $h_1(n,m)$ indicates whether or not a hydrogen bond is present between the molecules n and m which are aligned along the direction $\vec{\sigma}_1$ and so also for $h_2(n,m)$ and $h_3(n,m)$. To write the complete Hamiltonian for a lattice of a generic number of molecules, it is necessary to define, for each molecule, a term for each first neighbor (with a maximum of 3 first neighbors). For notational compactness, it is useful to define an adhesion matrix $\mathcal{A}$ with elements

$$\mathcal{A}_{nm} = \begin{cases} \alpha & \text{if } n \text{ and } m \text{ are neighborhood aligned along the direction } \alpha, \\ 0 & \text{otherwise} \end{cases} \tag{5.52}$$

Therefore the total Hamiltonian is

$$\beta H_{HB} = \sum_{\alpha=1}^{3} \sum_{n,m=0}^{N} \delta_{\alpha,\mathcal{A}_{n,m}} \left(\frac{R-\epsilon_H}{2} \sigma_{\alpha,n}\sigma_{\alpha,m} + \frac{R+\epsilon_H}{2} \right) \tag{5.53}$$

Each molecule of the lattice A is involved in three Hamiltonian terms of all the three kinds of interactions defined in Equation 5.50 and otherwise for the molecules in B, the molecules n and m cannot belong to the same lattice and this is a necessary condition to have $\delta_{\alpha,\mathcal{A}_{n,m}} \neq 0$.

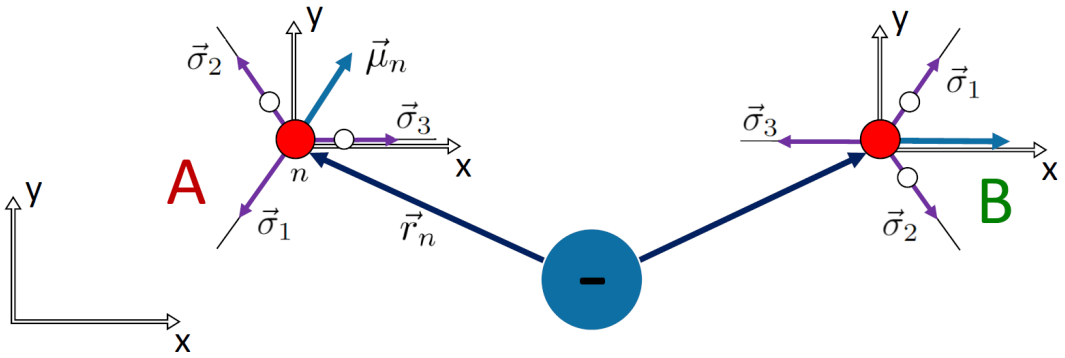


FIGURE 5.4: Illustration representing the mathematical elements involved in the calculation of an interaction between water molecules arranged in a hexagonal lattice and an ion at a generic point of a 2-dimensional station. In addition to the Cartesian coordinates assigned to each molecule to define the electric dipole, global coordinates are introduced to calculate the distance between the ion and the lattice sites.

Ion interaction

Let's introduce the interaction with an ion as defined in Equation 5.44. The ion-dipole interaction is given by the following formula

$$h_{I,n} = -k\frac{|q|\vec{r}_n \cdot \vec{\mu}_n}{r^3} \tag{5.54}$$

where Γ is a factor equal to the charge of the ion multiplied by a constant that can be considered equal to 1. To compute the distance r, with the associated radius vector $\vec{r}_n$, between a molecule n and the ion, it is necessary to associate a vector of Cartesian coordinates (x_n, y_n) to each site of the lattice. Considering how the electric dipole of a single molecule was defined in Equation 5.49, the resulting term is the following

$$\begin{aligned} \beta H_I &= \sum_{n=1}^{N} h_I(n) \\ h_I(n) &= \frac{\Gamma}{2}\frac{1}{r^3}\Big((2\sigma_{1,n} + \sigma_{2,n} + 1)\,\vec{r}_n \cdot \vec{\sigma}_1 + (2\sigma_{2,n} + \sigma_{1,n} + 1)\,\vec{r}_n \cdot \vec{\sigma}_2\Big) \end{aligned} \tag{5.55}$$

Figure 5.4 illustrates an example with a negative ion interacting with a molecule in the A lattice and one in the B lattice.

5.3.3 Hamiltonian operator mapping

At this point the lattice Hamiltonian should be rewritten as a qubit Hamiltonian. The Hamiltonian already has a form reminiscent of the Ising form and it

is therefore straightforward to map it. Since each water molecule has 4 different configurations for a square lattice and 3 for the hexagonal one, two qubits must be defined for each molecule, one for the $\vec{\sigma}_1$ direction and one for $\vec{\sigma}_2$.
For simplicity, in the square lattice the direction x will be indicated with the index 1 while y with 2 to standardize the notation with the case of the hexagonal lattice. For both the square and hexagonal lattices we need to define a qubit register to configure the first coordinate of each electric dipole in the lattice and a register for the second coordinate. For the square lattice the mapping is even simpler since you only need to define $\sigma_i^1 \equiv \langle s|\hat{\sigma}_z^{i,1}|s\rangle$, as well as for the direction 2, where $|s\rangle \in \{|0\rangle, |1\rangle\}^{\otimes 2N}$ is an element of the computational base, i is the index, or a couple of indices, of a single water molecule, and

$$\hat{\sigma}_z^{i,1} = I_1^{(0)} \otimes \cdots \otimes I_1^{(i-1)} \otimes \hat{\sigma}_{z,1}^{(i)} \otimes I_1^{(i+1)} \otimes \cdots \otimes I_1^{(N-1)} \otimes I_2^{(0)} \otimes \cdots \otimes I_2^{(N-1)} \tag{5.56}$$

is a $2N$ qubit operator which apply on the first qubit, of the i-th molecule, the Pauli matrix

$$\hat{\sigma}_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{5.57}$$

For the hexagonal lattice, the first register of N qubit encodes the σ_1 direction, the second the σ_2 direction. The following definitions therefore apply

$$\sigma_{\alpha,i} = \left\langle s|\hat{\sigma}_z^{\alpha,i}|s\right\rangle \tag{5.58}$$

ehere it is useful to specify the operator $\hat{\sigma}_z^{3,i}$ for a given i, this operator restricted in the space of 2 qubits defining a given molecule follows the definition in Equation 5.51

$$\hat{\sigma}_z^3 = -\left(\sigma_z \otimes I + I \otimes \sigma_z + I \otimes I\right) \tag{5.59}$$

In the case of the square lattice, the Hamiltonian does not contain terms of type $\hat{\sigma}_z^{i,1}\hat{\sigma}_z^{i,2}$ and therefore the states of the $2N$ qubit registers live in two separate Hilbert spaces of N qubit. The square lattice is therefore more easily addressed by a quantum computer since it is possible to separate the calculation between the 1 and 2 coordinates by reducing the number of qubits required.
In the case of the hexagonal lattice, there are terms with σ_3, therefore, since

$$\hat{\sigma}_z^3\hat{\sigma}_z^3 = \left(\hat{\sigma}_z^1 + \hat{\sigma}_z^2 + 1\right)\left(\hat{\sigma}_z^1 + \hat{\sigma}_z^2 + 1\right) = 2\left(\hat{\sigma}_z \otimes \hat{\sigma}_z + \hat{\sigma}_z \otimes I + I \otimes \hat{\sigma}_z + \frac{3}{2} I \otimes I\right) \tag{5.60}$$

it is necessary to explore the states where the two registers are entangled.
One peculiarity of the hexagonal lattice is that the water molecules have three

different configurations, so it was necessary to use two qubits per molecule. Two qubits correspond to 4 states so one state does not have a physical correspondent in the water lattice. So we had to add terms to the Hamiltonian to penalize one of the 4 states to two qubits. Since, as we have defined the hexagonal lattice, the water molecule can not have an electric dipole $(\sigma_n^1, \sigma_n^2) = (1, 1)$ then the state of the two corresponding qubits should not be in $|00\rangle$. Such a constraint corresponds to the inequalities

$$\sigma_{1,n} + \sigma_{2,n} - 1 \leq 0 \ \ \forall n \tag{5.61}$$

In the Hamiltonian, a penalty term has to be added in order to penalize non-physical states. A natural choice is

$$\beta \hat{H}_P = A \sum_{i=1}^{N} (\hat{\sigma}_z^{i,1} + \hat{\sigma}_z^{i,2} - 1) \tag{5.62}$$

However, this choice promotes the different physical states in a non-congruent way, therefore a better choice is the following

$$\beta \hat{H}_P = A \sum_{i=1}^{N} (\hat{\sigma}_z^{i,1} + 1)(\hat{\sigma}_z^{i,2} + 1) \tag{5.63}$$

because such a term is null for every physical state. For the reasons explained above, the hexagonal lattice turns out to be a more suitable benchmark for studying quantum variational algorithms given its complications. In the hexagonal case the Hamiltonian operator turns out to be overall

$$\begin{aligned}
\beta \hat{H} &= \beta \left(\hat{H}_{HB} + \hat{H}_I + \hat{H}_P \right) = \\
&= \sum_{\alpha=1}^{3} \sum_{i=0}^{N} \sum_{j=0}^{N} \delta_{\alpha, \mathcal{A}_{n,m}} \left(\frac{R - \epsilon_H}{2} \hat{\sigma}_z^{\alpha,i} \hat{\sigma}_z^{\alpha,j} + \frac{R + \epsilon_H}{2} \right) + \\
&+ \frac{\Gamma}{2} \sum_{i=0}^{N} \frac{1}{r_i^3} \Big(\left(2\hat{\sigma}_z^{1,i} + \hat{\sigma}_z^{2,i} + 1 \right) \vec{r}_i \cdot \vec{\sigma}_1 + \left(2\hat{\sigma}_z^{2,i} + \hat{\sigma}_z^{1,i} + 1 \right) \vec{r}_i \cdot \vec{\sigma}_2 \Big) + \\
&+ A \sum_{i=1}^{N} (\hat{\sigma}_z^{i,1} + 1)(\hat{\sigma}_z^{i,2} + 1)
\end{aligned} \tag{5.64}$$

Which turns out to be equivalent to a Hamiltonian of an Ising model with zero transverse field on $2N$ spin

$$\beta\hat{H} = \sum_{i\geq j}^{2N} \mathcal{J}_{ij}\hat{\sigma}_z^i\hat{\sigma}_z^j + \sum_{i}^{2N} b_i\hat{\sigma}_z^i \tag{5.65}$$

The system as defined corresponds to an Ising model with very sparse connectivity, this means that the matrix $\mathcal{J}$ has null values for the most part. The Hamiltonian terms in $\hat{H}_{HB}$ contribute to the couplings. In particular, defining C_α the number of couplings of type α in the lattice, then, In $\hat{H}$m C_1 Ising couplings come from lattice coupling of the type 1, C_2 for the type 2 and $4C_3$ for 3. The C_3 type 3 couplings contribute also with 4 bias terms. The Ion Interactions, instead, contribute only with $2N$ bias terms while the penalty terms contribute with N couplings and $2N$ bias terms.
Since each molecule has 3 couplings, one for each type, then approximately $C_\alpha \approx N/2\ \forall\alpha$. therefore, the total number of Pauli terms in $\hat{H}$ is the following

$$\begin{aligned}
&\#\text{1-qubit Pauli Terms} = 4C_3 + 2N + 2N \\
&\#\text{2-qubit Pauli Terms} = C_1 + C_2 + 4C_3 + N \\
&\Rightarrow \#\text{Pauli Terms} = C_1 + C_2 + 8C_3 + 5N \approx 10N
\end{aligned} \tag{5.66}$$

So the number of terms grows linearly with the number of molecules in the lattice. It is therefore a very sparse problem since the total number of terms for a fully connected Ising model at $2N$ spins is $N(2N+1)$.

Introduction of long-range interactions

A problem as sparse as the one defined above can be easily solved by classical solvers such as GUROBI and also by heuristic solvers such as simulated annealing. For this reason, it is necessary to introduce long-range interactions between the molecules in the lattice. Let's define the dipole-dipole interaction between all the molecules in the lattice. The dipole-dipole interaction between two molecules with electric dipole $\vec{\mu}_i$ ans $\vec{\mu}_j$ is defined as follows

$$E_{DD}(i,j) = k\frac{2\vec{\mu}_i\cdot\vec{\mu}_j}{r_{ij}^3} \tag{5.67}$$

The Hamiltonian corresponding to this interaction is obtained simply from the definition in Equation 5.49.

$$\begin{aligned}\beta H_{DD} &= \Omega \sum_{i,j} [\mathcal{A}_{ij} = 0] \left(\sum_{\alpha_1=1}^{2} \sum_{\alpha_2=1}^{2} \Sigma_{\alpha_1,i} \Sigma_{\alpha_2,j} \vec{\sigma}_{\alpha_1,i} \cdot \vec{\sigma}_{\alpha_2,j} \right) \\ \Sigma_{1,i} &\equiv 2\sigma_{1,i} + \sigma_{2,i} + 1 \\ \Sigma_{2,i} &\equiv 2\sigma_{2,i} + \sigma_{1,i} + 1\end{aligned} \tag{5.68}$$

where $[\cdot]$ is called Iverson bracket which returns 1 if the condition in the bracket is true and 0 otherwise, indeed such a term involves only molecules that are not first neighbors and for which a hydrogen bond is not established. With the addition of these interactions, the corresponding Ising model thus becomes fully connected, and therefore the number of Pauli terms of the operator $\hat{H}$ scales as $\mathcal{O}(N^2)$.

5.4 Results

In this section, the scaling assessments are presented regarding the different classical and quantum solvers in solving the problem of finding the ground state of the Hamiltonian as defined in Equation 5.64. As anticipated, the Hamiltonian that maps the system of N water molecules into a 2-dimensional lattice is equivalent to an Ising model, with zero transverse field, on $2N$ binary variables and therefore $2N$ qubits. In general, it is a spin-glass model

$$H = \sum_{i \geq j}^{2N} \mathcal{J}_{ij} z_i z_j + \sum_{i}^{2N} b_i z_i \tag{5.69}$$

where the dimensional term β is neglected. The variables $z \in \{-1, 1\}$ are often called Ising variables. Finding the ground state is therefore an optimization problem where we want to find the string of binary variables $z = (z_0, z_1, \ldots, z_{2N-1})$ that achieves the minimum energy

$$\overline{z} = \arg\min_{z} \sum_{i \geq j}^{2N} \mathcal{J}_{ij} z_i z_j + \sum_{i}^{2N} b_i z_i \tag{5.70}$$

An Ising variable problem like the one just defined can be reconverted into a binary variables problem $x_i \in \{0, 1\}$ via the transformation $z_i = 1 - 2x_i$. The resulting problem is a problem called quadratic unconstrained binary optimization problem or QUBO problem which can be written between the definition of

the quadratic form Q

$$\overline{x} = \arg\min_{x} \; x^T Q x \tag{5.71}$$

the matrix Q is a $2N \times 2N$ symmetric matrix and, in the case of a hexagonal water lattice with long-range dipole-dipole interaction, has not zero elements in general, and the problem is said fully connected. It should be emphasized that, although the solution of the two problems, Ising and QUBO, correspond via the formula $z_i = 1 - 2x_i$, the ground state energy differs by a constant offset term.

5.4.1 Benchmark strategy

Having defined the problem, it is necessary to define the window of instances of the problem to be solved on which to evaluate the performance of the various optimization solvers. The results that will be presented below will concern

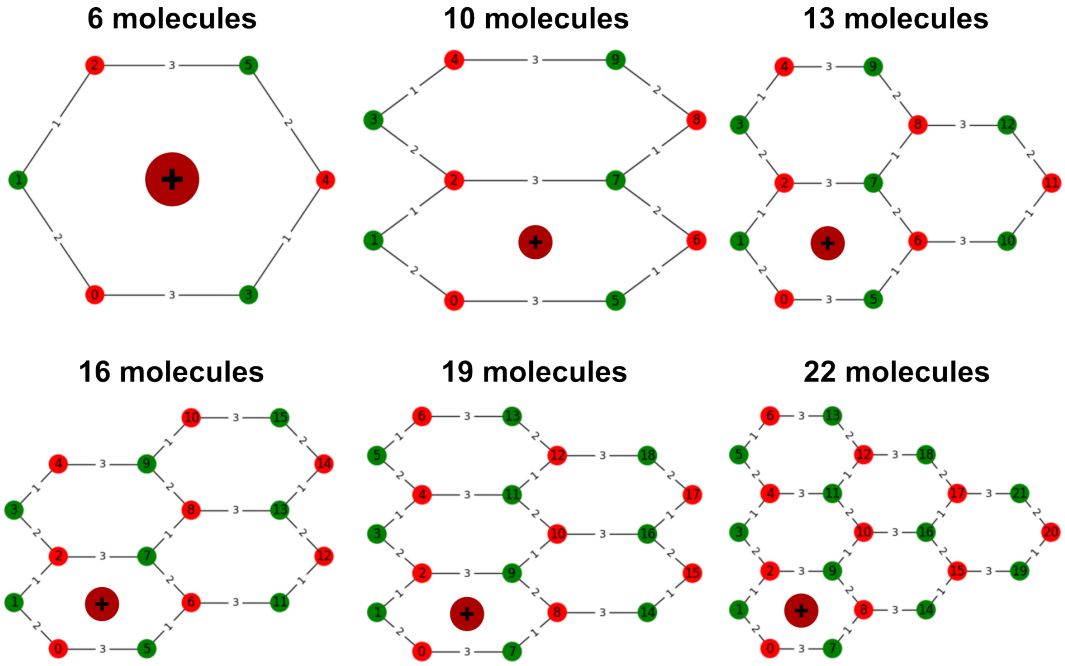


FIGURE 5.5: Illustration of the lattices chosen to test the optimization solvers. The figure shows the $N = 6$ cases where the lattices are composed of hexagons. For the other values of N between $N = 6$ and $N22$, the lattices have "unpaired" molecules, i.e. which do not belong to hexagons. By increasing N, molecules are added following the hexagonal topology starting from the lattice of 6 molecules which is a single hexagon. For example, the case with $N = 7$ corresponds to a lattice of 7 molecules with a hexagon and a molecule linked to one of the vertices of the hexagon.

17 different hexagonal lattices from $N = 6$ to $N = 22$ molecules. The case $N = 6$ corresponds to a single hexagon, the molecules will be added gradually

Constant	Value (kT)
ϵ_H	-2
R	0.5
Γ	1.25
Ω	1
A	10

TABLE 5.1: Values of the constants with which the systems on which the optimization solvers were tested are defined.

to form other hexagons as shown in Figure 5.5. In all the systems considered, the parameters ϵ_H, R, Γ, Ω and the penalty constant A are setted as shown in the Table 5.1. To calculate the solution of the QUBO problems associated with the different lattices, GUROBI was used as the reference solver [152]. With Gurobi it is possible to demonstrate the optimality of the solution to the optimization problem. GUROBI is a powerful optimization software suite widely acclaimed for its remarkable capabilities in tackling complex combinatorial problems. Within the suite, one prominent solver for addressing Quadratic Unconstrained Binary Optimization (QUBO) problems is the GUORBI solver. This specialized solver is a testament to the advanced techniques and algorithms harnessed by GUROBI, particularly the branch and bound method, for addressing QUBO problems with utmost efficiency and precision. The GUORBI solver employs a branch and bound approach, a fundamental algorithmic technique in mathematical optimization, to systematically explore the solution space of QUBO problems. By decomposing the problem into smaller subproblems and progressively narrowing down the search space, it efficiently identifies optimal solutions or guarantees optimality. This intelligent solver leverages a combination of linear programming relaxations, heuristics, and cutting-plane methods, adapting to the specific problem structure and complexities. In Figure 5.6 the results using GUROBI are shown. Such a test is useful to verify that the problem is hard for classical solvers. The QUBO problem solved by GUROBI is a version without the penalty terms. However, constraints have been added

$$x_{1i} + x_{2i} \geq 1 \tag{5.72}$$

where $x_{1i}, x_{2i} \in \{0, 1\}$ are the two binary variables defining the i-th molecule. Therefore the problem passed to the GUROBI solver precisely from the category of Binary Quadratic Programs (BQPs). The results show that GUORBI takes, clearly, an exponential time to find the solution in the range of $N = 6, \ldots, 22$ molecules. With GUROBI, when the dimensions of the problem are still small,

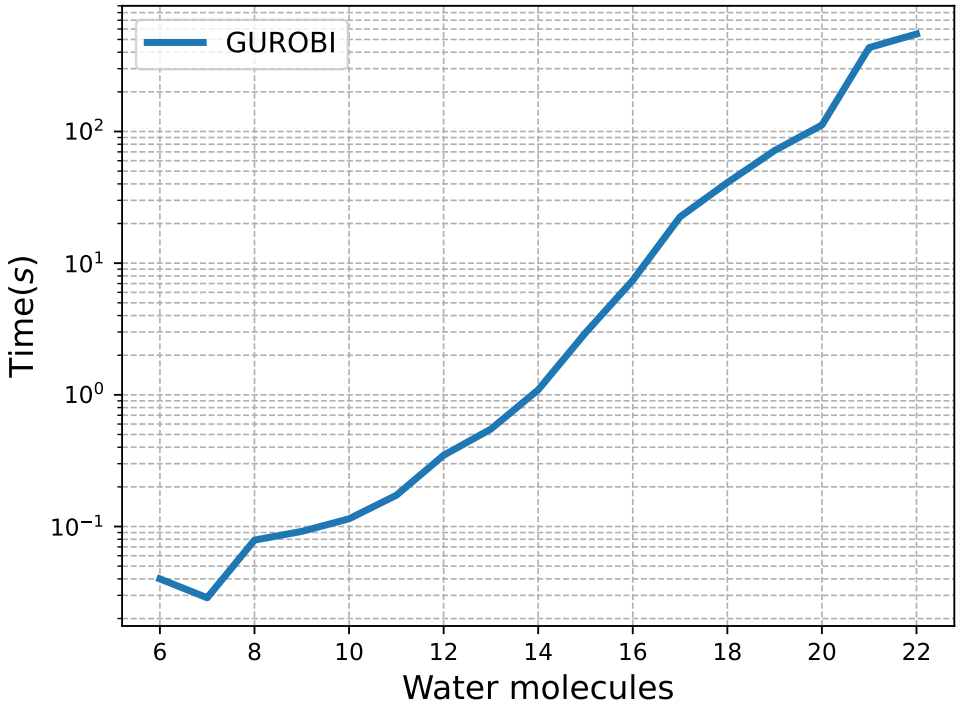


FIGURE 5.6: Average GUROBI runtime to solve the QUBO problems related to lattices of 6 to 22 molecules.

it is always possible to obtain the solution in relatively short times as in the case in Figure 5.6. For other solvers, as in the case of VQE and QAOA, which are heuristic, it is necessary to take into account the trials, or seeds, which have not led to the solution within a certain time limit. Particular metrics must therefore be defined to evaluate the performance of heuristic software. In the context of this and other works present in this document, an innovative metric called Mean First Solution Time (MFST) was defined.

Mean First Solution Time (MFST)

It is customary, when running algorithms, to define a maximal execution time, after which the computation is stopped and a new parameter (e.g. seed) is used to run the computation. This requires proper metrics to evaluate the overall expected execution time. In particular, the estimation should include the time spent in failures and not just the average time of successes. The estimated execution time then correlates with the allocated computing time and hence with the monetary budget for the computation. We therefore introduce a new metric, which takes fully into account the time spent in failed runs. We call this metric the Mean First Solution Time (MFST) (as it is inspired by the Mean

First Passage Time concept in physics [153]). Given a problem instance, the MFST is defined as:

$$T_{MFST} = \mathbb{E}\{k\}T_{max} + \mathbb{E}\{t\} \tag{5.73}$$

where $\mathbb{E}\{k\}$ is the expected number of failures before the first solution is found, T_{max} is the maximal allowed execution time (e.g. for one seed) it is not a random variable, and $\mathbb{E}\{t\}$ is the expected solution time (the averaged time of solved instances). The expected solution time $\mathbb{E}\{t\}$ can be easily estimated via the usual sample mean estimator:

$$\bar{t}_s = \frac{1}{|I_s|}\sum_{j\in I_s} t_j \tag{5.74}$$

where I_s is the set of solved instances and $|I_s|$ its cardinality. The variable k is a random variable which follows the negative hypergeometric distribution $\mathrm{NHG}(|I|, |I| - |I_s|, 1)$

$$k \sim \mathrm{NHG}(|I|, |I| - |I_s|, 1) = \frac{\binom{|I|-k-1}{|I|-|I_s|-k}}{\binom{|I|}{|I|-|I_s|}} \tag{5.75}$$

where $|I|$ is the cardinality of the run set. Therefore the expected value of the number of failures k before a success is:

$$\frac{|I| - |I_s|}{|I_s| + 1} \tag{5.76}$$

Hence, we estimate the MFST via the formula:

$$\overline{T}_{MFST} = \frac{|I| - |I_s|}{|I_s| + 1}T_{max} + \frac{1}{|I_s|}\sum_{j\in I_s} t_j \tag{5.77}$$

Clearly, if all instances are solved, the mean solution time is obtained. Another widely used metric for evaluating the scalability is the time to solution scaled by the solution probability (TTS). While this quantity is formally a time, it does not explicitly consider the maximal execution time, which is captured by the MFST. In particular the Time To Solution (TTS) is defined as:

$$T_{TTS} = \frac{\mathbb{E}\{t\}}{p} \tag{5.78}$$

where p is the solution probability; this can be estimated as $\bar{p} = |I_s|/|I|$, hence the estimator:

$$\overline{T}_{TTS} = \frac{\bar{t}_s}{\bar{p}} \tag{5.79}$$

It is easy to show that there is a simple relation between $\overline{T}_{TTS}$ and $\overline{T}_{MFST}$:

$$\overline{T}_{MFST} = \frac{|I| - |I_s|}{|I_s| + 1} T_{max} + \frac{|I_s|}{|I|} \overline{T}_{TTS} \tag{5.80}$$

This relation shows that, if all problems are solved ($|I_s|$=$|I|$), the two metrics are the same. However when not all instances are solved, the presence of T_{max} creates a discrepancy. If T_{max} is high, the TTS can significantly underestimate the real execution time. The variance of $\overline{T}_{MFST}$ is computed following the error propagation formula:

$$\begin{aligned} \sigma^2_{\overline{T}_{MFST}} &= \left(\frac{\partial \overline{T}_{MFST}}{\partial |I_s|} \right)^2 \sigma^2_{|I_s|} + \left(\frac{\partial \overline{T}_{MFST}}{\partial \bar{t}_s} \right)^2 \sigma^2_{\bar{t}_s} \\ &= T^2_{max} \frac{(|I| + 1)^2 |I_s|^2}{(|I_s| + 1)^4} \frac{1}{|I|} \frac{1 - \bar{p}}{\bar{p}} + \sigma^2_{\bar{t}_s} \end{aligned} \tag{5.81}$$

where:

$$\sigma^2_{\bar{t}_s} = \frac{\sigma^2_{t_s}}{|I_s|} \tag{5.82}$$

and $\sigma^2_{t_s}$ is the variance of the solution time.
For the time to solution, we can estimate the variance by observing that $|I_s|$ is distributed as a binomial distribution. That leads to:

$$\sigma^2_{\bar{p}} = \frac{\bar{p}(1 - \bar{p})}{|I|} \tag{5.83}$$

Using the error propagation formula, it follows that:

$$\sigma^2_{\overline{T}_{TTS}} = \overline{T}^2_{TTS} \left(\frac{1}{|I|} \frac{1 - \bar{p}}{\bar{p}} + \frac{\sigma^2_{\bar{t}_s}}{\bar{t}^2_s} \right) \tag{5.84}$$

All the presented formulas are meaningful if at least one instance has been solved, that is $|I_s| > 0$ or equivalently $p \in (0, 1]$.

Classical solvers

As defined in Chapter 1, Simulated annealing is one of the most exploited optimization techniques based on Markov chain Monte Carlo methods. To find the minimum energy structure of the lattice of water molecules, simply solve the relative QUBO problem as defined in Equation 5.71. To solve a QUBO problem with simulated annealing, we define as the initial condition a random bitstring $\boldsymbol{x}^{(0)} = \boldsymbol{x}_{in}$. We then generate a trial state $\boldsymbol{x}^{(1)}$ by flipping a random bit into the $\boldsymbol{x}^{(0)}$ bitstring. Generally a trial state $\boldsymbol{x}^{(k)}$ will be accepted with

probability

$$A(\boldsymbol{x}^{(k)} \rightarrow \boldsymbol{x}^{(k+1)}) = \min\left[1, \exp\left\{-\frac{E_{\boldsymbol{x}^{(k+1)}} - E_{\boldsymbol{x}^{(k)}}}{T_k}\right\}\right] \tag{5.85}$$

where $E_{\boldsymbol{x}} \equiv \boldsymbol{x}^T Q \boldsymbol{x}$ is the energy of the system up to an offset term. The sequence of temperatures along the T_k iterations remains to be defined. There are several choices, the most common being linear scheduling and geometric scheduling. In linear scheduling, the temperature decreases linearly with the iterations

$$T_k \rightarrow T_{k+1} = T_k - l \Rightarrow T_k = T_0 - kl \tag{5.86}$$

where $l > 0$. In the case of geometric scheduling there is instead a decreasing temperature following a geometric sequence:

$$T_k \rightarrow T_{k+1} = rT_k \Rightarrow T_k = r^k T_0 \tag{5.87}$$

where $0 < r < 1$. Figure 5.7 shows the performance of simulated annealing in the search for the ground state of the lattices in a window ranging from 6 molecules to 17 for a total of 12 points. Each point in the plot in the figure represents the computational cost expressed in Mean First Solution Samples or MFSS with a number of trials $N = 30$ (i.e. 30 different seeds). This MFSS metric is exactly defined with the MFST but instead of evaluating the execution times, in this case, the reference quantity is the number of iterations and therefore the number of samples generated, in other words, the length of the Markov chain. Each trial that leads to the solution of the QUBO problem, contributes a cost equal to the number of samples generated before obtaining the first bitstring relating to the ground state. Instead of a maximum time T_{max}, a maximum number of iterations K_{max} is therefore defined. The procedure for generating new samples scales in time like the energy calculation and therefore with the quadratic form $\boldsymbol{x}^T Q \boldsymbol{x}$. The results in Figure 5.7 were obtained with $K_{max} = 10^5$ and with geometric scheduling with $r = 0.999$ and $T_0 = 100$ (the temperature is defined as dimensionless as for the Hamiltonian). Also in Figure 5.7 there is also the performance relating to a parallel tempering simulated annealing which exploits the replica exchange technique to reduce the probability of becoming trapped in local minima. The results for parallel tempering were obtained by initializing 10 chains $\boldsymbol{x}^{(0),c} = \boldsymbol{x}^c_{in}$ where c is the chain index $c \in \{1, 2, \ldots, 10\}$. For each chain, a simulated annealing procedure is defined. Therefore, as before, the Markov chain is built with the accept/reject rule defined in Equation 5.85 with scheduling, also in this case, geometric, with

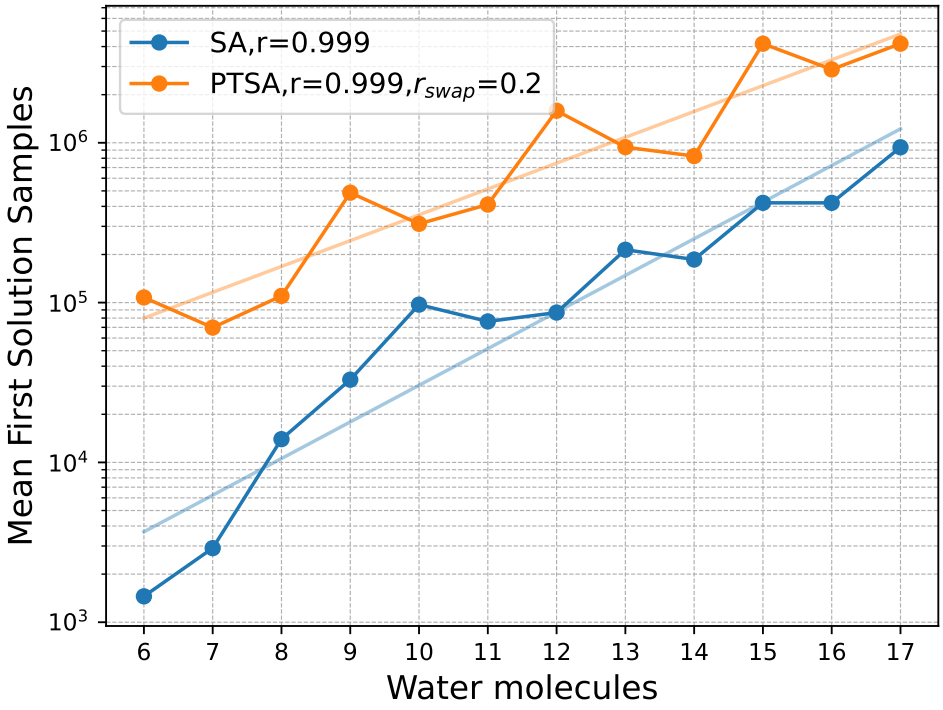


FIGURE 5.7: Performances, in terms of MFSS, of classical optimization solvers: Simulated Annealing (SA), blue plot, and Parallel Tempering Simulated Annealing (PTSA), orange plot. The temperature scheduling of SA and the chains ion PTSA follow a geometric progression with the cooling rate $r = 0.999$. In the PTSA the chains can be swapped with a rate of accept/reject steps on $r_{swap} = 0.2$ which means every 5 iterations.Each solver was run 30 times (with 30 different seeds) for each lattice.

$r = 0.999$. The initial temperature is different for each chain: $T_0^{(c)} = 10c$. The additional step concerns the possibility of exchanging the adjacent chains c and $c+1$. according to the acceptance rule

$$A(\boldsymbol{x}^{(k),c} \leftrightarrow \boldsymbol{x}^{(k),c+1}) = \min\left[1, \exp\left\{-\left(\frac{1}{T_k^{(c)}} - \frac{1}{T_k^{(c+1)}}\right)\left(E_{\boldsymbol{x}^{(k),c+1}} - E_{\boldsymbol{x}^{(k),c}}\right)\right\}\right] \tag{5.88}$$

A good choice for the exchange rate between replicas is every 5 step. Since the cost of parallel tempering depends on the number of replicas and each replica has a maximum cost of 10^5 iterations, $K_{max} = 10^6$ for the data in Figure 5.7. Figure 5.7 shows the log scale fit to an exponential function $e^{\mu N}$ where N is the number of molecules in the lattice. The scaling of these solutions, as for the quantum methods that will be discussed later, is evaluated with respect to the parameter μ. Table 5.2 shows the performances in terms of exponential scaling

Method	iterations	chains	swape rate	μ	$\delta\mu$
SA	10^5	-	-	0.5277	0.0028
PTSA	10^5	10	0.2	0.3721	0.0017

TABLE 5.2: Scaling performances in MFSS of the tested classical solutions. Parallel tempering simulated annealing (PTSA) clearly shows better performance as expected. The swap rate at 0.2 corresponds to 2 accept/rejection swap steps every 10 iterations and therefore one every 5. Being parallelized on 10 chains, the total samples collected with one PTSA run is 10^6, other implementation details in the caption of Figure 5.7

of the tested classical solutions, simulated annealing and parallel tempering, which refer to the slope of the fit $e^{\mu N}$ in log scale.

5.4.2 Quantum solutions tuning

Let's discuss the QAOA settings to obtain the best performances against the classical heuristic solvers. The first step was to evaluate performance in the standard QAOA setting. The QAOA has been defined as a specific version of the VQE with a physics-based ansatz circuit that exploits the background principle of the adiabatic theorem. In the standard setting of the algorithm, the qubit register is initialized in the state $|+\rangle^{2N}$, i.e. the ground state of the mixer Hamiltonian

$$\hat{H}_M = \sum_{i=0}^{2N} X_i \tag{5.89}$$

Figure 5.8 shows the performance in terms of scaling of the average number of samples (qubit measurements) to reach the solution for the first time. All runs were performed on the 32-qubit noise-free quantum simulator *ibmq_qasm_simulator*. The range of solved systems is from $N = 6$ to $N = 13$ molecules. The three curves represent three different values of p, i.e. the number of repetitions of the ansatz as defined in Equation 5.35. The three curves are the result of the expectation value of the mean first solution samples (MFSS) on $M = 15$ trials for each N instance and with a maximum number of samples of $K_{max} = 10^5$ based on a number of shots $S = 10^3$, to evaluate the cost function, and 100 parameters updates, i.e. iterations of the classic optimizer. The classic optimizer chosen is called COBYLA. Constrained Optimization BY Linear Approximation (or COBYLA) [154] is an implementation of Powell's nonlinear derivative-free constrained optimization that uses a linear approximation approach. The algorithm is a sequential trust-region algorithm that employs

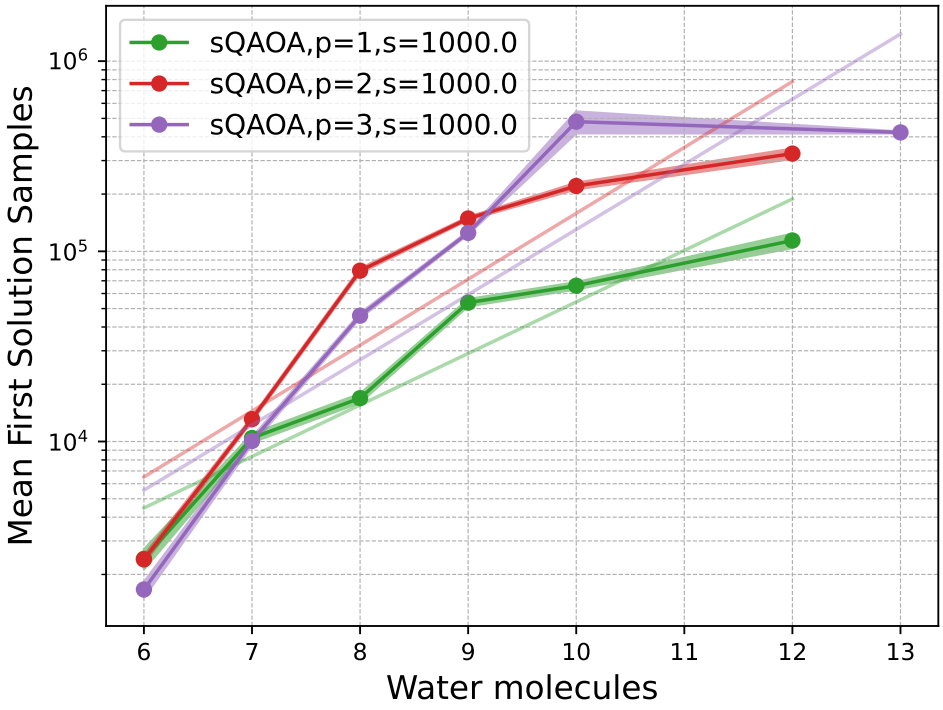


FIGURE 5.8: Results, in terms of MFSS, of a QAOA with standard mixer ($\otimes^{2N} R_X(\beta)$). Each point is the result of the MFSS estimator over 15 runs on the 32 qubit cloud simulator *ibmq_qasm_simulator*. The 15 runs are equally distributed in three CVaR setting $\alpha = 1, 0.5, 0.1$. At each iteration of the optimizer 1000 samples (measurements) are collected, therefore the cost of each run is evaluated with the number of iterations accumulated up to the first where the ground state is measured multiplied by the number of total measurements per iteration (1000). For this reason, the minimum cost is 1000 if the ground state is measured on the first iteration. The parameters β and γ are initialized considering as prototypical scheduling $(1 - \lambda(t))H_M + \lambda(t)H_c$ where $\lambda(t) = t \in [0, 1]$.

linear approximations to the objective and constraint functions, where the approximations are formed by linear interpolation at $n + 1$ points in the space of the variables and tries to maintain a regular-shaped simplex over iterations. This choice was made with the aim of not having to require the calculation of the gradient of the parametric circuits that it would require.

The potential of the CVaR cost function has also been explored. As defined in Equation 5.40, α is the parameter that represents the portion of the samples considered to build the cost function to pass to the classical optimizer. Different α leads to different performance since it is very tied to the initial condition. For this reason, the 15 trials which then translate the curves in Figure 5.35 are

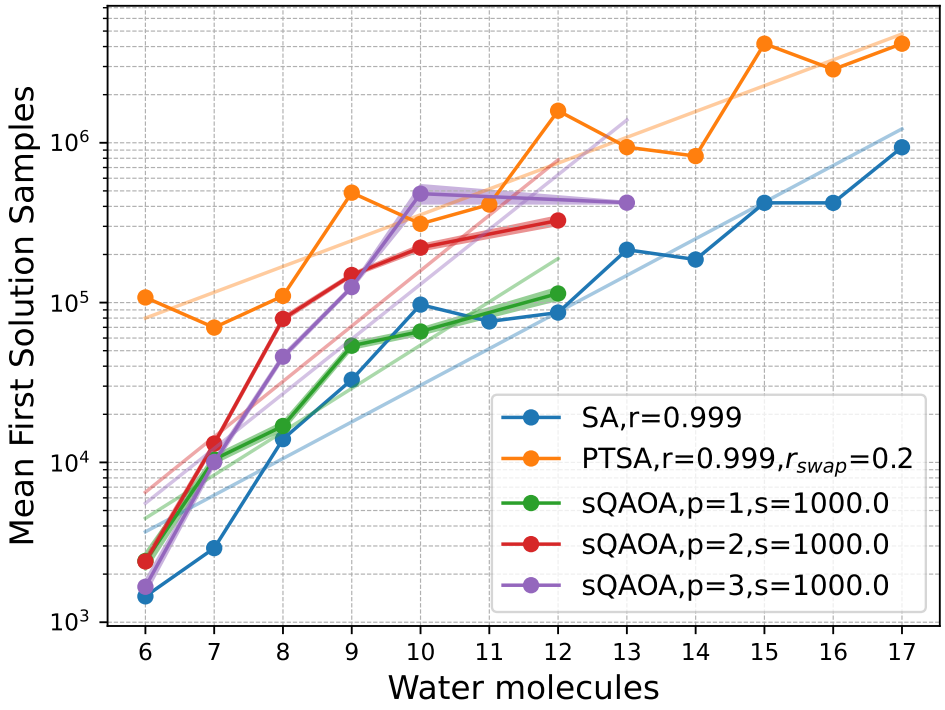


FIGURE 5.9: Comparative results between classical methods and QAOA with standard mixer. The case $p = 1$ has a behavior comparable to simulated annealing.

divided into groups of 5 trials for the cases with $\alpha = 1$, $\alpha = 0.5$ and $\alpha = 0.1$.

Figure 5.9 shows the performance of a standard version of QAOA against the optimized versions of simulated annealing and parallel tempering. In terms of MFSS, the case with $p = 1$ performs better than the case with a higher circuit depth. However, for $p = 1, 2, 3$ we obtain performances, in terms of scaling, worse than parallel tempering and a performance similar to simulated annealing only in the case $p = 1$. However, it should be pointed out that for $N > 10$ and therefore for the cases $N = 11, 12, 13$ there are cases where in none of the 15 trials the solution was reached in the 100 iterations. It therefore

Method	p	shots	μ	$\delta\mu$
QAOA	**1**	**1e3**	**0.6237**	**0.0118**
QAOA	2	1e3	0.7980	0.0346
QAOA	3	1e3	0.7890	0.0443

TABLE 5.3: Scaling performances in MFSS of QAOA with standard mixer. The best result is achieved by the case with $p = 1$ (in bold). However, the trend is evaluated in a region that includes the lattices of 6 and 7 molecules which turn out to be particularly simple to solve. Other details in the caption of Figure 5.8.

appears that, in a standard version, QAOA performs similarly if not worse than the classical heuristic methods taken into consideration. This result leads to reflection on how the QAOA setting can be improved in order to obtain better performances. The presence of non-feasible solutions that do not have a physical meaning can be of help to methods based on Markov chains such as simulated annealing given that passing from non-feasible states can help to escape from local minima. For variational quantum methods, they have no usefulness and can actually lead to plateaus. In fact, the number of unfeasible states grows exponentially with the number of molecules. Considering that for each molecule there is one, out of 4, unfeasible state, therefore, for N molecules, the number of unfeasible states is

$$\#\text{unfeasible solutions} = 4^N - 3^N \tag{5.90}$$

that, for large N, is $\mathcal{O}(4^N)$. Therefore, considering a uniform probability of sampling, the probability of sampling an unfeasible solution grows as

$$P = \frac{4^N - 3^N}{4^N} = 1 - \left(\frac{3}{4}\right)^N \xrightarrow{N\to\infty} 1 \tag{5.91}$$

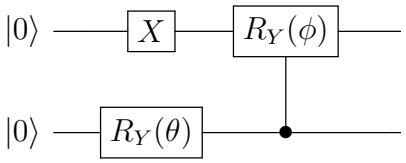
The probability of measuring an unfeasible state therefore becomes important for large N. A specific ansatz circuit design can therefore prevent this problem. For this purpose it is assumed that the variational circuit acts as follows

$$\left\langle \psi_{uf} \middle| U(\beta,\gamma) \middle| \tilde{\psi} \right\rangle = 0 \tag{5.92}$$

where $|\psi_{uf}\rangle$ is a generic unfeasible state and $\left|\tilde{\psi}\right\rangle$ is the initial state which is a superposition over all the feasible states. A simple choice for $\left|\tilde{\psi}\right\rangle$ is as follows

$$\left|\tilde{\psi}\right\rangle = \bigotimes_{i=0}^{N-1} \frac{1}{\sqrt{3}} \left(|01\rangle_i + |10\rangle_i + |11\rangle_i\right) \tag{5.93}$$

which can be simply obtained from the following circuit



By repeating this circuit for each pair of qubits relating to a molecule and setting

the angles as $\theta = \arccos 1/\sqrt{3}$ and $\phi = \arccos 1/\sqrt{2}$, the state in Equation 5.93 is obtained. To exploit the adiabatic theorem, a Hamiltonian mixer must therefore be constructed such that $\left|\tilde{\psi}\right\rangle$ is the ground state. For a generic choice of $\left|\tilde{\psi}\right\rangle$, a consistent choice for $U_M(\beta)$ is an operator of the type $U_M = 1 \oplus V$ where $V \in SU(3)$. A generic element of $SU(3)$ has 8 free parameters while if it is considered as unitary $U_M = 1 \oplus O$ where O is an orthogonal matrix in $SO(3)$ then only 3 parameters are involved. A generic element in $SO(3)$ is obtained via the following exponential map

$$O = \exp\{-i\,(xL_x + yL_y + zL_z)\} \tag{5.94}$$

Where L_x, L_y and L_z are the basis of then algebra of $SO(3)$ that generate rotations in 3-dimensional space along axis x, y and z respectively. In the reduced 2-qubit Hilbert space of only the feasible solutions, O is a rotation where the axis x, y and z corresponds to $|01\rangle$, $|10\rangle$ and $|11\rangle$. Therefore, to build a circuit on 2 qubits that realizes $U_M = 1 \oplus O$ we consider the decomposition into Pauli matrices of the generators of $U_M = 1 \oplus O$. The step to obtain the correct decomposition is to expand the L_i generators so that they are defined on a 2 qubit Hilbert space. Let $\tilde{L}_i$ be the representation in 4×4 matrices of the generators of $SO(3)$, the following decompositions are obtained.

$$\tilde{L}_x = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \frac{1}{2}\,(I \otimes Y - Z \otimes Y)$$

$$\tilde{L}_y = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix} = \frac{1}{2}\,(Y \otimes I - Y \otimes Z) \tag{5.95}$$

$$\tilde{L}_z = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \frac{1}{2}\,(X \otimes Y - Y \otimes X)$$

Therefore the mixer operator U_M can be written as

$$U_M = \bigotimes_{i=0}^{N-1} e^{-i\beta H_M^s} \tag{5.96}$$

where H_M is a Hamiltonian defined on the two qubits that encodes the state of a single molecule

$$H_M^s = \frac{x}{2}(Y_2 - Z_1Y_2) + \frac{y}{2}(Y_1 - Y_1Z_2) + \frac{z}{2}(X_1Y_2 - Y_1X_2) \tag{5.97}$$

Setting $x = y = z = -1/\sqrt{3}$ then the initial state $\left|\tilde{\psi}\right\rangle$ in Equation 5.93 is an eigenstate of U_M with eigenvalue -1. Given H_C is an Ising Hamiltonian that is diagonal in the computational basis then U_C acts by adding only relative phases. The Hilbert subspace of feasible solutions and their superpositions remains unchanged under the U_M transformation. The following section analyzes the performance of this version of QAOA, which will be indicated with sQAOA (since it acts on the unit sphere with axes $|01\rangle$, $|10\rangle$ and $|11\rangle$). In this version, the parametrized state $|\psi(\beta,\gamma)\rangle = U(\beta,\gamma)\left|\tilde{\psi}\right\rangle$ is a superposition of only the feasible solutions. The sQAOA is a quantum approximate optimization algorithm with constraint preserving mixers [155] where the mixer operator restricts the evolution to a sub-space of the full Hilbert space generated by feasible computational basis states.

5.4.3 Performances discussion

This section presents the results of the customized version of the QAOA which takes into account only the feasible solutions. Figure 5.10 shows the MFSS results of sQAOA with 1000 shots for each iteration of the COBYLA optimizer. In the figure, there are results with $p = 1$ and $p = 2$ in a window of 6 different lattices from 8 to 13 molecules. This new version reduces in absolute value the average number of samples necessary to obtain the solution, exceeding the performance for simulated annealing for cases such as the 10 and 11 molecule lattice but the scaling is comparable to simulated annealing and also to the standard version of the QAOA. However, the parallel tempering method is more performing in terms of scaling. also in this case the points relating to the sQAOA were obtained considering a CVaR version of the cost function with $\alpha = 1$, $\alpha = 0.5$ and $\alpha = 0.1$, respectively with 10 different tests for each α for a total of 30 trials. The case with $p = 1$ initially shows better scaling but beyond the 11 molecules it is unable to find the ground state of the lattice and

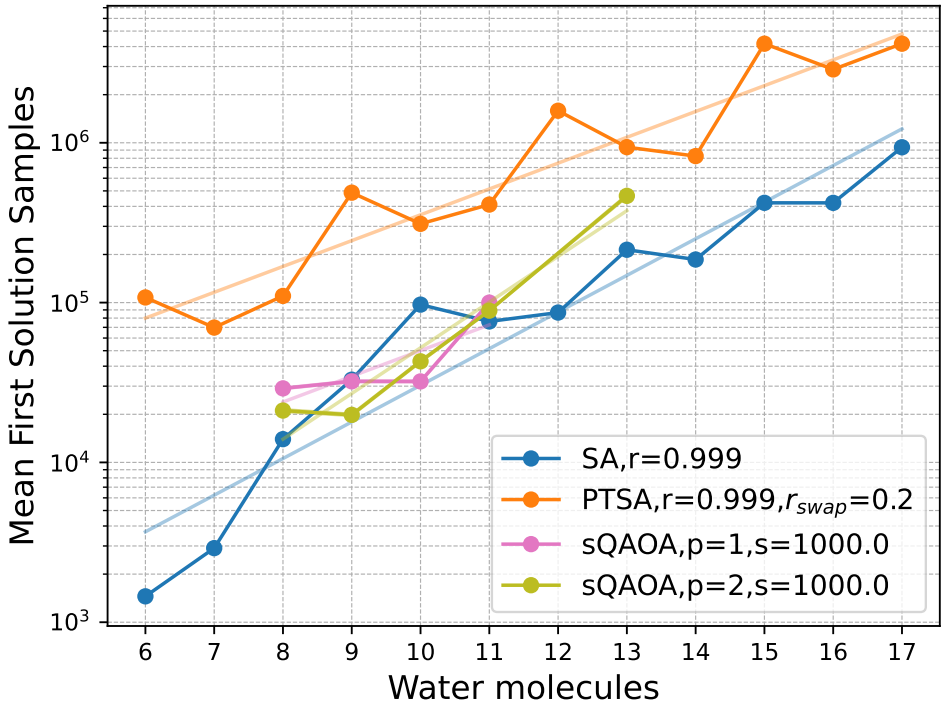


FIGURE 5.10: Results, in terms of MFSS, of a sQAOA that preserves the space of feasible solutions with 10^3 shots per iteration. Each point is the result of the MFSS estimator over 30 runs on the 32 qubit cloud simulator *ibmq_qasm_simulator*. The 30 runs are equally distributed in three CVaR setting $\alpha = 1, 0.5, 0.1$. The parameters β and γ are linearly initialized as in the previous QAOA implementation

therefore the error on the scaling estimate (see Table 5.4) does not allow us to define any type of advantage. Increasing the number of shots up to 10^4 gives a maximum cost of $K_{max} = 10^6$ equal to the case of parallel tempering. In Figure 5.11 there are the results of the sQAOA with $S = 10^4$. In all cases in the window between 6 and 11 molecules, where this setting was tested, better performances than parallel tempering are obtained in terms of the absolute value of the number of samples generated. As regards scaling, the 5.4 table shows that the performances are comparable to those of parallel timing and slightly better than simulated annealing. This is valid for both the cases at $p = 1$ and $p = 2$ which are very similar. Also in this case the trials for each lattice are 30 distributed equally on the cases with $\alpha = 1$, $\alpha = 0.5$, and $\alpha = 0.1$ for the cost function $CVaR$. In the last case, Figure 5.12, the performance are shown with an ansatz which instead does not draw inspiration from the adiabatic theorem as in the QAOA but is more linked to a standard version

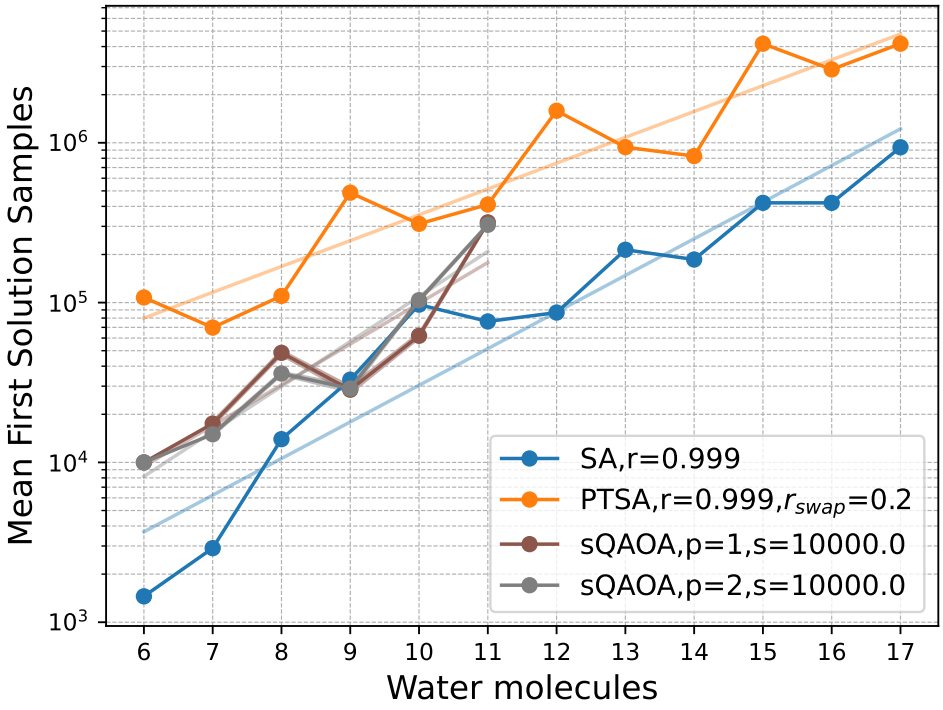


FIGURE 5.11: Results, in terms of MFSS, of a sQAOA with 10^4 shots for each iteration. The case $N = 6$ shows that each run reaches the ground state at the first iterations and, therefore, with the first set of 10^4 measurements. Each point is the result of the MFSS estimator over 30 runs on the 32 qubit cloud simulator *ibmq_qasm_simulator*. The 30 runs are equally distributed in three CVaR setting $\alpha = 1, 0.5, 0.1$. The parameters β and γ are linearly initialized as in the previous QAOA implementation

of the VQE. This version, which we call sVQE, is defined with an ansatz with $3N(p + 1)$ trainable parameters. With this version, we want to understand if by expanding the expressiveness of the variational ansatz by adding more parameters, better performances can be obtained. The ansatz of the sVQE is defined with a first layer of operators applied on the pairs of qubits that define a single molecule, an operator $U(\alpha, \beta, \gamma)$. Using the Euler angle formalism, for a single qubit we have that a generic rotation can be obtained by applying only rotations on the x and z axes of the Bloch sphere: $R_{\vec{n}}(\theta) = R_Z(\gamma)R_X(\beta)R_Z(\alpha)$. With this formalism, we obtain that a generic rotation in the Hilbert subspace at 2 qubit generated by from the feasible solutions, and considering only real amplitudes can be defined as

$$U(\alpha, \beta, \gamma) = e^{-i\gamma\tilde{L}_y} e^{-i\beta\tilde{L}_x} e^{-i\alpha\tilde{L}_y} \tag{5.98}$$

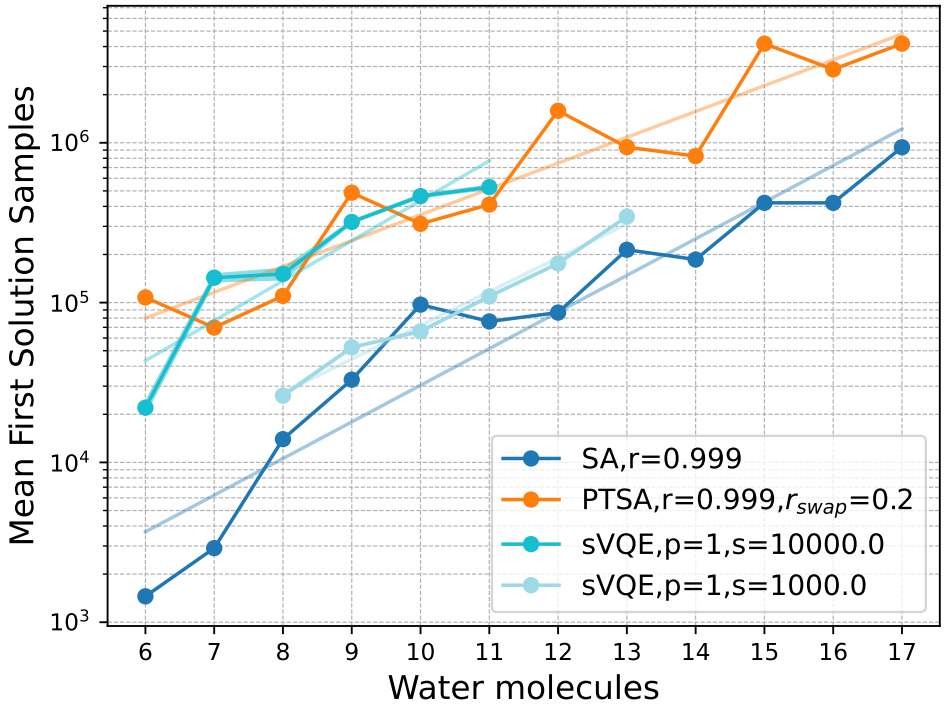


FIGURE 5.12: Results, in terms of MFSS, of a custom VQE (sVQE), that preserves the space of feasible solutions, with 10^3 and 10^4 shots. Each point is the result of the MFSS estimator over 30 runs on the 32 qubit cloud simulator *ibmq_qasm_simulator*. The 30 runs are equally distributed in three CVaR setting $\alpha = 1, 0.5, 0.1$. In this case, all lattices addressed by the algorithm have been solved at least once.

where $\tilde{L}_x$ and $\tilde{L}_y$ are defined in equation 5.95. The choice to compose the generic rotation with $\tilde{L}_x$ and $\tilde{L}_y$ is given by their decomposition into Pauli matrices which allows us to then decompose $U(\alpha, \beta, \gamma)$ in rotations on a single qubit without introducing trotterization approximations. The qubit register is however initialized as in Equation 5.93. After the first layer of operators $U(\alpha, \beta, \gamma)$, p repetitions of layers $U(\alpha_i, \beta_i, \gamma_i)$ are concatenated, interspersed with linear entanglement maps composed of CZ gates so you have a phase operator which therefore preserves the subspace of feasible solutions. The results in Figure 5.12 represent two sVQE settings, one with $S = 10^3$ shots and the second with $S = 10^4$. Both with $p = 1$ and with 30 trials per lattice equally divided with the same values of α tested so far. In the case with $S = 10^3$, testing from 8 to 13 molecules, slightly worse performances are obtained in absolute values compared to the same case of the sQAOA but with all the points, up to 13 molecules, where it is possible to get the solutions in the 30 trials. Scaling,

Method	p	shots	μ	$\delta\mu$
sQAOA	**1**	**1e3**	**0.3706**	**0.0344**
sQAOA	2	1e3	0.6584	0.0082
sQAOA	1	1e4	0.5872	0.0172
sQAOA	2	1e4	0.6485	0.0098
sVQE	**1**	**1e3**	**0.48675**	**0.0009**
sVQE	1	1e4	0.5761	0.0153

TABLE 5.4: Scaling performances in MFSS of the tested quantum solutions with constraints preserved circuit ansatz. The bold results are the most relevant but the sQAOA with $p = 1$ and 10^3 shots has a relative error of 10% and it is calculated on only 4 different problem sizes. The sVQE case with $p = 1$ and 10^3 has a more high accuracy and it was tested from a 16 to a 26 qubit problem. Simulations details in the caption of Figure 5.10, 5.11 and 5.12

on the other hand, is also comparable with simulated annealing. In the case $S = 10^4$, however, the curve is entirely above the respective case of the sQAOA at $S = 10^4$ without bringing any evident benefit.

5.5 Discussion and conclusions

The data shown concerns a still small window of size of the systems tested for clear conclusions. At these dimensions, the topology of the latex influences greatly and differently on the performance of the various methods tested. This introduces several fluctuations that prevent in-depth scaling evaluations. Generally, it can be seen from the results of classical solvers that there is a periodicity of 3 in the number of Molecules and this depends on the topology of the lattices tested and shown in Figure 5.5. In fact, with $N = 10$ and $N = 13$ the lattices are composed of hexagons, 2 in the case of $N = 10$ and 3 for $N = 13$, while for the case with $N = 11$ and $N = 12$, for example, there are spaced molecules that do not belong to closed hexagons. This affects possible frustrations that introduce local minima very close to the global minimum. However, this affects the performance of simulated annealing while for the other solvers, there are other lattices that are more critical. For parallel tempering, a very marked periodicity is visible where lattices with $N = 9$, $N = 12$, and $N = 15$ are more complicated to detect. These are cases where in theory unpaired 2s (which do not belong to formed hexes) should be easier to place. For the QAOA in the version with ansatz constraints preserving, the solution to $N = 12$ has never been found, thus showing the same difficulties as parallel tempering in solving

this case. For smaller sizes, however, the fluctuations do not seem to be very linked to the topology of the lattice, or at least not in a way that is as intuitive as for parallel tempering and simulated annealing. As regards the QAOA, in Figure 5.9 we note that the cases with $N = 6$ and $N = 7$ can be solved much more easily than the others. In these cases, in fact, the dimension of the Hilbert space corresponding to the qubit register is very small ($N = 6$ corresponds to a space of dimension $2^{2N} = 4096$) and therefore it is very likely to sample the basic state with 1000 shots. With $N > 7$ the problem becomes complicated enough to observe real scaling. The introduction of the customized ansatz, however, does not show particular advantages in terms of scaling at such small sizes but instead reduces in absolute value the number of samples to obtain the solution for the first time. However, the results can be improved by considering various points

- The results are very linked to the small size of the gratings considered, by increasing the number of shots it may be possible to deal with cases with $N > 20$ in order to obtain a more accurate evaluation of the scaling.
- The sVQE results suggest that greater expressiveness of the variational loop can help solve larger problems. Introducing more β parameters to train, and therefore defining a more general U_M operator that still preserves the constraints, could improve performance.
- The quantum methods considered were all tested with three different values of α (CVaR parameter). As with parallel tempering, variational quantum methods can also benefit from a parallelization over several α which would have the role of a temperature in this case. In fact, depending on the initial conditions, different *alpha* leads to different performances.

It should be underlined that the QAOA is designed as an approximation for the solution of optimization problems while in this case it was tested as a solver. QAOA was able to find the global optimum up to 26 qubits and in some cases to find the solution with a smaller total number of samples than simulated annealing.

Chapter 6

Benchmarking of non-Turing paradigms

6.1 Assessing the effectiveness of non-Turing paradigms on hard optimization problems

The Von Neumann architecture [156], and hence the Turing paradigm [157], has dominated computing since the time of Charles Babbage [158]. Moore's law [159] predicts an exponentially increasing number of transistors which can be translated to a sustained growth of the computing power. In recent years, this increase has come from specialized and highly parallel accelerators, chiefly graphics processing units (GPUs). The quest for ever-increasing performance now requires new approaches with more efficient energetic profiles and a reduced environmental impact. Alternative architectures need to be investigated in depth towards a more concrete green-computing implementation. Notwithstanding the traditional Turing machines may still be faster, technological signs of progress may quickly change this situation, providing the community with faster architectures that are energetically more sustainable. Non-traditional hardware/software platforms are today available, an example being quantum-gate-based computing assets[44]. These machines introduced by Deutsch [160], are the quantum generalizations of the class of Turing machines. As emerged from the study by Bernstein and Vazirani [161], which introduced the concept of quantum complexity classes, quantum computers can solve certain problems exponentially faster than classical computers. Quantum-gate-based computers are indeed universal computers; Adriano Barenco et al. [53] describe the quantum counterpart of the classical logical gates, called quantum gates, and provide a universal quantum gate set. These quantum gates are implemented through photonic qmodes [162], trapped ions qubits [163], [164], and superconducting qubits. Currently, many important players as Google [165] and IBM

[166] among others [167] are dedicating significant resources to the implementation of such computing systems. It is debated when the *quantum gold rush* [168] will generate practical utility [136], [169]–[171]. In [136] the Google team claimed to have reached Quantum Supremacy with their Sycamore QPU, while in [169] the IBM team contended such claim. Later, [170] and [171] showed how a modern supercomputer can match the aforementioned "quantum supremacy" performances. Besides this quantum, Turing-gate-based machinery, there are other non-classic alternatives such as the Quantum Annealer (D-Wave Systems [172]). There are also different ways to use classical phenomena to perform calculations, e.g. the Memcomputing Machine (Memcomputing Inc. [173]), the Simulated Bifurcation Machine (Toshiba [78]), and approaches such as the Ising machine [174] and p-bits [175]. To fairly evaluate these different computing paradigms, proper benchmarks must be defined to assess the potential of these architectures to solve difficult tasks faster than conventional computers. For instance in [176] benchmarks are defined for assessing the speed-up against classical solvers; in [177] and [178] methods are described to evaluate quantum computing performances. In the first part of the chapter two commercially widely available and paradigmatic non-Turing approaches are critically evaluated and compared to a classical solver (see Figure 6.1). D-Wave and Memcomputing can solve equivalent problems and they represent a quantum (not gate-based) and classical (not quantum) approach, respectively. D-Wave solves Quadratic Unconstrained Binary Optimization problems (QUBO) and is a quantum annealer (often also defined as AQC even if it does not respect the adiabatic condition). Memcomputing solves Integer Linear Programming problems (ILP) by mapping them to a physical circuit, with the physical circuit evolution emulated via a software. D-Wave is the most mature representative of the quantum annealing strategy. Memcomputing represents an approach based on differential equations with digital read-out (similar in spirit to the bifurcation machine [78]). The two approaches can be directly compared with an established baseline, namely the Gurobi suite of optimization methods [179]. These approaches are benchmarked on three difficult, well-known, and combinatorial problems of broad interest that can be expressed in ILP (Memcomputing) and QUBO (D-Wave) format: the Semiprime Factorization problem (FP), the Hard-Assignment Gromov-Wasserstein problem (GWP), and the Capacitated Helicopter Routing Problem (CHRP). The security of large part of the public key cryptography (RSA[180]) is based upon the assumed intractability of FP. GWP is a particularly hard example of the optimal transport theory [181], [182] and is an instance of the well-known Quadratic Assignment Problem [183], a

fundamental combinatorial problem. CHRP is an industrial optimization problem concerning the route scheduling of helicopters. For an accurate comparison, the Mean First Solution Time (MFST) which is the expected waiting time to obtain the first solution of the problem, is used as an evaluation metric. It is thus directly proportional to the expected total amount of monetary budget required to solve the problem. We compare this novel metric to the widely used probability-normalized Time To Solution metric (TTS). TTS has been used to evaluate performances of quantum annealers in [184], [185] and [186]. The results shows that the performances of the solvers strongly depend on the selected set of internal parameters. This is the first time AQCs and Memcomputing machines are compared to each other on such a comprehensive set of industrially and mathematically relevant problems, and it's the first time that the dependence of their performances on internal parameters is analyzed in detail.

6.2 Benchmark problems

6.2.1 Semiprime Factorization Problem (FP)

FP is a notoriously hard problem [187]. Given M, the number to be factored, one must find the prime factors p and q. The best-known classical algorithm for solving FP is the General Number Field Sieve whose complexity is not polynomial [188]:

$$\mathcal{O}\left(e^{\sqrt[3]{\frac{64}{9}}(\log M)^{1/3}(\log\log M)^{2/3}}\right) \tag{6.1}$$

In 1994, Shor proposed an algorithm [189] that solves FP in polynomial time with a gate-based quantum computer. Despite the method's correctness, its application is currently hampered by the limited number of qubits available in gate-based quantum machines. FP has high practical relevance because its prohibitive computational complexity forms the basis of the RSA [180] cryptographic system's security. To implement this problem on the examinated platforms, we formulated FP as an optimization problem. As shown in [190], FP can be converted into a satisfability problem (SAT) problem. The first step is to convert the equation $M = p \times q$ into a nonlinear system of equations, whose unknowns are the fixed point bit representation of the numbers. The nonlinear system is then converted to a linear system with constraints, where auxiliary variables are used with corresponding restraints. This form is an integer linear programming problem and can be run on VMM and Gurobi. To allow porting on D-Wave, we defined a QUBO problem where we recover the *quadratic form*

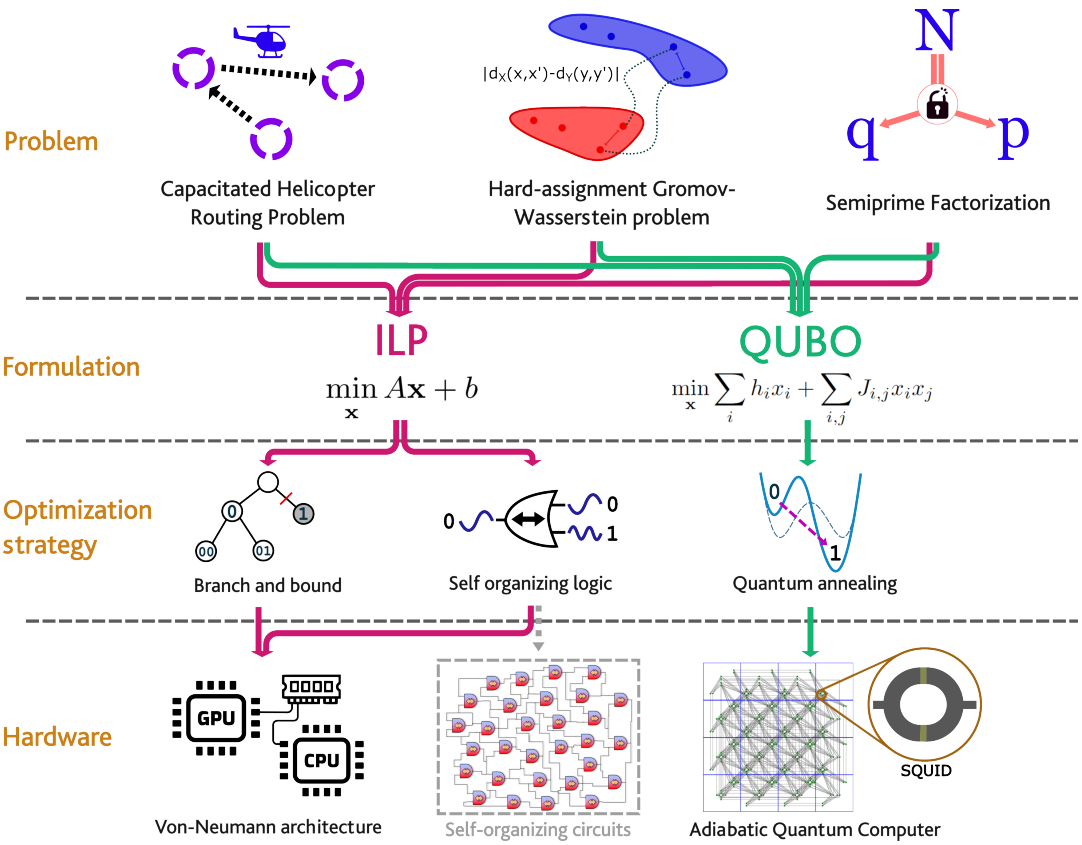


FIGURE 6.1: From the problem to the computing hardware. Three problems are formulated in ILP and QUBO forms and solved with three different solvers: i) the Gurobi optimization software based on branch and bound and other heuristics; ii) a Virtual Memcomputing Machine exploiting self-organizing logic; and iii) a Quantum Annealer. These solvers are physically implemented on hardware based on the Von-Neumann architecture or on an adiabatic quantum computer based on superconducting qubits. Memcomputing machines could be implemented on self-organizing memristor-based circuits.

by squaring the system of equations. The constraints can be imposed via a penalty term.

FP: Implementation details

The semiprime factorization problem is expressed as follows: let M be the number to be factored, one wants to find the prime factors p and q such that

$$M = p \times q \tag{6.2}$$

This problem can be solved as a binary optimization problem. For this, one needs the bit representation of the numbers:

$$M := \sum_{i=0}^{L_M-1} 2^i m_i, \quad p := \sum_{i=0}^{L_p-1} 2^i p_i, \quad q := \sum_{i=0}^{L_q-1} 2^i q_i \tag{6.3}$$

in which $m_i, p_i, q_i \in \{0, 1\}$.
The bit sizes L_p and L_q are unknowns, but one can arbitrarily consider $p \geq q$, from which it follows that $L_q \leq \lceil \frac{L_M}{2} \rceil$. To reduce the problem's variables, it is useful to consider that, since p and q are prime numbers, then $p_0 = q_0 = m_0 = 1$. Moreover, since the problem's complexity is not changed by a priori knowledge of the lengths L_p and L_q, then the most significant bit of p and q can be fixed to 1 if one intends to evaluate the problem's scaling, as in this case. Since VMM solves optimization problems in the ILP formulation, the SAT problem (derived from the equations 6.2 and the equation 6.3) must be converted to a linear version. An ILP formulation of a semiprime factorization problem is derived from the Column-Based Procedure [191]. The Column-Based Procedure, or Multiplication Table Method [192], involves splitting the equation 6.2, written in the binary form, into at most $L_p + L_q + 1$ equations. The factors of equations 6.2, rewritten according to the definitions in equation 6.3, can be grouped by collecting those terms with the same powers of 2 as coefficients. Therefore the final system comprises up to $L_M + 1$ equations:

$$\sum_{j=0}^{L_q-1} q_j p_{i-j} + \sum_{j=1}^{i} c_{i,j} - m_i - \sum_{j=1}^{L_i} 2^{j-i} c_{i,i+j} = 0\,, \quad 0 \leq i \leq L_M \tag{6.4}$$

where i is column index of the corresponding multiplication table $p \times q$ or, in other words, the power of 2 that unites these terms of the i-th equation. The number $L_i = \lceil \log_2 (L_q + i - m_i) \rceil$ is the number of carry variables introduced in the i-th equation.

The carry variables $c_{i,j}$ were added and subtracted from equation 6.2 to set to 0 the sum of the terms with the same power of 2 and thus obtain the system in equation 6.4. The system of equations is not linear due to the presence of terms like $p_i q_j$. These terms correspond to the logical AND operation whose equation $z_{ij} \equiv p_i q_j$ is equivalent to the following two linear inequalities:

$$\begin{cases} p_i + q_j \leq z_{i,j} + 1 \\ p_i + q_j \geq 2 z_{i,j} \end{cases} \tag{6.5}$$

From equation 6.4 and 6.5, an ILP problem $\mathcal{P}_F$ can be defined as follows:

$$\begin{cases} \sum_{j=0}^{L_q - 1} z_{i-j,j} + \sum_{j=1}^{i} c_{i,j} - m_i - \sum_{j=1}^{L_i} 2^{j-i} c_{i,i+j} = 0 \,, \quad 0 \leq i \leq L_M \\ p_i + q_j \leq z_{i,j} + 1 \,, \quad 1 \leq i \leq L_p - 2 \,, \quad 1 \leq j \leq L_q - 2 \\ p_i + q_j \geq 2 z_{i,j} \,, \quad 1 \leq i \leq L_p - 2 \,, \quad 1 \leq j \leq L_q - 2 \end{cases} \tag{6.6}$$

The optimization problem $\mathcal{P}_F$ is an ILP problem that does not bear a cost function, therefore a feasible solution is also the problem's solution.
To reduce the number of carry variables and the number of equality constraints, the problem's equations $\mathcal{P}_F$ can be rearranged in order to consider blocks of columns of the multiplication table. This method, called Blocks Multiplication Table Method [193], follows the same procedure of the previously cited Multiplication Table Method. But instead of the system in equation 6.4 (where each equation corresponds to one power of 2), the new equations correspond to a group of power of 2. For the implementations on the VMM, Gurobi, and D-Wave, we used the Blocks Multiplication Table Method with a block size of 2 columns. Therefore the total number of equations in the ILP problem is up to $\lceil \frac{L_M}{2} \rceil$.
The ILP formulation was used for VMM and Gurobi, but cannot be implemented on D-Wave devices. For D-Wave, the ILP problem $\mathcal{P}_F$ was converted into the corresponding QUBO (Quadratic Unconstrained Binary Optimization) formulation, which includes a penalty component that takes into account the constraints. The QUBO cost function for D-Wave is then:

$$\mathcal{P}_F^{\text{QUBO}}(p, q, z, c) = \sum_i H_i^2(p, q, z, c) + \lambda \sum_{i,j} R(p_i, q_j, z_{i,j}) \tag{6.7}$$

where H_i^2 are the square of the equalities in equations 6.6 but grouped by blocks of 2 to support the Blocks Multiplication Table Method. The penalty

terms $R_(p_i, q_j, z_{i,j})$ are QUBO terms corresponding to the inequalities:

$$R(p_i, q_j, z_{i,j}) = p_i q_j - 2p_i z_{i,j} - 2q_j z_{i,j} + 3z_{i,j} \tag{6.8}$$

The penalty factor λ was fixed to 2, as in [192], [193]. For D-Wave, we ran only 4 problems for each bit size, (instead of 5) due to limitations in the available computing time.

6.2.2 Hard–assignment Gromov–Wasserstein problem (GWP)

Optimal transport theory deals with the problem of moving mass from one place to another with minimal effort. This effort is accounted by a cost function; the integral of these mass moves has to be minimized[194]. The main constraint here is represented by mass conservation; this leads naturally to approach optimal transport as a mapping problem between probability distributions. The original problem has two main formulations: the hard formulation from Monge and the relaxed formulation from Kantorovich [194]. In both, one assumes that the two metric spaces involved are the same. The Gromov–Wasserstein distance was introduced by Mémoli [181] and it is an instance of optimal transport between metric spaces having different dimensions (nonregistered) [195]. The GWP finds application in generative machine learning [196], [197] and computer graphics [181], [182], among others. Finding this distance is equivalent to finding a permutation matrix that allows this mapping between distributions. In literature one can find several regularized, approximate or simplified forms of the GWP: Authors in [198] introduce the entropic regularization approach and uses it in combination with the Sinkhorn's matrix scaling algorithm for solving the GW problem; another paper [199] considers a variant of the optimal transport problem that restricts the set of admissible couplings to those having a low-rank factorization, achieving a linear time approximation for GWP. We will instead consider the problem's *hard–assignment* version, where the desired mapping between points is bijective and the probability distributions in the two spaces assign equal weight to all points. In this form, the cost function is an instance of a Quadratic Assignment Problem (QAP) [183], which makes it an NP-hard problem in general [200]. As anticipated, one is concerned with mapping two point clouds which belong to different metric spaces. One is given two sets of N points, $S_1 = \{x_1, ..., x_N\}$ and $S_2 = \{y_1, ..., y_N\}$, belonging to two distinct vector spaces, each equipped with a distance, $a_{ij} = d_1(x_i, x_j)$ and $b_{hk} = d_2(y_h, y_k)$. Further, one can define a distance between two pairs of points. Here, we used the squared euclidean metric $d(a_{ij}, b_{hk}) = (a_{ij} - b_{hk})^2$. We ultimately want to

find the permutation matrix γ, such that the following expression is minimized:

$$\mathcal{P}_{\mathrm{GW}}(\gamma) = \sum_{ij}\sum_{hk} d(a_{ij}, b_{hk})\gamma_{ih}\gamma_{jk} \ . \tag{6.9}$$

Since γ is a permutation matrix, the following expressions must hold:

$$\begin{aligned} &\gamma_{ij} \in 0, 1 \quad \forall i, j \\ &\sum_{i} \gamma_{ij} = 1 \quad \forall j \\ &\sum_{j} \gamma_{ij} = 1 \quad \forall i \end{aligned} \tag{6.10}$$

The Gromov–Wasserstein distance is the value attained at the minimum. The implementation therefore uses n^2 binary variables. This problem is already naturally in a quadratic form similar to QUBO, yet is constrained. To ensure γ is a permutation matrix we add the following restraints to the cost function via a penalization technique.

$$\begin{aligned} &\left(\sum_{i} \gamma_{ij} - 1\right)^2 \quad \forall j \\ &\left(\sum_{j} \gamma_{ij} - 1\right)^2 \quad \forall i \end{aligned} \tag{6.11}$$

For VMM and Gurobi, we transform this problem into a corresponding ILP by merging the products in γ into a single auxiliary variable.

GWP: Implementation details

To define the hard–assignment Gromov–Wasserstein problem, one begins by considering two sets of N points, $S_1 = \{x_1, ..., x_N\}$ and $S_2 = \{y_1, ..., y_N\}$, belonging to two distinct vector spaces each endowed of a distance, $a_{ij} = d_1(x_i, x_j)$ and $b_{hk} = d_2(y_h, y_k)$, respectively. One can define a distance between two pairs of points. Here, we used the squared euclidean metric $d(a_{ij}, b_{hk}) = (a_{ij} - b_{hk})^2$. To solve the hard–assignment Gromov–Wasserstein problem, one must find a permutation matrix γ ,such that the following expression is minimized:

$$\mathcal{P}_{\mathrm{GW}}(\gamma) = \sum_{ij}\sum_{hk} d(a_{ij}, b_{hk})\gamma_{ih}\gamma_{jk} \ . \tag{6.12}$$

Starting from the initial formulation $\mathcal{P}_{\mathrm{GW}}$, one can convert the problem into a Integer Linear Programming problem to support VMM and Gurobi. One can

indeed create a linear form by introducing n^4 binary variables such that:

$$z_{ihjk} = \gamma_{ih}\gamma_{jk} \tag{6.13}$$

This strategy is typically used to linearize a Quadratic Assignment Problem [201]. The expression to minimize is now:

$$\mathcal{P}^{\text{ILP}}_{\text{GW}}(\{z_{ihjk}\}) = \sum_{ij}\sum_{hk} d(a_{ij}, b_{hk}) z_{ihjk} \tag{6.14}$$

subject to

$$\gamma_{ih} + \gamma_{jk} \leq z_{ihjk} + 1 \quad \forall i, h, j, k \tag{6.15}$$

to enforce expression 6.13. Inequalities in 6.15 ensure that the condition $z_{ihjk} = 0$ implies $\gamma_{ij}\gamma_{jk} = 0$, while $\gamma_{ij}\gamma_{jk} = 1$ can be realized only if $z_{ihjk} = 1$. In addition, since $d(a_{ij}, b_{hk}) \geq 0$ for every combination of indices, minimizing 6.14 automatically ensures that, if $\gamma_{ij}\gamma_{jk} = 0$, then $z_{ihjk} = 0$. Thus the addition of the constraint in expression 6.15 is sufficient to impose $\gamma_{ij}\gamma_{jk} = z_{ihjk}$. One can reduce the number of variables in the problem by noting that, if $i = j$ or $h = k$, then the product on the right hand of expression 6.13 equals zero. Indeed, in this case, the two elements of the gamma matrix would belong to the same row or column, which means that at least one of them must equal zero. Furthermore, $z_{ihjk} = z_{jkih}$ holds true. Thus, in the cost function, one should consider only the terms z_{ihjk}, such that $i < j$:

$$\begin{aligned}
\mathcal{P}^{\text{ILP}}_{GW}(\gamma) &= \sum_{i>j}\sum_{hk} d(a_{ij}, b_{hk}) z_{ihjk} + \sum_{i<j}\sum_{hk} d(a_{ij}, b_{hk}) z_{ihjk} = \\
&= \sum_{i<j}\sum_{hk} \left(d(a_{ij}, b_{hk}) + d(a_{ji}, b_{kh})\right) z_{ihjk} = \\
&= 2\sum_{i<j}\sum_{hk} d(a_{ij}, b_{hk}) z_{ihjk} \ ,
\end{aligned} \tag{6.16}$$

since the distance matrices are symmetric. Note that the case $i = j$ has already been ruled out by the previous observation. The above approach reduces the number of z–variables from n^4 to $\frac{1}{2}n^2 \cdot (n-1)^2$, that is, the number of z variables is halved asymptotically.

As we already observed, the matrix γ must be a permutation matrix. The sum of each row must equal one, so one gets n equations with n coefficients, resulting in n^2 nonzero coefficients in the constraint equations. Since the same is true for

columns, the total number of nonzero coefficients is $2n^2$. Since VMM is able to implement inequalities only, equations of the form $a = b$ are automatically and internally mapped by VMM into two inequalities $a \leq b$ and $a \geq b$. This this leads to $4n^2$ nonzero coefficients in the constraint expressions. This is an internal remapping of VMM. Both Gurobi and VMM receive the same specification with the equalities as input file.

To implement inequalities, one should consider the condition on γ_{ij} and z_{ihjk} appearing in eq. 6.15. The constraint matrix thus contains $3n^2 \cdot (n^2 - 1)/2$ nonzero elements. The total number of nonzero coefficients in the constraint expressions is $\frac{1}{2}n^2 \cdot (3n^2 + 1)$, which become $\frac{1}{2}n^2 \cdot (3n^2 + 5)$ when implemented on VMM. Asymptotically, the number of constraints goes as $\frac{3}{2}n^4$, while the number of binary variables goes as $\frac{1}{2}n^4$, so that the number of constraints is 3 times the number of variables asymptotically. Similarly to the factorization case for D-Wave, we reformulate the constraints as penalty terms. This leads to following cost:

$$\mathcal{P}_{\mathrm{GW}}^{\mathrm{QUBO}}(\{\gamma_{ij}\}) = \mathcal{P}_{\mathrm{GW}}(\{\gamma_{ij}\}) + \lambda R(\{\gamma_{ij}\}) \tag{6.17}$$

where the first term is the original Gromov-Wasserstein cost and the penalty regularizer $R(\{\gamma_{ij}\})$ is ruled by the $\lambda > 0$ coefficient, which enforces the correctness of the γ matrix:

$$R(\{\gamma_{ij}\}) = -4\sum_{i,j}\gamma_{i,j} + \sum_{h,k,j}\gamma_{h,j}\gamma_{k,j} + \sum_{i,h,k}\gamma_{i,h}\gamma_{i,k} \tag{6.18}$$

$R(\{\gamma_{ij}\})$ correctly attains a minimum if and only if all the following constraints are satisfied:

$$\begin{aligned} \sum_{i}\gamma_{i,j} - 1 = 0 \quad \forall j \\ \sum_{j}\gamma_{i,j} - 1 = 0 \quad \forall i \end{aligned} \tag{6.19}$$

Indeed, starting from 6.19, we can square the left–hand size of both constraints, obtaining two expressions that attain the sole global minimum only when the constraints in 6.19 are satisfied (i.e. when γ is a permutation matrix). Summing those two expressions gives back exactly 6.18. The proper value for λ in 6.18 must be found by careful testing (see Section 6.3.2). This is because a higher or lower value can lead to only the satisfaction of the constraints or only to the minimization of the cost, respectively, while both are desired. The cost function in equation 6.17 contains every possible quadratic term of the variables.

The graph representing the problem is therefore fully connected, while D-Wave Advantage is topologically characterized by a very sparse graph of qubits. The solution to this hardware limitation is to use embedding techniques to link together nearby qubits with a strong ferromagnetic coupling, so that they behave as a single two-state system. With this approach, one can implement highly connected graphs.

6.2.3 Capacitated Helicopter Routing Problem (CHRP)

CHRP is a helicopter flight scheduling problem, in which one aims to minimize the overall flight time. CHRP was introduced in [202], based on the formulation of the Dial-a-ride problem (DARP) [203]. Each flight can have multiple legs to connect offshore oil rigs. The flights are scheduled to transport workers from heliport to rigs, from rig to rig, and from rig to heliport. The problem has limiting constraints, such as the maximum range for each helicopter type and maximum capacities for the weight of the workers and luggage. As a hard constraint, CHRP requires all workers to be transported. CHRP is then a multi-agent routing problem where agents (helicopters) interact through the temporal worker assignment constraints. These characteristics make CHRP very hard, even for small instances, and therefore intractable for real world scenarios. Indeed, CHRP and similar problems such as the DARP are NP-hard in the strong sense [204] since they generalize the Travelling Salesman problem with time windows, which is proven to be NP-complete [205]. Because of its hardness and commercial relevance, multiple heuristics have been developed to provide approximate solutions, using clustering search [206], genetic algorithms [207], and a League Championship Algorithm [208]. CHRP can be cast to an integer linear programming problem with all variables being binary. The problem is then automatically in QUBO form with a null quadratic term and thus usable on D-Wave. The problem size can scale with the number of rigs (locations) and the number of workers, while the maximum number of flights is usually set according to the number of workers. In the Integer Linear Programming format, all binary variables are decision variables. The formulation has two blocks of constraints. One block defines the helicopter routes as cyclic paths in a dynamic graph. The other block defines the assignment of workers to flights.

CHRP: Implementation details

Let us consider a directed graph $G = (V, E)$, where V is the set of vertices and E the set of edges. The vertex with label 0 maps the heliport, and the remaining vertices map the oil rigs. The graph is fully connected since we have legs connecting all rigs and heliport. We define binary variables $v_{r,t,f}$ being 1 if the flight f stops at the location (rig or heliport) r at time t and binary variable $e_{l,t,f}$ being 1 if the flight f includes the leg l departing at time t. The time t is an integer index that only defines the order of the events and not the actual time. Hence, t has a range that goes from 0 to T, with T being the maximum number of legs for a given flight (T can either be set by the user or evaluated as the maximum number of legs a flight can include given the helicopter range). The index $r \in E$ is 0 for the heliport and ranges from 1 to R for the rigs. The index $l \in E$ spans all possible $2R(R+1)$ directed edges. Finally, the f is the flight index, which ranges form 1 to F, with F either set by the user or estimated from the number of passengers. To define the equations, it is also useful to introduce maps $\delta : E \to V$ and $\alpha : E \to V$, which return the departing and arrival vertices of an edge, respectively. We can also define the formal inverse maps $\delta^{-1} : V \to E$ and $\alpha^{-1} : V \to E$, which return all outgoing edges from and

all incoming edges to a vertex, respectively. Using these definitions, we have

$$\sum_{l\in\delta^{-1}(r)} e_{l,t,f} \leq v_{r,t,f} \qquad t \in \{0,..,T-1\}, \forall r, f \tag{6.20}$$

$$\sum_{l\in\alpha^{-1}(r)} e_{l,t,f} = v_{r,t+1,f} \qquad t \in \{0,..,T-1\}, \forall r, f \tag{6.21}$$

$$\sum_{r} v_{r,t,f} \leq 1 \qquad \forall t, f \tag{6.22}$$

$$\sum_{r} v_{r,t+1,f} \leq \sum_{r} v_{r,t,f} \qquad t = \{0,...,T-1\}, \forall f \tag{6.23}$$

$$v_{r=0,t,f} \leq 1 \qquad t \in \{0,T\}, \forall f \tag{6.24}$$

$$v_{r,t,f} = 0 \qquad t \in \{0,T\}, r \in \{1,...,R\}, \forall f \tag{6.25}$$

$$\sum_{r=1}^{R} v_{r,t+1,f} + v_{r=0,t,f} \leq 1 \qquad t = \{0,...,T-1\}, \forall f \tag{6.26}$$

$$e_{l,t,f} = 0 \qquad l \in \delta^{-1}(r=0), t \in \{1,...,T\}, \forall f \tag{6.27}$$

$$\sum_{t=1}^{T} v_{r=0,t,f} \geq v_{r=0,t=0,f} \qquad \forall f \tag{6.28}$$

$$e_{l,t,f} = 0 \qquad l \in \delta^{-1}(r) \cap \alpha^{-1}(r), t \in \{0,...,T-1\}, \forall f \tag{6.29}$$

$$v_{r,t,f} + v_{r,t+1,f} \leq 1 \qquad t \in \{0,...,T-1\}, \forall r, f \tag{6.30}$$

$$\sum_{l,t} d_l e_{l,t,f} \leq range(f) \qquad \forall f. \tag{6.31}$$

This set of linear relations fully defines each flight. Equation (6.20) requires that, if a flight includes a location at a given time, then that flight can have one leg departing from that location at that time. Equation (6.21) requires that, if a flight includes a location at a give time, then that flight must have a leg arriving to that location at that time. Equation (6.22) requires that the flight includes at most one location at time. Equation (6.23) requires that, if at a given time the flight does not include any location, then it does not include locations at the subsequent times either. Equation (6.24) and (6.25) require that, at time $t = 0$ and $t = T$, if the flight includes locations, they must be the heliport. Equation (6.26) and (6.27) require that, at any time except $t = 0$, a flight cannot have departing legs from the heliport. This means that, if the heliport is visited at a time other than 0, then it must be the last stop. These two constraints are redundant. One would be enough. However, including both help the convergence of solvers like Gurobi since it includes more cutting planes. Equation (6.28) requires that, if a flight has legs, it must stop at the heliport twice. Equations (6.29) and (6.30) require that a helicopter cannot remain two

consecutive times at the same location. Again, these constraints are redundant but they help the convergence of solvers. Finally, equation (6.31) requires that a flight cannot exceed the range associated with the corresponding helicopter. We note that these constraints do not require that a flight has any legs, so we can have empty flights. Moreover, this formulation allows flights to include the same rig visited multiple nonconsecutive times.

This first set of equations defines closed routes for helicopters in terms of closed paths in a graph. However, they are actually independent routes since we have not yet included the worker assignments. To this end, we define the binary variables $pk_{p,t,f}$ being 1 if the helicopter operating the flight f picks up the passenger p at time t and $df_{p,t,f}$ being 1 if the helicopter operating the flight f drops off the passenger p at time t; the passenger index p ranges from 1 to P. We also define maps $\rho : \{1, ..., P\} \to V$ and $\gamma : \{1, ..., P\} \to V$, which return the pick up and drop off locations for each worker, respectively. Conversely, we define the inverse maps $\rho^{-1} : V \to \{1, ..., P\}$ and $V \to \gamma : \{1, ..., P\}$, which return all the workers that need to be picked up and dropped off at a given location, respectively. Using these variables with the previous variables, the assignment problem can be formalized as:

$$\sum_{t,f} pk_{p,t,f} = 1 \qquad \forall p \qquad (6.32)$$

$$\sum_{t} pk_{p,t,f} = \sum_{t} df_{p,t,f} \qquad \forall p, f \qquad (6.33)$$

$$\sum_{t} pk_{p,t,f} t \leq \sum_{t} df_{p,t,f} t \qquad \forall p, f \qquad (6.34)$$

$$\sum_{p \in \rho^{-1}(r)} pk_{p,t,f} + \sum_{p \in \gamma^{-1}(r)} df_{p,t,f} \geq v_{r,t,f} \qquad \forall r, t, f \qquad (6.35)$$

$$\sum_{p \in \rho^{-1}(r)} pk_{p,t,f} + \sum_{p \in \gamma^{-1}(r)} df_{p,t,f} \leq (|\rho^{-1}(r)| + |\gamma^{-1}(r)|) v_{r,t,f} \qquad \forall r, t, f \qquad (6.36)$$

$$\sum_{t'=0}^{t} \sum_{p} pk_{p,t',f} - \sum_{t'=0}^{t} \sum_{p} df_{p,t',f} \leq capacity(f) \qquad \forall t, f \qquad (6.37)$$

$$\sum_{t'=0}^{t} \sum_{p} w_p pk_{p,t',f} - \sum_{t'=0}^{t} \sum_{p} w_p df_{p,t',f} \leq maxweight(f) \qquad \forall t, f \qquad (6.38)$$

$$\sum_{t'=0}^{t} \sum_{p} wl_p pk_{p,t',f} - \sum_{t'=0}^{t} \sum_{p} wl_p df_{p,t',f} \leq maxluggage(f) \qquad \forall t, f \qquad (6.39)$$

This second set of equations assigns passengers to flights. Equation (6.32) enforces that all workers are picked up and none are excluded. Equation (6.33)

requires that, if the helicopter operating the flight f picks up the worker p at any time, then it must drop off the same passenger at some time. Equation (6.34) enforces that the drop off happens after the pick up. Equations (6.35) and (6.36) enforce that a flight has a leg to the location r if and only if (i.e. the vice versa is enforced too) the helicopter operating that flight either picks up or drops off a worker at that location. In equation (6.36), $|\rho^{-1}(r)|$ and $|\gamma^{-1}(r)|$ are the number of elements returned by the inverse maps. Equation (6.37) requires that the number of passengers on the helicopter operating the flight f does not exceed its maximum capacity at any time. Equation (6.38) requires that the total passenger weight on the helicopter operating the flight f does not exceed its maximum weight capacity at each instant of time. Equation (6.39) requires that the total luggage weight on the helicopter operating the flight f does not exceed its maximum luggage weight capacity at each instant of time. The model is finally completed by the cost function:

$$\min_{e} \sum_{l,t,f} d_l e_{l,t,f}. \tag{6.40}$$

6.3 Results

We challenged the platforms on the above-mentioned orthogonal hard problems: Semiprime Factorization probem (FP), Hard-Assignment Gromov-Wasserstein problem (GWP) and the Capacitated Helicopter Routing Problem (CHRP). To provide a fair comparison we take advantage of the previously introduced Mean First Solution Time (MFST). We perform a scalability assessment, analyze the parameters dependency of the solvers and the time required to deliver a first approximate solution.

6.3.1 Scalability assessment

Here, we discuss the scalability of the analyzed platforms in terms of MFST. This metric allows one to precisely capture the expected waiting time to obtain the first solution, or in other words the expected required computing time in an operative scenario. This is achieved by taking into account explicitly failed runs (no solutions found in the maximum allowed wall time). The timeout for D-Wave and VMM were set based on the available computing budget for each machine. D-Wave was run for dozens of seconds for each problem instance coherently with the granted computational time. The VMM was executed for a few hours while running the GPU backend, and for several days when utilizing

the CPU backend (see each problem section for precise timeout values). Gurobi was granted a timeout of 72 hours (maximum wall time of the IIT HPC infrastructure). For every problem, Gurobi was run on a cluster node with 32 cores (2 physical sockets). To evaluate the scalability and remove possible biases, we define n instances of a problem given a prescribed problem size N. We estimate the MFST of each problem and report the average MFSTs between all the problem instances given one single size. The scaling curve is the set of problems averaged MFSTs and results are reported in log-log scale.

Semiprime Factorization problem (FP)

We defined $n = 5$ different problem instances (five different p, q pairs). To run the benchmark on Gurobi, we randomized 5 times the seed for each problem instance, for a total of 25 runs per problem size. In Figure 6.2a, a plot with the average MFSTs of the performed runs is shown. Each point is the average MFST of the problem instances belonging to the same bit size N. The average MFST for VMM was estimated based on the same 5 instances and 120 different seeds for each instance. This setting was required to better estimate the solution probability. We used the standard CPU backend of the Memcomputing Software As a Service platform (Saas). For each run, VMM simulated 2 replicas of the circuit corresponding to FP, with a maximum timeout of 10 hours. To solve the problem, VMM usually runs a Monte Carlo algorithm to explore the space of the circuit parameters before simulating it. For this specific problem, no parameter space exploration was performed and the number of Markov chains was set to 120 to perform the calculation of the 120 different seeds in parallel on 60 cores. Hence, all the runs correspond to the same single circuit topology and parameter set. We analyzed the scaling with respect to the increasing problem size, namely the bit size in the interval $[14, 64]$. Gurobi was very effective in relative terms (see MFST in Figure 6.2), since it performed a quick presolve of about half of the instances under consideration. Gurobi's ability to reduce the number of variables and constraints of the optimization problem depends on the instance and thus on the semiprime to be factorized. For $N = 64$, Gurobi performed a presolve calculation effectively enough to reduce solution time by four orders of magnitude compared to the problems where this simplification was not possible. At $N = 68$, the problems that Gurobi could not simplify exceeded the wall time of 72 hours of computation. For this reason, problems with a bit size greater than 64 were not considered. In the range $N = 37$ to $N = 64$, the MFST for Gurobi scales with a slope of 16.91 ± 0.92 (see Figure 6.2b). All instances between $N = 15$ and $N = 42$ were solved at least once by

VMM. The linear fit of the MFSTs in the range between 37 and 42 bits resulted in a scaling with a slope of 14.64 ± 0.62. Therefore, VMM obtained a slightly better slope with respect to Gurobi, but the overall execution time was still superior, and VMM could not solve all instances for larger bit sizes. We also used D-Wave devices to solve FP in the interval between 14 and 17 bits. The number of device queries (which can be considered as the number of different seeds in a quantum device) was 10^4 up to $N = 14$ and then was gradually increased up to 8×10^4 at $N = 17$ (~ 12 seconds of computational time). The number of runs was doubled each time that N increased by a unit. We found no significant differences in the slope between D-Wave Advantage and the D-Wave 2000Q, but D-Wave Advantage proved slightly faster, probably because its greater connectivity allows for shorter physical qubit chains [209]. Overall, we found that D-Wave devices could only tackle small instances of the problems, whereas VMM and Gurobi could deal with significantly higher bit sizes. Gurobi was the fastest approach overall.

Hard-assignment Gromov-Wasserstein problem (GWP)

GWP's size and complexity depends on the variable N i.e. the number of points to match between the two sets. To run our benchmark, we defined 5 problem instances for each problem size. For each instance, multiple runs were performed with different random seeds for the solvers. Results are reported in Figure 6.2b. For Gurobi, each instance was solved 5 times with different random seeds, for a total of 25 runs for each problem size. For VMM, each instance was solved 5 times with different random seeds for the solver, for a total of 25 runs for each problem size, as for Gurobi. Contrary to FP and CHRP, we found an advantage in using the GPU backend of the Memcomputing Saas solver. We thus used GPUs to solve GWP on VMM, setting the timeout for each instance to $T_{max} = 1900$ seconds. For D-Wave Advantage, every instance was sampled with thousands of annealing cycles, going from $50,000$ samples for $N = 3, 4, 5$ up to $390,000$ samples when $N = 7, 8$. We report that $390,000$ samples resulted in ~ 60 seconds of access time for D-Wave Advantage. Gurobi solved all the runs for each instance of each problem size up to $N = 16$ points. At $N = 17$, the most computationally intensive instances required a wall time exceeding 72 hours, so we solved every instance only once (a single seed). The corresponding point is indicated with a $'+'$ marker on the plot. The slope for Gurobi was 16.89 ± 0.83. Figure 6.2b shows that the VMM achieved a better overall scaling than Gurobi. The slope of the fit in log-log scale was 5.89 ± 0.50, which was significantly better than the competitors. At $N = 12$ and $N = 13$, Gurobi and

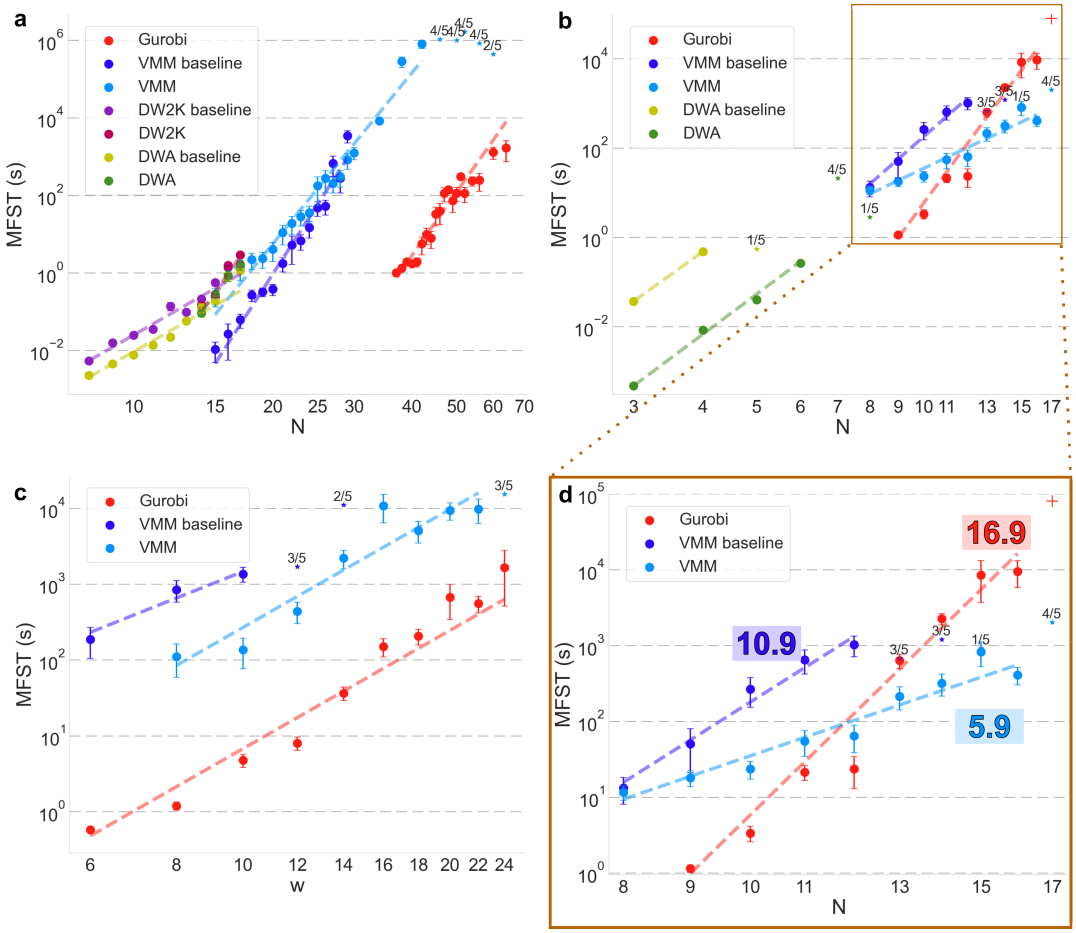


FIGURE 6.2: MFST plots for all the tested computing platforms, in log_{10}, log_{10} scale. Every problem size corresponds to 5 different problem instances using different seeds. Whenever a problem size on a given machine includes unsolved instances, a smaller dot is used and the number of solved instances is shown. The error bars represent the standard deviation. We report both baseline and parameter-optimized scaling results. In the scaling section, we discuss these results. **a**: FP MFST with respect to the number of bits of the semiprime. **b**: GWP MFST with respect to the number of points. The red plus mark is the point $N = 17$ for Gurobi, which was obtained by solving each instance only once, due to time constraints. **c**: CHRP MFST with respect to the number of workers. **d**: Zoom of the GWP plot showing the slopes for Gurobi and VMM solvers for the biggest problems. The VMM with enhanced settings achieved the best performances. The highlighted rounded slope values have the following values and standard deviations: Gurobi 16.89 ± 0.83; VMM 5.89 ± 0.50; VMM baseline 10.93 ± 0.98.

VMM required a similar time to solve the problem, but the wall times quickly diverged for bigger problem sizes. At $N = 16$, the VMM required an average MFST of $409 \pm 107s$ ($\sim$ 7 minutes) compared to the $9560 \pm 3730s$ required by Gurobi, i.e. VMM was $\sim$ 23 times faster than Gurobi. VMM was able to tackle problems almost up to the same size as Gurobi, but did not solve every instance of the hardest problem size, $N = 17$. D-Wave could only tackle small instances ($N \leq 8$), thus it is hard to make a sound comparison with other technologies. At $N = 8$, D-Wave Advantage solved only one of the five instances, finding the correct solution for this instance only once over $390,000$ trials. The resulting point (the rightmost in the D-Wave plot) is thus of limited statistical significance. Deeper insights will be achieved when the D-Wave hardware is advanced enough to tackle bigger instances of this problem. Overall, VMM was slightly less reliable than Gurobi in the biggest problem sizes, but showed the best scaling behaviour.

Capacitated Helicopter Routing Problem (CHRP)

We fixed the number of rigs to 25 and varied the number of workers (w) and the pick-up and drop-off locations generated at random. When converted into QUBO and embedded into chimera topology, the problem becomes too large to fit in D-Wave, even for a few passengers, thus the device was not included in the comparison plot. Summary results are presented in Figure 6.2c. We considered five instances for each problem size, and every instance was submitted to Gurobi and VMM with 5 different seeds, for a total of 25 submitted problems for every worker size. We used the GPU backend of the Memcomputing Saas, setting the timeout for each instance to 5 hours. The slopes of the two solvers in log-log scale were very similar for the problem sizes considered (5.2 for Gurobi versus 5.1 for VMM). In contrast to GWP, the CHRP did not show an advantage for VMM in scaling terms. Additionally, while Gurobi solved all instances, VMM was slightly less reliable beacause it could not solve all instances of the biggest size ($w = 24$). Overall, the results of the three benchmark problems show that D-Wave's current hardware can solve only very small instances. VMM managed the biggest problems well in most cases. VMM's scaling was similar to Gurobi, but was superior for one problem. We note that, for GWP, VMM used GPU hardware to achieve significant speed-ups. In contrast, Gurobi, and the branch-and-bound [210] method in general, is not particularly amenable to a GPU implementation. This is important because the ability to exploit GPU architectures may be critical to improving the overall computing time.

Scalability results for TTS

In this section, we report the results using the TTS as metric to measure scalability. The formula that correlates the estimations of MFST with the estimations of TTS is:

$$\overline{T}_{MFST} = \frac{|I| - |I_s|}{|I_s| + 1} T_{max} + \frac{|I_s|}{|I|} \overline{T}_{TTS} \tag{6.41}$$

where

$$\overline{TTS} \equiv \frac{\bar{t}_s}{p}, \quad p = \frac{|I_s|}{I} \tag{6.42}$$

TTS and MFST are identical every time all problem instances are correctly solved, which means $|I| = |I_s|$. This is the case for Gurobi in all the plots. TTS and MFST are also approximately identical for D-Wave. Indeed each D-Wave run time is approximately equal to T_{max}, which is the qpu total access time. For this reason, $\overline{T}_{TTS} \sim T_{max}|I|/|I_s|$. Considering then a large collection of trials, the estimation of MFST is:

$$\overline{T}_{MFST} \sim \frac{|I| - |I_s|}{|I_s| + 1} T_{max} + T_{max} \xrightarrow{|I| \to \infty} \frac{|I|}{|I_s|} T_{max} \tag{6.43}$$

which is approximately equal to $\overline{T}_{TTS}$.
For VMM, there is a slight difference between the two metrics, with TTS estimates generally being equal to or less than the MFST. This behavior is visible in the plot in figure 6.3 when the probability of success of the run is low (e.g. at $N = 60$).
The TTS diverges from the MFST when the probability to solve the problem is very low and the T_{max} is simultaneously rather different from the typical solution time. In these cases, using the MFST allows one by definition to capture correctly, in absolute terms, the average time needed to get the first solution. This metric should thus be preferred to the TTS.

6.3.2 Parameters dependency

D-Wave and VMM allow users to tune several parameters that directly affect the dynamic of the physical and simulated device, respectively. For each machine, we identified a single problem size on which to execute a tuning protocol. Those optimal parameters were then used for all the other problem sizes, improving performances. Below, we systematically compare the scaling results using the default parameters against their tuned version.

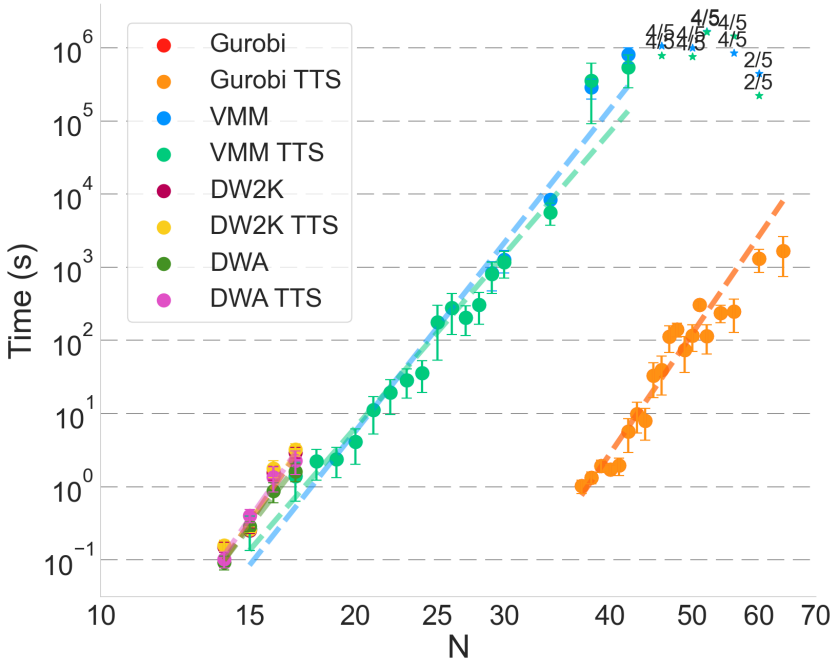


FIGURE 6.3: Comparison between MFST and TTS as metrics to estimate the computational cost to solve FP. The TTS slightly underestimates the scaling for VMM. Plot in $\log_{10}, \log_{10}$ scale where N is the number of bits of the semiprime.

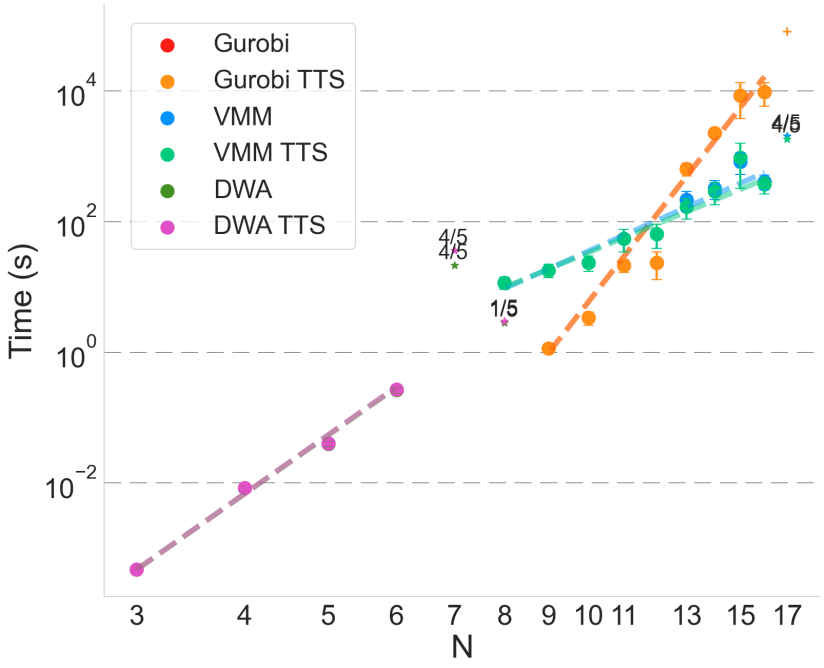


FIGURE 6.4: Comparison between MFST and TTS as metrics to estimate the computational cost to solve GWP. The two metrics are close to identical for each solver. Plot in $\log_{10}, \log_{10}$ scale where N is the number of points.

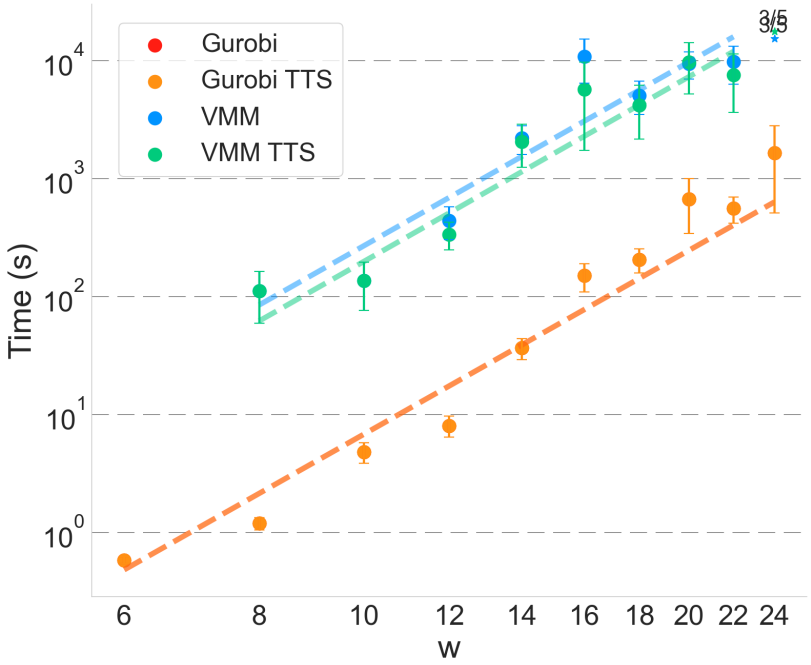


FIGURE 6.5: Comparison between MFST and TTS as metrics to estimate the computational cost to solve CHRP. The TTS slightly underestimates the actual expected cost to reach a solution for the first time for VMM. Plot in $\log_{10}, \log_{10}$ scale where w is the number of workers.

Memcomputing tuning

The VMM User Interface (UI) provides access to two main classes of tunable parameters. The first class sets the physical features of the circuit. The second class sets the simulation settings. The parameters that set the physical features, and hence the internal dynamics of the VMM, are 19. They represent electronic element characteristics like resistances, capacitances, and transistor model parameters. On the other hand, the simulation is performed according to several *simulation parameters* used to set the control unit of the VMM [173]. Examples are the total timeout for a job, the amount of virtual time to simulate, and limits for the wall time. The VMM UI includes a parallel tempering (replica exchange Markov Chain Monte Carlo) to test and optimize multiple circuit physical parameters. This is called the Dynamic Parameter Search (DPS) mode for the VMM. It runs multiple circuit realizations using different physical parameters. Each of these realizations are a Markov chain for the parallel tempering. Each Markov chain changes the physical parameters, simulates the circuit, returns a score based on the VMM performance for that parameter set, and accepts or rejects the change following the Metropolis–Hastings algorithms with adaptive temperatures. The user can set the number of Markov chains, the number of

iterations, the standard deviation, and the number of parallel processes. This latter hyper-parameter allows one to run multiple Markov chains in parallel. The hardware used were virtual machines running on Google Cloud (Intel Xeon 60 virtual core processor for the CPU VMs and NVIDIA 8 V100 GPUs for the GPU VMs). The number of parallel processes were therefore always set to equal or double the virtual machine cores or number of physical GPUs, depending on the type of virtual machine used. During the tests, it was clear that the physical parameter tuning would heavily impact the VMM performance. In our tests, six distinct rounds of DPS with 1000 iterations each were performed for the benchmark problems. Based on the problem-solving performance, the best physical parameter sets from each DPS round were selected and used to initialize the chains for the next round. In each round, we checked if the boundaries for the physical parameters needed to be enlarged or shrank based on the score distribution. The final result was a collection of sets of physical parameters. Multiple sets can be useful because the user can then run multiple realizations of VMM, increasing the convergence and so obtaining better results. Once good samples of physical parameters were found for one problem instance, we used these parameters to run the other problems of the same type but different sizes and instances. We did this using the Dynamic Solution Search (DSS) mode. However, once we had found the physical parameters, a few extra parameters for the VMM control unit were tuned. To describe these parameters, it is useful to briefly detail how the VMM control unit works. The control unit sets up the VMM, assembling and connecting the self-organizing gates to embed the problem. It sets the virtual time, checks the limit of wall time, and initializes the circuit components (e.g. capacitors, transitors). This initialization is set at random by default, but the user can feed a custom initialization (warm start) or use the rounding of the solution of the ILP relaxation. For our tests, we used random initialization. Moreover, the control unit can also restart the circuit, using a perturbation of the best solution found in the previous simulation. Therefore the user can set up the number of iterations where a restart will be operated, how to perturb the best assignment to use as an initial condition for the restart (the "switch fraction" parameter), the virtual time of each iteration, the limit to the wall time, and the total time out. Other advanced parameters for the control unit can be set, e.g. a target for objective function (when/if reached, the simulation ends and the VMM returns) and the number or replicas of the circuit. Each circuit actually comprises several coupled interconnected replicas of itself to enhance the convergence. In our tests, we used 2 replicas. Figure 6.2a shows how the parameter tuning affected the MFST of VMM in

PF, compared to the baseline. Figures 6.2b and 6.2c show the same for GWP and CHRP, respectively.

D-Wave tuning

We tuned the following parameters for the D-Wave devices:

- t_{ann}: the annealing time, namely the time duration of a single annealing cycle.
- λ: the regularization term that accounts for the penalty on the constraints. For the factorization problem, this parameter was obtained from previous literature [193].
- $c = J_{\text{chain}}/Q_{\max}$: the ratio between the intensity of the chains coupling, J_{chain}, and $Q_{\max} = \max_{i,j}\{a_i, b_{i,j}\}$, where a_i and $b_{i,j}$ are the biases and the couplings, respectively, of the QUBO problem Q submitted to the D-Wave device.

The parameter tuning procedure involves running $10,000$ annealing cycles for several combinations of the parameters and choosing the best one in terms of TTS (Time To Solution). For both the problems where D-Wave devices were used (semiprime factorization and hard–assignment Gromov–Wasserstein), we ran the parameter tuning on the biggest problem size.

Semiprime Factorization problem (FP)

For VMM, we considered $N = 29$ as the tuning point. In Figure 6.2a, a comparison between the results before and after the parameter tuning is shown. Instances $N \in [30, 42]$ were not solved using default parameters. After the parameter tuning, all the instances up to $N = 42$ were solved in less than $\sim 4 \times 10^3 s$. For D-Wave, the size $N = 14$ was used as the tuning point. The tuned parameters didn't result in an improved solution probability. The tuning procedure involved c and t_{ann}, at the point $N = 14$. We tested a total of 32 different combinations for c and t_{ann}, which can be found in table (6.1).
This strategy was used for both the available devices D-Wave 2000Q and D-Wave Advantage. The best value found was $c = 0.598$ for D-Wave 2000Q and $c = 0.490$ for D-Wave Advantage. Finally, the best annealing times found were $t_{ann} = 8$ for D-Wave Advantage and $t_{ann} = 1$ for D-Wave 2000Q (additional annealing times, namely $t_{ann} = 16, 32, 64$, were tested, keeping the c value fixed, but without any visible improvement).

Parameter	Values tested
c	0.330, 0.402, 0.490, 0.598, 0.729, 0.888, 1.083, 1.320
t_{ann}	1,2,4,8

TABLE 6.1: Set of tested values for D-Wave and semiprime factorization

As shown in Figure 6.2, the parameter tuning for $N = 14$ did not lead to sensible improvements in TTS for instances of size $N > 14$. Therefore, for the semiprime factorization problem, the optimized D-Wave parameters did not exhibit transferability.

Hard–Assignment Gromov-Wasserstein problem (GWP)

In the GWP, VMM without any parameter tuning solved only one of the five instances for $N = 15$, with no instances solved for $N = 16$ and $N = 17$ (see Figure 6.2b). The parameter tuning was performed on one instance at $N = 16$, enabling VMM to solve all other instances for $N = 16$ and all but one instance for $N = 17$. The tuning process also sensibly reduced the computing time for all $N < 16$ cases. This means that for GWP the parameter tuning of VMM shows good transferability: spending computational time to get the right parameters is an effort that systematically boosts the solver's ability to tackle new GWP instances, even at different problem sizes. The most remarkable effect was at $N = 12$, where the MFST of VMM was reduced by a factor of 16.
The effect for D-Wave Advantage was even more remarkable: the tuning procedure at $N = 6$ reduced the MFST by more than one order of magnitude at all problem sizes. The highest speed-up ($\sim$ 78 times faster) was found at $N = 3$. To tune the parameters, we used the $N = 6$ point. We tested a total of 64 different combinations for c and λ, whose values are reported in table (6.2). Here, we additionally tuned λ to limit the total needed computing time to stratify the parameter selection. That is, first we selected c and λ and then we optimized the annealing time. Results are shown in figure 6.6, where the color scale encodes the percentage probability that the solver found the global optimum. Figure 6.7 correspondingly shows the percentage of samples in which the constraints are satisfied (i.e. the solution corresponds to a correct permutation matrix). We observe that a high probability of satisfying the constraints does not always correspond to a high probability of finding the global optimum,

Parameter	Values tested
c	0.330, 0.402, 0.490, 0.598, 0.729, 0.888, 1.083, 1.320
λ	4,6.5,10,16,25.5,40,64,101

TABLE 6.2: Set of tested values for D-Wave and Gromov-Wasserstein

although the two quantities are partially correlated. Indeed, according to figure 6.7, one should choose $\lambda = 25.5$ or $\lambda = 40$, while in figure 6.6 $\lambda = 16$ seems a much more robust choice. Since the aim is to fully solve the problem, we chose $\lambda = 16$. The value of c was easier to pick since $c = 0.598$ produces consistently better results in both figures 6.6 and 6.7. Keeping $\lambda = 16$ and

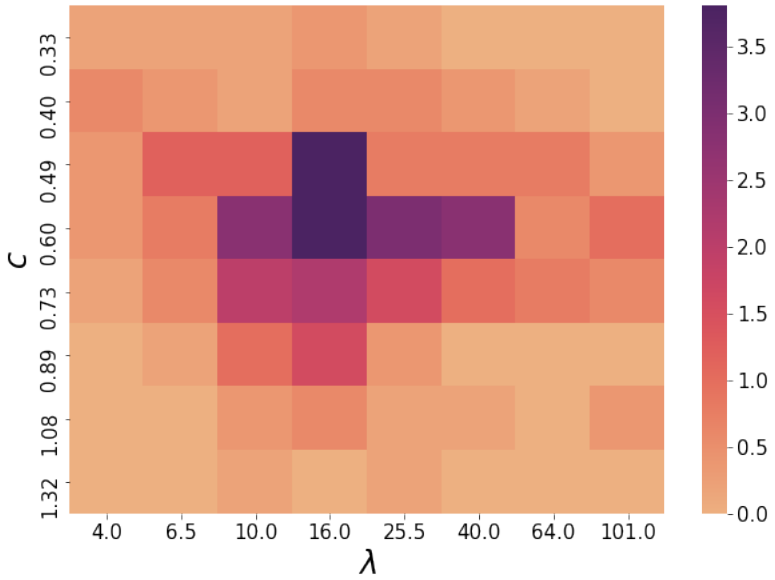


FIGURE 6.6: Test performed on the D-Wave machine for $N = 6$ for GWP. The color of each box encodes the percentage probability that the global optimum is found.

$c = 0.598$ fixed, we performed 10,000 annealing cycles for five different problems with $N = 6$, using seven different annealing times. Figure 6.8 represents the TTS versus t_{ann}. Increasing t_{ann} steadily reduces the TTS, in contrast to the semiprime factorization case. This dependence of TTS vs t_{ann} is in accordance with the adiabatic theorem, which predicts that increasing the annealing time will produce a better solution [63], [64]. Nonetheless, lower annealing times are useful for decoupling the quantum state from the outer environment. Indeed,

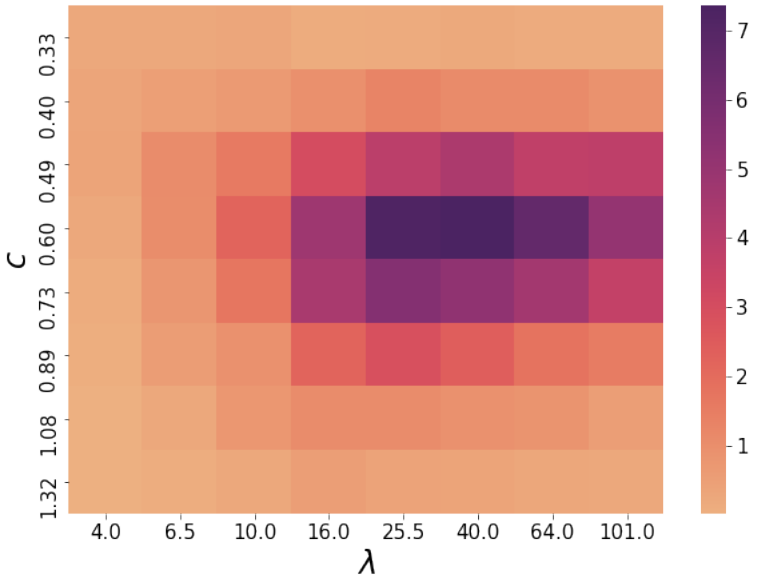


FIGURE 6.7: Test performed on the D-Wave machine for $N = 6$ for GWP. The color of each box encodes the percentage probability that the matrix used in GWP is a valid permutation matrix.

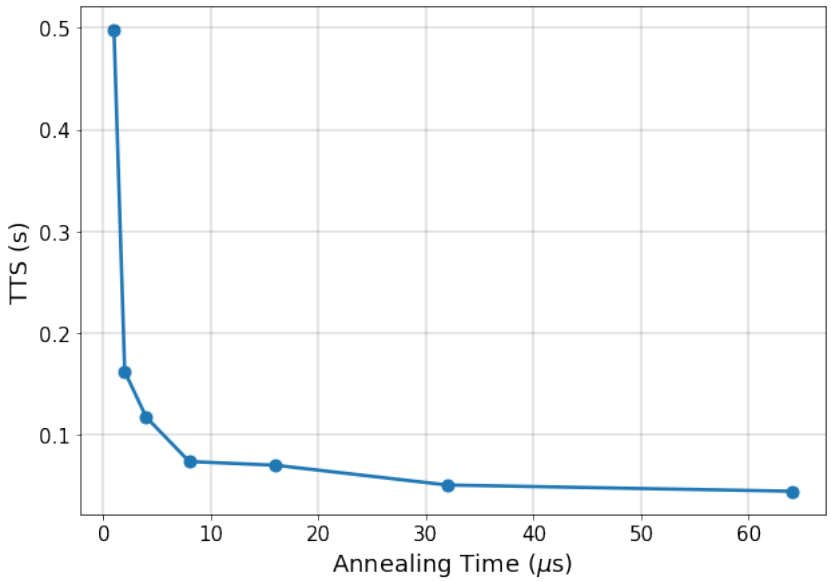


FIGURE 6.8: Test performed on the D-Wave machine for $N = 6$, $c = 0.598$, $\lambda = 16$ for GWP. For each annealing time (x-axis), we considered 5 different problems of that size and we performed 10.000 annealing cycles for each. The values on the y-axis represent the average TTS.

for flux qubits (the technology used in D-Wave), decoherence times are usually measured in dozens [211] or hundreds of nanoseconds [212]. Such times are

much lower than the fastest annealing time available on state-of-the-art Adiabatic Quantum Computers (AQC) (0.5 μs on D-Wave Advantage). Nonetheless, some authors have reported that AQCs appear to be surprisingly resilient to noise and imperfections and that they can show evidence of quantum behavior for annealing times in the order of microseconds [172], [213], [214]. Here, the annealing process seemed to benefit from longer annealing times, which could mean that the noise had a limited impact, or that the fluctuations induced by the outer noise boosted the system's ability to reach the global minimum [215]. On the other hand, in the semiprime factorization case, longer annealing times did not improve performances.

Capacitated Helicopter Routing Problem (CHRP)

Parameter tuning allowed VMM to reduce all its MFSTs, peaking to a ten-fold reduction for $p = 10$ (see Figure 6.2c). While the baseline VMM solved all instances up to $p = 10$, VMM with tuned parameters solved all instances up to $p = 22$. The tuned VMM could tackle problems with ~ 3.5 times the number of nonzero elements in the ILP problem matrix, compared to the baseline. The efficacy of the solver was therefore greatly increased, showing good transferability of the optimal parameters. Overall, we conclude that the tested parameter tuning procedures are fundamental for VMM and D-Wave solvers. For some instances of GWP and CHRP, using optimal parameters reduced by more than one order of magnitude the required time to find the global minimum. The ability to obtain an advantage using new computational approaches such as VMM and D-Wave heavily depends on the development of better and automated parameter tuning procedures.

6.3.3 Gap and latency driven analysis

CHRP finds direct application in industry. The ability to quickly reach a good solution, i.e. latency, is more relevant than being able to reach the global optimum in several real-world scenarios. Hence, we analyzed the solver's ability to achieve a certain gap from the true solution in a given amount of time. Figure 6.9a shows the gap from the optimal solution obtained by the two solvers with one minute as maximum wall time. When the solver did not find a feasible solution, the gap was considered 100%. For a number of workers $w = 22, 24$, Gurobi struggled to find a feasible solution in a minute, resulting in gap values up to $89 \pm 31\%$ at $w = 24$. VMM achieved better scores on the biggest sizes, with a $19 \pm 30\%$ gap at $w = 24$. The gap obtained by both solvers exhibits

great variability (as can be seen by the standard deviation) due to the different difficulty of the tackled instances. In Figure 6.9b, we report the time required by the two solvers to find the first feasible solution. When w increases, Gurobi takes a rapidly increasing amount of time to satisfy the constraints. However, VMM finds a feasible solution in a time that seems only mildly dependent on the problem size. For comparison, Gurobi takes 0.32 ± 0.73 seconds for $w = 10$ (first nontrivial case for the solver) and 208 ± 116 seconds for $w = 24$ (i.e. ~ 650 times slower due to increased problem complexity). VMM takes 16.5 ± 7.2 seconds for $w = 10$, and 44 ± 42 seconds for $w = 24$, which is only ~ 2.7 times slower on average.

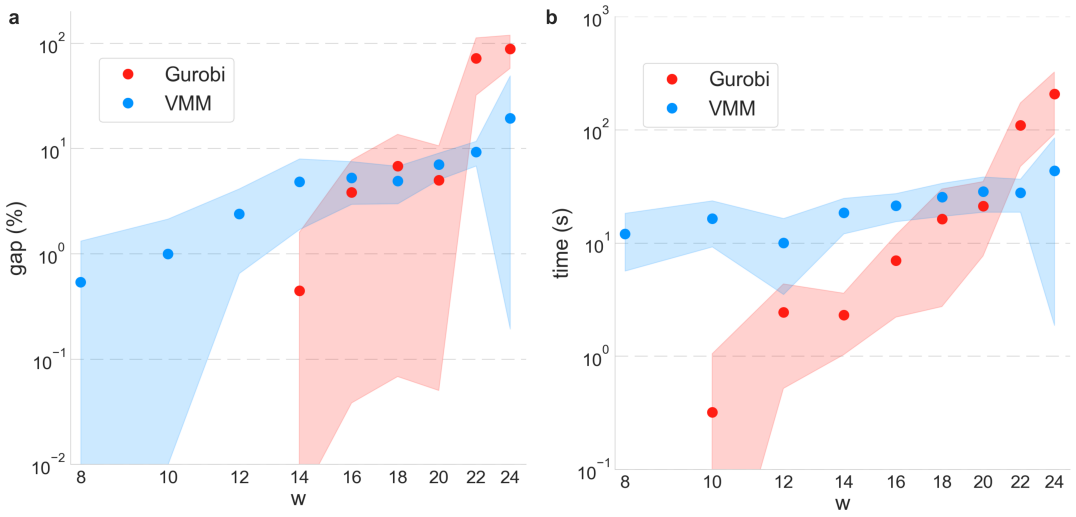


FIGURE 6.9: Performances of VMM and Gurobi on the CHRP problem, in log-log scale. Errorbars have been slipped whenever they included negative values. **a**: average gap percentage with respect to the optimal solution reached by VMM and Gurobi in 60 seconds, versus the number of workers. Gurobi was faster for small instances but its performances quickly deteriorated, while VMM was much more solid as the problem size increased. **b**: time required by VMM and Gurobi to reach the first feasible solution, versus the number of workers. The dependence of computing time on problem size clearly differed between the two solvers. VMM was more resilient to hard instances, resulting in a better wall time for $w = 22, 24$.

6.4 Discussion and Conclusions

Here, we challenged two novel optimization engines on three hard problems and compared them with the state-of-the-art classical Turing Gurobi ILP solver.

Gurobi was fast and reliable for all the tested problems. However, the non-classical solver VMM significantly outperformed Gurobi in absolute MFST and scaling for GWP. This finding shows that the VMM solver is very effective in some Integer Linear Programming classes. Yet, this performance boost was not effortless, being obtained only after parameter tuning. However, the parameter tuning, as for D-Wave, was performed for only one instance and the same parameters were used to solve the other ones. It is also worth mentioning that different parameter settings does not just lower the absolute convergence time, but also lower the slope of the scaling. VMM and D-Wave parameters could have been fine-tuned at different problem sizes and this could have significantly improved the scaling results. However, computing time limitations prevented this per-problem optimization and hence reaching larger problem sizes. In principle VMM has no intrinsic limits in dealing with bigger problem sizes; however, being a heuristic approach its efficiency depends on the parameters setting. A key factor is represented by the invariance of the problem structure at increasing problem sizes. When the invariance is preserved, good parameters at a given size become "portable" to bigger problems. If the structure invariance is not preserved (as in our hard problems) then this requires fine-tuning the parameters at each problem size. On a practical level, VMM was also the best solver for CHRP to find a low gap feasible solution for the hardest instances. VMM's performance, in terms of absolute time but not scaling, is partly due to its use of GPUs. In contrast, Gurobi is bound to CPUs because it exploits branch and cut and heuristics that are not easily parallelizable. VMM, instead, is natively parallelizable, so able to exploit GPU (and other distributed hardware) computing power. The fact that an algorithm is not prone to GPU porting may significantly impact its longevity. Indeed, given the current GPUs power, even a relatively modest algorithmic improvement which favours a GPU porting may render CPU-based method rapidly obsolete and slow. This consideration is a very practical one and holds even if the algorithm improvement does not affect scalability theoretically. In contrast to Gurobi and VMM, D-Wave quantum annealers tackled only small instances of the benchmark problems. D-Wave devices have often been tested on spin-glass models, which are not always representative of several applicative problems. For example, in [216] the authors suggest than spin-glass problems could lead to understating the performances achievable by AQCs; the study in [217] discusses how spin-glass benchmarks could advantage Simulated Annealing approaches with respect to AQCs; Authors in [184] show that industrially relevant problems are much

harder in general than solving spin-glass models. By testing the device on real-world problems cast to QUBO forms, we observed that the solver's abilities are currently limited. This is also because on D-Wave one needs to use penalties to enforce constraints, which is not ideal. The main current bottlenecks for this technology is the thermal noise, but the technology would also benefit from an increase in the number of qubits and a more connected topology. Despite such shortcomings, the tuning procedure we performed is able to accelerate D-Wave performances significantly (up to 78 times for the GWP). Similar approaches could prove to be fundamental in bringing practical applications of adiabatic quantum computers closer. VMM proved to be already today a valuable tool for some problems, also in terms of latency. Promisingly, VMM is only a circuit emulation, whereas a physical circuit would perform much faster and would be much more energy efficient albeit if possibly less flexible than the simulation. Energy efficiency is a key metric that future computing paradigms ought to carefully consider to grant long-term sustainability. We can envision that such non-Turing solvers might find the best application when used in tandem and not in isolation versus a classical counterpart [218]. We expect that a future challenge for High-Performance Computing will be to get the best of the two worlds, achieving speed and maintaining energetic efficiency and programmability.

6.5 Benchmarking of adiabatic quantum computers for in images feature extraction

Adiabatic Quantum Computer, developed by DWave, is one of the most mature quantum computational technologies currently competing to experimentally achieve practical usefulness.
Following the supremacy claim by Google [136] (later contested [170]), and the utility of quantum computing before fault tolerance claimed by IBM [219] (later contested [220]) the D-Wave team recently published experimental evidence that AQCs can achieve an advantage in terms of wall time with respect to SA on a spin-glass optimization benchmark [221]. Such results shed light on the current capabilities of AQCs, but the benchmarking strategy chosen, which is widely adopted in the literature, follows two popular tendencies that contain potential flaws. The first is comparing AQCs performances to simulated annealing algorithms, which cannot be considered a competitive classical approach in most cases. The second is choosing a spin glass problem as a benchmark, which can make it difficult to objectively evaluate AQCs capabilities in real-world scenarios

[184], [216], [217]. In [216] the authors show how spin-glass benchmark problems can lead to understating AQCs' performances; in [217] it is discussed how spin-glass benchmarks could advantage Simulated Annealing (SA) approaches with respect to AQCs; on the other hand, in [184] practical real-world problems are shown to be much harder in general than solving spin-glass models. We believe that testing AQCs on applications on the verge of machine learning such as that considered in the following sections, can provide useful insights to better characterize the actual level of readiness of this technology when applied to industrial real-world problems.

In contrast with the popular approach, an exhaustive inquiry concerning the capabilities of quantum computers must necessarily encompass an analysis of the outcomes achieved by applying such technology to real-world industrial problems. In particular, it is relevant to test AQCs on optimization problems suitable to be included in a machine learning application. Indeed, the field of machine learning is one of those that is thought to benefit most from quantum computing [222], [223]. We selected the problem of Feature Extraction [224] (FE), an important branch of computer vision that provides methods to automatically capture meaningful patterns or features associated with images, or image datasets. If properly selected, such features can be used for dimensionality reduction, acting as a basis set for a more compact latent space representation of the original data, which typically turns out beneficial to generate Machine Learning (ML) models with improved classification, recognition, and detection capabilities [225]. Among the most used classical techniques to tackle FE, one can find Principal Component Analysis (PCA) [226]–[229] and Independent Component Analysis (ICA) [230], [231]. Deep learning-based methods like Convolutional Neural Networks (CNNs) [232], [233] and Recurrent Neural Networks (RNNs) [234] also make use of FE techniques. Such approaches can be computationally demanding and may require substantial processing power depending on the size of the dataset involved or the level of sophistication of the selected algorithm.

AQCs offer practical means to explore quantum approaches to FE because of their ability to handle a high number of variables[235], [236], which is enabled by the availability of thousands of physical qubits and connectivity that characterize such devices[237]. O'Malley et al. [238], [239] were among the first to test AQCs to perform FE tasks. In [238], they introduced a workflow to perform Non-negative Binary Matrix Factorization (NBMF), which allows to exploit AQCs to factorize a collection of images as the product of two matrices, one of which must be binary-valued. The continuous matrix is interpreted as

the collection of basis "feature" images, while the binary one is the "weights" matrix that lists which feature images must be summed up to recompose the original images. Such an approach is widely based on the method to factorize matrix-vector multiplication on AQCs previously introduced by Li et al. [240]. In [239], they managed to boost the performances of the quantum FE algorithm by modifying the shape of the annealing schedule, similar to what was done for a similar problem in Ref.[241]. Members of our team have previously worked on applying different quantum computing paradigms [106] comprising AQCs [242], [243] to machine learning tasks. In this novel work, we explore opportunities, challenges, and current limitations of different generations of D-Wave quantum annealers when utilized to address FE tasks using NBMF methods. We compare the legacy lower noise D-Wave 2000Q (now dismissed) with 2000 qubits arranged in a Chimera topology [244] (2kQ), the D-Wave Advantage 4.1 [237] with 5000 qubits arranged in a Pegasus topology [245] (Adv1) , and the most recent D-Wave Advantage2 prototype 1.1 [246] with 500 qubits arranged in Zephyr topology (Adv2). Such devices represent three important steps in the history of AQCs, namely an increase in topology complexity, going from Chimera (8 couplers per qubit) to Pegasus (15 couplers per qubit) to Zephyr (20 couplers per qubit).

The comparison between different quantum annealing hardware allows us to assess the evolution in the computational capabilities of AQCs and provides us indications about the most relevant working conditions for which this technology is expected to challenge classical digital approaches. Our study is inspired in particular by O'Malley et al. [238], where the NBMF methodology was applied to extract salient features from gray-scale images of human faces. Here, we consider instead an extended, more complex, dataset comprising low-resolution RGB satellite images of airplanes, using the Adv1 processor to also double the maximum number of features extracted with the NBMF methodology in [238] (70 vs 35). Additionally, we applied a tuning procedure to choose the optimal values for both the problem parameters and the internal parameters of the AQCs, which allowed us to get better results in the computational time at our disposal. The evaluation is conducted first on the optimization problem involving matrix factorization and subsequently on a feature extraction and image reconstruction task.

6.6 Feature extraction on AQCs

6.6.1 Problem definition

Given a dataset of m images composed by $\sqrt{n} \times \sqrt{n}$ pixels, we seek to find the best possible representation of each image through a linear combination of $k << m$ basis images weighted with binary values, i.e. 0 or 1. In other words, we look for a set of k basis images to optimally span via linear combination the data-space of reference when only binary coefficients are made available. Such a problem can be formally cast in a matrix factorization problem. Given a matrix V of size $n \times m$, with its columns encoding the m flattened gray-scale images of the original dataset, the goal is to determine the non-negative matrices W (real) and H (binary) of size $n \times k$ and $k \times m$ respectively, for which $||V - WH||_F$ is minimum, where the norm is the Frobenius norm (square root of the summation of the square of every element in the matrix). The complexity of the task comes from the fact that both matrices W and H must be optimized to achieve the best possible approximation for V. We call W the basis-image matrix, and H the reconstruction, or latent matrix. The columns of the matrix W encode the flattened basis images used to reconstruct the original images of the database. The columns of the H matrix, instead, encode the binary weights associated with the basis images. These are to be interpreted as the values of the features of the original images in a binary latent space over which to perform classification. A lower number of basis images will generally imply a poorer capability to reconstruct the original images in the dataset. On the other hand, setting a higher number of features will diminish the advantages of FE, and each value in the binary latent space decomposition will carry less information.

6.6.2 Computational procedure

As discussed in Refs. [238], [240], the original problem of factorizing the matrix V into the matrix product WH via NBMF can be recast into an iterative optimization problem where either W and H are optimized one at a time, alternatively. The full problem can hence be decomposed into two consecutive optimization steps as follows:

$$\textbf{find\_W} := \arg \min_{X \in \mathbb{R}^{n \times k}} ||V - XH||_F + \alpha ||X||_F, \tag{6.44}$$

$$\textbf{find\_H} := \arg \min_{X \in \{0,1\}^{k \times m}} ||V - WX||_F, \tag{6.45}$$

where $||\cdot||_F$ represents the Frobenius norm, a measure of the distance between two matrices, and $\alpha \in \mathbb{R}^+$ a free parameter to be tweaked beforehand. The second term in eq.6.44 is used as a regularization component to penalize solutions with large $||X||_F$. Such a term forces the candidate H matrix to be sparse, so that images in V will be reconstructed by combining only some of the basis images in W. The two optimization problems are then solved iteratively until the condition $||V - WH||_F < \epsilon$ is matched, with ϵ being the desired threshold error in the reconstruction of V.

Finding W– the first problem is solved by finding the X matrix that minimizes the quadratic cost function

$$C = ||V - XH||_F^2 + \alpha||X||_F^2, \tag{6.46}$$

with H initialized as discussed in Sec. 6.7. This quadratic form is minimized via the Gurobi [179] mathematical programming solver. Gurobi implements a wide array of heuristics that make it a reference tool for the minimization of quadratic cost functions. It is a licensed software with a free license for academic utilization.

Finding H– as in Ref.[238], the original problem of finding the H matrix that best reproduces V given W can be reduced to a set of independent optimization sub-problems to be solved for each image in the dataset. We can indeed solve:

$$H_z = \arg\min_{\mathbf{q}\in\{0,1\}^k} ||V_z - W\mathbf{q}||_2, \tag{6.47}$$

where $z = 1, 2, ..., m$, and $\mathbf{q}$ is an array of k binary variables. The L^2 norm is used here in place of the Frobenius norm to account for dealing with vectors rather than matrices. Such a problem can be solved by minimizing the correspondent Quadratic Unconstrained Binary Optimization (QUBO) cost function, obtained by squaring the norm in Eq. 6.47:

$$Q(\mathbf{q}) = \sum_i a_i q_i + \sum_{i<j} b_{ij} q_i q_j, \tag{6.48}$$

with

$$\begin{aligned} a_i &= \sum_l W_{li}(W_{li} - 2V_{lz}), \\ b_{ij} &= 2\sum_l W_{li}W_{lj}. \end{aligned} \tag{6.49}$$

FIGURE 6.10: A selection of images of the *aircraft* class.

FIGURE 6.11: A selection of images of the *not-aircraft* class.

Such a cost function is then minimized with respect to the binary variables $q_i \in \{0, 1\}$. We solved the `find_H` step both using an AQC (hybrid quantum-classical workflow) and using Gurobi as for `find_W` (purely classical workflow).

6.6.3 Description of the dataset

We consider a dataset of satellite images that is publicly available under the CC-BY-SA license at `https://www.kaggle.com/rhammell/planesnet`. The dataset is actually composed of two sub-datasets containing respectively 8000 and 24000 image files in *.png* format. These two sub-datasets contain images that were previously classified as either *aircraft* or *not-aircraft*, examples of which are reported in Figs. 6.10, 6.11. The images display a squared aspect ratio and are composed of 20×20 RGB pixels which can be considered representative of the low-resolution images typically provided by low-cost constellation satellites. For the purpose of this work, we only consider *aircraft* images, which are transformed into a grey scale. Each of such images depicts a single near-centered aircraft at various zoom levels, in-plane orientations, and atmospheric and light conditions. Wings, tails, and tips of the aircraft are fully contained in the perimeter of the images in most cases. The number of details embodied in this dataset yields an overall complexity that we expect to capture only by using a relatively large number of features.

6.6.4 Embedding the problem

Any QUBO problem can in principle be submitted to an AQC [247]. The only limitations regard the total number of variables and the topology of the problem, namely the amount and structure of non-zero quadratic coefficients in Eq. 6.48. Given a problem with an acceptable number of variables, a proper *embedding*

procedure is required whenever the mathematical structure of the problem cannot be mapped directly on the AQC topology [248]. Such a procedure involves connecting multiple physical qubits together via a strong ferromagnetic coupling J_{chain}, which makes them behave like a single two-level quantum system, i.e. a logical qubit. The embedding approach augments the effective connectivity for each logical qubit and eventually allows for the mapping onto the AQC of problems with up to all-to-all connectivity. Obtaining a suitable embedding is generally non-trivial, and it is typically addressed through a procedure known as *minor embedding*. [249]. The mathematical structure of the NBMF problems considered in this work implies the presence of k binary variables connected in an all-to-all fashion by the quadratic coefficients b_{ij} appearing in Eq. 6.49. In principle, any of these coefficients could become equal to zero, but this condition can change at each iteration, and in general, most of the coefficients assume nonzero values. For this reason, it is practical to assume the problem is represented by a fully connected topology of k binary variables. A favorable consequence of this hypothesis is that the embedding procedure has to be performed only once for each tested k value. For the sake of completeness, we report the details of the minor embedding at various problem sizes in Table 6.3. The embeddings have been obtained using the `minorminer.find_embedding` function from Ocean, the SDK provided by D-Wave Systems [250].

	2kQ		Adv1		Adv2	
k	q	l_{chain}	q	l_{chain}	q	l_{chain}
10	34	3.40	16	1.60	16	1.60
20	121	6.05	52	2.60	47	2.35
30	273	9.10	115	3.83	93	3.10
40	505	12.62	199	4.97	161	4.02
50	791	15.82	297	5.94	252	5.04
60	-	-	422	7.03	-	-
70	-	-	555	7.93	-	-

TABLE 6.3: Number of qubits (q) and average chain length (l_{chain}) associated with the embeddings at different problem sizes k. Results are obtained running the `minorminer` software implemented in the Ocean SDK[250] multiple times until there was no improvement in the required number of qubits for ten consecutive trials. Missing values in the table correspond to those cases where minorminer did not return any embedding after ten consecutive trials.

6.7 Results

6.7.1 Optimization of the NBMF hyperparameters

The NBMF-based FE algorithm relies on some hyperparameters, namely the number of epochs of the iterative process, the initialization of the H matrix, and the regularization parameter α. The associated values are chosen within a preselected range in order to minimize the reconstruction error $||V - WH||_F$ at the end of the iterative NBMF process. In practice, this is achieved by conducting preliminary runs on a dataset of 625 images with $k = 50$, which are typical values for problem sizes relevant to this work. For this analysis, we exclusively use the Gurobi solver. In this work, Gurobi version 9.5.0 was used, and calculations were performed on a laptop CPU Intel i5-11400H. Preliminary experimental results showed that the reconstruction error between successive NBMF iterations decreases rapidly within the first 5 steps. For this reason, from now on, we set $n_{\text{epochs}} = 5$ as a meaningful number of iterations for the process. As for the initialization of H, we opt for a random uniform filling with a predetermined density H_{fill}, which represents the percentage of ones in it. In Figure 6.12, we show the behavior of the reconstruction errors as a function of α and H_{fill}.

From such analysis, it is evident that the reconstruction error has a mild dependence on these two hyperparameters over the tested ranges. We select $\alpha = 0.1$ and $\text{H}_{\text{fill}} = 5\%$ as the best values to minimize the reconstruction errors. We also report the dependence of the sparsity of the final H after 5 iterations as a function of α and H_{fill}. Figure 6.12 shows the reconstruction error $||V - WH||_F$ as a function of H_{fill} at different α values. The error varies slightly for different values of such hyperparameters, accounting to few percentage points across all the tested combinations for α and H_{fill}. The optimal combination resulting in the lowest reconstruction error was $\alpha = 0.1$ and $\text{H}_{\text{fill}} = 5\%$. Figure 6.13 shows the dependence of the sparsity of the final H after 5 iterations as a function of α and H_{fill}.

6.7.2 Tuning of the AQC parameters

Ocean SDK enables users to adjust multiple parameters that influence the dynamics of the physical device. It has been demonstrated that the performance of AQCs can be significantly boosted upon tuning these parameters, resulting

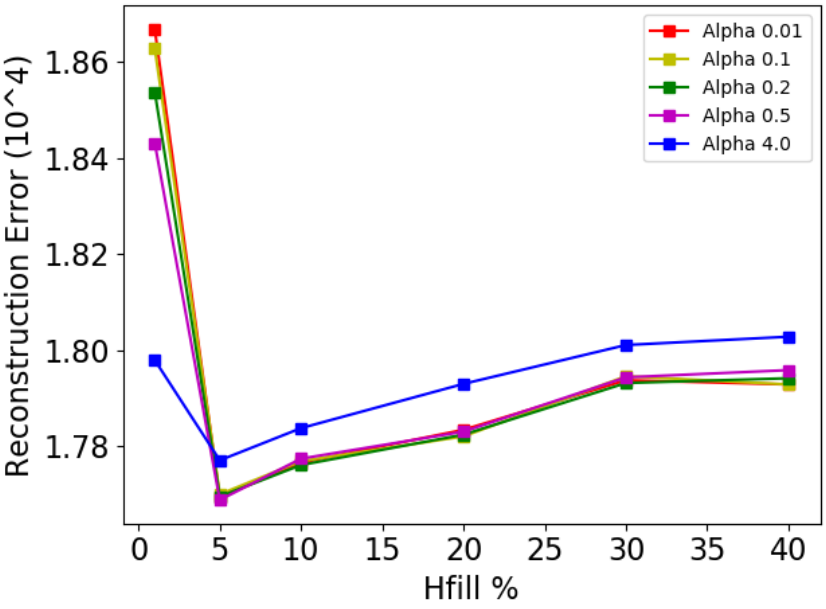


FIGURE 6.12: Hyperparameters selection: Reconstruction error as a function of H_{fill} at different α values. Results are obtained using the classical Gurobi solver for a dataset of 625 images with $k = 50$. Each datapoint is the mean of three independent runs using three different random seeds.

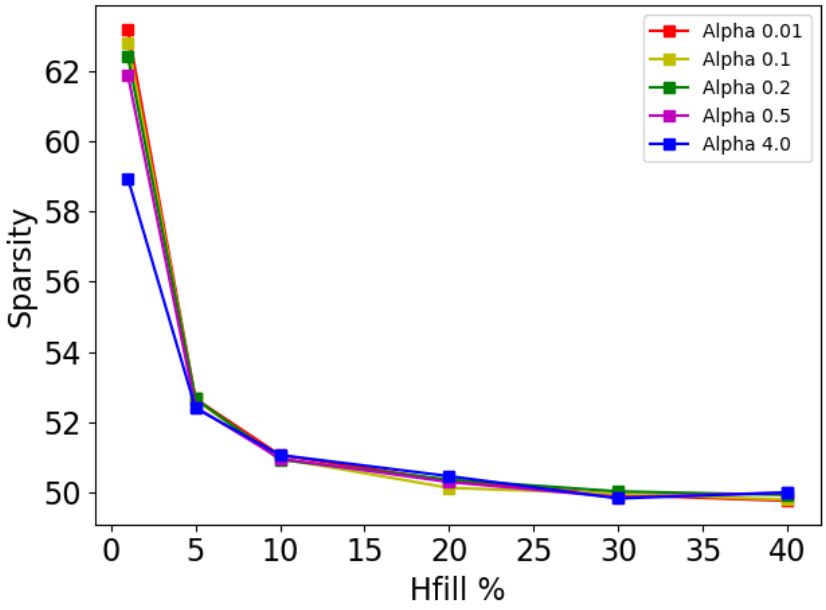


FIGURE 6.13: Final (percent) sparsity of H matrix as a function of H_{fill} at different α values. Results are obtained using the classical Gurobi solver for a dataset of 625 images with $k = 50$. Each data point is the mean of three independent runs using three different random seeds.

in improvements of up to two orders of magnitude in the average time to solution [251]. The parameters with the most substantial impact on performance are two:

- t_{ann}: the annealing time, namely the time duration of a single quantum annealing cycle.
- I_{chain}: parameter controlling the intensity of the chain coupling J_{chain} inside the AQC according to the following equation:

$$J_{\text{chain}} = I_{\text{chain}} \frac{2N_J}{N_{\text{vars}}} \sqrt{\frac{\sum_{\{i,j\},i\neq j} J_{ij}^2}{N_J}}, \tag{6.50}$$

  where N_J is the total number of quadratic connections in the original (not embedded) QUBO problem, and N_{vars} is the number of logical variables in the original QUBO problem (which implies $2\frac{N_J}{N_{\text{vars}}}$ is the average number of connections per logical variable). Note that in Ocean SDK $I_{\text{chain}} = 1.414$ by default, as prescribed in the function `chain_strength.uniform_torque_compensation` [252].

Determining the optimal values for these parameters is a non-trivial task in practical terms. With regard to the annealing time, one could rely on the adiabatic theorem that prescribes maximizing t_{ann} to obtain high-quality solutions [63], [64]. However, it is known that due to the coupling of the quantum state to the environment, longer annealing times can trigger quantum decoherence that deteriorates the overall device performance. Nonetheless, some studies[172], [213], [214] have shown evidence for a quantum behavior of AQCs even for annealing times much longer than the decoherence times of the flux qubits used in the devices under consideration (microseconds vs. dozens [211] or hundreds of nanoseconds [212]). As a result, only a direct tuning of this parameter can provide reliable indications of the optimal values to be used for tackling the problem of interest.

We therefore tested various combinations of t_{ann} and I_{chain}, and we selected the one that resulted in the lowest average gap from the global optimum for `find_H` instances. The gap for each problem instance is defined as follows:

$$\text{gap} = \frac{(C_{\text{best}} - C_{\text{optimal}})}{C_{\text{best}}} \cdot 100\ , \tag{6.51}$$

with C_{best} being the cost of the best solution found with the quantum device, and C_{optimal} being the global optimum of each problem instance found by Gurobi. Note that by definition this gap is bounded $0\% \leq gap \leq 100\%$. We performed

the test on 100 *single-column problems*, where a single column of the H matrix is optimized (see Eq. 6.47). Such test set was created by initializing V with 500 random images from the dataset, and H with random binary digits with $\mathrm{H_{fill}} = 5\%$. The initialization was repeated for 5 different seeds, then executing for each case `find_W` using Gurobi, and then selecting the first 20 column problems of `find_H` for each seed, for a total of 100 problems. Since we aim to use the optimal parameters to enhance the computation at different problem sizes, we decided to perform the tuning procedure at $k = 50$ for Adv1, while we chose $k = 30$ for Adv2 and 2kQ. This way the optimal parameters are obtained at a problem size that is not trivial and at the same time is not close to the maximum achievable size on the hardware (see Table 6.3).

To conduct a fair comparison, we fixed the overall QPU time t_{QPU} available for each combination of parameters. This is achieved by varying the number of annealing samples to compensate for the variable annealing time used in the analysis. Specifically, we decided to set $t_{\mathrm{QPU}} = 0.200s$ for each single-column problem, which we found sufficient to allow for relatively good solutions for the problems at hand while staying within the overall time budget at our disposal. In detail, this value allows for a number of samples per problem ranging from 919 for $t_{\mathrm{ann}} = 100\mu s$ to 1644 for $t_{\mathrm{ann}} = 4\mu s$ (the longest and shortest annealing times tested, respectively). The reason why the number of samples changes so little with respect to t_{ann} is due to the impact of the delay and readout times per sample, which contribute to t_{QPU} especially when t_{ann} is low. Note that t_{QPU} does not include the time required to communicate with the QPU over the Internet, which has a fixed duration. The left image in Figure 6.15 shows the average gap obtained at $k = 50$ for Adv1, while the right image in the same figure shows the percentage of *broken samples*. A sample is considered broken if at least one of the chains in the embedding contains antiparallel qubits.

Figure 6.14 shows a comparison between the heatmaps obtained at $k = 30$ for the three most recent D-Wave hardware models: D-Wave 2000Q (2000 qubits, 8 connections per qubit), Advantage 1 (5000 qubits, 15 connections per qubit), Advantage 2 (model 4.1, 500 qubits, 20 connections per qubit). Such heatmaps have been produced following the exact same procedure described in the main text. Figure 6.15 shows that the average gap appears to be independent of the annealing time. This is in contrast with the heatmap on the right, which shows that, on average, samples obtained using a shorter annealing time contain more broken samples. Apparently, the higher number of samples collected for shorter annealing times compensates for the lower average quality of the collected samples. Additionally, the fluctuations induced by the outer noise could be boosting

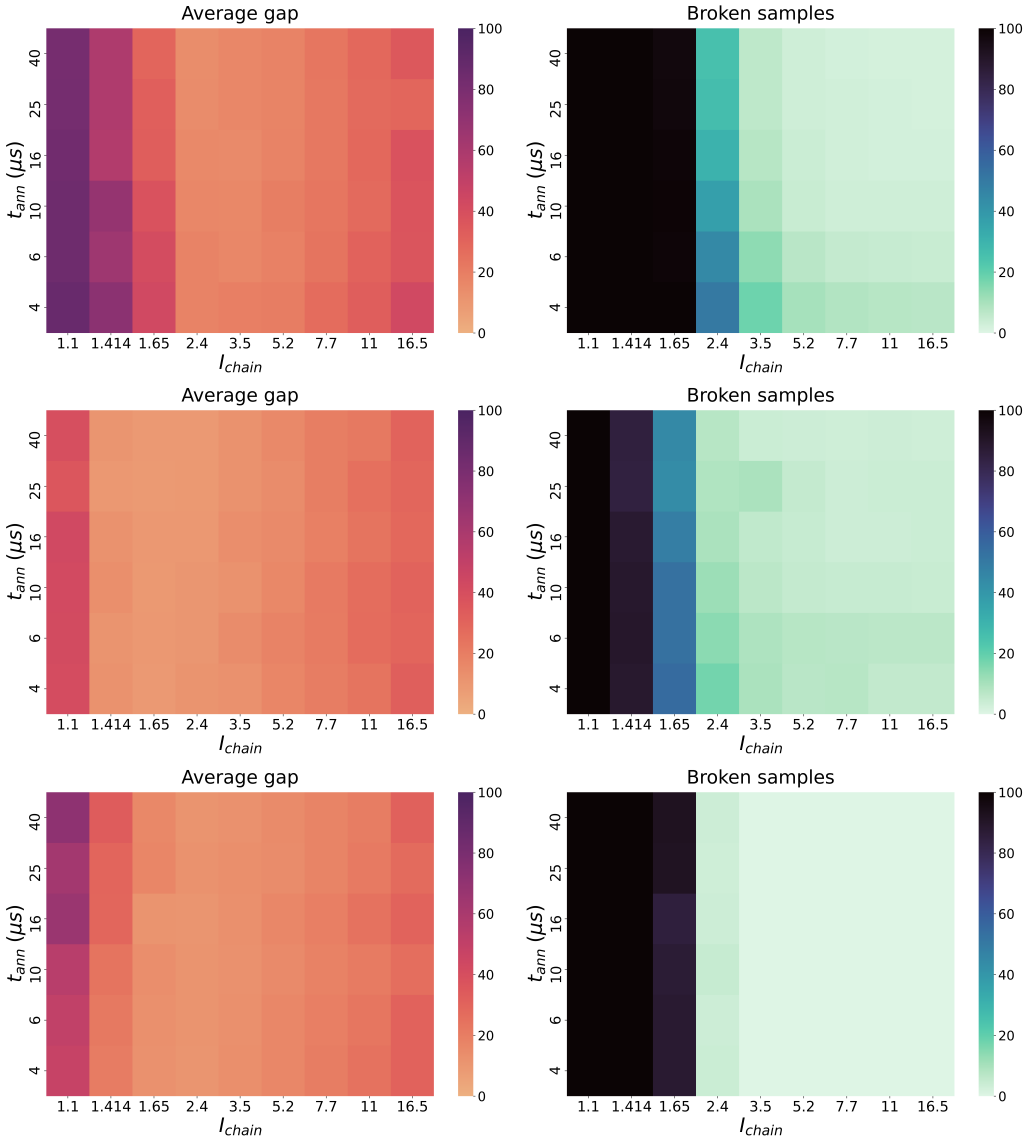


FIGURE 6.14: Heatmaps showing the average gap and average number of broken samples for every combination of annealing time t_{ann} and chain strength I_{chain}. From top to bottom: 2000Q, Adv1, Adv2. All devices were tested at $k = 30$. The experimental procedure to collect the data is explained in detail in the main text.

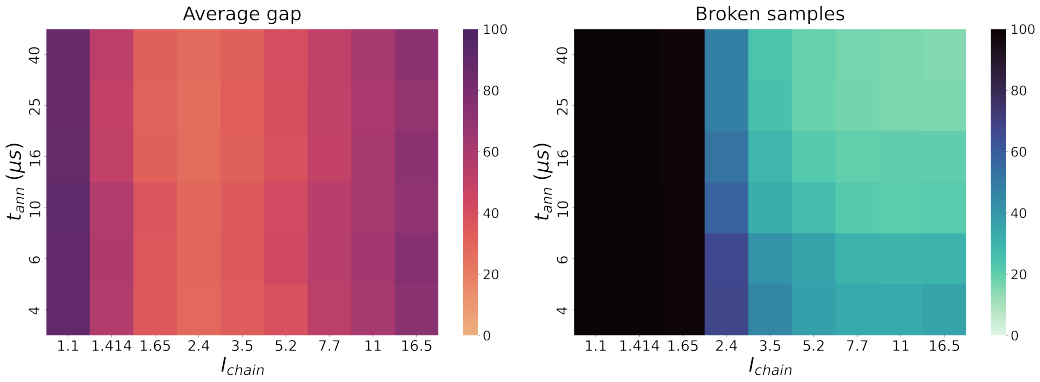


FIGURE 6.15: Average gap and average number of broken samples for every combination of annealing time t_{ann} and chain strength I_{chain}. In the case $k = 50$ for Advantage 4.1. Optimal setting is $t_{ann} = 25$ and $I_{\text{chain}} = 2.4$, corresponding to gap 26.8%.

the ability of the system to reach the global minimum, as suggested in [215]. On the other hand, the chain strength heavily influences the quality of the samples. If a sufficiently strong I_{chain} is imposed, the annealing process should produce samples whose chains are composed of parallel qubits. Such expectation is confirmed by the right heatmap in figure 6.15, which shows how higher I_{chain} values correspond to a lower average number of broken samples. Nonetheless, reducing I_{chain} lowers the energy gap between different configurations, raising the probability of both quantum tunneling and thermal fluctuations.

The optimal value for I_{chain} must balance these effects. The combination $t_{\text{ann}} = 25$ and $I_{\text{chain}} = 2.4$ resulted in the optimal gap of 26.8% and 48.1% broken samples, with only 1.43% of the chains broken. The low average number of single broken chains could explain the limited impact of a high number of broken samples. The optimal setting is to be compared with the D-Wave base setting $I_{\text{chain}} = 1.414$, which resulted in gap values ranging from 49.4% (at $t_{\text{ann}} = 25$) to 58.19% (at $t_{\text{ann}} = 6$), a percentage of broken samples $> 99.97\%$, and a percentage of broken chains $\in [19.25\%, 22.11\%]$ (varying the annealing time). The parameter tuning procedure effectively halved the number of samples containing broken chains, and reduced the number of broken chains by more than an order of magnitude, drastically improving the expected average gap from 49.4% to 26.8%.

6.7.3 Gap dependency on problem dimension and wall time

After tuning the NBMF hyperparameters and the quantum devices parameters, we can now compare the performance of quantum devices provided by D-Wave (hybrid workflow) and the classical solver Gurobi (purely classical workflow) on single-column problems. Our goal is to understand if there are indications of quantum advantage or conditions in which such quantum advantage could be reached earlier, for the set of problems considered in this work. The analysis is conducted at varying problem sizes, varying the value of k from 10 to 70, using the average gap defined in Eq.6.51 as an evaluation metric. For each problem size k, we establish a maximum allowed wall time for the AQCs, which depends on the average time required by Gurobi to solve problems of the same size. The computational time needed by the classical solver serves as an indicator of problem complexity at each size. This specific methodology enables us to employ the average gap obtained by AQCs as a metric to evaluate the quantum solver's ability to match classical capabilities.

We compared the solvers on a total of 250 single-column problems coming from 5 different initializations of matrix V and H (50 problems per initialization). We used the optimal hyperparameters and per-solver optimal I_{chain} and t_{ann} found previously. On average, Gurobi required 0.09 ± 0.02 to solve problems at $k = 10$, and 0.23 ± 0.12 to solve those at $k = 70$.

Figures 6.16**a-c** show the results obtained for the different solvers and different maximum wall times. For each size, when *mult*$= n$ it means that the quantum solver had a runtime n times longer the average time required by the classical solver to find the optimal solution at that problem size. Fig.s 6.16**d-f** present the same data but grouped with respect to the *mult* value so that it is easier to compare the performances of the solvers. Violin plots in Figs.6.16**g-h** show the distribution of the gaps obtained by sampling at different k values using the three D-Wave devices. Figures 6.16**a-f** highlight a monotonic increase. Given that we already corrected the runtime according to the problem complexity, this means that the quantum solvers performance is degrading as k increases, if compared to the classical solver.

We could expect that the k at which the tuning procedure has taken place will be boosted. We performed the optimal parameter search at $k = 50$ for Adv1, while 2000Q and Adv2 have been tuned at $k = 30$. 2kQ and Adv1 at mult$= 0.1$ do indeed show a swift decrease in the average gap at $k = 30$ and $k = 50$, respectively, but the change is not evident at any other multiplier nor in the Adv2

case. This behavior suggests that the optimal parameters we found exhibit good transferability, enhancing the performances even at different k values.
In addition to the previous evaluations based on the average optimal gap obtained on multiple problems, the violin plots in Figures6.16**g-h** allow us to compare the different solvers with respect to the distribution of all samples. In Figures 6.16**g** we can appreciate how, for $k \geq 40$, 2kQ tends to produce samples with a much higher gap if compared to Adv1. The impact of this behavior is confirmed by Figures6.16**d-f**, where the average gap attained by 2kQ rapidly grows for $k \geq 45$, while Adv1 and Adv2 are less affected. This detrimental effect is probably due to the different chain lengths required to embed problems on the three solvers (see Tab.6.3). At $k = 40$ the average length of the chains of the 2kQ embedding rises above 12 qubits, to reach almost 16 qubits at $k = 50$. On the other hand, Adv1 and Adv2 are much closer to each other, reaching 5.94 and 5.04 average length, respectively, at $k = 50$, which results in similar distributions in figure **h**.

6.7.4 Reconstruction error after the iterative process

After having analyzed the quantum hardware performance in solving single-column QUBO problems, we focused on analyzing the performances in solving the full interactive process that constitutes the FE NBMF-based algorithm. To this end, we compared the fully classical and hybrid quantum-classical FE workflows to extract $k = 50$ features from a dataset with $m = 500$ images. For the hybrid workflow, we used Adv1 with 2000 samples per each single-column `find_H` problem, setting the optimal parameters $I_{\text{chain}} = 2.4$, $t_{\text{ann}} = 25$. We performed runs with $n_{\text{epochs}} = 5$ iterations for both D-Wave and Gurobi, which led to a decrease in the reconstruction error $||V - WH||_F$, as shown in Fig.6.17 **d**. Figures 6.17**d** shows that the `findH` step in the hybrid workflow increases the reconstruction error achieved in the same epoch during the `findW` step. This means that, given the W matrix found at a certain iteration, D-Wave is finding an updated H matrix that is worse than the previous one. This is compensated at each new epoch by the `findW` step, in Eq.6.44, yielding to a reduction of the reconstruction error during the overall process. In absolute values the purely classical workflow is observed to outperform the hybrid one.
Figures 6.17**e-f** show an example of dominant feature images (basis images from matrix W) obtained by running the quantum and classical NBMF workflows. The corresponding basis images, shown at different epochs, resemble some sort of spherical harmonics. As expected from the reconstruction score, we can visually verify that from epoch 3 to 5 the feature images are modified only slightly.

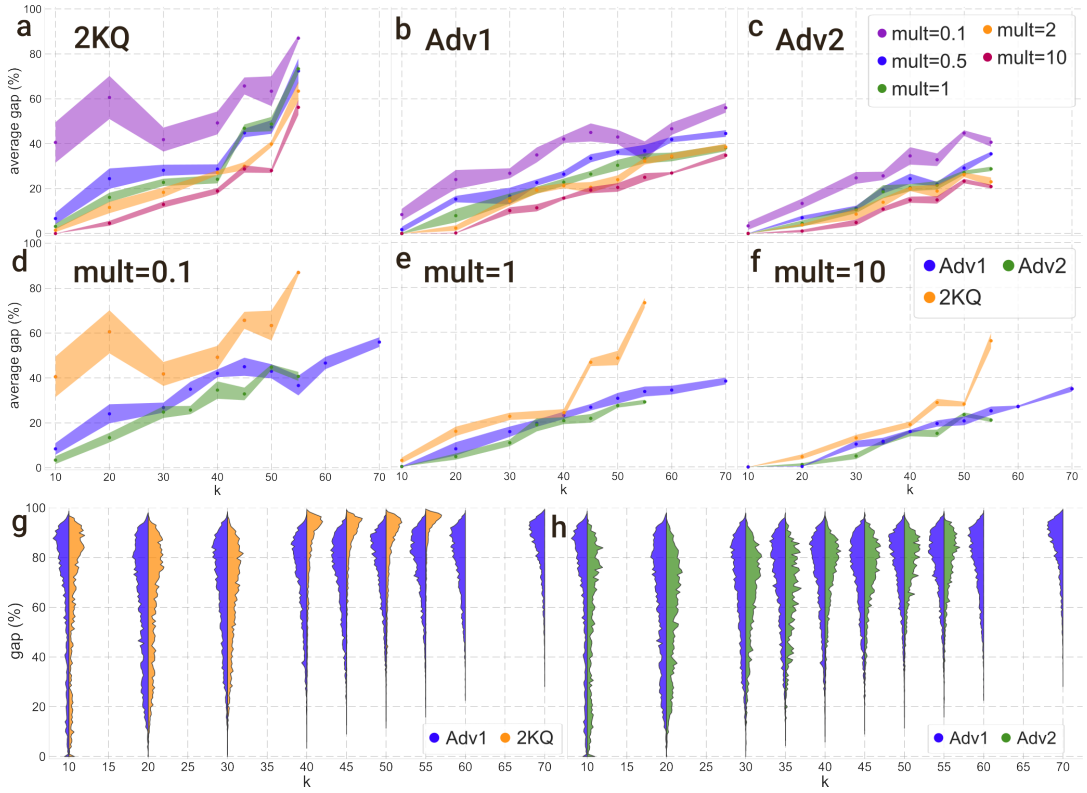


FIGURE 6.16: Performance analysis of the Advantage 4.1 (Adv1), the Advantage2_1.1 prototype (Adv2), and the 2000Q (2kQ) quantum computers. Plots **a** → **f** display the average gap from the exact solution as a function of the problem size k. Each data point is estimated by averaging the best gap obtained by the selected D-Wave QPU on 250 distinct single-column problems. The shaded area represents the standard deviation. The runs were executed using the tuned optimal parameters for each solver. The maximum wall time allowed for each D-Wave run is a multiple of the average time that Gurobi required to solve the same instance of the problem. Plots **g** and **h** compare the distribution of samples produced by the three D-Wave solvers. For every k, 10,000 samples were uniformly extracted from the whole collection of samples obtained from the previous runs on the 250 single-column problems. The samples produced by 2kQ display a distribution peaked towards higher gaps for $k \geq 40$, while samples produced by Adv1 and Adv2 display broader distributions.

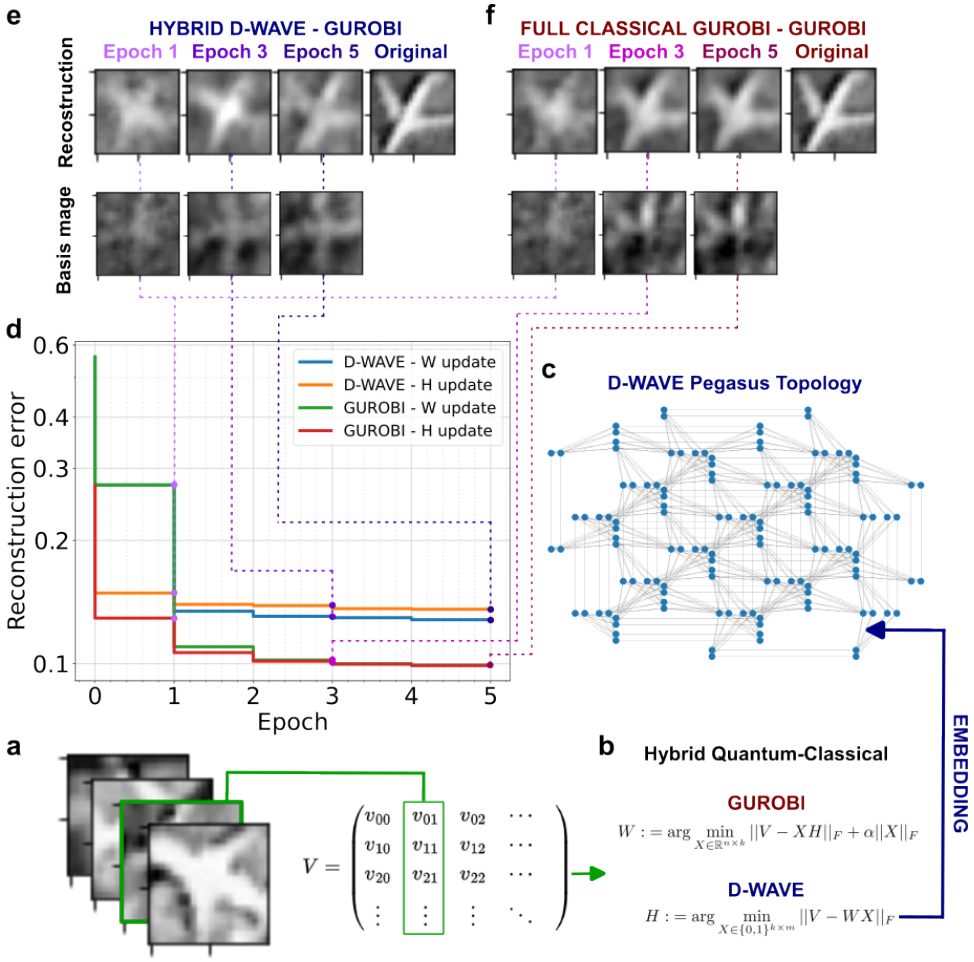


FIGURE 6.17: NBMF algorithm Workflow on AQC (D-Wave) and Gurobi. **a** In the lower-left panel are reported some samples of the $\sqrt{n} \times \sqrt{n}$ satellite images of aircrafts. The m images are flattened and stacked to form the $n \times m$ matrix V. **b** The optimization strategy is based on an iterative updating of the continuous values matrix W (always on Gurobi) and the binary matrix H (on D-Wave or Gurobi). **c** The optimization step to update H is embedded on the Pegasus, Chimera, and Zephyr graphs of the Adv1, 2kQ, Adv2 devices, respectively. **d** The reconstruction error $||V - WH||_F$, in logarithmic scale, after `find_W` and after `find_H` is shown at each epoch for the quantum-classical workflow and the full-classical one. **e-f** Reconstruction of a sampled image at the epoch 1, 3, and 5 respect the original image (series of images above), the images on the left side are the reconstruction with the quantum-classical workflow and in the right side the reconstruction obtained by the fully-classical workflow. The series of images below, instead, contains a sampled basis image, which is a column of W, at the epochs 1, 3, and 5.

The same figures also display examples of image reconstructions across the iteration process. As expected from the reconstruction score, at visual inspection the fully classical workflow provides a slightly sharper reconstructed image.

The hybrid workflow tends to combine more feature images to reconstruct images in V. Indeed, 34.6% of the elements in the final matrix H are equal to 1, as opposed to only 14.6% in the classical workflow. Thus, the hybrid workflow produced a potentially more informative decomposition of the images from the dataset into base features.

6.8 Conclusions

We analyzed the capabilities of AQCs on a task with applications in machine learning, namely performing feature extraction (FE) from a dataset of low-resolution satellite images of airplanes, exploiting an approach based on Non-negative Binary Matrix Factorization (NBMF). The NBMF is implemented through a hybrid iterative algorithm where the portion of the workflow corresponding to the optimization of the binary latent H matrix is offloaded to AQCs. This methodology can be used to perform FE on large datasets thanks to the possibility of decomposing the original latent H matrix optimization problem into a set of independent problems corresponding to single columns of the H matrix.

We considered three generations of AQCs provided by the D-Wave company, namely the legacy device 2000Q, Advantage 4.1, and the most recent prototype Advantage2_1.1. First, we devoted consistent efforts to fine-tuning the algorithm hyperparameters and the AQCs parameters. Using the optimal AQC parameters, we were able to reduce the average number of broken chains by one order of magnitude while halving the expected average gap from the optimal solution (results obtained on Advantage 4.1). Hence, we compared the performance of three hybrid quantum-classical workflows based on the three AQCs against that of a purely classical workflow based on the state-of-the-art Gurobi classical solver running on classical digital hardware. As an evaluation metric, we used the solution gap from the global optimum obtained at fixed runtimes and averaged over a selection of problem instances.

We observed all quantum solvers to provide a similar qualitative behavior in terms of gap performances as a function of the problem size, i.e. k. In particular, we observed the average gap to the exact solution to increase along with k. We can nonetheless appreciate a slight but statistically significant quantitative difference in the performance of the three AQCs, with Adv2 achieving the lowest gap values, Adv1 following close, and 2kQ falling short, particularly on larger problem sizes. This effect is more evident at shorter runtimes. The fast degrading of the performance of 2kQ for $k > 40$ is probably correlated

with the excessive chain length required to embed such problems on the sparse Chimera topology, which also notably affects the distribution of samples with respect to the attained gap. These observations suggest an incremental overall performance with the more recent solver generations, which seems to correlate mainly to the average chain length found by the embedding procedure. Such conclusion supports the idea, common in literature, that the topology of the available hardware currently represents the main bottleneck for adiabatic quantum computation [253]–[255].

During the iterative process, the optimization of H at fixed W made with D-Wave at a relatively large k increases the reconstruction error rather than reducing it as in the case of utilization of a classical solver. For small k, however, the average gap on single instances of the optimization problem remains close to zero for all quantum solvers. In particular, we observed that both Adv1 and Adv2 are capable of solving all the submitted instances at $k = 10$ with 0% gap (i.e. exactly) within the average time required by the Gurobi solver (mult= 1). This result suggests that recent generations of AQCs have finally reached competitive performances for industrially-relevant, small-sized problems.

At least for the set of problems considered here, the classical solvers provide in general still the reference tools. There are conditions, however, where the AQCs start to provide alternative solutions with comparable performances. In this perspective, the first applications where AQCs are expected to become competitive with classical counterparts are those where limited and short running times are required (e.g. real-time or near-real-time applications).

Bibliography

[1] G. Morra and G. Colombo, "Relationship between energy distribution and fold stability: Insights from molecular dynamics simulations of native and mutant proteins," *Proteins: Structure, Function, and Bioinformatics*, vol. 72, no. 2, pp. 660–672, 2008.

[2] R. Alhadeff, D. Assa, P. Astrahan, M. Krugliak, and I. T. Arkin, "Computational and experimental analysis of drug binding to the influenza m2 channel," *Biochimica et Biophysica Acta (BBA)-Biomembranes*, vol. 1838, no. 4, pp. 1068–1073, 2014.

[3] C. F. Reboul, G. R. Meyer, B. T. Porebski, N. A. Borg, and A. M. Buckle, "Epitope flexibility and dynamic footprint revealed by molecular dynamics of a pmhc-tcr complex," *PLoS computational biology*, vol. 8, no. 3, e1002404, 2012.

[4] R. E. Amaro, R. V. Swift, L. Votapka, W. W. Li, R. C. Walker, and R. M. Bush, "Mechanism of 150-cavity formation in influenza neuraminidase," *Nature communications*, vol. 2, no. 1, p. 388, 2011.

[5] Y. Zhang, H. Shen, M. Zhang, and G. Li, "Exploring the proton conductance and drug resistance of bm2 channel through molecular dynamics simulations and free energy calculations at different ph conditions," *The Journal of Physical Chemistry B*, vol. 117, no. 4, pp. 982–988, 2013.

[6] N.-j. Deng, W. Zheng, E. Gallicchio, and R. M. Levy, "Insights into the dynamics of hiv-1 protease: A kinetic network model constructed from atomistic simulations," *Journal of the American Chemical Society*, vol. 133, no. 24, pp. 9387–9394, 2011.

[7] S.-S. Chang, H.-J. Huang, and C. Y.-C. Chen, "High performance screening, structural and molecular dynamics analysis to identify h1 inhibitors from tcm database@ taiwan," *Molecular BioSystems*, vol. 7, no. 12, pp. 3366–3374, 2011.

[8] P. Lagüe, B. Roux, and R. W. Pastor, "Molecular dynamics simulations of the influenza hemagglutinin fusion peptide in micelles and bilayers:

Conformational analysis of peptide and lipids," *Journal of molecular biology*, vol. 354, no. 5, pp. 1129–1141, 2005.

[9] C. Sieben, C. Kappel, R. Zhu, *et al.*, "Influenza virus binds its host cell using multiple dynamic interactions," *Proceedings of the National Academy of Sciences*, vol. 109, no. 34, pp. 13 626–13 631, 2012.

[10] Z. Yang, Y. Nie, G. Yang, Y. Zu, Y. Fu, and L. Zhou, "Synergistic effects in the designs of neuraminidase ligands: Analysis from docking and molecular dynamics studies," *Journal of theoretical biology*, vol. 267, no. 3, pp. 363–374, 2010.

[11] T. Rungrotmongkol, M. Malaisree, N. Nunthaboot, P. Sompornpisut, and S. Hannongbua, "Molecular prediction of oseltamivir efficiency against probable influenza a (h1n1-2009) mutants: Molecular modeling approach," *Amino acids*, vol. 39, pp. 393–398, 2010.

[12] G. G. Dodson, D. P. Lane, and C. S. Verma, "Molecular simulations of protein dynamics: New windows on mechanisms in biology," *EMBO reports*, vol. 9, no. 2, pp. 144–150, 2008.

[13] D. Vlachakis, A. Karozou, and S. Kossida, "3d molecular modelling study of the h7n9 rna-dependent rna polymerase as an emerging pharmacological target," *Influenza Research and Treatment*, vol. 2013, 2013.

[14] C. Peri, P. Gagni, F. Combi, *et al.*, "Rational epitope design for protein targeting," *ACS Chemical Biology*, vol. 8, no. 2, pp. 397–404, 2013.

[15] D. R. Flower, K. Phadwal, I. K. Macdonald, P. V. Coveney, M. N. Davies, and S. Wan, "T-cell epitope prediction and immune complex simulation using molecular dynamics: State of the art and persisting challenges," *Immunome Research*, vol. 6, no. 2, pp. 1–18, 2010.

[16] D. Xu, E. I. Newhouse, R. E. Amaro, *et al.*, "Distinct glycan topology for avian and human sialopentasaccharide receptor analogues upon binding different hemagglutinins: A molecular dynamics perspective," *Journal of molecular biology*, vol. 387, no. 2, pp. 465–491, 2009.

[17] M. Lawrenz, J. Wereszczynski, R. Amaro, R. Walker, A. Roitberg, and J. A. McCammon, "Impact of calcium on n1 influenza neuraminidase dynamics and binding free energy," *Proteins: Structure, Function, and Bioinformatics*, vol. 78, no. 11, pp. 2523–2532, 2010.

[18] S. Gusarov, A. E. Kobryn, S. Stoyanov, and V. Veryazov, "Mathematical challenges in computational chemistry: Multiscale, multiconfigurational approaches, machine learning,"

[19] K. A. Fichthorn and W. H. Weinberg, "Theoretical foundations of dynamical monte carlo simulations," *The Journal of chemical physics*, vol. 95, no. 2, pp. 1090–1096, 1991.

[20] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[21] K. Lindorff-Larsen, P. Maragakis, S. Piana, M. P. Eastwood, R. O. Dror, and D. E. Shaw, "Systematic validation of protein force fields against experimental data," *PloS one*, vol. 7, no. 2, e32131, 2012.

[22] A. K. Harris, J. R. Meyerson, Y. Matsuoka, *et al.*, "Structure and accessibility of ha trimers on intact 2009 h1n1 pandemic influenza virus to stem region-specific neutralizing antibodies," *Proceedings of the National Academy of Sciences*, vol. 110, no. 12, pp. 4592–4597, 2013.

[23] S. Toxvaerd, "Hamiltonians for discrete dynamics," *Physical Review E*, vol. 50, no. 3, p. 2271, 1994.

[24] K. Vanommeslaeghe, E. Hatcher, C. Acharya, *et al.*, "Charmm general force field: A force field for drug-like molecules compatible with the charmm all-atom additive biological force fields," *Journal of computational chemistry*, vol. 31, no. 4, pp. 671–690, 2010.

[25] E. Paquet, H. L. Viktor, *et al.*, "Molecular dynamics, monte carlo simulations, and langevin dynamics: A computational review," *BioMed research international*, vol. 2015, 2015.

[26] P. Minary, M. E. Tuckerman, and G. J. Martyna, "Dynamical spatial warping: A novel method for the conformational sampling of biophysical structure," *SIAM Journal on Scientific Computing*, vol. 30, no. 4, pp. 2055–2083, 2008.

[27] T. Huber and W. F. van Gunsteren, "Swarm-md: Searching conformational space by cooperative molecular dynamics," *The Journal of Physical Chemistry A*, vol. 102, no. 29, pp. 5937–5943, 1998.

[28] S. Decherchi and A. Cavalli, "Thermodynamics and kinetics of drug-target binding by molecular simulation," *Chemical Reviews*, vol. 120, no. 23, pp. 12 788–12 833, 2020.

[29] D. Baker and A. Sali, "Protein structure prediction and structural genomics," *Science*, vol. 294, no. 5540, pp. 93–96, 2001.

[30] A. Özen, T. Haliloğlu, and C. A. Schiffer, "Dynamics of preferential substrate recognition in hiv-1 protease: Redefining the substrate envelope," *Journal of molecular biology*, vol. 410, no. 4, pp. 726–744, 2011.

[31] S. J. Wright, *Numerical optimization*. 2006.

[32] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[33] M. Sipser, "Introduction to the theory of computation," *ACM Sigact News*, vol. 27, no. 1, pp. 27–29, 1996.

[34] A. M. Turing *et al.*, "On computable numbers, with an application to the entscheidungsproblem," *J. of Math*, vol. 58, no. 345-363, p. 5, 1936.

[35] I. Wegener, *The complexity of Boolean functions*. John Wiley & Sons, Inc., 1987.

[36] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[37] T. M. F. Ramos, A. A. Almeida, and M. Ayala-Rincón, "Formalization of the computational theory of a turing complete functional language model," *Journal of Automated Reasoning*, vol. 66, no. 4, pp. 1031–1063, 2022.

[38] R. P. Feynman *et al.*, "Simulating physics with computers," *Int. j. Theor. phys*, vol. 21, no. 6/7, 2018.

[39] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[40] C. P. Williams, *Explorations in Quantum Computing*. Springer Publishing Company, Incorporated, 2008.

[41] W. Rossmann, *Lie groups: an introduction through linear groups*. Oxford University Press, USA, 2006, vol. 5.

[42] N. Hatano and M. Suzuki, "Finding exponential product formulas of higher orders," in *Quantum annealing and other optimization methods*, Springer, 2005, pp. 37–68.

[43] S. Adrien, "Master's thesis report: Introduction to quantum computing," 2018.

[44] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*, 2002.

[45] E. Rieffel and W. Polak, "Quantum computing," *The Handbook of Technology Management*, vol. 3, 2011.

[46] V. V. Shende and I. L. Markov, "On the cnot-cost of toffoli gates," *arXiv preprint arXiv:0803.2316*, 2008.

[47] I. Q. Team, "User guide - universality of quantum computation," 2019. DOI: https://quantum-computing.ibm.com/support/guides/introduction-to-quantum-circuits?section=5cae61a566c1694be21df8ce.

[48] D. P. DiVincenzo, "The physical implementation of quantum computation," *Fortschritte der Physik: Progress of Physics*, vol. 48, no. 9-11, pp. 771–783, 2000.

[49] A. Bocharov, K. M. Svore, *et al.*, "From reversible logic gates to universal quantum bases," *Bulletin of EATCS*, vol. 2, no. 110, 2013.

[50] C. M. Dawson and M. A. Nielsen, "The solovay-kitaev algorithm," *arXiv preprint quant-ph/0505030*, 2005.

[51] A. Y. Kitaev, "Quantum computations: Algorithms and error correction," *Russian Mathematical Surveys*, vol. 52, no. 6, p. 1191, 1997.

[52] Y. Zhiyenbayev, V. Akulin, and A. Mandilara, "Quantum compiling with diffusive sets of gates," *Physical Review A*, vol. 98, no. 1, p. 012 325, 2018.

[53] A. Barenco, C. H. Bennett, R. Cleve, *et al.*, "Elementary gates for quantum computation," *Physical review A*, vol. 52, no. 5, p. 3457, 1995.

[54] A. W. Harrow, B. Recht, and I. L. Chuang, "Efficient discrete approximations of quantum gates," *Journal of Mathematical Physics*, vol. 43, no. 9, pp. 4445–4451, 2002.

[55] A. Y. Kitaev, A. Shen, and M. N. Vyalyi, *Classical and quantum computation*. American Mathematical Soc., 2002.

[56] S. Lloyd, "Almost any quantum logic gate is universal," *Physical review letters*, vol. 75, no. 2, p. 346, 1995.

[57] D. E. Deutsch, A. Barenco, and A. Ekert, "Universality in quantum computation," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 449, no. 1937, pp. 669–677, 1995.

[58] F. Vatan and C. Williams, "Optimal quantum circuits for general two-qubit gates," *Physical Review A*, vol. 69, no. 3, p. 032 315, 2004.

[59] A. Zulehner and R. Wille, "Compiling su (4) quantum circuits to ibm qx architectures," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ACM, 2019, pp. 185–190.

[60] C. C. McGeoch, *Adiabatic quantum computation and quantum annealing: Theory and practice*. Springer Nature, 2022.

[61] E. Ising, "Beitrag zur theorie des ferromagnetismus," *Zeitschrift für Physik A Hadrons and Nuclei*, vol. 31, no. 1, pp. 253–258, 1925.

[62] D. Sherrington and S. Kirkpatrick, "Solvable model of a spin-glass," *Physical review letters*, vol. 35, no. 26, p. 1792, 1975.

[63] S. Morita and H. Nishimori, "Mathematical foundation of quantum annealing," *Journal of Mathematical Physics*, vol. 49, no. 12, p. 125 210, 2008.

[64] S. Morita and H. Nishimori, "Convergence of quantum annealing with real-time schrödinger dynamics," *Journal of the Physical Society of Japan*, vol. 76, no. 6, p. 064 002, 2007.

[65] S. M. Girvin, "Circuit qed: Superconducting qubits coupled to microwave photons," *Quantum Machines: Measurement and Control of Engineered Quantum Systems*, p. 113, 2011.

[66] A. Irastorza Gabilondo, "Quantum computation with superconductors," 2017.

[67] V. Bouchiat, D. Vion, P. Joyez, D. Esteve, and M. Devoret, "Quantum coherence with a single cooper pair," *Physica Scripta*, vol. 1998, no. T76, p. 165, 1998.

[68] M. Raizen, J. Gilligan, J. C. Bergquist, W. M. Itano, and D. J. Wineland, "Ionic crystals in a linear paul trap," *Physical Review A*, vol. 45, no. 9, p. 6493, 1992.

[69] J. A. Bergou, M. Hillery, and M. Saffman, *Quantum information processing*. Springer, 2021.

[70] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[71] G. Lindblad, "On the generators of quantum dynamical semigroups," *Communications in Mathematical Physics*, vol. 48, pp. 119–130, 1976.

[72] A. R. Ana Neri, "Quantum computation: Ibm q experience," MAPI DOCTORAL PROGRAMME IN COMPUTER SCIENCE, 2018.

[73] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," *Physical Review A*, vol. 100, no. 3, p. 032 328, 2019.

[74] M. Sisodia, A. Shukla, and A. Pathak, "Experimental realization of non-destructive discrimination of bell states using a five-qubit quantum computer," *Physics Letters A*, vol. 381, no. 46, pp. 3860–3874, 2017.

[75] D. Gottesman, *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

[76] S. Ghose, A. Boroumand, J. S. Kim, J. Gómez-Luna, and O. Mutlu, "Processing-in-memory: A workload-driven perspective," *IBM Journal of Research and Development*, vol. 63, no. 6, pp. 3–1, 2019.

[77] L. O. Chua and G. Lin, "Non-linear optimization with constraints: A cook-book approach," *International Journal of Circuit Theory and Applications*, vol. 11, no. 2, pp. 141–159, 1983.

[78] K. Tatsumura, M. Yamasaki, and H. Goto, "Scaling out ising machines using a multi-chip architecture for simulated bifurcation," *Nature Electronics*, vol. 4, no. 3, pp. 208–217, 2021.

[79] F. Rothganger, C. D. James, and J. B. Aimone, "Computing with dynamical systems," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, IEEE, 2016, pp. 1–3.

[80] L. Perko, *Differential equations and dynamical systems*. Springer Science & Business Media, 2013, vol. 7.

[81] D. V. Arnold and H.-G. Beyer, "Performance analysis of evolution strategies with multi-recombination in high-dimensional rn-search spaces disturbed by noise," *Theoretical Computer Science*, vol. 289, no. 1, pp. 629–647, 2002.

[82] S. Wiggins *et al.*, "Introduction to applied nonlinear dynamical systems and chaos [electronic resource],"

[83] A. M. Liapunov, *Stability of motion*. Elsevier, 2016.

[84] W. Rautenberg, *A concise introduction to mathematical logic*. Springer, 2006, vol. 39.

[85] M. Di Ventra, *MemComputing: fundamentals and applications*. Oxford University Press, 2022.

[86] F. L. Traversa and M. Di Ventra, "Polynomial-time solution of prime factorization and np-complete problems with digital memcomputing machines," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 2, 2017.

[87] Y. V. Pershin and M. Di Ventra, "Memory effects in complex materials and nanoscale systems," *Advances in Physics*, vol. 60, no. 2, pp. 145–227, 2011.

[88] M. Di Ventra and F. L. Traversa, *Self-organizing logic gates and circuits and complex problem solving with self-organizing circuits*, US Patent 9,911,080, Mar. 2018.

[89] S. Arora and B. Barak, *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[90] N. Goldenfeld, *Lectures on phase transitions and the renormalization group*. CRC Press, 2018.

[91] C. G. Langton, "Computation at the edge of chaos: Phase transitions and emergent computation," *Physica D: nonlinear phenomena*, vol. 42, no. 1-3, pp. 12–37, 1990.

[92] M. A. Munoz, "Colloquium: Criticality and dynamical scaling in living systems," *Reviews of Modern Physics*, vol. 90, no. 3, p. 031 001, 2018.

[93] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified np-complete problems," in *Proceedings of the sixth annual ACM symposium on Theory of computing*, 1974, pp. 47–63.

[94] F. L. Traversa and M. Di Ventra, "Memcomputing integer linear programming," *arXiv preprint arXiv:1808.09999*, 2018.

[95] H. M. Cezar, S. Canuto, and K. Coutinho, "Dice: A monte carlo code for molecular simulation including the configurational bias monte carlo method," *Journal of Chemical Information and Modeling*, vol. 60, no. 7, pp. 3472–3488, 2020.

[96] G. Fishman, *Monte Carlo: concepts, algorithms, and applications*. Springer Science & Business Media, 2013.

[97] P. Glasserman, *Monte Carlo methods in financial engineering*. Springer, 2004, vol. 53.

[98] E. Ferraro and E. Prati, "Is all-electrical silicon quantum computing feasible in the long term?" *Physics Letters A*, p. 126 352, 2020.

[99] M. De Michielis, E. Ferraro, E. Prati, *et al.*, "Silicon spin qubits from laboratory to industry," *Journal of Physics D: Applied Physics*, vol. 56, no. 36, p. 363 001, 2023.

[100] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *Contemporary Mathematics*, vol. 305, pp. 53–74, 2002.

[101] S. Herbert, "Quantum monte carlo integration: The full advantage in minimal circuit depth," *Quantum*, vol. 6, p. 823, 2022.

[102] P. Rebentrost, B. Gupt, and T. R. Bromley, "Quantum computational finance: Monte carlo pricing of financial derivatives," *Physical Review A*, vol. 98, no. 2, p. 022 321, 2018.

[103] A. Montanaro, "Quantum speedup of monte carlo methods," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 471, no. 2181, p. 20 150 301, 2015.

[104] G. Agliardi, M. Grossi, M. Pellen, and E. Prati, "Quantum integration of elementary particle processes," *Physics Letters B*, vol. 832, p. 137 228, 2022.

[105] J. Allcock and S. Zhang, "Quantum machine learning," *National Science Review*, vol. 6, no. 1, pp. 26–28, 2019.

[106] M. Maronese, L. Moro, L. Rocutto, and E. Prati, "Quantum compiling," in *Quantum Computing Environments*, Springer, 2022, pp. 39–74.

[107] M. Lazzarin, D. E. Galli, and E. Prati, "Multi-class quantum classifiers with tensor network circuits for quantum phase recognition," *Physics Letters A*, vol. 434, p. 128 056, 2022.

[108] R. Molteni, C. Destri, and E. Prati, "Optimization of the memory reset rate of a quantum echo-state network for time sequential tasks," *Physics Letters A*, vol. 465, p. 128 713, 2023.

[109] G. Agliardi and E. Prati, "Optimal tuning of quantum generative adversarial networks for multivariate distribution loading," *Quantum Reports*, vol. 4, no. 1, pp. 75–105, 2022.

[110] N. Stamatopoulos, D. J. Egger, Y. Sun, *et al.*, "Option pricing using quantum computers," *Quantum*, vol. 4, p. 291, 2020.

[111] S. Chakrabarti, R. Krishnakumar, G. Mazzola, N. Stamatopoulos, S. Woerner, and W. J. Zeng, "A threshold for quantum advantage in derivative pricing," *Quantum*, vol. 5, p. 463, 2021.

[112] F. Oz, R. K. Vuppala, K. Kara, and F. Gaitan, "Solving burgers' equation with quantum computing," *Quantum Information Processing*, vol. 21, pp. 1–13, 2022.

[113] K. Binder, *Monte Carlo and molecular dynamics simulations in polymer science*. Oxford University Press, 1995.

[114] J. Preskill, "Fault-tolerant quantum computation," in *Introduction to quantum computation and information*, World Scientific, 1998, pp. 213–269.

[115] E. Prati, "Quantum neuromorphic hardware for quantum artificial intelligence," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 880, 2017, p. 012 018.

[116] S. Woerner and D. J. Egger, "Quantum risk analysis," *npj Quantum Information*, vol. 5, no. 1, pp. 1–8, 2019.

[117] M. Maronese, L. Moro, L. Rocutto, and E. Prati, "Quantum compiling," in *Quantum Computing Environments*, Springer, 2022, pp. 39–74.

[118] Y. Suzuki, S. Uno, R. Raymond, T. Tanaka, T. Onodera, and N. Yamamoto, "Amplitude estimation without phase estimation," *Quantum Information Processing*, vol. 19, no. 2, pp. 1–17, 2020.

[119] D. Grinko, J. Gacon, C. Zoufal, and S. Woerner, "Iterative quantum amplitude estimation," *npj Quantum Information*, vol. 7, no. 1, pp. 1–6, 2021.

[120] K. Wright, K. M. Beck, S. Debnath, *et al.*, "Benchmarking an 11-qubit quantum computer," *Nature communications*, vol. 10, no. 1, p. 5464, 2019.

[121] Y. Nam, J.-S. Chen, N. C. Pisenti, *et al.*, "Ground-state energy estimation of the water molecule on a trapped-ion quantum computer," *npj Quantum Information*, vol. 6, no. 1, p. 33, 2020.

[122] P. Dagum, R. Karp, M. Luby, and S. Ross, "An optimal algorithm for monte carlo estimation," *SIAM Journal on computing*, vol. 29, no. 5, pp. 1484–1496, 2000.

[123] M. Huber, "Improving monte carlo randomized approximation schemes," *arXiv preprint arXiv:1411.4074*, 2014.

[124] L. Grover and T. Rudolph, "Creating superpositions that correspond to efficiently integrable probability distributions," *arXiv preprint quant-ph/0208112*, 2002.

[125] V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum logic circuits," in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, 2005, pp. 272–275.

[126] T. Häner, M. Roetteler, and K. M. Svore, "Optimizing quantum circuits for arithmetic," *arXiv preprint arXiv:1805.12445*, 2018.

[127] A. C. Vazquez, R. Hiptmair, and S. Woerner, "Enhancing the quantum linear systems algorithm using richardson extrapolation," *ACM Transactions on Quantum Computing*, vol. 3, no. 1, pp. 1–37, 2022.

[128] T. Giurgica-Tiron, I. Kerenidis, F. Labib, A. Prakash, and W. Zeng, "Low depth algorithms for quantum amplitude estimation," *arXiv preprint arXiv:2012.03348*, 2020.

[129] K. Plekhanov, M. Rosenkranz, M. Fiorentini, and M. Lubasch, "Variational quantum amplitude estimation," *arXiv preprint arXiv:2109.03687*, 2021.

[130] S. Aaronson and P. Rall, "Quantum approximate counting, simplified," in *Symposium on Simplicity in Algorithms*, SIAM, 2020, pp. 24–32.

[131] R. J. Rossi, *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons, 2018.

[132] K. Nakaji, "Faster amplitude estimation," *arXiv preprint arXiv:2003.02417*, 2020.

[133] S. Certo, A. D. Pham, and D. Beaulieu, "Benchmarking amplitude estimation on a superconducting quantum computer," *arXiv preprint arXiv:2201.06987*, 2022.

[134] Amazon, "Amazon braket," Amazon Web Service, Tech. Rep., 2020. [Online]. Available: https://aws.amazon.com/braket/.

[135] M. Brooks, "Beyond quantum supremacy: The hunt for useful quantum computers," *Nature*, vol. 574, no. 7776, pp. 19–22, 2019.

[136] F. Arute, K. Arya, R. Babbush, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[137] Y. Wu, W.-S. Bao, S. Cao, *et al.*, "Strong quantum computational advantage using a superconducting quantum processor," *Physical review letters*, vol. 127, no. 18, p. 180 501, 2021.

[138] A. Peruzzo, J. McClean, P. Shadbolt, *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, p. 4213, 2014.

[139] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023 023, 2016.

[140] J. MacDonald, "Successive approximations by the rayleigh-ritz variation method," *Physical Review*, vol. 43, no. 10, p. 830, 1933.

[141] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, "Connecting ansatz expressibility to gradient magnitudes and barren plateaus," *PRX Quantum*, vol. 3, no. 1, p. 010 313, 2022.

[142] G. E. Crooks, "Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition," *arXiv preprint arXiv:1905.13311*, 2019.

[143] T. Haug, K. Bharti, and M. Kim, "Capacity and quantum geometry of parametrized quantum circuits," *PRX Quantum*, vol. 2, no. 4, p. 040 309, 2021.

[144] B. Koczor and S. C. Benjamin, "Quantum natural gradient generalized to noisy and nonunitary circuits," *Physical Review A*, vol. 106, no. 6, p. 062 416, 2022.

[145] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem," *Science*, vol. 292, no. 5516, pp. 472–475, 2001.

[146] L. Veis and J. Pittner, "Adiabatic state preparation study of methylene," *The Journal of Chemical Physics*, vol. 140, no. 21, 2014.

[147] J. Roland and N. J. Cerf, "Quantum search by local adiabatic evolution," *Physical Review A*, vol. 65, no. 4, p. 042 308, 2002.

[148] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[149] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Physical Review X*, vol. 10, no. 2, p. 021 067, 2020.

[150] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, "Improving variational quantum optimization using cvar," *Quantum*, vol. 4, p. 256, 2020.

[151] S. Dutta, Y. Lee, and Y. Jho, "Hydration of ions in two-dimensional water," *Physical Review E*, vol. 92, no. 4, p. 042 152, 2015.

[152] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, https://www.gurobi.com/documentation/current/refman/index.html, Accessed: 10-Jul-2023, 2023.

[153] M. Gitterman, "Mean first passage time for anomalous diffusion," *Phys. Rev. E*, vol. 62, no. 5, p. 6065, 2000.

[154] M. Powell, "Advances in optimization and numerical analysis," in *Proceeding of the 6th Workshop on Optimization and Numerical Analysis*, 1994, pp. 5–67.

[155] F. G. Fuchs, K. O. Lye, H. Møll Nilsen, A. J. Stasik, and G. Sartor, "Constraint preserving mixers for the quantum approximate optimization algorithm," *Algorithms*, vol. 15, no. 6, p. 202, 2022.

[156] J. Schmidhuber, "Colossus was the first electronic digital computer," *Nature*, vol. 441, no. 7089, pp. 25–25, 2006.

[157] R. Herken, *The Universal Turing Machine A Half-Century Survey*. Springer-Verlag, 1995.

[158] L. F. Menabrea, "Sketch of the Analytical Engine (1843) with notes by the translator, Ada Agusta, Countess of Lovelace," in https://doi.org/10.7551/mitpress/12274.003.0005, 2021, ch. 3, pp. 9–26.

[159] S. E. Thompson and S. Parthasarathy, "Moore's law: The future of si microelectronics," *Mater. today*, vol. 9, no. 6, pp. 20–25, 2006.

[160] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proc. Math. Phys. Eng. Sci.*, vol. 400, no. 1818, pp. 97–117, 1985.

[161] E. Bernstein and U. Vazirani, "Quantum complexity theory," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1411–1473, 1997.

[162] Xanadu Quantum Technologies, *\NoCaseChange{https://www.xanadu.ai/hardware}*, 2021.

[163] IonQ, *Ionq – trapped ion quantum computing*, https://ionq.com/, 2021.

[164] Honeywell Quantum Solutions, *\NoCaseChange{https://www.honeywell.com/}*, 2021.

[165] Google Quantum AI, *\NoCaseChange{https://quantumai.google/hardware}*, 2021.

[166] IBM Quantum, *Https://quantum-computing.ibm.com/*, 2021.

[167] Rigetti Computing, *\NoCaseChange{https://www.rigetti.com/what-we-build}*, 2021.

[168] E. Gibney, "Quantum gold rush: The private funding pouring into quantum start-ups," *Nature*, vol. 574, no. 7776, pp. 22–24, Oct. 2019. DOI: 10.1038/d41586-019-02935-4. [Online]. Available: https://doi.org/10.1038/d41586-019-02935-4.

[169] E. Pednault, J. Gunnels, D. Maslov, and J. Gambetta, *\NoCaseChange{https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/}*, 2019.

[170] Y. Liu, X. Liu, F. Li, *et al.*, "Closing the" quantum supremacy" gap: Achieving real-time simulation of a random quantum circuit using a new sunway supercomputer," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–12.

[171] F. Pan, K. Chen, and P. Zhang, "Solving the sampling problem of the sycamore quantum circuits," *Phys. Rev. Lett.*, vol. 129, p. 090 502, 9 Aug. 2022. DOI: 10.1103/PhysRevLett.129.090502. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.129.090502.

[172] M. W. Johnson, M. H. Amin, S. Gildert, *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, p. 194, 2011.

[173] F. L. Traversa and M. Di Ventra, "Universal memcomputing machines," *IEEE Trans. Neural. Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2702–2715, 2015.

[174] N. Mohseni, P. L. McMahon, and T. Byrnes, "Ising machines as hardware solvers of combinatorial optimization problems," *Nat. Rev. Phys.*, vol. 4, no. 6, pp. 363–379, May 2022. DOI: 10.1038/s42254-022-00440-8. [Online]. Available: https://doi.org/10.1038/s42254-022-00440-8.

[175] K. Y. Camsari, B. M. Sutton, and S. Datta, "P-bits for probabilistic spin logic," *Appl. Phys. Rev.*, vol. 6, no. 1, p. 011 305, 2019. DOI: 10.1063/1.5055860. eprint: https://doi.org/10.1063/1.5055860. [Online]. Available: https://doi.org/10.1063/1.5055860.

[176] M. Kowalsky, T. Albash, I. Hen, and D. A. Lidar, "3-regular three-xorsat planted solutions benchmark of classical and quantum heuristic optimizers," *Quantum Sci. Technol.*, vol. 7, no. 2, p. 025 008, Feb. 2022. DOI: 10.1088/2058-9565/ac4d1b. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/ac4d1b.

[177] T. F. Rønnow, Z. Wang, J. Job, *et al.*, "Defining and detecting quantum speedup," *science*, vol. 345, no. 6195, pp. 420–424, 2014.

[178] E. Knill, R. Laflamme, R. Martinez, and C.-H. Tseng, "An algorithmic benchmark for quantum information processing," *Nature*, vol. 404, no. 6776, pp. 368–370, 2000.

[179] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual, https://www.gurobi.com*, 2021.

[180] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, ISSN: 0001-0782. DOI: 10.1145/359340.359342. [Online]. Available: https://doi.org/10.1145/359340.359342.

[181] F. Mémoli, "Gromov–wasserstein distances and the metric approach to object matching," *Found. Comput. Math.*, vol. 11, no. 4, pp. 417–487, 2011.

[182] G. Peyré, M. Cuturi, and J. Solomon, "Gromov-wasserstein averaging of kernel and distance matrices," in *International Conference on Machine Learning*, PMLR, 2016, pp. 2664–2672.

[183] E. L. Lawler, "The quadratic assignment problem," *Manage. Sci.*, vol. 9, no. 4, pp. 586–599, 1963.

[184] A. Perdomo-Ortiz, A. Feldman, A. Ozaeta, *et al.*, "Readiness of quantum optimization machines for industrial applications," *Phys. Rev. Appl.*, vol. 12, no. 1, p. 014 004, 2019.

[185] I. Hen, J. Job, T. Albash, T. F. Rønnow, M. Troyer, and D. A. Lidar, "Probing for quantum speedup in spin-glass problems with planted solutions," *Phys. Rev. A*, vol. 92, no. 4, p. 042 325, 2015.

[186] T. Albash and D. A. Lidar, "Demonstration of a scaling advantage for a quantum annealer over simulated annealing," *Physical Review X*, vol. 8, no. 3, p. 031 016, 2018.

[187] S. G. Krantz, *The proof is in the pudding: The changing nature of mathematical proof*. Springer, 2011.

[188] A. K. Lenstra, H. W. Lenstra, M. S. Manasse, and J. M. Pollard, "The number field sieve," in *The development of the number field sieve*, Springer, 1993, pp. 11–42.

[189] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, Ieee, 1994, pp. 124–134.

[190] C. J. Burges, "Factoring as optimization," *Microsoft Research MSR-TR-200*, 2002.

[191] R. Dridi and H. Alghassi, "Prime factorization using quantum annealing and computational algebraic geometry," *Scientific reports*, vol. 7, no. 1, pp. 1–10, 2017.

[192] S. Jiang, K. A. Britt, A. J. McCaskey, T. S. Humble, and S. Kais, "Quantum annealing for prime factorization," *Scientific reports*, vol. 8, no. 1, p. 17 667, 2018.

[193] R. Mengoni, D. Ottaviani, and P. Iorio, "Breaking rsa security with a low noise d-wave 2000q quantum annealer: Computational times, limitations and prospects," *arXiv preprint arXiv:2005.02268*, 2020.

[194] L. Ambrosio, L. A. Caffarelli, Y. Brenier, G. Buttazzo, C. Villani, and S. Salsa, *Optimal Transportation and Applications*. Springer Berlin Heidelberg, 2003. DOI: 10.1007/b12016. [Online]. Available: https://doi.org/10.1007/b12016.

[195] F. Mémoli, "Spectral gromov-wasserstein distances for shape matching," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, IEEE, 2009, pp. 256–263.

[196] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, PMLR, 2017, pp. 214–223.

[197] C. Bunne, D. Alvarez-Melis, A. Krause, and S. Jegelka, "Learning generative models across incomparable spaces," in *International Conference on Machine Learning*, PMLR, 2019, pp. 851–861.

[198] J. Solomon, G. Peyré, V. G. Kim, and S. Sra, "Entropic metric alignment for correspondence problems," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 4, pp. 1–13, 2016.

[199] M. Scetbon, G. Peyré, and M. Cuturi, "Linear-time gromov wasserstein distances using low rank couplings and costs," in *International Conference on Machine Learning*, PMLR, 2022, pp. 19 347–19 365.

[200] T. Vayer, R. Flamary, R. Tavenard, L. Chapel, and N. Courty, "Sliced gromov-wasserstein," in *Neural Information Processing Systems*, 2019.

[201] R. E. Burkard, E. Cela, P. M. Pardalos, and L. S. Pitsoulis, "The quadratic assignment problem," in *Handbook of combinatorial optimization*, Springer, 1998, pp. 1713–1809.

[202] M. T. Fiala Timlin and W. R. Pulleyblank, "Precedence constrained routing and helicopter scheduling: Heuristic design," *Interfaces*, vol. 22, no. 3, pp. 100–111, 1992.

[203] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem (darp): Variants, modeling issues and algorithms," *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, pp. 89–101, 2003.

[204] R. W. Calvo and A. Colorni, "An effective and fast heuristic for the dial-a-ride problem," *4or*, vol. 5, no. 1, pp. 61–73, 2007.

[205] M. W. Savelsbergh, "Local search in routing problems with time windows," *Annals of Operations research*, vol. 4, no. 1, pp. 285–305, 1985.

[206] R. de Alvarenga Rosa, A. Manhães Machado, G. Mattos Ribeiro, and G. Regis Mauri, "A mathematical model and a clustering search metaheuristic for planning the helicopter transportation of employees to the production platforms of oil and gas," *Comput. Ind. Eng.*, vol. 101, pp. 303–312, 2016, ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2016.09.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835216303382.

[207] A. Motta, R. Vieira, and J. Soletti, "Optimal routing offshore helicopter using genetic algorithm," in *2011 6th IEEE Joint International Information Technology and Artificial Intelligence Conference*, vol. 2, 2011, pp. 6–9.

[208] A. Husseinzadeh Kashan, A. Abbasi-Pooya, and S. Karimiyan, "A rig-based formulation and a league championship algorithm for helicopter routing in offshore transportation," in *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*, A. J. Kulkarni, S. C. Satapathy, T. Kang, and A. H. Kashan, Eds., Springer Singapore, 2019, pp. 23–38.

[209] M. Maronese, L. Moro, L. Rocutto, and E. Prati, "Quantum compiling," *arXiv preprint arXiv:2112.00187*, 2021.

[210] L. Mitten, "Branch-and-bound methods: General formulation and properties," *Oper. Res.*, vol. 18, no. 1, pp. 24–34, 1970.

[211] C. Kaiser, J. Meckbach, K. Ilin, *et al.*, "Aluminum hard mask technique for the fabrication of high quality submicron nb/al–alox/nb josephson junctions," *Superconductor science and technology*, vol. 24, no. 3, p. 035 005, 2010.

[212] R. Harris, J. Johansson, A. Berkley, *et al.*, "Experimental demonstration of a robust and scalable flux qubit," *Physical Review B*, vol. 81, no. 13, p. 134 510, 2010.

[213] S. Boixo, T. Albash, F. M. Spedalieri, N. Chancellor, and D. A. Lidar, "Experimental signature of programmable quantum annealing," *Nature communications*, vol. 4, p. 2067, 2013.

[214] T. Albash and D. A. Lidar, "Decoherence in adiabatic quantum computation," *Physical Review A*, vol. 91, no. 6, p. 062 320, 2015.

[215] L. Buffoni and M. Campisi, "Thermodynamics of a quantum annealer," *Quantum Science and Technology*, vol. 5, no. 3, p. 035 013, 2020.

[216] H. G. Katzgraber, F. Hamze, and R. S. Andrist, "Glassy chimeras could be blind to quantum speedup: Designing better benchmarks for quantum annealing machines," *Phys. Rev. X*, vol. 4, no. 2, p. 021 008, 2014.

[217] H. G. Katzgraber, F. Hamze, Z. Zhu, A. J. Ochoa, and H. Munoz-Bauza, "Seeking quantum speedup through spin glasses: The good, the bad, and the ugly," *Phys. Rev. X*, vol. 5, no. 3, p. 031 026, 2015.

[218] A. Callison and N. Chancellor, "Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond," *Phys. Rev. A*, vol. 106, no. 1, Jul. 2022. DOI: 10.1103/physreva.106.010101. [Online]. Available: https://doi.org/10.1103%5C%2Fphysreva.106.010101.

[219] Y. Kim, A. Eddins, S. Anand, *et al.*, "Evidence for the utility of quantum computing before fault tolerance," *Nature*, vol. 618, no. 7965, pp. 500–505, 2023.

[220] J. Tindall, M. Fishman, M. Stoudenmire, and D. Sels, "Efficient tensor network simulation of ibm's kicked ising experiment," *arXiv preprint arXiv:2306.14887*, 2023.

[221] A. D. King, J. Raymond, T. Lanting, *et al.*, "Quantum critical dynamics in a 5,000-qubit programmable spin glass," *Nature*, pp. 1–6, 2023.

[222] V. Dunjko, J. M. Taylor, and H. J. Briegel, "Quantum-enhanced machine learning," *Physical review letters*, vol. 117, no. 13, p. 130 501, 2016.

[223] L. Moro and E. Prati, "Anomaly detection speed-up by quantum restricted boltzmann machines," *Communications Physics*, vol. 6, no. 1, p. 269, 2023.

[224] W. K. Mutlag, S. K. Ali, Z. M. Aydam, and B. H. Taher, "Feature extraction methods: A review," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1591, 2020, p. 012 028.

[225] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction," *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 56–70, 2020.

[226] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, IEEE Computer Society, 1991, pp. 586–587.

[227] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[228] R. Bro and A. K. Smilde, "Principal component analysis," *Analytical methods*, vol. 6, no. 9, pp. 2812–2831, 2014.

[229] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," *arXiv preprint arXiv:1403.2877*, 2014.

[230] A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural networks*, vol. 13, no. 4-5, pp. 411–430, 2000.

[231] O. Déniz, M. Castrillon, and M. Hernández, "Face recognition using independent component analysis and support vector machines," *Pattern recognition letters*, vol. 24, no. 13, pp. 2153–2157, 2003.

[232] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[233] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, Springer, 2014, pp. 818–833.

[234] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[235] K. Mato, R. Mengoni, D. Ottaviani, and G. Palermo, "Quantum molecular unfolding," *Quantum Science and Technology*, vol. 7, no. 3, p. 035 020, 2022.

[236] C. F. Negre, H. Ushijima-Mwesigwa, and S. M. Mniszewski, "Detecting multiple communities using quantum annealing on the d-wave system," *Plos one*, vol. 15, no. 2, e0227538, 2020.

[237] D-Wave Systems Inc., *Qpu-specific physical properties: Advantage_ system4.1*, 2022.

[238] D. O'Malley, V. V. Vesselinov, B. S. Alexandrov, and L. B. Alexandrov, "Nonnegative/binary matrix factorization with a d-wave quantum annealer," *PloS one*, vol. 13, no. 12, e0206653, 2018.

[239] J. Golden and D. O'Malley, "Reverse annealing for nonnegative/binary matrix factorization," *Plos one*, vol. 16, no. 1, e0244026, 2021.

[240] R. Y. Li, R. Di Felice, R. Rohs, and D. A. Lidar, "Quantum annealing versus classical machine learning applied to a simplified computational biology problem," *NPJ quantum information*, vol. 4, no. 1, p. 14, 2018.

[241] D. Ottaviani and A. Amendola, "Low rank non-negative matrix factorization with d-wave 2000q," *arXiv preprint arXiv:1808.08721*, 2018.

[242] L. Rocutto and E. Prati, "A complete restricted boltzmann machine on an adiabatic quantum computer," *International Journal of Quantum Information*, vol. 19, no. 04, p. 2 141 003, 2021.

[243] L. Rocutto, C. Destri, and E. Prati, "Quantum semantic learning by reverse annealing of an adiabatic quantum computer," *Advanced Quantum Technologies*, vol. 4, no. 2, p. 2 000 133, 2021.

[244] D. Vert, R. Sirdey, and S. Louise, "On the limitations of the chimera graph topology in using analog quantum computers," in *Proceedings of the 16th ACM international conference on computing frontiers*, 2019, pp. 226–229.

[245] K. Boothby, P. Bunyk, J. Raymond, and A. Roy, "Next-generation topology of d-wave quantum processors," *arXiv preprint arXiv:2003.00133*, 2020.

[246] D-Wave Systems Inc., *Qpu-specific physical properties: Advantage2_prototype1.1*, 2022.

[247] M. Maronese, L. Moro, L. Rocutto, and E. Prati, "Quantum compiling," in *Quantum Computing Environments*, Springer, 2022, pp. 39–74.

[248] J. Cai, W. G. Macready, and A. Roy, "A practical heuristic for finding graph minors," *arXiv preprint arXiv:1406.2741*, 2014.

[249] V. Choi, "Minor-embedding in adiabatic quantum computation: I. the parameter setting problem," *Quantum Information Processing*, vol. 7, pp. 193–209, 2008.

[250] D-Wave Systems, *Ocean SDK public Github repository*, https://github.com/dwavesystems/dwave-ocean-sdk, Accessed: 27-Sep-2023, 2023.

[251] L. Rocutto, M. Maronese, F. Traversa, S. Decherchi, and A. Cavalli, "Assessing the effectiveness of non-turing computing paradigms," *submitted*, 2023.

[252] *D-Wave github repository - `uniform_torque_compensation` https://github.com/dwavesystems/dwave-system/blob/1.18.0/dwave/embedding/chain_strength.py#L38*, 2021.

[253] V. Dumoulin, I. J. Goodfellow, A. Courville, and Y. Bengio, "On the challenges of physical implementations of rbms," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[254] J. Clark, T. West, J. Zammit, X. Guo, L. Mason, and D. Russell, "Towards real time multi-robot routing using quantum computing technologies," in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, 2019, pp. 111–119.

[255] E. Zardini, M. Rizzoli, S. Dissegna, E. Blanzieri, and D. Pastorello, "Reconstructing bayesian networks on a quantum annealer," *arXiv preprint arXiv:2204.03526*, 2022.