ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Quantum Artificial Intelligence

QAL-BP: AN AUGMENTED LAGRANGIAN QUANTUM APPROACH FOR THE BIN PACKING PROBLEM

CANDIDATE
Lorenzo Cellini

SUPERVISOR
Prof. Michele Lombardi

CO-SUPERVISOR
Antonio Macaluso, PhD.

Academic year 2021-2022
Session 3rd

Contents

List of Figures

Abstract

Bin packing is one of the most studied NP-Hard problem in combinatorial optimization, but it still remains one of the hardest to solve. On the other hand, emerging quantum technologies have proven to have the potential to provide exponential speedup over classical computing, for certain classes of problems, including combinatorial optimization problems. In this work we present a novel heuristic QUBO formulation for the well known bin packing problem (BPP) based on the augmented Lagrangian method and an analytical estimation of penalty multipliers that makes the model more generalizable to different input instances. We tested our approach on a set of bin packing instances, by using a real quantum computer based on Quantum Annealing (QA). The experiments demonstrate that the proposed approach is capable of finding good quality solutions for small-sized instances of the BPP. The proposed approach is also compared with existing classical state-of-the-art algorithms for the BPP, and we show that it strongly outperforms them in terms of running time. This new approach paves the way to the research for a generalizable QUBO formulation for BPP and to the study of new QUBO formulations for other combinatorial problems with inequality constraints.

1 Introduction

Bin packing is a well-known combinatorial optimization problem with a wide range of applications in various fields, such as logistics, resource allocation, scheduling and so on. Given a set of items of different sizes and a set of bins with a fixed capacity, the objective is to minimize the number of bins required to pack all the items.

Although bin packing is one of the most studied NP-Hard problem in combinatorial optimization, it still remains one of the hardest to solve. On the other hand, emerging quantum technology have proven to have the potential to provide exponential speedup over classical computing for certain classes of problems, including combinatorial optimization problems.

Among various quantum computing paradigms, two prominent approaches are Quantum Annealing (QA) and Quantum Approximate Optimization Algorithm (QAOA). Both approaches have been shown to provide promising results for solving combinatorial optimization problems, but they have different strengths and weaknesses. A key difference between QA and QAOA is that with QAOA you can increase the precision arbitrarily, whereas QA will only find the solution with probability 1 as $\mathbf{T} \to \infty$.

Another difference is the type of quantum hardware that each approach uses. QA uses a quantum annealer, which is a specialized quantum computer designed to solve optimization problems. In contrast, QAOA uses a universal quantum computer, which can be built using various quantum hardware platforms.

Furthermore, as per today's technologies, QA can handle bigger problems, in terms of the number of variables.

Overall, both QA and QAOA have shown promising results for solving combinatorial optimization problems. The choice between these approaches depends on the size of the problem being solved and the available quantum hardware.

In this thesis, we propose a novel heuristic QUBO formulation, based on the augmented Lagrangian method, for some classes of instances of bin packing, that is specifically designed to be solved by quantum annealing. Our model overcomes the limitations of existing formulations and can solve larger problem instances than any previous quantum based approach.

One of the challenges in solving bin packing using quantum annealing is the need to map the problem onto a quadratic unconstrained binary optimization (QUBO) formulation that can be solved by the quantum annealer. To date, only two QUBO formulations for bin packing have been proposed, but, as it will be discussed in chapter 4, one of them suffers from a limitation in the size of the problem instance, due to the use of slack variables to model the inequality constraint; while, the other one, has been derived following a bottom-up reasoning and no evidences of its scalability, with respect the input instance size, are given.

Moreover, we propose an analytical, heuristic, estimation of penalty multipliers, in contrast to the use of a recursive method, such as Alternating Direction Method of Multipliers (ADMM), as it is usually done when dealing with augmented Lagrangian.

In order to demonstrate the effectiveness of our proposed method, we tested it on a real quantum annealer from D-Wave Inc. . The experiments prove that the proposed approach is capable of finding good quality solutions for small-sized instances of the BPP. We also compare our model with existing classical state-of-the-art algorithms for the BPP, and we show that it strongly outperforms them in terms of running time.

In summary, the goal of this thesis is to propose a novel heuristic QUBO formulation based on the augmented Lagrangian method for bin packing and to demonstrate its effectiveness in solving "small" problem instances using a real quantum annealer.

The contributions of this work can lead to new insights into the application of quantum computing for combinatorial optimization problems and provide a foundation for future research in this area. This new approach paves the way to new QUBO formulations for other combinatorial problems with inequality constraints, without the need to approximate penalty multipliers by a recursive method.

The structure of this thesis is the following: in chapter 2 a basic introduction to quantum mechanics, necessary to understand how quantum annealing works, is given; in chapter 3 a survey of the classical algorithms presented in literature, to solve the bin packing problem, is presented; in chapter 4 we present our novel heuristic QUBO formulation for the bin packing problem based on the augmented Lagrangian method, and we compare it with the other proposed formulations currently available in literature; in chapter 5 we show experimental results of our model, when solved by a real quantum computer, together with a comparison with the classical state-of-the-art algorithms.

2 Background theory for Quantum Computing

The exchange of ideas between physics and computer science has consistently been beneficial to both fields. For example solid-state physics fostered the development of semiconductor-based technologies. Similarly, mathematical models inspired by physical phenomena helped in designing advanced algorithms in several fields, such as operation research.

Quantum computation is a scientific and engineering field focused on developing information processing devices and algorithms that exploit the principles of quantum mechanics and its extraordinary phenomena. It is a combination of physics, mathematics, computer science and information theory. It offers tremendous computational power and exponential speed compared to traditional computers by manipulating the quantum state of microscopic objects such as atoms, electrons and photons.

Besides to further advancing the theoretical and experimental foundations of this discipline, new fields like quantum machine learning ([65], [8]) and quantum image processing ([76]) are emerging. Quantum computing and the broader field of quantum technologies (embracing computing, communications, cryptography and sensing) are becoming an attractive emerging branch of high-tech business.

Because of the difference in progression speed between theoretical and experimental quantum computing, the vast majority of quantum algorithms have been designed but not implemented on quantum hardware.

A central goal in quantum computing is the development of quantum algorithms and quantum hardwares, on which quantum algorithms can be tested, in order to analyse challenging scientific and engineering problems.

In this chapter, theorethical concepts from both computer science and physics are presented, in order to give a definitoin of "hard" computational problems, and to understand how emerging quantum technologies can help in solving them.

2.1 Theory of Computation

2.1.1 Computational complexity

In computer science, computational complexity of an algorithm refers to the amount of resources, in terms of time and memory storage, reqiured to run it. Computational complexity focuses on classifying problems in complexity classes and estimating the number of elementary steps required to run an algorithm. There are many different complexity classes ([1] comprises hundreds of them), but the most important, and the ones of nterest here, are the P, NP, NP-complete and NP-hard classes ([52], [31], [34], [3]).

Time complexity of an algorithm A is measured through *asymptotic analysis*, that estimates, for each input length, the largest amount of time needed by A to solve any problem instance of that size (that is the worst-case over all inputs of size n). Because of this, time complexity is referred to using *big O* notation, defined as follow:

Definition 2.1.1 (Big O notation)**.** Let $f, g : \mathbb{N} \mapsto \mathbb{N}$. It is said that $f(n) = O(g(n))$ iff $\exists\, \alpha, n_0 \in \mathbb{N}$ such that $\forall n \geq n_0 \implies f(n) \leq \alpha g(n)$. Then, $g(n)$ is an asymptotic upper bound for $f(n)$ and f is said to be of the order of g. Informally, $f(n) = O(g(n))$ means that f grows as g or slower.

Depending on $g(n)$, an algorithm can have *polynomial-time*, *exponential-time* or *factorial-time* complexity. Usually, polinomially bounded algorithms are also called *tractables*, meaning that they are solvable in a reasonable amount of time (for a human being), in contrast to exponential and factorial time complexity algorithms that are referred to as *intractables*.

2.1.2 Complexity classes

The two most important complexity classes in the computatinal theory are **P** and **NP**, where:

- a problem is in **P** if it is solvable in polynomial time by a DTM (Deterministic Touring Machine);

- a problem is in **NP** if a candidate solution (usually called *certificate*) can be verified (accepted or rejected) in polynomial time, or equivalently, if it can be solved in polinomial time by a NTM (Non-deterministic Touring Machine) [34].

For problems in NP the following theorem holds:

Theorem 2.1.1. If decision problem $A \in NP$ then there is a polynomial $p(n)$, where n is the size of an arbitrary input data, such that A can be solved by a deterministic algorithm having time complexity $O(2^{p(n)})$.

The other two important classes, closely related to NP are the *NP-hard* and *NP-complete* classes. In particular:

Definition 2.1.2 (**NP-hard**)**.** A problem Y is an NP-hard problem iff every problem $B_i \in$ NP is Karp-reducible to Y. Note that NP-hard problems are not required to be in NP.

Definition 2.1.3 (**NP-complete**)**.** A decision problem Y is an NP-complete problem iff i) $Y \in$ NP and ii) for each problem $B_i \in$ NP there is a Karp-reduction $f_i : B_i \mapsto Y$.

The focus of this thesis is one of the most famous combinatorial optimization problem, the bin packing problem, so it is worth to define also this class of problems.

Definition 2.1.4 (**Combinatorial optimization problem**)**.** Let E be a finite set with cardinality $|E| = n$, P_E the power set of E (hence $|P_E| = 2^n$) and the function $C : P_E \mapsto \mathbb{R}$. The general setup of a combinatorial optimization problem is to find an element $P \in P_E$ such that $C(P) = min_{P_i \in P_E} C(P_i)$.

Optimization problems are not in NP because they are not decision problems. For them there exist two specific classes, analogous to P and NP. These are **PO** (P-Optimization) and **NPO** (NP-Optimization), such that:

Definition 2.1.5 (**NPO**)**.** An optimization algorithm P belongs to NPO if:

1. the set of instances I is recognizable in polynomial time;
2. there exist a polynomial q such that, given an instance $x \in I$, for any $y \in SOL(x)$,

$|y| \leq q(|x|)$ and, besides, for any y such that $|y| \leq q(|x|)$, it is decidable in polynomial time whether $y \in SOL(x)$; being $SOL(x)$ the set of feasible solutions given the input x;

3. the objective function $obj(x)$ is computable in polynomial time.

It holds also the following theorem (proven in [4]):

Theorem 2.1.2. For any optimization problem P in NPO, the corresponding decision problem P_D belongs to NP.

The relationship between classes NP and NPO, which holds in the case of nondeterministic computations, can be translated, in the case of deterministic algorithms, by the following definition, that introduces the class of PO problems:

Definition 2.1.6 (PO). An optimization problem P belongs to the class PO if it is in NPO and there exists a polynomial-time computable algorithm $\mathcal{A}$ such that, for any instance $x \in I$, returns an optimal solution $y \in SOL(x)^*$, together with its value $obj(x)^*$.

Optimization problems are usually addressed by different classes of algorithms, such as exact methods, approximate methods, (meta-) heuristic search and enumeration methods, as it will be shown in 3.1.

One of the heuristics of interest in this work is the so called *simulated annealing*, described in the next section.

2.2 Simulated Annealing

Simulated annealing is a meta-heuristic ispired by metallurgy, were one tries to bring the metal to its lowest potential energy state by acting on its temperature. It is a meta-heuristic employed to approximate global optimization in a large search space for an optimization problem. It is often used when the search space is discrete (for example the traveling salesman problem, the boolean satisfiability problem, protein structure prediction and job-shop scheduling). For problems where finding an approximate global optimum is more important than finding a precise local optimum in a fixed amount of

time, simulated annealing may be preferable to exact algorithms such as gradient descent or branch and bound.

In 1983, this approach was used by Kirkpatrick, Gelatt Jr. and Vecchi [43] for a solution of the traveling salesman problem. They also proposed the name "simulated annealing".

Simulated annealing can be formalized as follows [37]:

given the combinatorial optimisation problem $\Lambda(\alpha_1, \alpha_2, ..., \alpha_n)$ where $\alpha_i \in \{0, 1\}$ and D the set of all possible combinations of α_i, for each element $\alpha \in D$ $\exists N(\alpha)$, the neighborhoord of α, which consists of elements $\alpha_i \in D$ that are close to α, according to a given metric. Finally, let $f : D \mapsto \mathbb{R}$ be a cost function that assigns a value to each element of D. The goal is to minimise f through the procedure described in 1.

Algorithm 1 Simulated annealing

Select an initial value $\omega \in D$;
Select the temperature change counter $k = 0$;
Select a temperature cooling schedule, t_k;
Select an initial temperature $T = t_0 \geq 0$;
Select a repetition schedule, M_k, that defines the number of iterations executed at each temperature t_k;

while Stopping criterion is not met **do**
    $m = 0$
    while $m! = M_k$ **do**
        generate a new candidate solution $\omega' \in \mathbb{N}(\omega)$;
        compute $\Delta = f(\omega') - f(\omega)$;
        if $\Delta \leq 0$ **then**
            $\omega = \omega'$;
        else
            $\omega = \omega'$ with probability $\exp\left(\frac{-\Delta}{t_k}\right)$;
        end if
        $m = m + 1$;
    end while
    $k = k + 1$;
end while

In 2.4.3 it will be shown the quantum counterpart of simulated annealing, the so called

"quantum annealing", but first, it is necessary to introduce some basic concepts of quantum mechanics in order to understand the physical reasons behind quantum annealing.

2.3 Quantum Mechanics

Quantum mechanics is a fundamental theory in physics that describes the behavior of matter and energy at the atomic and subatomic level. It was developed in the early 20^{th} century as a way to explain phenomena that could not be explained by classical physics, such as the observed behavior of particles like electrons.

In the following sections, after some historical elements, fundamental notions of quantum mechanics, necessary to understand the aspects of interest of quantum computing presented in this thesis, are given.

2.3.1 A Brief History of Quantum Mechanics

Quantum mechanics has a complex history that involves many scientists and theories. In 1900, Max Planck proposed that light is emitted in small, quantized packets of energy, called photons, that come as integral multiple of the quantity

$$E = h\nu = \hbar\omega \tag{2.1}$$

where $h \approx 6.63 \cdot 10^{-34} Js$ is the Planck's constant, and $\hbar \equiv h/2\pi = 1.06 \cdot 10{-}34 Js$. Being ν of order $10^{15} s^{-1}$, the quant of energy $h\nu$ is very small and so it appears as a continous spectrum in classical scale applications. However, in late 19^{th}century, physicists had to face a new enigma: the *blackbody radiation* problem. The issue was that the classical theory of radiation predicted that an infinite amount of energy was supposed to be emitted by such an object, which of course could not be correct. By introducing the hypothesis of quantized radiation, Planck got rid of the problem of the infinite energy and predicted the shape of the power curve as function of the temperature, although he did not understand that the quantization was inherent to light itself.

In 1905, Albert Einstein used this idea to explain the photoelectric effect, in which light

causes the emission of electrons from certain materials. He was the first to state that the quantization is in fact inherent to light and this packets of light can be interpreted as particles, today known as *photons*.

In 1913, Niels Bohr proposed that electrons in atoms have wave-like properties, which helped explaining certain observed behaviors, like quantized energy levels, of hydrogen atoms.

In 1924, Louis de Broglie suggested that *all* particles are associated with waves, leading to the concept of wave-particle duality. This proposal was a big step, because many things that are true for photons are not true for massive (and non-relativistic) particles. It turned out to be correct, in view of the fact that the resulting predictions agreed with experiments.

In the 1920s, Werner Heisenberg, Erwin Schrödinger, Max Born and Paul Dirac developed the theory of quantum mechanics, which used mathematical equations known as wave functions to describe the behavior of particles on the atomic and subatomic scale. These equations could predict the probability of a particle's location or energy at any given time, but not both with the same certainty degree, leading to the *Heisenberg's uncertainty principle*.

Initially, Schrödinger thought that the wave function represented the spatial charge density. It was Born to correctly interpret Schrödinger's wave as a probability amplitude These probability are not the result of some degree of *ignorance* about some variables of the system, but they represent *truly* random phenomena and there are not *hidden variable* (as stated by the Bell's theorem that solves the EPR paradox) that, if known, will make a quantum phenomen not random.

Over the years, quantum mechanics has been developed and refined. It had a significant impact on our understanding of the fundamental nature of matter and energy and on our understanding of the universe leading to numerous practical applications, including transistors, lasers, and modern computing technologies such as quantum computers.

2.3.2 Postulates of Quantum Mechanics

A physical system is generally described by three basic ingredients: states, observables, and dynamics (or law of time evolution).

A quantum description consists of a Hilbert space of states, observables are self-adjoint operators on the space of states, time evolution is given by a one-parameter group of unitary transformations on the Hilbert space of states.

Quantum mechanics is based on the following postulates:

1. **Quantum state**: the state of an isolated physical system is represented, at a fixed time t, by a state vector $|\Psi\rangle$ belonging to a Hilbert space $\mathcal{H}$ called the state space.

    1.1. *Composite system postulate* (follows from the previous one): the Hilbert space of a composite system is the Hilbert space tensor product of the state spaces associated with the component systems. For a non-relativistic system consisting of a finite number of distinguishable particles, the component systems are the individual particles.

2. **Observables**: every measurable physical quantity $\mathcal{A}$ is described by a Hermitian operator A acting in the state space $\mathcal{H}$. This operator is an observable, meaning that its eigenvectors form a basis for $\mathcal{H}$. The result of measuring a physical quantity $\mathcal{A}$ must be one of the eigenvalues of the corresponding observable A.

3. **Results of measurement**: when the physical quantity $\mathcal{A}$ is measured on a system in a normalized state $|\Psi\rangle$, the probability of obtaining an eigenvalue (denoted a_n for discrete spectra and α for continuous spectra) of the corresponding observable A is given by the squared amplitude of the appropriate wave function (projection onto corresponding eigenvector).

4. **Collapse of the wave function**: if the measurement of the physical quantity $\mathcal{A}$ on the system in the state $|\Psi\rangle$ gives the result a_n, then the state of the system, immediately after the measurement, is the normalized projection of $|\Psi\rangle$ onto the eigensubspace associated with a_n.

5. **Time evolution of the system**:

   - the time evolution of the state vector $|\Psi(t)\rangle$ is governed by the Schrödinger equation, where $H(t)$ is the observable associated with the total energy of the system (called the Hamiltonian):

$$i\hbar\frac{d}{dt}|\Psi(t)\rangle = H(t)|\Psi(t)\rangle \tag{2.2}$$

   or equivalently,

   - the time evolution of a closed system is described by a unitary transformation on the initial state:

$$|\Psi(t)\rangle = U(t, t_0)|\Psi(t_0)\rangle \tag{2.3}$$

2.3.3 Wave Function Ψ

A wave function, in quantum physics, is a mathematical description of the quantum state of an isolated quantum system. The wave function is a complex-valued probability amplitude, and the probabilities for the possible results of measurements made on the system can be derived from it.

According to the superposition principle of quantum mechanics, wave functions can be added together and multiplied by complex numbers to form new wave functions and Hilbert spaces. The inner product between two wave functions is a measure of the overlap between the corresponding physical states and it is used in the foundational probabilistic interpretation of quantum mechanics, the *Born rule*, relating transition probabilities to inner products.

The Schrödinger equation determines how wave functions evolve over time. They behave qualitatively like other waves, such as water waves or waves on a string, because the Schrödinger equation is mathematically a type of wave equation. This explains the name "wave function", and gives rise to wave-particle duality. However, the wave function in quantum mechanics describes a kind of physical phenomenon, still open to different interpretations, which fundamentally differs from that of classic mechanical waves.

In Born's statistical interpretation of non-relativistic quantum mechanics, the squared modulus of the wave function, $|\Psi|^2$, is a real number interpreted as the probability density of measuring a particle as being at a given place (or having a given momentum) at a given time, and possibly having definite values for discrete degrees of freedom. The integral of this quantity, over all the system's degrees of freedom, must be 1, in accordance with the probability interpretation. This is called the normalization condition. Since the wave function is complex-valued, only its relative phase and relative magnitude can be measured: its value does not tell anything about the magnitudes or directions of measurable observables; one has to apply quantum operators, whose eigenvalues correspond to sets of possible results of measurements, to the wave function Ψ and calculate the statistical distributions for measurable quantities.

For example, the state of a single non relativistic particle without spin is completely described by its wave function,

$$\Psi(x,t) \tag{2.4}$$

where x is position and t is time. This is a complex-valued function of two real variables x and t. For one spinless particle in one dimension, if the wave function is interpreted as a probability amplitude, the square modulus of the wave function, the positive real number

$$|\Psi(x,t)|^2 = \Psi^*(x,t)\Psi(x,t) = \rho(x,t) \tag{2.5}$$

is interpreted as the probability density that the particle is at x. If the particle's position is measured, its location cannot be determined from the wave function, but is described by a probability distribution. In particular, the probability that its position x will be in the interval $a \leq x \leq b$ is the integral of the density over this interval:

$$P_{a\leq x\leq b}(t) = \int_a^b |\Psi(x,t)|^2 dx \tag{2.6}$$

where t is the time at which the particle was measured. This leads to the normalization condition:

$$\int_{-\infty}^{\infty} |\Psi(x,t)|^2 dx = 1 \tag{2.7}$$

because if the particle is measured, there is 100% probability that it will be *somewhere*.

Thanks to the interpretation of the wave function as a probability density distribution, many, otherwise unsolvable, phenomena have been explained. The one of interest in this thesis, for its relation with quantum annealing, is the quantum tunneling effect described in the next section.

2.3.4 Quantum Tunneling

Quantum tunnelling is a quantum mechanical phenomenon whereby a wave function can propagate through a potential barrier higher than the wave's (particle's) total energy. This phenomenon is interesting and important because it violates the principles of classical mechanics.

The transmission through the barrier can be finite and depends exponentially on the barrier height and barrier width. The wavefunction may disappear on one side and reappear on the other. Quantum tunneling is not predicted by the laws of classical mechanics where surmounting a potential barrier requires a total amount of energy (kinetic + potential) greatr then the barrier height.

Quantum tunneling plays an essential role in several physical phenomena and is at the basis of quantum annealing (2.4.3).

A simple mental example to understand it and envision the difference with respect to the classical counterpart consists in thinking about a ball that tries to climb a hill. Classical mechanics predicts that particles that do not have enough energy to classically surmount a barrier cannot reach the other side. Thus, a ball without sufficient energy to surmount the hill would roll back down. Another example is a ball that hit a wall: if the ball lacks the energy to penetrate the wall it will bounce back.

In quantum mechanics, these particles can, with a small probability, *tunnel* to the other side, thus crossing the barrier.

The reason for this difference comes from treating matter as having properties of both waves and particles, and the Heisenberg uncertainty principle, which defines a limit on how precisely the position and the momentum of a particle can be simultaneously known. Thus, the probability of a given particle's existence on the opposite side of an intervening

barrier is non-zero, and such particle will appear on the other side in proportion to this probability (see 2.1).

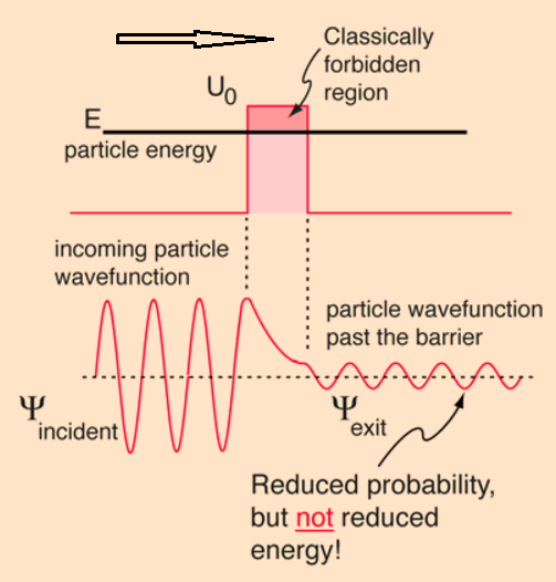

Figure 2.1: Quantum tunneling.

2.3.5 Hamiltonian of a Quantum System

In quantum mechanics, the Hamiltonian of a system is an operator corresponding to the total energy of that system, including both kinetic and potential energy. Its spectrum, the system's energy spectrum or its set of energy eigenvalues, is the set of all possible outcomes obtainable from a measurement of the system's total energy. The Hamiltonian takes different forms depending on the characteristics of the system under analysis, such as whether there is single or several particles in the system, interaction between particles, kind of potential energy, time varying potential or time independent one.

In general, the Hamiltonian is commonly expressed as the sum of operators corresponding to the kinetic and potential energies of a system. For a system of N particles, it takes the form:

$$\hat{H} = \sum_{n=1}^{N} \hat{T}_n + \hat{V} \tag{2.8}$$

where $\hat{T}_i$ is the kinetic energy of particle i and $\hat{V}$ is the potential energy of the system,

that depends on evolution time and spatial arrangement of particles.

The Hamiltonian operator is the fundamental block for the description of a quantum system and, from a computer science point of view, it can be used as a tool to express optimization problems in the jargon of quantum computers (in particular quantum annealers). This will be further explained in 2.4.1.

The last ingredients, necessary to understand quantum computing and in particular quantum annealing are the *adiabatic theorem* and the concept of *spin*, explained in the next paragraphs.

2.3.6 The Adiabatic Theorem

The adiabatic theorem of quantum mechanics was firstly stated by Max Born and Vladimir Fock in 1928 whith this words:

> *"A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum."*

In other words, a quantum mechanical system subjected to gradually changing external conditions adapts its functional form and remains in the same eigenstate as the initial one.

This can be mathematically formalized in the following way: given a slowly changing Hamiltonian $H(t)$ with instantaneous eigenstates $|n(t)\rangle$ and corresponding energies $E_n(t)$, a quantum system evolves from the initial state

$$|\Psi(0)\rangle = \sum_n c_n(0)|n(0)\rangle \tag{2.9}$$

to the final state

$$|\Psi(n)\rangle = \sum_n c_n(n)|n(n)\rangle \tag{2.10}$$

where

$$c_n(t) = c_n(0)e^{i\theta_n(t)}e^{i\gamma_n(t)},$$

$$\text{with} \quad \theta_m(t) = \frac{-1}{\hbar}\int_0^t E_m(t')dt' \qquad \text{the dynamical phase,} \tag{2.11}$$

$$\text{and} \quad \gamma_m(t) = i\int_0^t \langle m(t')|\dot{m}(t')\rangle dt' \quad \text{the geometric phase.}$$

In particular, $|c_n(t)|^2 = |c_n(0)|^2$ meaning that if the system starts in an eigenstate of $H(0)$ it remains in an eigenstate of $H(t)$ during the entire evolution (it only changes its phase).

As it will be discussed in 2.4.3, the adiabatic theorem is the theoretical basis that inspired the Quantum Annealing algorithm.

2.3.7 Spin

At subatomic level all particles have a property called *spin*, that can be imagined as an intrinsic angular momentum. Despite the name, particles do not literally spin around an axis, indeed quantum mechanical spin has no correspondence in classical physics.

As the name suggests, spin was originally conceived as the rotation of a particle around some axis. While the question of whether elementary particles actually rotate is ambiguous (as they appear point-like), this picture is correct insofar as spin obeys the same mathematical laws as quantized angular momenta do; in particular, spin implies that the particle's phase changes with angle. On the other hand, spin has some peculiar properties that distinguish it from orbital angular momenta: i) spin quantum numbers may take half-integer values; ii) although the direction of its spin can be changed, an elementary particle cannot be made to spin faster or slower; iii) the spin of a charged particle is associated with a magnetic dipole moment with a g-factor differing from 1. This could occur classically only if the internal charge of the particle were distributed differently from its mass.

The conventional definition of the spin quantum number is $s = \frac{n}{2}$, where n can be any non-negative integer.

The quantum-mechanical operator associated with spin-$\frac{1}{2}$ observables is:

$$\boldsymbol{S} = \frac{\hbar}{2}\boldsymbol{\sigma} \tag{2.12}$$

where its cartesian components are:

$$S_x = \frac{\hbar}{2}\sigma_x, \qquad S_y = \frac{\hbar}{2}\sigma_y, \qquad S_z = \frac{\hbar}{2}\sigma_z. \tag{2.13}$$

For the special case of spin-$\frac{1}{2}$ particles, σ_x, σ_y and σ_z are the three *Pauli matrices*:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.14}$$

Spin is an important concept in quantum mechanics because it plays a role in the behavior of subatomic particles and the way they interact with each other and with other forms of energy (as it will be shown in 2.4.1). It is also important in the study of the fundamental properties of matter and the fundamental forces of nature.

2.4 Quantum Computing

Quantum computing is a rapidly-emerging technology that harnesses the laws of quantum mechanics, such as superposition, interference, and entanglement to solve problems too complex for classical computers. Although current quantum computers may be too small to outperform classical computers for practical applications, larger realizations are believed to be capable of solving certain computational problems, such as integer factorization (which underlies RSA encryption), substantially faster than classical computers.

There are several models of quantum computation with the most widely used being quantum circuits. Other models are: the quantum Turing machine, quantum annealing, and adiabatic quantum computation. The majority of models are based on the quantum bit, or "qubit", which is somewhat analogous to the bit in classical computation. A qubit can be in a 1 or 0 quantum state, or in a superposition of the 1 and 0 states. When it is measured, however, its wave function collapses always into one between 0 or 1. The probability of the outcome depends on the qubit's quantum state immediately prior to measurement.

Efforts towards building a physical quantum computer focus on technologies such as transmons, ion traps and topological quantum computers, which aim to create high-quality qubits. These qubits may be designed differently, depending on the full quantum computer's computing model, as to whether quantum logic gates, quantum annealing, or adiabatic quantum computation are employed. There are currently a number of significant obstacles that prevent to construct useful quantum computers: it is particularly difficult to maintain qubits' quantum states, as they suffer from quantum decoherence, the lost of information of a quantum system due to its interaction with the environment. Quantum computers therefore require error correction.

Any computational problem that can be solved by a classical computer can also be solved by a quantum computer. Conversely, any problem that can be solved by a quantum computer can also be solved by a classical computer, at least in principle, given enough time. In other words, quantum computers obey the Church-Turing thesis. This means that while quantum computers provide no additional advantages over classical computers

in terms of computability, quantum algorithms for certain problems have significantly lower time complexities than corresponding known classical algorithms. Notably, quantum computers are believed to be able to quickly solve certain problems that no classical computer could solve in any feasible amount of time. A feature known as "quantum supremacy". The study of the computational complexity of problems with respect to quantum computers is known as quantum complexity theory.

Quantum complexity theory introduces a new complexity class, the *Bounded-error Quantum Polynomial time*, representing the class of problems solvable in polynomial time by an innately probabilistic quantum Turing machine. A decision problem is a member of BQP if there exists a quantum algorithm (an algorithm that runs on a quantum computer) that solves the decision problem with high probability (i.e. the probability goes to 1 when some parameter n goes to ∞) and is guaranteed to run in polynomial time. A run of the algorithm will correctly solve the decision problem with a probability of at least $2/3$.

As previously mentioned, adiabatic quantum computing is one of the possible models of quantum computation. An adiabatic quantum computer, based on quantum annealing, decomposes computation into a slow continuous transformation of an initial Hamiltonian into a final Hamiltonian, whose ground states contain the solution. One of the possible application is to solve the so called "Ising model".

In the next sections, the adiabatic quantum computation and its application in solving complex artificial intelligence problems are explained.

2.4.1 Ising Model

Meta-heuristics are mathematical models inspired by nature. The most useful models of nature are those that can be used to represent a large number of completely different systems. Understanding how such models work then leads to understand all of the physical systems the model can be used to represent. One of the most useful model in nature is the *Ising model* from statistical mechanics.

The Ising model (or Lenz-Ising model or Ising-Lenz model), named after the physicists Ernst Ising and Wilhelm Lenz, is a mathematical model of ferromagnetism in statistical

mechanics. The model consists of discrete variables that represent magnetic dipole moments of atomic spins that can be in one of two states (+1 or -1). The spins are arranged in a graph, usually a lattice (where the local structure repeats periodically in all directions), allowing each spin to interact with its neighbors. Neighboring spins that agree have a lower energy than those that disagree; the system tends to the lowest energy but heat disturbs this tendency, thus creating the possibility of different structural phases.

In mathematical terms, the Ising model is:

$$H(\sigma) = -\sum_{\langle i,j \rangle} J_{ij}\sigma_i\sigma_j - \mu\sum_j h_j\sigma_j \tag{2.15}$$

where the first sum is over pairs of adjacent spins (every pair is counted once). The notation $\langle i, j \rangle$ indicates that sites i and j are nearest neighbors. J_{ij} is the coupling term between spins i and j. h_j is the external magnetic field interacting with the j-th spin. μ is the magnetic moment.

Over time it was realized that 2.15 could be used to model many different physical systems. Any system that describes a set of individual elements (modeled by the spins s_j) interacting via pairwise interactions (the quadratic terms $s_i s_j$) can be described in this framework. In the period 1969 to 1997, more than 12,000 papers were published using the Ising model to describe systems in fields ranging from artificial intelligence to zoology.

But why is the Ising model so important for computer science? It is well known that in a system described by 2.15 the probability to observe a particular spin configuration σ is proportional to $\exp\left\{-\frac{H(\sigma)}{T}\right\}$, where T is the temperature of the system. At high temperature, $T \gg H(\sigma)$, all the possible configurations have the same probability, but at low temperature, the configurations having the lowest energy are the most likely state, with the limit of them being the only observable states at $T = 0$. This is a very interesting point, because in 1982, Francisco Baharona [7], showed that finding the ground state of an Ising model is an NP-hard problem by mapping an arbitrary SAT formula to the Ising model and showing that the SAT formula is satisfiable if and only if the lowest energy state of the Ising model is 0.

Ising models corresponding to physical systems are seamlessly solved by nature, and from here it comes the idea to exploit its computational power, by manufacturing an hardware that follows thermodynamics and statistical mechanics laws and let it solve Ising models.

So the high level idea behind quantum computing via quantum annealing is to map any NP-hard problem to the Ising model, then let a physical device to solve it and read its final configuration.

In the next section, a different formulation of Ising model, more suitable for optimization problems, is presented.

2.4.2 QUBO Formulation

The Ising model can be easily rewritten as a Quadratic Unconstrained Binary Optimization (QUBO) problem. Indeed, QUBOs are Ising models where the spin variables $s_i \in \{-1, +1\}$ are transformed into binary variables $x_i \in \{0, 1\}$. This transformation is easily realized through $s_i = 2x_i - 1$. If the QUBO objective is written as $H(x) = \sum_{i,j} x_i Q_{i,j} x_j$, where the linear terms arise from the diagonal elements due to $x_i^2 = x_i$ for binary variables, then

$$H(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{Q}\boldsymbol{x} \rangle = \langle \boldsymbol{x}, \widetilde{\boldsymbol{Q}}\boldsymbol{x} \rangle + \langle \widetilde{\boldsymbol{q}}, \boldsymbol{x} \rangle = \gamma + \langle \boldsymbol{s}, \boldsymbol{J}\boldsymbol{s} \rangle + \langle \boldsymbol{h}, \boldsymbol{s} \rangle \tag{2.16}$$

where

$$\gamma = \langle \mathbf{1}, \widetilde{\boldsymbol{Q}}\mathbf{1} \rangle / 4 + \langle \mathbf{1}, \widetilde{\boldsymbol{q}} \rangle / 2 \tag{2.17}$$

$$J = uptr(\widetilde{\boldsymbol{Q}} + \widetilde{\boldsymbol{Q}}^\top)/4 \tag{2.18}$$

$$h = \widetilde{\boldsymbol{q}}/2 + \langle \widetilde{\boldsymbol{Q}} + \widetilde{\boldsymbol{Q}}^\top, \mathbf{1} \rangle / 4 \tag{2.19}$$

Here $\widetilde{\boldsymbol{q}}$ is the vector of diagonal elements of Q, $\widetilde{\boldsymbol{Q}}$ is the matrix of off-diagonal elements of Q (the diagonal elements are zeroed), for a square matrix A the operation $uptr(A)$ zeroes the lower triangular part of A, and $\mathbf{1}$ is the vector of whose all components are 1. Thus, up to an irrelevant constant there is a simple relationship between the $\boldsymbol{h}, \boldsymbol{J}$ of an Ising model and the $\boldsymbol{Q}$ of a QUBO.

So far it has been shown that:

1. there are NP-hard problems that Nature can solve efficiently;
2. a subset of them can be formalized with the Ising model;
3. the Ising model can be reformulated (in polynomial time) as a QUBO (and vice versa).

So, by re-formulating a classical NP-hard problem into a QUBO, it could be possible to exploit the efficiency of Nature to optimally solve it.

In 4 this idea will be applied to the classical computer science problem of bin packing.

But before doing this, the way how Nature is employed to solve QUBO problems must be described. In the next section, such a technique, called *quantum annealing* is presented.

2.4.3 Quantum Annealing

Quantum annealing is a physical process exploited by quantum computation in order to solve combinatorial optimization problems. Quantum annealing is a restricted form of adiabatic quantum computation 2.3.6 but the range of problems that can be explored using this paradigm is vast and relevant to many fields of science and technology.

The most famous and technologically advanced hardware for quantum computing is from D-Wave Inc. [40].

The Hamiltonian of the D-Wave quantum computer can be written as:

$$\mathcal{H}_{ising} = \underbrace{-\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right)}_{\text{Final Hamiltonian}} \tag{2.20}$$

where $\hat{\sigma}_{x,z}^{(i)}$ are Pauli matrices operating on a qubit q_i, and h_i and $J_{i,j}$ are the qubit biases and coupling strengths.

The Hamiltonian is the sum of two terms, the initial Hamiltonian and the final Hamiltonian:

- Initial Hamiltonian: the lowest-energy state of the initial Hamiltonian is when all qubits are in a superposition state of 0 and 1. This term is also called the "tunneling Hamiltonian".
- Final Hamiltonian: the lowest-energy state of the final Hamiltonian is the answer to the problem that one is trying to solve. The final state is a classical state, and includes the qubit biases and the couplings between qubits. This term is also called the "problem Hamiltonian".

In quantum annealing, the system begins in the lowest-energy eigenstate of the initial Hamiltonian. As it anneals, it introduces the problem Hamiltonian, which contains the biases and couplers, and it reduces the influence of the initial Hamiltonian, has shown in 2.2. At the end of the anneal, it is in an eigenstate of the problem Hamiltonian. Ideally, it has stayed in the minimum energy state throughout the quantum annealing process (because of the adiabatic theorem 2.3.6) so that, by the end, it is in the minimum energy state of the problem Hamiltonian and therefore has an answer to the problem that has to be solved. By the end of the anneal, each qubit is a classical object.

Certain factors may cause the system to jump from the ground state into a higher energy state. One of them is thermal fluctuations that exist in any physical system. Another one is running the annealing process too quickly, violating the adiabatic assumptions. Because no real-world computation can run in perfect isolation, quantum annealing may be thought of as the real-world counterpart of adiabatic quantum computing 2.3.6, a theoretical ideal.
In the next section, the classical formulation of the bin packing problem will be described along a survey from the literature, of some of the methods employed to solve it.

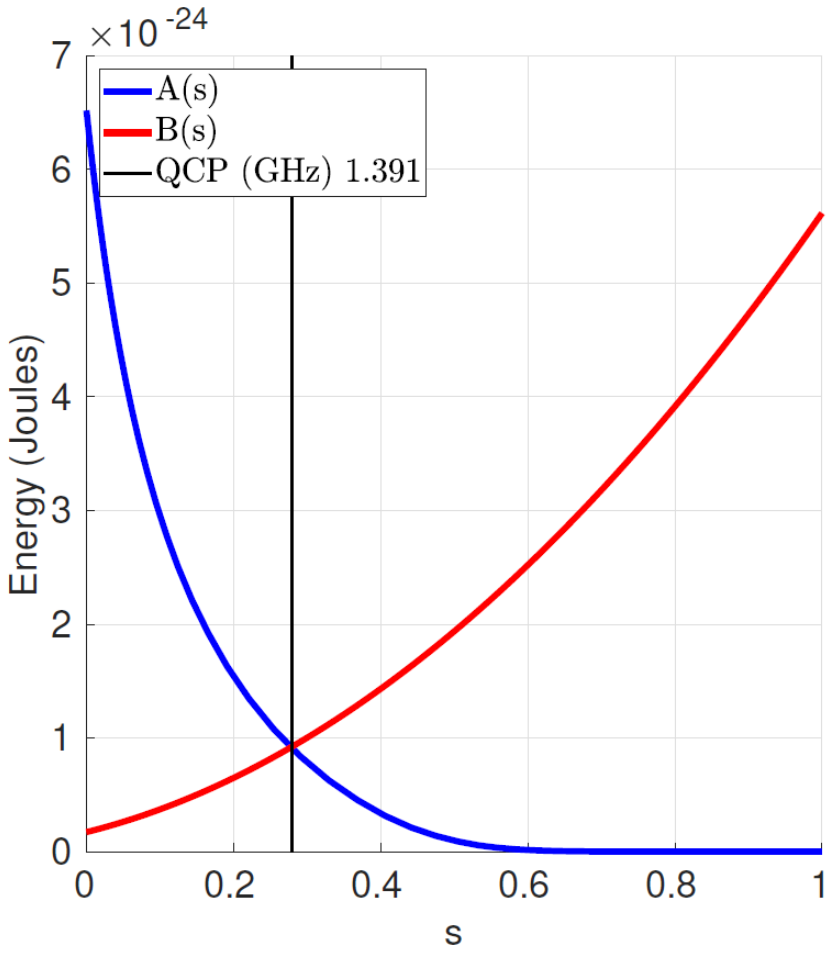

Figure 2.2: Annealing functions $A(s), B(s)$. Annealing begins at $s = 0$ with $A(s) \gg B(s)$ and ends at $s = 1$ with $A(s) \ll B(s)$. Data shown are representative of D-Wave 2X systems.

3 The Bin Packing Problem

The bin packing problem (BPP) is a classic optimization problem in computer science and operations research. It involves packing objects of different sizes into containers, or bins, with a limited capacity. The goal is to minimize the number of bins needed to pack all the objects.

A formal mathematical formulation of the bin packing problem can be expressed as follows: given a set of n items of given integer size (or weight) $w_j (j = 1, \ldots, n)$ the goal is to pack them into the minimum number of identical bins of integer capacity C.

Let m be any upper bound on the solution value and let introduce y_i, x_{ij} two sets of binary variables such that: $y_i (i = 1, \ldots, m)$ takes the value 1 if and only if bin i is used in the solution and $x_{ij} (i = 1, \ldots, m; j = 1, \ldots, n)$ takes the value 1 if and only if item j is packed into bin i. A possible simple Integer Linear Programming (ILP) model of the problem is ([51]):

$$\arg\min_{x,y} \sum_{i=1}^{m} y_i \tag{3.1}$$

$$\text{s.t.} \sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j = 1..n \tag{3.2}$$

$$\sum_{j=1}^{n} w_j x_{ij} \leq C y_i \qquad \forall i = 1..m \tag{3.3}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i = 1..m, \forall j = 1..n \tag{3.4}$$

$$y_i \in \{0, 1\} \qquad \forall i = 1..m \tag{3.5}$$

A variant of interest in practice is the so-called *online* bin packing: here the items of different size are supposed to arrive sequentially, and the decision maker has to decide whether to select and pack the currently observed item, or else to let it pass. Each decision is without recall. In contrast, *offline* bin packing allows rearranging the items in the hope of achieving a better packing once additional items arrive. This of course requires additional storage for holding the items to be rearranged.

3.1 Classical approaches to Bin Packing

In the offline bin packing all items are available from the very beginning and the goal is to find the best arrangement in a specified number of bins or, alternatively, minimizing their number.

Garey and Johnson [31] showed that BPP problem is **strongly NP-hard**, by transformation from the 3-Partition problem, while its decision formulation, i.e. the problem of deciding whether a set of items can fit into a specified number of bins, is **strongly NP-complete**.

A problem π is said to be strongly NP-complete if $\pi \in$ NP-Complete and there exists a polynomial p over the integers for wich π_p is also NP-Complete, where π_p denotes the subproblem of π obtained by restricting it to only those instances I such that $max(I) \leq p(length(I))$.

Despite its worst-case hardness, optimal solutions to very large instances of the problem can be produced with sophisticated algorithms. In addition, many approximation algorithms exist.

The bin packing problem is among the most intensively studied problems in combinatorial optimization: for example, the two recent surveys only, on exact methods (Delorme, Iori, and Martello [19]) and approximation algorithms (Coffman, Csirik, Galambos, Martello, and Vigo [16]), consider in total over 230 different references.

Most solution methodologies have been tried on these problems: different kinds of ILP

models, lower bound computations, branch-and-bound, branch-and-price, constraint programming, approximation algorithms, heuristics, and meta-heuristics.

3.1.1 Approximate algorithms

To measure the performance of an approximation algorithm there are two approximation ratios considered in the literature. For a given list of items L the number $A(L)$ denotes the number of bins used when algorithm A is applied to list L, while $OPT(L)$ denotes the optimum number for this list. The absolute worst-case performance ratio R_A for an algorithm A is defined as

$$R_A \equiv \inf\{r \geq 1 : A(L)/OPT(L) \leq r \; \forall L\}.$$

On the other hand, the asymptotic worst-case ratio R_A^∞ is defined as

$$R_A^\infty \equiv \inf\{r \geq 1 : \exists N > 0, A(L)/OPT(L) \leq r \; \forall L : OPT(L) \geq N\}.$$

Equivalently, R_A^∞ is the smallest number such that, for some constant K, for all lists L [16]:

$$A(L) \leq R_A^\infty \cdot OPT(L) + K.$$

Additionally, one can restrict the lists to those for which all items have a size of at most α. For such lists, the bounded size performance ratios are denoted as $R_A(\text{size} \leq \alpha)$ and $R_A^\infty(\text{size} \leq \alpha)$.

Approximation algorithms for bin packing can be classified into two categories:

- Online heuristics, that consider the items in a given order and place them one by one inside the bins. These heuristics are also applicable to the online version of this problem.
- Offline heuristics, that modify the given list of items e.g. by sorting the items by size. These algorithms are no longer applicable to the online variant of this problem. However, they have an improved approximation guarantee while maintaining the advantage of their small time-complexity.

Heuristics used by both the two classes are, among others:

- **Next-Fit (NF)**: always keeps a single open bin. When the new item does not fit into it, it closes the current bin and opens a new bin. Its advantage is that it is a bounded-space algorithm, since it only needs to keep a single open bin in memory. Its disadvantage is that its asymptotic approximation ratio is 2. In particular, $NF(L) \leq 2 \cdot OPT(L) - 1$, and for each $N \in \mathbb{N}$ there exists a list L such that $OPT(L) = N$ and $NF(L) = 2 \cdot OPT(L) - 2$. For each algorithm A that is an *AnyFit*-algorithm it holds that $R_A^\infty(\text{size} \leq \alpha) \leq R_{NF}^\infty(\text{size} \leq \alpha)$.

- **First-Fit (FF)**: keeps all bins open, in the order in which they were opened. It attempts to place each new item into the first bin in which it fits. Its approximation ratio is$FF(L) \leq \lfloor 1.7OPT \rfloor$, and there is a family of input lists L for which $FF(L)$ matches this bound.

- **Best-Fit (BF)**: keeps all bins open, but attempts to place each new item into the bin with the maximum load in which it fits. Its approximation ratio is identical to that of FF, that is: $BF(L) \leq \lfloor 1.7\text{OPT} \rfloor$, and there is a family of input lists L for which $BF(L)$ matches this bound.

Online Heuristics

In the online version of the bin packing problem, the items arrive one after another and the (irreversible) decision where to place an item has to be made before knowing the next item or even if there will be another one.

There are many simple algorithms that use the following general scheme:

- for each item in the input list:

    1. If the item fits into one of the currently open bins, then put it in one of these bins;

    2. Otherwise, open a new bin and put the new item in it.

The algorithms differ in the criterion by which they choose the open bin for the new item

in step 1.

Anyway, this thesis focuses on the offline version of the bin packing problem.

Offline Heuristics

In the offline version of bin packing, the algorithm can see all the items before starting to place them into bins. This allows to attain improved approximation ratios. The simplest technique used by offline algorithms is:

1. ordering the input list by descending size;
2. run an online algorithm on the ordered list.

The most famous algorithms in this family are:

- **First-fit-decreasing (FFD)**: orders the items by descending size, then calls First-Fit. Its approximation ratio is $FFD(I) = \frac{11}{9}OPT(I) + \frac{6}{9}$, and this is tight [75].
- **Next-fit-decreasing (NFD)**: orders the items by descending size, then calls Next-Fit. Its approximate ratio is slightly less than 1.7 in the worst case [6]. It has also been analyzed probabilistically [17]. Next-Fit packs a list and its inverse into the same number of bins. Therefore, Next-Fit-Increasing has the same performance as Next-Fit-Decreasing [29].

3.1.2 Heuristics and meta-heuristics

As for most NP-hard problems, starting from the early nineties many meta-heuristic approaches of all kinds have been proposed for the BPP, such as simulated annealing, Tabu search, population based algorithms, evolutionary and genetic heuristics coupled with hyper-heuristics, variable neighborhood search meta-heuristics etc. [5, 12, 24, 27, 28, 35, 36, 42, 46, 48, 49, 60, 64, 66–68, 71]

3.1.3 Pseudo-polynomial

Although model 3.1 involves a polynomial number of variables and constraints it is not very efficient in practice as shown in [19]. The literature has consequently focused on the

study of models with better computational performance, including pseudo-polynomial models. The drawback of these models is that the number of variables depends not only on the number of items but also on the bin capacity.

One of the first attempt, the so called "one-cut" formulation, was independently developed by Rao in 1976 [56] and by Dyckhof in 1981 [22]. Stadtler [69] studied the combinatorial structure of the one-cut model and compared it with the column generation approach, concluding that it could tackle only a subset of real world problems, tackled instead by the column generation approach.

Other relevant approaches are the "DP-flow" formulation by Cambazard and O'Sullivan [13], a method that relies on dynamic programming; and the "Arc-flow" formulation by Valério de Carvalho [15].

3.1.4 Exact methods

The first attempts to exactly solve the BPP were developed between the 50's and 60's using LP relaxations and dynamic programming (see [25, 32]). Starting from the 70's, research in this field focused on branch-and-bound techniques and then on branch-and-price.

Branch and bound

The branch and bound method is a mathematical optimization technique that is used to solve optimization problems, particularly those involving combinatorial optimization. It involves systematically exploring all possible solutions to a problem, assigning each solution a lower and upper bound on the potential value of the solution, and then iteratively branching into new solutions and bounding their potential value in order to find the optimal solution. The goal is to find the optimal solution by eliminating suboptimal solutions, or "bounding" them, through a process of elimination.

This method is typically used for problems with large numbers of possible solutions, such as traveling salesman or scheduling problems. The algorithm depends on efficient estimation of the lower and upper bounds of regions/branches of the search space. If no bounds are available, the algorithm degenerates to an exhaustive search. The root of

the three is usually initialized using a sub-optimal solution found by any chosen heuristic.

The two most famous and successful algorithms are the MTP, proposed by S.Martello and P.Toth [51], and BISON by Scholl et al. [64].

The first one was a branch-and-bound algorithm based on a novel dominance criterion and improved reduction procedures, later adopted by several authors. The second one, was the combination of some of the most powerful techniques from MTP with other emerging ones like Tabu search.

Branch and price

In applied mathematics, branch and price is a method of combinatorial optimization for solving integer linear programming (ILP) and mixed integer linear programming (MILP) problems with many variables. The method is a hybrid of branch-and-bound and column generation methods, where at each node of the search tree, columns may be added to the linear programming relaxation (LP relaxation).

The algorithm typically begins by using a reformulation, such as Dantzig-Wolfe decomposition, to form what is known as the Master Problem. The decomposition is performed to obtain a problem formulation that gives better bounds when its relaxation is solved rather than when the relaxation of the original formulation is solved. Unfortunately, the decomposition usually contains many variables and so a modified version, called the *Restricted Master Problem*, that only considers a subset of the columns is solved. Then, to check for optimality, a subproblem called the *pricing problem* is solved to find columns that can enter the basis and reduce the objective function (for a minimization problem). This involves finding a column that has a negative reduced cost.

Note that the pricing problem itself may be difficult to solve, but, since it is not necessary to find the column with the most negative reduced cost, heuristic and local search methods can be used. The subproblem must only be solved to completion in order to prove that an optimal solution to the Restricted Master Problem is also an optimal solution to the Master Problem. Each time a column is found with negative reduced cost, it is added to the Restricted Master Problem and the relaxation is reoptimized. If no columns

can enter the basis and the solution to the relaxation is not integer, then branching occurs.

Most branch and price algorithms are problem specific since the problem must be formulated in such a way that effective branching rules can be formulated and, so that the pricing problem is relatively easy to solve. If cutting planes are used to tighten LP relaxations within a branch and price algorithm, the method is known as *"branch-and-price-and-cut"*.

The most important contributions to branch-and-price method are from Gilmore and Gomory [32], Nitsche et al. [58], Caprara and Monaci [23] and Valério de Carvalho [14].

In the next section a different approach to the bin packing problem is presented: the idea is to reformulate the classic bin packing problem as a QUBO and solve it exploiting the computational power of quantum computing.

4 QUBO Formulation of Bin Packing

In this chapter, firstly, the other two QUBO formulations of the bin packing already presented in the literature are analyzed, then, our novel approach based on the framework of augmented Lagrangian is explained.

At the time of writing of this thesis, the only two QUBO models presented in literature, that aim to solve the bin packing problem end to end, are [47] in 2019 and [53] in 2022. In this work, I will refer to the first one as "Pseudo-Polynomial" formulation, and to the second one as "Unbalanced penalization".

There is another approach in literature, the one proposed in [2], where authors address the BPP with an hybrid approach, using quantum annealing to solve the sub-problem of filling a single bin. Not being an "end-to-end" approach, the latter will be ignored in this thesis.

4.1 Related works

4.1.1 Pseudo-Polynomial formulation

The first QUBO formulation of the Bin Packing Problem appeared in [47] in 2019. In this publication, the author corrects some wrong QUBO formulations given by Lucas in [50] and proposed some additional mappings of NP-complete and NP-hard problems into QUBO.

The proposed formulation is: given the variables $x_{ij}, i \in \{1, .., N\}, j \in \{1, .., K\}$, where $x_{ij} = 1$ if weight j is placed in bin i, variables $y_i, i \in \{1, .., N\}$, where $y_i = 1$ if bin i is not empty, and variables $z_{ik}, i \in \{1, .., N\}, k \in \{1, .., C\}$, where $z_{ik} = 1$ if bin i has been filled up to level k, that is, when the sum of the weights of the objects in bin i

equals exactly k. The Hamiltonian can be written as:

$$H_A = A\sum_{i=1}^{N}\left(x_i - \sum_{k=1}^{C} z_{ik}\right)^2 \tag{4.1}$$

$$+A\sum_{j=1}^{K}\left(1 - \sum_{i=1}^{N} x_{ij}\right)^2 \tag{4.2}$$

$$+A\sum_{i=1}^{N}\left(\sum_{k=1}^{C} k z_{ik} - \sum_{j=1}^{K} w_j x_{ij}\right)^2 \tag{4.3}$$

$$+A\sum_{i=1}^{N}(1 - x_i)\sum_{j=1}^{K} x_{ij} \tag{4.4}$$

$$H_B = B\sum_{i=1}^{N} x_i \tag{4.5}$$

with $A > 2B > 0$. The first term of H_A constraints the bin to be filled up to a unique level, while not used bins are not filled to any level at all. The second term ensures that every item is allocated to a bin. The third term is a penalization for configurations where bins are over filled, violating the capacity constraint, and the last term of H_A ensures that only not empty bins are counted. The term H_B represents the classical objective function, being the number of used bins.

The requirements $A > 2B$ ensures that bins are never filled beyond their capacity in favour of using fewer bins, as proved in [47]. Moreover, not allocating items is not favourable because it results in a penalty of A times the number of not allocated items, which is larger then 2B times the number of bin "saved" by non allocating items.

Although this is a correct QUBO formulation for the BPP, it has a weakness that could dramatically impact its scalability, and so, its applicability, on near term quantum devices: the first and third term of H_A present the variable z_{ik} that is the one-hot encoding of the bin capacity C. In particular, the third term of H_A is a "by the book" translation of the capacity constraint 3.3 into a penalty term.

Indeed, in combinatorial optimization problems there exist two classes of constraints:

- equality constraint, in the form **Ax = B**, and

- inequality constraint, expressed as $\mathbf{Ax} \leq \mathbf{B}$.

When mapping combinatorial optimization problems into QUBO, the equality constraint is mapped into a squared penalty term of the form $\alpha(\mathbf{Ax - B})^2$, while the inequality constraint is firstly converted into an equality one, introducing a slack variable **s** such that $\mathbf{Ax + s = B}$, and then mapped into its squared form $\beta(\mathbf{Ax + s - B})^2$ as for the previous case. Here α and β are the penalty multipliers that must properly be defined.

The introduction of a slack variables brings into the problem a considerable amount of additional variables: first of all, a slack variable is needed for each constraint in 3.3, and second, it must be binary encoded in order to define a proper QUBO model.

A possible way to do this is by hot-encoding the slack and this is exactly what has been done in [47]. It is worth to mention that there exists at least one alternative and better approach: the log-encoding. Here a constant N is represented by $M := \lfloor logN \rfloor$ new variables, instead of N of the one-hot encoding.

In order to count the number of variables in this QUBO formulation, without loss of generality, it can be assumed that given an instance of the BPP with n items, where $max(w_j) \leq C$, the upper bound on the number of required bins is $m = n$. Let's call $BPP(n, C)$ a particular instance of the BPP with n items and bins of capacity C. The total number of variables is $n(n + 1 + C)$. In 4.1 it is shown how this formulation scales with respect to the number of items of a BPP istance and to bin capacity.

It can be noticed that the introduction of the slack variables, not only adds nC more variables, but also makes this model a *pseudo polynomial* formulation, where the number of variables depends on the particular value of one of the input, the bin capacity C. This is a crucial observation because, as mentioned in [45], modern QPUs allow to run only small instances due to qubits topology and connectivity and a pseudo polynomial Hamiltonian could easily become intractable even for small instances.

4.1.2 Unbalanced penalization formulation

The second relevant QUBO model for the bin packing problem is the one presented by Barrera et al. in 2022 [53]. They start by assuming that an inequality constraint

$g(x) = \Sigma_i l_i x_i - C \leq 0$ can be approximated by the exponential function $e^{g(x)}$. In order to have a valid QUBO model, the exponential is expanded up to its second order Taylor's term, such that $e^{g(x)} \approx 1 + g(x) + \frac{1}{2}g(x)^2$. Thus their model resembles to:

$$H = obj(y_i) \tag{4.6}$$

$$+\lambda_0 \sum_{i=1}^{n} \left(\sum_{j=1}^{m} x_{ij} - 1 \right)^2 \tag{4.7}$$

$$+\lambda_1 \sum_{j=1}^{m} \left(\sum_{i=1}^{n} w_i x_{ij} - C y_i \right)^2 \tag{4.8}$$

$$+\lambda_2 \sum_{j=1}^{m} \left(\sum_{i=1}^{n} w_i x_{ij} - C y_i \right) \tag{4.9}$$

where $\lambda_{0,1,2}$ are estimated through the Nelder-Mead optimization method.

Although this model is similar to ours, there are some important differences, from both theretical and experimental point of view, that will be more evident in the next section.

From the theoretical point of view, they derived their model with a *bottom up* approach, by first approximating the infinite step function of the inequality constraint with the exponential function, and then by approximating the latter with its Taylor expansion. Instead, we chose the opposite path: we started from a general method, already known in optimization theory, and applied it to a specific problem, thus following a *top down* approach, with the advantage to be more generalizable to other COPs. Moreover, they estimated their penalty multipliers with an optimization method, while we estimated them analytically.

From the experimental point of view, Barrera et al. did not test their model on a real quantum device: they tested it by simulating QAOA using OpenQAOA and JUQCS (Jülich universal quantum computer simulator). Another important difference is that they chose QAOA, as the quantum framework to solve their problems, while we used Quantum Annealing. Quantum annealing is a more mature quantum technology with respect to QAOA, thus it allows to run bigger problems, in terms of number of QUBO variables. Moreover, QAOA must be "trained" before being able to solve a problem, while QA

must not. This represents an enormous advantage of QA over QAOA in terms of generalizability and scalability. Furthermore, they did not show how their model scales on instances with different number of items, but only on different randomly generated instances with the same number of items. In contrast, we will show how our model scales well across instances with different number of items.

In the next section, after a brief explanation of the Augmented Lagrangian method, our Augmented Lagrangian model, and its advantage with respect to the pseudo polinomial formulation, is presented.

4.2 Augmented Lagrangian formulation

In this section we present our novel heuristic QUBO formulation for the Bin Packing problem based on the Augmented Lagrangian method (AL).

Augmented Lagrangian methods are a class of algorithms for solving constrained optimization problems. They have similarities to penalty methods, in that they replace a constrained optimization problem by a series of unconstrained problems and add a penalty term to the objective; the difference is that the augmented Lagrangian method adds yet another term, designed to mimic a Lagrange multiplier. So, given the constrained problem:

$$\min f(\mathbf{x}) \tag{4.10}$$

$$\text{s.t. } c_i(\mathbf{x}) = \mathbf{b}\ \forall i \in \mathcal{D} \tag{4.11}$$

the augmented Lagrangian method transforms it into:

$$\min \Phi(\mathbf{x}) = f(\mathbf{x}) + \sum_{i \in \mathcal{D}} \rho_i (c_i(\mathbf{x}) - \mathbf{b})^2 + \sum_{i \in \mathcal{D}} \lambda_i (c_i(\mathbf{x}) - \mathbf{b}) \tag{4.12}$$

Usually, in CSP and COP it is useful to introduce also *redundant constraints*, i.e. constraints that do not change the set of feasible solutions, but help the solver to converge

faster. In the specific case of bin packing, we introduce the following redundant constraint:

$$\sum_{j=1}^{n} y_i x_{ij} = \sum_{j=1}^{n} x_{ij} \quad \forall i = 1..m \tag{4.13}$$

meaning that if a bin i is not used, i.e. $y_i = 0$, it can not contain any item, i.e. $\sum_{j=1}^{n} x_{ij} = 0$.

Inspired by augmented Lagrangian framework and the just mentioned redundant constraints, our proposed model is:

$$\arg\min_{x,y} \delta \sum_{i=1}^{m} y_i \tag{4.14}$$

$$+ \sum_{i=1}^{m} \lambda_i \left(\sum_{j=1}^{n} w_j x_{ij} - c_i y_i \right) \tag{4.15}$$

$$+ \sum_{i=1}^{m} \rho_i \left(\sum_{j=1}^{n} w_j x_{ij} - c_i y_i \right)^2 \tag{4.16}$$

$$+ \theta \sum_{j=1}^{n} \left(\sum_{i=1}^{m} x_{ij} - 1 \right)^2 \tag{4.17}$$

$$+ \gamma \sum_{i=1}^{m} (1 - y_i) \sum_{j=1}^{n} x_{ij} \tag{4.18}$$

$$\text{s.t. } x_{ij} \in \{0, 1\} \qquad \forall i = 1..m, \forall j = 1..n \tag{4.19}$$

$$y_i \in \{0, 1\} \qquad \forall i = 1..m \tag{4.20}$$

The two penalties 4.15 and 4.16 are the augmented Lagrangian expansion of 3.3 and it can been observed that for unfeasible configurations both terms add a penalty $\lambda_i s_i + \rho_i s_i^2$, while for feasible configurations the linear term is negative and so it gives a reward to the solver. The crucial aspect is to carefully estimate λ and ρ values in order to correctly model the solution space. The same holds for θ in 4.17: this term represents a penalization of θ, when item j is not placed and $(k-1)\theta$ if item j is placed k times. It is worth to notice that this penalty term is not the augmented Lagrangian expansion of 3.2 but it is a pure squared penalty: the reason is that in this case we don't want to give a reward to the solver when an item is not placed at all. Finally, the last term is the

penalty term associated to the redundant constraints 4.13. It represents a penalization of $\sum_{j \in \mathcal{J}} x_{ij}$ when the set of items $\mathcal{J}$ is assigned to bin i without setting the corresponding y_i to 1.

It is worth to notice that in the standard AL approach, inequality constraints are first transformed into equality ones, by introducing a slack variable, and then added to the Lagrangian as per 4.12. Instead, the AL formulation proposed in this work does not contain the slack variable but directly the capacity constants c_i. This is an important aspect that brings the advantage with respect to the pseudo polynomial approach and must be taken into account during penalty estimation.

Usually, estimation of penalty multipliers is done using the so called Alternating Direction Method of Multipliers ADMM or other recursive approximation methods. In this work we propose an analytical estimation of penalty multipliers and we will experimentally show how well it works on the selected input instance.

Our claim is that this method works for the set of instances selected, but its general applicability is still to be proven in a future work.

Penalties estimation We can now derive heuristics conditions for the penalty multipliers. We chose to design our conditions based on approximate worst-case reasoning, so that they can be expected to lead to optimal or slighly sub-optimal solutions for most BP instances.

An abstract form for the augmented Lagrangian terms associated to a bin is given by:

$$\lambda_i(s_i - c_i y_i) + \rho_i(s_i - c_i y_i)^2 \tag{4.21}$$

Let's start by considering the case where $y_i = 0$. In this situation, using the smallest bin usage amount should be as expensive as using the bin, i.e.:

$$\lambda_i(w_{min} - 0) + \rho_i(w_{min} - 0)^2 \geq 1 \tag{4.22}$$

where w_{min} is the smallest item weight, i.e. $w_{min} = \min\{w_j\}$. If this condition is satisfied, then using even more capacity can only make things worse, i.e. the solver will

naturally choose to set $y_i = 1$. If $y_i = 0$ and no capacity is used, we have:

$$\lambda_i(0-0) + \rho_i(0-0)^2 \geq -1 \tag{4.23}$$

which is trivially true. Let's consider now the case where $y_i = 1$. In this situation, exceeding the capacity by any amount should be as expensive as using one more bin, i.e.:

$$\lambda_i(c_i + w_{min} - c_i) + \rho_i(c_i + w_{min} - c_i)^2 \geq 1 \tag{4.24}$$

which conveniently is the same condition as above. The last condition to fit the parabola 4.21 concerns the feasibility region. Unlike in the case where $y_i = 0$, now it is possible to satisfy the constraint with some slack. In this case, the Lagrangian term might provide a reward (i.e. negative cost) in case the constraint is satisfied with some slack. We need such reward to be small enough that it does not provide an incentive for using another bin, i.e.:

$$\lambda_i(-\frac{c_i}{2}) + \rho_i(-\frac{c_i}{2})^2 \geq 0 \tag{4.25}$$

Basically here we are fitting a parabola using the three conditions 4.22, 4.23 and 4.25 in order to approximate the wall shape of the real feasibility region.

We can therefore obtain values for λ and ρ by stating all conditions for their least restrictive values:

$$w_{min}\lambda_i + w_{min}^2\rho_i = 1 \tag{4.26}$$

$$-\frac{c_i}{2}\lambda_i + \frac{{c_i}^2}{4}\rho_i = 0 \tag{4.27}$$

From here we obtain:

$$\lambda_i = \frac{c_i}{w_{min}\left(2w_{min} + c_i\right)} \tag{4.28}$$

$$\rho_i = \lambda_i\frac{2}{c_i} = \frac{2}{w_{min}\left(2w_{min} + c_i\right)} \tag{4.29}$$

The next step is to calibrate θ_j.

The abstract Lagrangian term associated to item j is:

$$\theta_j(p_j - 1)^2 \tag{4.30}$$

where $p_j \in \mathbb{N}$ being the number of times item j has been assigned to a bin. Because now we are only talking about assignment and not capacities, we can get rid of index j and rewrite the previous equation as:

$$\theta(p_j - 1)^2 \tag{4.31}$$

Moreover we want to force the solver to assign all items at maximum to one bin, so the penalty should arise every time $p_j \neq 1$. Indeed, in case $p_j = 1$ we have:

$$\theta(1 - 1)^2 \geq -1 \tag{4.32}$$

which is true.
In case $p_j = 0$, i.e. item j not assigned to any bin, the inequality boils down to:

$$\theta \geq 2 \tag{4.33}$$

where the 2 is chosen in order to have a penalty greater then the cost of opening a new bin. It remains to calibrate γ.
The abstract Lagrangian term associated to the γ term is:

$$\gamma(1 - y_i)k_i \tag{4.34}$$

This term comes into play only when $y_i = 0$ and $k_i \neq 0$ by adding k_i times the penalty γ. We want the minimum penalty to be at least equal to the cost of opening a new bin, thus

$$\gamma \geq 1. \tag{4.35}$$

The last parameter to be estimated is δ: although it is not properly a penalty term, being it the multiplier of the objective function, it is useful to have it there in order to control

all the other parameters, for example to avoid problems when dealing with too small numbers. Another reason to use this multiplier is to correct the undesirable behaviour of this model for certain configurations of items: sometimes it can happen that the model prefers configurations where one or more bins are slightly overfilled because opening a new one costs too much. Obviously this depends on the particular combination of weights of the istance with respect the bin capacity, and their number. To correct this behaviour we require:

$$\delta \leq \lambda s_{min} + \rho s_{min} \tag{4.36}$$

where s_{min} is the minimum amount of capacity that can be exceeded, so $s_{min} \geq 1$.

Model analysis In terms of number of variables this model is way smaller then its pseudo-polynomial counterpart in [47]. In this case, given a BPP instance $BPP(n, C)$ the number of variables is $n(n+1)$, given by the number of bins plus the n^2 variables x_{ij}. We are assuming $m = n$ as upper bound on the number of bins. This formulation presents a double advantage: first of all the number of variables is independent on the instance characteristics (bin capacity); secondly, the reduced number of variables let the model be more suitable for today's QPUs. We believe, but this must to be proven in a future work, that independency with respect to bin capacity makes the model more stable in terms of penalty multipliers values.

Fig. 4.1 shows a comparison of the asymptotic performace, in the number of variables, of the two approaches. The continous dark red line at *Num of variables = 180* represents the maximum number of fully connected variables usable in D-Wave Advantage QPU with 5640 qubits.

Contribution This work brings several contributions to the Quantum AI field, that can be summarized as follows:

- we found a connection between QUBO models and augmented Lagrangian methods that can leverage all the work already done in these two fields, thus paving the way for future synergies;

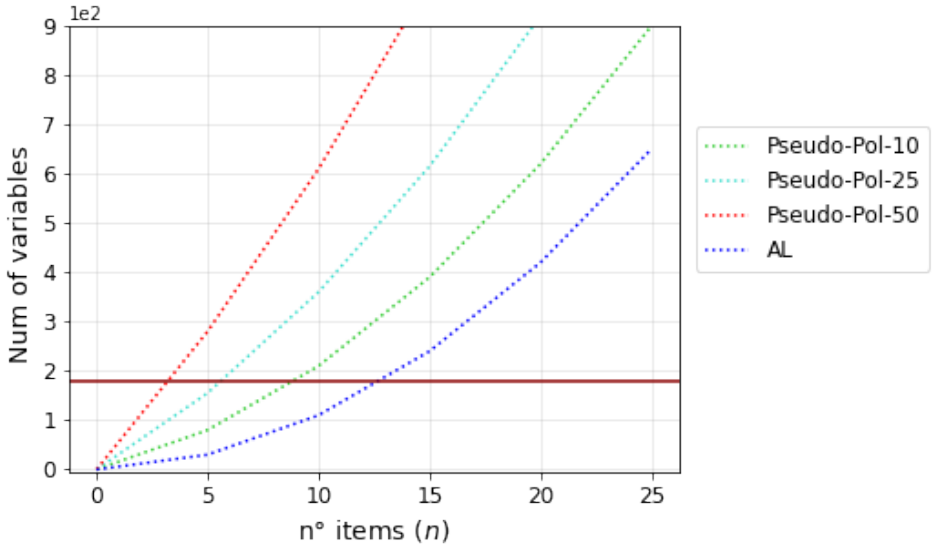

Figure 4.1: Comparison of the growth of the number of variables for the Pseudo polynomial and Augmented Lagrangian models with respect the number of items and bins capacity. For the first one, three values of C are shown.

- we derived a QUBO formulation of the bin packing that scales well with instance dimension, on a selected set of instances, in the limit of modern QPUs;
- our QUBO formulation for the bin packing problem is independent on bins capacity;
- we propose an analytical method for penalties estimation, in the context of augmented Lagrangian, and we show that this estimation works well for the set of tested instances;
- we are the first in history to solve the bin packing problem, end to end, on a real quantum device, thus letting us to experimentally compare our model performances against the state of the art in the classical field;
- we are the first to solve the bin packing problem, end to end, using Quantum Annealing.

In the next section the experimental performance analysis and comparison with respect to the classical state of the art is presented.

5 Experimental results of QAL-BP

In this chapter we show how our model performs when solved by a real quantum device through quantum annealing. Then we compare these results with the state of the art in the classical field. Before doing this, we describe the experimental hardware and software employed and the instances used in tests.

5.1 Experimental setup

In the following paragraphs we describe the experimental setup where performance tests have taken place. Then we discuss the dataset used to evaluate our model and the model parameters chosen starting from the sufficient conditions shown in 4.2.

5.1.1 Hardware and software

To determine the correcteness and performance of our model, we tested it on the most recent quantum annealing device by D-Wave System Inc., the *Advantage_system4.1*. It consists of more than 5000 qubits, allowing to run problems up to 180 fully connected binary variables. Because of the little availability of resources, we did not fine-tuned the QPU solving strategy, in terms of annealing time, chain strength and minor embedding; thus, the results presented here refer to the default configuration of D-Wave QPU. This is an important aspect to keep in mind while reading the rest of this chapter, because we are going to show a performance comparison between our model, solved by Quantum Annelaing with default configuration, and GUROBI, the state of the art solver for MILP. Runtime performance of the latter instead, is evaluated by runinng it in Google COLAB environment, on a virtual machine with 2vCPU@2.2GHz and 13GB RAM.

5.1.2 Tested instances

All the experiments are carried out on a set of eight classes of randomly generated instances, ranging from 3 to 10 items, with fixed bin capacity equal to 10, although our

model can in principle handle different capacities per bin. Item weights range between 4 and 10. The apparent small range of instances comes from technology limitations of today's QPUs: as mentioned before and shown in 4.1, we are restricted to 180 fully connected binary variables, and above all, it will be shown later that the more we get close to that limit the more the solution degradates, due to technological immaturity of quantum hardware and absence of fine-tuning. The randomly generated instances are tshown in 5.1.

5.1.3 Models parameters

In 4.2 we derived the sufficient domain condintions for our model penalty multipliers. Based on those we chose:

$$\delta = 0.15 \tag{5.1}$$
$$\lambda = 0.1389 \tag{5.2}$$
$$\rho = 0.0278 \tag{5.3}$$
$$\theta = 2 \tag{5.4}$$
$$\gamma = 1 \tag{5.5}$$

For what concerns the pseudo-polynomial model by [47], because the author did not estimated A and B multipliers, we tried to find them through a grid search over a parameter space of 88 pairs, spanned by:

$$A \in [.1, .3, .5, 1, 3, 5, 10, 25, 50, 100, 1000] \tag{5.6}$$
$$B \in [.01, .05, .1, .5, 1, 2, 5, 10] \tag{5.7}$$

Over 3520 problems (88 parameters pairs $\times$ 40 instances), simulated annealing found a solution for only 5 of them. This results highlights another weakness of this model: besides being it pseudo-polynomial, using the same penalty multiplier A for each penalty term prevents it to converge. Because of these poor results, we decided to omit this model from the rest of the analysis.

instance name	bin capacity	weights list	num of weights	w_min	w_max	lower bound
bpp_3_10_23	10	[4, 8, 6]	3	4	8	2
bpp_4_10_23	10	[8, 5, 4, 8]	4	4	8	3
bpp_5_10_23	10	[4, 4, 8, 8, 9]	5	4	9	3
bpp_6_10_23	10	[7, 5, 5, 5, 4, 9]	6	4	9	4
bpp_7_10_23	10	[9, 7, 8, 6, 9, 6, 7]	7	6	9	5
bpp_8_10_23	10	[4, 5, 7, 5, 6, 4, 6, 4]	8	4	7	4
bpp_9_10_23	10	[7, 6, 8, 4, 8, 4, 9, 6, 4]	9	4	9	6
bpp_10_10_23	10	[5, 8, 6, 7, 10, 9, 4, 10, 7, 4]	10	4	10	7
bpp_3_10_42	10	[4, 9, 8]	3	4	9	2
bpp_4_10_42	10	[7, 7, 10, 4]	4	4	10	3
bpp_5_10_42	10	[8, 5, 4, 7, 10]	5	4	10	4
bpp_6_10_42	10	[9, 9, 9, 9, 7, 4]	6	4	9	5
bpp_7_10_42	10	[9, 7, 7, 6, 5, 10, 9]	7	5	10	5
bpp_8_10_42	10	[8, 6, 9, 7, 7, 7, 5, 4]	8	4	9	5
bpp_9_10_42	10	[7, 10, 4, 10, 9, 5, 8, 5, 9]	9	4	10	7
bpp_10_10_42	10	[8, 6, 4, 10, 7, 10, 8, 9, 9, 5]	10	4	10	7
bpp_3_10_123	10	[4, 8, 8]	3	4	8	2
bpp_4_10_123	10	[4, 10, 5, 5]	4	4	10	3
bpp_5_10_123	10	[5, 6, 5, 6, 9]	5	5	9	3
bpp_6_10_123	10	[7, 10, 7, 5, 9, 9]	6	5	10	5
bpp_7_10_123	10	[10, 10, 4, 7, 5, 5, 5]	7	4	10	5
bpp_8_10_123	10	[9, 9, 5, 6, 9, 5, 8, 7]	8	5	9	6
bpp_9_10_123	10	[10, 9, 5, 9, 9, 5, 7, 9, 5]	9	5	10	7
bpp_10_10_123	10	[5, 5, 4, 7, 4, 8, 6, 5, 6, 4]	10	4	8	5
bpp_3_10_90	10	[8, 6, 4]	3	4	8	2
bpp_4_10_90	10	[8, 5, 7, 6]	4	5	8	3
bpp_5_10_90	10	[6, 7, 8, 7, 4]	5	4	8	3
bpp_6_10_90	10	[7, 8, 9, 9, 10, 6]	6	6	10	5
bpp_7_10_90	10	[6, 4, 4, 4, 8, 9, 6]	7	4	9	4
bpp_8_10_90	10	[7, 10, 8, 8, 8, 5, 5, 8]	8	5	10	6
bpp_9_10_90	10	[9, 6, 4, 10, 10, 5, 4, 4, 6]	9	4	10	6
bpp_10_10_90	10	[9, 6, 8, 7, 8, 10, 9, 6, 9, 10]	10	6	10	8
bpp_3_10_510	10	[5, 8, 6]	3	5	8	2
bpp_4_10_510	10	[7, 9, 5, 5]	4	5	9	3
bpp_5_10_510	10	[6, 10, 4, 9, 4]	5	4	10	3
bpp_6_10_510	10	[5, 5, 9, 10, 8, 6]	6	5	10	4
bpp_7_10_510	10	[9, 7, 9, 4, 10, 10, 8]	7	4	10	6
bpp_8_10_510	10	[9, 10, 8, 9, 4, 4, 9, 5]	8	4	10	6
bpp_9_10_510	10	[5, 9, 10, 9, 7, 8, 4, 10, 6]	9	4	10	7
bpp_10_10_510	10	[10, 5, 9, 5, 8, 9, 7, 4, 6, 9]	10	4	10	7

Table 5.1: The set of bin packing instances chosen to test all models under evaluation.

5.2 QAL-BP performance evaluation

In order to asses QAL-BP performace we randomly generated five instances for each of the eight classes and we solved them using Gurobi, simulated annealing, quantum annealing and an exact solver (the latter only for instances with 3 and 4 items).

Solving a problem on a quantum annealing device consists in a complex chain of tasks like, calling D-Wave APIs, queueing the task, solving it and so on. Some of this operations, such as the annealing time, can affect the solution quality. Moreover, when analysing QPU runtime performance, there are several time intervals involved. In order to make a meaningful comparison with Gurobi, the correct statistic to take into account, that represents the time to solution, is the so called *"qpu_sampling_time"*, as shown in 5.1.

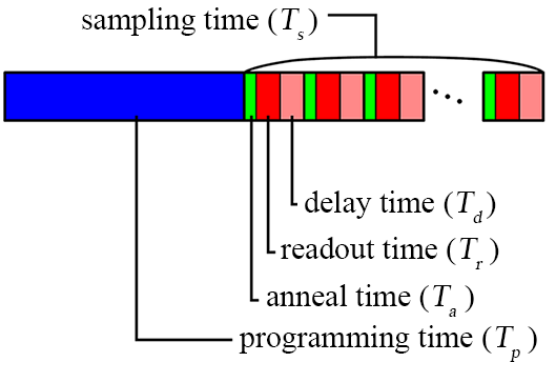

Figure 5.1: Runtime breakdown of problem programming and sampling by a D-Wave QPU.

A charecteristic parameter of both quantum and simulated annealing is the *number of reads*, that represents the number of anneals. We chose $num\_reads = 1000$ and, in order to be comparable, we solved each instance 1000 times with Gurobi and then we take the average runtime as Gurobi's TTS. In order to get an idea of how much hard the bin packing problem is, already at small scales, we tried to solve each instance also using the exact solver from D-Wave library. In 5.2 and 5.3 is shown the mean time to solution TTS across the five instances for each class and for each solver.

This astonishing result shows that our model, when solved by quantum annealing on a

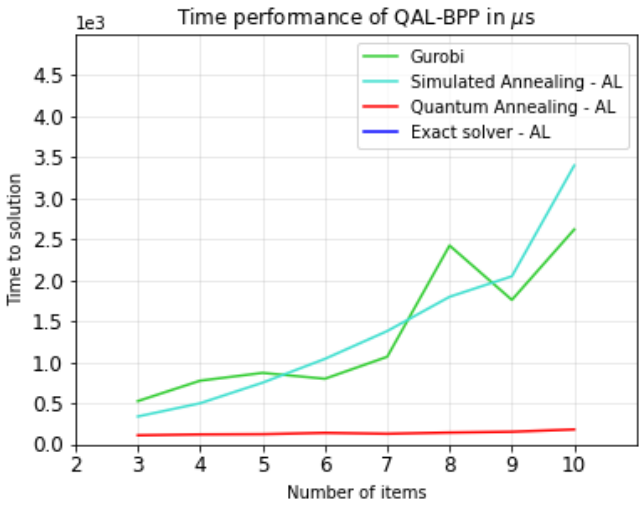

Figure 5.2: Time to solution comparison between Quantum Annealing, Simulated Annealing, Gurobi and an exact solver.

real quantum computer, dramatically outperforms Gurobi, the state-of-the-art solver for MILP problems.

In 5.5, the probability of being feasible for the best found solution, for both simulated and quantum annealing, is shown. We can observe that for "large" instances (with respect current QPUs limitations), the solution quality degradates, but looking at how simulated annealing performs on the same instances, in fig.5.6, allows us to say that QA bad performance on "large" scales is mainly due to technology immaturity. For these instances, QA reaches a minimum that is higher than the one reached by simulated annealing. The fact that simulated annealing can reach a better minimum proves that the wrong answers from quantum annealing are manily due to technology immaturity or the solver being not fine-tuned.

In fig 5.4 we can see the minimum number of bins found by each solver (except for the exact one) and for each instance. We can observe that Gurobi outperforms our heuristic model, in both the annealing techniques, in only two cases: the instances *bpp_8_10_23* and *bpp_10_10_123*. This results means, on one hand, the strength of our heuristic formulation, and, on the other one, it is compatible with our formulation being a **heuristic** model.

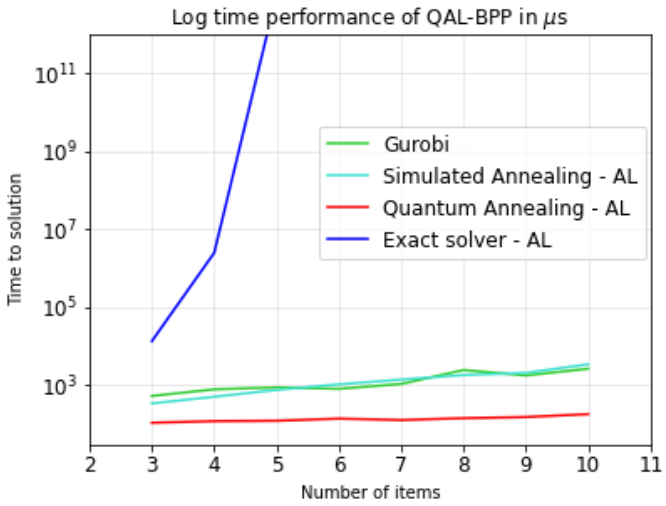

Figure 5.3: Time to solution comparison, in logarithmic scale, between Quantum Annealing, Simulated Annealing, Gurobi and an exact solver.

Moreover, in 5.7 we show that, even for instances where QA fails to reach a feasible minimum, the first feasible solution is close to the minimum, allowing to search for the first feasible in a reasonable time (checking if a solution is feasible can be done in polynomial time by definition of NP-Hard). Obviously, in this case the first feasible solution will likely not be the best one.

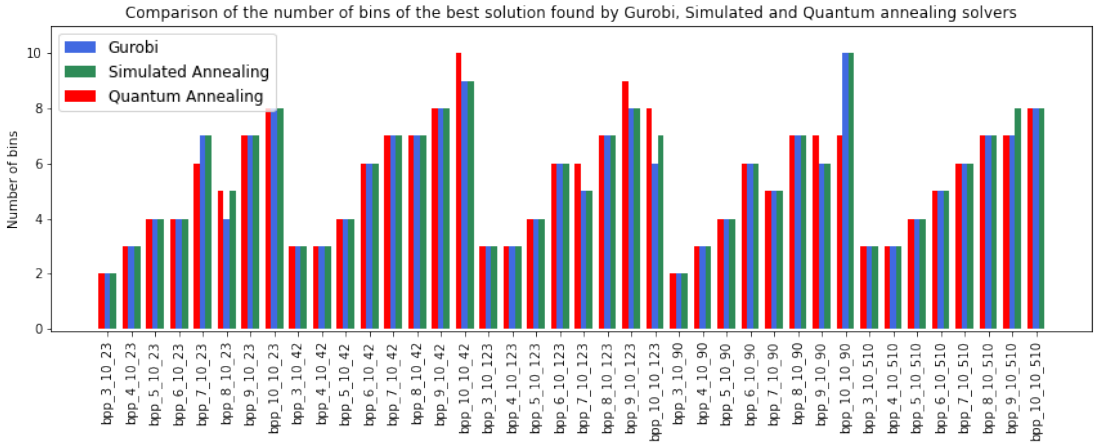

Figure 5.4: Comparison of the number of bins of the best solution found by Gurobi, Simulated and Quantum annealing solvers.

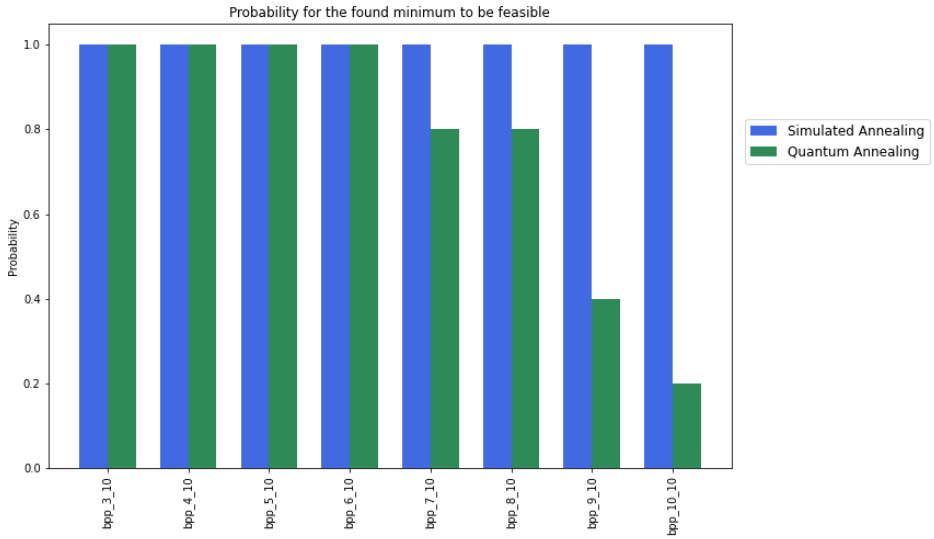

Figure 5.5: Probability of being a feasible solution for the minimum found by Simulated and Quantum Annealing.

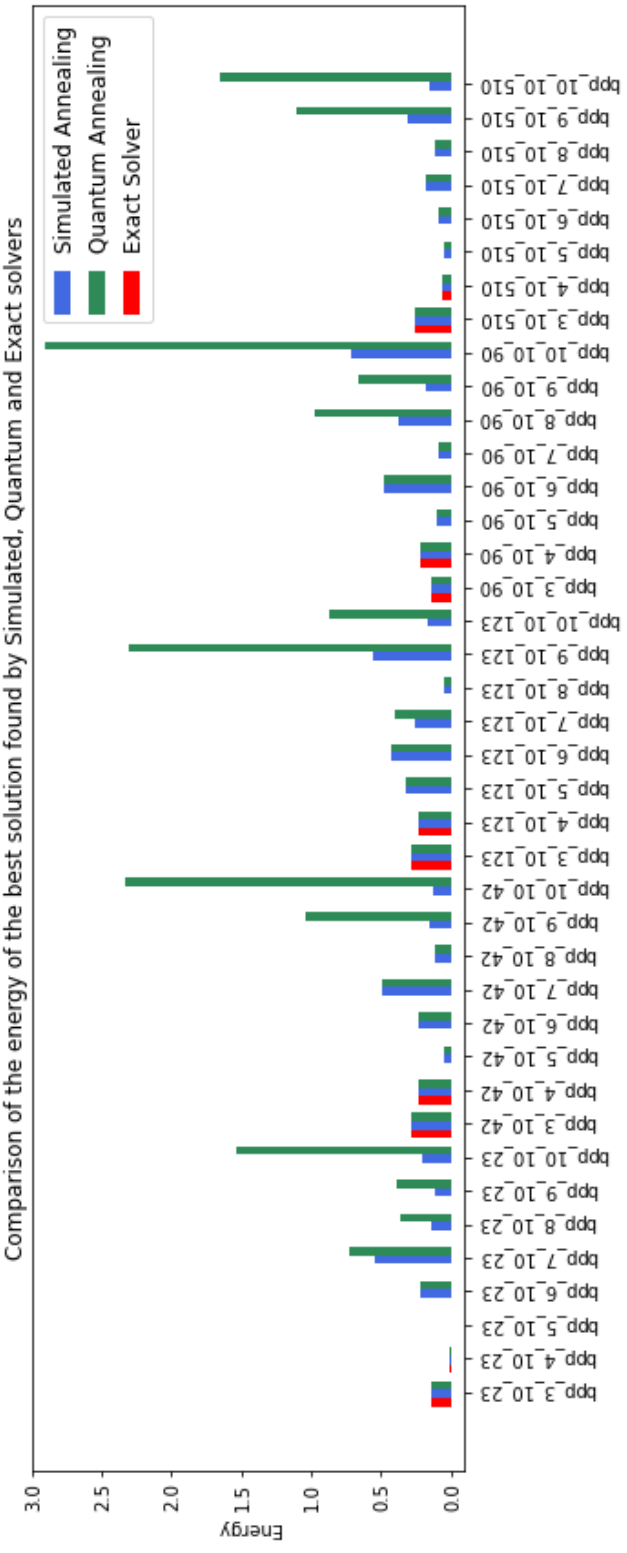

Figure 5.6: Comparison of the energy of the best solution found by simulated, quantum annealing and the exact solver when a minimum is reached.

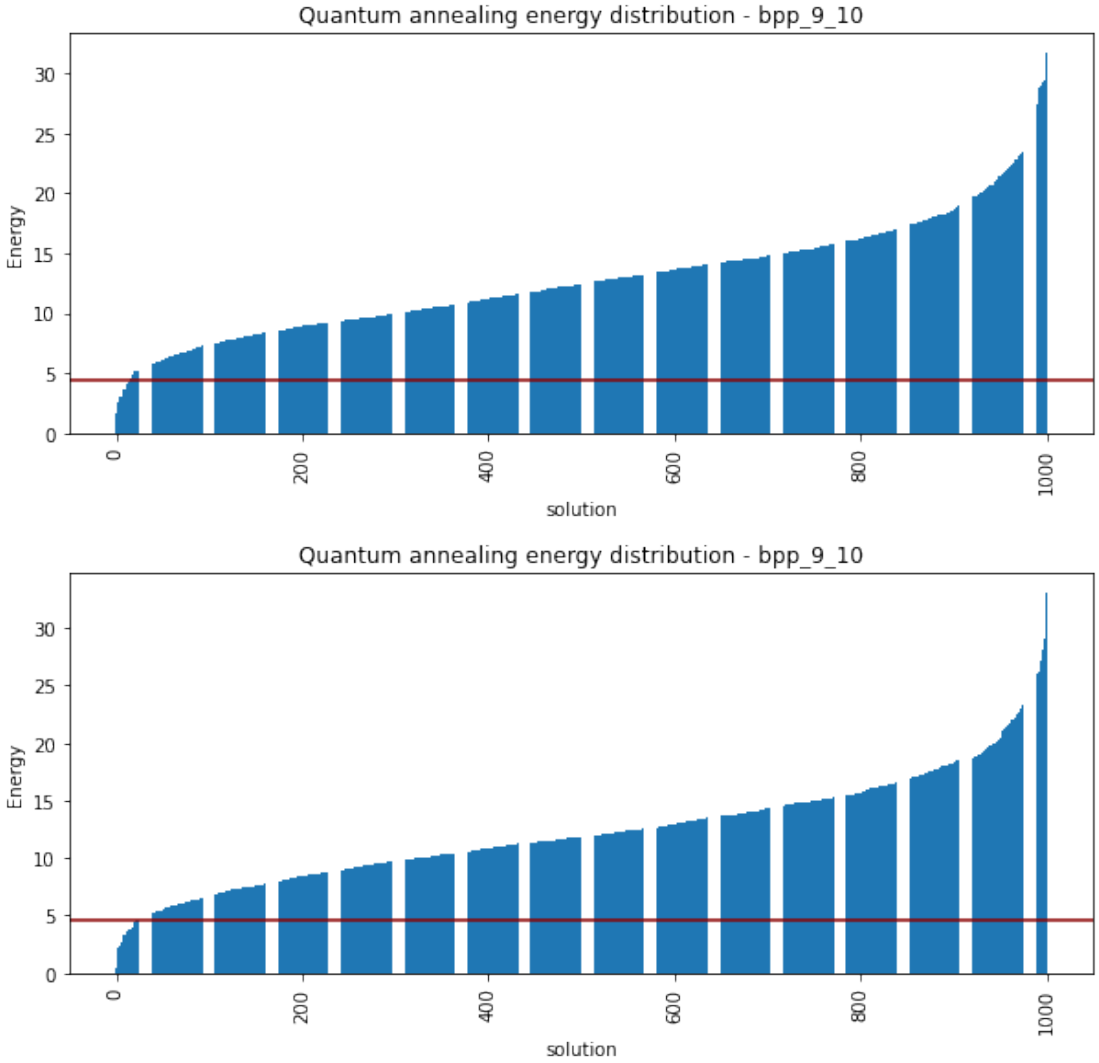

Figure 5.7: Energy distribution for two different instances of 9 items. The red horizontal line represents the first feasible solution in the sample set. In fig.5.6 we see that this energy level is always greater then the true minimum found by the simulated annealing.

6 Conclusion

In this thesis, we proposed a novel heuristic QUBO formulation, based on the augmented Lagrangian method, for solving the bin packing problem using quantum annealing. Moreover, we proposed an analytical estimation of model penalty terms that allows to avoid to rely on a recursive approximation method, such as ADMM, thus making the proposed approach more generalizable to different input instances and to other combinatorial problems.
We demonstrated that our approach can solve larger problem instances than any previous QUBO formulation for bin packing and can outperform state-of-the-art classical solvers, in terms of time to solution, when solved by a quantum annealer.
Furthermore, this work represents the first attempt to solve the bin packing problem, end to end, by a real quantum computer through quantum annealing: indeed, as previously discussed, other experimental attempts in literature addressed the bin packing problem by simulation (QAOA) or using hybrid search strategy, like the one proposed in [2].

Moreover, this thesis represents the first performace comparison, for the bin packing problem, between the classical state-of-the-art (Gurobi) and the quantum state-of-the-art (D-Wave *Advantage_system4.1*).

The proposed method can, in principle, be extended to other combinatorial optimization problems that involve inequality constraints, in order to formulate them as QUBO problems, enabling the application of quantum computing to a wider range of problems.

However, there are still some limitations and challenges that need to be addressed in future work. First of all, the generalizability of our model, to a generic BPP instance or, to other COP, is still to be proven. Indeed, our model works on the classes of instances presented in this thesis, but we have not yet demonstrated its generalizability to any BPP instance.

Another challenge is the limited number of qubits on current quantum annealers, which restricts the size of problem instances that can be solved. This, prevented us from testing our model on bigger instances, and so, from assessing its scalability on a wider range of input examples.

Another one is the presence of noise and errors in the quantum annealer, which can affect the quality of the solution. This effect becomes stronger when dealing with large problems or sparse QUBO matrices, as shown also by our experimental results when compared to simulated annealing.

To overcome these challenges, future work can investigate the use of more advanced quantum hardware, which can provide higher accuracy and higher number of qubits.
Another direction for future work is to explore hybrid approaches that combine classical and quantum methods, in order to solve problems of size otherwise not addressable by modern QPUs.
Other work that must to be done is to theorethically investigate our model in order to prove whether it is applicable to all kind of bin packing instances or only to some classes of it.
Another direction of reasearch is to study the generalizability of the proposed approach to other combinatorial optimization problems.

Besides all the mentioned ones, of great importance would be to test our model using QAOA, in order to check if we reached the so called *quantum advantage*. This term is used when a quantum computer can perform a particular computation significantly faster than even the best classical computer.

In conclusion, in this thesis we presented a novel heuristic QUBO formulation for some classes of instances of the bin packing problem and we solved it using a real quantum computer through quantum annealing. We showed its astonishing performace with respect the classical state-of-the-art algorithms on small instances, meaning that this performance gap would increase for bigger problems. This work paves the way to the research of finding generalizable approaches for QUBO models, and at the same time, to the establishment of a new quantum advantage record.

Bibliography

[1] Scott Aaronson and Chris Bourke. “The Complexity Zoo”. In: (Jan. 2008).

[2] Mikel Garcia-de Andoin, Izaskun Oregi, Esther Villar-Rodriguez, Eneko Osaba, and Mikel Sanz. *Comparative Benchmark of a Quantum Algorithm for the Bin Packing Problem*. 2022. DOI: 10.48550/ARXIV.2207.07460. URL: https://arxiv.org/abs/2207.07460.

[3] Giorgio Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999. ISBN: 3540654313.

[4] Giorgio Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. 1st. Berlin, Heidelberg: Springer-Verlag, 1999. ISBN: 3540654313.

[5] Ruibin Bai, Jacek Blazewicz, Edmund Burke, Graham Kendall, and Barry Mccollum. “B.: A simulated annealing hyper-heuristic methodology for flexible decision support”. In: *4OR* 10 (Mar. 2012). DOI: 10.1007/s10288-011-0182-8.

[6] B. S. Baker and E. G. Coffman Jr. “A Tight Asymptotic Bound for Next-Fit-Decreasing Bin-Packing”. In: *SIAM Journal on Algebraic Discrete Methods* 2.2 (1981), pp. 147–152. DOI: 10.1137/0602019. eprint: https://doi.org/10.1137/0602019. URL: https://doi.org/10.1137/0602019.

[7] F Barahona. “On the computational complexity of Ising spin glass models”. In: *Journal of Physics A: Mathematical and General* 15.10 (1982), p. 3241. DOI: 10.1088/0305-4470/15/10/028. URL: https://dx.doi.org/10.1088/0305-4470/15/10/028.

[8] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. "Quantum machine learning". In: *Nature* 549.7671 (2017), pp. 195–202. DOI: 10.1038/nature23474. URL: https://doi.org/10.1038%2Fnature23474.

[9] Zhengbing Bian, Fabián A. Chudak, William G. Macready, and Geordie Rose. "The Ising model : teaching an old problem new tricks". In: 2010.

[10] Bin Packing. *Bin Packing Problem — Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/wiki/Bin_packing_problem.

[11] Endre Boros and Peter L. Hammer. "Pseudo-Boolean optimization". In: *Discrete Applied Mathematics* 123.1 (2002), pp. 155–225. ISSN: 0166-218X. DOI: https://doi.org/10.1016/S0166-218X(01)00341-9. URL: https://www.sciencedirect.com/science/article/pii/S0166218X01003419.

[12] Edmund Burke, Matthew Hyde, and Graham Kendall. "Evolving Bin Packing Heuristics with Genetic Programming". In: Jan. 2006, pp. 860–869. ISBN: 978-3-540-38990-3. DOI: 10.1007/11844297_87.

[13] Hadrien Cambazard and Barry O'Sullivan. "Propagating the Bin Packing Constraint Using Linear Programming". In: vol. 6308. Sept. 2010, pp. 129–136. ISBN: 978-3-642-15395-2. DOI: 10.1007/978-3-642-15396-9_13.

[14] Josè M. Valèrio de Carvalho. "Using Extra Dual Cuts to Accelerate Column Generation". In: *INFORMS Journal on Computing* 17.2 (2005), pp. 175–182. DOI: 10.1287/ijoc.1030.0060. eprint: https://doi.org/10.1287/ijoc.1030.0060. URL: https://doi.org/10.1287/ijoc.1030.0060.

[15] Josè M. Valèrio de Carvalho. "Exact solution of bin-packing problems using column generation and branch-and-bound". In: *Ann. Oper. Res.* 86 (1999), pp. 629–659.

[16] Edward G. Coffman Jr., János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. "Bin Packing Approximation Algorithms: Survey and Classification". In: *Handbook of Combinatorial Optimization*. Ed. by Panos M. Pardalos, Ding-Zhu Du, and Ronald L. Graham. New York, NY: Springer New York, 2013, pp. 455–531. ISBN: 978-1-4419-7997-1. DOI: 10.1007/978-1-4419-7997-1_35. URL: https://doi.org/10.1007/978-1-4419-7997-1_35.

[17] J. Csirik, G. Galambos, J.B.G. Frenk, A.M. Frieze, and A.H.G. Rinnooy Kan. "A probabilistic analysis of the next fit decreasing bin packing heuristic". In: *Operations Research Letters* 5.5 (1986), pp. 233–236. ISSN: 0167-6377. DOI: https://doi.org/10.1016/0167-6377(86)90013-1. URL: https://www.sciencedirect.com/science/article/pii/0167637786900131.

[18] Arnab Das and Bikas K. Chakrabarti. "Quantum annealing and analog quantum computation". In: *Reviews of Modern Physics* 80.3 (Sept. 2008), pp. 1061–1081. DOI: 10.1103/revmodphys.80.1061. URL: https://doi.org/10.1103%2Frevmodphys.80.1061.

[19] Maxence Delorme, Manuel Iori, and Silvano Martello. "Bin packing and cutting stock problems: Mathematical models and exact algorithms". In: *European Journal of Operational Research* 255.1 (2016), pp. 1–20. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2016.04.030. URL: https://www.sciencedirect.com/science/article/pii/S0377221716302491.

[20] Maxence Delorme, Manuel Iori, and Silvano Martello. "BPPLIB: a library for bin packing and cutting stock problems". In: *Optimization Letters* 12 (Mar. 2018), pp. 1–16. DOI: 10.1007/s11590-017-1192-z.

[21] György Dósa. In: *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Ed. by Bo Chen, Mike Paterson, and Guochuan Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–11. ISBN: 978-3-540-74450-4.

[22] Harald Dyckhoff. "A New Linear Programming Approach to the Cutting Stock Problem". In: *Operations Research* 29.6 (1981), pp. 1092–1104. DOI: 10.1287/opre.29.6.1092. eprint: https://doi.org/10.1287/opre.29.6.1092. URL: https://doi.org/10.1287/opre.29.6.1092.

[23] José Carlos Díaz Díaz, Mauro Dell'Amico, Manuel Iori, Romeo Rizzi, and Alberto Caprara. "Friendly Bin Packing Instances without Integer Round-up Property". In: *Mathematical Programming* 150 (June 2014), pp. 1–13. DOI: 10.1007/s10107-014-0791-z.

[24] Samuel Eilon and Nicos Christofides. “The Loading Problem”. In: *Management Science* 17.5 (1971), pp. 259–268. DOI: 10.1287/mnsc.17.5.259. eprint: https://doi.org/10.1287/mnsc.17.5.259. URL: https://doi.org/10.1287/mnsc.17.5.259.

[25] Kurt. Eisemann. “The Trim Problem”. In: *Management Science* 3 (1957), pp. 279–284.

[26] Diego de Falco and Dario Tamascelli. “An introduction to quantum annealing”. In: *RAIRO - Theoretical Informatics and Applications* 45.1 (Jan. 2011), pp. 99–116. DOI: 10.1051/ita/2011013. URL: https://doi.org/10.1051%2Fita%2F2011013.

[27] E. Falkenauer and A. Delchambre. “A genetic algorithm for bin packing and line balancing”. In: *Proceedings 1992 IEEE International Conference on Robotics and Automation*. 1992, 1186–1192 vol.2. DOI: 10.1109/ROBOT.1992.220088.

[28] Emanuel Falkenauer. “A hybrid grouping genetic algorithm for bin packing”. In: *Journal of Heuristics* 2 (1996). DOI: 10.1007/BF00226291. URL: https://doi.org/10.1007/BF00226291.

[29] David C Fisher. “Next-fit packs a list and its reverse into the same number of bins”. In: *Operations Research Letters* 7.6 (1988), pp. 291–293. ISSN: 0167-6377. DOI: https://doi.org/10.1016/0167-6377(88)90060-0. URL: https://www.sciencedirect.com/science/article/pii/0167637788900600.

[30] Diana Franklin and Frederic Chong. “Challenges in Reliable Quantum Computing”. In: Jan. 2004, pp. 247–266. ISBN: 1-4020-8067-0. DOI: 10.1007/1-4020-8068-9_8.

[31] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co Ltd, 1979. ISBN: 9780716710455.

[32] R Gilmore and Ralph Gomory. “A Linear Programming Approach to the Cutting Stock Problem I”. In: *Oper Res* 9 (Jan. 1961). DOI: 10.1287/opre.9.6.849.

[33] Fred Glover, Gary Kochenberger, and Yu Du. *A Tutorial on Formulating and Using QUBO Models*. 2018. DOI: 10.48550/ARXIV.1811.11538. URL: https://arxiv.org/abs/1811.11538.

[34] Oded Goldreich. *P, NP, and NP-Completeness: The Basics of Computational Complexity*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511761355.

[35] Pedro Gomez-Meneses and Marcus Randall. "A Hybrid Extremal Optimisation Approach for the Bin Packing Problem". In: vol. 5865. Dec. 2009. ISBN: 978-3-642-10426-8. DOI: 10.1007/978-3-642-10427-5_24.

[36] Jatinder N. D. Gupta and Johnny C. Ho. "A new heuristic algorithm for the one-dimensional bin-packing problem". In: *Production Planning & Control* 10.6 (1999), pp. 598–603. DOI: 10.1080/095372899232894. eprint: https://doi.org/10.1080/095372899232894. URL: https://doi.org/10.1080/095372899232894.

[37] Darrall Henderson, Sheldon H. Jacobson, and Alan W. Johnson. "The Theory and Practice of Simulated Annealing". In: *Handbook of Metaheuristics*. Ed. by Fred Glover and Gary A. Kochenberger. Boston, MA: Springer US, 2003, pp. 287–319. ISBN: 978-0-306-48056-0. DOI: 10.1007/0-306-48056-5_10. URL: https://doi.org/10.1007/0-306-48056-5_10.

[38] Tony Hey. "Quantum computing: An introduction". In: *Computing & Control Engineering Journal* 10 (Sept. 1999), pp. 105 –112. DOI: 10.1049/ccej:19990303.

[39] Jack Hidary. *Quantum Computing: An Applied Approach*. Jan. 2019. ISBN: 978-3-030-23921-3. DOI: 10.1007/978-3-030-23922-0.

[40] D-Wave System Inc. *What is Quantum Annealing?* https://docs.dwavesys.com/docs/latest/c_gs_2.html.

[41] Ising model. *Ising model — Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/wiki/Ising_model.

[42] T. Kampke. "Simulated Annealing: Use of New Tool in Bin Packing". In: *Ann. Oper. Res.* 16.1-4 (Jan. 1988), pp. 327–332. ISSN: 0254-5330. DOI: 10.1007/BF02283751. URL: https://doi.org/10.1007/BF02283751.

[43] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671. eprint: https://www.science.org/doi/pdf/10.1126/science.220.4598.671. URL: https://www.science.org/doi/abs/10.1126/science.220.4598.671.

[44] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. "The unconstrained binary quadratic programming problem: A survey". In: *Journal of Combinatorial Optimization* 28 (July 2014). DOI: 10.1007/s10878-014-9734-0.

[45] Michiya Kuramata, Ryota Katsuki, and Kazuhide Nakata. "Larger Sparse Quadratic Assignment Problem Optimization Using Quantum Annealing and a Bit-Flip Heuristic Algorithm". In: *2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA)*. IEEE, 2021. DOI: 10.1109/iciea52957.2021.9436749. URL: https://doi.org/10.1109%2Ficiea52957.2021.9436749.

[46] Rhydian Lewis. "A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing". In: *Computers & Operations Research* 36 (July 2009), pp. 2295–2310. DOI: 10.1016/j.cor.2008.09.004.

[47] Bas Lodewijks. *Mapping NP-hard and NP-complete optimisation problems to Quadratic Unconstrained Binary Optimisation problems*. 2019. DOI: 10.48550/ARXIV.1911.08043. URL: https://arxiv.org/abs/1911.08043.

[48] Kok-Hua Loh, Bruce Golden, and Edward Wasil. "Solving the one-dimensional bin packing problem with a weight annealing heuristic". In: *Computers& Operations Research* 35.7 (2008). Part Special Issue: Includes selected papers presented at the ECCO'04 European Conference on combinatorial Optimization, pp. 2283–2291. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2006.10.021. URL: https://www.sciencedirect.com/science/article/pii/S0305054806002826.

[49] Eunice Lopez-Camacho, Hugo Terashima-Marin, and Peter Ross. "A hyper-heuristic for solving one and two-dimensional bin packing problems". In: *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation* (2011).

[50] Andrew Lucas. "Ising formulations of many NP problems". In: *Frontiers in Physics* 2 (2014). DOI: 10.3389/fphy.2014.00005. URL: https://doi.org/10.3389%2Ffphy.2014.00005.

[51] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley and sons, 1990. ISBN: 0471924202.

[52] S. Mertens. "Computational complexity for physicists". In: *Computing in Science & Engineering* 4.3 (2002), pp. 31–47. DOI: 10.1109/5992.998639.

[53] Alejandro Montanez-Barrera, Dennis Willsch, Alberto Maldonado-Romo, and Kristel Michielsen. *Unbalanced penalization: A new approach to encode inequality constraints of combinatorial problems for quantum optimization algorithms*. 2022. DOI: 10.48550/ARXIV.2211.13914. URL: https://arxiv.org/abs/2211.13914.

[54] David Morin. "Chapter 10 Introduction to quantum mechanics". In: 2010.

[55] David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning". In: *Discrete Optimization* 19 (2016), pp. 79–102. ISSN: 1572-5286. DOI: https://doi.org/10.1016/j.disopt.2016.01.005. URL: https://www.sciencedirect.com/science/article/pii/S1572528616000062.

[56] M.R.Rao. "On the cutting stock problem". In: *Journal of the Computer Society of India* (1976).

[57] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011. ISBN: 9781107002173. URL: https://www.amazon.com/Quantum-Computation-Information-10th-Anniversary/dp/1107002176?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1107002176.

[58] Christoph Nitsche, Guntram Scheithauer, and Johannes Terno. "Tighter relaxations for the cutting stock problem". In: *European Journal of Operational Research* 112.3 (1999), pp. 654–663. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(97)00404-9. URL: https://www.sciencedirect.com/science/article/pii/S0377221797004049.

[59] Frank Phillipson and Irina Chiscop. *Multimodal Container Planning: a QUBO Formulation and Implementation on a Quantum Annealer*. 2020. DOI: 10.48550/ARXIV.2007.01730. URL: https://arxiv.org/abs/2007.01730.

[60] Marcela Quiroz, Laura Cruz-Reyes, Jose Torres-Jimenez, Claudia Santillán, Héctor Fraire-Huacuja, and Adriana Alvim. "A grouping genetic algorithm with controlled gene transmission for the bin packing problem". In: *Computers & Operations Research* 55 (Oct. 2014), pp. 52–64. DOI: 10.1016/j.cor.2014.10.010.

[61] R.L. Rao and S.S. Iyengar. "Bin-packing by simulated annealing". In: *Computers & Mathematics with Applications* 27.5 (1994), pp. 71–82. ISSN: 0898-1221. DOI: https://doi.org/10.1016/0898-1221(94)90077-9. URL: https://www.sciencedirect.com/science/article/pii/0898122194900779.

[62] Luis F. Zuluaga Rodolfo A. Quintero. *Characterizing and Benchmarking QUBO Reformulations of the Knapsack Problem*. 2020. URL: https://engineering.lehigh.edu/sites/engineering.lehigh.edu/files/_DEPARTMENTS/ise/pdf/tech-papers/21/21T_028.pdf.

[63] J. J. Sakurai. *Modern Quantum Mechanics (Revised Edition)*. 1st ed. Addison Wesley, Sept. 1993. ISBN: 0201539292. URL: http://www.worldcat.org/isbn/0201539292.

[64] Armin Scholl, Robert Klein, and Christian Jürgens. "BISON: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem". English. In: *Computers and Operations Research* 24.7 (1997), pp. 627–645.

[65] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. "An introduction to quantum machine learning". In: *Contemporary Physics* 56.2 (2014), pp. 172–185. DOI: 10.1080/00107514.2014.964942. URL: https://doi.org/10.1080%2F00107514.2014.964942.

[66] Kevin Sim and Emma Hart. "Generating Single and Multiple Cooperative Heuristics for the One Dimensional Bin Packing Problem Using a Single Node Genetic Programming Island Model". In: July 2013. DOI: 10.1145/2463372.2463555.

[67] Kevin Sim, Emma Hart, and Ben Paechter. "A Hyper-Heuristic Classifier for One Dimensional Bin Packing Problems: Improving Classification Accuracy by Attribute Evolution". In: *Parallel Problem Solving from Nature - PPSN XII*. Ed. by Carlos A. Coello Coello, Vincenzo Cutello, Kalyanmoy Deb, Stephanie Forrest, Giuseppe Nicosia, and Mario Pavone. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 348–357. ISBN: 978-3-642-32964-7.

[68] Alejandro Sosa, Hugo Terashima-Marín, José Carlos Ortiz-Bayliss, and Santiago Conant-Pablos. "Grammar-based Selection Hyper-heuristics for Solving Irregular Bin Packing Problems:" in: July 2016, pp. 111–112. DOI: 10.1145/2908961.2908970.

[69] Hartmut Stadtler. "A comparison of two optimization procedures for 1- and 1 1/2-dimensional cutting stock problems". In: *OR Spektrum* 10 (June 1988), pp. 97–111. DOI: 10.1007/BF01720208.

[70] Luka Tadic, Petar Afric, Lucija Sikic, Adrian Kurdija, Vladimir Klemo, Goran Delac, and Marin Silic. "Analysis and Comparison of Exact and Approximate Bin Packing Algorithms". In: May 2019, pp. 919–924. DOI: 10.23919/MIPRO.2019.8757047.

[71] Richard Vahrenkamp. "Random search in the one-dimensional cutting stock problem". In: *European Journal of Operational Research* 95.1 (1996), pp. 191–200. ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(95)00198-0. URL: https://www.sciencedirect.com/science/article/pii/0377221795001980.

[72] Salvador E. Venegas-Andraca, William Cruz-Santos, Catherine McGeoch, and Marco Lanzagorta. "A cross-disciplinary introduction to quantum annealing-based algorithms". In: *Contemporary Physics* 59.2 (Apr. 2018), pp. 174–197. DOI: 10.1080/00107514.2018.1450720. URL: https://doi.org/10.1080%2F00107514.2018.1450720.

[73] Supreeth Mysore Venkatesh, Antonio Macaluso, and Matthias Klusch. "BILP-Q". In: *Proceedings of the 19th ACM International Conference on Computing Frontiers*. ACM, May 2022. DOI: 10.1145/3528416.3530235. URL: https://doi.org/10.1145%2F3528416.3530235.

[74] Supreeth Mysore Venkatesh, Antonio Macaluso, and Matthias Klusch. *GCS-Q: Quantum Graph Coalition Structure Generation*. 2022. DOI: 10.48550/ARXIV.2212.11372. URL: https://arxiv.org/abs/2212.11372.

[75] Tomáš Vyskočil, Scott Pakin, and Hristo Djidjev. "Embedding Inequality Constraints for Quantum Annealing Optimization: First International Workshop, QTOP

2019, Munich, Germany, March 18, 2019, Proceedings". In: Jan. 2019, pp. 11–22. ISBN: 978-3-030-14081-6. DOI: 10.1007/978-3-030-14082-3_2.

[76] Fei Yan, Abdullah Iliyasu, and Salvador Venegas-Andraca. "A survey of quantum image representations". In: *Quantum Information Processing* 15 (Jan. 2016). DOI: 10.1007/s11128-015-1195-6.

[77] Kouki Yonaga, Masamichi J. Miyama, and Masayuki Ohzeki. *Solving Inequality-Constrained Binary Optimization Problems on Quantum Annealer*. 2020. DOI: 10.48550/ARXIV.2012.06119. URL: https://arxiv.org/abs/2012.06119.