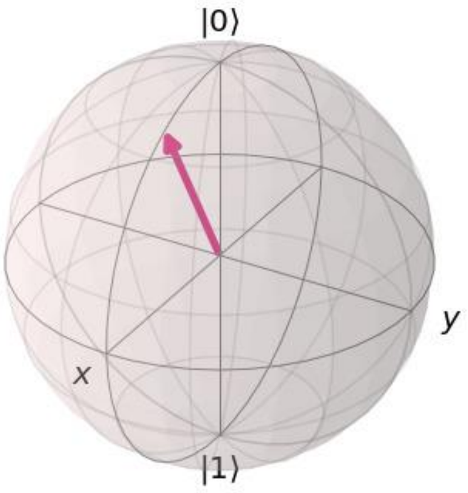


Quantum Machine Intelligence: Mapping AI Applications

Carla Maria Alves Pereira da Silva

Doutoramento em Ciência de Computadores
Departamento de Ciência de Computadores
2021

Orientador
Inês de Castro Dutra, Professor Auxiliar
Faculdade de Ciências da Universidade do Porto

Coorientador
Ana Cristina Costa Aguiar, Professor Auxiliar
Faculdade de Engenharia da Universidade do Porto

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
2021

Abstract

Computer chips are composed by large amounts of transistors that flip between states, on and off, to build binary digits. These traditional computers based on bits, store all data as 1s or 0s. In the meantime, quantum computers based on qubits have appeared, which are similar to standard bits but where each qubit can be 1 and 0 simultaneously, according to principles of quantum mechanics (QM).

Noisy intermediate-scale quantum (NISQ) devices surface, along with other quantum technologies, and have been used to explore many applications with the promise of reducing computational complexity. In this new computational era, promising applications of quantum computing are challenges from artificial intelligence (AI), e.g. machine learning (ML), where the amount of computing resources needed for particular tasks can be a drawback. From this viewpoint, quantum machine learning (QML) arises as a research field that relates the interdisciplinary topics quantum mechanics and machine learning which seeks to incorporate data and information processing in a quantum perspective.

Currently, there exists some available model directions of quantum computing (QC): (a) the gate-based qubits universal quantum processors, e.g., IBM's, (b) the hybrid classical quantum model, e.g., Xanadu's and (c) the adiabatic-model quantum accelerated annealing, e.g., D-Wave's. Those models are currently stretching the bridge between gate-based model quantum computing and the adiabatic paradigm. Various research works have been exploring these models to implement QML algorithms.

The directions and prevailing QC and QML frameworks are designed so that applications can be mapped from classical to quantum formats. However, mapping applications, especially ML, to quantum devices remains a very complex task. Some efforts have been made towards solving this problem, but no general solution exists. This research contributes in this field by proposing application mappings based on variational circuits and on propositional logic. Our solution covers both gate-based and adiabatic-based model applications.

In a gate-based perspective, we use a hybrid scheme inspired in the existing PennyLane's variational circuits and embedding templates applied to datasets (e.g. Iris plant, traffic, breast cancer). In an adiabatic scenario, we propose a novel mapping methodology that maps well-known problems (e.g. graph coloring, travelling salesperson problem) to quantum annealing, which we call quantum propositional logic (QPL). This methodology provides a method to solve combinatorial optimization problems in complex energy landscapes by exploiting thermal fluctuations. The steps for mapping each problem are based on pseudo-Boolean constraints starting from the problem formulated as a set of constraints represented in propositional logic, and later converted to a quadratic unconstrained binary optimization problem (QUBO). This mapping is not limited to QUBO, because the formulation and mapping can also be translated into a generic artificial neural network, allowing to find solutions for a combination of multiple propositional logic problems.

Resumo

Os chips de computador são compostos por grandes quantidades de transistores que alternam entre os estados, *on* e *off*, para construir dígitos binários. Esses computadores tradicionais baseados em *bits* armazenam todos os dados como 1s ou 0s. Recentemente, surgiram os computadores quânticos baseados em *qubits*, que são semelhantes aos baseados em *bits*, mas onde cada *qubit* pode ser 1 e 0 simultaneamente, de acordo com os princípios da mecânica quântica (MQ).

Dispositivos *noisy intermediate-scale quantum* (NISQ) surgiram, a par de outras tecnologias quânticas, e têm sido usados para explorar muitas aplicações com a promessa de reduzir a complexidade computacional. Nesta nova era computacional, aplicações promissoras de computação quântica são desafios da inteligência artificial (IA), por exemplo, aprendizagem automática (AA), onde a quantidade necessária de recursos computacionais para determinadas tarefas podem ser insuficientes. Deste ponto de vista, aprendizagem automática quântica (AAQ) surge como um campo de investigação que interliga os tópicos, mecânica quântica e aprendizagem automática que visa unir dados e processamento de informação numa perspectiva quântica.

Atualmente, existem algumas direções de modelos para computação quântica (CQ): (a) processadores quânticos universais de *qubits* baseados em circuitos, por exemplo, IBM's, (b) modelo clássico quântico híbrido, por exemplo Xanadu e (c) modelo adiabático, por exemplo D-Wave. Esses modelos fazem a ponte entre a CQ do modelo de circuitos e o paradigma adiabático. Vários trabalhos de investigação têm explorado esses modelos para implementar algoritmos AAQ.

As direções e plataformas de CQ e AAQ predominantes são desenhadas para que aplicações que possam ser mapeadas de formatos clássicos para quânticos. No entanto, o mapeamento de aplicações, especialmente de aplicações de AA, para dispositivos quânticos continua a ser uma tarefa muito complexa. Alguns esforços foram feitos para resolver este problema, mas não existe uma solução geral. Esta pesquisa contribui neste campo ao propor mapeamento de aplicações baseadas em circuitos variacionais e na lógica proposicional. A nossa solução cobre aplicações de modelos baseados em circuitos e adiabáticos.

Primeiro, numa perspetiva de circuitos, consideramos um esquema híbrido inspirado nos circuitos variacionais de PennyLane com aplicação a conjuntos de dados (p. e. planta Iris, tráfego, cancro da mama). Num cenário adiabático, é proposta uma metodologia de mapeamento que mapeia problemas bem conhecidos (p. e. coloração de grafos, problema do caixeiro viajante), a que chamamos lógica proposicional quântica (LPQ). Esta metodologia sugere um método para resolver problemas de otimização combinatória em paisagens de energia complexas, explorando flutuações térmicas. As etapas de mapeamento de cada problema são baseadas em restrições pseudo-Booleanas a partir da sua formulação como um conjunto de restrições representadas em lógica proposicional, e posteriormente convertidas num problema de *quadratic unconstrained binary optimization* (QUBO). Este mapeamento não está limitado ao QUBO, uma vez que a formulação e o mapeamento também pode ser traduzido numa rede neuronal artificial, permitindo encontrar soluções para uma combinação de múltiplos problemas de lógica proposicional.

Acknowledgements

I would like to gratefully acknowledge all the people that have directly or indirectly contributed to the achievement of this thesis. Thank you!

First and foremost, I would like to express my gratitude to my supervisors, Prof. Inês Dutra and Prof. Ana Aguiar, for providing me with guidance and support in my professional development, for the stimulating talks and by the continuous opening to the topic, mentorship, collaboration, and follow-up of the proposed work. Also, because of their acceptance to enter with me in this challenge which embraces the world dynamics around quantum computing, full of new discoveries and achievements, uncovering new paradigms for many of us. This thesis was only possible given their broad knowledge, sharp reasoning, analytical skills and exceptional ability to work.

I also wish to thank Prof. Alessandra Di Pierro, Prof. Frank K. Wilhelm-Mauch, and my internal reviewers, for agreeing to be members of my defense committee and for generously offering their time, and patience throughout the review of this document, in the hope that this document may help and be of great interest to the community.

Quantum machine intelligence, in particular, quantum machine learning came to me through the book "Quantum Machine Learning: What Quantum Computing Means to Data Mining" by Prof. Peter Wittek (in memoriam). His work inspired me and made me curious to learn more and deepen the topic. I would like to thank for his acceptance to be the external jury of my pre-thesis defense. His valuable comments concerning variational circuits and how a bridge between gate-model quantum computing and the adiabatic paradigm are made, boosted my work in progress.

I would like to thank Prof. Marcus S. Dahlem, for the acceptance to be part of the beginning of this challenge and for the talks on the foundations of quantum mechanics.

In 2018, I participated in sessions related to the architecture of the IBM Q Experience platform, giving my feedback to IBM as a user, with the aim of making improvements to the IBM Q Experience platform for all users. From these sessions I met Anna Obikane, whom I wish to thank for the direct assistance, where we exchanged experiences concerning my beginning steps using the IBM Q Experience platform, it was very enriching. Also, for connecting me with Brian Ingmanson that in turn introduced me to other IBMers: Robert Loredo, Amira Abbas and Pooja Mistry, who participated in the virtual school organized by me and Prof. Inês Dutra, Prof. Miguel Areias and Prof. Carlos Ferreira, entitled "Summer School on Machine Learning and Big Data with Quantum Computing" (SMBQ 2020), along with participation of Prof. João Gama, Prof. Jaime Cardoso, Prof. Albert Bifet, Prof. Luís Paulo Santos and Dr. Maria Schuld.

I also wish to thank QuantaLab and Prof. Luís Barbosa for managing my access to devices with more qubits from IBM Q Experience through the University of Minho Academic Q Hub.

I would like to thank Prof. Priscila V. M. Lima for the discussions and energizing talks concerning the adiabatic model in quantum computing and contributions to my work in quantum propositional logic.

Thank you also to the students Diogo Fernandes and Vanda Azevedo for our joint collaboration on quantum challenges.

Thank you to Joel Gottlieb and Victoria Goliber from D-Wave Systems for the help in clarifying my doubts regarding the Quantum Processing Unit (QPU) from D-Wave's.

Finally, but not least, I would like to express my infinite gratitude to the encouragement and support given by my family and friends, with a special thanks to my husband Pedro Araújo and his valuable contribution, with enriching exchanges of ideas in the part of propositional logic.

I'm really grateful.

Carla Silva

Contributions

- Carla Silva, Marcus Dahlem, and Inês Dutra. Simulating markov transition probabilities in a quantum environment. YQIS-17: 3rd International Conference for Young Quantum Information Scientists, Friedrich-Alexander-Universität Erlangen-Nürnberg, Max Planck Institute for the Science of Light, Erlangen, Germany, 3-6 October 2017 *(poster)*
- Carla Silva. What is quantum AI? *ITNOW*, 59(4):21, December 2017 *(article)*
- Carla Silva, Marcus Dahlem, and Inês Dutra. Driven tabu search: a quantum inherent optimisation. 24th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, London, UK, August 2018 *(abstract/talk)*
- Carla Silva, Inês Dutra, and Marcus Dahlem. Driven tabu search: a quantum inherent optimisation. *ArXiv e-prints*, August 2018 *(article)*
- Carla Silva, Vanda Azevedo, Diogo Fernandes, and Inês Dutra. Introduction to Quantum Computing with QISKit - a pratical guide. IBM Teach Me Quantum Award 2018: Top Submissions, 2018 *(mention as the best top submissions in the qiskit community [1])*
- Diogo Fernandes, Carla Silva, and Inês Dutra. Using Grover's search quantum algorithm to solve Boolean satisfiability problems, part 2. *XRDS: Crossroads, The ACM Magazine for Students* 26 (2), 68-71, November 2019 *(article)*
- Diogo Fernandes, Carla Silva, and Inês Dutra. Erratum: Using Grover's search quantum algorithm to solve Boolean satisfiability problems, part 2. *XRDS: Crossroads, The ACM Magazine for Students* 26 (3), 57-57, April 2020 *(article correction)*
- Carla Silva, Carlos Ferreira, Inês Dutra, Miguel Areias. Summer School on Machine Learning and Big Data with Quantum Computing (SMBQ 2020), Porto, Portugal (Online), September 7-8, 2020 *(member of the event organization [2])*
- Carla Silva, Ana Aguiar, Priscila M. V. Lima, and Inês Dutra. Mapping graph coloring to quantum annealing. *Quantum Machine Intelligence* 2 (2), 1-19, November 2020 *(article)*

[1]https://github.com/qiskit-community/qiskit-community-tutorials/blob/master/awards/teach_me_quantum_2018/README.md

[2]https://smbq2020.dcc.fc.up.pt

- Carla Silva, Ana Aguiar, and Inês Dutra. Quantum Binary Classification (Student Abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, A Virtual Conference, 2-9 February 2021 *(abstract/poster)*
- Carla Silva. Exploring extrapolation for molecular dissociation profiles. Machine Learning for Quantum 2021, held online, 1-5 March 2021 *(abstract/poster)*
- Carla Silva, Ana Aguiar, Priscila M. V. Lima, and Inês Dutra. Mapping a logical representation of TSP to quantum annealing. *Quantum Information Processing* 2021 *(article)*
- Vanda Azevedo, Carla Silva, and Inês Dutra. Quantum transfer learning for breast cancer detection. *Quantum Machine Intelligence* 2021 *(article) (submitted to the Special Issue in Honour of Prof. Peter Wittek)*

To my family.

Contents

List of Tables

List of Figures

Listings

Acronyms

QPL	Quantum Propositional Logic
QC	Quantum Computing
QAI	Quantum Artificial Intelligence
QML	Quantum Machine Learning
ML	Machine Learning
AI	Artificial Intelligence
QNN	Quantum Neural Network
NISQ	Noisy Intermediate-Scale Quantum
QPU	Quantum Processing Unit
SA	Simulated Annealing
QA	Quantum Annealing
AQC	Adiabatic Quantum Computation
BQM	Binary Quadratic Model
QUBO	Quadratic Unconstrained Binary Optimization

Chapter 1

Introduction

Quantum Computing (QC) is in the Noisy Intermediate-Scale Quantum (NISQ) era [161], which means that today's quantum computers are not yet big enough or advanced to be fault-tolerant. The quantum computers that we have now are still limited in size and prone to noise and to errors. Here the intermediate-scale plays the role of the size of the available quantum devices which are currently capable to perform certain tasks. The noisy part, concerns the imperfect control over the qubits which can lead to small errors, that can be big enough in a long run to not give us the correct answer for a certain task. Although quantum computations currently face these imperfections, in 2019 Google claimed for quantum supremacy when its quantum computer, Sycamore, completed a calculation in just few minutes that its researchers estimated would have taken 10,000 years in a supercomputer [12]. In response to Google achievement, which concerns a quantum device that does something impractical for a conventional computer, IBM rebuttal argued that a simulation of the same task could be conducted on a classical system in 2.5 days and with greater fidelity [1]. In the meantime, China's leading quantum research group also claimed for quantum supremacy with the Jiuzhang system, which produced results in minutes to a problem that would take more than 2 billion years in the world's third most powerful supercomputer [226]. Both approaches have different basis, Google builds quantum circuits using superconducting materials, while China's group resorts to photons, different quantum technologies that are inspiring the world of computing research.

This difference of orders of magnitude of performing some computational task in a quantum computer and in a classical computer is very challenging. Currently, we have quantum computers, that are able to perform some computational tasks in a faster way than classical computers [18, 90]. So the natural question here is: is there an actual benefit of using a quantum computer? What are its limitations? What kinds of problems can we actually solve? How can we program a quantum computer? [67] What are the advantages of quantum computers over classical devices? What is the purpose of advances in QC? Are we looking for a new working model to solve problems that are not solved in a von Neumman's machine? The question is whether we can take advantage of these computers to make computations faster, or to reduce the kind of resources

[1]https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/

needed. For instance, by reducing the memory or decreasing the amount of data that we need to attain some models or some conclusions. So this is a very interesting question, what can these quantum computers do? This is a question that is hard to answer for many reasons. One reason is that we really don't fully understand what are the capabilities of quantum computers, at least in a practical sense, although in theory we have several different definitions, the first proposed by the physicist Richard Feynman in the 80's with the presentation of a basic model for a quantum computer [61].

Nowadays, QC comprises the intersection of many fields of knowledge, e.g. quantum mechanics, computer science, mathematics, statistics, engineering, where all play the most relevant roles to handle all technological issues and developments. As such, this quantum supremacy seems to show that quantum computers are more powerful in some sense, in some task. This ability was proven before in 1994 by Peter Shor [184] and in 1996 by Lov K. Grover [75]. In sum, if we solve those questions we probably would be able to solve long-standing problems regarding classical computational complexity. QC has been explored for many years. Currently, with the advance of quantum technology, it is possible to move from theoretical to practical experiences. The pursuit to build a quantum processor to take advantage of the quantum belief, has shown some of the first prototypes to emerge in the last decade. However, we still have some limitations and/or challenges to address from a hardware and software agnostic perspective:

(a) how to deal with interference and the current barriers in manufacturing, controlling and protecting quantum systems from noise [155];

(b) how to improve the compilation phase and the translation of logical instructions on physical devices which can lead to inefficient and high-latency programs [182];

(c) how to describe robustness as being capable of reporting the noise tolerance and resilience of a quantum algorithm [213];

(d) how to program a quantum computer having control of the physical devices and the difference from conventional programming [81];

(e) how to encode a problem to be solved by a quantum computer and different types of encoding (e.g. [44, 115, 125, 179, 214, 220]).

This research work concentrates on one of these limitations: mapping applications to quantum computers. We first show how to map applications to a gate-based quantum computer and then we move to the adiabatic model, where we propose a mapping strategy, which is based on the formulation of a problem as a propositional logic constraint satisfaction problem. We start by firstly showing work done in the field of Quantum Artificial Intelligence (QAI), in particular Quantum Machine Learning (QML), by selecting quantum algorithms that deal with datasets (Iris plant, traffic, breast cancer) to solve classification problems. Afterwards, we present our proposed generalized pipeline for the mapping of problems such as graph coloring or traveling salesperson problem (TSP). The pipeline is based in pseudo-Boolean constraints that can be mapped to a quantum annealer where we first formulate the problem as a set of constraints represented in propositional logic and then we convert the set of constraints to an energy minimization problem. Next, we transform the formulation to a Quadratic Unconstrained Binary Optimization (QUBO) and solve it, as shown in this thesis, resorting to the D-Wave's graph.

Moreover, QC [147] is an emerging technology that has as one of its challenges, accelerate the training of machine learning algorithms and, therefore, contribute to solve data related tasks and optimization problems. QML [18] has been a recent topic of research where questions have already been tackled on how quantum computers can be used for data-driven prediction and decision making. The symbiosis of machine learning and quantum information is made by a bridge of mathematical concepts which can in one hand, unlock the analyses of measurement data from quantum experiments, on the other hand, contribute to new machine learning algorithms. The recent advances in QC seems to indicate that machine learning is a suitable field of application. As machine learning usually represents a form of multivariate optimization which can be mapped e.g. to quantum annealing and adiabatic model quantum computing [175, 215].

Currently, QC has followed some different directions and the information is scattered, specially regarding models, and applications. Many models of quantum computation are available (e.g. the circuit-model, the adiabatic model, measurement-based model, quantum walk model, one clean qubit model, etc.). Although the view of this thesis aims to be quantum platform agnostic, the approach of this thesis is based in: (a) the IBM gate-based quantum processors and programming environment, (b) the Xanadu hybrid classical quantum model, and (c) the commercial D-Wave adiabatic model. We start with the Open Quantum Assembly Language (OpenQASM) [41] for representation of quantum instructions, launched as part of IBM's Quantum Information Software Kit (Qiskit) [2]. Followed by Xanadu's Strawberry Fields, the open-source software for photonic quantum computing, and PennyLane [15], a Python library for quantum machine learning, automatic differentiation, and optimization of hybrid quantum-classical computations which allows hybrid CPU-GPU-QPU computations. Finally, to handle the adiabatic-based quantum computing paradigm, we resort to the D-Wave's ocean software stack [2] which provides tools that can convert a problem to a form that is solvable on a quantum device.

1.1 Main Contribution

This thesis begins with a gate-based approach in which supervised machine learning problems are chosen. In particular classification, by first exploring small datasets like the Iris plant and a traffic dataset, and later, breast cancer data. The aim is to study the performance of current quantum devices in solving this type of problems. Afterwards, we explore the adiabatic model. The main contribution of this work is to show how to map well-known problems to quantum computers from a circuit model and adiabatic model perspective. In particular, this thesis provides a method for mapping combinatorial problems to quantum devices by the intersection between quantum

[2]https://docs.ocean.dwavesys.com/

computing and propositional logic via constraint satisfiability, in a novel approach using a pipeline capable of solving any optimization problem defined by logical constraints. In addition, in the scope of the NP problems, the motivation of this work is to (a) map well-known optimization problems to a quantum computing (e.g. graph coloring, traveling salesperson problem), (b) compare classical (simulated annealing) and quantum (quantum annealing) methods, and (c) explore optimal and approximate solutions. In brief, the contribution of this thesis can be summarized as follows:

- Two problems of supervised machine learning (classification)
  - Quantum binary classification
  - Quantum transfer learning for breast cancer detection
- Two problems of search and optimization
  - Mapping graph coloring to quantum annealing
  - Mapping a logical representation of TSP to quantum annealing

This thesis is organized as follows. The first four chapters give an introduction to quantum computing and its computational models, quantum machine learning, constraints, propositional logic and satisfiability. The remaining chapters contain our contributions as a collage of papers. We end this thesis by summarising our contributions and giving perspectives for future work.

1.2 Thesis Overview

Chapter 1: Introduction: The recent advances in quantum computing indicate that applications can take advantage of the new quantum available platforms and devices, based in the current new challenges around QC, this thesis, covers the topics:

(1) how to map problems to quantum computers; (2) show the mapping resorting to applications; (3) explain how recent advances in quantum technologies allow us to go further in certain computation tasks; (4) resort to hybrid computations (classical and quantum) and to the adiabatic paradigm to solve problems; (5) as example, we studied the Iris plant, traffic, breast cancer, graph coloring and traveling salesperson problem.

Chapter 2: Concepts and Terminology: In chapter 2, material on circuit model and adiabatic quantum computing is introduced, quantum machine learning and quantum propositional logic, as follows:

(1) concepts of quantum computing are introduced, such as, qubit, Bra-ket notation, quantum systems, reversible *vs* irreversible computation, multi-qubit systems, quantum algorithms and complexity, adiabatic concepts, adiabatic theorem and annealing processes; (2) definitions on quantum machine learning with topics, such as, classical machine learning and variational quantum concepts (e.g. variational circuits, quantum nodes, quantum

gradients, hybrid computation); (3) background of Satisfaction and Conjunctive Normal Form (CNF), pseudo-Boolean problems and the integration between them and the problem of Satisfiability and mapping Satisfiability in Linear Programming 0-1.

Chapter 3: Mapping Applications to Quantum Hardware: In chapter 3, we discuss various kinds of mappings, encodings or embeddings, highlighting relevant work in the field regarding mapping problems to quantum hardware, as follows: (1) from two perspectives, gate-based model mapping, namely, basis embedding, amplitude embedding, angle embedding and other forms of embedding; and also, adiabatic model mapping based on encoding using a mathematical formulation; (2) describes the thesis roadmap which led to a pipeline construction of a novelty contribution in the field; (3) describes a final generalized strategy to map classical problems to quantum hardware, the method for modeling optimization problems using Quantum Propositional Logic (QPL).

Chapter 4: Results and Discussion: In chapter 4, we briefly explain the target problem, summarise the type of model (gate-based or adiabatic), the type of hardware to conduct the experiments (simulator or real quantum device) and the main findings, for the challenges: (1) mapping quantum classification problems; (2) mapping problems to Quantum Annealing (QA).

Chapter 5: Quantum Binary Classification: In chapter 5 it is shown how to construct a quantum binary classifier. It will be first introduced the necessary background on quantum classification. This chapter concerns the topic of supervised binary classification with amplitude encoding and a quantum classifier, resorting to a gradient-descent optimiser, with applications to biology (Iris plant) and traffic. In brief, it is shown how to construct a gate-based model application.

Chapter 6: Quantum Transfer Learning for Breast Cancer Detection: In chapter 6 resorting to a gate-based approach and inspired on the work of Mari et al., we train a set of hybrid classical-quantum neural networks using Transfer Learning (TL). The goal was to solve the problem of classifying full-image mammograms into malignant and benign, provided by BCDR. While analyzing different performance metrics, specific charts were used to indicate the parts of the mammograms that were being targeted by the networks.

Chapter 7: Mapping Graph Coloring to Quantum Annealing: Chapter 7 introduces the mapping of a graph coloring problem based in pseudo-Boolean constraints to a working graph of the D-Wave's. We start from the problem formulated as a set of constraints represented in propositional logic. We use the SATyrus approach to transform this set of constraints to an energy minimization problem. We convert the formulation to a quadratic unconstrained binary optimization problem (QUBO), applying polynomial reduction when needed, and solve the problem using different approaches.

Chapter 8: Mapping a Logical Representation of TSP to Quantum Annealing: This chapter 8 presents the mapping of the traveling salesperson problem (TSP) based in pseudo-Boolean constraints to a working graph of the D-Wave's. We first formulate the problem as a set of constraints represented in propositional logic. With this regard, we resort to the SATyrus approach to convert a set of constraints to an energy minimization problem. Afterwards, we transform the formulation to a quadratic unconstrained binary optimization problem (QUBO) and solve the problem using different approaches.

Chapter 9: Conclusions and Future Work: Presents a summary highlighting key points of the thesis, a conclusion is given and the discussion of interesting remark, as well as, direction for future work.

Chapter 2

Concepts and terminology

In this chapter, are detailed the main concepts and terminology of each topic presented in this thesis: quantum computing from gate-based to adiabatic modelling, quantum machine learning and quantum propositional logic, as follows:

(a) concepts of quantum computing are introduced, such as, qubit, Bra-ket notation, quantum systems, reversible *vs* irreversible computation, multi-qubit systems, quantum algorithms and complexity, adiabatic concepts, adiabatic theorem and annealing processes;

(b) definitions on quantum machine learning with topics, such as, classical machine learning and variational quantum concepts (e.g. variational circuits, quantum nodes, quantum gradients, hybrid computation);

(c) background of Satisfaction and Conjunctive Normal Form (CNF), pseudo-Boolean problems and the integration between them and the problem of Satisfiability and mapping Satisfiability in Linear Programming 0-1.

2.1 Quantum Computing

A quantum device is a kind of hardware built on top of quantum mechanics processes and consists of transforming the input data following a unitary operation, uses gate operations and performs measurements and is described by a quantum circuit [138]. Instead of working with binary discrete digits (bits), it works with quantum bits, where 0 and 1 can overlap in time. The simplest possible quantum system (*2-state system*) can hold precisely one *bit* of information. However, the *qubit* (unit of quantum information) has two possible states $|0\rangle$ and $|1\rangle$ defined as a finite-dimensional quantum system that forms a computational basis – two basis states composed by two distinct quantum states that the *qubit* can be in physically. In a general form of expression, it represents a linear combination of x, y, a linear algebra concept denoting any expression of the form $ax + by$ where x, y are vectors and a, b are coefficients. In the context of quantum mechanics the vectors are a vector in the *Hilbert space*, an abstract vector space which holds the structure of an inner product that allows length and angle to be measured, and a quantum *superposition* that allows a physical system to be in more than one basic state simultaneously. This phenomena occurs when the underlying operations acting are linear combinations [61, 147].

In classical computer devices, we apply basic logical operators (NOT, NAND, XOR, AND, OR) on *bits* to develop complex operations. In a gate-based quantum device, consisting of a quantum network of quantum logic gates, we perform a fixed unitary operator on selected *qubits*. The unitary operator is a function $U : H \rightarrow H$ where H is a Hilbert Space (inner product space), such that $\langle U_x, U_y \rangle_H = \langle x, y \rangle_H$ for all vectors x and y in H. The calculations are performed resorting to logical quantum gates (e.g. Pauli-X, Pauli-Y, Pauli-Z, CNOT) and on the opposite of classical logic gates, quantum logic gates are reversible (e.g. the Toffoli gate).

The reversible computing concept begins with physicist Rolf Landauer of IBM where he argued the logically irreversible computational operations and the implications for the thermodynamic behavior of the device that performed the operations. According to Landauer's the most fundamental laws of physics are reversible. In that sense, for instance, if we know the complete knowledge of the state at some time of a closed system (meaning that no measurement is performed on the system), we should then be able to reverse the process and determine the system's exact state at any previous time, according to physics laws. As such, every time we overwrite a bit of information, the previous information does not have been physically destroyed, it is kept in the device's thermal environment [64].

The model of quantum computation is a system based in complex concepts, such as, light travelling, and is ruled by fundamental principles, such as, Schrodinger's wave equation and Heisenberg's uncertainty principle. Therefore, quantum computations have foundations characterised by the Schrödinger equation that can be described as follows, $i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H(t)|\psi(t)\rangle$ where i is the imaginary unit, $\hbar = \frac{h}{2\pi}$ is the reduced Planck constant, $|\psi(t)\rangle$ is a linear combination of all possible *qubit* states (2^N, where N is the number of *qubits*) and describes the state of the whole quantum computer at time t, and $H(t)$ is the time-dependent Hamiltonian modeled in the gate-based quantum computer (e.g. IBM, Google, Intel, Rigetti) or annealer hardware (e.g. D-Wave Systems).

The mapping process of a circuit to satisfy quantum processor restrictions is known as the compiling, mapping, synthesis, transpiling, or routing problem [9]. In brief, the quantum circuits can be described by: data (*qubits*), operations (*quantum gates - unitary transformations*) and results (*measurements*).

Currently, companies in the field of quantum computing have been racing to handle problems related with the number of *qubits* and their interconnection, or other forms of quantum computation, e.g. cluster state, trapped ions, photons, NV-centers, SQUIDs. Some quantum platforms offer free available services allowing to conduct experiments so we can be knowledgeable with quantum environments and quantum scripting languages (e.g. IBM Q Experience, D-Wave Leap). Thus, two available models and directions of quantum computing to conduct quantum experiments can be: (a) *gate-based model* and (b) *adiabatic model*. In the gate-base model a quantum algorithm is represented as a quantum circuit which is compiled to specific quantum gates and executed on a quantum device or simulator. In the adiabatic model we encode a problem into an *Ising Hamiltonian* that allows to find approximate solutions resorting to heuristic methods, with variables that represent magnetic dipole moments of atomic *spins* which can be in one of two states ($+1$ or -1) embedded into the quantum hardware graph (chimera). In this case, we use a quantum annealer or a classical solver to sample low-energy states which are the optimal solutions of a problem being solved [7, 130, 210].

In the next section a description on circuit model quantum computing is introduced and concepts and definitions are given, such as, *qubit*, Bra-ket notation, superposition, quantum circuit, quantum logic gates, entanglement, interference, measurement, quantum operations, multi-*qubit* systems. In addition, is also shown the description of gates as mappings and examples of quantum hardware and common quantum gates, with code scripts examples that execute quantum operations, ending with a discussion on algorithms and complexity.

2.2 Circuit Model Quantum Computing

Quantum computers are different from binary digital electronic computers based on transistors. Common digital computing encodes the data into binary digits (bits), each of which is always in one of two definite states (0 or 1). Quantum computing uses a different computational paradigm from classical, uses quantum bits [70]. A gate-based *qubit* quantum computer stores the data in quantum bits and allows to write all possible values into a register simultaneously or entangle two *qubits*, properties that do not exist in traditional computers. Furthermore, a universal gate quantum computing system is where basic quantum circuit operations, can be handled to run complex algorithms. Examples are Shor's algorithm for integer factorization in polynomial time [185] (to break RSA cryptography) and Grover's algorithm for black-box search in $O(\sqrt(2^n))$ time [75] (to perform a faster search). Besides these two most notable algorithms many more can be mentioned [1].

In gate-based quantum computing, a unitary operation is typically decomposed into elementary gates. Gate-model quantum computers are composed by circuits with **qubit** registers and gates operating on them. In the quantum machine we have as input a set of quantum instructions that are mapped to the device and after that a result is produced, which is not easily interpretable, and it's not easy to measure either. We have to perform qubit **measurements** and turn them into 0 or 1. The state of the **qubit** can be $|0\rangle$, $|1\rangle$, or a **superposition** of them. The superposition allows the **qubit** to be in both states at the same time, also, it can be defined as a weighted unit combination between the basic states 0 and 1. Moreover, **entanglement** can occur when pairs or more than two particles are generated or interact in a way that the quantum state of each particle can't be described independently of the state of the other(s). Those quantum mechanics phenomena occur during quantum computations and these operations are executed and results

[1]https://math.nist.gov/quantum/zoo/

are collected using measurements.

2.2.1 Qubit, Bra-ket notation and definitions

The basic concept in quantum computing is a *qubit*. A *qubit* is a unit of quantum information that assigns a value to the two classical answers *yes* and *no* - corresponding to the two classical bits 1 and 0, respectively.

Definition 1. *Qubit. A qubit is a unit vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ of the Hilbert space $\mathbb{C}^2$.*

The basis $\{|0\rangle, |1\rangle\}$ is known as the standard basis. A unitary transformation is defined by its effect on basis states, extended linearly to the whole space, e.g. the Hadamard transformation can be defined by:

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, \quad |1\rangle \rightarrow \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \tag{2.1}$$

which corresponds to the matrix $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$. The Hadamard transformation creates superpositions: $H|0\rangle = |+\rangle, \quad H|1\rangle = |-\rangle$.

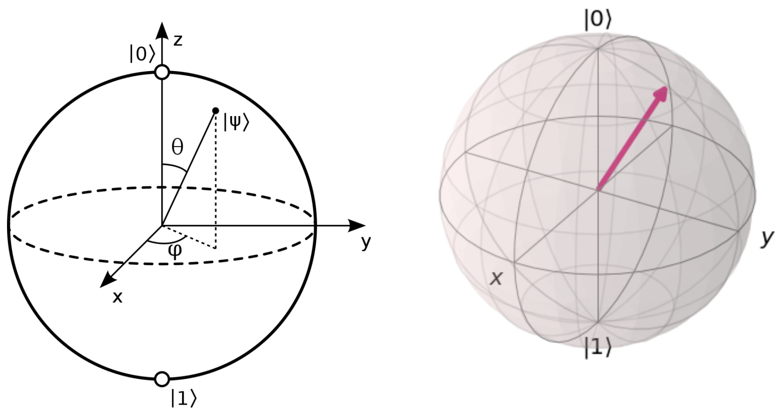
In quantum mechanics, bra-ket notation is a standard notation for representing quantum states. In order to calculate the scalar product of vectors, the notation uses angle brackets $\langle\ \rangle$, and a vertical bar $|$. The scalar product is then $\langle\phi|\psi\rangle$ where the right part is the "ket" (a column vector) and the left part is the "bra" - the Hermitian transpose of the ket (a row vector). Bra-ket notation was introduced in 1939 by Paul Dirac and is also known as the Dirac notation.

A *qubit* can "take" infinitely many different values and it is continuous. Also, it can be in a *superposition* of these states $x|0\rangle + y|1\rangle$, where x and y are complex numbers such that $|x|^2 + |y|^2 = 1$. Assuming two states represented by two vectors, the *superposition* is the addition vector. Moreover, in a classical computer, the maximum integer number that can be stored in N *bits* is $2^N - 1$, e.g. if we consider $N = 3$ we can store numbers from 0 to 7 ($2^N - 1 = 7$), with 7 being represented as 00000111 in binary form (obviously, this will depend on the encoding used by the underlying architecture). In a quantum *qubit* storage, for example, with 3 *qubits*, we can obtain coefficients for $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$ and $|111\rangle$. As such, in the classic machine we store 1 number at a time, in the quantum machine we store $2^N - 1$ numbers at the same time.

Postulate 1: A quantum bit

> *Associated to any isolated physical system is a complex vector space with inner product (i.e. a Hilbert space) known as the state space of the system. The system is completely described by its state vector, which is a unit vector in the system's state space* [147].

A closed system depict a unit vector in a complex vector space. The *qubit* is a unit vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ of the Hilbert space $\mathbb{C}^2$ where the basis $\{|0\rangle, |1\rangle\}$ is known as the standard basis. In addition, we can obtain spherical coordinates for a point in $\mathbb{R}^3$. Resorting to the Bloch sphere, shown in Figure 2.1, we can write a single-*qubit* state $|\psi\rangle$ using the representation: $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$ with $\varphi \in [0, 2\pi)$ and $\theta \in [0, \pi]$.



(a) *Geometrical representation* (b) *Qiskit representation*

Figure 2.1: Bloch sphere, representation of the pure state space of a two-level (or two-state) quantum mechanical system (*qubit*) (Source: (a) Bloch sphere from Wikipedia and (b) Qiskit script with two instructions: `from qiskit.visualization import plot_bloch_vector; plot_bloch_vector([0,0.5,0.9]))`.

Definition 2. *Superposition. A pure qubit state is a linear superposition of the basis states when we measure the qubit. The probability of outcome $|0\rangle$ is $|\alpha|^2$ and the probability of outcome $|1\rangle$ is $|\beta|^2$, where α and β are constrained by the expression $|\alpha|^2 + |\beta|^2 = 1$.*

2.2.2 Quantum systems

Postulate 2: Evolution of quantum systems

> *The evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state of $|\psi'\rangle$ of the system at time t_2 by a unitary operator U which depends only on times t_1 and t_2* [147].

The evolution of a closed system in a fixed time interval is described by U.

Definition 3. *Quantum circuit. A model for quantum computation where a computation is a sequence of quantum logic gates.*

Definition 4. *Quantum logic gates. A quantum circuit operating on a small number of qubits. Quantum logic gates are reversible.*

The **depth** of the quantum gate structure means the number of time steps (time complexity) needed for the quantum operations in the circuit to run on the quantum hardware. The **width** is the maximum number of *qubits* allocated at any one time. Circuits are networks composed of wires that carry bit values to gates to perform elementary operations on those bits and can be described by Boolean linear algebra (using vectors). Moreover, the *gates* are basic units that perform Boolean logical operations.

The CNOT gate can represent gates applied to two wires. Therefore, we can use the **Controlled NOT** (CNOT) which acts on two bits, the **control** bit and the **target** bit. The goal is to apply the NOT operation to the target if the control bit is 1. In the opposite of a classical computer gate, CNOT is an XOR gate with two outputs. For example, if we set the value of the control bit to $c = 0$ or to $c = 1$, CNOT performs the mapping to the target bit to $t \oplus c$ with $\oplus$ representing the logical exclusive-or operation, or addition modulo 2. The CNOT transforms the quantum state as follows:

$$\text{CNOT}(a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle) = a\,|00\rangle + b\,|01\rangle + c\,|11\rangle + d\,|10\rangle \tag{2.2}$$

$$\text{CNOT}\begin{pmatrix}0\\0\\1\\0\end{pmatrix} \equiv \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\begin{pmatrix}0\\0\\1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix} \tag{2.3}$$

Definition 5. *Entanglement. A physical phenomenon that occurs when pairs or more than two particles are generated or interact in forms that the quantum state of each particle can't be described independently of the state of the other(s).*

Definition 6. *Quantum interference. Interaction of waves that are correlated. Two waves can add together to create a wave of greater amplitude (constructive interference) or subtract from each other to create a wave of lesser amplitude (destructive interference), it depends on their relative phase.*

Postulate 3: Measurement

> Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the the measurement outcomes that may occur in the experiment [147].

The interaction between a *qubit* with other *qubits* allows measurement to occur. Besides, the state is observed with a probability ≤ 1 which suggests uncertainty in the result states and cannot be copied. Furthermore, *interference* (interaction of waves that are correlated) can give rise to a measurement-like state collapse named *decoherence*.

Definition 7. *Measurement. Measurement of a quantum state yields a classic state measure* $(|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle) = |i\rangle$*, with probability* $|\alpha_i|^2$*. The quantum superposition collapses into the measured state. The* $\alpha_i's$ *cannot be accessed directly, i.e., cannot be measured, though there are algorithms to estimate: amplitude estimation, quantum tomography.*

The model of computation of a quantum device can be summarized as (a) the machine is initialized and has a state which is contained in a quantum register, (b) the state evolves according to an algorithm described by operations, and (c) the information on the state of the quantum register is obtained performing a measurement operation. Similar to a Turing machine, a quantum machine goes from one of a set of *input* states to one of a set of *output* states. However, although the *output* is fully determined by the *input* state, it is not observable [49]. We cannot perform several independent measurements of $|\psi\rangle$ since we cannot copy the state (no-cloning theorem [33, 217]).

2.2.3 Reversible *vs* irreversible computation

Typically general computations (e.g. Turing machine) are logically irreversible [14, 170] and have to deal with energy dissipation issues. Logically reversible computing helps to overcome this problem. Also, in quantum computation it is often needed to remove the garbage *qubits* accumulated at the end of a reversible computation, since these can destroy the interference important for the quantum algorithms. Most of the irreversible computation can be converted into a reversible computation. Figure 2.2 shows an example of the classical irreversible and gate (left) and its reversible version (right).

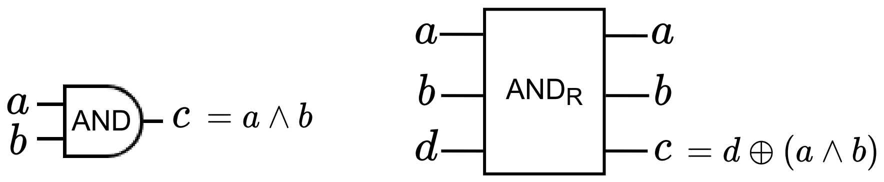


Figure 2.2: AND gate and reversible AND (AND$_\text{R}$)

The reversible AND corresponds to the Toffoli quantum gate which is a generalization of CNOT. Instead of just one control bit, there exists two bits that control if the third bit is flipped. Contrasting to the classical AND gate, Toffoli adds one extra input and two extra outputs, which are copies of the two inputs. Fixing the additional input bit to 0 and dropping the copies we can simulate the classical non-reversible AND gate. As such, from an irreversible circuit with depth T and space S, we can build a reversible version that uses a total of $O(S + ST)$ space and depth T.

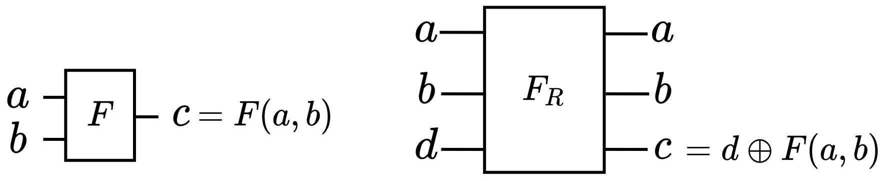


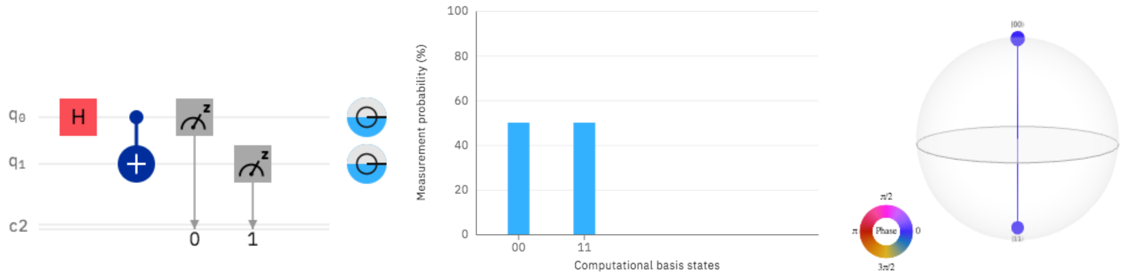
Figure 2.3: F gate and reversible F (F_R)

Figures 2.2 and 2.3 source and adapted from[2].

In general, if we consider a gate F which computes a Boolean function $c = F(a, b)$, its reversible version would be F_R which maps the inputs to outputs with $c = d \oplus F(a, b)$. This generalization applies to any gates with two input wires and one output wire Figure 2.3 presents this generalization.

2.2.4 Multi-qubit systems

Postulate 4: Multi-qubit systems "The state space of a composite physical system is the tensor product of the state spaces of the component physical systems." [147]. The tensor product captures the *superposition* in quantum systems. Furthermore, when we perform measurement, the probability of outcome $|0\rangle$ is $|\alpha|^2$ and the probability of outcome is $|1\rangle$ is $|\beta|^2$, α and β are constrained by the expression $|\alpha|^2 + |\beta|^2$. In addition, *entanglement* is a property of a multi-qubit system that occurs when pairs or more than two particles, interact in a way that the quantum state of each one, cannot be described separately. Figure 2.4 shows a state, we can notice that the state has 50% probability of being measured in the state $|00\rangle$, and 50% chance of being measured in the state $|11\rangle$.



(a) *Circuit* (b) *Histogram* (c) *Q-sphere*

Figure 2.4: Quantum circuit at IBM Quantum Experience

Gates as Mappings. A function f representing a gate can be defined as $f : \{0,1\}^n \rightarrow \{0,1\}^m$ where $\{0,1\}^n$ means the permutations of n 0's and 1's. For example, $\{0,1\}^2$ indicates the set 00, 01, 10, 11, with $n = 2$ bits, map into a m bit register. Boolean functions are mappings of the form $f : \{0,1\}^n \rightarrow \{0,1\}$, e.g. XOR, AND and OR gates are Boolean functions for $n = 2$, also, the NOT gate is a Boolean function with $n = 1$ [228].

Quantum Hardware. Quantum gates/operations matrices which act on a qubit are represented by a 2×2 unitary matrix U. The operation is to multiply the matrix that represents the gate with the vector that depicts the quantum state $|\psi\rangle = U\,|\psi\rangle$ where e.g. 2-qubit gates are 4×4 matrices, and 3-qubit gates are 8×8 matrices. The qubits need to be connected and interact to implement operations. However, not all possible connections can be implemented due to physical reasons, such as, interference of one qubit with another. When programming a quantum

[2]https://quantum-computing.ibm.com/composer/docs/iqx/guide/shors-algorithm

machine, due to the topology of each quantum hardware, we should choose a qubit connection and set the "ideal" coupling map. In the IBM quantum machines, the compiler transforms the user program in order to take advantage of the machine's topology, Figure 2.5 [23].

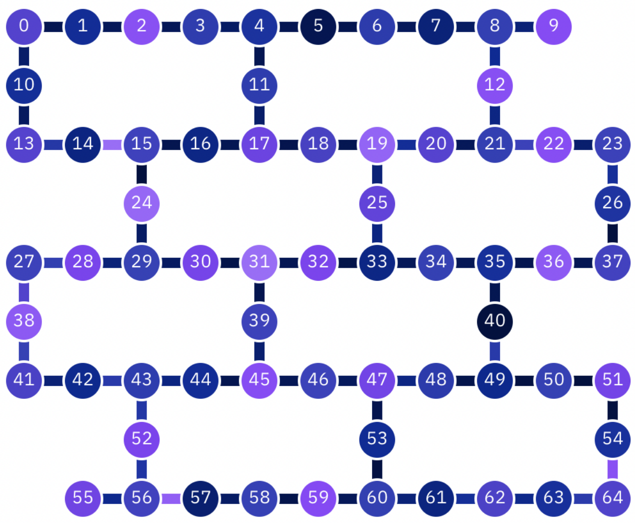


Figure 2.5: IBM Q Experience Manhattan topology with main characteristics: 65 *qubits* and 32 *quantum volume* (source: https://quantum-computing.ibm.com)

Script 2.1 shows an example. We start with the most common operations in quantum computing (Table 2.1) by importing from qiskit the libraries QuantumCircuit to initialize our circuit (qc) and QuantumRegister to register (q) *qubits*. We first perform an operation resorting to the Hadamard gate on *qubit* 0 (line 5). Then, we proceed to a CNOT on control *qubit* 0 and target *qubit* 1 (line 9). Ending with a Toffoli where the last *qubit* (2) is the target, the rest are controls (line 13). The circuit_drawer allows us to picture the circuit of each operation.

Listing 2.1: Example using Qiskit

```
1  from qiskit import QuantumCircuit, QuantumRegister
2  from qiskit.tools.visualization import circuit_drawer
3  q = QuantumRegister(1)
4  qc = QuantumCircuit(q)
5  qc.h(q[0])
6  circuit_drawer(qc)
7  q = QuantumRegister(2)
8  qc = QuantumCircuit(q)
9  qc.cx(q[0],q[1])
10 circuit_drawer(qc)
11 q = QuantumRegister(3)
12 qc = QuantumCircuit(q)
13 qc.ccx(q[0],q[1],q[2])
14 circuit_drawer(qc)
```

Table 2.1: Common quantum gates

Gate	Matrix
Hadamard gate (H)	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Pauli-X, NOT gate (X)	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Z gate (Z)	$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Rotation-X (Rx)	$\begin{bmatrix}\cos(\theta/2) & -\imath\sin(\theta/2)\\-\imath\sin(\theta/2) & \cos(\theta/2)\end{bmatrix}$
Rotation-Y (Ry)	$\begin{bmatrix}\cos(\theta/2) & -\sin(\theta/2)\\-\sin(\theta/2) & \cos(\theta/2)\end{bmatrix}$
Controlled-NOT (CNOT, CX)	$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix}$

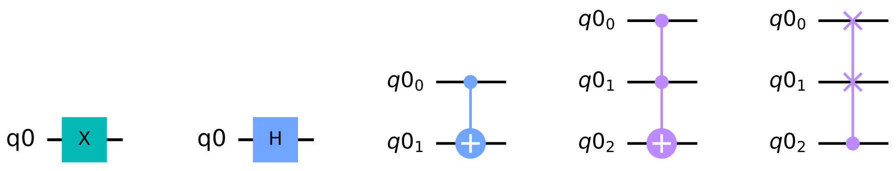
[a] The Hadamard gate rotates the states $\|0\rangle$ and $\|1\rangle$ to $\|+\rangle$ and $\|-\rangle$, respectively, and is useful to create superpositions.

[b] The Pauli-X gates flip the $\|0\rangle$ state to $\|1\rangle$, and vice versa, and is equivalent to Rx, for the angle π.

[b] The Pauli-Z gates flip the $\|+\rangle$ state to $\|-\rangle$, and vice versa, and is equivalent Rz, for the angle π.

[c] The Rx and Ry gates require an angle (in radians) and correspond to rotating the qubit state around the x and y axis, respectively, by the given angle.

[d] The controlled-NOT acts on two qubits, one as 'control' and the other as 'target' and performs an X on the target whenever the control is in state $\|1\rangle$. In case the control qubit is in superposition, it creates entanglement.



(a) *NOT gate* (b) *Hadamard* (c) *CNOT* (d) *Toffoli* (e) *Fredkin*

Figure 2.6: Examples of gates in Qiskit

The quantum gates described in the *IBM's website guide*[3] show how to apply the basic operations of quantum computing. The code in Listing 2.2 is based in OpenQASM IBM's quantum assembly language. We set two quantum and classical registers (lines 3 and 4), perform a Hadamard (line 5), followed by a CNOT operation (line 6), ending with a measurement (line 7).

[3]https://quantum-computing.ibm.com

We add the h gate in the *qubit* 0, putting this *qubit* in **superposition**, then we add the cx gate on control *qubit* 0 and target *qubit* 1, putting the *qubits* in **entanglement**. We finally add a Measure gate to see the state, Figure 2.4.

Listing 2.2: OpenQASM: Hello World

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[2];
creg c[2];
h q[0];
cx q[0],q[1];
measure q -> c;
```

The next script 2.3 shown consists of a Python environment and commands to reproduce the same example resorting to a QASM simulator backend to execute the circuit with 1024 shots giving rise to the histogram in Figure 2.4(b).

Listing 2.3: Qiskit: Hello World

```
from qiskit import ClassicalRegister, QuantumRegister, QuantumCircuit
from qiskit import execute
from qiskit import BasicAer
from qiskit.tools.visualization import plot_histogram, circuit_drawer
backend = BasicAer.get_backend('qasm_simulator')
q = QuantumRegister(2)
c = ClassicalRegister(2)
qc = QuantumCircuit(q, c)
qc.h(q[0])
qc.cx(q[0], q[1])
qc.measure(q, c)
job_exp = execute(qc, backend=backend, shots=1024)
plot_histogram(job_exp.result().get_counts(qc))
circuit_drawer(qc)
```

Quantum scripts can be implemented using different software development kits (SDKs), frameworks or programming languages, e.g. OpenQASM, Qiskit (IBM), Cirq, Forest/pyqil (Rigetti), Q# (Microsoft), Ocean (D-Wave), ProjectQ, PennyLane (Xanadu), TensorFlow Quantum (Google), tket/pytket (Cambridge Quantum Computing), IonQ, XACC.

2.2.5 Quantum algorithms and complexity

According to [147] quantum algorithms can be divided in two main types: (a) algorithms based on Shor's quantum Fourier transform, some of the most popular known algorithms are built on top of Shor's with a exponential speed-up over their classical counterparts. Besides, the classic Fast Fourier transform runs on $O(n \log n)$ time and the quantum version on $O(\log^2 n)$. And, (b) algorithms based on Grover's (used on quantum search). This method allows a quadratic speed-up to some of the best classical algorithms. If the classical algorithm runs on the worst case on $O(N)$, Grover's search algorithm can be improved to run on $O(\sqrt{N})$.

In classical computing there are two main complexity classes: (a) P: class of problems that can be solved by an algorithm in polynomial time, (b) NP: class of problems to which it is unknown if there are solutions in polynomial time. In quantum computing there are equivalent complexity classes: (a) Bounded-error Quantum Polynomial (BQP) time which is the class of decision problems that can be solved by a quantum computer in polynomial time, (b) Quantum Merlin Arthur (QMA), which is, in an informal way, the set of decision problems for which when the answer is YES, there is a polynomial-size quantum proof (a quantum state) which convinces a polynomial-time quantum verifier of the fact with high probability. Moreover, when the answer is NO, every polynomial-size quantum state is rejected by the verifier with high probability. Several problems in computer science are known to be NP-hard (non-deterministic polynomial time)-hard or even NP-complete and difficult to handle on traditional computers. These problems can be mapped and implemented in quantum physical devices instead of traditional digital computers in a BQP time.

In the next section a description on adiabatic model quantum computing is introduced, along with the concepts and theorem. The annealing process is discussed with regard to the quantum annealing Quantum Annealing (QA) and simulated annealing Simulated Annealing (SA).

2.3 Adiabatic Model Quantum Computing

In brief, quantum algorithms evolve by applying an infinitesimal operator (the Hamiltonian) to the initial state which end with a final quantum state. Adiabatic quantum computation (AQC) uses the quantum mechanics properties to achieve that final state (solution). The model that it has been shown so far is called circuit model or gate-based model, an alternative model is named adiabatic. The circuit model is more used and is equivalent to the adiabatic [5].

To deal with the adiabatic-based quantum computing paradigm we resort to the D-Wave's Ocean software stack. Nowadays, the variational quantum implementations are linking the gate-model QC and the adiabatic paradigm. ML tasks are suitable to be processed through the adiabatic-model quantum computer. The gate-model quantum computer can be useful if classical big data can be efficiently converted to a quantum format. Since speed-up is possible, quantum computers are accelerating the development of artificial intelligence, and AI in its turn is contributing to new quantum technologies.

> ”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.” [27, chap. Adiabatic Model Quantum Computing]

2.3.1 Adiabatic concepts and theorem

D-Wave Systems has been developing their own version of a quantum computer that uses annealing, which is different from the gate model based approaches from IBM, Intel, Rigetti and Google. Currently, D-Wave's has more than 5,000 qubits and 35,000 couplers, a larger, denser, and powerful graph for building quantum applications. Unlike universal quantum computers that operate using quantum gates and the principles of the circuit model, devices that are specialize primarily in solving combinatorial optimization problems typically uses QA [113]. QA can be described as a metaheuristic to find the global minimum of a given objective function over a given set of candidate solutions by a process that uses quantum fluctuations. Similar to SA, a metaheuristic for optimization that consists of a local search technique, and is based on an analogy with thermodynamics. In QA the focus has been the performance of quantum annealers against classical algorithms. QA allow solving local search problems in multivariable optimization when there is a need of finding the maximum or minimum for a cost function that comprises independent variables and a large number of instances. An effort that has not yet been addressed with gate models, explained by the large number of *qubits* available in QA, which enables scaling tests over several orders of magnitude of problem sizes [8]. Also, in QA, on the contrary to the circuit based model, the solution to a problem is encoded in the ground state of the quantum system and its susceptibility to environmental issues [36].

Furthermore, QA (also known as the quantum adiabatic algorithm or adiabatic quantum optimization) starts from the ground state of the initial Hamiltonian and concerns on continuously deforming the Hamiltonian such that the system attain the final ground state (e.g from a Ising model) which solves the optimization problem. Besides, the *adiabatic theorem* (explained below) of quantum mechanics ensure that QA finds the final ground state. However, should consider if the run-time is sufficiently large with respect to the inverse of the quantum ground state energy gap. In addition, despite all the sophisticated QA process, it does not mean that generally perform better than classical optimization algorithms [139].

In brief, to describe Adiabatic Quantum Computation (AQC), the system begins in a ground state of a time dependent Hamiltonian $H(s)$ where $s \in [0, 1]$ that is slowly being modified from an initial form H_0 to a final form H_1, in a time $t(s)$ with $t(0) = 0$, $t(1) = \tau$, considering an interpolation between the two Hamiltonians. Starting with $|\psi(0)\rangle$ in the ground state of H_0 the system evolves according to Schrödinger's equation and end up near the ground state of H_1 [57, 166]. The AQC Hamiltonian can be written as follows,

$$H(s) = (1 - s)\, H_0 + sH_1 \tag{2.4}$$

where the Hamiltonian H_1 ground state encodes the solution of a combinatorial optimization problem. Adiabatic quantum computation encodes the solution to a problem in the ground state of the final Hamiltonian, as it occurs in quantum annealing. The adiabatic theorems can have approximate or rigorous versions as described in [7].

Nowadays, there are many types of architectures for quantum computers. In the gate-model the goal is to control and handle the evolution of the quantum states in time in a circuit perspective. In quantum annealing, the goal is to initialize the system in a delocalized state, i.e. anywhere in space, where gradually converges to the description of our problem, ending in the answer we are trying to find by means of energy. In the D-Wave's system we can formulate the problem in different perspectives that allows us to map optimization problems to find the lowest energy state of a system representing the problem. Quantum annealing (as a metaheuristics) aids in finding the global minimum of a given objective function over a given set of candidate states.

Ising model. Shows the relationships between the spins, represented by couplings and where the objective function is as follows:

$$\mathrm{E}_{ising}(\boldsymbol{s}) = \sum_{i=1}^{N} h_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{i,j} s_i s_j \tag{2.5}$$

where the *linear coefficients* corresponding to *qubit* biases are h_i, and the *quadratic coefficients* corresponding to coupling strengths are $J_{i,j}$, and the variables are either "spin up" ($\uparrow$) or "spin down" ($\downarrow$) states which correspond to +1 and -1 values, s_i and s_j. A problem can also be formulated as a Quadratic Unconstrained Binary Optimization (QUBO) where variables can be *true* and $false$ with states corresponding to 1 and 0 values.

Ocean software is an open-source SDK from D-Wave Systems for solving hard problems with quantum computers[4].

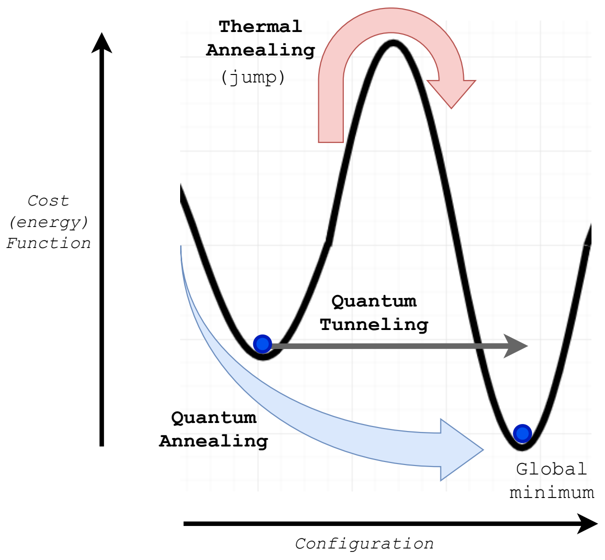


Figure 2.9: Graphical representation of quantum tunnelling. Figure adapted from Wikipedia on the topic "Quantum annealing"

In some cases, quantum tunneling can escape from local minima that overcomes conventional methods such as SA.

[4]https://docs.ocean.dwavesys.com

In sum, the qubits are interconnected to exchange information and are connected via couplers, which are also superconducting loops. The interconnection of qubits and couplers, and the control of the magnetic fields, creates programmable quantum devices. When a qubit is in both 0 and 1 state at the same time, is in superposition. When the problem-solving process ends, the superposition collapses into one of the two classical states, 0 or 1, which means that the Quantum Processing Unit (QPU) achieved a solution, and all qubits are in their final states and their values are returned as a bit string.

2.3.2 Annealing process

QA is analogous to SA. In nature, physical systems tend to evolve toward their lowest energy state, e.g., objects slide down hills. It is generally most efficient to move downhill and avoid climbing hills that are too high. Quantum annealing can be explained comparing to a landscape with valleys, mountains, tunnels and water. Once the problem to be solved is translated into an energy landscape, the machine finds the lowest point by following the laws of quantum mechanics. In brief, a QA, solves hard optimisation problems by evolving a known initial configuration at non-zero temperature towards the ground state of a Hamiltonian [24].

Algorithm 1: SA [169]

```
input : problem (a problem), schedule (a mapping from time to "temperature")
static : current (a node), next (a node), T (a "temperature" controlling the probability of
         downwards steps)
1  function SA(problem, schedule)
2  current ← Make-Node(Initial-State[problem])
3  for t ← 1 to ∞ do
4      T ← schedule[t]
5      if T=0 then
6          current
7      end
8      next ← a randomly selected successor of current
9      $\Delta E \leftarrow Value[next] - Value[current]$
10     if T=0 then
11         current ← next
12     else
13         current ← next only with probability $e^{\frac{\Delta E}{T}}$
14     end
15 end
16 return a solution state
```

Algorithm 26 refers to the main routine QA which calls alternatively the subroutines Local Optimization 6 and Quantum Transition 2. The aim of the Local Optimization is to carry out a local descendent from the current configuration. The Quantum Transition generates a transition from the current configuration according to the surrounding environment.

Algorithm 2: QA [11, 46]

```
input : $init$ (initial condition), $\nu$ (control parameter), $t_{max}$ (duration), $t_{drill}$ (tunnel time), $t_{loc}$
        (local opt. time)
function QA($init, \nu, t_{max}, t_{drill}, t_{loc}$)
$t \leftarrow 0$
$\epsilon \leftarrow init$
$v_{min} = cost(\epsilon)$
while $t < t_{max}$ do
    $j \leftarrow 0$
    repeat
        $i \leftarrow 0$
        repeat
            $\epsilon \leftarrow$ QT($\epsilon, \nu, t_{max}$)
            if $cost(\epsilon) < v_{min}$ then
                $v_{min} \leftarrow cost(\epsilon)$
                $i, j \leftarrow 0$
            else
                $i \leftarrow i + 1$
            end
        until $i > t_{loc}$
        $\epsilon \leftarrow$ LO($\epsilon$)
        if $cost(\epsilon) < v_{min}$ then
            $v_{min} \leftarrow cost(\epsilon)$
            $j \leftarrow 0$
        end
    until $j < t_{drill}$
    draw a trajectory of length $\nu t_{max}$ and jump there
    LO($\epsilon$)
end
```

Algorithm 3: Quantum Transitions [11, 46]

```
input : $\epsilon$ (initial condition), $\nu t$ (chain length), $Neigh$ (set of neighbours to estimate)
function QT($\epsilon, \nu t, Neigh$)
for all $neighbourk \in Neigh$ do
    estimate the wave function $\psi_\nu(k)$
end
$best \leftarrow$ select a neighbour in $Neigh$ with probability proportional to $\psi_\nu$
return $best$
```

Algorithm 4: Local Optimization [11, 46]

```
input : $\epsilon$ (initial condition)
function LO($\epsilon$)
return *the best solution found by any steepest descent strategy*
```

Comparison between SA and QA 2.2.

Table 2.2: SA versus QA

	SA	QA
Comp.	**i)** Reheating; **ii)** Statistically seeks wider not deeper valleys; **iii)** Gets stuck to local minimum.	**i)** Is not locked by barriers; **ii)** Tunnel into deeper valleys; **iii)** Finds global minimum.

2.4 Quantum Machine Learning

”The quest of machine learning is ambitious: the discipline seeks to understand what learning is, and studies how algorithms approximate learning. Quantum machine learning takes these ambitions a step further: quantum computing enrolls the help of nature at a subatomic level to aid the learning process.” [215, Peter Wittek]

In the 80’s, Feynman [61] proposed quantum computing after performing quantum mechanics simulations on a classical computer. Nowadays, the experiments at the smallest scales of energy levels of atoms and subatomic particles lead us to the new merged field Quantum Machine Learning (QML) [39]. The classical Machine Learning (ML) [22, 140] can be roughly fit into three main categories depending on the type of data: supervised learning (classification or regression), unsupervised learning (e.g. clustering) and reinforcement learning (e.g. deep reinforcement learning). Those categories in a quantum domain allow to take an advantage of combining fields. For example, quantum computing and data science to speed-up processing, ML optimization applied to quantum experiments, or the exploration of quantum-enhanced learning agents. Currently, quantum computing [147] is approaching the mainstream with the introduction of classical ML algorithms to analyze quantum systems and, conversely, quantum versions of ML algorithms e.g., quantum support vector machines, quantum principal component analysis, quantum neural networks or quantum deep learning. The goal is to solve large scale and complex computational issues by a QML procedure, improving a classical ML method, giving rise to, e.g., quantum pattern recognition, quantum classification and/or solving large combinatorial problems [18].

Essentially all ML classical problems are being upgraded to a quantum level [174, 215] where the supervised quantum prediction occurs: (a) when we know the quantum states that describe the system, and (b) we map classical problems from a typical Euclidean space onto the quantum states in the corresponding Hilbert space [173]. QML is focused in the exchange between quantum computing and ML, and seek techniques from one topic that can solve the problems of the other. Moreover, ML concerns the exploration of large vectors in high-dimensional spaces suitable for quantum computation since it allows to manipulate high-dimensional vectors in large tensor product spaces [126].

Arguably, the three main popular quantum models in the literature are gate-based, hybrid classical quantum and annealing-based [62, 114]. These models pose computational challenges in the areas of quantum languages, quantum compilers and quantum algorithms. For example, it is not trivial to encode and map an input and program an algorithm that, given the input will produce the desired output in a quantum hardware. There are specific techniques we should consider namely for encoding classical data, such as, basis, angle, amplitude, variational/trained or higher order embedding encoding [31, 80, 127, 177, 205]. Moreover, results of a quantum hardware are not trivial to retrieve due to the physics behind measuring the qubit's (equivalent to a classical bit) values [179].

Figure 2.10 shows four different possible approaches that merge the topics quantum computing and machine learning [6, 55].

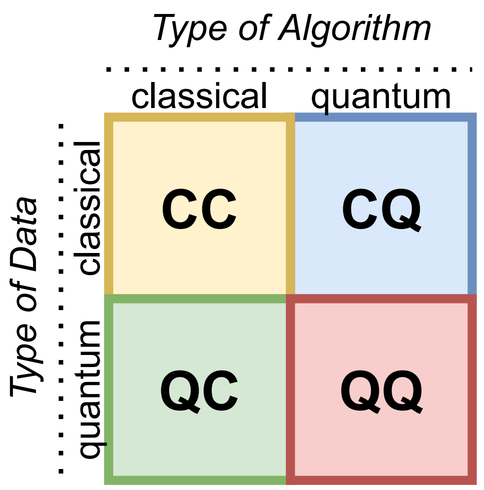


Figure 2.10: The first letter indicates the type of system and the second letter the type of device. Also, "C" means classical and "Q" means quantum. Figure adapted from [175].

For example, in quantum-classical (QC) the ML extract knowledge from quantum measurement data. On the other hand, classical-quantum (CQ) resorts to classical data and quantum algorithms to solve ML challenges.

2.4.1 Machine learning

ML is a branch of AI that applies algorithms to incorporate the relationships among data and information. Let $\{x_1, ..., x_N\}$ be a large set of N examples called a *training set*. A ML algorithm (a learning algorithm) can be expressed as a function $y(x)$ which takes a new example x as input and give rise to an output vector y where the function $y(x)$ is determined during the *training phase*, also known as the *learning phase*, based on the training data. Once the mathematical model (created by the training process) is trained, is then capable to identify new examples in a *test set*.

Generalization occurs when the algorithm is able to categorize correctly new examples different from the data used for training. If an algorithm performs well on the *training set* but fails to *generalize*, we can say $overfitting$ occurs. Usually, the original input variables are *preprocessed* using the same steps in the training data and in the test data, this phase is also known as *feature extraction*. This procedure has many advantages, such as, making easier for an algorithm to distinguish between classes, or for speed-up computations.

Applications in which the data provides given examples and for each example it specifies an outcome are known as *supervised learning* problems (e.g. classification, regression). In other problems, inference from data consist of input examples without labeled responses, is known as *unsupervised learning* problems (e.g. clustering, association). Additionally, *reinforcement learning* is characterized by actions in a given scenario in order to maximize a reward. Those are types of ML problems. Furthermore, the ML process can be summarized by the steps: (a) data acquisition and preprocessing where occurs data set integration, outliers removal, etc., (b) feature selection and extraction for identification and extraction of relevant characteristics from data, (c) model selection where a model is chosen considering the task to be solved, and (d) validation where a performance measure is calculated according to the task, e.g. accuracy (classification) and mean absolute error (regression) evaluated on a validation set [22, 79, 140].

2.4.2 Variational quantum concepts

> ”Quantum systems produce atypical patterns that classical systems are thought not to produce efficiently, so it is reasonable to postulate that quantum computers may outperform classical computers on machine learning tasks. The field of quantum machine learning explores how to devise and implement quantum software that could enable machine learning that is faster than that of classical computers. Recent work has produced quantum algorithms that could act as the building blocks of machine learning programs, but the hardware and software challenges are still considerable.” [18, Jacob Biamonte et al.]

As for example, a machine learning model can be defined considering a *variational quantum classifier* which uses *variational circuits* where one circuit associates the gate parameters with data inputs, and the other depends on trainable parameters, ending with a measurement.

Variational quantum circuits are quantum algorithms that depend on the following steps: (a) preparation of a fixed initial state (e.g., the zero state), (b) a quantum circuit $U(\theta)$ parameterized by a set of parameters θ, and (c) the measurement of an observable at the output.

In PennyLane[5] classical computations and quantum are linked through quantum nodes which execute a variational circuit (a parametrized quantum computation) on a quantum device. Besides, it computes the gradients of quantum nodes which allows hybrid computations. PennyLane is based in four concepts: hybrid computation, quantum nodes, variational circuits and quantum gradients.

[5]https://pennylane.readthedocs.io

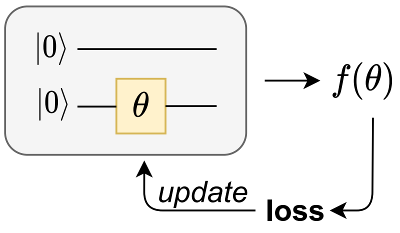


Figure 2.11: The principle of a variational circuit (adapted from: https://pennylane.ai/qml/glossary/variational_circuit.html)

Definition 8. *Quantum nodes. Are like a black box where a basic computational unit is programmed on a quantum circuit and only classical data can enter or exit.*

Definition 9. *Hybrid computation. Is composed by algorithms that integrate both classical and quantum processing, according to the rules of quantum nodes.*

Definition 10. *Variational circuits. Are quantum algorithms that can be optimized resorting to hyperparameters. First, we prepare a fixed initial state, then a quantum circuit is parameterized by both the input x and the function parameters θ. Afterwards, a measurement of an observable is made at the output.*

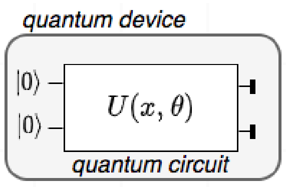


Figure 2.12: Varational circuit

Figure 2.12 adapted from source[6].

Definition 11. *Quantum gradients. Are defined by the gradient of a quantum node $\nabla f(x, \theta)$ calculated by shifting one variable and resorting to a linear combination of two quantum nodes. The same quantum device allows us to perform the computations of quantum nodes and the calculation of the gradients of quantum nodes.*

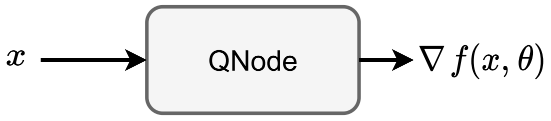


Figure 2.13: Hybrid computation with quantum nodes and quantum gradients

Figure 2.13 adapted from source[7].

[6]https://pennylane.readthedocs.io/en/user-docs-refactor/concepts/overview.html
[7]https://pennylane.ai/qml/glossary/variational_circuit.html

Strawberry Fields[8] is a full-stack Python library for designing, simulating, and optimizing continuous variable (CV) quantum optical circuits [106]. PennyLane is a cross-platform Python library for quantum machine learning, automatic differentiation, and optimization of hybrid quantum-classical computations [15].

Quantum computing is approaching the mainstream with the introduction of classical ML algorithms to analyse quantum systems and quantum versions of ML algorithms e.g. Quantum Support Vector Machine (QSVM), Quantum Principal Component Analysis (QPCA), Quantum reinforcement learning, Quantum Bayesian inference [18]. These advances can provide improvements in Artificial Intelligence (AI) to solve demanding data-related issues using hybrid quantum-classical models [54, 175].

QML arises from the need to quantise learning by converting a classical algorithm to its quantum counterpart, and many algorithms are being implemented e.g. a quantum binary classifier that resembles a multi layer perceptron [175, 178]. Putting big classical data in *superposition* remains a challenge. In the *quantum learning phase* we can convert input data into quantum states to *learn*, process and store the learning result in *qubits*. However, we cannot use again the stored information, and this should be considered during the *test phase*. The supervised quantum prediction occurs when, (i) we know the quantum states that describe the system, and (ii) we map classical problems from a typical Euclidean space onto the corresponding Hilbert space [173].

Still, quantum research direction points out that quantum computers can successfully *learn* resorting to the QPU, a physical chip that contains interconnected *qubits*. Nevertheless, as pointed out by [141], in a gate-based approach, the quantum computer performance depends on the number of physical *qubits*, their connectivity, number of gates applied to prior errors or, *decoherence* that occurs when a quantum system is not well isolated, available hardware gate set, and number of operations that can run in parallel. Physical properties that may influence the mapping of classical problems and compromise the *learning phase*.

Comparison of classical ML algorithms and their quantum counterparts based in computational complexity and learning performance have been introduced [39, 215]. Typically involve the manipulation and classification of large vectors in high-dimensional spaces. Although quantum advantage has not been fully proved, operations that involve the manipulation of high-dimensional vectors in large tensor product spaces according to [126] are faster than in classical computations. For instance, the quantum SVM approach shows that run time in training and classification equals $O(logNM)$ with N being the dimension of the feature space and M the number of training vectors. A fast quantum evaluation of inner products is made when approximating the least-squares problem in M due to a matrix inversion algorithm [165].

Furthermore, e.g. binary classification can be represented by a simple perceptron, where the input and weights are encoded on the quantum hardware based on quantum information principles [199]. Moreover, on the opposite to the conventional bit registers, the fundamental

[8]https://strawberryfields.readthedocs.io

computational units in deep learning are continuous vectors and tensors converted in high dimensional spaces, suitable to adapt to the quantum domain [105]. Besides all existing quantum ML algorithms versions, Quantum Theory (QT) has been introduced as a replacement of classical statistics and ML algorithms which should outperform classical classification methods [203].

In QML some hot topics consist on, e.g. how to load the input, to read the output and the circuits size. To overcome certain issues, a number of algorithms in QML uses procedures such as Quantum Random Access Memory (QRAM) which allow queries in superposition and are based on the exponential speed-up of methods such as: Quantum Fourier Transform (QFT), Quantum Phase Estimation, HHL - an example of Quantum Basic Linear Algebra Subroutines (QBLAS). In addition, other quantum algorithms used in QML include amplitude amplification and QA. Also, kernel methods are discussed [137, 172].

2.5 Quantum Propositional Logic

The next section concerning classical logic that leads to Quantum Propositional Logic (QPL) is based on the work "SATyrus2: Compilando Especificações de Raciocínio Lógico" of Bruno França Monteiro [142].

Propositional logic can aid to define simple constraints that are important building blocks for complex problems. In the experiments, we map the problem to a working graph of the D-Wave Systems. Our aim with this work is to translate original problems into a quantum problem by (i) solving constraints problems using the language of the D-Wave System and (ii) contribute to the exploration of new technologies and reasoning in the context of the Constraint Satisfaction Problem (CSP).

2.5.1 Satisfiability and conjuntive normal form

The problem of Boolean Satisfaction (usually referred to as SAT [69]) is a decision problem that consists of finding an attribution of values to the propositional variables belonging to any Boolean formula in order to make it true (satisfied). If there is no such attribution, the formula is said to be unsatisfiable. Formulas are expressions that satisfy the following laws:

- A propositional variable is a formula.
- If p is a propositional variable, its negation, $\neg p$, is also a formula.
- If p and q are formulas and $\circ$ is a binary operator (e.g. $\wedge$, $\vee$, $\rightarrow$), then $(p \circ q)$ is a formula.

Given any formula, the Satisfiability problem can be stated as follows: is it possible to assign Boolean values $\{V, F\}$ to propositional variables in such a way that the formula is evaluated as true? For example, consider the Formula 2.6:

$$p \wedge (\neg q \vee r) \tag{2.6}$$

Table 2.3 represents the Formula 2.6 truth table. Through it, it can be seen that there are three value assignments that make Formula 2.6 true. In the context of the Satisfaction problem, each of these five attributions is called a solution to the problem.

Table 2.3: Truth table for the Formula 2.6

p	q	r	$\neg q \vee r$	$p \wedge (\neg q \vee r)$
V	V	V	V	V
V	V	F	F	F
V	F	V	V	V
V	F	F	V	V
F	V	V	V	F
F	V	F	F	F
F	F	V	V	F
F	F	F	V	F

The Boolean Satisfaction problem is NP-Complete [40]. A special case for this problem is the situation in which the formula to be satisfied is found in Conjunctive Normal Form (CNF). A formula is in Conjunctive Normal Form when it is composed of conjunctions of clauses. Clauses are literal disjunctions. These, in turn, are propositional variables or negations of propositional variables where p and $\neg q$ are examples of literals. The disjunction of these literals, $(\neg q \vee r)$, is an example of a clause. A clause is satisfied if there is at least an assignment of values to the variables belonging to the clause that makes it true. Consider the Formula 2.7:

$$((p \rightarrow q) \wedge (q \rightarrow r)) \wedge \neg(p \rightarrow r) \tag{2.7}$$

Table 2.4: Truth table for the Formula 2.7

p	q	r	$(p \rightarrow q) \wedge (q \rightarrow r)$	$\neg(p \rightarrow r)$	$((p \rightarrow q) \wedge (q \rightarrow r)) \wedge \neg(p \rightarrow r)$
V	V	V	V	F	F
V	V	F	F	V	F
V	F	V	F	F	F
V	F	F	F	V	F
F	V	V	V	F	F
F	V	F	F	F	F
F	F	V	V	F	F
F	F	F	V	F	F

The Formula 2.7 (with the corresponding truth Table 2.4) is unsatisfiable, that is, there is no attribution of values to the variables p, q and r that makes the formula true as a whole.

Considering the above, an optimization problem can be formulated: given a formula written in Conjunctive Normal Form, what is the attribution of Boolean values to the variables that maximizes the number of satisfied clauses? This problem, called the Maximum Satisfiability Problem (MAX-SAT), has many applications [77].

2.5.2 Mapping satisfiability in linear programming 0-1

Mapping rules. Consider the following mapping (H) that relates Boolean values to the integers 0 and 1, described in [121]:

- $H(True) = 1$
- $H(False) = 0$
- $H(\neg p) = 1 - H(p)$
- $H(p \wedge q) = H(p) \times H(q)$
- $H(p \vee q) = H(p) + H(q) - H(p \wedge q)$

The rules above map logical expressions written in the Conjunctive Normal Form to integer values belonging to the set 0,1. More specifically, the true Boolean is mapped to 1, while false propositions are mapped to 0. In the context of the pseudo-Boolean programming problem that can be obtained through this mapping, only two solutions are possible: 1, if the original formula is satisfied, and 0, otherwise. This implies the need to maximize the objective function constructed by the mapping, since, among the two possible values of a solution, 1 is the highest value. If the objective function was to be minimized, the mapping should be applied to the negation of the original Boolean formula.

Thus, given a formula ϕ written at the CNF, a pseudo-Boolean programming problem can be obtained if the mapping described above is applied to the formula $\neg\phi$: the problem will be to minimize the objective function constructed by converting between formulas and algebraic expressions. If a solution with a cost equal to 0 is found, the original formula ϕ is satisfied. The mapping presented here relates SAT to the Linear Programming 0-1 problem by constructing an objective function - hereinafter called the *energy function* - obtained by applying the H function to the negation of the formula to be satisfied 2.8:

$$E = H(\neg\phi) \tag{2.8}$$

Consider the following trivial example: you want to satisfy Formula 2.9. To do this, it is necessary to find at least an assignment of Boolean values to the propositional variables a and b in order to make Formula 2.9 true ($\phi = V$).

$$\phi = \neg a \vee b \tag{2.9}$$

The energy function corresponding to Formula 2.9 can be obtained through the direct application of the mapping rules defined previously. The application of these rules, together with a brief logical development of the obtained proposals, gives rise to the following energy function:

$$E = H(\neg\phi) = H(\neg(\neg a \vee b)) = H(a \wedge \neg b) = H(a) \times H(\neg b) = \underbrace{H(a) \times (1 - H(b))}_{\text{energy function}} \tag{2.10}$$

Replacing (for reasons of clarity) the symbols $H(a)$ and $H(b)$ with a^* and b^* in the obtained energy function, the original problem, that is, satisfying Formula 2.9, becomes the following pseudo-boolean linear programming problem:

$$minimize : E = a^* \times (1 - b^*) \tag{2.11}$$

where $a^* \in \{0, 1\}$ and $b^* \in \{0, 1\}$.

Solving the above problem is trivial: $a^* = 0$ is one of the possible solutions. A family of problems can be represented using such a formulation.

In short, this chapter described the main concepts and terminology of each topic presented in this thesis: quantum computing from gate-based to adiabatic modeling, quantum machine learning and quantum propositional logic. In the next chapter, we discuss the mapping of applications to quantum machines. We start by enumerating some encoding techniques, and how to perform the mapping according to each problem and quantum device.

Chapter 3

Mapping applications to quantum hardware

Mapping classical problems to quantum machines it is a hard task, since we need to accurately comprise the data to encode and the mappings are tailored to each problem and machine. In this context, there exist libraries already available that we can find e.g. in PennyLane, in the section *embeddings templates* which encode input features into the quantum state of the circuit that can have trainable parameters and repeated layers. This is a complex topic given the range and type of problems we can possibly map to a quantum machine, novel approaches are always surfacing. In this section, we discuss various kinds of mappings, encodings or embeddings, highlighting relevant work in the field. As well as, we describe a method for modeling optimization problems using Quantum Propositional Logic (QPL).

3.1 Gate-based Model Mapping

A way to encode (or embed) our problems into a quantum machine, can be by transforming classical data points as quantum states in a Hilbert space, through a quantum feature map $\varphi : \mathbb{R}^N \to \mathbb{C}^{2^n}$ in a form of dimensionality reduction. The quantum state provides a probability distribution for the result of each possible measurement. In that sense, considering a classical data point x with N features, when translated to a 2^n-dimensional vector $\varphi(x)$, it describes the initial quantum state $|\varphi(x)\rangle$. As such, to map the information to *qubits*, we can consider we have an input dataset $D = \{x_1, ..., x_m, ..., x_M\}$ with M samples, for $m = 1, ..., M$ where M represents N-dimensional vectors x_m.

Basis Embedding. If we consider that each example is a N-bit binary string $x_s = (b_1, ..., b_N)$ with $b_n \in \{0, 1\}$ with $n = 1, ..., N$, in the basis embedding case, x_m is directly mapped to quantum state $|x_m\rangle$. Therefore, the number n of *qubits* should be at least equal to N. As such, D is mapped in the following way:

$$|D\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{S} |x_m\rangle$$

If we consider two data points ($x_1 = 10$ and $x_2 = 01$) the mapping takes the form $|x_1\rangle = |10\rangle$ and $|x_2\rangle = |01\rangle$,

$$|D\rangle = \frac{1}{\sqrt{2}} |10\rangle + \frac{1}{\sqrt{2}} |01\rangle$$

Amplitude Embedding. Also known as wave function embedding which transforms the data points into amplitudes of the quantum state. In this case, a normalized N-dimensional data point x is mapped through the amplitudes of a n-qubit quantum state $|\psi_x\rangle$ as shown:

$$|\psi_x\rangle = \sum_{i=1}^{N} x_i |i\rangle$$

where x_i is the i-th element of x that can have different numeric data types, and $|i\rangle$ represents the i-th computational basis state. The procedure steps are the following: (a) normalize x, (b) calculate the corresponding amplitude embedding, (c) considering D, is the result of concatenating the input examples x_m, (d) normalize the vector as $|\alpha|^2 = 1$, and (e) representation of D is then,

$$|D\rangle = \sum_{i=1}^{2^n} \alpha_i |i\rangle$$

with α_i elements of α. If the case that the number of amplitudes to encode, is less than 2^n, then we can pad non-informative constants to α. In addition, the number of amplitudes to be encoded is $N \times M$ which demands $n \leq log_2(N \times M)$. As such, if we considered 5 data points with 3 features for each data point, the result is $5 \times 3 = 15$ amplitudes which requires $n \leq log_2 15 = 4$ *qubits* which represents $2^4 = 16$ possible states.

Angle Embedding. In this method we apply rotations on the x-axis or y-axis or z-axis using quantum gates along with the values which we want to encode. Applying this embedding on a dataset D, means that the number of rotations will be the same as the number of features in D. Therefore, the N-dimensional sample takes N *qubits* to generate the set of quantum states. Each data feature is encoded into a corresponding *qubit* q_i via Pauli rotations.

Besides these encoding methods, we can also have: **Displacement Embedding** this technique encodes N features into displacement amplitudes r or phase ϕ of M modes, **IQP Embedding** this encodes n features into n *qubits* using diagonal gates of the IQP circuit, **QAOA Embedding** encodes N features into $n > N$ *qubits* using a layered trainable quantum circuit based in QAOA ansatz, and, **Squeezing Embedding** which encodes N features into the squeezing amplitudes $r \geq 0$ or phase $\phi \in [0, 2\pi)$ of M modes. More on the topic e.g. **Trainable Embedding** can be found here [127], since we can also train the encoder circuit using machine learning techniques. Resorting to one of these encodings we implemented an algorithm for quantum binary classification (B.1) which is discussed later.

3.2 Adiabatic Model Mapping

The best known adiabatic model can be found at the D-Wave Systems and some hard problems can benefit from it since can be mapped directly to a Quantum Processing Unit (QPU). Problems from real world applications can be formulated as optimization problems, such as the TSP, searching for the optimal solution which can have multiple types of encodings [227].

Encoding using a Mathematical Formulation. As an example, we implemented (A.1) the TSP based in Hopfield and Tank [87], with the following energy function 3.1:

$$E = \frac{A}{2}\Big[\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1,k\neq j}^{n} v_{ij}v_{ik}\Big] + \frac{B}{2}\Big[\sum_{j=1}^{n}\sum_{i=1}^{n}\sum_{k=1,k\neq i}^{n} v_{ij}v_{kj}\Big] + \frac{C}{2}\Big[\sum_{i=1}^{n}\sum_{j=1}^{n}(v_{ij}-n)^2\Big] + \frac{D}{2}\Big[\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\sum_{k=1}^{n} dist_{ij}v_{ik}(v_{jk+1}+v_{jk-1})\Big] \tag{3.1}$$

Where *dist* corresponds to the distance between cities, v represents a Boolean variable, and n the number of cities. We considered $A = B - (D \times dL)$, $B = (3 \times dU) + C$, $D = 1/dU$ and $C = 1$, and $dL = dU/2$ with dU equals to the sum of edge weights in the graph. With this formulation in QUBO we achieve solutions for the TSP for $n \leq 6$, we followed the pipeline shown in 3.1. Some references with the study of parameters A, B, C and D were analyzed [131, 160, 200].

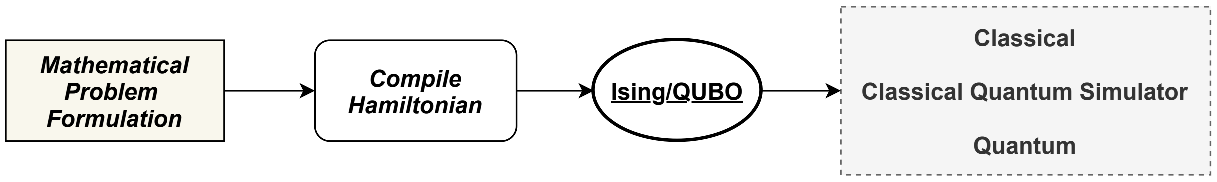


Figure 3.1: Pipeline of the typical approach to solve problems formulated in a mathematical manner translated to a Ising or Quadratic Unconstrained Binary Optimization (QUBO) model.

Regarding the latest news of the D-Wave's quantum anneling systems, more *qubits* were added and their connectivity was changed. However, even though from the 2000Q to the Advantage system, there have been advances at QPU level, is still hard to embed problems with large number of variables into the physical *qubits*. In order to overcome this issue, one of possibility is to resort to a solver such as *QBSolv*.

QBSolv. A decomposing solver which finds a minimum value of a large QUBO problem [211]. *QBSolv* can break the problem into smaller problems, solve these subproblems, and from that solutions build the answer of the proposed problem. The subproblems are solved using a classical solver running the tabu algorithm [157].

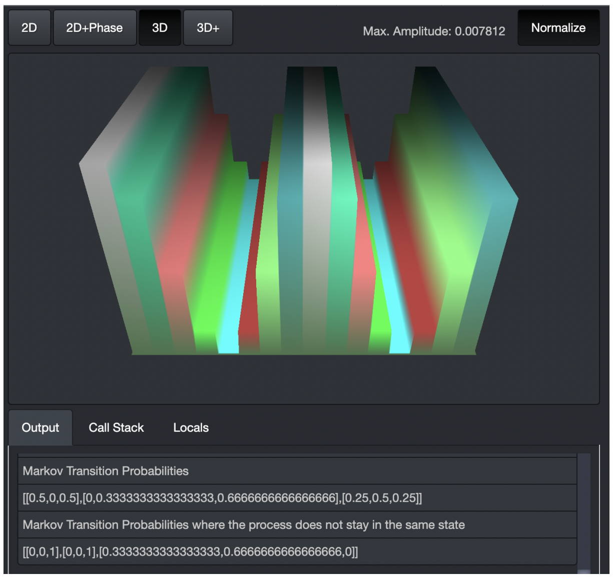
Algorithm 5: TSP Hamiltonian (PyQUBO) for mathematical formulation

```
function exp1 ($n, v, A$);
Parameters: Number of cities, binary vector, A
Expression : 1
for $i$ to $n$ do
    for $j$ to $n$ do
        for $k$ to $n$ do
            if $k \neq j$ then
                $exp+ \leftarrow v(i,j)v(i,k)$;
            end
        end
    end
end
$exp \leftarrow (A/2) \times Constraint(exp)$;
return $exp$
function exp2 ($n, v, B$);
Parameters: Number of cities, binary vector, B
Expression : 2
for $j$ to $n$ do
    for $i$ to $n$ do
        for $k$ to $n$ do
            if $k \neq i$ then
                $exp+ \leftarrow v(i,j)v(k,j)$;
            end
        end
    end
end
$exp \leftarrow (B/2) \times Constraint(exp)$;
return $exp$
function exp3 ($n, v, C$);
Parameters: Number of cities, binary vector, C
Expression : 3
for $i$ to $n$ do
    for $j$ to $n$ do
        $exp+ \leftarrow (v(i,j) - n)^2$;
    end
end
$exp \leftarrow (C/2) \times Constraint(exp)$;
return $exp$
function exp4 ($n, v, D$);
Parameters: Number of cities, binary vector, D
Expression : 4
for $i$ to $n$ do
    for $j$ to $n$ do
        if $j \neq i$ then
            for $k$ to $n$ do
                $exp+ \leftarrow dist(i,j)v(i,k)(v(j,k+1) + v(j,k-1))$;
            end
        end
    end
end
$exp \leftarrow (D/2) \times Constraint(exp)$;
return $exp$
H = exp1 ($n, v, A$) + exp2 ($n, v, B$) + exp3 ($n, v, C$) + exp4 ($n, v, dist, D$)
```

3.2.1 A generalized strategy to map classical problems to quantum hardware

The work on this thesis [186] was based on an empirical and exploratory search in the field of Quantum Computing (QC) and was conducted in accordance with the tools as they emerged.

Research #0. We started with an exploratory analysis, in 2017, using Google's open source quantum computing playground project [1]. We did some explorations with a work entitled "Simulating Markov Transition Probabilities in a Quantum Environment" [188]. We performed experiments with Markov chains by modeling them into a quantum algorithm capable of being simulated on a quantum computer. We used the Google Quantum Computing Playground, a GPU-accelerated quantum computer with a 3D quantum state visualization, a browser-based WebGL Chrome using the language QScript. We implemented a Markov chain assuming that the process is homogeneous in time and built the transition matrix by estimating Markov probabilities for the states. We modeled a transition in which the process does not stay in the same state, eliminating the self-loop transitions and normalizing the remaining probabilities.



```
Markov Transition Probabilities
[[0.5,0,0.5],[0,0.3333333333333333,0.6666666666666666],[0.25,0.5,0.25]]
Markov Transition Probabilities where the process does not stay in the same state
[[0,0,1],[0,0,1],[0.3333333333333333,0.6666666666666666,0]]
```

Figure 3.2: Result of script implementation in the Google's quantum playground. The global quantum vector state: height of the surface as amplitude, and color as phase.

Research #1. Afterwards, we moved to the IBM Q platform [2] which was in its first steps of existence. At that time we developed the work "Driven Tabu Search: A Quantum Inherent Optimisation" [189, 190]. We developed a driven quantum version of the Tabu search algorithm, which has been well understood for solving combinatorial or nonlinear problems. The experiments were implemented in the Python programming language using the Quantum Information Software Kit (QISKit) - a software development kit (SDK) for working with the Open Quantum Assembly Language (OpenQASM) and the IBM QX. We use as backend the ibmqx5 (16-qubits), and both high performance computing and classical simulator. In a quantum-metaheuristic procedure, we

[1]https://www.quantumplayground.net/#/playground/5652224143261696
[2]https://www.ibm.com/quantum-computing/

challenge the quantum search space to be maximized, derived from the advantage of interacting quantum technologies with classical implementations. In a quantum combinatorial optimisation, an entanglement-metaheurisc can uncover optimal solutions and accelerate the optimisation process by using entangled states. Therefore, we conduct simulation-based experiments with two types of quantum initial populations. The sample solutions are composed by 16 *qubits* of combined pairs - with non replacement, and with replacement. Until we obtain the best solution, we build the neighborhood, evaluate the system and detect whether the algorithm falls in a local optimum. In those cases, we perform the entanglement between *qubits* if they are unequal; otherwise, we use superposition in the redundant *qubit*. This particular feature of enhanced-entanglement showed best solution in *qubit* 1 and *qubit* 3, and a system evaluation score of 60. An enhanced-superposition in *qubit* 1 presented an evaluation score of 28, and it was the best solution found. The quantum inherent optimisation allows us to highlight the best solution and algorithm performance according to the input combined set.

Research #2. Later, we discover the Xanadu's Stranberry Fields and PennyLane software, and continue our gate-based model explorations with a work entitled "Quantum Binary Classification (Student Abstract)" [192]. We implement a quantum binary classifier where given a dataset of pairs of training inputs and target outputs our goal is to predict the output of a new input. The script is based in a hybrid scheme inspired in an existing PennyLane's variational classifier and to encode the classical data we resort to PennyLane's amplitude encoding embedding template. This work is later detailed in an upcoming section.

Research #3. This research work was similar to the previous, a gate-based approach resorting to the framework PennyLanne from Xanadu. In this work, we explore a quantum approach to a machine learning problem. Based on the work of Mari et al., we train a set of hybrid classical-quantum neural networks using Transfer Learning (TL). Our task was to solve the problem of classifying full-image mammograms into malignant and benign, provided by BCDR. Throughout the course of our work, heatmaps were used to highlight the parts of the mammograms that were being targeted by the networks while evaluating different performance metrics. Our work shows that this method may hold benefits regarding the generalization of complex data, however, further tests are needed. We also show that, depending on the task, some architectures perform better than others. This work is later detailed in an upcoming section.

Research #4. After attending several talks in the field, we decided to focus our work on adiabatic computing, bringing the work of Priscila Lima *et al.* to quantum computing, in particular D-Wave's QPU. In one of her talks Priscila Lima mentioned a modeling using SATyrus. We decided to embrace this method and start our proposal, which had as its starting point the work entitled "Mapping graph coloring to quantum annealing" [191]. This work introduces the mapping of a graph coloring problem based on pseudo-Boolean constraints to a working graph of the D-Wave Systems Inc. We start from the problem formulated as a set of constraints represented in propositional logic. We use the SATyrus approach to transform this set of constraints to an energy minimization problem. We convert the formulation to a quadratic unconstrained binary optimization problem (QUBO), applying polynomial reduction when needed, and solve the problem using different approaches: (a) classical QUBO using simulated annealing in a von Neumann machine; (b) QUBO in a simulated quantum environment; (c) actual quantum 1, QUBO using the D-Wave quantum machine and reducing polynomial degree using a D-Wave library; and (d) actual quantum 2, QUBO using the D-Wave quantum machine and reducing polynomial degree using our own implementation. We study how the implementations using these approaches vary in terms of the impact on the number of solutions found (a) when varying the penalties associated with the constraints and (b) when varying the annealing approach, simulated (SA) versus quantum (QA). This work is later detailed in an upcoming section.

Research #5. The next work "Mapping a Logical Representation of TSP to Quantum Annealing" followed the previous work and as a way to reinforce the proposed pipeline, making this method a generic method to be considered in the context of adiabatic computation. This work presents the mapping of the traveling salesperson problem (TSP) based in pseudo-Boolean constraints to a graph of the D-Wave Systems Inc. We first formulate the problem as a set of constraints represented in propositional logic and then resort to the SATyrus approach to convert the set of constraints to an energy minimization problem. Next, we transform the formulation to a quadratic unconstrained binary optimization problem (QUBO) and solve the problem using different approaches: (a) classical QUBO using simulated annealing in a von Neumann machine, (b) QUBO in a simulated quantum environment, (c) QUBO using the D-Wave quantum machine. Moreover, we study the amount of time and execution time reduction we can achieve by exploring approximate solutions using the three approaches. This work is later detailed in an upcoming section.

Resorting to Lima *et al.* formulation, the mapping of all constraints is represented by a energy function as shown in Equation 3.1 and derived from pipeline 3.1. As such, the general approach to solve problems formulated in propositional logic as explained here is represented by the generic pipeline shown in 3.3.

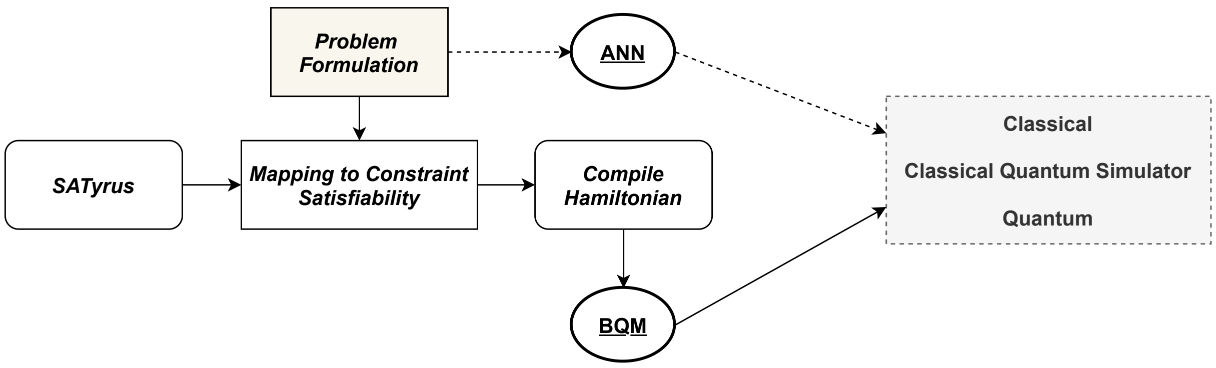


Figure 3.3: Pipeline of the general approach to solve problems formulated in QPL.

SATyrus. A neuro-symbolic architecture for modeling and solving optimization problems that uses logic, through mapping a problem specification into sets of pseudo-Boolean constraints. The goal is to transform a problem into a energy function using a logical declarative language that specifies and compile a target problem. The resulting energy function is then mapped into a higher-order Hopfield network of stochastic neurons to find its global minima [122].

ANN. Artificial Neural Networks, or ANN, is a data processing paradigm inspired by how the biological nervous system, such as the brain, processes data. It is composed of a large number of densely connected processing elements (neurons) that work together to solve a problem. More on ANN [86].

BQM. The D-Wave's solvers work on binary problems [3], and the inputs for these solvers use standard format known as a Binary Quadratic Model (BQM). The BQM encodes Ising (±1) and QUBO (0/1) models used by samplers such as the D-Wave system, represented by the following equation:

$$E(\mathbf{v}) = \sum_{\mathbf{i=1}} \mathbf{a_i v_i} + \sum_{\mathbf{i<j}} \mathbf{b_{i,j} v_i v_j} + \mathbf{c} \qquad \mathbf{v_i} \in \{\mathbf{-1}, \mathbf{+1}\} \text{ or } \{\mathbf{0}, \mathbf{1}\} \tag{3.2}$$

where a_i corresponds to the weight associated with each *qubit*, which influences the $qubit's$ to collapse into its two possible final states, and $b_{i,j}$ constant associated with each *coupler* (related to the connectivity), that controls the influence of one *qubit* on another.

In Figure 3.3 solid lines represent the path to achieve the classical, quantum classical simulator or quantum output by: *SATyrus* → *Problem Formulation / Mapping to Constraint Satisfiability* → *Compile Hamiltonian* → *BQM* or we can follow an alternative path, such as: *Problem Formulation* → *ANN*. By mapping of logical formulas into algebraic expressions generates an energy function representative of the problem. In that sense, any problem that is translated into logic can be solved using the pipeline shown in Figure 3.3. However, we should take into account the size of the problem since the mapping to the quantum machine depends on the number of variables and logical/physical *qubits* available [151].

This chapter described the mapping of applications to quantum hardware, from two perspectives: gate-based model mapping, namely, basis embedding, amplitude embedding, angle embedding and other forms of embedding; and also, adiabatic model mapping based on encoding using a mathematical formulation, and a final generalized strategy to map classical problems to quantum hardware, with a previous roadmap of research works that allow us to achieve the final pipeline. In the next chapter, we discuss the results of the mapping applications to quantum machines. We start by the gate-based experiments with the quantum classification problems, and then proceed to the adiabatic model with the mapping for solving the graph coloring problem and the TSP through Quantum Annealing (QA).

[3] Currently, besides the problems defined by two output values {0,1}, is also possible to solve problems defined on discrete sets of output values, such as (A...Z), through the Discrete Quadratic Model (DQM) solver.

Chapter 4

Results and discussion

The results concerning this thesis are in the publications, which are presented in the next chapters. The thesis feature research results presented next are a summary concerning the quantum classification problems (Iris plant and traffic data, and also, breast cancer data) and mapping graph coloring and travelling salesperson problem to Quantum Annealing (QA). For each experiment, we will briefly explain the target problem, summarise the type of model (gate-based or adiabatic), the type of hardware to conduct the experiments (simulator or real quantum device) and the main findings.

4.1 Quantum Binary Classification

We use a quantum binary classifier applied to the Iris dataset and to a car traffic dataset. The goal is to see how well the quantum classifier is able to discriminate two classes. In the first example, corresponding to characteristics of the plant Iris and in the second to the traffic states meaning congestion or not congestion. To conduct the experiments we use a 2-*qubit* simulator and a gate-based model. We resort to an existing PennyLane's variational classifier, consider a training and validation set, encode the classical data using the PennyLane's *amplitude* encoding embedding template. Futhermore, we resort to the Adam gradient-descent optimizer and the *loss* was estimated iteratively. The experiments showed that the classes were well discriminated and that we attained high *accuracy* scores, which were also compared with a classical neural network.

4.2 Quantum Transfer Learning for Breast Cancer Detection

In this research work, also using the PennyLane's resources, we train a set of hybrid classical-quantum neural networks using Transfer Learning (TL). The goal was to solve the problem of classifying full-image mammograms into malignant and benign, provided by BCDR. The experiments showed that this method may hold benefits regarding the generalization of complex data, however, further tests are needed. We also show that, depending on the task, some architectures perform better than others. The classical residual neural network, without resourcing to TL, achieved a maximum of 67% accuracy while our other experiments using TL achieved 84%. To conduct the experiments we use a classical machine, a quantum simulator and a real quantum device working in a gate-based model environment. The results resorting to different approaches were then compared.

4.3 Mapping Graph Coloring to Quantum Annealing

We conduct experiments to map the graph coloring to QA, that in brief, aim at given n colors, find a way of coloring the vertices V of a graph G such that no two adjacent vertices are colored using same color. As such, we generate Erdös-Rényi graphs G with different graph interconnectivy: a fully disconnected graph, a disconnected graph, a connected graph, and a fully connected graph. We start with a sanity check for both classical and quantum implementations to ensure that results are valid. Then, we show the execution times of each set of experiments. Afterwards, we show the optimal solutions, possible solutions and non solutions varying α and β which allow us to oppose classical and quantum implementations and infer their behavior and expected energy values. At the end, we describe the problem mapped onto the Quantum Processing Unit (QPU) with examples from the quantum annealer. In these experiments, we followed the adiabatic model and resort to the D-Wave's *Chimera* graph. Our study was carried out taking into account a 5-nodes graph, with classical and quantum experiments, where both implementations obtained the same solutions for all graphs. For instance, in the sanity check part, the fully disconnected graph was colored with 1 color (optimal solution), the disconnected graph was colored with 2 colors (optimal solution), the connected graph was colored with 3 colors (also optimal solution) and the fully connected graph was colored with 5 colors (only solution). Furthermore, when varying the penalties α and β and using less safe bounds, our results show that the classical Quadratic Unconstrained Binary Optimization (QUBO) and the quantum annealer are capable of finding an optimal solution.

Although classical and quantum experiments, searched for solutions using the same number of iterations, the quantum experiments generated a higher number of optimal solutions. In that sense, the quantum annealing seems to work as a better heuristic search. Besides, in the classical experiments, we observe that keeping the same values for α and β does not have impact in the search for solutions. In addition, the quantum results, in general, showed a higher variation on the number of solutions (optimal and non optimal).

4.4 Mapping a Logical Representation of TSP to Quantum Annealing

Similar to the graph coloring problem, we conduct experiments to map the travelling salesperson problem to QA, where the goal is to calculate the minimum distances tours between cities for different number of cities n. As before, we show the results of the execution times of each set of experiments. Afterwards, we indicate the optimal solutions and possible solutions and expected energy standardized values of classical `C`, quantum simulator `C_Q_sim` and quantum hardware `Q` in optimal and approximate solutions perspective.

In addition, we describe the problem mapped onto the QPU with broken and non-broken solutions. With our approach, we were able to find optimal solutions until 7 cities. Since the method is heuristic-based, non-optimal solutions can be returned as solution, it was the case for the graph G of size 8. In these experiments, we followed the adiabatic model and resort to the D-Wave's *Pegasus* graph. In the case of the Simulated Annealing (SA) we use the minimum number of iterations (sweeps, sw) needed to obtain an optimal solution to each graph G, in order to reduce the execution times. That number iterations, varies according to the graph size.

To achieve the QA execution times rigorously we considered the QPU sample times, explained in another section later in the context of the TSP.

Results show that for every graph size tested we can always obtain at least one optimal solution. In addition, the D-Wave machine can find optimal solutions more often than its classical counterpart for the same number of iterations and number of repetitions. Execution times, however, can be some orders of magnitude higher than the classical or simulated approaches for small graphs. For a higher number of nodes, the average execution time to find the first optimal solution in the quantum machine is 26% (n=6) and 47% (n=7) better than the classical.

In addition, we calculate the approximate solutions, which are based in the distance of possible solutions to the optimal solution. We notice, in the execution times, that there is a quantum advantage in time for some graph sizes to achieve the optimal solution, bringing out the quantum advantage that can be possible for large graphs. However, the same does not occur if we are seeking for an approximate solution. In that sense, if an optimal solution is needed for a problem with a large number of nodes, we should considered a quantum device.

In short, this chapter described the main findings of each feature experiment conducted in this thesis. From the next chapter onwards, the articles relating to the experiences and results discussed here, will be shown in detail in each separate chapter.

Chapter 5

Quantum Binary Classification (Student Abstract)

Carla Silva, Ana Aguiar, Inês Dutra

in the Proceedings of the AAAI Conference on Artificial Intelligence, 2021

Abstract

We implement a quantum binary classifier where given a dataset of pairs of training inputs and target outputs our goal is to predict the output of a new input. The script is based in a hybrid scheme inspired in an existing PennyLane's variational classifier and to encode the classical data we resort to PennyLane's amplitude encoding embedding template. We use the quantum binary classifier applied to the well known Iris dataset and to a car traffic dataset. Our results show that the quantum approach is capable of performing the task using as few as 2 qubits. Accuracies are similar to other quantum machine learning research studies, and as good as the ones produced by classical classifiers.

5.1 Introduction

Near-term quantum devices involve random processes and are built to transform the input data following a unitary operation, gate operations and measurements and are described by a quantum circuit. A classical *bit* has a state of either 0 or 1 and is the smallest quantity of non-probabilistic information. The simplest possible quantum system (*2-state system*) can hold precisely one *bit* of information. However, the *qubit* (unit of quantum information) has two possible states $|0\rangle$ and $|1\rangle$ defined as a finite-dimensional quantum system that forms a computational basis - two basis states composed by two distinct quantum states that the *qubit* can be in physically. Fault-tolerant quantum computers use few physical *qubits* to encode each logical *qubit*. These *qubits* are also used for error correction where the logical information is encoded through the

Algorithm 6: Circuit node

```
1 procedure CIRCUIT(weights,features):
2     AmplitudeEmbedding($features, qubits = [0, 1], pad = 0.0, normalized$)
3     for $w$ in weights do
4         Layer($w$)
5     end
6     return Expectation value Pauli Z
```

Algorithm 7: Layer function

```
1 procedure Layer($w$):
2     Rot($w[0, 0], w[0, 1], w[0, 2], qubits = 0$)
3     Rot($w[1, 0], w[1, 1], w[1, 2], qubits = 1$)
4     CNOT($qubits = [0, 1]$)
```

relationship of the *qubits*, also known as entanglement. In this work we focus on a quantum binary classifier that resembles a multilayer perceptron [175, 178, 203].

5.2 Quantum Supervised Binary Classification

The inference is performed with the model by initializing a state preparation circuit encoding the input into the amplitudes of the quantum device, resorting to a model circuit U_θ with trained parameters. The optimizer uses a *loss* function and initial parameters. Let $\mathcal{X}$ be the inputs and $\mathcal{Y}$ the outputs (actual instance labels). Given a dataset $D = (x_1, y_1), ..., (x_M, y_M)$ with pairs of training inputs $x^m \in \mathcal{X}$ and target outputs $y^m \in \mathcal{Y}$ for $m = 1, ..., M$, our goal is to predict the output $y \in \mathcal{Y}$ of a new input $x \in \mathcal{X}$. The binary classification task on an N-dimensional real input space can be then defined using $\mathcal{X} = \mathbb{R}^N$ and $\mathcal{Y} = \{0, 1\}$.

5.2.1 Amplitude encoding and quantum classifier

The embeddings that we can found in PennyLane[1] are templates to encode *features* into a quantum state. The *amplitude* encoding, encodes 2^n features into the amplitude vector of n *qubits* with padded dimension 2^n, Algorithm 6. Our circuit layer (Algorithm 7) consists of rotations on one *qubit* as well as CNOTs that entangle the *qubit* with its neighbour, according the n *features* of the dataset D.

Inference is performed with the model $f(x, \theta) = y$ by initialising a *state preparation* circuit $S_\mathcal{X}$ encoding the input $\mathcal{X}$ into the amplitudes of the quantum device, resorting to a model circuit U_θ (parametrised unitary matrix) with classification parameters θ (trained by a variational scheme), and a single *qubit* measurement which gives the probability of the model predicting 0 or 1.

[1]https://pennylane.ai/

5.2.2 Gradient-descent optimiser

The optimiser uses a *loss* function and initial parameters, and through differentiation performs the gradient descent in order to choose a set of optimal hyperparameters for the learning algorithm. We resort to the well known Adam gradient-descent optimizer with adaptive learning rate, first and second moment $x^{(t+1)} = x^{(t)} - \eta^{(t+1)} \frac{a^{(t+1)}}{\sqrt{b^{(t+1)}+\epsilon}}$, with the update rules,

$$
\begin{aligned}
a^{(t+1)} &= \frac{\beta_1 a^{(t)} + (1-\beta_1)\nabla f(x^{(t)})}{(1-\beta_1)}, \\
b^{(t+1)} &= \frac{\beta_2 b^{(t)} + (1-\beta_2)(\nabla f(x^{(t)}))^{\odot 2}}{(1-\beta_2)}, \\
\eta^{(t+1)} &= \eta^{(t)} \frac{\sqrt{(1-\beta_2)}}{(1-\beta_1)}
\end{aligned}
$$

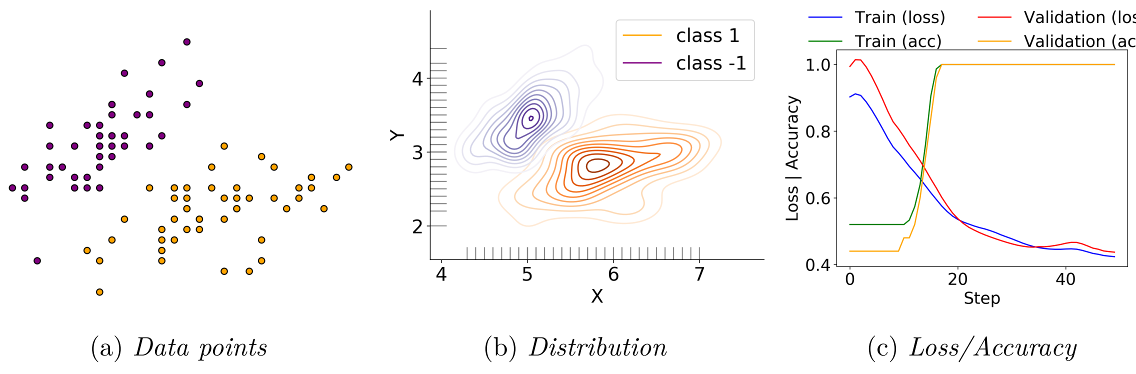
where $(\nabla f(x^{(t-1)}))^{\odot 2}$ refers to the element-wise square operation (each element in the gradient is multiplied by itself) and at start the first and second moment are zero. The *loss* function was built using the standard square loss that measures the distance between target labels and model predictions. The model optimises the weights such that the *loss* function is minimised.

5.3 Materials and Methods

In order to model a car traffic problem we search for a data set in an open source repository. The *Traffic* dataset was retrieved from the *R package* (LPCM), package to analyse traffic patterns. It concerns a *fundamental diagram* with observations of speed and flow from 9th of July 2007, 9am, to 10th of July 2007, 10pm, on Lane 5 of the Californian Freeway SR57-N, VDS number 1202263. The original car traffic dataset has a total of 444 samples. The *features* in the dataset are: Lane5Flow, Lane5Speed, Lane5Density. The output variable was created (Lane5congestion) based in two classes resorting to the *fundamental diagram* relationship $q - v$ where we calculate the critical point $v(q_{max})$ (max flow) to obtain the *critical speed* and assign 1 to values below the critical speed (meaning congestion) and -1, otherwise.

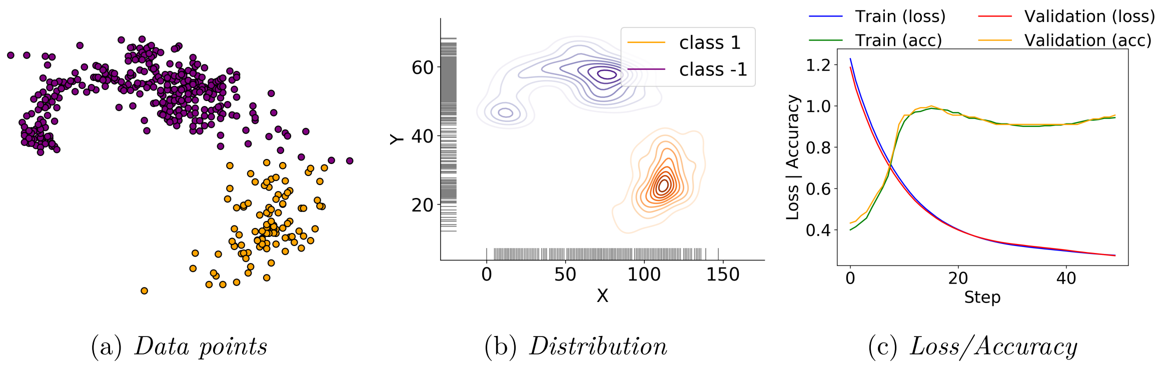
The *Iris* dataset was used in R.A. Fisher's 1936 paper. The original dataset has a total of 150 samples of three species of plants (50 of each). We create two splitted datasets (*Iris A* and *Iris B*), each one with 100 instances, and based in two classes. The *features* in the data sets are: SepalLength, SepalWidth, PetalLength, PetalWidth, and Species is the *target variable.* As is well known, class setosa distinguishes very well from the other two classes, so we should expect almost perfect accuracy results for *Iris A* since we only take the first two classes. Our goal is to map the proposed binary classifier onto quantum simulator, analyse whether the variation in the number of *qubits* has impact on the results obtained for accuracy in classification. Also, if the results obtained are very unlike if we use a simulator or a real quantum device.

5.4 Experiments



(a) *Data points* (b) *Distribution* (c) *Loss/Accuracy*

Figure 5.1: *Iris A*



(a) *Data points* (b) *Distribution* (c) *Loss/Accuracy*

Figure 5.2: *Traffic*

Figures 5.1 and 5.2 show data distribution and classification results for the *Iris A* and *Traffic*, namely, *loss* and *accuracy*, retrieved from the quantum simulator, considering the training and validation sets. We performed a comparison with a classical neural network which also resulted in *accuracy* equals to 1 for the *Iris A* dataset and 0.982 for *Traffic*.

The circuit parameters were adjusted to maximise the classification *accuracy* and minimise the *loss*. As such, a *gradient descent* is performed to adjust the circuit to minimise the *loss* function. The *loss* function is estimated by iteratively running the model to compare estimated predictions with respect to the *ground truth* (known values of y).

5.5 Conclusion and Perspectives

In this work, we resort to *amplitude encoding* where we use the *pad* argument for automated padding. In the future, we will explore other encoding classical data methods, e.g., *angle* which encodes N features into the rotation angles of n *qubits* where $N \leq n$, *variational/trained* or *higher order* embedding [80, 127].

Chapter 6

Quantum Transfer Learning for Breast Cancer Detection

Vanda Azevedo, Carla Silva, Inês Dutra

submitted to the Special Issue in Honour of Prof. Peter Wittek of the Quantum Machine Intelligence, 2021.
Note: The authors Vanda Azevedo and Carla Silva contributed to this research equally.

Abstract

One of the areas with the potential to be explored in Quantum Computing (QC) is Machine Learning (ML), giving rise to Quantum Machine Learning (QML). In an era when there is so much data, ML may benefit from either speed, complexity or smaller amounts of storage. In this work, we explore a quantum approach to a machine learning problem. Based on the work of Mari et al., we train a set of hybrid classical-quantum neural networks using Transfer Learning (TL). Our task was to solve the problem of classifying full-image mammograms into malignant and benign, provided by BCDR. Throughout the course of our work, heatmaps were used to highlight the parts of the mammograms that were being targeted by the networks while evaluating different performance metrics. Our work shows that this method may hold benefits regarding the generalization of complex data, however, further tests are needed. We also show that, depending on the task, some architectures perform better than others. Nonetheless, our results were superior to those reported in the state-of-the-art (84% against 76.9%, respectively). In addition, experiments were conducted in a real quantum device, and results were compared with the classical and simulator.

6.1 Introduction

It was in 1981 when Richard Feynman [61] first proposed a basic model for a quantum computer, tackling the inability of classical computers to simulate the physical world. In 1985, David Deutsch proposed the Quantum Turing Machine, thus formalizing a Universal Quantum Computer [49]. The potential of quantum computers was later proven by Peter Shor [185] in 1994. Shor developed an algorithm that solves in polynomial time prime factorization and discrete logarithms, an exponential advantage over known classical algorithms. Shortly after, in 1996, another major step was taken by Grover et al. [75], by developing an algorithm that finds a given value in an array in $\mathcal{O}(\sqrt{N})$ steps, where N is the array size. Classical algorithms will take a minimum of $N/2$ steps.

Since then, a lot of progress has been made, particularly in the past decade. Nowadays, there's an on-going investment by technology companies such as Google, Microsoft, IBM, or D-Wave, making it possible to access cloud-based quantum computers. One of the most promising applications of quantum computing is in ML. Depending on the model, computational complexity may be an obstacle. Recent studies, such as the one performed by IBM and MIT [80] or the one published by Biamonte et al. [18], conclude that ML could benefit from the exponentially large quantum state space through controllable entanglement and interference, bringing speed and efficiency to the process.

In ML, different types of neural networks models have been broadly explored, such as convolutional neural networks (CNN) or recurrent neural networks (RNN). A good example of such models for QML are CNNs. CNNs are one of the best algorithms in regards to image content identification and have shown exemplary performance in several tasks [89]. However, one of its known disadvantages is its complexity. The further we go into a neural network, the more complex are the features it can recognize since they end up aggregating and recombining. Improving the speed of these networks can have a huge impact when training models that require detailed images as input, such as mammograms.

Breast cancer is the type of cancer with the most incidence in women worldwide, with about 1.7 million new cases diagnosed in 2012, representing roughly 25 percent of all cancers in women. It is also the second most frequent cause of cancer death in women, after lung cancer [66]. In this research, we present an approach where we make use of a quantum method in the aid of breast cancer screening. Our proposal is inspired by the work of Mari et al. [133] on TL in hybrid classical-quantum neural networks. Our task was to classify full-image mammograms using pretrained classical neural networks resourcing to a quantum enhanced TL method. The advantage of the TL method, is that, it can be applied to a task where a model can be reused as the starting point on another task. In contrast to the work of Mari et al. [133], we evaluate large images, compare our results across different architectures, and, using the top performer, we apply a series of tests to evaluate the model against its classical counterparts, using different performance metrics. We conduct classical experiments without and with TL where we can noticed the advantage of resorting to TL, for example through accuracy 67% *versus* 84%, respectively. In addition, we resort to a quantum simulator varying the circuit depth from 1 to 4 where we also observe an accuracy of 84% for depth equals 1 and 4. Finally, we use a real quantum device where we observe an accuracy of 81%. These results are very promising, since we are in the noisy intermediate-scale quantum (NISQ) era [161], where quantum devices are associated with the need for error correction and still are in their early stage. Moreover, the quantum models version showed a faster training comparing to the classical version which is also quite interesting.

The research work is structured as follows: in section 6.2 we show the related work in the field, in section 6.3 we introduce the details on the QML topic. Section 6.4 describes the experimental setup. The summarizing of the data used, test of the proposed model, and results are in section 6.5. Finally, in section 6.6, we outline the resulting contribution and present some future work.

6.2 Related Work

In this section we summarize the research work on imaging on the topics QML, TL from a clinical perspective, and also, breast cancer screening using Deep Learning (DL). In order to compare the quality of the classifiers, we will be using common metrics to assess classification models. In what follows, we consider binary classification, where True Positive (TP) corresponds to the number of positive examples that are predicted as being positive, True Negative (TN) corresponds to negative examples that are predicted as negative, False Positive (FP) corresponds to a negative example that was predicted as a positive, and False Negative (FN) corresponds to a positive example that was predicted as a negative [109]. Based on these counters, the following metrics are defined:

- Sensitivity, Recall or True Positive Rate (TPR): measures the rate of actual positives that are acknowledged as positives by the classifier.

$$\frac{TP}{TP+FN}$$

- Specificity or True Negative Rate (TNR): measures the classifier's capacity to isolate negative results.

$$\frac{TN}{TN+FP}$$

- Precision: measures the rate of correctly classified instances for one class.

$$\frac{TP}{TP+FP}$$

- Accuracy: measures the amount of correctly classified instances of any class.

$$\frac{TP+TN}{TP+TN+FP+FN}$$

- F1-score: is the harmonic mean of precision and recall, with equal weights for both.

$$2 * \frac{precision * recall}{precision + recall}$$

- Area Under the Curve (AUC): AUC represents the degree of separability between classes. It makes use of Receiver Operating Characteristics (ROC) curve, which measures TPR, in the y-axis, against FPR (1-TPR), in the x-axis. AUC is a very common indicator of quality performance used in image classification.

6.2.1 Imaging in quantum machine learning

In an initial search on image processing algorithms for circuit-based quantum models, it is clear that there are several interesting papers solving tasks such as the algorithm proposed by Duan et al. [53] for dimensionality reduction, the quantum feature extraction framework proposed by Zhang et al. [225], the quantum representation of color digital images presented by Sang et al. [171] or the quantum image edge extraction algorithm proposed by Zhou et al. [68] based on improved Sobel Operator.

In regards to circuit-based Quantum Neural Networks, Henderson et al. [83] empirically evaluated the potential benefit of quanvolutional layers by comparing three types of models built on the MNIST dataset: CNNs, Quantum Neural Network (QNN), and CNN with additional non-linearities introduced, concluding that QNN models showed a faster training. Grant et al. [74] used MNIST, Iris, and a synthetic dataset of quantum states to compare the performance for several different parameterizations. Tang and Shu [202] used rough sets (RS) and QNN in order to recognize electrocardiogram (ECG) signals. Zhang et al. [223] proved that when compared to the random structure QNN, QNN with tree tensor (TT-QNN) architectures have gradients that vanish polynomially with the qubit number showing better trainability and accuracy for binary classification.

Skolik et al. [194] focus on solving the problem of barren plateaus of the error surface caused by the low depth of circuits by incrementally growing circuit depth during optimization and updating subsets of parameters in each training step. Kerenidis et al. [104] proposed a quantum algorithm for evaluating and training deep convolutional neural networks for both the forward and backward passes, providing practical evidence for its efficiency using the MNIST dataset. Kaur et al. [103] presented a novel ensemble-based quantum neural network in order to overcome the over-fitting issues present in speaker recognition techniques.

6.2.2 Transfer learning in clinical imaging

Several studies [38, 107, 183] support the use of TL using CNN for the analysis of medical images. Certain methods are more frequently employed according to the clinical object of study (e.g. brain, breast, eye, etc.), the image acquisition method (e.g. X-rays, ultrasound, or magnetic resonance, etc.), the depth of the sample, and the size of the dataset.

Although there is no validated proof of which method works best for a given clinical problem, Morid et al. [143] suggest that AlexNET is the most commonly CNN model used for brain magnetic resonance images [129, 209] and breast X-rays [116, 154], while DenseNET for lung X-rays [124, 219] and shallow CNN models for skin and dental X-rays [149, 224]. In addition, with smaller datasets, the most frequently applied TL approach is feature extracting, while fine-tuning is more used with larger datasets; additionally, data augmentation (e.g. translation, rotation of images), as a mean to feed more artificially generated samples to the CNN model in exchange of computational stress, is most frequently employed along with fine-tuning TL approach. Furthermore, studies using fine-tuning TL approaches use fully connected layers (contrary to traditional classifiers) more often than studies employing feature extracting TL approaches, which is due to the fact that training fully connected layers usually requires larger datasets when compared to training traditional classifiers.

In brief, the majority of studies do not benchmark their CNN model against any other model, and the few that actually do, compare against only one model [143]. Quantum transfer learning introduces low depth quantum circuits as a subroutine for a classical model. In this context, Mari et al. [133] propose a method that focuses on the paradigm in which a pretrained classical network is modified and amplified by a final variational quantum circuit.

Moreover, Acar et al. [3] applied the quantum transfer learning method, in different quantum real processors of IBM as well as in different simulators, in order to aid Coronavirus 2019 (COVID-19) detection by using a small number of Computed Tomography (CT) images as a diagnostic tool. Gokhale et al. [72] present an extension to the quantum transfer learning approach formerly applied to image classification in order to solve the image splicing detection problem. Zen et al. [222] proposed a method that evaluates the potential of transfer learning in order to improve the scalability of neural-network quantum states. In the current era of intermediate-scale quantum technology, Mari's method [133] is the most appealing as it allows optimal pre-processing of images with any state-of-the-art classical network and to process the most relevant features into a quantum computer.

6.2.3 Breast cancer screening using deep learning

The most popular datasets are the Mammography Image Analysis Society (MIAS) database with 322 image samples, the Digital Database for Screening Mammography (DDSM) containing 2500, and the Breast US Image with 250. Datasets can be used to extract Regions of Interest (ROIs) to perform detection and segmentation and ResNet variations as well as Inception, AlexNet, GoogleNet, and CaffeNet models have been used in breast cancer image analysis [48].

In the survey by Debelee et al. [48], we are presented with the available imaging modalities: Screen-film mammography (SFM), Digital mammography (DM), Ultrasound (US), Magnetic resonance imaging (MRI), Digital Breast Tomosynthesis (DBT) or a combination of modalities. The Digital Image category has been the most effective and commonly used breast imaging modality despite having some limitations which include low specificity, leading to unnecessary biopsies.

An interesting work by Shen [181] consists of an end-to-end training algorithm for whole-image mammograms projected to reduce the reliance on lesion annotations. Initially, it requires annotations to train the patch classifier consisting of ROI images, then a whole image classifier can be trained using only image-level labels. INbreast and DDSM datasets were used, showing that a whole image model trained on DDSM can be easily transferred to INbreast without using its lesion annotations and using a smaller amount of training data.

Studies, where transfer learning is used in breast cancer image analysis, can also be found; Mehra et al. [136] compared the results of three networks (VGG16, VGG19, and ResNet50) using BreakHis dataset, a dataset of histological breast cancer images, and applied data augmentation, using AUC, accuracy, and Accuracy-Precision Score (APS) as performance measures. The pretrained networks were used as feature extractors and the extracted features were used to train logistic regression classifiers. A similar work was presented by Kassani et al. [101]. Also with a histological dataset, de Matos et al. [47] used Inception-v3 as a feature extractor and a SVM classifier trained on a tissue labeled colorectal cancer dataset aiming to remove irrelevant patches. By doing so before training a second SVM classifier, this study shows that the accuracy improves when classifying malignant and benign tumors. Huynh et al. [92] used data obtained from the University of Chicago Medical Center, consisting of 219 lesions on full-field digital mammography images and 607 ROIs about each lesion. By comparing SVM classifiers based on the CNN-extracted image features and their computer-extracted tumor features in the task of distinguishing between benign and malignant breast lesions this study concluded TL is a great tool when there are no large datasets. It is worth mentioning that Li et al. [118] proposed a very interesting work building a fuzzy rule-based computer-aided diagnosis for mass classification of mammographic images using the Breast Cancer Data Repository (BCDR) dataset [128].

In brief, the specificity and sensitivity of screening mammography are reported to be 89–97% and 77–87%, respectively. These metrics describe the performance of the models with reported false positive rates between 1–29% and sensitivities between 29–97% [167]. Regarding BCDR - specifically BCDR-D01 and BCDR-D02, the digital image datasets - using a CNN structure and by undersampling the dataset in order to balance both classes, Hepsag et al. [84] obtained 62% accuracy, 75% training accuracy, 46% precision, 53% recall, and 51% F1-score. However, different results were achieved when selecting:

- masses: 88% test accuracy, 98% training accuracy, 86% precision, 90% recall, and F1-score of 88%,
- calcifications: 84%, training accuracy of 98%, precision of 91%, recall of 76%, and F1-score of 83%.

Cardoso et al. [32] use BCDR in a different context, using ROI to perform a cross-sensor evaluation of mass segmentation methods. Diz et al. [51] applied KNN, LibSVM, Decision Trees, Random Forest and Naive Bayes to BCDR and achieved 89.3 to 64.7% for classifying each class benign/malignant; 75.8 to 78.3% for classifying dense/fatty tissue and 71.0 to 83.1% for identification of a finding. Fontes et al. [63] achieved the maximum accuracy of 76.9%, AUC of 74.9%, sensitivity of 84.88%, and specificity of 64.91% by applying a customized variant of Google's InceptionV3 with the use of a Softmax classification layer as the output to perform the classification task.

However, there seems to be no studies where classical-quantum hybrid models are used to identify breast cancer in mammograms.

6.2.4 Transfer learning

Conventionally, deep neural networks need large amounts of labeled datasets and very powerful computing resources to solve challenging computer vision problems, e.g. feature extraction and classification. TL can be described as the improvement of learning a new task through the transfer of knowledge from a related (learned) task [153]. This technique only works if the model features learned from the first task are general. It can be done using labeled or unlabeled data [162]. From a pretrained model we can perform two types of tasks: (a) fine-tuning and (b) feature extraction.

In fine-tuning, we essentially retrain the model, updating most or all of the parameters having the pretrained weights as a basis. In feature extraction, we use the encoded features of the pretrained model to train the final layer's weights (classifier) from which we derive predictions and reshape it to have the same number of outputs as the number of classes in the new dataset and freeze all the other layers, meaning that the pretrained CNN model works as a fixed feature extractor.

Transfer learning from natural image datasets, such as ImageNet, whose library is comprised of roughly one thousand classes (e.g. dog, car, or human, etc.), to medical imaging, such as chest pathology x-rays with about 5-14 classes (e.g. edema, pleural effusion, or cardiomegaly, etc.), is often used to avoid training the whole model [143].

6.3 Quantum Machine Learning

In Quantum Computing (QC), instead of working with binary digits (bits) we work with quantum bits, where 0 and 1 can overlap in time. The qubit (unit of quantum information) has two possible states $|0\rangle$ and $|1\rangle$. In the context of QC, the Hilbert space represents an abstract vector space which allows e.g. a quantum *superposition* meaning that a physical system can be in more than one state simultaneously. The last few decades have seen significant advances in the fields of deep learning and quantum computing. As of today, huge amounts of data are being generated, pushing the interest surrounding the research at the junction of the two fields, leading to the development of quantum deep learning and quantum-inspired deep learning techniques. The upcoming topic, is based on Schuld's description [175].

6.3.1 Angle embedding

Angle embedding is an interesting approach in the context of neural networks. Assume a neural network and the network input $\theta = w_0 + w_1 * x_1 + ... + w_N * x_N$ is written into the angle of an ancilla or net input qubit, which will be entangled with an output qubit in some arbitrary state $|\psi\rangle$, $R_y(2v)\,|0\rangle \otimes |\psi\rangle_{out}$, where v corresponds to an angle and R_y corresponds to the rotation around the y-axis, shown in equation 6.1:

$$R_y(2v) = \begin{bmatrix} \cos v & -\sin v \\ \sin v & \cos v \end{bmatrix} \tag{6.1}$$

Knowing this, we have what is also known as a **quron** in the context of quantum neural networks:

$$R_y(2v)\,|0\rangle = \cos v\,|0\rangle + \sin v\,|1\rangle \tag{6.2}$$

The next step is to prepare the output qubit $R_y(2\varphi(v))\,|\psi\rangle$, we can do this using a nonlinear activation φ dependent on v that will rotate it.

6.3.2 QNN and variational circuits

A QNN is a machine learning model or algorithm that combines elements from quantum computing and artificial neural networks. Over the past decades, the term has been used to describe different ideas, ranging from quantum computers simulating the exact computations of neural nets, to general trainable quantum circuits that carry only little resemblance with the multi-layer perceptron structure. We will be focusing mainly on Variational Circuits since it is the one we will be using in this work, basing its description on the work of Mari et al. [133]. Increasingly, the term “quantum neural network” has been used to refer to variational or parameterized quantum circuits. Despite being mathematically different from the inner workings of neural networks, the term highlights the “modular” feature of quantum gates in a circuit, along with the use of tricks from training neural networks used in the optimization of quantum algorithms [56].

Classical-quantum hybrid approaches consisting of relatively low-depth quantum circuits called "variational algorithms" are the near term solution [17, 34, 132, 178]. We can define a quantum layer as a unitary operation that can be processed by a low-depth variational circuit, producing the output state $|y\rangle$, by acting on the input state $|x\rangle$ of n_q quantum subsystems (e.g., qubits or continuous variable modes).

$$\mathcal{L} : |x\rangle \rightarrow |y\rangle = U(w)\,|x\rangle$$

where w is an array of classical variational parameters. A quantum layer could be, for example, a sequence of single-qubit rotations followed by a fixed sequence of entangling gates. Notice that, unlike a classical layer, a quantum layer retains the Hilbert-space dimension of the input states. This fact is due to the fundamental unitary nature of quantum mechanics and should be taken into account when designing quantum networks.

A variational quantum circuit is a concatenation of q quantum layers, equivalent to the product of many unitaries parametrized by different weights:

$$\mathcal{Q} = \mathcal{L}_q \circ ... \circ \mathcal{L}_2 \circ \mathcal{L}_1$$

A real vector x needs to be embedded into a quantum state $|x\rangle$ in order to enter a quantum network, this can also be done, depending on x, by a variational embedding layer and applied to some reference state (e.g. ground state).

$$\mathcal{E} : x \rightarrow |x\rangle = E(x)\,|0\rangle$$

The following step is to apply single-qubit rotations to x. The embedding layer $\mathcal{E}$, unlike $\mathcal{L}$, maps from a classical vector space to a quantum Hilbert space. Conversely, the extraction of a classical output vector y from the quantum circuit can be obtained by measuring the expectation values of n^q local observables $\hat{y} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_{n^q}]$. This process can be defined as a measurement layer, that maps a quantum state to a classical vector:

$$\mathcal{M} : |x\rangle \rightarrow y = \langle x|\,\hat{y}\,|x\rangle$$

The full quantum network, including the initial embedding layer and the final measurement can be written as:

$$\mathcal{F} = \mathcal{M} \circ \mathcal{Q} \circ \mathcal{E}$$

The full network maps a classical vector space to a classical vector space depending on classical weights. Despite containing a quantum computation hidden in the quantum circuit, $\mathcal{F}$ is simply a black-box analogous to a classical deep network. However, there are technical limitations and physical constraints that should be taken into account: in a quantum network we don't have complete freedom when choosing the number of features in each layer, these numbers are often linked to the size of the physical system. Typically, embedding layers encode each classical element of x into a single subsystem where:

$$\#inputs = \#subsystems = \#outputs$$

This can be overcome by:

a. adding ancillary subsystems and discarding or measuring some in the middle of the circuit,

b. engineering more complex embedding and measuring layers,

c. adding pre-processing and post-processing classical layers.

6.4 Experimental Setup

Our method consists of a classical-to-quantum TL scheme, based on the one proposed by Mari et al. [133] where we assume two networks, A and B:

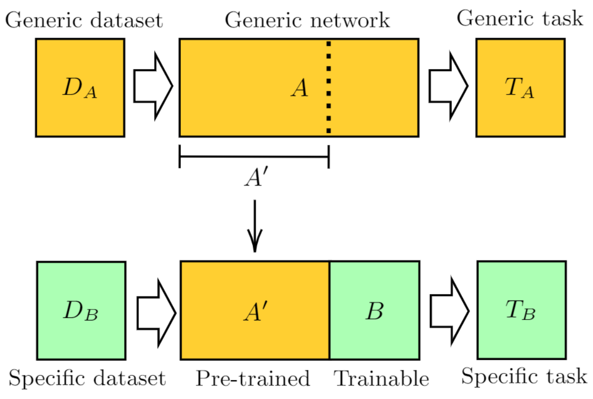


Figure 6.1: General representation of the transfer learning method [133]

where A' is the result of removing some of the final layers from network A, trained on dataset D_A to perform task T_A. A' will be used as a feature extractor for B. B is the network that we want to train using the new dataset D_B for some new task T_B. In our problem, this can be translated to:

- D_A: ImageNet, image dataset with 1000 classes.
- A: a pretrained ResNet18.
- T_A: classification, 1000 labels.
- A': a pretrained ResNet18 without the final linear layer, serving as an extractor of 512 features.
- D_B: BCDR, mammogram image dataset with 2 classes.
- B: DressedQuantumNet, a dressed quantum circuit with 512 input features and 2 real outputs, proposed by Mari et al. [133].
- T_B: classification, 2 labels.

We decided it would be interesting to perform an initial comparison between different convolutional network architectures to verify which one best fits our data. Therefore, we chose some of the most widely used architectures in the literature. Using the previous scheme as reference, network A will be replaced by **AlexNet** [110], **VGG19** [193], **DenseNet161** [89], and **ResNeXt50** [218] for the experiments. Further experiments will be done using the one with the best performance.

6.4.1 Dressed quantum circuit

The Dressed Quantum Circuit (DressedQuantumNet) consists of the classifier that will attach to the final linear layer of the pretrained model.

Assuming a Resnet18 is the pretrained model, the classifier will receive 512 real values as input to the circuit and will output 2 real values. The only trainable part of the network is the quantum classifier. Therefore, the number of trainable parameters can be calculated using the circuit's input size, its depth, and output size: $512 * n\_qubits + n\_qubits * depth + output\_size$. Trainable parameters' count will be presented along our results. The following represents the dressed quantum circuit:

$$\hat{\mathcal{Q}} = \mathcal{L}_{4\rightarrow 2} \circ \mathcal{Q} \circ \mathcal{L}_{512\rightarrow 4}$$

where $\mathcal{L}_{512\rightarrow 4}$ is a pre-processing layer that consists of an affine operation followed by a non-linear function $\varphi = tanh$ applied element-wise, Q is the variational circuit and $\mathcal{L}_{4\rightarrow 2}$ is a linear classical layer without activation (i.e.$\varphi(y) = y$).

The 4 real variables obtained from $\mathcal{L}_{512\rightarrow 4}$ are then embedded in the quantum circuit by applying a Hadamard gate(H) and performing a rotation around the y axis of the Bloch sphere parametrized by a classical vector x:

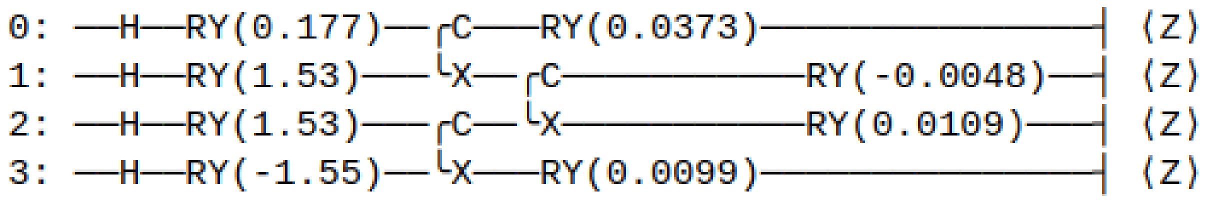
$$\mathcal{E}(x) = \bigotimes_{k=1}^{4} R_y(x_k \frac{\pi}{2}) H \left|0\right\rangle$$

The trainable circuit is composed of q variational layers $\mathcal{Q} = \mathcal{L}_q \circ ... \circ \mathcal{L}_2 \circ \mathcal{L}_1$ where:

$$\mathcal{L}(w) : \left|x\right\rangle \rightarrow \left|y\right\rangle = K \bigotimes_{k=1}^{4} R_y(w_k) \left|x_k\right\rangle$$

where K is an entangling unitary operation made of three controlled NOT gates.

Finally, the 4 output states are measured on the classical register using the Pauli-Z matrix and passed to the linear classical layer, producing 2 output states. The classification is done according to $argmax(y)$, where $y = (y1, y2)$ is the output of the dressed quantum circuit. An illustration of the circuit is shown in Figure 6.2.



```
0: ──H──RY(0.177)──╭C───RY(0.0373)──────────────────┤ ⟨Z⟩
1: ──H──RY(1.53)───╰X──╭C──────────────RY(-0.0048)──┤ ⟨Z⟩
2: ──H──RY(1.53)───╭C──╰X──────────────RY(0.0109)───┤ ⟨Z⟩
3: ──H──RY(-1.55)──╰X───RY(0.0099)──────────────────┤ ⟨Z⟩
```

Figure 6.2: Dressed Quantum Circuit

6.4.2 Overview

The implementation for this research was based on the one provided by Mari et al. [133]. Concerning the data, it was loaded using our own dataloader (BCDR), and transformed using random horizontal and vertical flips for data augmentation. The input channels are also normalized using mean values and standard deviations of ImageNet. In order to get reproducible results, a manual seed was set for the dataloader.

Hybrid classical-quantum model. We start by loading a pretrained model and managed the following considerations:

a. Since we want to use the pretrained network as a feature extractor, we need to freeze every layer so that the weights aren't updated during training, the only weights we will be updating are the classifier's.

b. We determine the size of the vector that will be entering DressedQuantumNet and set DressedQuantumNet as the model's classifier layer.

c. Cross-entropy was used as a loss function, Adam optimizer was selected to update the weights of the model at each training step, and a scheduler was set to decay the learning rate by *gamma* every *step size*.

d. Finally, we train the model.

6.5 Materials, Tests and Results

Model comparison tests were performed, classical without and with TL, also comparison using a quantum simulator and resorting to a quantum device available at IBM Quantum (https://quantum-computing.ibm.com/).

In this section we give a description of the data used and how it was handled, followed by testing, analysis and discussion of the obtained results. Tests were run on Intel Core i7-10510U CPU @ 1.80GHz×8 with 16GB RAM. The quantum experiments were conducted using the IBM Quantum device `ibm_lagos` which is detailed later.

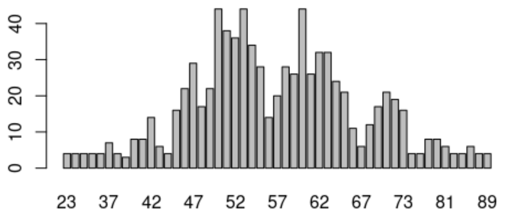
6.5.1 Materials, data description and processing

In this section we briefly describe the data and the needed processing for the data preparation in order to attain the achieved results. We used RStudio in this portion of our work. The data used was provided by BCDR [128]. According to their website [76], "the creation of BCDR was supported by the IMED Project (for Development of Algorithms for Medical Image Analysis) aimed at creating medical image repositories and massive exploration of Computer-Aided Diagnosis (CADx) methods on GRID computing resources. The IMED project was carried out by INEGI, FMUP-CHSJ – University of Porto, Portugal and CETA-CIEMAT, Spain between March 2009 and March 2013. Recently, in October of 2013, the IMED project was renewed and Aveiro University began to be part of this consortium. Now, the four institutions continue to actively augment and develop the BCDR."

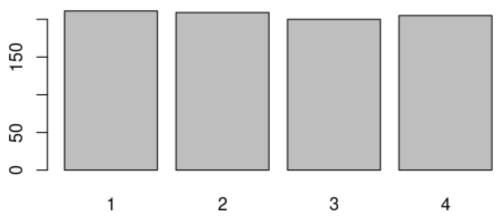
The data is composed of two folders (**BCDR_D01** and **BCDR_D02**) divided by study and patient, each containing several files corresponding to cranio-caudal (CC) and Medium-Lateral

Oblique (O) views of one or both sides - left (L) and right (R) - of a breast. The data also includes information, such as, the biopsy result. This new dataframe is composed of 49 columns and 2696 rows. From $image\ filename$ we cropped out image view and side and created a new column $view$. The new column $view$ was filled according to the following: $LCC = 1$, $LO = 2$, $RCC = 3$ and $RO = 4$.

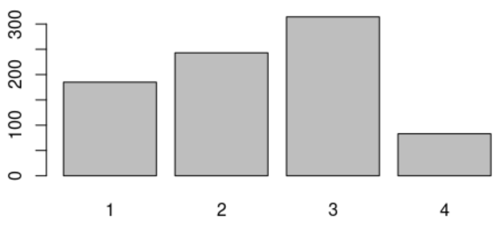
When looking for columns with at least 30% of NA values, we also made sure to trim every value since there were some white spaces in the dataset. Variables with a single class were also eliminated. Columns with 2 to 10 different values were converted to factor. We did the same for rows, eliminating those with at least 30% of NA values, no rows were found. Column $classification$ had a row with value 2, this row was eliminated. Finally, we changed $classification$ class $Malignant$ to *1* and $Benign$ to *0* and applied the **unique** function. The resulting dataframe contains 909 rows and 10 columns. Note that the same mammogram can contain the description of several different masses/calcifications. Knowing this, 269 mammograms contain nodules, 688 contain calcifications, 82 contain microcalcifications, 8 contain architectural distortions and 51 contain stroma distortions. Since we will be analyzing unique images, we eliminate duplicated $image\ filename$ from the dataset, and end up with 825 rows. To check if the images are well represented, we analyzed features age, $view$, $density$ and $classification$. Figure 6.3 shows data distribution of the features age, $view$, $density$ and $classification$.



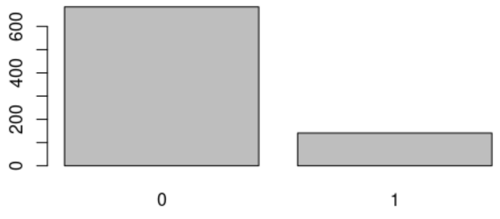
(a) Age distribution in *data*.



(b) View distribution in *data*(1=LCC, 2= LO, 3=RCC, 4=RO).



(c) Density distribution in *data*.

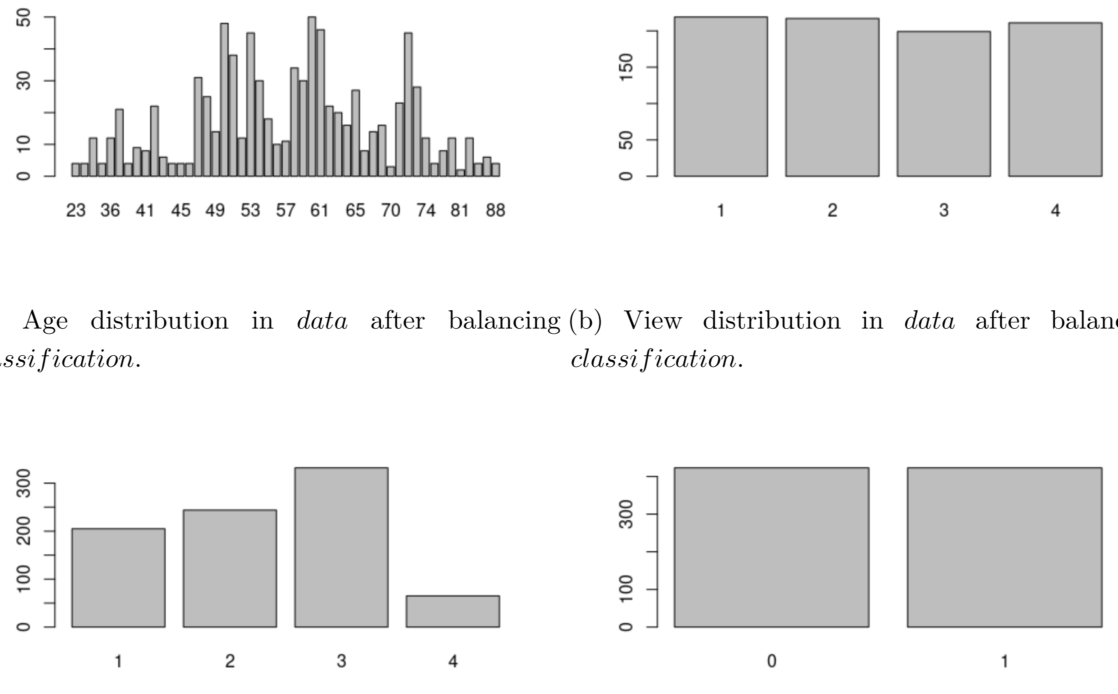


(d) Classification distribution in *data*.

Figure 6.3: Data distribution of the features age, $view$, $density$ and $classification$

As we can see, *classification* is unbalanced. Class 1 has 141 samples and class 0 has 684. We fixed this by oversampling class 1 by adding 2 copies of every sample with $classification = 1$ and cropping the excess samples with $classification = 0$. Each set of copies will be associated with the angle by which it will be rotated. We chose angle 60 for the first set of copies, and angle 45 for the second set. Since every image is subjected to random horizontal and vertical flips during train and test phases, we kept the angles to the first quadrant such that they don't result in the same images. The distribution, after balancing *classification*, can be seen below.

Figure 6.4 shows data distribution of the features *age*, *view*, *density* and *classification* after balancing the feature *classification*.



(a) Age distribution in *data* after balancing *classification*.

(b) View distribution in *data* after balancing *classification*.

(c) Density distribution in *data* after balancing *classification*.

(d) Classification distribution in *data* after balancing *classification*.

Figure 6.4: Data distribution of the features *age*, *view*, *density* and *classification* after balancing *classification*

6.5.2 Tests and results

In this chapter, we present:

- an initial model comparison using a classical approach in order to determine which of the models has the best performance. We also determine the proper learning rate for each of the models that will be further tested;

- tests and results using our QuantumDressedNetwork as a classifier resourcing to a quantum simulator in order to pick which hybrid classical-quantum model will be further tested;
- tests and results using a classical model without and with TL to benchmark against our hybrid model. Then a comparison between results obtained from these models;
- tests and results using our QuantumDressedNetwork as a classifier resourcing to a real quantum device.

Python and PennyLane platform were used in this portion of our work. Heatmaps which explicitly model the contributions of each pixel in the feature maps of a CNN to the final output were used to aid result comparison. These heatmaps result from using GradCAM [180], an earlier gradient-based visual explanation method, and GradCAM++ [37]. Note that these networks have different architectures and, therefore, the final convolutional layer, the one evaluated by GradCAM and GradCAM++, is different. Some of the functions were deprecated and were therefore changed manually.

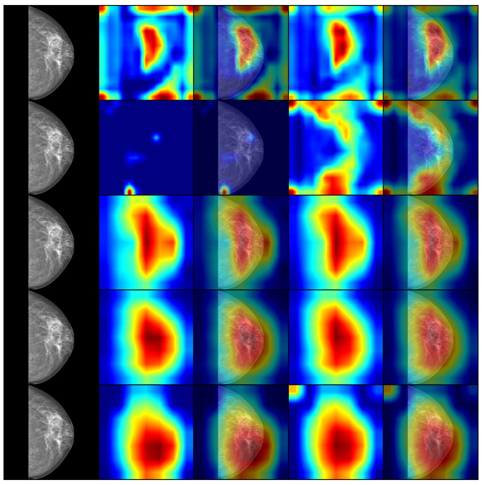
	step	# epochs	# batch size	gamma lr scheduler	step size
Model Comparison	0.4	30	32	0.1	5
Classical	0.0004	42	32	0.1	15
Quantum simulator	0.0004	42	32	0.1	15

Table 6.1: Parameters set for each experimental approach

We defined class 0 as the negative class as it represents a benign mammogram, and class 1 as a positive class as it represents a malignant mammogram.

In Table 6.1 we can observe the values of the parameters for the different approaches, where, *step* is the step of the learning rate, *# epochs* is the number of training epochs, *# batch size* is the number of samples for each training step, *gamma lr scheduler* is the learning rate reduction applied every 10 epochs and *step size* concerns the learning rate updates every step size epochs. *Model comparison* corresponds to the initial comparison of all the models and *Classical* to the parameters set in the classical without and with TL.

Model comparison. A positive case and a negative case were tested against AlexNet, VGG19,DenseNet161, ResNet18, and ResNeXt50_32x4d respectively. For simplicity, we used a simple linear transformation as classifier.


(a) Negative case

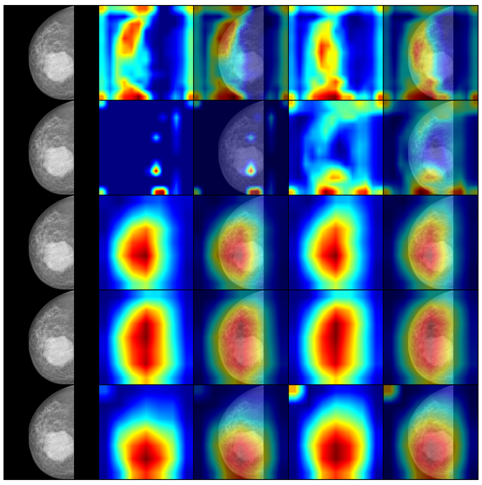

(b) Positive case

Figure 6.5: The rows in a) and b) show the results from AlexNet, VGG19, DenseNet161, ResNet18, and ResNeXt50_32x4d, respectively. The first column corresponds to the original image, the following two result from GradCAM (without image overlap and with image overlap) and the last two result from GradCAM++ (without image overlap and with image overlap).

In Figure 6.5, the warmer the color, the more the network is targeting the area. Knowing this, DenseNet, Resnet, and ResNeXt seem to do a pretty good job at targeting the main areas where masses/calcifications are present.

Now we need to have an idea of which learning rate to use. The starting learning rate is 0.4 and will be adjusted by the model by a power of -1 every 5 epochs. The goal is to find the optimal learning rate through the graphic, by choosing the global minimum, as suggested in the paper by Smith et al. [195] so we can give every model a fair chance to converge.

Table 6.2 shows the training results for DenseNet, ResNet and ResNeXt.

Model	DenseNet	ResNet	ResNeXt
Loss	[4.52,30.15]	[3.21,21.89]	[5.99,52.32]
Total parameters	26476418	11177538	22984002
Parameters trained	4418	1026	4098
Time (min)	87	16	55

Table 6.2: Training results for DenseNet, ResNet and ResNeXt. The interval corresponds to the minimum/maximum loss values obtained during testing phase.

The best learning rate should be the value that is approximately in the middle of the sharpest downward slope. Therefore, we have that DenseNet's, ResNet's, and ResNeXt's optimal learning rate is around 0.0004. We will keep $step\_size = 10$ because TL requires small learning rates regardless.

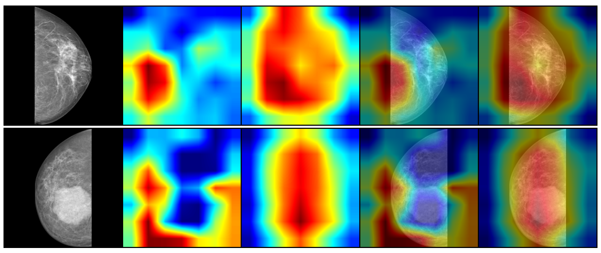
Quantum simulator. Using the best parameters mentioned we trained and tested DenseNet, ResNet and ResNext in a quantum simulator, with the QuantumDressedNetwork as a classifier for Resnet. We also tune the last parameter needed; the circuit's depth. According to Mari et al. [133], a characteristic of the TL approach is the existence of an intermediate optimal value for the quantum depth.

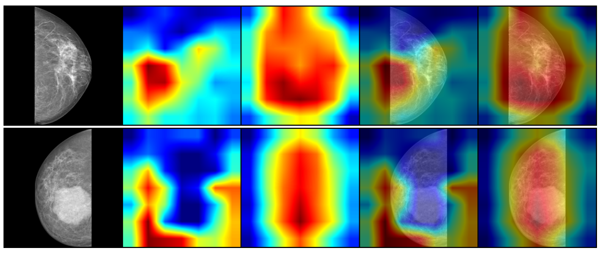
Model	DenseNet	ResNet	ResNeXt
Loss	[0.46,0.64]	[0.47,0.67]	[0.53,0.66]
Accuracy	[0.71,0.84]	[0.54,0.83]	[0.58 0.83]
Precision	[0.77,0.98]	[0.53,0.99]	[0.90,0.99]
Recall	[0.61,0.75]	[0.57,0.91]	[0.20,0.69]
F1-score	[0.68,0.81]	[0.67,0.81]	[0.33,0.80]
Specificity	[0.78,0.98]	[0.18,0.97]	[0.94,0.99]
AUC	0.69	0.75	0.59
Total parameters	26480854	11178582	22988118
Trainable parameters	8854	2070	8214
Time (min)	104	38	74

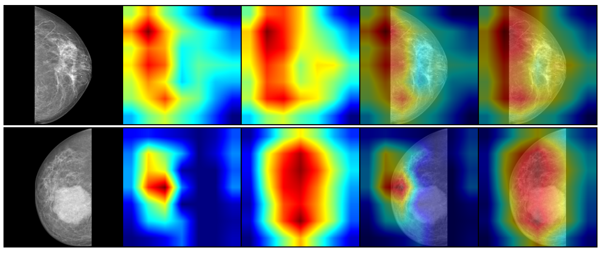
Table 6.3: Performance metrics results for DenseNet, ResNet and ResNeXt, for the quantum simulator. The interval corresponds to the minimum/maximum values obtained during testing phase.

Table 6.3 shows the performance metrics results for DenseNet, ResNet and ResNeXt, in the quantum simulator. Results are very similar for all three models. However, ResNet achieved a considerably higher Recall and AUC. Based on this, we will choose ResNet. Additionally, it is faster in comparison. Figure 6.6 shows results varying the circuit depth from 1 to 4.

Classical (without and with) TL. Figure 6.7a shows classical results without TL obtained from a ResNet, meaning that the network was trained from scratch. Figure 6.7b shows classical results with TL obtained from a ResNet, meaning that the only trained layers were the classifier's. These will be used as benchmarks to our model. For simplicity, we used a simple linear transformation as classifier.


(a) Circuit depth = 1


(b) Circuit depth = 2


(c) Circuit depth = 3

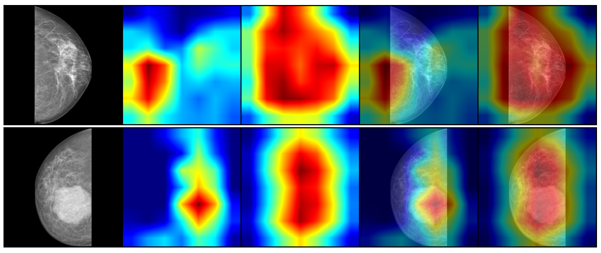
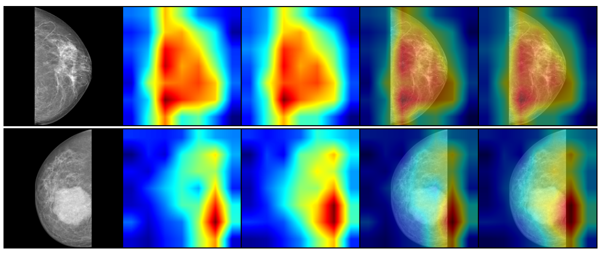

(d) Circuit depth = 4

Figure 6.6: The rows in a), b), c), and d) show the results for a positive and for a negative sample, respectively, using Resnet. The first column corresponds to the original image, the following two result from GradCAM (without image overlap and with image overlap) and the last two result from GradCAM++ (without image overlap and with image overlap).


(a) Classical without TL

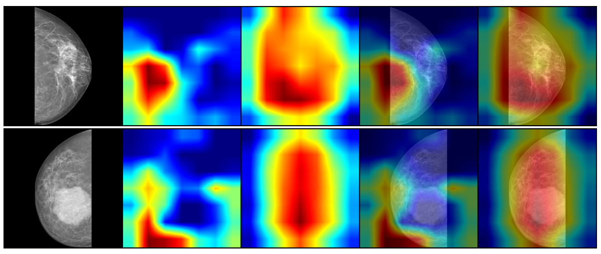

(b) Classical with TL

Figure 6.7: The rows in a) and b) show the results for a positive and for a negative sample, respectively. The first column corresponds to the original image, the following two result from GradCAM (without image overlap and with image overlap) and the last two result from GradCAM++ (without image overlap and with image overlap).

Let's consider the models:

- C: the classical ResNet.
- C+TL: the classical ResNet using Transfer Learning.
- CQ1, CQ2, CQ3 and CQ4: the hybrid classical-quantum ResNet models using Transfer Learning, whose variational quantum circuits have a depth of 1, 2, 3, and 4, respectively (using the simulator).

As mentioned before, performance metrics are calculated considering that the class malignant is the positive class.

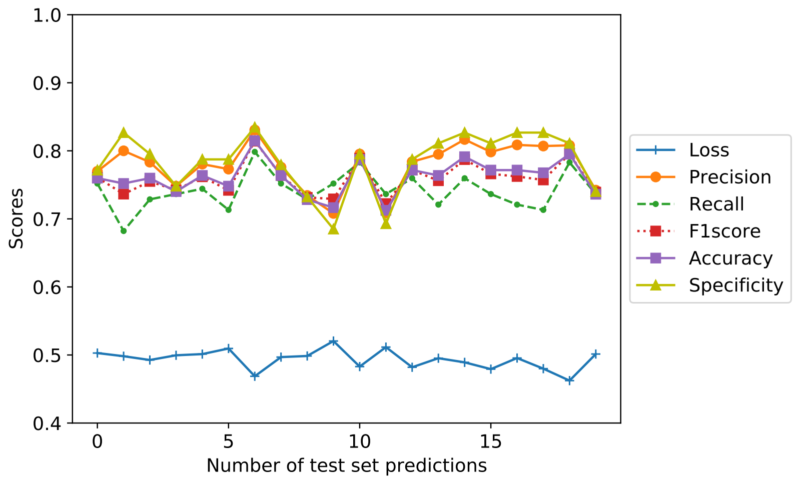
Model	C	C+TL	CQ1	CQ2	CQ3	CQ4
Loss	0.66	0.38	0.41	0.42	0.44	0.42
Precision	0.75	0.95	1.00	0.95	0.99	0.97
Recall	0.52	0.73	0.69	0.70	0.61	0.71
F1-score	0.61	0.82	0.82	0.83	0.80	0.82
Accuracy	0.67	0.84	0.84	0.83	0.83	0.84
Specificity	0.83	0.96	1.00	0.96	0.99	0.98
AUC	0.70	0.58	0.77	0.77	0.74	0.69
Trainable parameters	11689512	1026	2066	2070	2074	2078
Time (min)	48	24	40	51	69	82

Table 6.4: Performance metrics results for test set in all ResNet models tested. These results were on the same dataset, same train-test split and with a fixed seed. Results using the hybrid classical-quantum are executed in the simulator.

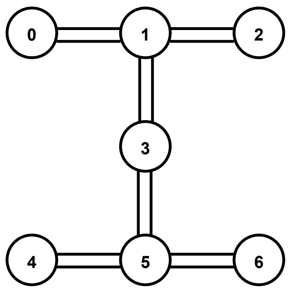
CQ1 has the higher overall results between training and test, namely precision, specificity and AUC score. It also has the lowest loss value. No further testing was needed on depth as the AUC value started to degrade. The resulting model is used on an IBM Quantum device for testing.

Quantum device. Due to the waiting times, IBM Quantum experiments were done using the best model state that resulted from training in the simulator. This means that only the test part was done in the quantum machine. We ran the file containing our network (that resulted from the Dressed Quantum Circuit with depth=1 - CQ1 - in the state where the best F1-score was measured), on an IBM Quantum device.

Figure 6.8 shows the quantum results achieved for the metrics under consideration resorting to the device `ibm_lagos`, for 20 repetitions of the testing in the same testset. As, in this experiment, we use data augmentation for each run, results are not the same at each repetition, but the model seems to be stable.



(a) Metrics for the test set on the IBM Quantum device



(b) `ibm_lagos` topology

Figure 6.8: Results for the quantum experiments on the IBM Quantum device

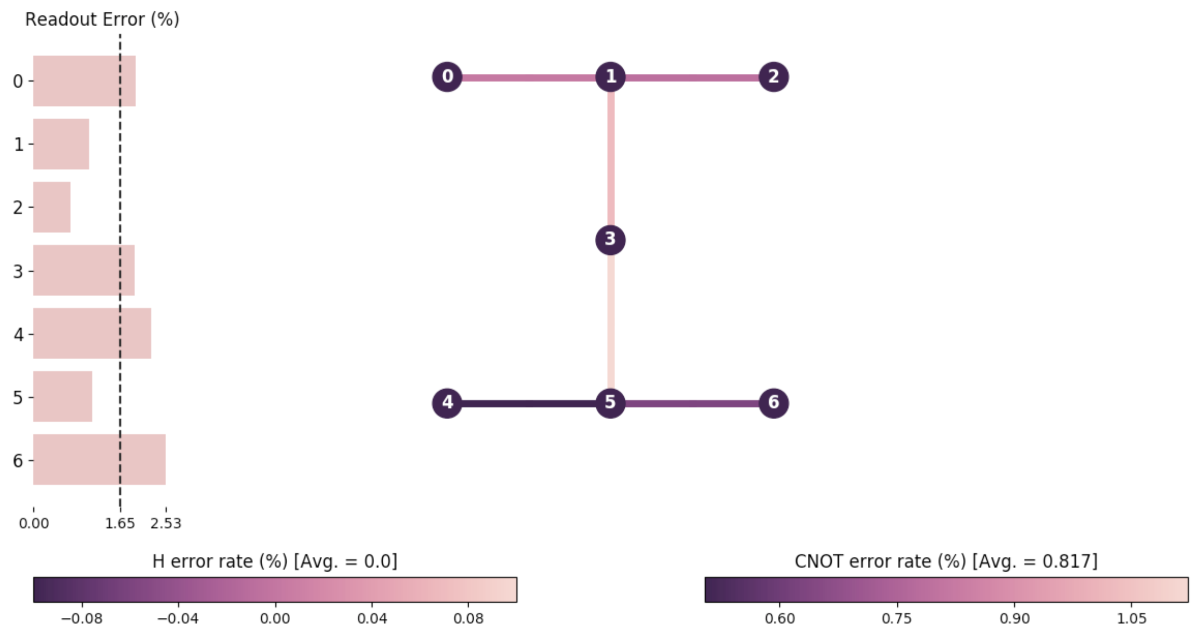


Figure 6.9: `ibm_lagos` error map

Loss	Precision	Recall	F1-score	Accuracy	Specificity	Time (hours)
0.47	0.83	0.80	0.81	0.81	0.84	43.7

Table 6.5: Performance metrics results for test set in the quantum device, using the best model found by the quantum simulator.

In Table 6.5 we show the metrics results for the best classification given by the quantum device among the 20 repetitions.

At first, we believed that the testing results variation seen in Figure 6.8a originated from the quantum machine error rates. The configuration of the qubits connectivity showing associated errors of the quantum device we use, `ibm_lagos`, is shown in Figure 6.9. The main characteristics of this device are: 7 qubits, 32 of quantum volume, a Falcon r5.11H processor type, version 1.0.1, the basis gates (CX, ID, RZ, SX, X), 6.595e-3 of average CNOT error, 1.466e-2 of average readout error, 106.39 us average T1 and 79.45 us average T2. The readout error of each qubit is shown on the left hand side. Errors can be as high as 2.53%. The average error for Hadamard and CNOT gates are also shown. As the errors are not too high to justitfy changes in performance results, we performed an experiment without data augmentation.

Using the resulting model from CQ1 as a base, and without performing any data augmentation techniques, we tested the same exact 32 samples (the size of one epoch):

- once in the quantum simulator;
- twice on *ibmq_quito* (due to queue size).

The results were exactly the same:

- 6 of our samples were positive but incorrectly classified as negative;
- 26 of our samples were negative and correctly classified as negative.

As expected, the results were the same for testing of the same samples. This leads us to believe that noise has little or no interference in our model and that the quantum simulator does a good job of simulating a quantum device.

6.6 Conclusion and Future Work

It is too soon to say whether or not QML can be an advantage. Our results only point to a good generalization of complex data which needs further testing. The classical residual neural network, without resourcing to TL, achieved a maximum of 67% accuracy while our other experiments using TL achieved 84%. When comparing classical ResNet with and without applying TL method we can see that the task execution time was cut by half. The results are not so drastic when comparing the classical-classical and the classical-quantum hybrid networks making use of TL. The results for every metric are far more consistent between training and testing and between epochs when using the dressed quantum circuit, so much so that AUC value went from 58% to 77%, leading us to believe that a hybrid-quantum network does present some advantage as AUC measures how well the model separates two classes. Our overall results surpassed those from the literature that also applied a classical transfer learning technique - accuracy of 84% against 76.9%, AUC of 77% against 74.9% and specificity of 100% surpassed 64.91%. However, regarding recall we obtained 69% and 84.88% was mentioned in the literature. Nevertheless, the results from the quantum device show a recall closer to that of the literature, 81%.

As is known in the community, there has been a lot of work in the field of error correction regarding quantum machines. However, despite the errors associated to the used quantum device `ibm_lagos`, good results were obtained with the quantum experiments. Our results seem to confirm what has been said in the quantum machine learning literature: the results seem to converge towards an optimal solution much earlier, which leads us to believe that such a large number of iterations in the quantum version will not be needed compared to the classical version.

As future work, an interesting approach could be to separate the dataset into masses and calcifications and only then perform classification. That way, we would be evaluating different specific features in a similar scenario and would probably get clearer results. It would also be interesting to test different datasets. We also would like to explore other gate-based quantum devices with smaller associated errors. Lastly, the best way to know how well this model performs would be to see how the data appears in our Hilbert Space. Ideally, both classes would appear very separately in it, as tight clusters.

In addition, in this research we conduct the experiments using the angle embedding technique since with this method the features can be learned. The same is not true if we use, for example, the basis embedding because it is theoretically impossible. In addition, if we had used e.g. the amplitude embedding, it could have been too complex to compute gradients with respect to features, which may not be feasible. However, in the future other embeddings can be considered [127].

Chapter 7

Mapping Graph Coloring to Quantum Annealing

Carla Silva, Ana Aguiar, Priscila M. V. Lima, Inês Dutra

in the Quantum Machine Intelligence, 2020

Abstract

Quantum annealing provides a method to solve combinatorial optimization problems in complex energy landscapes by exploiting thermal fluctuations that exist in a physical system. This work introduces the mapping of a graph coloring problem based in pseudo-Boolean constraints to a working graph of the D-Wave Systems Inc. We start from the problem formulated as a set of constraints represented in propositional logic. We use the SATyrus approach to transform this set of constraints to an energy minimization problem. We convert the formulation to a quadratic unconstrained binary optimization problem (QUBO), applying polynomial reduction when needed, and solve the problem using different approaches: (a) classical QUBO using simulated annealing in a von Neumann machine, (b) QUBO in a simulated quantum environment, (c) actual quantum 1, QUBO using the D-Wave quantum machine and reducing polynomial degree using a D-Wave library, and (d) actual quantum 2, QUBO using the D-Wave quantum machine and reducing polynomial degree using our own implementation. We study how the implementations using these approaches vary in terms of the impact on the number of solutions found (a) when varying the penalties associated with the constraints and (b) when varying the annealing approach, simulated (SA) versus quantum (QA). Results show that both SA and QA produce good heuristics for this specific problem, although we found more solutions through the QA approach.

7.1 Introduction

Quantum computing is an emerging field where science seeks to solve NP-hard problems more efficiently than classical algorithms, taking advantage of quantum phenomena such as entanglement and tunneling [112, 147]. Current technology has allowed the building of different quantum devices that can actually solve small problems. Practical experiments have been performed in these devices in order to better study their behavior and reliability. Some of these devices make use of the concept of Quantum Annealing (QA) [65], having the D-Wave family of computers as one of its representatives [94].

One important problem that has been solved by QA models is constrained optimization [82]. In fact, quantum approaches have been used to solve several applications of constrained optimization including circuit-based fault diagnosis, nurse scheduling, traffic optimization, portfolio optimization and vehicle routing [19, 58, 93, 97, 146, 206].

One particular type of constrained optimization is the one based on binary variables, i.e., each variable can assume one of only two possible values 0 or 1. We explore the logical constraint-based formulations of problems as used in the SATyrus approach [122], and explore the graph coloring problem. A pseudo-Boolean optimization [28] of our target problem is obtained by representing the hard combinatorial problems as pseudo-Boolean constraints based in the one presented in [123]. We compare a classical and a quantum implementation of the graph coloring modeled as boolean constraints using a Quadratic Unconstrained Boolean Optimization (QUBO) solver and a quantum solver based in symbolic/algebraic formulation with polynomial reduction. Our main contribution is twofold. First, we show how to map a SATyrus modelled problem to a quantum annealer. Second, we explore the quantum system dynamicity by varying the problem penalties.

According to Kudo [111], other approaches based on quantum annealing for graph coloring have been proposed. For example, in a QA approach resorting to the Ising model, real-time quantum simulation requires 2^{qN} dimensions to color a graph with N nodes and q colors. In a Constrained Quantum Annealing (CQA) approach, the dimension of the Hilbert space is reduced to q^N, meaning that the number of small energy levels is lower, as such, the dimension reduction achieves better performance. Also, the CQA approach [82] assigns spins to each node of the graph G, associating a multiple disjoint constraint (with a physical meaning concerning magnetization) to a *qubit*, which is 'up' if vertex i is colored with color k, and is 'down' otherwise, according to the objective function. The results show that while a penalty based embedding requires a fully connected graph, in the CQA approach less additional edges are required to satisfy the constraint.

The remainder of this paper is organized as follows: in the first section we present the background where we define (i) the mapping of the graph coloring to constraint satisfiability, (ii) the QUBO formulation, (iii) the polynomial reduction, and (iv) how adiabatic quantum computation works. The following section contains the implementations where we describe the classical graph coloring mapping and the quantum annealer mapping. Next, we show the experimental results with discussion. Finally, in the last section we conclude and suggest future work.

7.2 Background

Logic can be used to define a set of constraints of a problem which can be mapped by combining optimization problems into energy minimization in such a way that global minima correspond to the set of solutions to the original problem. In this section, we start by defining the constraints for the graph coloring problem and describe the techniques used to find solutions by energy minimization in a classical and quantum perspective.

7.2.1 Mapping combinatorial problems to constraint satifiability

One elegant way of solving problems is to specify the hard combinatorial problems as pseudo-Boolean constraints defining an energy landscape representing the space state of solutions of the target problem, as in Lima *et al.* [123]. From that representation we can use any solver to output a solution. Lima *et al.* describe how to map two well-known combinatorial problems: graph coloring and the traveling salesperson. First, these problems are represented as boolean constraints that are, in turn, mapped to an energy equation that corresponds to a higher-order Hopfield network, where the goal is to minimize the energy. From the energy equation, we can use any solver to find an optimal solution. Very briefly, we recall the mapping process for the graph coloring problem [123].

The graph coloring problem consists in assigning colors to certain elements of a graph subject to certain constraints. Let $G = (V, A)$ be an undirected graph, where V is the graph's vertex set and A the set of edges. For a boolean modelling, we may define each pair vertex-color and each color as a variable. The Integrity Constraints are then: (1) every vertex must have one color assigned to it, (2) two neighbouring vertices cannot have the same color, (3) a vertex cannot have more than one color, (4) if a color is assigned to a vertex, then the corresponding unit in matrix "colors" must be activated. Also, the Optimality Constraint is defined by (5) the number of activated elements in matrix "colors" must be minimum. As an example of logic representation and mapping, and assuming that vc_{ij} is a boolean variable representing that region i is colored or not colored with color j, the integrity constraint (1) can be represented as the disjunction $\forall i, \forall k \mid 1 \leq i \leq n, 1 \leq k \leq n : \vee(vc_{ik})$. Following the transformations defined by Lima *et al.*, we have the mapping of all constraints to the energy function as shown in Equation 7.1.

$$
\begin{aligned}
E = \beta\Big[\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{k'=1,k'\neq k}^{n} vc_{ik}vc_{ik'}\Big] + \alpha\Big[\sum_{i=1}^{n}\sum_{k=1}^{n}(1 - vc_{ik})\Big] + \\
\alpha\Big[\sum_{i=1}^{n}\sum_{i'=1,i'\neq i}^{n}\sum_{k=1}^{n} vc_{ik}vc_{i'k}neigh_{ii'}\Big] + \alpha\Big[\sum_{i=1}^{n}\sum_{k=1}^{n} vc_{ik}(1 - c_k)\Big] + \sum_{k=1}^{n} c_k
\end{aligned}
\tag{7.1}
$$

Where *neigh* corresponds to G's adjacency matrix, c is the color vector, and vc is a colored region in G. As QUBO does not allow objective function constraints, new terms can be added to the final QUBO quadratic expression to model the objective function through penalty terms [207]. Therefore, the constraints are associated to a penalty (Equation 7.2) where α is applied to the types (1, 2 and 4) constraints and β to the type (3) constraint, with h being a small number and n the number of colors which is the same as the number of vertices in the graph.[1].

$$\begin{cases} \alpha = (n * 1) + h \\ \beta = ((n^3 + n^2 + 1) * \alpha) + h \end{cases} \tag{7.2}$$

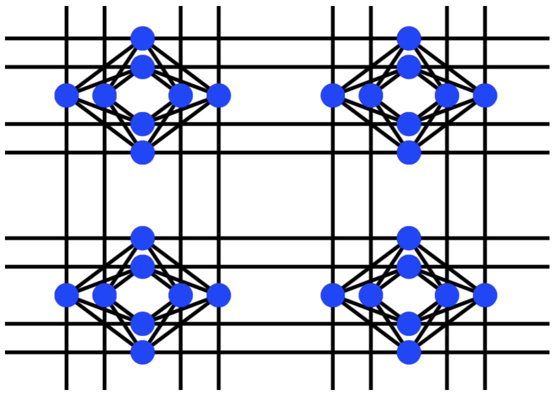
In their original work, Lima *et al.* represented and mapped the problem to be solved by a higher-order Hopfield network. In this work we will explore the QUBO solver, since there is an implementation of QUBO for the quantum device we use.

7.2.2 Adiabatic quantum computation

The term quantum annealing refers to the realization of the Adiabatic Quantum Computation (AQC), or quantum adiabatic optimization [102]. AQC can be performed using a commercially spin quantum accelerated annealing built by D-Wave Systems Inc [94]. A quantum annealing approach allows the system to be initialized and search for a neighbor solution gradually at the same time the system follows the changes, ending in a configuration that outputs one good solution, similarly to a simulated annealing, but in a quantum context. The goal is to find the global minimum of an objective function of candidate solutions resorting to thermal fluctuations in an energy landscape, which allows transitions between states in the process of searching for the global minimum [46, 99, 100, 108].

Current technology has enabled scalable quantum computing and allowed D-Wave Systems Inc to create *quantum processing units* (QPUs) with more than 2000 quantum bits (`DW_2000Q_5`). These can be accessed using Leap [96], a real-time quantum application environment. Figure 7.1) shows a quantum hardware graph (*Chimera*). A Chimera graph is composed of unit cells arranged in 8-vertex bipartite graphs (i.e. $K_{4,4}$). In quantum annealing, processors find low-energy solutions by seeking low energy states of a problem, i.e. a good solution. The process of finding a solution in a quantum annealer device is therefore a suitable approach to optimization problems and problems that require good low-energy samples (probabilistic sampling problems).

[1]The problem formulated this way assumes that there is a solution, suboptimal, that assigns one distinct color to each region in the map


(a) A 2 × 2 array of unit cells

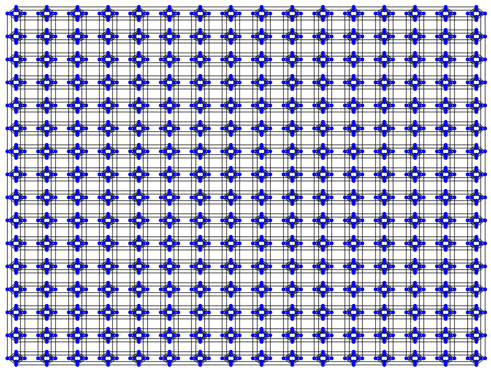

(b) A 16 × 16 array of unit cells

Figure 7.1: *Chimera* interconnection graph. Each cell is a 4 × 4 bipartite graph

The D-Wave quantum annealing processor has the goal to minimize the energy of an Ising/QUBO configuration whose pairwise interactions lie on the edges of a *Chimera* graph. Before QUBO problems are solved on quantum computers, the corresponding graph must be minor-embedded (or compiled) onto a *Chimera* graph [26, 43]. However, the complex behavior of the energy landscape sometimes leads to the chain break in the 2000Q *Chimera* when compiling (embedding) an optimization problem into the annealing hardware which can be a major bottleneck [73]. This may happen because a chained set of *qubits* can represent the same variable of the mapped graph (a vertex). During compilation, this chain may break (this occurs when the *qubits* of the same chain are not in the same state) due to compiler optimizations. To handle this problem, we may provide the compiler with a chain break fraction, a quantity defined per-sample of the fraction of chains which may be broken.

7.2.3 Quadratic unconstrained binary optimization

QUBO is an NP hard problem, used to solve many applications and suitable to algorithms implementation in quantum annealing [10, 204]. Adiabatic quantum computers can aid to solve a QUBO problem, using the quantum mechanical process called quantum annealing, therefore being able to solve any NP-hard problem like graph coloring [43]. In computation, NP-hard and NP-complete problems can be described through an Ising formulation, e.g., partitioning problems, binary integer linear programming, covering and packing problems, problems with inequalities, coloring problems, Hamiltonian cycles, tree problems, graph isomorphisms [130]. In this work, we formulate graph coloring as a quadratic unconstrained binary optimization problem. QUBO is a powerful mathematical tool that can aid in mapping a problem to a quantum annealing computer. The QUBO solution is mapped onto a physical *qubit* network structure with size and edge density restrictions [117]. In a QUBO [71] problem, variables can be *true* and *false* where states correspond to 1 and 0 values. The QUBO model is equivalent to the Ising model and can be translated to an Ising Hamiltonian by a linear transformation (Equation 7.3).

- In matrix notation, we intend to minimize the function:

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i<j} Q_{i,j} x_i x_j \tag{7.3}$$

  where the diagonal terms, $Q_{i,i}$, are the *linear coefficients* and the nonzero off-diagonal terms are the *quadratic coefficients* $Q_{i,j}$ and x_i and x_j are the variables.

- In scalar notation, the objective function is:

$$\mathrm{E}_{qubo}(a_i, b_{i,j}; q_i) = \sum_i a_i q_i + \sum_{i<j} b_{i,j} q_i q_j. \tag{7.4}$$

  where a_i are the *linear coefficients*, $b_{i,j}$ the *quadratic coefficients*, and q_i and q_j are the variables.

The problem to be solved by a D-Wave system is given by Equation 7.4 which also may represent an Artificial Neural Network with symmetric binary connections. In this case, q_i is the *qubit* i which participates in the annealing cycle and settles into one of the final states {0,1}, with *weight* a_i which influences the $qubit's$ tendency to collapse into its two possible final states, with q_i and q_j the coupler which allow one *qubit* to influence the other. $b_{i,j}$ is the same as the *binary weight* of Artificial Neural Networks [120], also strength, which controls the influence exerted by one *qubit* on another. To represent the graph coloring problem, we can then resort to the QUBO model by using a transformation in the node assignment constraints and using a transformation on the adjacency constraints. The built QUBO model has an objective function given by $x^t Q x$ where Q is determined by the problem formulation and each constraint. The terms impose the constraints for each vertex and provide energy penalty each time constraints are violated. In the way the combinatorial problems are formulated we can not directly apply the QUBO solver, given that some terms of the final expression are not quadratic. We then resort to a method to perform polynomial reduction to the non-quadratic terms.

7.2.4 Polynomial reduction

Polynomial reduction techniques translate higher order expressions to lower order. Translating higher order expressions to a quadratic formulation can be very tricky, but, fortunately, for our formulation, we only need to reduce terms from degree 3 to 2, which simplifies the problem (for an excellent discussion about degree reduction of polynomials, please check [198]). The reduction by minimum selection allows to lower our problem's polynomial degree explained by Equation 7.5:

$$xyz = \underset{w}{max}\{w(x+y+z-2)\} \tag{7.5}$$

where x, y, z are minimized binary variables in a pseudo-binary function and w is the ancillary binary variable indicating $true$ or $false$ status which are able to replace quadratic terms for

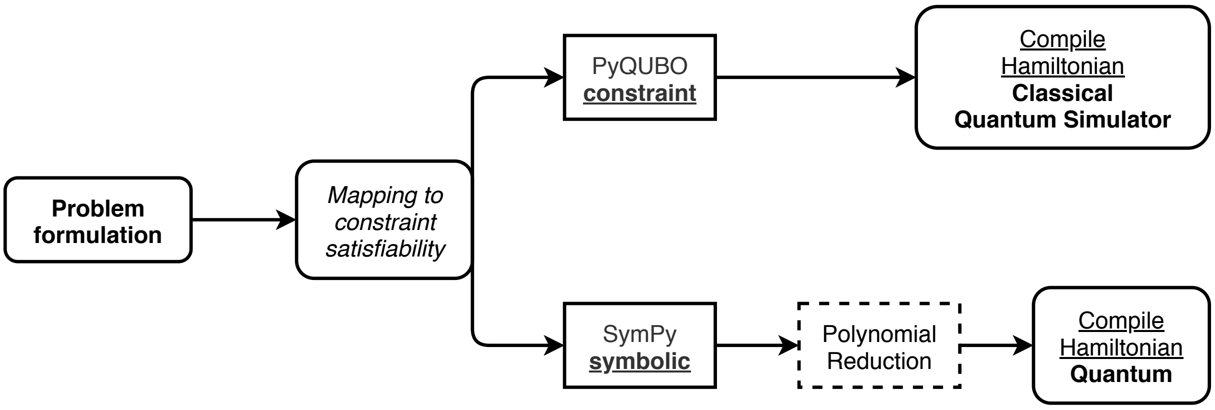

Problem formulation
Mapping to constraint satisfiability
PyQUBO constraint
Compile Hamiltonian Classical Quantum Simulator
SymPy symbolic
Polynomial Reduction
Compile Hamiltonian Quantum

Figure 7.2: Main scheme for the classical and quantum versions of our method

cubic terms. In that sense, to lower a higher-degree polynomial to QUBO, we can replace terms in the form of $axyz$, where a is a real number, according to Equation 7.6:

$$axyz = \begin{cases} aw(x+y+z-2) & a<0 \\ a\{w(x+y+z-1)+(xy+yz+zx)-(x+y+z)+1\} & a>0 \end{cases} \tag{7.6}$$

7.3 Methodology

Figure 7.2 shows the main steps of our methodology. We start with the boolean constraint formulation according to Lima *et al.*. We then transform the boolean expressions into algebraic expressions (all $\vee$ are replaced by arithmetical "+" and all $\wedge$ by "*", multiplication). We sum all together to obtain the energy function that will be minimized. We then apply polynomial reduction to the quantum version and feed the final expression to QUBO solvers. As the penalties used are "safe" (we use an upper bound assuming a maximum n), we empirically study how the use of less safe penalties affects the solution and violates constraints. The α's and β's can be over-estimated because, the first level of penalties derives from an upper bound of the objective function. The worse it is that upper bound, the bigger this first penalty is (usually named "alpha") and that "gap" is propagated to the other penalty levels, making the energy landscape more difficult to traverse in order to find global minima. This form of calculating penalties will be called "safe", as it guarantees the correspondence between global minima of the original problem and those of the energy equation. By contrast, if we relax this gap between the different penalty level values, we might still find global minima in the case where there are better (tighter) upper bounds for the target problem that could not be part of the formulation. In this case, we will name this value attribution to penalty levels as "unsafe".

Next, we present the algorithms used in our methodology. Code is available at [187].

Algorithm 8 takes the number of colors and h (small constant) and returns the multiplicatives

α and β. Algorithm 9 compiles the model based on the Hamiltonian H for the classical solver (PyQUBO [201]). Algorithm 10 prepares the graph coloring Hamiltonian built from symbolic computing (SymPy) to a quantum approach. Algorithm 11 implements the polynomial reduction.

D-Wave's Leap provides the function `make_quadratic` from the `dimod` library which allows to create a binary quadratic model from a higher order polynomial. In this work, we will use the term `Q_mq` to refer to the use of this library and `Q_ms` to refer to the use of polynomial reduction by minimum selection implemented by us. In both implementations, subexpressions `exp` correspond to a given problem constraint.

We use an *EmbeddingComposite* which fixes the chain using a default chain break algorithm. This is a problem-dependent issue, so we choose a value of *chainstrength* (a parameter of the algorithm) corresponding to the maximum value of the order of magnitude of the QUBO terms of the graph coloring problem. The *chainstrength* should be a value of the same order of magnitude as the rest of the terms in the QUBO/Ising, since, when the problem is scaled, it will guarantee that the forces keeping the member *qubits* will be comparable to all other forces [16, 30, 151].

The quantum experiments were conducted on the D-Wave 2000Q system, and the classical in a MacBook Air, 1.8 GHz Intel Core i5 processor, 8 GBytes of memory 1600 MHz DDR3. All times reported are the average of 10,000 iterations (for the PyQUBO simulated annealing iterations and for the D-Wave iterations). The SA number of run repetitions is 10,000. The number of iterations for each one of the 10,000 repetitions of the SA is 5,000. Besides, we should consider that the quantum annealer experiments have, in addition to the execution time, the travel time from the remote quantum machine to the local machine.

Algorithm 8: Graph Coloring penalties multiplicatives.

```
1 function alpha ($n, h$);
  Parameters: Number of colors, small constant
  Alpha     : Multiplicative
2 return $n + h$
3 function beta ($n, alpha, h$);
  Parameters: Number of colors, alpha, small constant
  Beta      : Multiplicative
4 return $((n^3 + n^2 + 1)alfa) + h$
5 $A \leftarrow alpha(n, h)$;
6 $B \leftarrow beta(n, A, h)$;
7 return $A, B$
```

7.4 Results and Discussion

In this section, we display the results beginning with a sanity check and the execution times of each set of experiments. Afterwards, we show the optimal solutions, possible solutions and non solutions varying α and β which allow us to oppose classical and quantum implementations and infer their behavior and expected energy values. At the end, we describe the problem mapped onto the QPU with examples from the quantum annealer.

Algorithm 9: Graph Coloring Hamiltonian (PyQUBO)

```
function exp1 ($n, vc, B$);
Parameters : Number of colors, binary vector, $\beta$
Integrity  : $Constraints$
for $i$ to $n$ do
    for $k$ to $n$ do
        for $k'$ to $n$ do
            if $k' \neq k$ then
                $exp \leftarrow exp + Constraint(vc(i,k) * vc(i,k'))$;
            end
        end
    end
end
$exp \leftarrow Constraint(B * exp)$;
return $exp$
function exp2 ($n, vc, A$);
Parameters : Number of colors, binary vector, $\alpha$
Integrity  : $Constraints$
for $i$ to $n$ do
    for $k$ to $n$ do
        $exp \leftarrow exp + Constraint(1 - vc(i,k))$;
    end
end
$exp \leftarrow Constraint(A * exp)$;
return $exp$
function exp3 ($n, vc, neigh, A$);
Parameters : Number of colors, binary vector, graph adjacency matrix, $\alpha$
Integrity  : $Constraints$
for $i$ to $n$ do
    for $k$ to $n$ do
        for $i'$ to $n$ do
            if $i' \neq i$ then
                $exp \leftarrow exp + Constraint(vc(i,k) * vc(i',k) * neigh(i,i'))$;
            end
        end
    end
end
$exp \leftarrow Constraint(A * exp)$;
return $exp$
function exp4 ($n, vc, A$);
Parameters : Number of colors, binary vector, $\alpha$
Integrity  : $Constraints$
for $i$ to $n$ do
    for $k$ to $n$ do
        $exp \leftarrow exp + Constraint(vc(i,k) * (1 - c(k)))$;
    end
end
$exp \leftarrow Constraint(A * exp)$;
return $exp$
function exp5 ($n$);
Parameters : Number of colors
Optimality : $Constraints$
for $k$ to $n$ do
    $exp \leftarrow exp + c(k)$;
end
return $exp$
H = exp1 ($n, vc, B$) + exp2 ($n, vc, A$) + exp3 ($n, vc, neigh, A$) + exp4 ($n, vc, A$) + exp5 ($n$)
```

Algorithm 10: Graph Coloring Hamiltonian (SymPy)

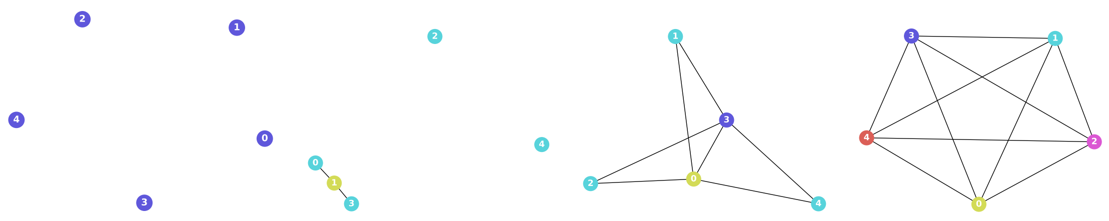
```
function exp1 $(n, vc, B)$;
Parameters : Number of colors, binary vector, $\beta$
Integrity   : Constraints
for $i$ to $n$ do
    for $k$ to $n$ do
        for $k'$ to $n$ do
            if $k' \neq k$ then
                $exp \leftarrow exp +$ algebraic constraint expression variable "vc" and indexes $i,\ k,\ k'$;
            end
        end
    end
end
$exp \leftarrow B * exp$;
return $exp$
function exp2 $(n, vc, A)$;
Parameters : Number of colors, binary vector, $\alpha$
Integrity   : Constraints
for $i$ to $n$ do
    for $k$ to $n$ do
        $exp \leftarrow exp +$ algebraic constraint expression variable "vc" and indexes $i,\ k$;
    end
end
$exp \leftarrow A * exp$;
return $exp$
function exp3 $(n, vc, neigh, A)$;
Parameters : Number of colors, binary vector, graph adjacency matrix, $\alpha$
Integrity   : Constraints
for $i$ to $n$ do
    for $k$ to $n$ do
        for $i'$ to $n$ do
            if $i' \neq i$ then
                $exp \leftarrow exp +$ algebraic constraint expression variables "vc", "neigh" and indexes $i,\ k,\ i'$;
            end
        end
    end
end
$exp \leftarrow A * exp$;
return $exp$
function exp4 $(n, vc, A)$;
Parameters : Number of colors, binary vector, $\alpha$
Integrity   : Constraints
for $i$ to $n$ do
    for $k$ to $n$ do
        $exp \leftarrow exp +$ algebraic constraint expression variables "vc", "c" and indexes $i,\ k$;
    end
end
$exp \leftarrow A * exp$;
return $exp$
function exp5 $(n)$;
Parameters : Number of colors
Optimality  : Constraints
for $k$ to $n$ do
    $exp \leftarrow exp +$ algebraic constraint expression variable "c" and index $k$;
end
return $exp$
H = exp1 $(n, vc, B)$ + exp2 $(n, vc, A)$ + exp3 $(n, vc, neigh, A)$ + exp4 $(n, vc, A)$ + exp5 $(n)$
```

Algorithm 11: Polynomial reductions: reduction by minimum selection.

```
function expression ($terms$);
Terms                     : Terms with degree 3
Polynomial reductions     : Expression with polynomial reductions
for $t$ to $terms$ do
    $term \leftarrow t$;
    if $t.count(" * ") = 3$ then
        $subterm \leftarrow t.split(" * ")$;
        if $subterm(0) > 0.0$ then
            $term \leftarrow subterm(0) + " * (w * (" + subterm(1) + subterm(2) + subterm(3) + " - 1.0)" + "(" + subterm(1) + " * " + subterm(2) + subterm(2) + " * " + subterm(3) + subterm(3) + " * " + subterm(1) + ") - (" + subterm(1) + subterm(2) + subterm(3) + ")" + "1.0)"$;
        end
        else
            $term \leftarrow subterm(0) + " * w * (" + subterm(1) + subterm(2) + subterm(3) + " - 2.0)"$;
        end
    end
    $exp \leftarrow exp + term$;
end
```

7.4.1 Sanity check and execution time

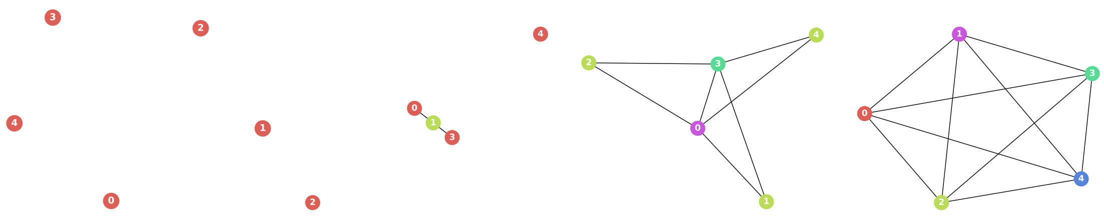
Before presenting further results, we perform a sanity check for both classical and quantum implementations just to guarantee that results are reliable. Following [168], we generate Erdös-Rényi graphs G and we experiment small graph coloring problems with different graph interconnectivy: a fully disconnected graph, a disconnected graph, a connected graph, and a fully connected graph. Results of coloring are shown in Figure 7.3 (classical coloring) and Figure 7.4 (quantum coloring). Both implementations obtained the same solutions for all graphs. The fully disconnected graph was colored with 1 color (optimal solution). The disconnected graph was colored with 2 colors (optimal solution). The connected graph was colored with 3 colors (also optimal solution) and the fully connected graph was colored with 5 colors (only solution possible in terms of number of colors). The results shown in Figures 7.3 and 7.4 were obtained with $\alpha = 5$ and $\beta = 755$ (obtained by using n as the number of colors, which is also the same as the number of vertices).



(a) Fully disconnected (b) Disconnected (c) Connected (d) Fully connected

Figure 7.3: Classical: sanity check for mapping the graph coloring

We concentrate our study in the 5-nodes graph of Figure 7.4(c). As this is a small problem it is trivial to check which solutions are optimal and which are not. For this graph, the optimal



(a) Fully disconnected (b) Disconnected (c) Connected (d) Fully connected

Figure 7.4: Quantum: sanity check for mapping the graph coloring

coloring number is 3. So, we checked all outputs for the conditions below and counted the number of (a) optimal solutions, (b) possible solutions and (c) non solutions. If we could find all optimal solutions for this problem, this would be 7200.

a. Optimal solutions

- $\exists c_i \in \{0,1\} : \sum_{i=0}^{4}(c_i = 1) = 3$
- $\exists vc_{ij}, vc_{ik} \in \{0,1\} : \sum_{i=0}^{4}\sum_{j=i+1}^{4}\sum_{k=0}^{n} \neg(vc_{ij} = 1 \wedge vc_{jk} = 1)$
- $\exists neigh_{ij}, vc_{ik}, vc_{jk} \in \{0,1\} : \sum_{i=0}^{4}\sum_{j=i+1}^{4}(neigh_{ij} = 1)\sum_{k=0}^{n}(vc_{ik} \neq vc_{jk})$

b. Possible solutions

- $\exists c_i \in \{0,1\} : \sum_{i=0}^{4}(c_i = 1) >= 3$
- $\exists vc_{ij}, vc_{ik} \in \{0,1\} : \sum_{i=0}^{4}\sum_{j=i+1}^{4}\sum_{k=0}^{n} \neg(vc_{ij} = 1 \wedge vc_{jk} = 1)$
- $\exists neigh_{ij}, vc_{ik}, vc_{jk} \in \{0,1\} : \sum_{i=0}^{4}\sum_{j=i+1}^{4}(neigh_{ij} = 1)\sum_{k=0}^{n}(vc_{ik} \neq vc_{jk})$

c. Non solutions

- $3. \notin 1. \wedge 2.$

In the conditions, c_i corresponds to the activated color i, vc_{ik} to a colored region i with color k, $neigh$ to the graph adjacency matrix and n is the number of colors activated. For optimal solutions we just need to check if the number of c_i variables with value set to 1 is 3 and check for inconsistencies (constraint violations) in the other variables. For non-optimal solutions, the number of c_i variables will be greater than 3, but we also need to check for inconsistencies. Non solutions are all that do not fit the conditions (1) and (2).

7.4.2 Varying α and β

In this work, the penalty values α and β are calculated using maximum values of n and therefore are safe. However, could we still achieve (optimal) solutions by changing these safe values? What are the values of α and β that do not allow finding an optimal solution or any solution? Can we visit a smaller part of the search space and still find an optimal solution by changing the values of α and β? Do the classical and quantum implementations find the same solutions and behave the same as we vary the values of α and β? In order to answer these questions, we performed various experiments decreasing the values of α and β. We pick the $G(5,3)$ with 3 colors and 5 vertices with connectivity of type (c) shown in Figures 7.3 and 7.4 and performed a first experimental part of a set of three with $\alpha = \{1, 5, 755\}$ and $\beta = \{1, 5, 755\}$, shown in Figures 7.5, 7.6 and 7.7.

When varying the penalties and using less safe bounds, our results show that the classical QUBO and the quantum annealer are always capable of finding an optimal solution, which answers our two first questions. In a second experimental part (shown in the Appendix) we again picked the $G(5,3)$ with 3 colors and 5 vertices, with connectivity of type (c) and decreased the values of α and β, ranging from a higher penalty to a lower as shown in Table 7.1. This Table shows the execution times, in microseconds, the differences between the classical and quantum execution times, and the number of returned solutions for the quantum solver, with the variation of α and β. The average execution time is 4046.1 μs for the classical runs (`C`), 1169.3 μs for the classical quantum simulator (`C_Q_sim`), 624.7 μs for the quantum annealing using the D-Wave quadratic library for polynomial reduction (`Q_mq`) and 606.7 μs for the quantum annealing using our implementation of polynomial reduction (`Q_ms`). For all the other experimental sets (varying only α and varying only β) we show the execution time in Tables 7.2 and 7.3.

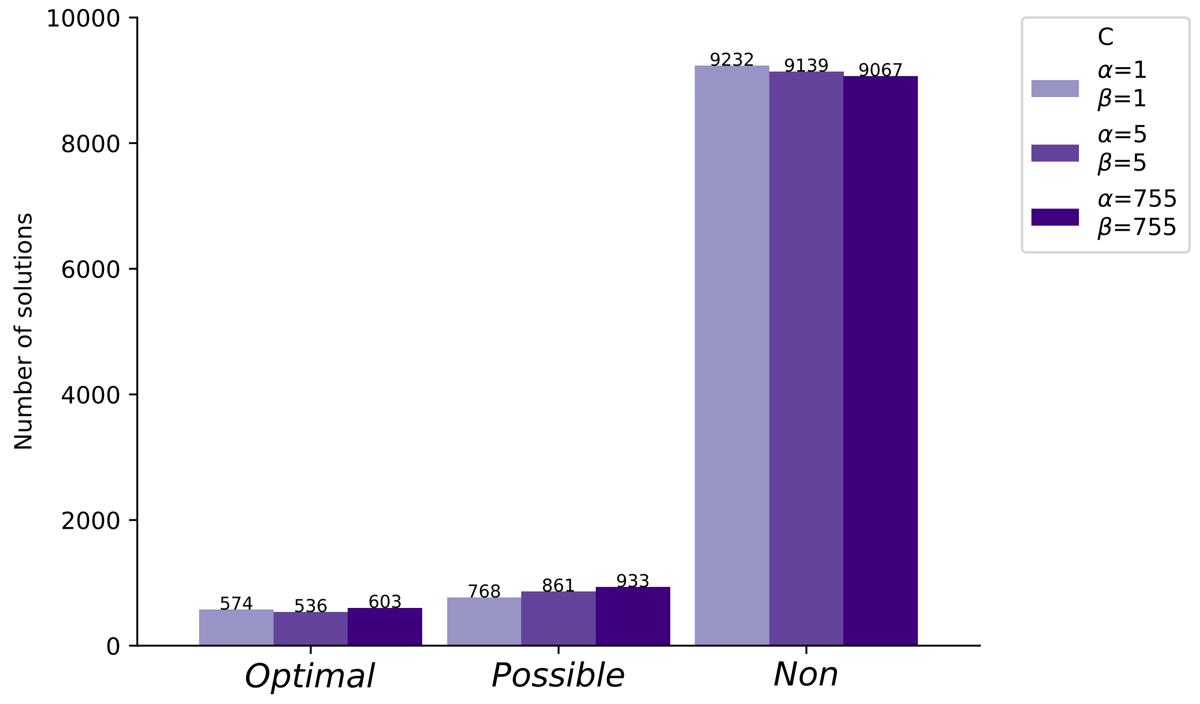
Table 7.1: Time (μs) of classical `C` and quantum `Q` experiments for graph coloring with 5 *colors* varying α and β

α	5	4	3	2	1
β	755	604	453	302	151
`C` (μs)	4137.8	4038.7	3980.5	3984.8	4088.5
`C_Q_sim` (μs)	1053.7	1140.3	1092.2	1379.9	1180.2
`Q_mq` (μs)	566.6	712.2	580.4	591.0	673.4
`Q_ms` (μs)	596.8	597.6	587.8	583.8	667.7

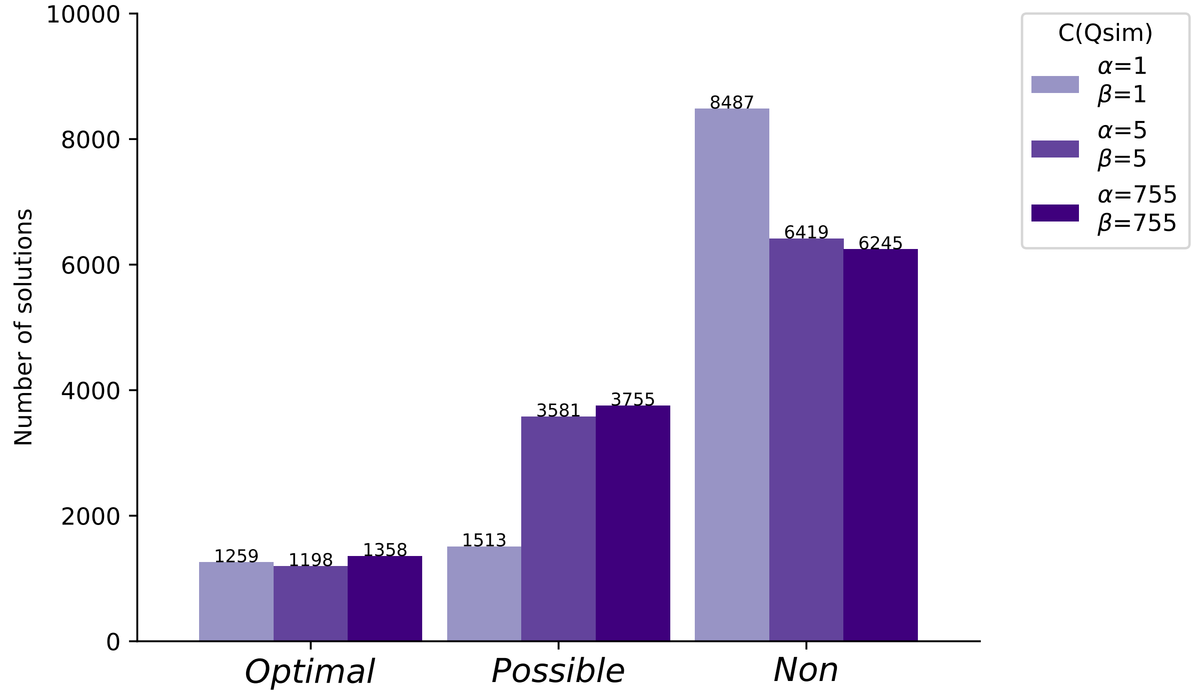
Set displayed in Table 7.1 is calculated according Equation 7.2 with α ranging from 1 to 5 and replacing in $\beta = ((n^3 + n^2 + 1) * \alpha) + h$ with h equal a small number, e.g. 0.0000005. In this work, we have as reference $\alpha = 5$ and $\beta = 755$ obtained as functions $\alpha = alpha(n, h)$ and $\beta = beta(n, \alpha, h)$ with n equal 5.

As we can observe from the plots, the experiments that ran in the D-Wave produced a higher number of optimal solutions. Both implementations, classical and quantum searched for solutions using the same number of iterations. These results seem to indicate that the quantum annealing searches for solutions in a better space. In that case, the quantum annealing seems to work as a better heuristic search.

For the classical implementations, keeping the same values for α and β does not seem to impact the search for solutions. The quantum versions have a higher variation on the number of

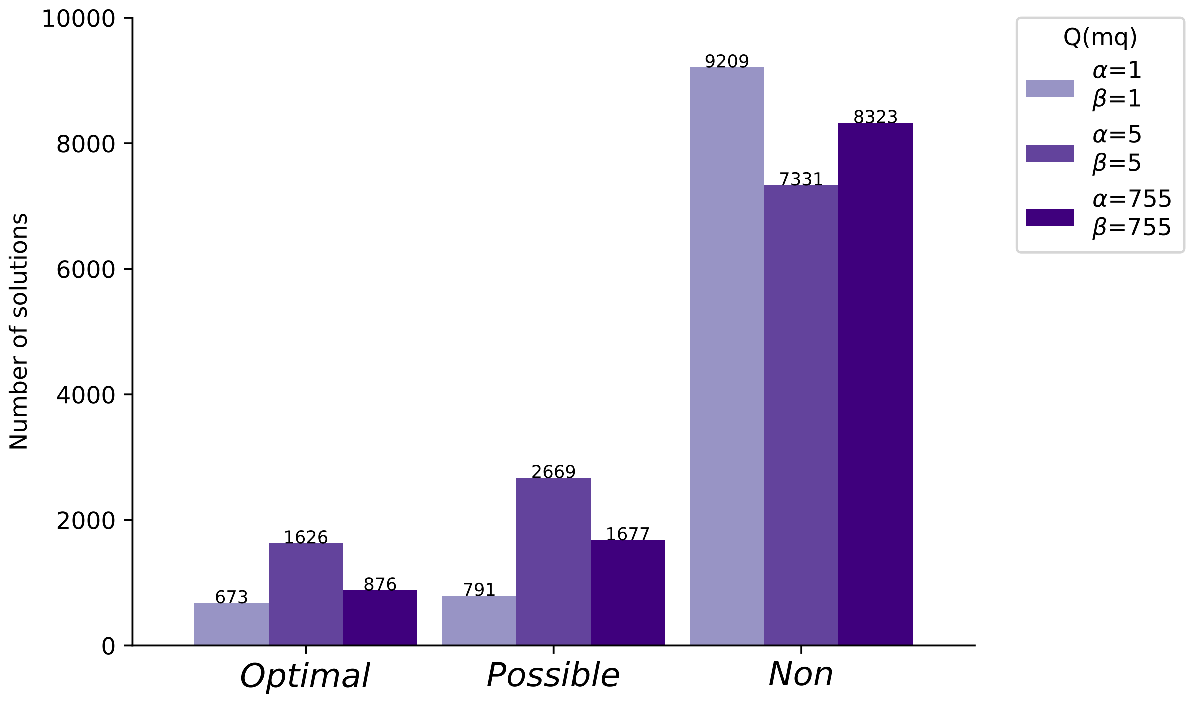


(a) `C`

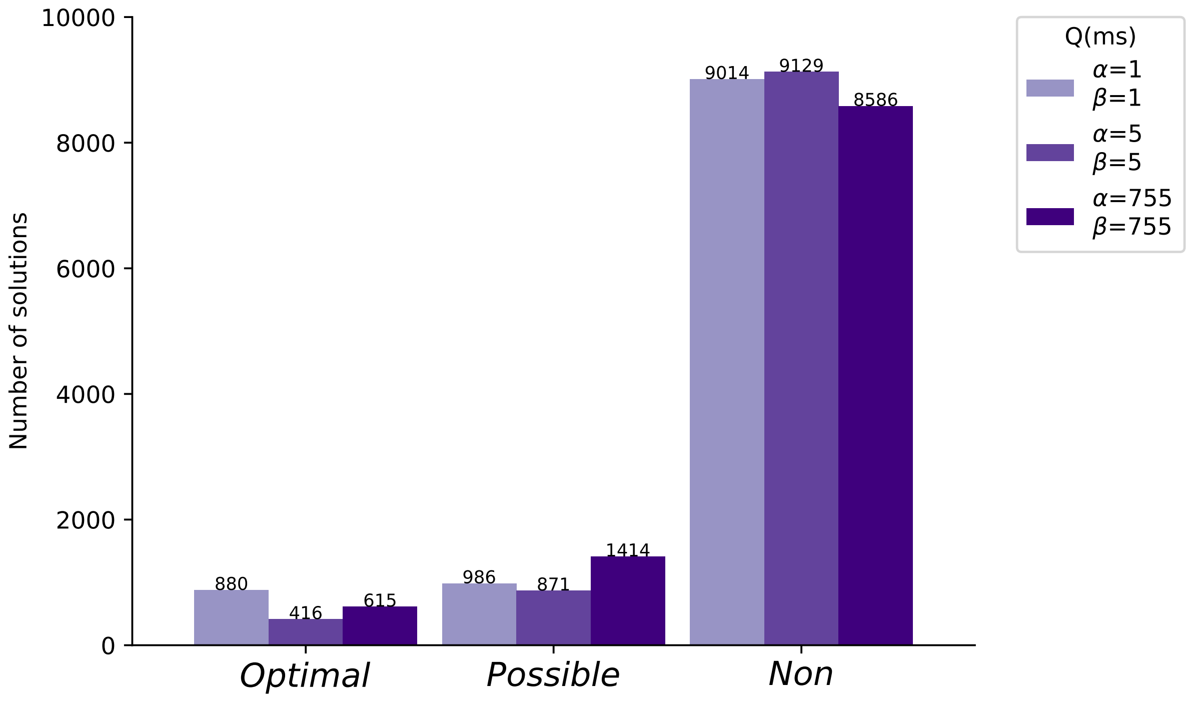


(b) `C_Q_sim`

Figure 7.5: Solutions counts for pure classical `C` and classical quantum simulator `C_Q_sim` for 3 sets ($\alpha = 1, \beta = 1$) and ($\alpha = 5, \beta = 5$) and ($\alpha = 755, \beta = 755$)

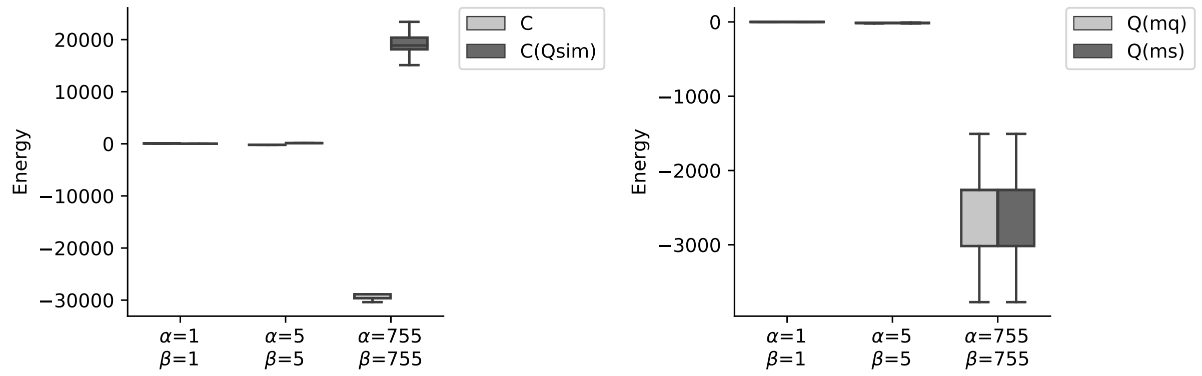


(a) `Q_mq`

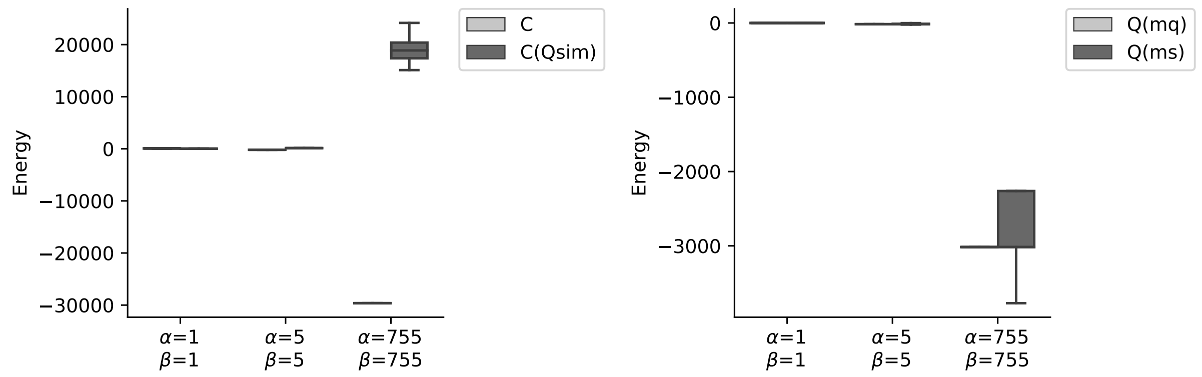


(b) `Q_ms`

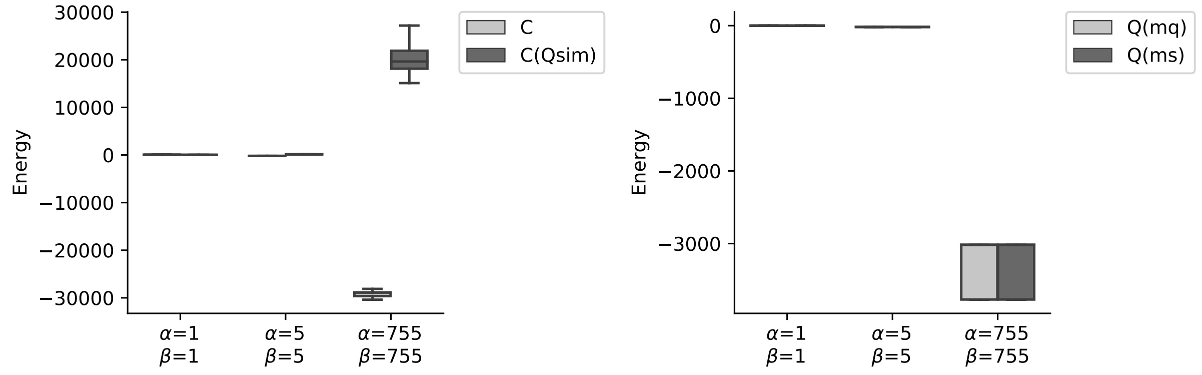
Figure 7.6: Solutions counts for quantum, both methods of polynomial reduction `Q_mq` and `Q_ms` for 3 sets ($\alpha = 1, \beta = 1$) and ($\alpha = 5, \beta = 5$) and ($\alpha = 755, \beta = 755$)



(a) Optimal solutions



(b) Possible solutions



(c) Non solutions

Figure 7.7: Energy values of classical `C` and both methods of polynomial reduction `Q_mq` and `Q_ms` for 3 sets ($\alpha = 1, \beta = 1$) and ($\alpha = 5, \beta = 5$) and ($\alpha = 755, \beta = 755$)

Table 7.2: Time (μs) of classical `C` and quantum `Q` experiments for graph coloring with 5 *colors* varying α

α	5	4	3	2	1
β	755	755	755	755	755
`C` (μs)	4002.4	3958.2	4688.5	4411.1	4049.3
`C_Q_sim` (μs)	1313.1	1054.9	1178.3	1022.2	1299.5
`Q_mq` (μs)	809.0	753.4	793.2	801.5	866.8
`Q_ms` (μs)	869.9	763.4	827.5	802.8	784.9

Table 7.3: Time (μs) of classical `C` and quantum `Q` experiments for graph coloring with 5 *colors* varying β

α	5	5	5	5	5
β	755	604	453	302	151
`C` (μs)	4226.3	4164.8	4190.6	4085.0	4373.9
`C_Q_sim` (μs)	1214.0	1102.7	1190.5	1258.5	1284.5
`Q_mq` (μs)	586.3	576.4	644.3	585.2	576.2
`Q_ms` (μs)	850.6	607.2	803.7	618.0	579.0

solutions (optimal and non optimal), in general.

In the boxplots that report the energy values for variations of α and β, we can observe that changes in β impact more the quality of solutions than changes in α. Besides, the classical SA pushes energy values towards more negative values, which is controversial, given that all quantum approaches find optimal solutions with energy close to zero, which was expected.

7.4.3 Problem mapped onto the QPU

To map a problem onto a QPU we resort to a process known as *minor embedding* which allows the mapping of logical *qubits* to physical *qubits* and often requires *chains* as explained before associated to a *strength*. The nodes and edges on the graph described in variables are translated to the *qubits* and *couplers* in the Chimera shown in Figure 7.1. Each logical *qubit* in the graph of the objective function can be represented by one or more physical *qubits*.

We illustrate the graph $G(5, 3)$ with 3 colors and 5 vertices with $\alpha = 5$ and $\beta = 755$, Figure 7.8, mapping variables to assigned *qubits* on both `Q_mq` and `Q_ms` methods. Also, the binary optimal results shown in a grid, Figure 7.9 for quantum results (`Q_mq` and `Q_ms`). For example, Figure 7.8 shows the embedded graph in the QPU hardware by assigning the vertices onto the physical *qubits*. In this Figure, we can notice that one variable can be mapped to several qubits during compilation. Also, one *qubit* can be used by multiple variables along the execution time. As an example, in the `Q_mq` experiment, variable 'vc30' was mapped to *qubits* [535, 527, 521, 393, 543, 536] while in the experiment `Q_ms`, variable 'vc30' was mapped to different *qubits*: [485, 493, 477] of the *Chimera*. In Figures 7.9, we show the selected colors for each experiment (i.e. 'c_k' equal 1, *black color*, when activated with k colors from 0 to 4), as well as, the selected edges of the resulting colored graph (i.e. 'vc_{ij}' equal 1, *black color*, varying ij according to colors k).

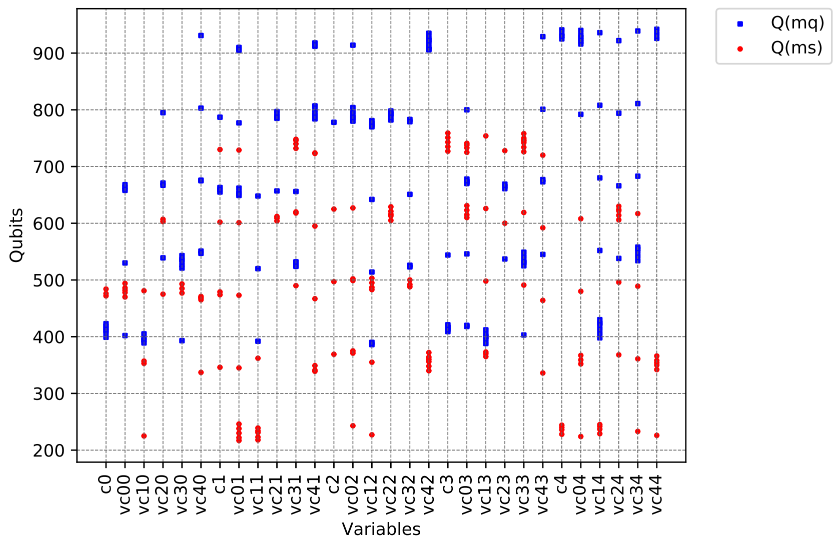
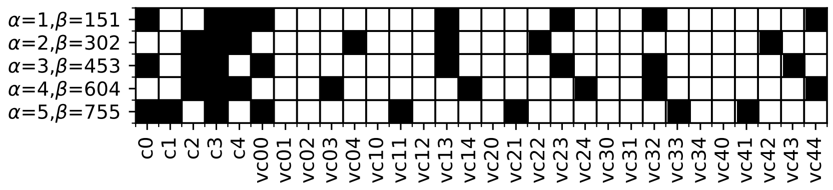
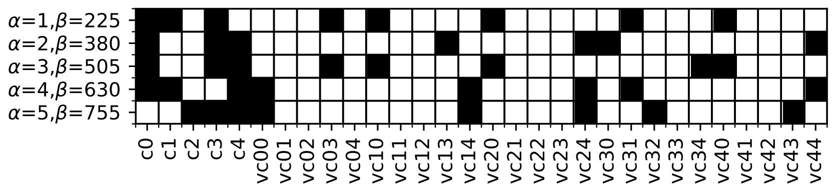


Figure 7.8: Mapping variables to assigned *qubits* on both `Q_mq` and `Q_ms` methods for $\alpha = 5$ and $\beta = 755$ for test in Figure 7.9



(a) `Q_mq`



(b) `Q_ms`

Figure 7.9: Binary optimal results: (α=1, β=151), (α=2, β=302), (α=3, β=453), (α=4, β=604), (α=5, β=755) on both `Q_mq` and `Q_ms` methods for test (varying α and β)

7.5 Conclusions and Future Work

Resorting to the adiabatic model we encoded the graph coloring problem into the quantum hardware graph (Chimera). We used a quantum annealer solver to sample low-energy states which are the optimal solutions of the problem.

Besides a classical implementation, both pure classical and simulating a quantum environment based on the chimera graph, we developed an algorithm that translates pseudo-boolean constraints to quantum annealing based in QUBO and symbolic computing. Since quantum annealers can not assure the finding of an optimal (lowest energy) solution and can be considered more than a formal solver, an automatic heuristic-finding [156], we study how the variation of penalties affects the number of solutions. We compare two implementations of quantum annealing using the D-Wave machine with classical implementations of QUBO.

In the D-Wave machine, besides the execution time reported, jobs needed to wait in a queue till being scheduled to run. We reported the number of optimal solutions found in each experiment, by performing a more controlled variation of α and β, fixing one of them and varying the other, giving much more insight about the performance of the quantum implementation regarding the quality of the explored landscape.

We conclude from our study that (a) experiments made varying α and β have small impact on the number of solutions found, (b) high β values cause the energy values to vary substantially and (c) the SA and QA approaches found a good number of solutions, leading to the remark that SA and QA produces good heuristics for the specific graph coloring problem, in particular the QA procedure. As future work, we intend to expand the experiments to larger graphs and other problems, such as, the travelling salesperson problem.

Chapter 8

Mapping a Logical Representation of TSP to Quantum Annealing

Carla Silva, Ana Aguiar, Priscila M. V. Lima, Inês Dutra
in the Quantum Information Processing, 2021

Abstract

This work presents the mapping of the traveling salesperson problem (TSP) based in pseudo-Boolean constraints to a graph of the D-Wave Systems Inc. We first formulate the problem as a set of constraints represented in propositional logic and then resort to the SATyrus approach to convert the set of constraints to an energy minimization problem. Next, we transform the formulation to a quadratic unconstrained binary optimization problem (QUBO) and solve the problem using different approaches: (a) classical QUBO using simulated annealing in a von Neumann machine, (b) QUBO in a simulated quantum environment, (c) QUBO using the D-Wave quantum machine. Moreover, we study the amount of time and execution time reduction we can achieve by exploring approximate solutions using the three approaches. Results show that for every graph size tested with the number of nodes less than or equal to 7, we can always obtain at least one optimal solution. In addition, the D-Wave machine can find optimal solutions more often than its classical counterpart for the same number of iterations and number of repetitions. Execution times, however, can be some orders of magnitude higher than the classical or simulated approaches for small graphs. For a higher number of nodes, the average execution time to find the first optimal solution in the quantum machine is 26% (n=6) and 47% (n=7) better than the classical.

8.1 Introduction

D-Wave Systems [94] has been developing its own version of a quantum computer that uses annealing, which is different from the gate-based model approaches of IBM, Intel, Rigetti and Google [148]. Currently, D-Wave has quantum machines with more than 5,000 qubits and 35,000 couplers, a larger, denser, and powerful graph for building quantum applications. Unlike universal quantum computers that operate using quantum gates and the principles of the circuit model [196], D-Wave devices are specialized primarily in solving combinatorial optimization problems typically using quantum annealing (QA) [98]. QA uses quantum fluctuations to find the global minimum of a given objective function over a given set of candidate solutions, in a way similar to simulated annealing (SA), which is based on thermal fluctuations [159]. In brief, QA exploits the tunnel effect to escape local minima rather than the thermal hill-climbing used by SA [158]. Various applications take advantage of annealing techniques, e.g. learning agents, biology, wireless technology, vehicle routing, chemistry [13, 29, 119, 197, 208], among others.

Explorations in quantum logic [42] and algorithms for mapping Boolean constraint satisfaction problems (CSPs) to QA have been proposed [88, 91, 191]. The aim of this work is to perform the mapping of the travelling salesperson problem (TSP) by using pseudo-Boolean constraints to a working graph of the D-Wave Systems. The procedure in this research is similar to the one presented by Silva et al. [191], applied to the graph coloring problem, resorting to the formulation of combinatorial problems by using the SATyrus approach (a SAT-based Neuro-Symbolic Architecture for Constraint Processing) [123]. While that work concentrated on evaluating the impact of penalties on obtaining optimal solutions, this focuses on finding approximate solutions. The final aim is to provide a general framework to solve classes of combinatorial problems such as graph coloring, travelling salesperson, minimum spanning tree, knapsack, among others [130]. In fact, some works have already highlighted the benefits of solving optimization problems using QA [52, 145].

Other approaches based on quantum annealing for TSP have been proposed. For example, Martoňák et al. [134] suggest a Monte Carlo QA scheme for the symmetric TSP, based on a constrained Ising-like representation. The performance is compared with the standard thermal SA, and the results highlight the importance of QA methods. In addition, Warren [211, 212] shows implementations of the symmetric TSP on a quantum annealer resorting to two approaches: (1) by finding approximate solutions, and (2) by finding an optimal tour. The author ends with a discussion of the hardware snags that hinder the best way for solving the TSP due to the connectivity between the artificial spins which are sparse and limited on the D-Wave. Furthermore, to overcome hardware restrictions, studies on reducing limitations of quantum annealer in solving optimization problems under constraints are being made [150].

The remainder of this paper is organized as follows: in the first section we present the background where we define (i) the mapping of the travelling salesperson problem to constraint satisfiability, (ii) concepts on adiabatic quantum computation and the mapping formulation. The following section contains the implementations where we describe the travelling salesperson problem mapping from classical and annealer perspective through the methodology. Afterwards, we show the experimental results with discussion, ending with conclusion and future work.

8.2 Background

In this section, a method is shown for modeling optimization problems using logical constraints which is exemplified for the TSP. We explain the methods used that uncover solutions by energy minimization in a classical and quantum perspective. We describe the foundations on which the adiabatic model application is based later on. The concepts of pseudo-Boolean problems and the integration between them and the problem of satisfiability are addressed.

8.2.1 Mapping combinatorial problems to constraint satisfiability

In our context a combinatorial problem is represented as logical Boolean constraints which are then mapped to an energy expression where the goal is to minimize the energy. As such, we show the mapping process for the TSP according to this formulation [123]. The traveling salesperson problem can be defined as follows. Let $G = (V, E)$ be an undirected graph, where V is the graph's vertex set (cities) and E the set of edges (connecting two cities). The goal is to find a tour of minimum cost (distance). To model the problem, two types of constraints are modeled: Integrity and Optimality Constraints. The Integrity Constraints are: (1) all n cities must take part in the tour, (2) two cities cannot occupy the same position in the tour, (3) a city cannot occupy more than one position in the tour. Also, the Optimality Constraint is defined by (4) the cost between two adjacent cities in the tour. Resorting to Lima *et al.*'s formulation, the mapping of all constraints is represented by the energy function as shown in Equation 8.1.

As an example, the first part of the equation represents the Integrity Constraint (1), which is initially represented by the logical disjunction: $\forall i, \forall j \mid 1 \leq i \leq n, 1 \leq j \leq n : \vee_j(v_{ij})$, where v_{ij} is a Boolean variable representing a city i appearing in the j_th position in the tour.

$$\begin{gathered} Energy = \alpha\Big[\sum_{i=1}^{n}\sum_{j=1}^{n}(1 - v_{ij})\Big] + \beta\Big[\sum_{i=1}^{n}\sum_{i'=1, i'\neq i}^{n}\sum_{j=1}^{n}(v_{ij}v_{i'j})\Big] + \\ \beta\Big[\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{j'=1, j'\neq j}^{n}(v_{ij}v_{ij'})\Big] + \Big[\sum_{i=1}^{n}\sum_{i'=1, i'\neq i}^{n}\sum_{j=1}^{n-1} dist_{ii'}v_{ij}v_{i'(j+1)}\Big] \end{gathered} \tag{8.1}$$

The *dist* corresponds to the distance between cities and v is a Boolean variable. We then model the objective function through penalty terms. As such, the constraints are associated to a penalty (Equation 8.2) where α is applied to the constraints of type 1 and β to the constraints of types 2 and 3, with h being a small number and n the number of cities (the number of vertices in the graph).

$$\begin{cases} dist = max\{dist_{ij}\} \\ \alpha = ((n^3 - 2n^2 + n) * dist) + h \\ \beta = ((n^2 + 1) * \alpha) + h \end{cases} \tag{8.2}$$

As an example, we implemented [187] the TSP based in Hopfield [85] and Tank [87], with the following energy function 8.3:

$$\begin{aligned} Energy = \frac{A}{2}\Big[\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1,k\neq j}^{n} v_{ij}v_{ik}\Big] + \frac{B}{2}\Big[\sum_{j=1}^{n}\sum_{i=1}^{n}\sum_{k=1,k\neq i}^{n} v_{ij}v_{kj}\Big] + \\ \frac{C}{2}\Big[\sum_{i=1}^{n}\sum_{j=1}^{n}(v_{ij} - n)^2\Big] + \frac{D}{2}\Big[\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\sum_{k=1}^{n} dist_{ij}v_{ik}(v_{jk+1} + v_{jk-1})\Big] \end{aligned} \tag{8.3}$$

Where *dist* corresponds to the distance between cities, v represents a Boolean variable, and n the number of cities. We considered $A = B - (D \times dL)$, $B = (3 \times dU) + C$, $D = 1/dU$ and $C = 1$, and $dL = dU/2$ with dU equals to the sum of edge weights in the graph. With this formulation using the QUBO approach we achieve optimal solutions for the TSP for $n \leq 6$. For the study of parameters A, B, C and D we analyze the references [131, 160, 200].

8.2.2 Adiabatic quantum computation and the mapping

In the context of quantum mechanics, a Hamiltonian is a mathematical concept which describes the physical system by mapping certain states to energies. As such, in the D-Wave system the Hamiltonian can be defined as:

$$H_{Ising} = \underbrace{\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i\hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right)}_{\text{Final Hamiltonian}} \tag{8.4}$$

where $\hat{\sigma}_{x,z}^{(i)}$ are Pauli matrices (a set of three 2 x 2 complex matrices) operating on a *qubit* q_i, and h_i and $J_{i,j}$ represent the *qubit* biases and coupling strengths. $A(s)$ and $B(s)$ are energy scaling functions that progress according to the anneal fraction, s.

Quantum annealing (also known as the quantum adiabatic algorithm or adiabatic quantum optimization) starts from the ground state of the initial Hamiltonian applying successive transformations till the system attains the final ground state which solves the optimization problem. The *adiabatic theorem* of quantum mechanics ensures that QA finds the final ground state (solution of the optimization). According to Mishra et al. [139], this process does not generally allows the system to perform better than classical optimization algorithms.

More formally, the system begins in a ground state of a time dependent Hamiltonian $H(s)$ (with $s \in [0, 1]$), that is slowly modified from an initial form H_0 to a final form H_1, in a time $t(s)$ with $t(0) = 0$, $t(1) = \tau$, considering an interpolation between the two Hamiltonian states. Starting with $|\psi(0)\rangle$ in the ground state of H_0 the system evolves according to Schrödinger's equation and end up near the ground state of H_1 [57, 166]. The Adiabatic Quantum Computation (AQC) Hamiltonian can be written as follows,

$$H(s) = (1 - s)\, H_0 + sH_1 \tag{8.5}$$

where the Hamiltonian H_1 ground state encodes the solution of a combinatorial optimization problem. In brief, AQC encodes the solution to a problem in the ground state of the final Hamiltonian where the adiabatic theorem can have approximate or rigorous versions as described in [7].

Nowadays, there are many types of architectures for quantum computers. In the gate-based model, the goal is to control and handle the evolution of the quantum states in time, in a circuit perspective. In quantum annealing, the goal is to initialize the system in a delocalized state, i.e. anywhere in space, where it would gradually converge to our solution by minimizing the energy. Quantum annealing (as a metaheuristics) aids in finding the global minimum of a given objective function over a given set of candidate states. For example, one possible mapping is through the Ising model, which can solve Boolean problems with the objective function below, which shows the relationship between the spins, represented by couplings. In this case, the *linear coefficients*, corresponding to *qubit* biases, the *quadratic coefficients*, corresponding to coupling strengths, the variables are either "spin up" (↑) or "spin down" (↓) states which correspond to +1 and -1 values. A problem can also be formulated as a QUBO where variables can be *true* and $false$ with states, respectively, corresponding to 1 and 0 values. The Ising and QUBO models are mathematically equivalent, it is possible to translate from one model to the other. The QUBO-based equation is defined as follows:

$$\text{Energy}_{QUBO}(\boldsymbol{x}) = \sum_{i}^{N} a_i x_i + \sum_{i<j}^{N} b_{i,j} x_i x_j. \tag{8.6}$$

where a_i are the *linear coefficients*, $b_{i,j}$ the *quadratic coefficients*, and x_i and x_j are the variables. To translate a QUBO model to an Ising model, $x_i \to \frac{s_i+1}{2}$, on the other hand, from Ising to QUBO, $s_i \to 2x_i - 1$.

Currently, D-Wave Systems allows access to quantum processing units (QPUs) with more than 5000 quantum bits which can be accessed using the real-time quantum application environment, Leap [96]. Figure 8.1 shows one of the D-Wave QPU architectures, the *Pegasus* topology. The D-Wave's QPU's are fabricated with X qubits and Y couplers in a *named* topology and the D-Wave QPU is a not fully connected lattice of interconnected *qubits*. *Pegasus* is the D-Wave Advantage topology following the D-Wave 2000Q QPU fabricated with 2048 *qubits* and 6016 couplers in a *Chimera* topology.

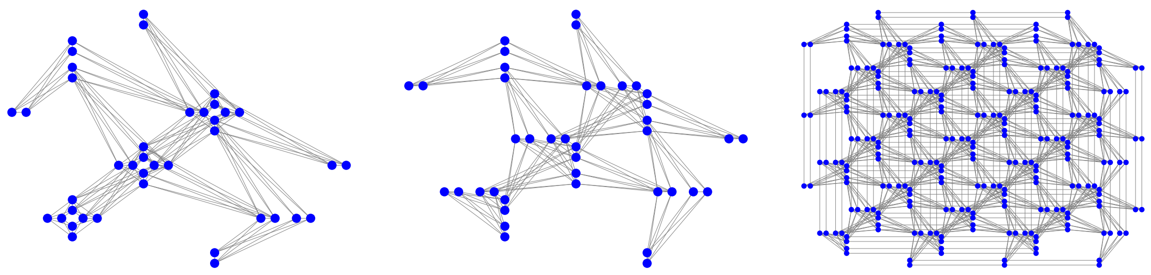

(a) $K_{4,4}$ configuration (b) L configuration (c) Unit cells in a P_4 graph

Figure 8.1: *Pegasus* topology with *qubits* represented as dots and couplers as lines

We use the notation C_n for the *Chimera* graph with size n and P_n for the *Pegasus* topology family. Furthermore, the D-Wave's topologies are characterized by a nominal length and degree. The nominal length concerns to how many *qubits*, through internal couplers, each *qubit* is orthogonally connected, and the degree is the number of different *qubits* each *qubit* is connected through couplers. As such, in the *Chimera* topology, *qubits* have a nominal length of 4 and degree of 6, and in the *Pegasus* topology, *qubits* have a nominal length of 12 and degree of 15 [95].

As mentioned in [191], the D-Wave quantum annealing processor has the goal to minimize the energy of an Ising/QUBO configuration whose pairwise interactions lie on the edges of a *Chimera* or *Pegasus* graph. However, the Ising/QUBO problem should first be embedded [16, 45] onto the working graph. Chain breaks (i.e., the measurement of all *qubits* in the chain giving different values) may occur during embedding an optimization problem into the annealing hardware. To tackle this issue, we can provide a quantity based in upper values of the maximum value of QUBO coefficients. A chain of (multiple) physical *qubits* in the QPU represents each logical variable (a binary variable in the graph), and the chain break fixing is needed when the physical *qubits* in a chain do not have all the same value, 0 or 1, when the annealing cycle ends, so they can be mapped back to the original problem, in this case, the chain strength plays a relevant role [163].

Problem mapped onto the QPU. When mapping a problem to the QPU (e.g. *Pegasus* shown in Figure 8.1), the nodes and edges on the graph that represents the problem are translated to the *qubits* and *couplers* of the working graph. In this scenario, each logical *qubit*, in the graph of the objective function, can be represented by one or more physical *qubits*, and this process is known as *minor embedding*. In this embedding technique, each logical variable (binary variable in the original graph) is represented by a chain of physical *qubits* in the QPU where all the physical *qubits* in a given chain should have the same value at the end of the annealing cycle. Besides, we should consider the chain strength c which is the weight that is associated to the edges. If this value is not large enough, the physical *qubits* in the chain will not take the same value as we mentioned before, and the chain may break and make it hard to find an optimal solution. This constant c is typically the maximum value of the QUBO coefficients or an upper close value to this maximum.

8.3 Methodology

In this section, we discuss the methods used to conduct the research, in particular, we depict the mapping process, show the algorithms that allow the mapping of the formulated problem, and the experimental setup.

8.3.1 The mapping process

Figure 8.2 shows the main scheme steps for the mapping process. We begin with the Boolean constraint formulation, then we develop the algorithms described in our methodology (code is available at [187]).

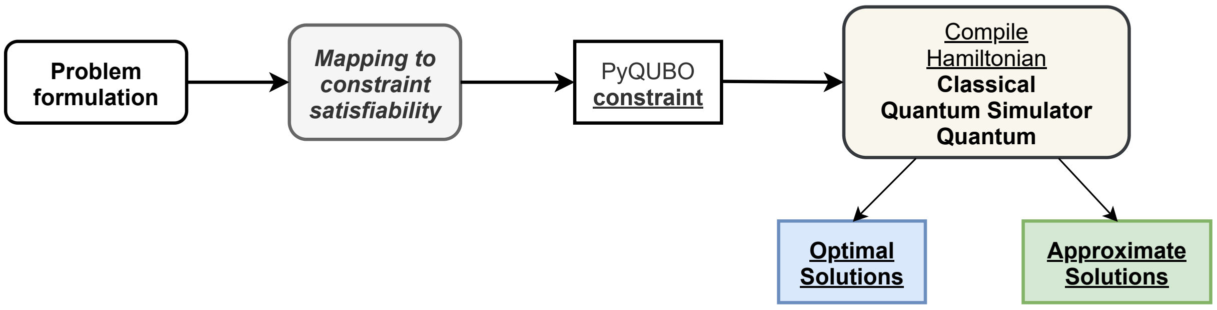


Figure 8.2: Main scheme steps for the mapping process

Algorithm 12 takes the number of cities and h (small constant) and returns the multiplicatives α and β. Algorithm 13 compiles the model based on the Hamiltonian created from the expressions which represent the constraints (PyQUBO [221] is used for that).

As the proposed work is heuristic-based, there is a chance that the returned solution is not optimal. Usually, the QPU is asked to return a sample of R results, to raise the chances of finding an optimal solution [135]. These R results contain combinations of variables values that can be solutions and no solutions. In order to count solutions, optimal and not optimal, we take the output list of the annealing and check for the conditions below to count the number of (a) optimal solutions (satisfy all the listed conditions) and (b) possible solutions (satisfy all the listed conditions but not with the minimum distance tour). The attained results were compared with the achieved results by resorting to the Python library `tsp`.

- ***Conditions to ensure optimal solutions***
  - no broken constraints: all $c \in R$ such that $c = 0$
  - $v_{ij} \in \{0, 1\} = 1$ for the n cities
  - different cities for each vertex of an edge

- same number of edges and vertices: $n_{edges} = n_{vertices}$
- graph is connected
- path starts and ends in the same city
- path has n nodes
- a 2-edge connection for each node

Algorithm 12: TSP penalties multiplicatives.

```
1 function alpha ($n, h, maxdist$);
  Parameters : Number of cities, small constant
  Alpha      : Multiplicative
2 return $(((n^3) - (2 * n^2) + n) * maxdist) + h$
3 function beta ($n, alpha, h$);
  Parameters : Number of cities, alpha, small constant
  Beta       : Multiplicative
4 return $(((n^2) + 1) * alpha) + h$
5 $A \leftarrow alpha(n, h, maxdist)$;
6 $B \leftarrow beta(n, A, h)$;
7 return $A, B$
```

8.3.2 Experimental setup

We performed three types of experiments:

- classical (`C`): for this experiment, we used a classical simulated annealing algorithm and the `dwave-neal` and `PyQUBO` libraries. We vary number of reads (10,000; default=10), number of sweeps (according to each graph size; default=1,000).
- quantum simulation (`C_Q_sim`): we use the same parameters as before but in these experiments we added the `dwave-networkx` library which allows to simulate the *Pegasus* architecture, along with the `dimod` and `dwave-system` which provide the `Structure-Composite` in order to build the sampler.
- quantum hardware (`Q`): we resort to the *Pegasus* topology from D-Wave's, and we change the chain strength (set as explained previously, based in the maximum QUBO coefficient value), the default is "None". The number of reads is equal to 10,000 (the default is 1,000) and the *answer mode* equal to "raw" so we can get the results sorted by output order and not ordered from the lowest energy (and maximum number of occurrences) which is by default "histogram".

We setup the following experiments for the three solvers (classical `C`, quantum simulator `C_Q_sim` and quantum hardware `Q`) for the TSP varying the number of cities (n). The experiments were conducted according to the achieved outputs:

Algorithm 13: TSP Hamiltonian (PyQUBO)

```
function exp1 $(n, v, A)$;
Parameters : Number of cities, binary vector, $\alpha$
Constraints : Integrity
for $i$ to $n$ do
    for $j$ to $n$ do
        $exp \leftarrow exp + Constraint(1 - v(i, j))$;
    end
end
$exp \leftarrow Constraint(A * exp)$;
return $exp$
function exp2 $(n, v, B)$;
Parameters : Number of cities, binary vector, $\beta$
Constraints : Integrity
for $i$ to $n$ do
    for $i'$ to $n$ do
        for $j$ to $n$ do
            if $i' \neq i$ then
                $exp \leftarrow exp + Constraint(v(i, j) * vc(i', j))$;
            end
        end
    end
end
$exp \leftarrow Constraint(B * exp)$;
return $exp$
function exp3 $(n, v, B)$;
Parameters : Number of cities, binary vector, $\beta$
Constraints : Integrity
for $i$ to $n$ do
    for $j$ to $n$ do
        for $j'$ to $n$ do
            if $j' \neq j$ then
                $exp \leftarrow exp + Constraint(v(i, j) * v(i, j'))$;
            end
        end
    end
end
$exp \leftarrow Constraint(B * exp)$;
return $exp$
function exp4 $(n, v, dist)$;
Parameters : Number of cities, binary vector, tour's cost
Constraints : Optimality
for $i$ to $n$ do
    for $i'$ to $n$ do
        for $j$ to $n - 1$ do
            if $i' \neq i$ then
                $exp \leftarrow exp + Constraint(dist(i, i') * v(i, j) * v(i', j + 1))$;
            end
        end
    end
end
return $exp$
H = exp1 $(n, v, A)$ + exp2 $(n, v, B)$ + exp3 $(n, v, B)$ + exp4 $(n, v, dist)$
```

a. All solutions

- Execution time (μs)
- Possible and optimal solutions counts (#)
- Possible solutions energy and standardized tour length.

b. Optimal and approximate solutions

- Execution time (μs)
- Tour length (l)

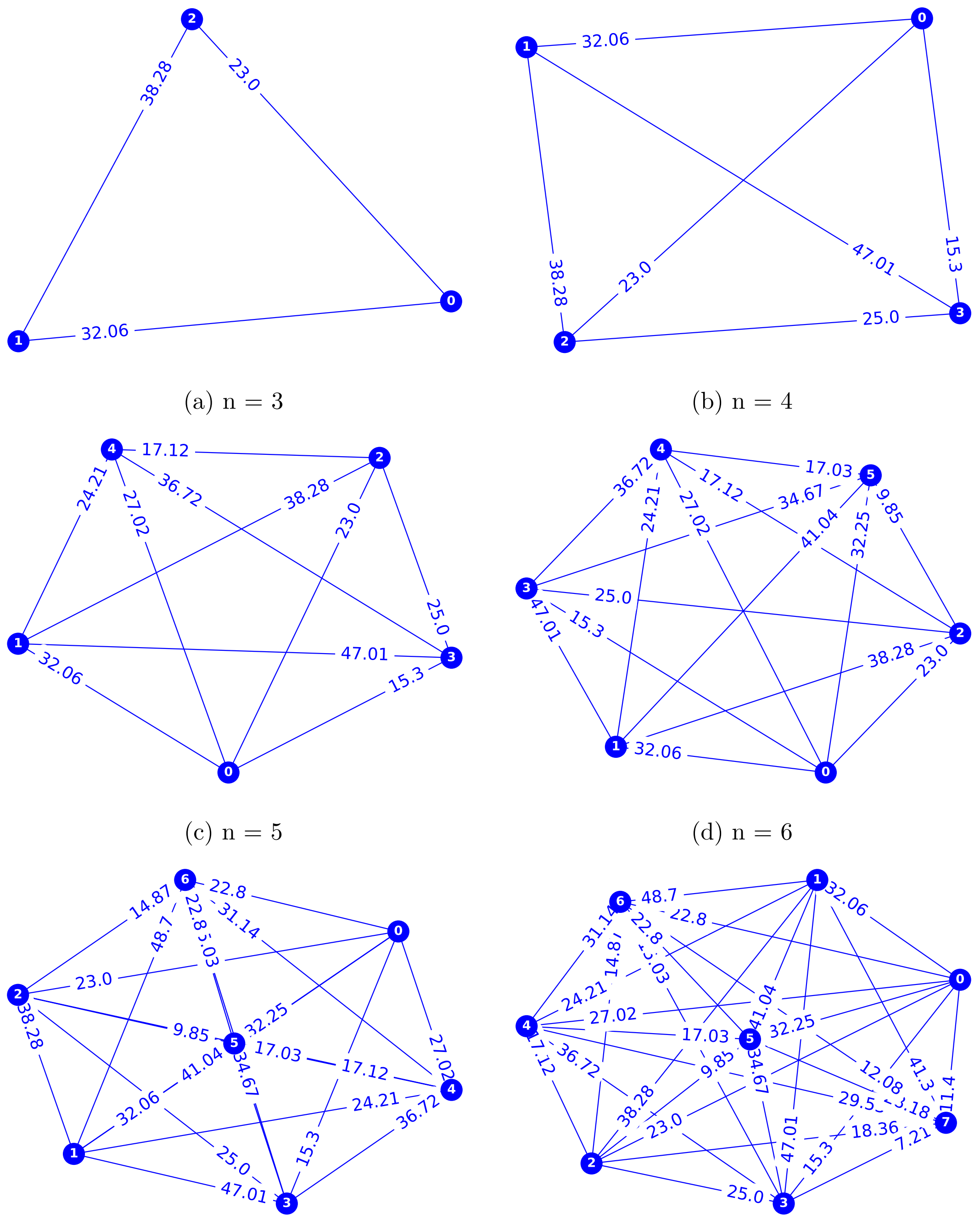
The goal with standardization is to transform all values to the same scale, making the data agnostic to the hardware type (classical or quantum) and annealing (simulated or quantum), for comparison purposes.

Graphs. We generate random integers coordinates based on numbers e.g. between 0 and 50, and we build the graphs of size n, $3 \leq n \leq 8$. Figure 8.3 shows all graphs with their calculated pairwise Euclidean distances between cities, according to their coordinates. Figure 8.4 shows the minimum distance tours for each graph size.

Solutions. The optimal solutions correspond to the minimum tour length for a graph G of size n, and the approximate solutions correspond to the closer first solution near the optimal. For example, as shown in Table 8.1, let's suppose **s__a** is the optimal solution on time **t__4** with tour length **l__10**. According to this table, the closest approximate solution **s__b** with tour length **l__20** is obtained at time **t__1**. If we are interested in a good (not necessarily optimal) solution, we would save 3 time units in the search.

Table 8.1: Output example to represent optimal and approximate solutions

Solution (s)	Energy (e)	Tour length (l)	Time (t)
s__b	e__b	l__20	t__1
s__b	e__b	l__20	t__2
s__b	e__b	l__20	t__3
s__a	**e__a**	**l__10**	**t__4**
s__c	e__c	l__50	t__5
s__c	e__c	l__50	t__8
s__a	**e__a**	**l__10**	**t__7**
...	...	...	...
s__p	e__p	l__30	t__10,000



(a) n = 3 (b) n = 4

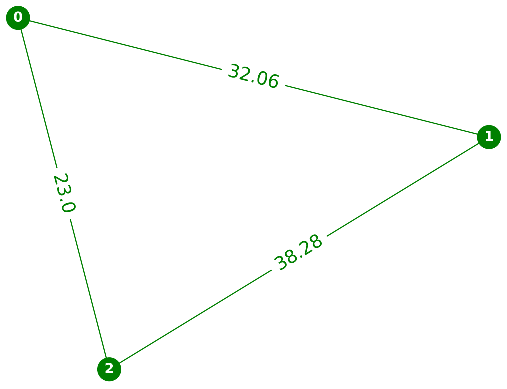
(c) n = 5 (d) n = 6

(e) n = 7 (f) n = 8

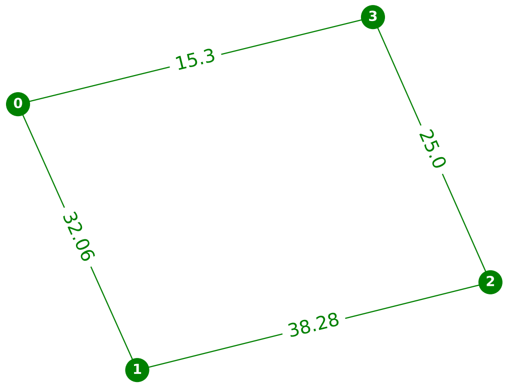
Figure 8.3: All distances between cities varying the number of cities

Hardware. The quantum experiments were conducted on D-Wave's Advantage systems with topology *Pegasus*, and the classical in a MacBook Air, 1.8 GHz Intel Core i5 processor, 8 GBytes of memory 1600 MHz DDR3.

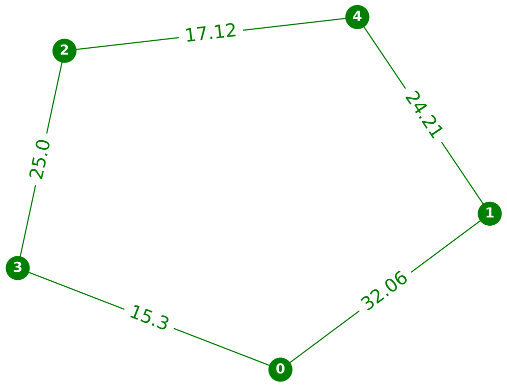
Times. We ran experiments using 10,000 iterations. For both classical experiments (`C` and `C_Q_sim`), we calculated average execution time of the 10,000 iterations. In the case of the quantum



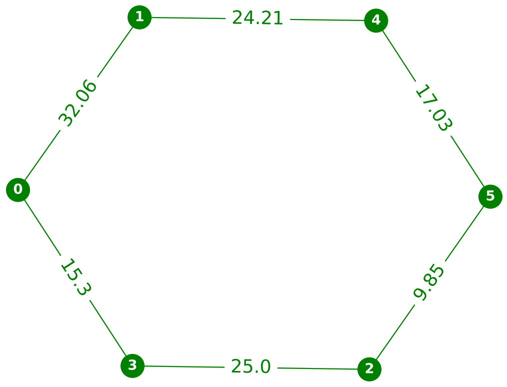
(a) n = 3, distance = 93.34



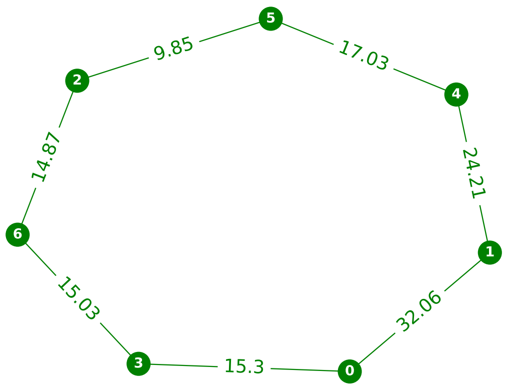
(b) n = 4, distance = 110.64



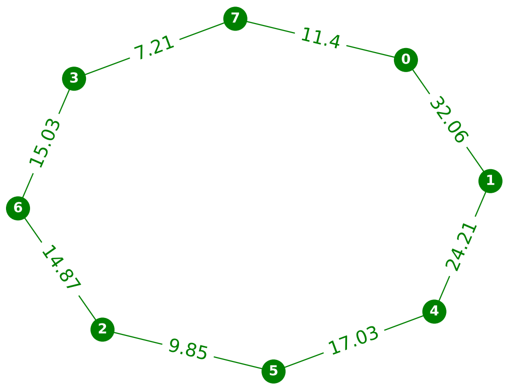
(c) n = 5, distance = 113.69



(d) n = 6, distance = 123.45



(e) n = 7, distance = 128.35



(f) n = 8, distance = 131.66

Figure 8.4: Minimum tour length varying the number of cities

annealer experiments, execution times are broken down in various categories: network latency and QPU access time are sources of overheads reported by the `qpu_access_overhead_time` variable. This overhead concerns low-level operations at initialization time of 10-20 ms for Advantage systems, as is the case with our experiences. In this work, we focus on the QPU access time which is divided into two parts: a one-time initialization step to program the QPU (programming time) and sampling times for execution on the QPU (sampling time). In order to be fair and measure only the time taken to obtain one solution in the quantum machine we use the sampling time, which is broken down into anneal (anneal time), the read of the sample from the QPU (readout time), and thermalization (time for the QPU to hit again its initial temperature, delay time). The time taken to obtain each solution is then given by the sum of the QPU times for each sample: `qpu_anneal_time`, `qpu_readout_time` and `qpu_delay_time`[1]. A simplified view of the execution time breakdown for quantum experiments is shown in Figure 8.5.

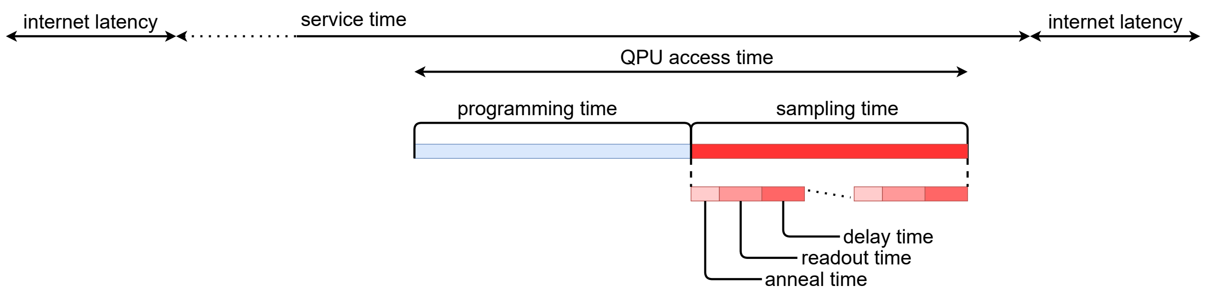


Figure 8.5: A simplified diagram of the sequence of steps to execute a quantum machine instruction (QMI) on a D-Wave system adpated from D-Wave's documentation

We take into account that the QPU call to sample a QUBO is *non blocking*, which means that as the operation is being performed (sent through the cloud, sampled on the QPU, and returned), other non-related operations can be performed in parallel.

SA sweeps. We begin with both classical solvers by exploring the SA algorithm where the minimum number of iterations (sweeps, sw) needed to obtain an optimal solution to each graph G varies. After some tests, we found minimum $sw = 2$ for graphs of sizes $3, 4, 5, 6$ and $sw = 9$ for size 7. Therefore, we use these values as input to the SA algorithms (classical `C` and classical quantum simulator `C_Q_sim`) to achieve the best performance, since reducing the number of sweeps reduces the execution time of the algorithms.

Metrics. The metrics used are execution time, tour length and energy. In particular, the energy is very relevant, since quantum annealing processors return low-energy solutions, and applications such as the TSP, require the minimum energy given that we model it as an optimization problem.

Objective. Our goal is to answer the following questions for each n (number of cities) and type of experiment (classical or quantum): (a) how do the execution times vary, (b) what is the number of possible and optimal solutions, (c) how the energy and the distance vary for each type of solution, and (d) how far is each approximate solution from the corresponding optimal solution in value and in time.

[1]More on QPU times can be found at:
https://docs.dwavesys.com/docs/latest/timing_qa_cycle_time.html
https://docs.dwavesys.com/docs/latest/timing_overview.html

8.4 Results and Discussion

In this section, we show the results of the execution times of each set of experiments. Afterwards, we indicate the optimal solutions and possible solutions and expected energy standardized values of classical `C`, quantum simulator `C_Q_sim` and quantum hardware `Q` in optimal and approximate solutions perspective. Finally, we describe the problem mapped onto the QPU with broken and non-broken solutions.

8.4.1 Time, solution, energy and distance

Table 8.2 shows the average execution times, in microseconds (μs), to produce each output (possible and non possible solutions, e.g. n = 4 cities with results for just 3 cities). The average execution time for all graph sizes is 4.9 μs for the classical runs (`C`), 28.1 μs for the classical quantum simulator (`C_Q_sim`), and 95.4 μs for the quantum annealing using the D-Wave's hardware (`Q`).

Table 8.2: Time (μs) for 10,000 iterations of classical `C`, quantum simulator `C_Q_sim` and quantum hardware `Q` experiments for TSP varying the number of cities (n)

n	3	4	5	6	7
`C` (μs)	1.3	2.1	3.4	5.3	12.4
`C_Q_sim` (μs)	24.4	23.6	24.2	27.4	41.1
`Q` (μs)	90.0	102.0	82.0	103.0	100.0

Next, we discuss the results of possible and optimal solutions. As we can notice in Table 8.3, the experiments that ran in the D-Wave, in general, produced a smaller number of possible solutions. The average counts are 872 for the classical runs (`C`), 872 for the classical quantum simulator (`C_Q_sim`), and 435 for the quantum hardware (`Q`). However, for n equals 5 and 6 the quantum version obtained more possible solutions. Table 8.4 shows that the classical experiments, in general, obtained more optimal solutions. The average counts are 431 for the classical runs (`C`), 431 for the classical quantum simulator (`C_Q_sim`), and 207 for the quantum version (`Q`). The results indicate that the classical procedures performed a better heuristic search for optimal solutions for n equals 3, 4 and 7, while the quantum annealing performed better for n equals 5 and 6. It is important to notice that for each type of experiment, at least one optimal solution was found.

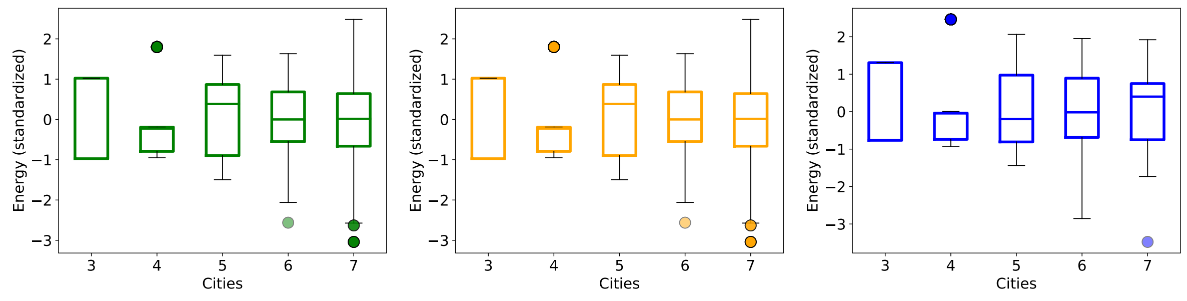
Figure 8.6 shows the energy values of classical `C`, quantum simulator `C_Q_sim` and quantum hardware `Q` experiments for TSP varying the number of cities. The values are standardized with mean 0 and standard deviation 1. As we can observe, `C`, `C_Q_sim` and `Q` report similar behavior in energy values and variance within each group for each n, with `Q` presenting a smaller number of outliers.

Table 8.3: Possible solutions: counts (#) of classical `C`, quantum simulator `C_Q_sim` and quantum hardware `Q` experiments for TSP varying the number of cities (n)

n	3	4	5	6	7
`C` (#)	1842	858	281	60	1319
`C_Q_sim` (#)	1842	858	281	60	1319
`Q` (#)	781	711	393	227	62

Table 8.4: Optimal solutions: counts (#) of classical `C`, quantum simulator `C_Q_sim` and quantum hardware `Q` experiments for TSP varying the number of cities (n)

n	3	4	5	6	7
`C` (#)	1842	280	28	2	3
`C_Q_sim` (#)	1842	280	28	2	3
`Q` (#)	781	197	51	5	2



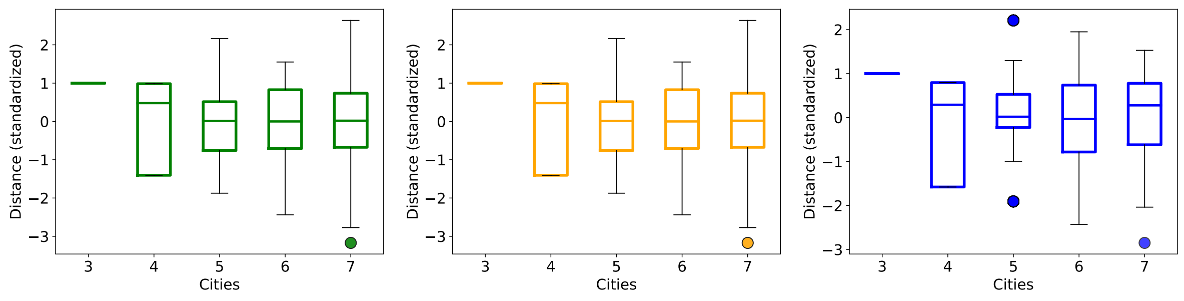
(a) `C` (b) `C_Q_sim` (c) `Q`

Figure 8.6: Possible solutions energy (standardized values with mean 0 and standard deviation 1)

Similarly, Figure 8.7 shows the tour length values (possible values, not optimal i.e. does not imply minimum value) of classical `C`, quantum simulator `C_Q_sim` and quantum hardware `Q` experiments for TSP varying the number of cities. As before, the values are standardized with mean 0 and standard deviation 1. All the scenarios report similar behavior regarding tour length values, similar to the previous comparison.

8.4.2 Approximate solutions

Table 8.5 shows the values of the first optimal tour length solution for each n and the closest approximate solution in time. The time values displayed on Table 8.5 correspond to the time needed to achieve the first found solution, for both cases, optimal or approximate, since the beginning of the execution. As we can notice in Table 8.5, there is a quantum advantage in time for $n = 6$ and 7 to achieve the optimal solution, highlighting the quantum advantage that can be possible for larger graphs. However, the same is not true if we are seeking for an approximate



(a) `C` (b) `C_Q_sim` (c) `Q`

Figure 8.7: Possible solutions tour length (standardized values with mean 0 and standard deviation 1)

Table 8.5: Tour length (l) and time (μs) of classical `C`, quantum simulator `C_Q_sim` and quantum hardware `Q` experiments for TSP varying the number of cities (n) for the first optimal and closest approximate solutions. Results for n=3 are shown only to validate the implementation

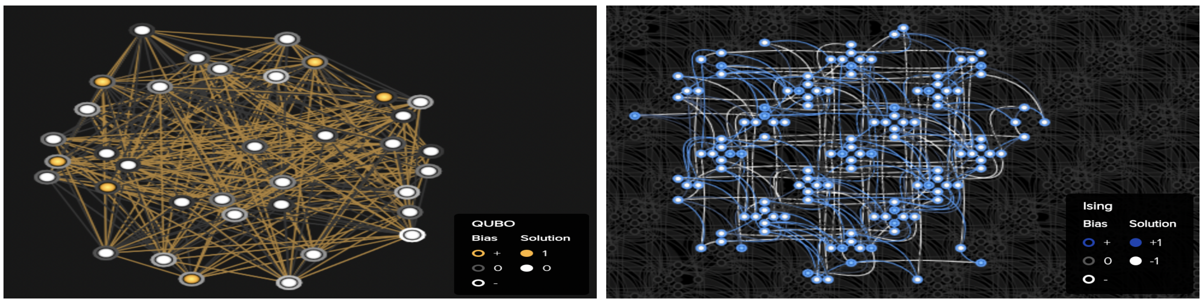
		C		C_Q_sim		Q	
n	Solutions	(l)	(μs)	(l)	(μs)	(l)	(μs)
3	Optimal	93.3	11.9	93.3	219.7	93.3	630.0
	Approximate	93.3	11.9	93.3	219.7	93.3	630.0
4	Optimal	110.6	41.0	110.6	471.5	110.6	38862.0
	Approximate	123.6	34.8	123.6	400.8	123.6	63648.0
5	Optimal	113.7	2667.5	113.7	19175.6	113.7	18696.0
	Approximate	126.6	3878.1	126.6	27878.5	126.6	103484.0
6	Optimal	123.5	23199.5	123.5	119361.0	123.5	15759.0
	Approximate	136.4	14943.7	136.4	76884.9	133.2	20394.0
7	Optimal	128.4	9075.2	128.4	30097.5	128.4	7200.0
	Approximate	136.4	1498.1	136.4	4968.3	147.3	19300.0

solution. In this case, the classical version has better performance. As such, the classical process can be recommended for cases in which it is not possible or it is not necessary to achieve optimal solutions. In fact, for the graph of size 8 the quantum annealing could not find an optimal solution.

From the results, it looks like the quantum annealing is more efficient during the search, jumping to better solutions earlier.

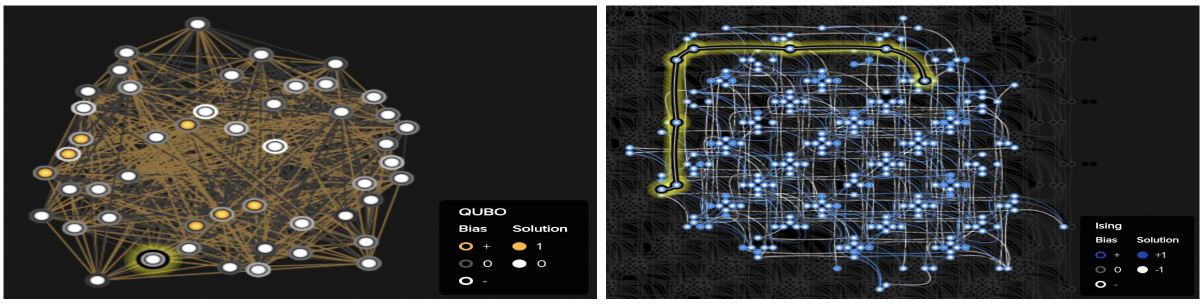
8.4.3 Problem mapping

Figures 8.8, 8.9 and 8.10 show the state of the qubits after obtaining solutions (on the left, the QUBO mapping and on the right the Ising mapping). The embedded problem gives us a perspective of the mapping into the graph of the D-Wave's by using the `dwave-inspector`



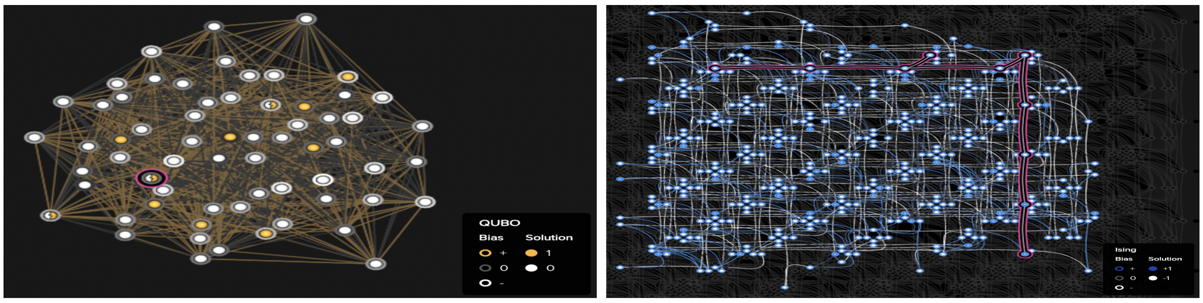
(a) QUBO (b) Ising

Figure 8.8: Mapping TSP n = 6 cities onto the QPU



(a) QUBO (b) Ising

Figure 8.9: Mapping TSP n = 7 cities onto the QPU. Solution (with warnings)



(a) QUBO (b) Ising

Figure 8.10: Mapping TSP n = 8 cities onto the QPU. Broken solution, chain length equal 8 (greater than 7)

tool which provides a graphical interface for examining problems and solutions. Each image shows the mapping of a different n. Boolean variables that are solutions, are colored. The QUBO model represents an objective function of binary variables characterized by an upper-diagonal matrix, where diagonal terms are the linear coefficients and the nonzero off-diagonal terms are the quadratic coefficients. The Ising model concerns an objective function of variables represented by physical Ising spins which consider the biases and the interactions between spins. The gray chains are connecting groups of *qubits*. Each chain should contain *qubits* that are all the same color (either all 0 or all 1). A chain is broken if it connects *qubits* of unlike colors.

8.4.4 Mapping larger graphs

There has been a discussion about the idea of employing adiabatic computations to solve NP problems, TSP is one of those cases [130] seeking if an adiabatic approach may outperform a classical one. Lately, considering the D-Wave's quantum annealing systems, more *qubits* were added and their connectivity was improved. Yet, even though there have been advances from the 2000Q to the Advantage system, is still difficult to embed problems with large number of variables into the physical *qubits*. As for example, we generate some graphs G and we experiment different G sizes. From these conducted experiments we can notice that no matter the number of edges in the graph, the embedding density is the same. In this context, only the number of nodes is relevant. The QUBO adjacency matrix corresponds to the number of nodes squared where the logical *qubits* are embedded to the physical *qubits*. In that sense, 2 logical *qubits* can require 4 physical *qubits* (2 physical *qubits* for each logical), as illustrated in Figures 8.11 and 8.12.

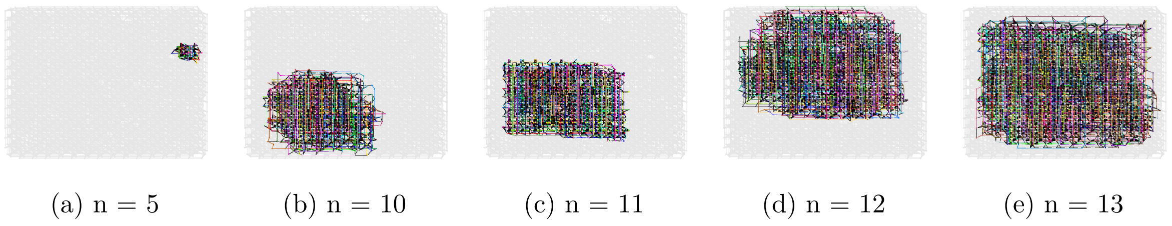

(a) n = 5 (b) n = 10 (c) n = 11 (d) n = 12 (e) n = 13

Figure 8.11: Embedded graphs into the Advantage_system1.1

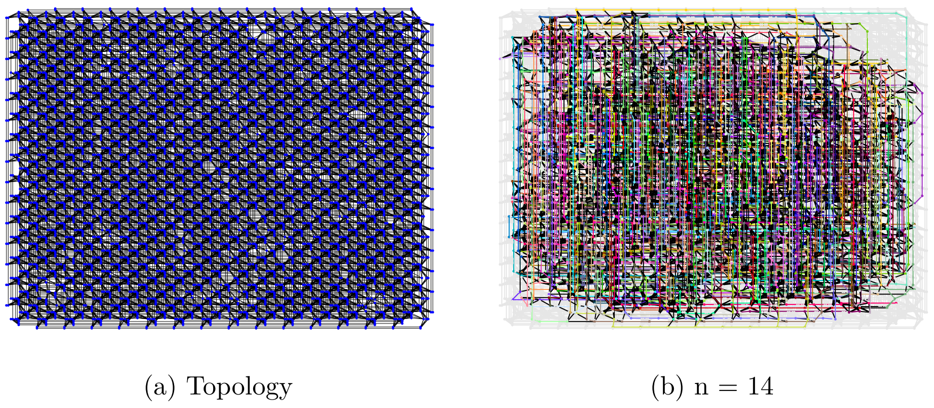

(a) Topology (b) n = 14

Figure 8.12: Embedded graph n = 14 into the Advantage_system1.1

In order to overcome the issue of embedding large problems, one possible choice is to use a solver such as *Leap's Hybrid*[2], or since we need flexibility in the input parameters and in order to generate multiple answers, to increase the chances of a possible solution, we can resort to the *QBSolv*[3]. A decomposing solver which finds a minimum value of a large QUBO problem. *QBSolv* can break the problem into smaller ones, solve these subproblems, and from those solutions construct the answer. The subproblems are solved using a classical solver with the *tabu* search algorithm [157].

[2] https://docs.dwavesys.com/docs/latest/doc_leap_hybrid.html
[3] https://docs.ocean.dwavesys.com/projects/qbsolv/en/latest/

8.5 Conclusions and future work

We implemented an algorithm that translates pseudo-Boolean constraints to quantum annealing, which solves the associated optimization problem using the QUBO solver. We perform experiments using a classical implementation of simulated annealing, a classical-quantum simulation of the same algorithm and a pure quantum environment. The problem solved is the well-known TSP. In the quantum implementation, we sampled low-energy states which correspond to the optimal solutions of the travelling salesperson problem. We resorted to quantum annealing and to encoding the problem into the D-Wave's quantum hardware graph *Pegasus*.

We conclude from our study that (a) both QA and SA attained optimal solutions for the TSP, (b) increasing the number of sweeps in the simulated annealing algorithm has an impact on achieving optimal solutions, as well as, the value of the chain strength parameter in the quantum version. Since the beginning of the experiments we observed the need to choose this input parameter in the quantum solver (if it is not specified, we may not obtain optimal solutions) and (c) it seems there is an advantage on using quantum annealing to reach better solutions faster. Moreover, the results showed that the quantum machine can have a modest advantage when finding optimal solutions over the classical machine. In addition, the quantum approach seems to be more suitable if an optimal solution is needed for a problem with larger number of nodes.

Furthermore, each graph maps differently in the hardware according to *qubits* connectivity. Besides, quantum annealing quickly jumps to the optimal solution. After finding an optimal solution, the machine stabilizes, and the other solutions are discarded.

In order to deepen and better explain the annealing processes, as future work, we intend to explore and draw the problem heuristics path to depict the mapping landscape which will allow to better describe the search space. In addition, we would like to deepen the study on embedding large problems in the D-Wave quantum annealer [152] and on quantum algorithms to accelerate constraint programming [25]. We also would like to compare our results with different types of mathematical modelling.

Chapter 9

Conclusions and future work

In this research, we study the mapping of problems to quantum devices from two perspectives: gate-based and adiabatic model. In a gate-based environment, we mapped classification problems e.g. Iris plant, traffic, and breast cancer to Noisy Intermediate-Scale Quantum (NISQ) era devices. From an adiabatic perspective, we created a generic pipeline that translates pseudo-Boolean constraints to quantum annealing based in the Quadratic Unconstrained Binary Optimization (QUBO) technique and compare the results between classical, classical-quantum simulations and a pure quantum environment. The adiabatic model allowed us to encode the graph coloring and the traveling salesperson problems into a quantum hardware graph and explore these optimization problems. We used a quantum annealer solver to sample low-energy states which are the optimal solutions of the problems. In the case of the TSP we also compute algorithms for optimal solutions, as well as, to achieve approximate solutions. We conclude from our study that both Simulated Annealing (SA) and Quantum Annealing (QA) approaches found a good number of solutions, leading to the remark that SA and QA produces good heuristics for the specific problems. Several problems in computer science are known to be NP-hard or even NP-complete and difficult to handle on traditional computers. Nowadays, quantum machines are at our disposal and aim to solve these kinds of problems. In this thesis, we showed that is possible to map and implement algorithms to solve complex problems on these quantum machines, although we still in the early stage of the technology, we can resort to methods such as, Quantum Propositional Logic (QPL).

In this chapter, we present the thesis conclusions and future work of the feature research on: quantum binary classification, quantum transfer learning for breast cancer detection, and mapping graph coloring and traveling salesperson problem to QA, as well as, other final considerations.

9.1 Quantum Binary Classification

The achieved results for the quantum version showed high *accuracy* scores using as few as 2 *qubits* and a simulator. The values are similar to other Quantum Machine Learning (QML) and Machine

Learning (ML) feature research. We encode the classical data using the method of *amplitude encoding*, in the future, we intend to explore other methods, such as, *angle*, *variational/trained* or *higher order* embedding [80, 127].

9.2 Quantum Transfer Learning for Breast Cancer Detection

We train a set of hybrid classical-quantum neural networks using TL. The goal of this research was to solve the problem of classifying full-image mammograms into malignant and benign. The conducted research shows that depending on the task, some architectures perform better than others and that this method can have advantages concerning the generalization of complex data. In sum, the overall results overstep those from the literature that also applied a classical TL technique.

(a) the classical residual neural network, without resorting to TL, achieved a maximum of 67% accuracy while our other experiments using TL achieved 84%;

(b) in addition, the AUC from 58% to 77%, shows that a hybrid-quantum network present an advantage;

(c) the overall results surpassed those from the literature that also applied a classical transfer learning technique:

- accuracy of 84% against 76.9%
- AUC of 77% against 74.9%
- specificity of 100% surpassed 64.91%

(d) but not for the recall, where we attained a recall of 69% and 84.88% was mentioned in the literature, however the quantum version got 81%;

(e) other approach could be to perform classification after separating the dataset into masses and calcifications;

(f) as future work plot a representation of the data in the Hilbert Space;

(g) in this research we conduct the experiments using the angle embedding technique, in the future other embeddings can be considered.

9.3 Mapping Graph Coloring to Quantum Annealing

We encoded the graph coloring problem into the quantum hardware graph *Chimera* and we used a quantum annealer solver to sample low-energy states to achieve the optimal solutions resorting to the adiabatic model. In this research, we implemented an algorithm to translate pseudo-Boolean constraints to QA based on QUBO and symbolic computing. Besides, the pure classical (QUBO and SA), and the quantum environment simulations in a classical machine, we also compare implementations using the QUBO and the D-Wave quantum machine and performing polynomial reduction degree when needed (our own implementation and using a D-Wave's library).

In addition, we study how varying the penalties α and β has impact in the number of solutions found. In order to better control the experiments, we decided to fix the α and β, one at a time, and analyze from there the solutions found as a way to obtain a better representation of the explored landscape and the search space. In summary, we concluded that:

(a) experiments made varying α and β have small impact on the number of solutions found;

(b) high β values cause the energy values to vary substantially;

(c) both SA and QA approaches found a good number of solutions, in particular the QA procedure.

As further work, we intend to perform the experiments using larger graphs and explore other problems.

9.4 Mapping a Logical Representation of TSP to Quantum Annealing

We encoded the traveling salesperson problem into the quantum hardware graph *Pegasus*. Resorting to the adiabatic model we used a quantum annealer solver to sample low-energy states to achieve the optimal solutions. In this research, we developed an algorithm to translate pseudo-Boolean constraints to QA based on the QUBO technique and compare the results between pure classical (on a traditional computer), classical-quantum simulations (on a traditional computer) and a pure quantum environment (on D-Wave's). As such, we sample low-energy states which aim to be the optimal solutions of the TSP. In brief, we concluded that:

(a) both QA and SA attained optimal solutions for the TSP;

(b) we notice that increasing the number of *sweeps* in the SA algorithm had impact to achieve the optimal solutions;

(c) we notice that changing the *chain strength* parameter also had impact to achieve the optimal solutions;

(d) the quantum version show a slight advantage when finding optimal solutions over the classical version;

(e) the quantum version can be more suitable if an optimal solution is needed for a problem with a large number of nodes.

In the forthcoming studies, we intend to make the graphic recreation of the search space traversed of the problem heuristics path. Besides, we have interest in deepen the study on large problems embedding onto the D-Wave's and the exploration of quantum algorithms to accelerate constraint programming. In addition, perform experiments using the QUBO directly without the propositional logic part.

9.5 Remark

Nowadays, we can execute quantum scripts on multiple quantum hardware architectures, from superconducting *qubits* to trapped-ions. Quantum tools are diverse, e.g., IBM's Qiskit, Xanadu's PennyLane, D-Wave's Ocean, Google's Cirq, Microsoft's QDK, Rigetti's Forest, and are accurately designed for the best algorithms achievements and performance on those architectures. However, these tools, like many others, have advantages and disadvantages which can be related with the purpose of the code implemented in each of them, or tasks to be performed, hardware limitations, or the currently feasibility of the technology itself. For instance, the IBM's Qiskit, Xanadu's PennyLane and D-Wave's Ocean are all written in Python which allows for easy reading and understanding the composed quantum scripts. On the other hand, an *n-qubit* algorithm has to be implemented according to hardware limitations, considering the coupling maps. As an example, in the gate-based systems (e.g. IBM's), on specific cases, we may need to implement a n-bit Toffoli gate, and that type of operation needs a coupling map with a higher degree of connectivity. Furthermore, error mitigation is also a hot topic in quantum computing, since the execution time and also gate errors increase with the number of circuits and may affect the accuracy of the algorithms. Nevertheless, if we resort to quantum simulators the results coincide with the theory and are not affected by decoherence and gate errors. However, in those cases we are not taking full advantage of quantum systems capabilities such as the D-Wave's hybrid approach resorting to quantum annealing, where D-Wave's hybrid solvers use a combination of classical and quantum resources, designed to solve problems that exceed the size of the Quantum Processing Unit (QPU).

The D-Wave's QPU has more than 5,000 *qubits*, the advantage of this complex system is that it is designed to applications such as optimization problems and probabilistic sampling problems, which need the real minimum energy and low-energy samples. Ocean software stack provides tools that implements the computations needed to transform a problem to a form solvable on a quantum solver, such as, the Binary Quadratic Model (BQM) class that contains Ising and quadratic QUBO models used by samplers, a library to construct a BQM from a constraint satisfaction problem over binary variables, tools for working with Chimera graphs, tools to optimize chain strength, tools to embed Ising problems onto quantum annealers, tools to solve a constraint satisfaction problem (CSP) using an Ising model or a QUBO and, a decomposing solver which finds a minimum value of a large QUBO problem by splitting it into pieces.

Qiskit construction is based in modularly, to be easier the addition of extensions for circuit optimizations and backends. On the other hand, Qiskit provides a set of circuits and pulses, quantum algorithms for machine learning, optimization and chemistry applications, remote-access backends to execute code on multiple quantum hardware architectures, noise mitigation implementations to decrease the impact of noise using built-in modules for noise characterization and circuit optimization. Qiskit has some classification algorithms such as QSVM (Quantum Support Vector Machine), VQC (Variational Quantum Classifier), and the QGAN (Quantum Generative Adversarial Network) algorithm. The same and a wider broad variety of quantum machine learning algorithms can be found in Xanadu's PennyLane. In brief, Qiskit and PennyLane explore variational techniques and both provide quantum and classical computations. Besides the machine learning tools, PennyLane is designed to allow computations that can include multiple quantum devices from unlike vendors. It allows external quantum devices to be easily added to PennyLane by installing plugins (Qiskit, Cirq, Rigetti Forest, Microsoft QDK, Strawberry Fields, AQT, Honeywell, ProjectQ). It has many features and advantages that come into play, such as, built-in automatic differentiation of quantum circuits, support for hybrid quantum and classical models and connects quantum hardware with PyTorch, TensorFlow, and NumPy, provides optimization, variational circuits which are quantum algorithms that depend on tunable parameters and can be optimized, hybrid computations and the capability to train circuits, the paradigm of making quantum algorithms differentiable and thereby trainable, representation of classical data as a quantum state, the mathematical map that embeds classical data into a quantum state, and the capability to explore multiple backends.

The current technologies allow to adapt the physical systems parameters to each problem, shifting quantum computations paradigms from universal quantum computers, fault-tolerant QPUs (still years away), to the quantum machine learning paradigm on near-term quantum devices where the algorithms are learned. However, all theses quantum systems, unlike what we are used to on our classic machines, cannot execute jobs at once. Since the quantum devices are available on the cloud, these are queued and only executed when the QPU is available.

9.6 Future Work

In quantum computation, we deal with errors in the form of noise, faults and loss of quantum coherence (decoherence), caused by e.g. temperature fluctuations, electromagnetic waves and interactions with the surrounding environment. These effects, can change the quantum properties of the quantum device. In current quantum devices the interplay between coherence and providing the robust error correction required for large-scale computation is not yet being achieved. Furthermore, NISQ devices are limited to qubit number, qubit connectivity and qubit/gate fidelity. In that sense, it is hard to engineer and program a quantum computer. Strategies are being considered, namely, hybrid quantum-classical algorithms approaches to deal with noisy environments.

In order to evolve the quantum computing, many lines of investigation can be followed, from the use only of simulators in supercomputers or real quantum machines, from the study of its limitations, to the creation of new quantum algorithms. The quantum realm still has a vast and enthusiastic path of exploration ahead. In this sense, and given that we are dealing with very recent technologies, many suggestions for future work can be enumerated, as there are still many open questions and possibilities.

(a) Currently NISQ devices are noisy and slow. In the context of QML, training a quantum circuit, usually requires many measurements which is a procedure that can take a long time, e.g. in the optimization step when computing the gradient. As such, contributions concerning the decrease of the training time, can be of great interest for the community.

(b) Explorations of other quantum technologies and platforms, since there exist different technologies at the core of each platform. For example, IBM, semiconductor quantum computing which requires the machine to be supercooled, and IonQ, trapped ion quantum computing which allows the machines to run at room temperature. In the same way, there are problems that fit naturally to adiabatic computations, such as optimization problems, the same goes for other types of quantum computations, with technologies suitable for different types of challenges.

(c) Perform deeper analysis and compare the mapping of problems to the *qubits*. Not the same problem, as different technologies have more suitable problems, but better explore the architectures limitations, and difficulties to map any problem to a quantum machine. The connectivity of the *qubits* plays an important role on the mapping, sometimes conducting the mapping of problems to quantum machines is too rigid.

(d) In the context of the Adiabatic Quantum Computation (AQC), explore the capabilities of the D-Wave's hybrid solver which use quantum and classical resources together. Besides, for large problems, explore the *QBSolv* which is a decomposing solver that finds a minimum value of a large QUBO problem by splitting it into small problems. In addition, also investigate the discrete quadratic model (DQM) which is a polynomial over discrete variables, where a discrete variable represents categorical values of some domain, such as, blue, red, black, gray.

(e) In the context of the QPL, a propositional logic mapping was conducted in this thesis. However, it would be also interesting to build a mapping capable to map first-order logic (predicate logic) problems into quantum devices. Propositional logic is the foundation of first-order logic, however, first-order logic uses quantifiers or relations that could be translated to a quantum machine and be an advantage to problem-solving. There are already some works that deal with first-order logic in a quantum perspective [21, 216].

(f) In addition, a comparison of the approach developed in this thesis to solve problems formulated in QPL with other works would be an interesting challenge [20].

Currently, quantum technologies look for speed-up in a wide range of applications. However, quantum states are extremely fragile, and fault-tolerant systems [35] need precise error correction and error mitigation to attain the speed-ups over their counterparts. To suppress the noise in quantum gates, NISQ technology (e.g. quantum computers with 50-100 *qubits*) aims to overcome capabilities of classical digital computers [161]. The quantum computer performance depends on parameters, such as, number of physical *qubits*, connectivity between *qubits*, or the number of gates or operations that can be run in parallel. Since we are dealing with not yet fully deployed technologies, some studies discuss that the hybrid quantum-classical variational approach may be used to variationally suppress certain types of quantum errors. In that sense, a wide broad research is been performed [1, 4, 176] where we can notice that e.g. QML is being constantly updated since it handles fast growing quantum technologies and innovative development. Quantum tools and platforms have been proposed and can be used to implement QML algorithms [62], namely, Qiskit from IBM Quantum or Strawberry Fields and PennyLane from Xanadu. ML tasks can be implemented in a quantum context, and is possible to compare complexity and performance with classical and parallel implementations. These implementations are suitable to solve demanding data-related issues using hybrid quantum-classical models [175]. In this research, we describe the key recent developments and progress in a wide research exploring Artificial Intelligence (AI) scenarios in a quantum domain [54], opening the topic to new perspectives and new developments.

Appendices

Appendix A

Algorithm: Quantum TSP based in Hopfield-Tank

Listing A.1: Quantum TSP based in Hopfield-Tank algorithm

```python
# Script by Carla Silva 2021 :: TSP Quantum implementation based on:
# Hopfield-Tank TSP formulation
# https://link.springer.com/content/pdf/10.1007/BF00339943.pdf
""" TSP

    Formulation of the problem for a graph random G=(V,E) with a number of cities n.

"""
#### NOTE: GIVES OPTIMAL SOLUTIONS UNTIL N<=6.

from pyqubo import Array, solve_qubo, Constraint
import matplotlib.pyplot as plt
import networkx as nx
import time
import sys
import random
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
import neal
import pandas as pd
from dwave.system.samplers import DWaveSampler
from dwave.system.composites import EmbeddingComposite
import dimod
import dwave.inspector

def plot_city(cities, n, sol = {}):
    n_city = len(cities)
    cities_dict = dict(cities)

    G = nx.Graph()
    for city in cities_dict:
        G.add_node(city)

    for i in range(n_city):
        for j in range(n_city):
            G.add_edge(i, j,distance=round(dist(i, j, cities),2))

    f = plt.figure()
    pos = nx.spring_layout(G)
    labels = nx.get_edge_attributes(G,'distance')
    nx.draw(G,pos, with_labels=1, node_size=400, font_weight='bold', font_color='w', node_color = 'black', edge_color = 'black')
    nx.draw_networkx_edge_labels(G,pos,label_pos=0.2, edge_labels=labels,font_color='black')
    plt.axis("off")
    f.savefig('figAllDistancesN'+str(n)+'.pdf', bbox_inches='tight')

    G = nx.Graph()
    for city in cities_dict:
        G.add_node(city)

    # draw path
    city_order = []
    for i, v in sol.items():
        if v == 1:
            city_order.append(int(i[2]))
            city_order.append(int(i[5]))
    city_order = list(city_order)
    n_city = len(city_order)
    for k in range(0,n_city-1,2):
        i = city_order[k]
        j = city_order[k+1]
        G.add_edge(i, j, distance=round(dist(i, j, cities),2))

    sumD = 0
    for u,v in G.edges:
        sumD += G[u][v]['distance']

    f = plt.figure()
    pos = nx.spring_layout(G)
    labels = nx.get_edge_attributes(G,'distance')
    nx.draw(G,pos, with_labels=1, node_size=400, font_weight='bold', font_color='w', node_color = 'orange', edge_color = 'orange')
    nx.draw_networkx_edge_labels(G,pos,label_pos=0.5, edge_labels=labels,font_color='orange')
    plt.axis("off")
    f.savefig('figN'+str(n)+'Distance='+str(sumD)+'.pdf', bbox_inches='tight')

def dist(i, j, cities):
    pos_i = np.array(cities[i][1])
    pos_j = np.array(cities[j][1])
    return np.linalg.norm(np.subtract(pos_i,pos_j)) # Euclidean distance

def exp1(n, v, A):
    exp = 0.0
    for i in range(n):
        for j in range(n):
            for k in range(n):
                if (k != j):
                    exp += v[i,j]*v[i,k]

    exp = (A/2) * Constraint(exp, label="exp")
    return(exp)
```

```python

def exp2(n, v, B):
    exp = 0.0
    for j in range(n):
        for i in range(n):
            for k in range(n):
                if (k != i):
                    exp += v[i,j]*v[k,j]

    exp = (B/2) * Constraint(exp, label="exp")
    return(exp)

def exp3(n, v, C):
    exp = 0.0
    for i in range(n):
        for j in range(n):
                exp += (v[i,j]-n)**2

    exp = (C/2) * Constraint(exp, label="exp")
    return(exp)

def exp4(n, v, D):
    exp = 0.0
    for i in range(n):
        for j in range(n):
            if (j != i):
                for k in range(n-1):
                    exp += dist(i,j,cities)*(v[i,k])*(v[j,k+1]+v[j,k-1])

    exp = (D/2) * Constraint(exp, label="exp")
    return(exp)

if __name__ == "__main__":


    """## Traveling Salesman Problem (TSP)

    Find the shortest route that visits each city and returns to the origin city.
    """

    n = int(sys.argv[1]) # Nodes/Cities

    """Prepare binary vector with  bit $(i, j)$ representing to visit $j$ city at time
        ↪ $i$"""

    v = Array.create('v', (n, n), 'BINARY')


    random.seed(123)
    a = tuple((random.randint(0,50),random.randint(0,50)) for i in range(n))
    b = tuple((i,a[i]) for i in range(n))
    cities = list(b)

    # maximum distance
    maxdist = 0
    for i in range(n):
        for j in range(n):
            if dist(i,j,cities) > maxdist:
                maxdist = dist(i,j,cities)

    n_city = len(cities)
    cities_dict = dict(cities)

    G = nx.Graph()
    for city in cities_dict:
        G.add_node(city)

    for i in range(n_city):
        for j in range(n_city):
            G.add_edge(i, j,distance=round(dist(i, j, cities),2))

    dU = G.size(weight="weight")
    dL = dU/2
    C = 1
    D = 1 / dU
    B = (3 * dU) + C
    A = B - (D * dL)
    print("dU,dL,A,B,C,D",dU,dL,A,B,C,D)
    orig_stdout = sys.stdout

    f = open('tspResults_'+'n'+str(n)+'.txt', 'w')
    sys.stdout = f

    start_time = time.time()

    print("--------------------------------------------------------------------------------")
    print("\n# TSP PROBLEM WITH n CITIES ON QUANTUM SOLVER #\n")
    print("--------------------------------------------------------------------------------")

    print("--------------------------------------------------------------------------------")
    print("1st expression:")
    print("--------------------------------------------------------------------------------")
    print(exp1(n, v, A))

    print("--------------------------------------------------------------------------------")
    print("2nd expression:")
    print("--------------------------------------------------------------------------------")
    print(exp2(n, v, B))

    print("--------------------------------------------------------------------------------")
    print("3rd expression:")
    print("--------------------------------------------------------------------------------")
    print(exp3(n, v, C))

    print("--------------------------------------------------------------------------------")
    print("4th expression:")
    print("--------------------------------------------------------------------------------")
    print(exp4(n, v, D))

    # Define hamiltonian H
    H = exp1(n, v, A) + exp2(n, v, B)  + exp3(n, v, C)  + exp4(n, v, D)


    # Compile model
    model = H.compile()

    # Create QUBO
    qubo, offset = model.to_qubo()
```

```python

    print("------------------------------------------------------------------------")
    print("\nQUBO:\n")
    print("------------------------------------------------------------------------")

    print(qubo)

    print("------------------------------------------------------------------------")
    print("\nD-WAVE OUTPUT:\n")
    print("------------------------------------------------------------------------")

    sampler = EmbeddingComposite(DWaveSampler(endpoint='https://cloud.dwavesys.com/sapi', token='', solver={'topology__type': 'pegasus'}))


    # Submit to the D-Wave with nr number of reads
    # Reads number
    nr = 10000
    # Chain strength
    c = max(qubo.values()) + 10

    print("chain break: ",c)

    start_time = time.time()

    response = sampler.sample_qubo(qubo, num_reads = nr, auto_scale=True, return_embedding=True, chain_strength = c, answer_mode = "raw")

    print("QPU access time (us):\t", response.info['timing']['qpu_access_time'])

    elapsed_time = time.time() - start_time

    print("Wall time (us):\t\t", elapsed_time*1000000)

    # Inspect
    dwave.inspector.show(response)

    # create dataframe if we want to store all values
    df = []
    minD = sys.maxsize # long value
    for datum in response.data(['sample', 'energy', 'num_occurrences','chain_break_fraction']):
        df.append({"Sample": datum.sample, "Energy": datum.energy, "Occurrences": datum.num_occurrences, "Chain_break_fractions": datum.chain_break_fraction})
        print(datum.sample, "Energy: ", datum.energy, "Occurrences: ", datum.num_occurrences,"Chain break fractions:", datum.chain_break_fraction)

        if (sum(datum.sample.values())==n): # Consider only instances with n nodes (same as the n of the problem).
            s = 0
            for i, v in datum.sample.items():
                if ((int(i[2]) != int(i[5])) and (int(v) == 1)): # Found an edge. Check if i != j (different nodes).
                    s = s + 1
            if ((s == n)): # Check the number of edges = number of nodes.
                ### Draw graph
                n_city = len(cities)
                cities_dict = dict(cities)
                G = nx.Graph()
                for city in cities_dict:
                    G.add_node(city)
                # draw path
                city_order = []
                for i, v in datum.sample.items():
                    if v == 1:
                        city_order.append(int(i[2]))
                        city_order.append(int(i[5]))

                city_order = list(city_order)
                n_city = len(city_order)
                for k in range(0,n_city-1,2):
                    i = city_order[k]
                    j = city_order[k+1]
                    G.add_edge(i, j, distance=round(dist(i, j, cities),2))

                sumD = 0
                for u,v in G.edges:
                    sumD += G[u][v]['distance']

                # Check if graph is connected and with a cycle.
                try:
                    G1 = list(nx.cycle_basis(G.to_undirected())) # check if is one cicle
                        ↪ with n nodes.
                    if (nx.is_connected(G) and nx.find_cycle(G) and (len(G1[0])==n) and (
                        ↪ sumD < minD)):
                        minD = sumD
                        minE = datum.energy
                        maxO = datum.num_occurrences
                        sample = datum.sample
                        chain = datum.chain_break_fraction
                except:
                    break

    df = pd.DataFrame(df)
    df.to_csv('TSP'+str(n)+'.csv',index=False)
    pd.set_option('display.float_format', lambda x: '%.20f' % x)
    pd.options.display.max_colwidth = 10000
    print(df.to_string(index=False))

    print("---------------------------------------------------------------------")
    print("\nSAMPLE WITH MINIMUM ENERGY AND MAXIMUM OCCURRENCES:\n")
    print("---------------------------------------------------------------------")
    print(sample, "Energy: ", minE, "Occurrences: ", maxO, "Chain break fractions:",
        ↪ chain)

    plot_city(cities, n, sample)

    print("---------------------------------------------------------------------")
    print("\nTIME\n")
    print("---------------------------------------------------------------------")

    print("Times:\t\t", response.info)

    sys.stdout = orig_stdout
```

Appendix B

Algorithm: Quantum Binary Classification

Listing B.1: Quantum binary classification algorithm

```
# Script by Carla Silva 2020 :: Quantum binary classification using PennyLane resources
from pennylane.templates.embeddings import AmplitudeEmbedding
import pennylane as qml
from pennylane import numpy as np
from pennylane.optimize import AdamOptimizer
from sklearn import datasets
from sklearn import preprocessing
from sklearn.datasets import load_iris
from numpy import *
import itertools
import pandas as pd
import time
import sys
from matplotlib import pyplot as plt
import seaborn as sns

font = {'size' : 20}
plt.rc('font', **font)

file = "classification.txt"

start_time = time.time()
orig_stdout = sys.stdout
f = open(file, 'w')
sys.stdout = f

dev = qml.device('default.qubit', wires=2, shots=1024)

def layer(W):

    qml.Rot(W[0, 0], W[0, 1], W[0, 2], wires=0)
    qml.Rot(W[1, 0], W[1, 1], W[1, 2], wires=1)

    qml.CNOT(wires=[0, 1])

@qml.qnode(dev)
```

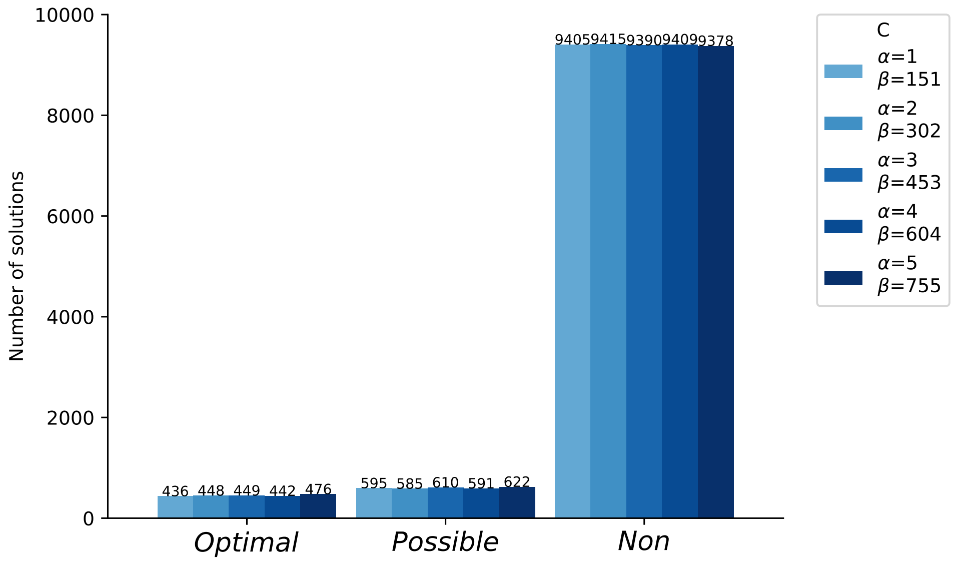
```python
def circuit(weights, feat=None):

    AmplitudeEmbedding(feat, [0,1], pad=0.0, normalize=True)

    for W in weights:
        layer(W)

    return qml.expval(qml.PauliZ(0))

def variational_classifier(var, feat=None):

    weights = var[0]
    bias = var[1]

    return circuit(weights, feat=feat) + bias

def square_loss(labels, predictions):

    loss = 0
    for l, p in zip(labels, predictions):
        loss = loss + (l - p) ** 2
    loss = loss / len(labels)

    return loss

def accuracy(labels, predictions):

    loss = 0
    for l, p in zip(labels, predictions):
        if l == p:
            loss = loss + 1
    loss = loss / len(labels)

    return loss


def loss(weights, features, labels):

    predictions = [variational_classifier(weights, feat=f) for f in features]

    return square_loss(labels, predictions)

iris = datasets.load_iris()
data = iris.data[(iris.target == 0) | (iris.target == 1)]

N = data.shape[1]
X = data[:, 0:N]
Y = data[:, -1]
Y = iris.target[(iris.target == 0) | (iris.target == 1)]
Y[Y == 0] = -1

fig = plt.figure()
plt.scatter(X[:,0][Y== 1], X[:,1][Y== 1], c='orange', marker='o', edgecolors='k', label="class 1")
plt.scatter(X[:,0][Y==-1], X[:,1][Y==-1], c='purple', marker='o', edgecolors='k', label="class -1")
plt.xlabel('X')
plt.ylabel('Y')
plt.axis('off')
```

```
fig.savefig('originalData1a.png', bbox_inches= 'tight',pad_inches = 0.1, dpi=300)

f, ax = plt.subplots()
sns.kdeplot(X[:,0][Y== 1], X[:,1][Y== 1],cmap="Oranges",legend=True, label="class 1")
sns.rugplot(X[:,0][Y== 1], color="gray", ax=ax)
sns.rugplot(X[:,1][Y== 1], color="gray", vertical=True, ax=ax)
sns.kdeplot(X[:,0][Y==-1], X[:,1][Y==-1],cmap="Purples",legend=True, label="class -1")
sns.rugplot(X[:,0][Y==-1], color="gray", ax=ax)
sns.rugplot(X[:,1][Y==-1], color="gray", vertical=True, ax=ax)
ax.set_xlabel('X')
ax.set_ylabel('Y')
leg = ax.legend()
leg.legendHandles[0].set_color('orange')
leg.legendHandles[1].set_color('purple')
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
f.savefig('originalData1b.png', bbox_inches = 'tight',pad_inches = 0.1, dpi=300)

features = np.array([x for x in X])

np.random.seed(0)
num_data = len(Y)
num_train = int(0.75 * num_data)
index = np.random.permutation(num_data)
feats_train = features[index[:num_train]]
Y_train = Y[index[:num_train]]
feats_val = features[index[num_train:]]
Y_val = Y[index[num_train:]]

X_train = X[index[: num_train]]
X_val   = X[index[num_train :]]

np.random.seed(0)
num_qubits = 2
num_layers = 6
var_init = (np.random.randn(num_layers, num_qubits, 3), 0.0)
opt = AdamOptimizer(0.01)
batch_size = 8
loss_progressT = []
loss_progressV = []
train_progress = []
val_progress = []
var = var_init
n = 50

for it in range(n):

    batch_index = np.random.randint(0, num_train, (batch_size, ))
    feats_train_batch = feats_train[batch_index]
    Y_train_batch = Y_train[batch_index]
    var = opt.step(lambda v: loss(v, feats_train_batch, Y_train_batch), var)
    predictions_train = [np.sign(variational_classifier(var, feat=f)) for f in
        ↪ feats_train]
    predictions_val = [np.sign(variational_classifier(var, feat=f)) for f in feats_val]
    acc_train = accuracy(Y_train, predictions_train)
    acc_val = accuracy(Y_val, predictions_val)
    lossT = loss(var, feats_train, Y_train)
    lossV = loss(var, feats_val, Y_val)
    loss_progressT.append(lossT)
    loss_progressV.append(lossV)
    train_progress.append(acc_train)
    val_progress.append(acc_val)
    print("Iter: {:5d} | Loss train: {:0.7f} | Acc train: {:0.7f} | Loss validation:
        ↪ {:0.7f} | Acc validation: {:0.7f} "
          "".format(it+1, lossT, acc_train, lossV, acc_val))

print("--- %s seconds ---" % (time.time() - start_time))

print("CIRCUIT")
print(circuit.draw())

sys.stdout = orig_stdout

Plotting the loss, train and validation
fig = plt.figure()
plt.plot(loss_progressT, label='Train (loss)', color='blue')
plt.plot(train_progress, label='Train (acc)', color='green')
plt.plot(loss_progressV, label='Validation (loss)', color='red')
plt.plot(val_progress, label='Validation (acc)', color='orange')
plt.ylabel('Loss | Accuracy')
plt.xlabel('Step')
plt.legend(loc='upper center',bbox_to_anchor=(0.5, 1.25), ncol=2, fancybox=True, shadow=
    ↪ True, frameon=False)
fig.savefig('lossAccuracy.png', bbox_inches = 'tight',pad_inches = 0.1, dpi=300)

print("--- %s seconds ---" % (time.time() - start_time))
sys.stdout = orig_stdout
```

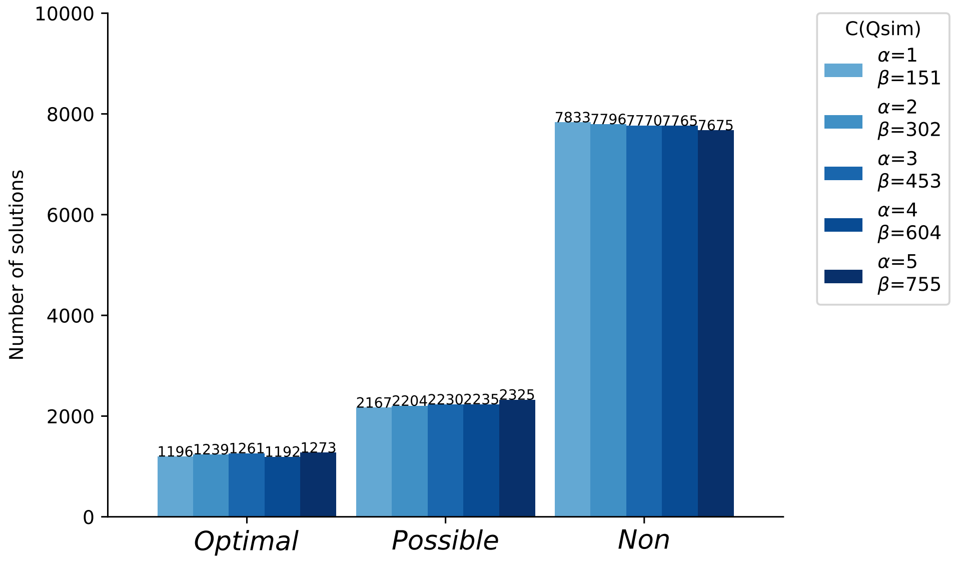
Appendix C

Mapping Graph Coloring to Quantum Annealing

C.1 Varying α and β

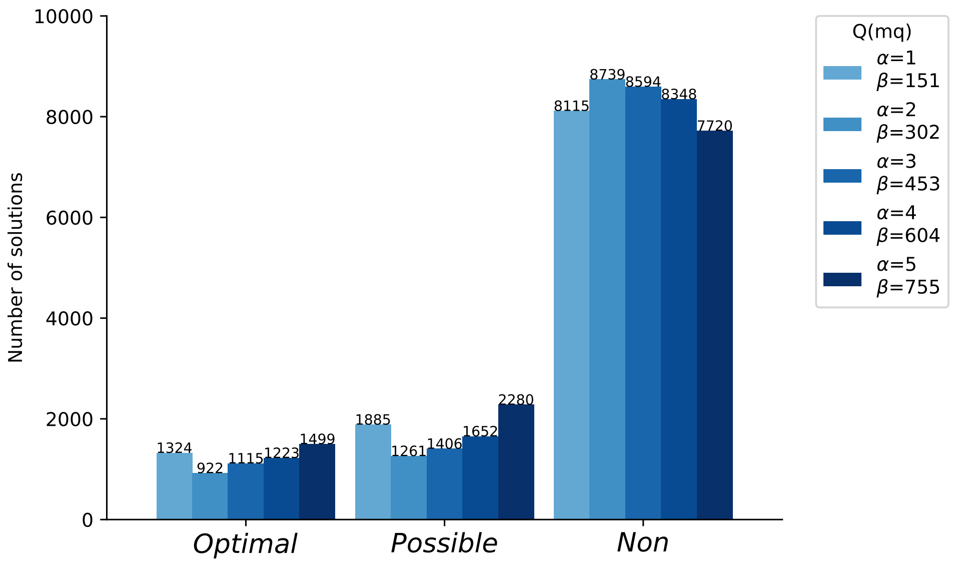


(a) `C`

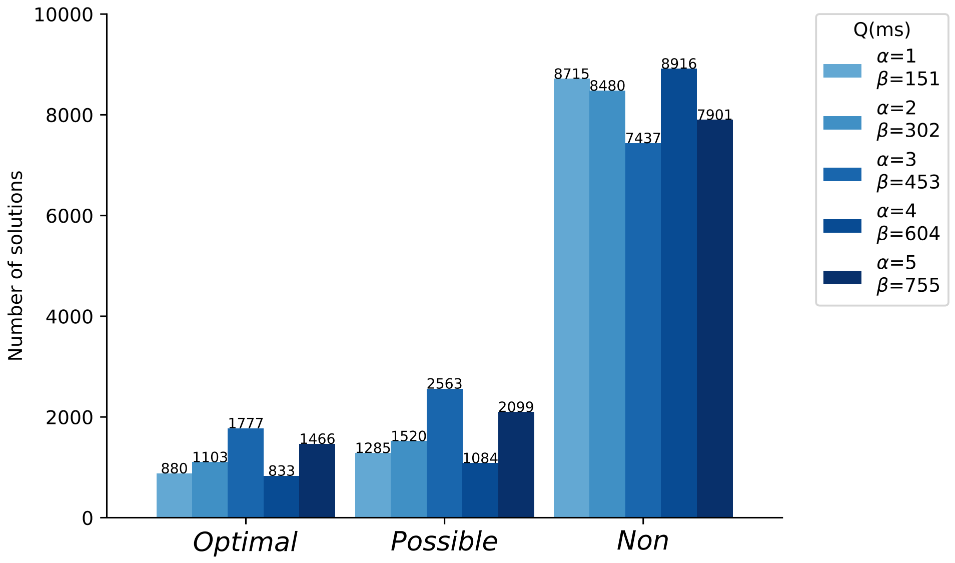


(b) `C_Q_sim`

Figure C.1: Solutions for pure classical `C` and classical quantum simulator `C_Q_sim` and both methods of polynomial reduction `Q_mq` and `Q_ms` (varying α and β) (First part)

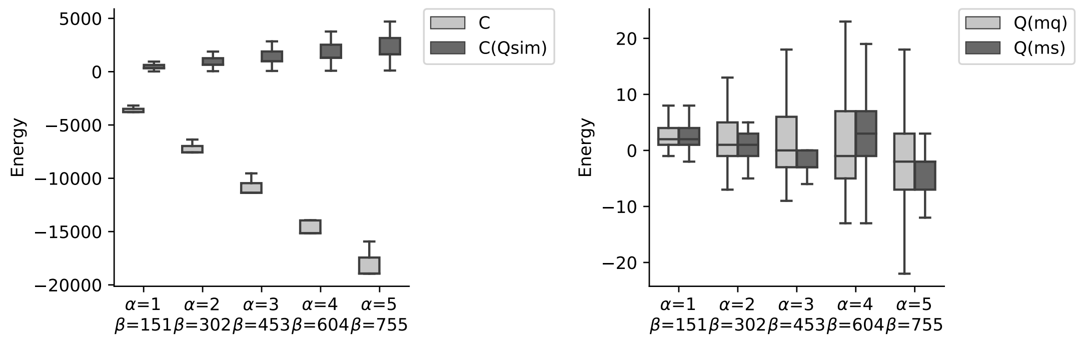


(a) `Q_mq`

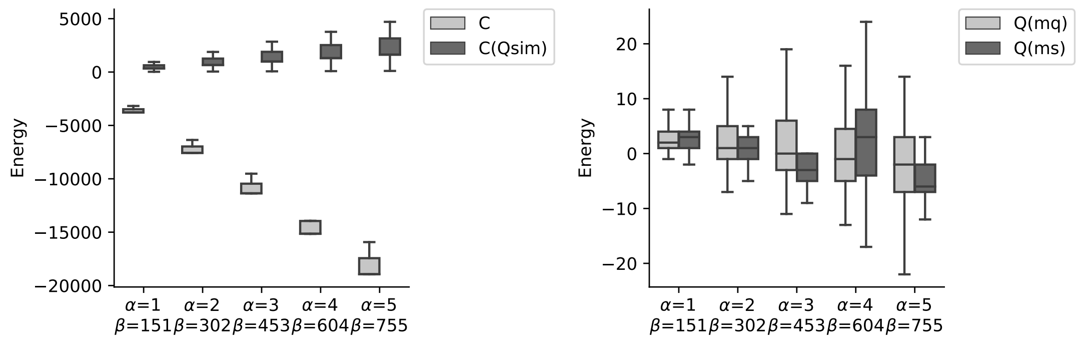


(b) `Q_ms`

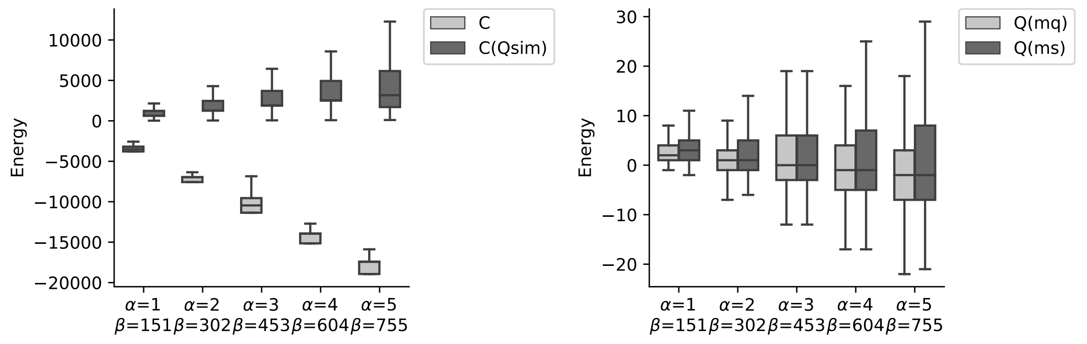
Figure C.2: Solutions for pure classical `C` and classical quantum simulator `C_Q_sim` and both methods of polynomial reduction `Q_mq` and `Q_ms` (varying α and β) (Second part)



(a) Optimal solutions

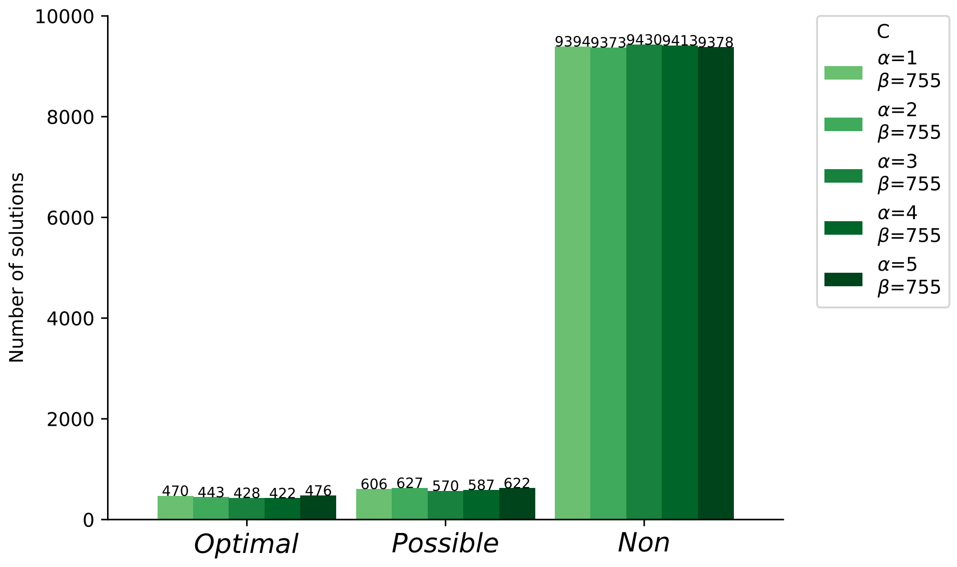


(b) Possible solutions

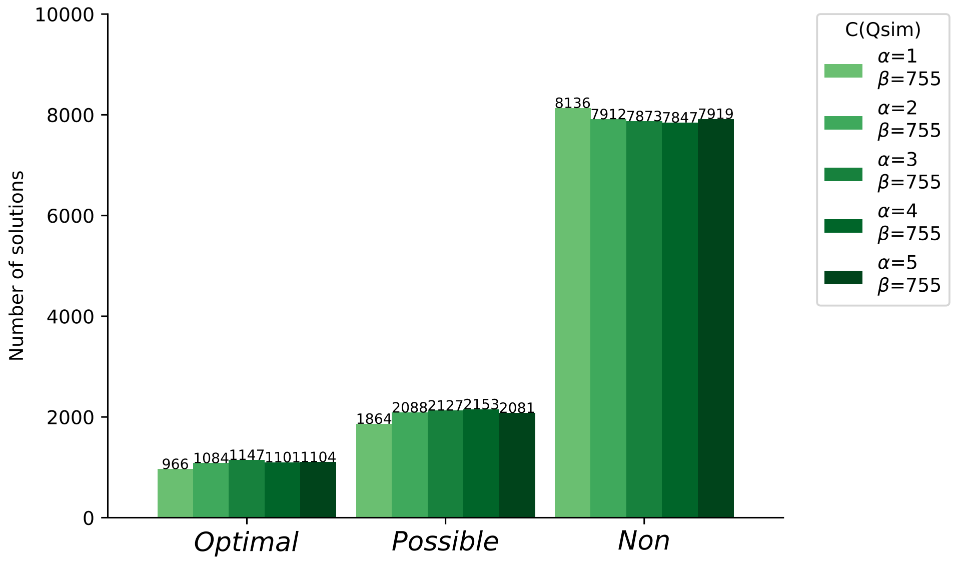


(c) Non solutions

Figure C.3: Energy values of classical `C` and both methods of polynomial reduction (`Q_mq` and `Q_ms`) (varying α and β)

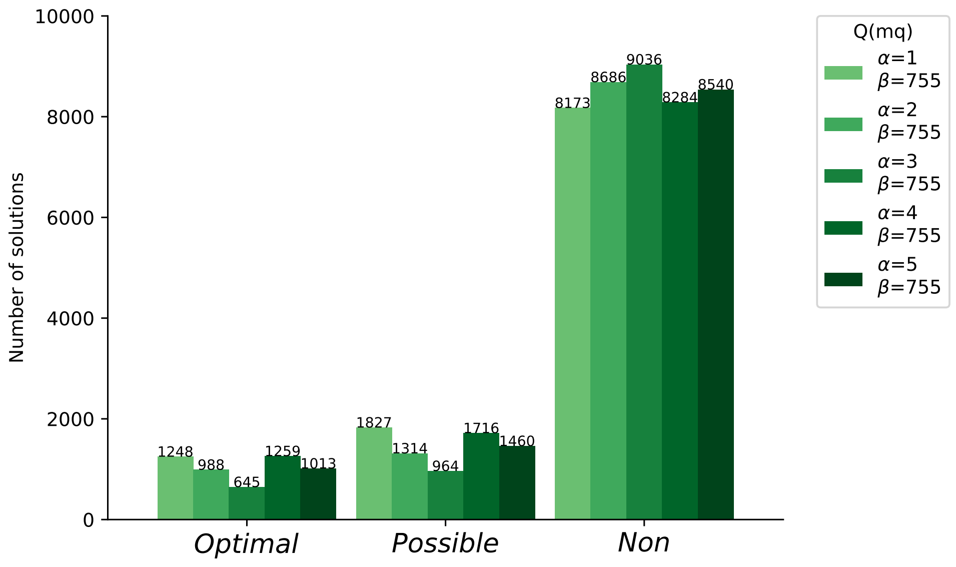


(a) `C`

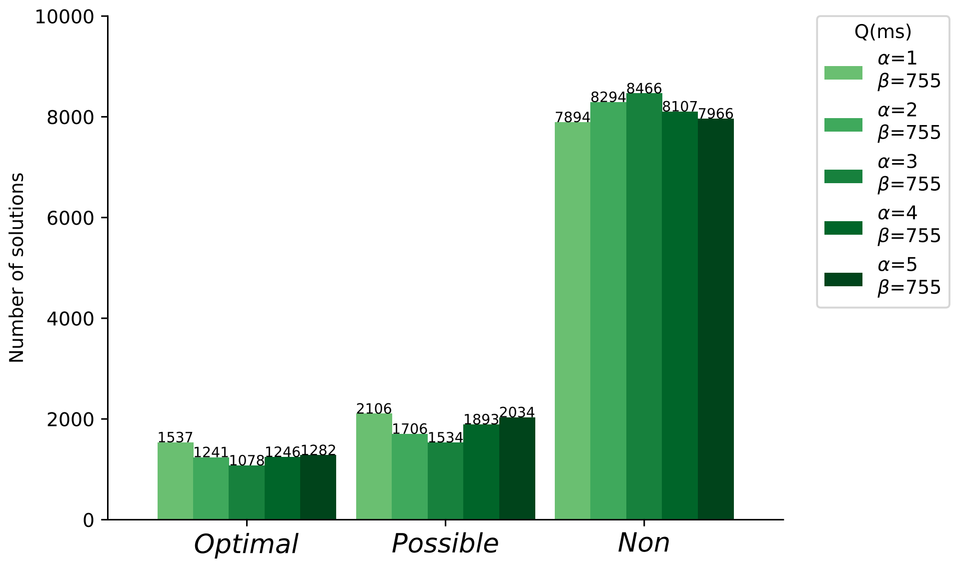


(b) `C_Q_sim`

Figure C.4: Solutions for pure classical `C` and classical quantum simulator `C_Q_sim` and both methods of polynomial reduction `Q_mq` and `Q_ms` (varying α) (First part)

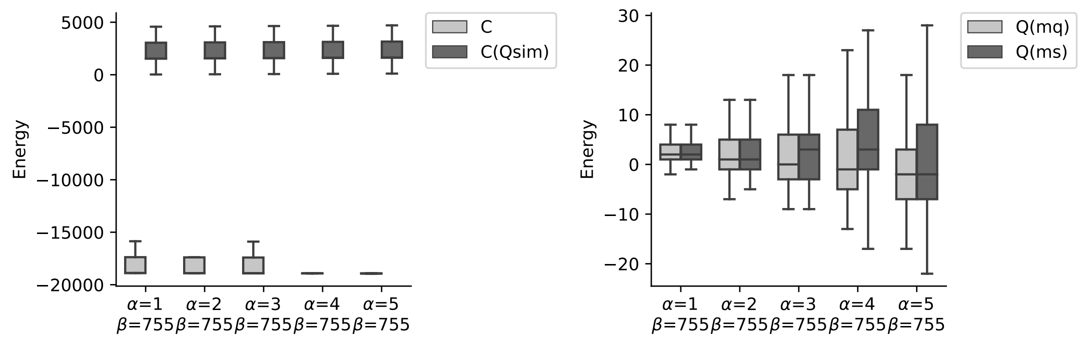


(a) Q_mq

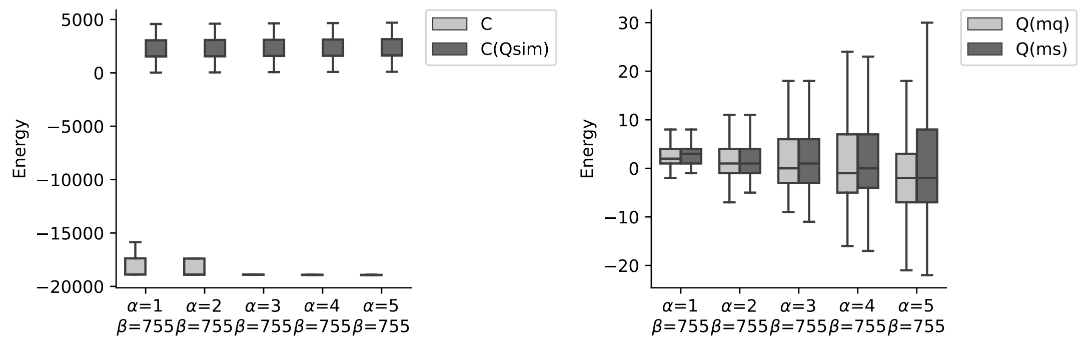


(b) Q_ms

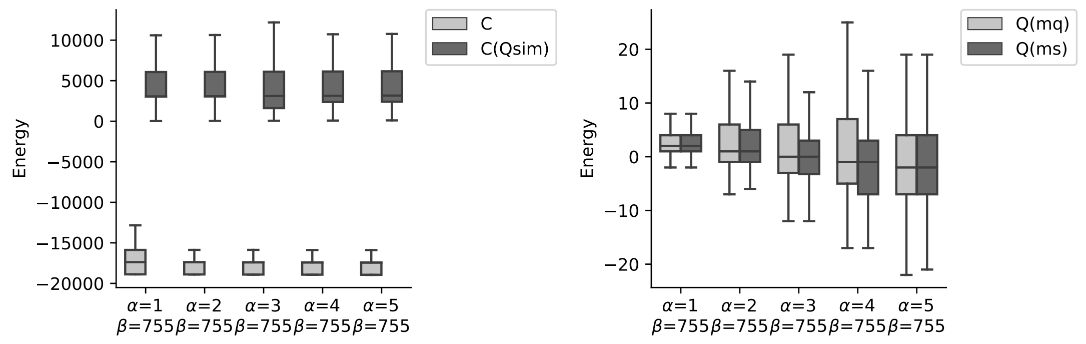
Figure C.5: Solutions for pure classical C and classical quantum simulator C_Q_sim and both methods of polynomial reduction Q_mq and Q_ms (varying α) (Second part)



(a) Optimal solutions

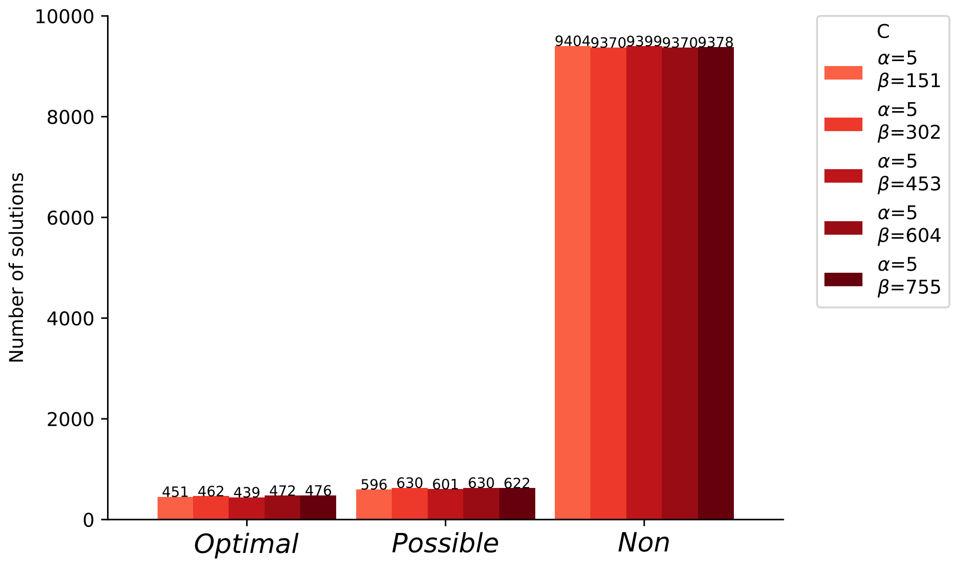


(b) Possible solutions

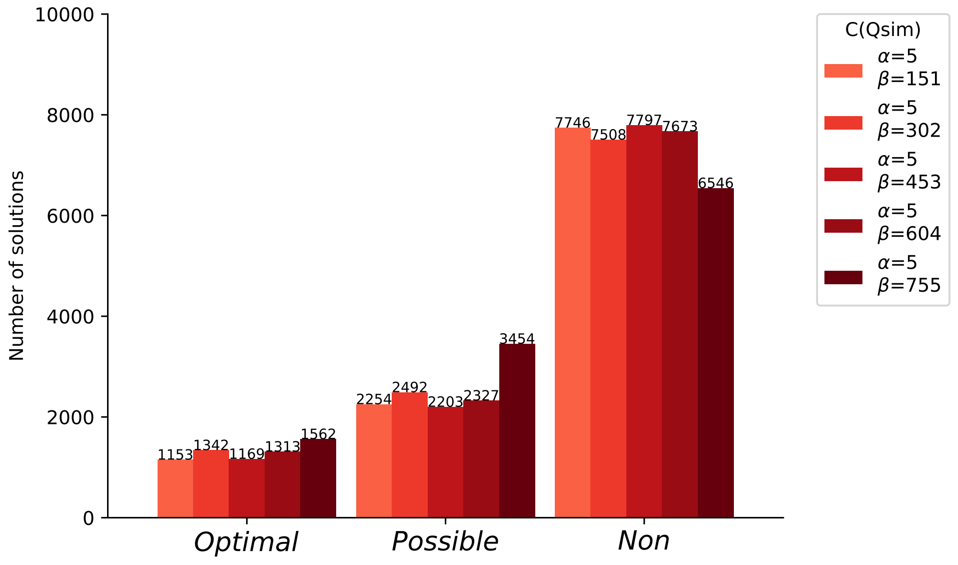


(c) Non solutions

Figure C.6: Energy values of classical `C` and both methods of polynomial reduction (`Q_mq` and `Q_ms`) (varying α)

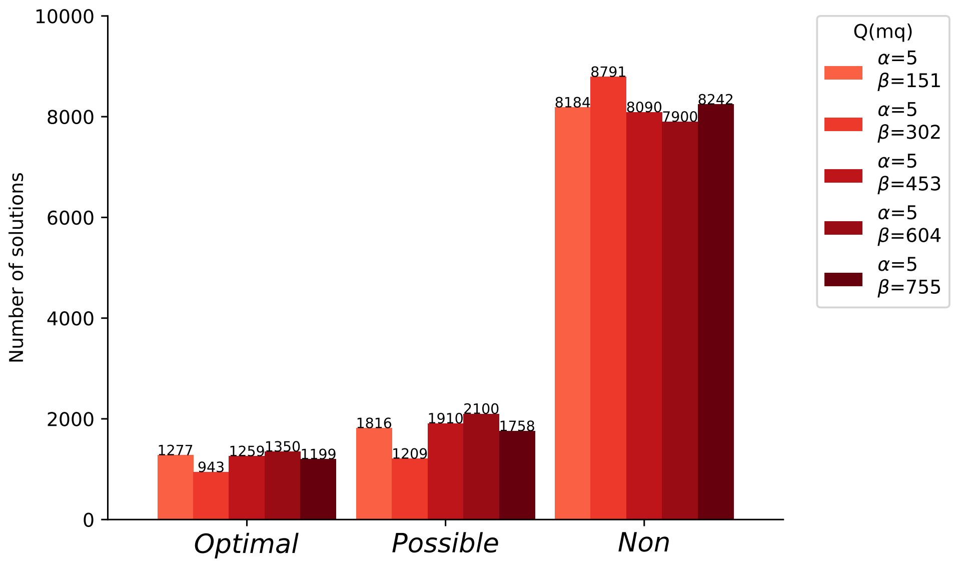


(a) C

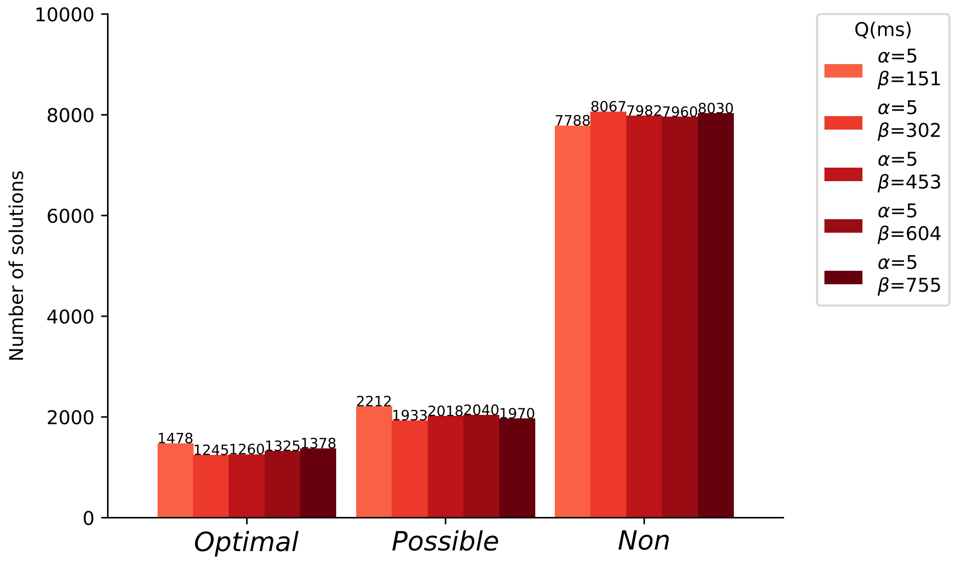


(b) C_Q_sim

Figure C.7: Solutions for pure classical C and classical quantum simulator C_Q_sim and both methods of polynomial reduction Q_mq and Q_ms (varying β) (First part)

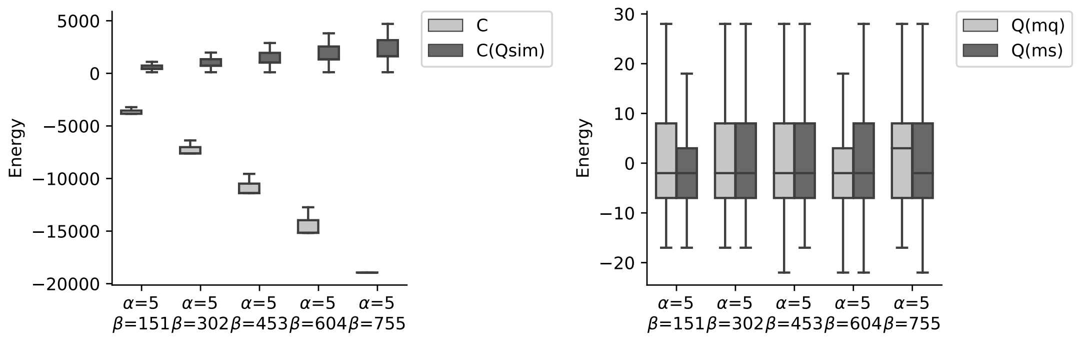


(a) `Q_mq`

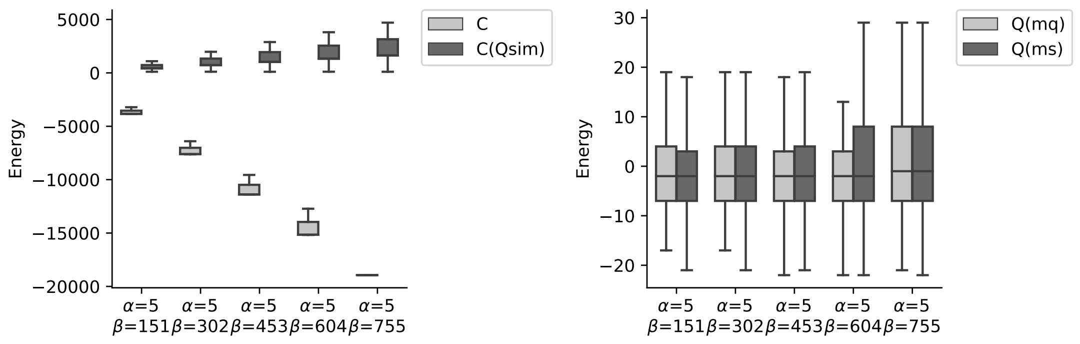


(b) `Q_ms`

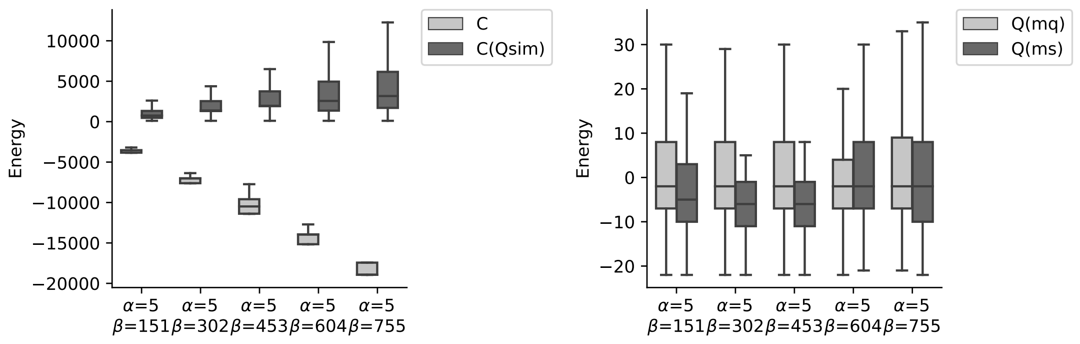
Figure C.8: Solutions for pure classical `C` and classical quantum simulator `C_Q_sim` and both methods of polynomial reduction `Q_mq` and `Q_ms` (varying β) (Second part)



(a) Optimal solutions



(b) Possible solutions



(c) Non solutions

Figure C.9: Energy values of classical `C` and both methods of polynomial reduction (`Q_mq` and `Q_ms`) (varying β)

Appendix D

Algorithms: Deutsch's, Deutsch-Jozsa's, Grover's, Shor's, HHL's

In this section, we present algorithms that provide quantum speed-up, e.g. Deutsch's, Grover's, and Shor's algorithms, which can be realized in today's quantum hardware with limited number of *qubits*.

Deutsch's algorithm.

In 1985, David Deutsch proposed a very simple algorithm: given a circuit (an oracle U_f) that implements a one-bit Boolean function, which aims to determine whether the function is constant (returns the same value c in $\{0,1\}$ on all inputs x) or balanced (returns 1 on one input and 0 on the other). In the classical scenario, we need to consult the oracle twice, to compute both values of the function, resorting to the quantum black-box U_f, we just need one superposition call, since we can take advantage of quantum parallelism together with interference (constructive and destructive). In brief, we were given a Boolean function whose input is 1 bit, $f : \{0,1\} \to \{0,1\}$ and we aim to know if it is constant. If the function is constant, we will measure 0, otherwise, 1.

Deutsch's algorithm is the simpler case of Deutsch-Jozsa algorithm where $f(x)$ takes 1-bit as input.

Deutsch-Jozsa's algorithm.

As in Deutsch's algorithm, based in the same principles, Deutsch-Jozsa's [50], resorting to an oracle U_f, consider a function $f(x)$ that takes as input n-bit strings x and returns 0 or 1. In short, given a Boolean function whose input are n bits, $f : \{0,1\}^n \to \{0,1\}$, we aim to know if it is constant. If the function is constant, we receive $|0\rangle$, otherwise (if the function is balanced), we obtain a string different from $|0\rangle$, Figure D.1.

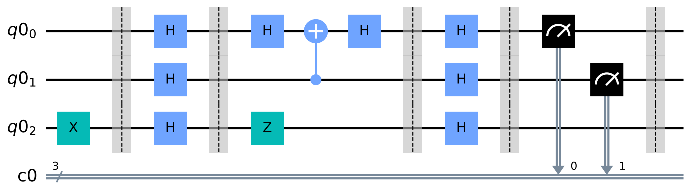


Figure D.1: Deutsch-Jozsa's algorithm circuit

Other interesting algorithms, with related concepts are, the Bernstein-Vazirani and Simon.

Listing D.1: Deutsch-Jozsa's algorithm

```
# Source (broken link):
# https://github.com/Qiskit/ibmqx-user-guides/blob
#/master/rst/full-user-guide/004-Quantum_Algorithms/
#080-Deutsch-Jozsa_Algorithm.rst

from qiskit import IBMQ, BasicAer
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, execute

qr = QuantumRegister(3)  # Initialize qubits
cr = ClassicalRegister(3)  # Initialize bits for record measurements
circuit = QuantumCircuit(qr, cr)

circuit.x(qr[2])  # initialize the ancilla qubit in the |1> state

circuit.barrier()

# First step of quantum algorithms - Prepare the superposition
# For superposition, we apply the Hadamard gate on all qubits
circuit.h(qr[0])
circuit.h(qr[1])
circuit.h(qr[2])

circuit.barrier()

# Oracle function
circuit.h(qr[0])
circuit.cx(qr[1], qr[0])
circuit.z(qr[2])
circuit.h(qr[0])

circuit.barrier()

# Apply Hadamard gates after querying oracle function
circuit.h(qr[0])
circuit.h(qr[1])
circuit.h(qr[2])

circuit.barrier()

# Measure qubit
circuit.measure(qr[0], cr[0])
circuit.measure(qr[1], cr[1])

circuit.barrier()
```

```

# Run our circuit with local simulator
backend = BasicAer.get_backend('qasm_simulator')
shots = 1024
results = execute(circuit, backend=backend, shots=shots).result()
answer = results.get_counts()
print("Simulator result")
for c2c1c0 in answer:
    print(f"{c2c1c0} is observed in {answer[c2c1c0]} times")
# If we measure |0>^n, then f is constant, other results tell us that f is balanced
circuit.draw(output='mpl', filename='circ2.pdf')
```

Grover's algorithm.

Grover's algorithm [75] is used to solve search problems, as in Deutsch-Jozsa's algorithm, U_f computes the function $f : \{0,1\}^n \to \{0,1\}$ where what we want to find is the one that verifies $f(x) = 1$ and is based on the concept of inversion about the mean. Grover's algorithm performs $O(\sqrt{N})$ iterations where each iteration query U_f and performs a call to Grover's diffusion operator which boost the amplitudes of the states that met the condition, Figure D.2.

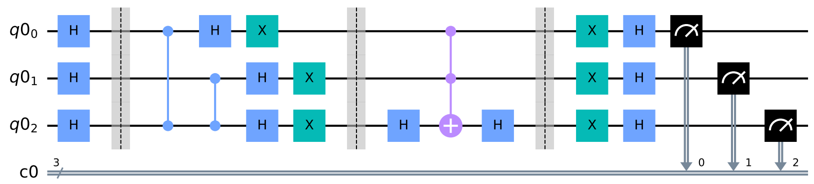


Figure D.2: Grover's algorithm circuit

Resorting to AQUA from Qiskit we use the Grover's search algorithm to solve the problem of determining if there exists a way to satisfy a given Boolean formula which is called the Boolean satisfiability problem (propositional satisfiability problem or SAT). SAT, was the first computer science problem to be proven as nondeterministic polynomial time complete (NP-Complete). In SAT, formulas can be expressed in conjunctive normal form (CNF). The formula is a conjunction (AND) of clauses, where each clause is a disjunction (OR) of literals (variables). The CNF can be in DIMACS format. The file consists in lines (a) starting with "c" which is a comment line, and (b) a line that starts with "p" which identifies the type of problem e.g. "cnf", the number of variables and the number of clauses. Afterwards, we write a clause line which consists of spaced numbers ending with a zero. The Python code below provides a possible solution for $(\neg x1 \vee \neg x2 \vee \neg x3) \wedge (x1 \vee \neg x2 \vee x3) \wedge (x1 \vee x2 \vee \neg x3) \wedge (x1 \vee \neg x2 \vee \neg x3) \wedge (\neg x1 \vee x2 \vee x3)$ which outputs an expression that satisfies the CNF: $(x1 \vee \neg x2 \vee x3)$.

We start by importing the libraries `Aer` allows us to choose the `qasm_simulator` backend to run the script. The `LogicalExpressionOracle` creates an oracle of the string of the desired logical expression and `Grover` links the oracle to the Grover's search algorithm.

Listing D.2: Grover's on Qiskit: import libraries

```
from qiskit import Aer
from qiskit.aqua.components.oracles import LogicalExpressionOracle
from qiskit.aqua.algorithms import Grover
```

A Boolean logical expression in the DIMACS CNF format where 'c' is the comments line and 'p' the line with the definition of the cnf number of variables and number of expressions. In the other lines, are all the expressions.

Listing D.3: Grover's on Qiskit: DIMACS CNF

```
sat_cnf = """
c Example DIMACS 3-sat
p cnf 3 5
-1 -2 -3 0
1 -2 3 0
1 2 -3 0
1 -2 -3 0
-1 2 3 0
"""
```

The `.run` instantiates the Grover algorithm and oracle and runs it on the backend with seed and number of repeats defined by the user.

Listing D.4: Grover's on Qiskit: run the algorithm

```
backend = Aer.get_backend('qasm_simulator')
oracle = LogicalExpressionOracle(sat_cnf)
algorithm = Grover(oracle)
result = algorithm.run(backend, seed_simulator = 123, shots = 10000)
print(result["measurement"])
print(result["result"])
```

Listing D.5: Grover's on Qiskit: output measurement and result

```
Output("measurement"): {'100': 322, '000': 2769, '001': 326, '011': 2793, '111': 302, '
    ↪ 110': 321, '101': 2862, '010': 305}
Output("result"): 1, -2, 3
```

The Oracle will feed the information to the quantum computer which is a quantum black box that can recognize the solutions to our problem. When using Grover's to solve SAT problems, since the algorithm is based upon searching a database for a solution, we can treat all the possible outcomes as a database we intend to search. The true outputs are the solutions, the remaining items are false [59, 60].

Listing D.6: Grover's algorithm

```
# Source: https://www.quantiki.org/wiki/grovers-search-algorithm
from qiskit import IBMQ, BasicAer
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, execute

qr = QuantumRegister(3)  # Initialize qubits
cr = ClassicalRegister(3)  # Initialize bits for record measurements
circuit = QuantumCircuit(qr, cr)

# We want to search two marked states
# |101> and |110>

# Apply Hadamard to all qubits
circuit.h(qr)
circuit.barrier()

# Phase oracle (Marks states |101> and |110> as results)
circuit.cz(qr[2], qr[0])
circuit.cz(qr[2], qr[1])

# Inversion around the average
circuit.h(qr)
circuit.x(qr)
circuit.barrier()
circuit.h(qr[2])
circuit.ccx(qr[0], qr[1], qr[2])
circuit.h(qr[2])
circuit.barrier()
circuit.x(qr)
circuit.h(qr)

# Measure
circuit.measure(qr, cr)

# Run our circuit with local simulator
backend = BasicAer.get_backend('qasm_simulator')
shots = 1024
results = execute(circuit, backend=backend, shots=shots).result()
answer = results.get_counts()
print(answer)
circuit.draw(output='mpl', filename='circ3.pdf')
```

Shor's algorithm.

Peter Shor invented in 1994 the Shor's algorithm [184, 185] which finds a factor of a n-bit integer, performs integer factorization in polynomial time. In brief, it solves the problem of given an integer N, find its prime factors. In this sense, a factorization problem can be reduced to a factorization of N to the problem of finding the period of an integer which depends on $f(x) = a^x$ *mod* N, where a and N are positive integers, $a < N$ with the period (or order, r) such that a^r mod $N = 1$, Figure D.3.

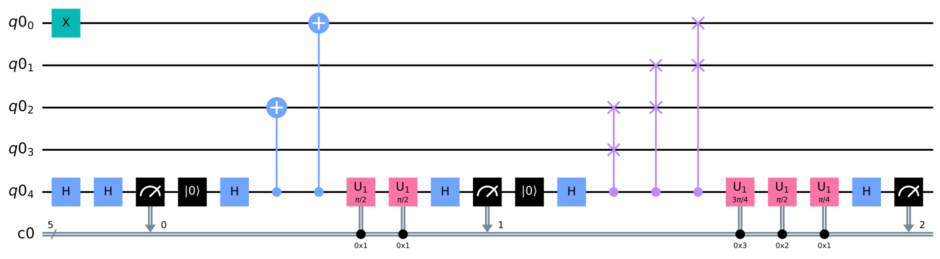


Figure D.3: Shor's algorithm circuit

The Shor's algorithm, where given a large integer N (typically several hundred digits long), factorize N as a product of primes:

a. Pick random number $a < N$.

b. Compute greatest common divisor, $gcd(a, N)$.

c. If $gcd(a, N)! = 1$, then it is a non-trivial factor of N.

d. Otherwise, use the quantum period-finding subroutine to find r, the period of the following function: $f(x) = a^x modN$

e. If r is odd, go back to step 1.

f. If $a^r/2 = -1(modN)$, go back to step 1.

g. $gcd(a^r/2 + 1, N)$ and $gcd(a^r/2 - 1, N)$ are both nontrivial factors of N.

The computer scientists Ethan Bernstein and Umesh Vazirani, in 1993 defined a new complexity class named BQP (bounded-error quantum polynomial time). This class contains all the decision problems - problems with a yes or no answer, suited for quantum computers to solve. To distinguish between two complexity classes is to find a problem that is probably in one and not the other. For example, the paper by Raz and Tal [164] proves that a quantum computer needs less hints than a classical computer to solve the forrelation problem. This problem concerns two sequences of digits from two random numbers generators, asking the machine if there are two sequences completely independent from each other, or are they related some how? Aaronson introduced this forrelation problem in 2009 and proved that it belongs to BQP, and is not in PH, since there's no algorithm in PH that can solve the problem.

Listing D.7: Shor's algorithm

```
# Source: https://github.com/Qiskit/qiskit-community-tutorials/blob/
#b9266a4f9c1f6b3f4cf5117d9c443f9f1c3518cb/algorithms/
#shor_algorithm.ipynb

import math

from qiskit import IBMQ, BasicAer
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, execute

# We'll build circuit for a^x mod 15 for a = 2
qr = QuantumRegister(5)
cr = ClassicalRegister(5)

circuit = QuantumCircuit(qr, cr)

# Initialize q[0] to |1>
circuit.x(qr[0])

# Apply a**4 mod 15
circuit.h(qr[4])
circuit.h(qr[4])
circuit.measure(qr[4], cr[0])
circuit.reset(qr[4])
```

```

# Apply a**2 mod 15
circuit.h(qr[4])
circuit.cx(qr[4], qr[2])
circuit.cx(qr[4], qr[0])
circuit.u1(math.pi/2., qr[4]).c_if(cr, 1)
circuit.u1(math.pi/2., qr[4]).c_if(cr, 1)
circuit.h(qr[4])
circuit.measure(qr[4], cr[1])
circuit.reset(qr[4])

# Apply a mod 15
circuit.h(qr[4])
circuit.cswap(qr[4], qr[3], qr[2])
circuit.cswap(qr[4], qr[2], qr[1])
circuit.cswap(qr[4], qr[1], qr[0])
circuit.u1(3.*math.pi/4., qr[4]).c_if(cr, 3)
circuit.u1(math.pi/2., qr[4]).c_if(cr, 2)
circuit.u1(math.pi/4., qr[4]).c_if(cr, 1)
circuit.h(qr[4])
circuit.measure(qr[4], cr[2])

# Run our circuit with local simulator
backend = BasicAer.get_backend('qasm_simulator')
shots = 1024
results = execute(circuit, backend=backend, shots=shots).result()
answer = results.get_counts()
print(answer)
# We see the measurements yield x = 0, 2, 4 and 6 with equal(ish) probability.
# Using the continued fraction expansion for x/2^3, we note that only x = 2 and
# 6 give the correct period r = 4, and thus the factors p = gcd(a^{r/2}+1,15) = 3
# and q = gcd(a^{r/2}-1,15) = 5.
circuit.draw(output='mpl', filename='circ4.pdf')
```

HHL's algorithm.

Applying quantum phase estimations to solve linear systems of equations was proposed in 2009 by Harrow, Hassidim and Lloyd [78]. The problem can be defined as, given the matrix $A \in \mathbb{C}^{N\times N}$ and the vector $\vec{b} \in \mathbb{C}^N$, we aim to find $\vec{x} \in \mathbb{C}^N$ satisfying $A\vec{x} = \vec{b}$. The algorithm is based in (a) computation of the eigenvalues (quantum phase estimation), (b) inversion of the eigenvalues, and (c) uncomputation of the eigenvalues (inverse of quantum phase estimation), Figure D.4.

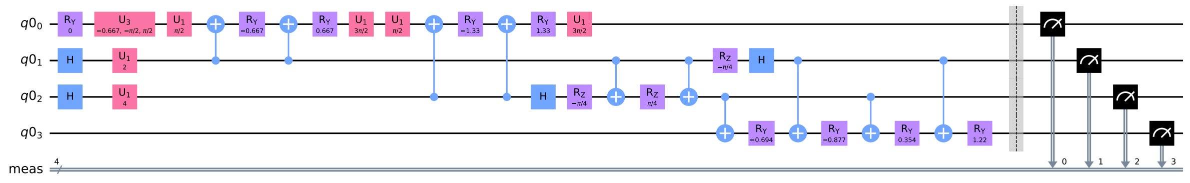


Figure D.4: HHL's algorithm circuit

Listing D.8: HHL's algorithm

```python
# Source: https://qiskit.org/textbook/ch-applications/hhl_tutorial.html

from qiskit import QuantumRegister, QuantumCircuit
import numpy as np

t = 2  # This is not optimal; As an exercise, set this to the
       # value that will get the best results. See section 8 for solution.

nqubits = 4  # Total number of qubits
nb = 1  # Number of qubits representing the solution
nl = 2  # Number of qubits representing the eigenvalues

theta = 0  # Angle defining |b>

a = 1  # Matrix diagonal
b = -1/3  # Matrix off-diagonal

# Initialise the quantum and classical registers
qr = QuantumRegister(nqubits)

# Create a Quantum Circuit
qc = QuantumCircuit(qr)

qrb = qr[0:nb]
qrl = qr[nb:nb+nl]
qra = qr[nb+nl:nb+nl+1]

# State preparation.
qc.ry(2*theta, qrb[0])

# QPE with e^{iAt}
for qu in qrl:
    qc.h(qu)

qc.u1(a*t, qrl[0])
qc.u1(a*t*2, qrl[1])

qc.u3(b*t, -np.pi/2, np.pi/2, qrb[0])


# Controlled e^{iAt} on \lambda_{1}:
params=b*t

qc.u1(np.pi/2,qrb[0])
qc.cx(qrl[0],qrb[0])
qc.ry(params,qrb[0])
qc.cx(qrl[0],qrb[0])
qc.ry(-params,qrb[0])
qc.u1(3*np.pi/2,qrb[0])

# Controlled e^{2iAt} on \lambda_{2}:
params = b*t*2

qc.u1(np.pi/2,qrb[0])
qc.cx(qrl[1],qrb[0])
qc.ry(params,qrb[0])
qc.cx(qrl[1],qrb[0])
qc.ry(-params,qrb[0])
qc.u1(3*np.pi/2,qrb[0])
```

```

# Inverse QFT
qc.h(qrl[1])
qc.rz(-np.pi/4,qrl[1])
qc.cx(qrl[0],qrl[1])
qc.rz(np.pi/4,qrl[1])
qc.cx(qrl[0],qrl[1])
qc.rz(-np.pi/4,qrl[0])
qc.h(qrl[0])

# Eigenvalue rotation
t1=(-np.pi +np.pi/3 - 2*np.arcsin(1/3))/4
t2=(-np.pi -np.pi/3 + 2*np.arcsin(1/3))/4
t3=(np.pi -np.pi/3 - 2*np.arcsin(1/3))/4
t4=(np.pi +np.pi/3 + 2*np.arcsin(1/3))/4

qc.cx(qrl[1],qra[0])
qc.ry(t1,qra[0])
qc.cx(qrl[0],qra[0])
qc.ry(t2,qra[0])
qc.cx(qrl[1],qra[0])
qc.ry(t3,qra[0])
qc.cx(qrl[0],qra[0])
qc.ry(t4,qra[0])
qc.measure_all()

print("Depth: %i" % qc.depth())
print("CNOTS: %i" % qc.count_ops()['cx'])
qc.draw(fold=100, output='mpl', filename='circ5.pdf')
```

Appendix E

Quantum operations

In this section are shown concepts which aid to understand quantum operations-based systems.

Preliminaries. *Given vector spaces V and W over $\mathbb{C}$ with bases $e_1, ..., e_m$ and $f_1, ..., f_n$, the tensor product $V \otimes W$ is another vector space over $\mathbb{C}$ of dimension mn. The space $V \otimes W$ has a bilinear operation $\otimes : V \times W \rightarrow V \otimes W$, and it has basis $e_i \otimes f_j \forall i = 1, ..., m,\ j = 1, ..., n$.*

Relationship with Kronecker product (or outer product). *Given $A \in \mathbb{C}^{m \times n}$, $B \in \mathbb{C}^{p \times q}$, the Kronecker product $A \otimes B$ is the matrix $D \in \mathbb{C}^{mp \times nq}$. Given standard basis over the vector spaces $\mathbb{C}^{m \times n}$ and $\mathbb{C}^{p \otimes q}$, the bilinear operation $\otimes$ of $\mathbb{C}^{m \times n} \otimes \mathbb{C}^{p \times q}$ is the Kronecker product.*

$$D = A \otimes B = \begin{pmatrix} a_{11}B ... a_{1n}B \\ a_{21}B ... a_{2n}B \\ \vdots \\ a_{m1}B ... a_{mn}B \end{pmatrix} \tag{E.1}$$

$$x \otimes y = \begin{pmatrix} 0.30 \\ 0.70 \end{pmatrix} \otimes \begin{pmatrix} 0.10 \\ 0.10 \\ 0.40 \\ 0.40 \end{pmatrix} = \begin{pmatrix} 0.30 \times 0.10 \\ 0.30 \times 0.10 \\ 0.30 \times 0.40 \\ 0.30 \times 0.40 \\ 0.70 \times 0.10 \\ 0.70 \times 0.10 \\ 0.70 \times 0.40 \\ 0.70 \times 0.40 \end{pmatrix} = \begin{pmatrix} 0.03 \\ 0.03 \\ 0.12 \\ 0.12 \\ 0.07 \\ 0.07 \\ 0.28 \\ 0.28 \end{pmatrix} \tag{E.2}$$

Properties of the tensor product. *Let $A, B : \mathbb{C}^{m \times m}$, $C, D \in \mathbb{C}^{n \times n}$ be linear maps on V and W respectively, $u, v \in \mathbb{C}^m$, $w, x \in \mathbb{C}^n$, and $a, b \in \mathbb{C}$. The tensor product satisfies:*

- Bilinearity
  - $(u + v) \otimes w = u \otimes w + v \otimes w$

  - $u \otimes (w + x) = u \otimes w + u \otimes x$
  - $(au) \otimes (bw) = ab(u \otimes w)$

- Associativity
  - $A \otimes (B \otimes C) = (A \otimes B) \otimes C$
  - $x \otimes (u \otimes v) = (x \otimes u) \otimes v$

- Properties for linear maps
  - $(A \otimes C)(B \otimes D) = AB \otimes CD$
  - $(A \otimes C)(u \otimes w) = Au \otimes Cw$

Vector spaces and braket notation. *Given a complex Euclidean space* $\mathbb{S} \equiv \mathbb{C}^n$, $|\psi\rangle \in \mathbb{S}$ *denotes a column vector, and* $\langle\psi| \in \mathbb{S}^*$ *denotes a row vector that is the conjugate transpose of* $|\psi\rangle$*, i.e.,* $\langle\psi| = |\psi\rangle^*$*.* $|\psi\rangle$ *is called a ket,* $\langle\psi|$ *is called a bra.*

We work with complex Euclidean spaces of the form $(\mathbb{C}^2)^{\otimes q}$ and with the standard basis [144].

Structure of the standard basis. *The standard basis for* $\mathbb{C}^2$ *is denoted by:*

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{E.3}$$

.

The four basis elements of $(\mathbb{C}^2)^{\otimes 2} = \mathbb{C}^2 \otimes \mathbb{C}^2$ are:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad |01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{E.4}$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad |11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{E.5}$$

The standard basis for $(\mathbb{C}^2)^{\otimes q}$ is given by the following 2^q vectors:

$$\underbrace{|0\rangle \otimes ... \otimes |0\rangle \otimes |0\rangle}_{\text{q times}} = \underbrace{|00...00\rangle}_{\text{q digits}} \tag{E.6}$$

$$\underbrace{|0\rangle \otimes ... \otimes |0\rangle \otimes |1\rangle}_{\text{q times}} = \underbrace{|00...01\rangle}_{\text{q digits}} \tag{E.7}$$

$$\vdots \tag{E.8}$$

$$\underbrace{|1\rangle \otimes ... \otimes |1\rangle \otimes |1\rangle}_{\text{q times}} = \underbrace{|11...11\rangle}_{\text{q digits}} \tag{E.9}$$

Assumption. *The state of a q-qubit quantum register is a unit vector in* $(\mathbb{C}^2)^{\otimes q} = \mathbb{C}^2 \otimes ... \otimes \mathbb{C}^2$.

State of a quantum register. *A quantum computer has a state stored in a quantum register. Classical registers are made up of bits. Quantum registers are made up of qubits. The state of a q-bit classical register lives in q-dimensional space. The state of a q-qubit quantum register lives in* 2^q*-dimensional complex space.*

The state of a single qubit is:

$$\alpha|0\rangle + \beta|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{E.10}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.
The state of a q-qubit quantum register is:

$$\sum_{\overrightarrow{j} \in \{0,1\}^q} \alpha_{\overrightarrow{j}} |\overrightarrow{j}\rangle \tag{E.11}$$

with $\sum_{\overrightarrow{j} \in \{0,1\}^q} |\alpha_{\overrightarrow{j}}|^2 = 1$

Basis states and superposition. *q qubits are in a basis state if their state* $|\psi\rangle = \sum_{\overrightarrow{j} \in \{0,1\}^q} \alpha_{\overrightarrow{j}} |\overrightarrow{j}\rangle_q$ *is* $|\psi\rangle = |\overrightarrow{k}\rangle$ *for some* $\overrightarrow{k} \in \{0,1\}^q$*. Otherwise, they are in a superposition.*
Example: given general 2-qubit state $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$*, the following is a basis state:* $|\psi\rangle = |01\rangle$ *(i.e.,* $\alpha_{00} = \alpha_{10} = \alpha_{11} = 0, \alpha_{01} = 1$*), and this is a superposition:* $|\psi\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle$*. Thus, superposition = linear combination of basis states.*

Product states and entanglement. *A quantum state* $|\psi\rangle \in (\mathbb{C}^2)^{\otimes q}$ *is a product state if it can be written as* $|\psi_1\rangle \otimes ... \otimes |\psi_q\rangle$ *with 1-qubit states. Otherwise, it is entangled.*
Example: product state. Consider 2-qubit state: $\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$*. This is a product state because it is equal to:* $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$*.*
Example: entangled state. Consider 2-qubit state: $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$*. This is an entangled state, because it cannot be expressed as a product of two 1-qubit states.*

Bibliography

[1] Amira Abbas, Maria Schuld, and Francesco Petruccione. On quantum ensembles of quantum classifiers. *Quantum Machine Intelligence*, 2(1):1–8, Jun 2020.

[2] Héctor Abraham et al. Qiskit: An open-source framework for quantum computing. 2019.

[3] Erdi Acar and Ihsan Yilmaz. Covid-19 detection on ibm quantum computer with classical-quantum transfer learning. *medRxiv*, 2020.

[4] Jeremy Adcock, Euan Allen, Matthew Day, Stefan Frick, Janna Hinchliff, Mack Johnson, Sam Morley-Short, Sam Pallister, Alasdair Price, and Stasja Stanisic. Advances in quantum machine learning. *arXiv e-prints*, 2015.

[5] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM J. Comput.*, 37(1):166–194, April 2007.

[6] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Machine learning in a quantum world. In Luc Lamontagne and Mario Marchand, editors, *Advances in Artificial Intelligence*, pages 431–442, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[7] Tameem Albash and Daniel A. Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1), Jan 2018.

[8] Tameem Albash and Daniel A. Lidar. *Demonstration of a Scaling Advantage for a Quantum Annealer over Simulated Annealing*, volume 8. American Physical Society, Jul 2018.

[9] Carmen G. Almudever, Lingling Lao, Robert Wille, and Gian G. Guerreschi. Realizing quantum algorithms on real quantum computing devices. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 864–872. IEEE, 2020.

[10] Md Zahangir Alom, Brian Van Essen, Adam T. Moody, David Peter Widemann, and Tarek M. Taha. Quadratic unconstrained binary optimization (qubo) on neuromorphic computing system. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3922–3929, May 2017.

[11] Bruno Apolloni, Nicolò Cesa-Bianchi, and Diego De Falco. A numerical implementation of quantum annealing. In *Stochastic processes, physics and geometry (Ascona and Locarno, 1988)*, pages 97–111, Teaneck, NJ, 1990. World Sci. Publ.

[12] Frank Arute et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, Oct 2019.

[13] Ramin Ayanzadeh, Milton Halem, and Tim Finin. Reinforcement quantum annealing: A hybrid quantum learning automata. *Scientific Reports*, 10(1):7952, May 2020.

[14] Charles H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973.

[15] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Száva, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv e-prints*, 2020.

[16] David E. Bernal, Kyle E. C. Booth, Raouf Dridi, Hedayat Alghassi, Sridhar Tayur, and Davide Venturelli. Integer programming techniques for minor-embedding in quantum annealers. In Emmanuel Hebrard and Nysret Musliu, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 112–129, Cham, 2020. Springer International Publishing. ISBN: 978-3-030-58942-4.

[17] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S. Kottmann, Tim Menke, Wai-Keong Mok, Sukin Sim, Leong-Chuan Kwek, and Alán Aspuru-Guzik. Noisy intermediate-scale quantum (nisq) algorithms, 2021.

[18] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, Sep 2017.

[19] Zhengbing Bian, Fabian Chudak, Robert Brian Israel, Brad Lackey, William G. Macready, and Aidan Roy. Mapping constrained optimization problems to quantum annealing with application to fault diagnosis. *Frontiers in ICT*, 3:14, 2016.

[20] Zhengbing Bian, Fabian Chudak, William Macready, Aidan Roy, Roberto Sebastiani, and Stefano Varotti. Solving sat and maxsat with a quantum annealer: Foundations and a preliminary report. In *International Symposium on Frontiers of Combining Systems*, pages 153–171. Springer, 2017.

[21] Zhengbing Bian, Fabian Chudak, William Macready, Aidan Roy, Roberto Sebastiani, and Stefano Varotti. Solving sat (and maxsat) with a quantum annealer: Foundations, encodings, and preliminary results. *Information and computation*, 275:104609, 2020.

[22] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN: 0387310738.

[23] bmq_manhattan v1.0.2, IBM Quantum team. Retrieved from https://quantum-computing.ibm.com, 2020.

[24] Sergio Boixo, Troels F. Rønnow, Sergei V. Isakov, Zhihui Wang, David Wecker, Daniel A. Lidar, John M. Martinis, and Matthias Troyer. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics*, 10(3):218–224, Feb 2014.

[25] Kyle E. C. Booth, Bryan O'Gorman, Jeffrey Marshall, Stuart Hadfield, and Eleanor Rieffel. Quantum-accelerated global constraint filtering. In Helmut Simonis, editor, *Principles and Practice of Constraint Programming*, pages 72–89, Cham, 2020. Springer International Publishing. ISBN: 978-3-030-58475-7.

[26] Tomas Boothby, Andrew D. King, and Aidan Roy. Fast clique minor generation in chimera qubit connectivity graphs. *Quantum Information Processing*, 15(1):495–508, Jan 2016.

[27] Max Born and Vladimir Fock. Beweis des Adiabatensatzes. *Zeitschrift fur Physik*, 51(3-4): 165–180, March 1928.

[28] Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1):155 – 225, 2002.

[29] Michał Borowski, Paweł Gora, Katarzyna Karnas, Mateusz Błajda, Krystian Król, Artur Matyjasek, Damian Burczyk, Miron Szewczyk, and Michał Kutwin. New hybrid quantum annealing algorithms for solving vehicle routing problem. In Valeria V. Krzhizhanovskaya, Gábor Závodszky, Michael H. Lees, Jack J. Dongarra, Peter M. A. Sloot, Sérgio Brissos, and João Teixeira, editors, *Computational Science – ICCS 2020*, pages 546–561, Cham, 2020. Springer International Publishing. ISBN: 978-3-030-50433-5.

[30] Jun Cai, William G. Macready, and Aidan Roy. A practical heuristic for finding graph minors. *arXiv e-prints*, 2014.

[31] Yudong Cao, Gian Giacomo Guerreschi, and Alán Aspuru-Guzik. Quantum neuron: an elementary building block for machine learning on quantum computers. *arXiv e-prints*, 2017.

[32] Jaime S Cardoso, Nuno Marques, Neeraj Dhungel, Gustavo Carneiro, and Andrew P Bradley. Mass segmentation in mammograms: A cross-sensor comparison of deep and tailored features. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1737–1741. IEEE, 2017.

[33] Ginés Carrascal, Alberto A. del Barrio, and Guillermo Botella. First experiences of teaching quantum computing. *The Journal of Supercomputing*, 77(3):2770–2799, Mar 2021.

[34] M. Cerezo, A. Arrasmith, and R. et al. Babbush. Variational quantum algorithms. *Nat Rev Phys*, 3:625–644, 2021.

[35] Rui Chao and Ben W. Reichardt. Fault-tolerant quantum computation with few qubits. *npj Quantum Information*, 4(1):42, 2018.

[36] Tobias Chasseur, Stefan Kehrein, and Frank K. Wilhelm. Environmental effects in quantum annealing. *arXiv e-prints*, 2018.

[37] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847. IEEE, 2018.

[38] Sihong Chen, Kai Ma, and Yefeng Zheng. Med3d: Transfer learning for 3d medical image analysis. *arXiv e-prints*, 2019.

[39] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society of London Series A*, 474(2209):20170551, Jan 2018.

[40] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[41] Andrew W. Cross, Ali Javadi-Abhari, Thomas Alexander, Niel de Beaudrap, Lev S. Bishop, Steven Heidel, Colm A. Ryan, John Smolin, Jay M. Gambetta, and Blake R. Johnson. Openqasm 3: A broader and deeper quantum assembly language. *arXiv e-prints*, 2021.

[42] Songsong Dai. A note on implication operators of quantum logic. *Quantum Machine Intelligence*, 2(2):15, Nov 2020.

[43] Prasanna Date, Robert Patton, Catherine Schuman, and Thomas Potok. Efficiently embedding qubo problems on adiabatic quantum computers. *Quantum Information Processing*, 18(4):117, 2019.

[44] Prasanna Date, Davis Arthur, and Lauren Pusey-Nazzaro. Qubo formulations for training machine learning models. *Scientific Reports*, 11(1):10029, May 2021.

[45] Nike Dattani and Nick Chancellor. Embedding quadratization gadgets on chimera and pegasus graphs. *arXiv e-prints*, 2019.

[46] Diego de Falco and Dario Tamascelli. An introduction to quantum annealing. *RAIRO - Theoretical Informatics and Applications*, 45(1):99–116, Jan 2011.

[47] Jonathan de Matos, Alceu de S Britto, Luiz ES Oliveira, and Alessandro L Koerich. Double transfer learning for breast cancer histopathologic image classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[48] Taye Girma Debelee, Friedhelm Schwenker, Achim Ibenthal, and Dereje Yohannes. Survey of deep learning in breast cancer image analysis. *Evolving Systems*, 11(1):143–163, 2020.

[49] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[50] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.

[51] Joana Diz, Goreti Marreiros, and Alberto Freitas. Applying data mining techniques to improve breast cancer diagnosis. *Journal of medical systems*, 40(9):203, 2016.

[52] Hristo N. Djidjev, Guillaume Chapuis, Georg Hahn, and Guillaume Rizk. Efficient combinatorial optimization using quantum annealing. *arXiv e-prints*, 2018.

[53] Bojia Duan, Jiabin Yuan, Juan Xu, and Dan Li. Quantum algorithm and quantum circuit for a-optimal projection: Dimensionality reduction. *Physical Review A*, 99(3):032311, 2019.

[54] Vedran Dunjko and Hans J. Briegel. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics*, 81(7):074001, 2018.

[55] Vedran Dunjko, Jacob M. Taylor, and Hans J. Briegel. Quantum-enhanced machine learning. *Phys. Rev. Lett.*, 117:130501, Sep 2016.

[56] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv e-prints*, 2018.

[57] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum Computation by Adiabatic Evolution. *arXiv e-prints*, 2000.

[58] Sebastian Feld, Christoph Roch, Thomas Gabor, Christian Seidel, Florian Neukart, Isabella Galter, Wolfgang Mauerer, and Claudia Linnhoff-Popien. A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer. *Frontiers in ICT*, 6:13, 2019.

[59] Diogo Fernandes, Carla Silva, and Inês Dutra. Using grover's search quantum algorithm to solve boolean satisfiability problems, part 2. *XRDS*, 26(2):68–71, November 2019.

[60] Diogo Fernandes, Carla Silva, and Inês Dutra. Erratum: Using grover's search quantum algorithm to solve boolean satisfiability problems, part 2. *XRDS*, 26(3):57, April 2020.

[61] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, Jun 1982.

[62] Mark Fingerhuth, Tomáš Babej, and Peter Wittek. Open source software in quantum computing. *PloS one*, 13(12):e0208561, 2018.

[63] JP Pereira Fontes, MA Guevara Lopez, et al. Representation learning approach to breast cancer diagnosis. European Congress of Radiology-ECR 2019, 2019.

[64] Michael P. Frank. Throwing computing into reverse. *IEEE Spectrum*, 54(9):32–37, 2017.

[65] Keisuke Fujii. Quantum speedup in stoquastic adiabatic quantum computation. *arXiv e-prints*, 2018.

[66] World Cancer Research Fund. Breast cancer: How diet, nutrition and physical activity affect breast cancer risk, aug 2020.

[67] Sara Gamble. Quantum computing: What it is, why we want it, and how we're trying to get it. In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2018 Symposium*. National Academies Press, 2019.

[68] Wenshuo Gao, Xiaoguang Zhang, Lei Yang, and Huizhong Liu. An improved sobel edge detection. In *2010 3rd International conference on computer science and information technology*, volume 5, pages 67–71. IEEE, 2010.

[69] Ian P. Gent and Toby Walsh. The search for satisfaction. Technical report, 1999.

[70] Austin Gilliam, Charlene Venci, Sreraman Muralidharan, Vitaliy Dorum, Eric May, Rajesh Narasimhan, and Constantin Gonciulea. Foundational patterns for efficient quantum computing. *arXiv e-prints*, 2019.

[71] Fred Glover, Gary Kochenberger, and Yu Du. Quantum bridge analytics i: a tutorial on formulating and using qubo models. *4OR*, 17(4):335–371, 2019.

[72] Angelina Gokhale, Mandaar B Pande, and Dhanya Pramod. Implementation of a quantum transfer learning approach to image splicing detection. *International Journal of Quantum Information*, 18(05):2050024, 2020.

[73] Timothy D. Goodrich, Blair D. Sullivan, and Travis S. Humble. Optimizing adiabatic quantum program compilation using a graph-theoretic framework. *Quantum Information Processing*, 17(5):118, Apr 2018.

[74] Edward Grant, Marcello Benedetti, Shuxiang Cao, Andrew Hallam, Joshua Lockhart, Vid Stojevic, Andrew G Green, and Simone Severini. Hierarchical quantum classifiers. *npj Quantum Information*, 4(1):1–8, 2018.

[75] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. ISBN: 0897917855.

[76] M. A. Guevara. Breast cancer digital repository, aug 2020.

[77] Pierre Hansen and Brigitte Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303, Dec 1990.

[78] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.

[79] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning.* Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001. ISBN: 0387952845.

[80] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, Mar 2019.

[81] Bettina Heim, Mathias Soeken, Sarah Marshall, Chris Granade, Martin Roetteler, Alan Geller, Matthias Troyer, and Krysta Svore. Quantum programming languages. *Nature Reviews Physics*, 2(12):709–722, Dec 2020.

[82] Itay Hen and Federico M. Spedalieri. Quantum annealing for constrained optimization. *Phys. Rev. Applied*, 5:034007, Mar 2016.

[83] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence*, 2(1):1–9, 2020.

[84] Pınar Uskaner Hepsağ, Selma Ayşe Özel, and Adnan Yazıcı. Using deep learning for mammography classification. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 418–423. IEEE, 2017.

[85] John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[86] John J. Hopfield. Artificial neural networks. *IEEE Circuits and Devices Magazine*, 4(5): 3–10, 1988.

[87] John J. Hopfield and David W. Tank. ”neural” computation of decisions in optimization problems. *Biological Cybernetics*, 52(3):141–152, Jul 1985.

[88] Feng Hu, Lucas Lamata, Mikel Sanz, Xi Chen, Xingyuan Chen, Chao Wang, and Enrique Solano. Quantum computing cryptography: Finding cryptographic boolean functions with quantum annealing by a 2000 qubit d-wave quantum computer. *Physics Letters A*, 384 (10):126214, 2020.

[89] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[90] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R. McClean. Power of data in quantum machine learning. *Nature Communications*, 12(1), May 2021.

[91] Hasham Hussain, Muhammad Bin Javaid, Faisal Shah Khan, Archismita Dalal, and Aeysha Khalique. Optimal control of traffic signals using quantum annealing. *Quantum Information Processing*, 19(9):312, Aug 2020.

[92] Benjamin Q Huynh, Hui Li, and Maryellen L Giger. Digital mammographic tumor classification using transfer learning from deep convolutional neural networks. *Journal of Medical Imaging*, 3(3):034501, 2016.

[93] Kazuki Ikeda, Yuma Nakamura, and Travis S. Humble. Application of quantum annealing to nurse scheduling problem. *Scientific Reports*, 9(1):12837, 2019.

[94] D-Wave Systems Inc. D-wave, 2020.

[95] D-Wave Systems Inc. D-wave system documentation, 2020.

[96] D-Wave Systems Inc. Leap, 2020.

[97] Hirotaka Irie, Goragot Wongpaisarnsin, Masayoshi Terabe, Akira Miki, and Shinichirou Taguchi. Quantum annealing of vehicle routing problem with time, state and capacity. In Sebastian Feld and Claudia Linnhoff-Popien, editors, *Quantum Technology and Optimization Problems*, pages 145–156, Cham, 2019. Springer International Publishing. ISBN: 978-3-030-14082-3.

[98] Konrad Jalowiecki, Andrzej Wieckowski, Piotr Gawron, and Bartlomiej Gardas. Parallel in time dynamics with quantum annealers. *Scientific Reports*, 10(1):13534, Aug 2020.

[99] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation. part i, graph partitioning. *Oper. Res.*, 37(6):865–892, October 1989.

[100] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Phys. Rev. E*, 58:5355–5363, Nov 1998.

[101] Sara Hosseinzadeh Kassani, Peyman Hosseinzadeh Kassani, Michal J Wesolowski, Kevin A Schneider, and Ralph Deters. Breast cancer diagnosis with transfer learning and global pooling. *arXiv e-prints*, 2019.

[102] Helmut G. Katzgraber, Firas Hamze, Zheng Zhu, Andrew J. Ochoa, and H. Munoz-Bauza. Seeking quantum speedup through spin glasses: The good, the bad, and the ugly. *Phys. Rev. X*, 5:031026, Sep 2015.

[103] Rupinderdeep Kaur, RK Sharma, and Parteek Kumar. An efficient speaker recognition using quantum neural network. *Modern Physics Letters B*, 32(31):1850384, 2018.

[104] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. *arXiv e-prints*, 2019.

[105] Nathan Killoran, Thomas R. Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd. Continuous-variable quantum neural networks. *arXiv e-prints*, 2018.

[106] Nathan Killoran, Josh Izaac, Nicolás Quesada, Ville Bergholm, Matthew Amy, and Christian Weedbrook. Strawberry Fields: A Software Platform for Photonic Quantum Computing. *Quantum*, 3:129, March 2019.

[107] Hak Gu Kim, Yeoreum Choi, and Yong Man Ro. Modality-bridge transfer learning for medical image classification. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5. IEEE, 2017.

[108] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[109] C Rama Krishna, Maitreyee Dutta, and Rakesh Kumar. *Proceedings of 2nd International Conference on Communication, Computing and Networking: ICCCN 2018, NITTTR Chandigarh, India*, volume 46. Springer, 2018.

[110] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[111] Kazue Kudo. Constrained quantum annealing of graph coloring. *Physical Review A*, 98(2), Aug 2018.

[112] Thaddeus D Ladd, Fedor Jelezko, Raymond Laflamme, Yasunobu Nakamura, Christopher Monroe, and Jeremy Lloyd O'Brien. Quantum computers. *Nature*, 464(7285):45–53, 2010.

[113] Trevor Lanting, Anthony J Przybysz, A Yu Smirnov, Federico M Spedalieri, Mohammad H Amin, Andrew J Berkley, Richard Harris, Fabio Altomare, Sergio Boixo, Paul Bunyk, et al. Entanglement in a Quantum Annealing Processor. *Physical Review X*, 4(2):021041, April 2014.

[114] Ryan LaRose. Overview and Comparison of Gate Level Quantum Software Platforms. *Quantum*, 3:130, March 2019.

[115] Ryan LaRose and Brian Coyle. Robust data encodings for quantum classifiers. *Physical Review A*, 102(3), Sep 2020.

[116] Daniel Lévy and Arzav Jain. Breast mass classification from mammograms using deep convolutional neural networks. *arXiv e-prints*, 2016.

[117] Mark Lewis and Fred Glover. Quadratic unconstrained binary optimization problem preprocessing: Theory and empirical analysis. *Networks*, 70(2):79–97, 2017.

[118] Fangyi Li, Changjing Shang, Ying Li, and Qiang Shen. Interpretable mammographic mass classification with fuzzy interpolative reasoning. *Knowledge-Based Systems*, 191:105279, 2020.

[119] Richard Y. Li, Rosa Di Felice, Remo Rohs, and Daniel A. Lidar. Quantum annealing versus classical machine learning applied to a simplified computational biology problem. *npj Quantum Information*, 4(1):14, Feb 2018.

[120] Priscila M. V. Lima. Q-satyrus: Mapping neuro-symbolic reasoning into an adiabatic quantum computer. In *Proceedings of the Twelfth International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017*, 2017.

[121] Priscila M. V. Lima, M. Mariela M. Morveli-Espinoza, and Felipe M. G. França. Logic as energy: A sat-based approach. In Francesco Mele, Giuliana Ramella, Silvia Santillo, and Francesco Ventriglia, editors, *Advances in Brain, Vision, and Artificial Intelligence*, pages 458–467, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN: 978-3-540-75555-5.

[122] Priscila Machado Vieira Lima, M. Mariela Morveli-Espinoza, Glaucia C. Pereira, and Felipe M.G. França. Satyrus: a sat-based neuro-symbolic architecture for constraint processing. In *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, pages 6–pp, Nov 2005.

[123] Priscila M.V. Lima, Glaucia C. Pereira, M. Mariela M. Morveli-Espinoza, and Felipe M.G. França. Mapping and combining combinatorial problems into energy landscapes via pseudo-boolean constraints. In *Brain, Vision, and Artificial Intelligence*, pages 308–317, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN: 978-3-540-32029-6.

[124] Yunbi Liu, Xiao Zhang, Guangwei Cai, Yingyin Chen, Zhaoqiang Yun, Qianjin Feng, and Wei Yang. Automatic delineation of ribs and clavicles in chest radiographs using fully convolutional densenets. *Computer methods and programs in biomedicine*, 180:105014, 2019.

[125] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, pages 1–5, 2021.

[126] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv e-prints*, 2013.

[127] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning. *arXiv e-prints*, 2020.

[128] Miguel G. Lopez, Naimy Posada, Daniel C. Moura, Raúl Ramos Pollán, José M. Franco Valiente, César Suárez Ortega, Manuel Solar, Guillermo Diaz-Herrero, Isabel M.A. Pereira Ramos, Joana Loureiro, et al. BCDR: a breast cancer digital repository. In *15th International conference on experimental mechanics*, volume 1215, 2012.

[129] Siyuan Lu, Zhihai Lu, and Yu-Dong Zhang. Pathological brain detection based on alexnet and transfer learning. *Journal of computational science*, 30:41–47, 2019.

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

[131] Jacek Mańdziuk. Solving the travelling salesman problem with a hopfield-type neural network. *Demonstratio Mathematica*, 29(1):219–232, 1996.

[132] S. Mangini, F. Tacchino, D. Gerace, D. Bajoni, and C. Macchiavello. Quantum computing models for artificial neural networks. *IOPScience*, 134(1):10002, apr 2021.

[133] Andrea Mari, Thomas R Bromley, Josh Izaac, Maria Schuld, and Nathan Killoran. Transfer learning in hybrid classical-quantum neural networks. *arXiv e-prints*, 2019.

[134] Roman Martoňák, Giuseppe E. Santoro, and Erio Tosatti. Quantum annealing of the traveling-salesman problem. *Physical Review E*, 70(5), Nov 2004.

[135] Catherine C. McGeoch. Theory versus practice in annealing-based quantum computing. *Theoretical Computer Science*, 816:169 – 183, 2020.

[136] Rajesh Mehra et al. Breast cancer histology images classification: Training from scratch or transfer learning? *ICT Express*, 4(4):247–254, 2018.

[137] Riccardo Mengoni and Alessandra Di Pierro. Kernel methods in quantum machine learning. *Quantum Machine Intelligence*, 1(3):65–71, Dec 2019.

[138] Kristel Michielsen, Madita Nocon, Dennis Willsch, Fengping Jin, Thomas Lippert, and Hans De Raedt. Benchmarking gate-based quantum computers. *Computer Physics Communications*, 220:44 – 55, 2017.

[139] Anurag Mishra, Tameem Albash, and Daniel A. Lidar. Finite temperature quantum annealing solving exponentially small gap problem with non-monotonic success probability. *Nature Communications*, 9(1):2917, Jul 2018.

[140] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN: 9780070428072.

[141] Nikolaj Moll, Panagiotis Barkoutsos, Lev S. Bishop, Jerry M. Chow, Andrew Cross, Daniel J. Egger, Stefan Filipp, Andreas Fuhrer, Jay M. Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Müller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3:030503, Jul 2018.

[142] Bruno França Monteiro. Satyrus2: Compilando especificações de raciocínio lógico. Master's thesis, Universidade Federal do Rio de Janeiro (UFRJ) / Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia (COPPE), Cidade Universitária, Centro de Tecnologia, Bloco H, Sala 319 Post office box: 68511 Zip code: 21941-972 Rio de Janeiro - RJ - Brasil Avenida Athos da Silveira Ramos, 3 2010.

[143] Mohammad Amin Morid, Alireza Borjali, and Guilherme Del Fiol. A scoping review of transfer learning research on medical image analysis using imagenet. *arXiv e-prints*, 2020.

[144] Giacomo Nannicini. An introduction to quantum computing, without the physics. *arXiv e-prints*, 2017.

[145] Christian FA Negre, Hayato Ushijima-Mwesigwa, and Susan M Mniszewski. Detecting multiple communities using quantum annealing on the d-wave system. *Plos one*, 15(2): e0227538, 2020.

[146] Florian Neukart, Gabriele Compostella, Christian Seidel, David von Dollen, Sheir Yarkoni, and Bob Parney. Traffic flow optimization using a quantum annealer. *Frontiers in ICT*, 4: 29, 2017.

[147] Michael A. Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2010.

[148] Peter Nimbe, Benjamin Asubam Weyori, and Adebayo Felix Adekoya. Models in quantum computing: a systematic review. *Quantum Information Processing*, 20(2):80, Feb 2021.

[149] Fabrizio Nunnari, Chirag Bhuvaneshwara, Abraham Obinwanne Ezema, and Daniel Sonntag. A study on the fusion of pixels and patient metadata in cnn-based classification of skin lesion images. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 191–208. Springer, 2020.

[150] Masayuki Ohzeki. Breaking limitation of quantum annealer in solving optimization problems under constraints. *Scientific Reports*, 10(1):3126, Feb 2020.

[151] Shuntaro Okada, Masayuki Ohzeki, Masayoshi Terabe, and Shinichiro Taguchi. Improving solutions by embedding larger subproblems in a d-wave quantum annealer. *Scientific Reports*, 9(1):2098, 2019.

[152] Shuntaro Okada, Masayuki Ohzeki, Masayoshi Terabe, and Shinichiro Taguchi. Improving solutions by embedding larger subproblems in a d-wave quantum annealer. *Scientific reports*, 9(1):1–10, 2019.

[153] Emilio Soria Olivas. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques*. IGI Global, 2009.

[154] Emmanuel Lawrence Omonigho, Micheal David, Achonu Adejo, and Saliyu Aliyu. Breast cancer: Tumor detection in mammogram images using modified alexnet deep convolution neural network. In *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, pages 1–6. IEEE, 2020.

[155] Carlos Outeiral, Martin Strahm, Jiye Shi, Garrett M. Morris, Simon C. Benjamin, and Charlotte M. Deane. The prospects of quantum computing in computational molecular biology. *WIREs Computational Molecular Science*, 11(1), May 2020.

[156] Scott Pakin. Performing fully parallel constraint logic programming on a quantum annealer. *Theory and Practice of Logic Programming*, 18(5-6):928–949, 2018.

[157] Gintaras Palubeckis. Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. *Annals of Operations Research*, 131(1):259–282, Oct 2004.

[158] Davide Pastorello and Enrico Blanzieri. Quantum annealing learning search for solving qubo problems. *Quantum Information Processing*, 18(10):303, Aug 2019.

[159] Adam Pearson, Anurag Mishra, Itay Hen, and Daniel A Lidar. Analog errors in quantum annealing: doom and hope. *NPJ Quantum Information*, 5:1–9, 2019.

[160] Jean-Yves Potvin. State-of-the-art survey—the traveling salesman problem: A neural network perspective. *ORSA Journal on Computing*, 5(4):328–348, 1993.

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

[162] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766, 2007.

[163] Jack Raymond, Ndiame Ndiaye, Gautam Rayaprolu, and Andrew D. King. Improving performance of logical qubits by parameter tuning and topology compensation. *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Oct 2020.

[164] Ran Raz and Avishay Tal. Oracle separation of BQP and PH. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:107, 2018.

[165] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Phys. Rev. Lett.*, 113:130503, Sep 2014.

[166] Ben W. Reichardt. The quantum adiabatic optimization algorithm and local minima. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, page 502–510, New York, NY, USA, 2004. Association for Computing Machinery. ISBN: 1581138520.

[167] Dezső Ribli, Anna Horváth, Zsuzsa Unger, Péter Pollner, and István Csabai. Detecting and classifying lesions in mammograms with deep learning. *Scientific reports*, 8(1):1–7, 2018.

[168] Eleanor G. Rieffel, Davide Venturelli, Bryan O'Gorman, Minh B. Do, Elicia M. Prystay, and Vadim N. Smelyanskiy. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing*, 14(1):1–36, Jan 2015.

[169] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010. ISBN: 978-0-13-207148-2.

[170] Mehdi Saeedi and Igor L. Markov. Synthesis and optimization of reversible circuits—a survey. *ACM Computing Surveys*, 45(2):1–34, Feb 2013.

[171] Jianzhi Sang, Shen Wang, and Qiong Li. A novel quantum representation of color digital images. *Quantum Information Processing*, 16(2):42, 2017.

[172] Maria Schuld. Quantum machine learning models are kernel methods. *arXiv e-prints*, 2021.

[173] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Phys. Rev. Lett.*, 122:040504, Feb 2019.

[174] Maria Schuld and Francesco Petruccione. Quantum machine learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 1034–1043. Springer US, Boston, MA, 2017. ISBN: 978-1-4899-7687-1.

[175] Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*. Quantum Science and Technology. Springer International Publishing, 2018. ISBN: 9783319964232.

[176] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[177] Maria Schuld, Mark Fingerhuth, and Francesco Petruccione. Implementing a distance-based classifier with a quantum interference circuit. *EPL (Europhysics Letters)*, 119(6):60002, Sep 2017.

[178] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *arXiv e-prints*, 2018.

[179] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys. Rev. A*, 103: 032430, Mar 2021.

[180] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[181] Li Shen. End-to-end training for whole image breast cancer diagnosis using an all convolutional design. *arXiv e-prints*, 2017.

[182] Yunong Shi, Nelson Leung, Pranav Gokhale, Zane Rossi, David I. Schuster, Henry Hoffmann, and Frederic T. Chong. Optimized compilation of aggregated instructions for realistic quantum computers. *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, Apr 2019.

[183] Chuen-Kai Shie, Chung-Hisang Chuang, Chun-Nan Chou, Meng-Hsi Wu, and Edward Y Chang. Transfer representation learning for medical image analysis. In *2015 37th annual international conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 711–714. IEEE, 2015.

[184] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[185] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997.

[186] Carla Silva. What is quantum ai? *ITNOW*, 59(4):21, dec 2017.

[187] Carla Silva and Inês Dutra. Code gc [available.], 2020.

[188] Carla Silva, Marcus Dahlem, and Inês Dutra. Simulating markov transition probabilities in a quantum environment. *Poster presented at 3rd International Conference for Young Quantum Information Scientists 3-6 October 2017, Max Planck Institute for the Science of Light, Friedrich-Alexander Universitat Erlangen-Nurnberg*, 2017.

[189] Carla Silva, Marcus Dahlem, and Inês Dutra. Driven tabu search: A quantum inherent optimisation. *1st Workshop on Quantum Machine Learning of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, 2018.

[190] Carla Silva, Inês Dutra, and Marcus Dahlem. Driven tabu search: a quantum inherent optimisation. *arXiv e-prints*, 2018.

[191] Carla Silva, Ana Aguiar, Priscila M. V. Lima, and Inês Dutra. Mapping graph coloring to quantum annealing. *Quantum Machine Intelligence*, 2(2):16, Nov 2020.

[192] Carla Silva, Ana Aguiar, and Inês Dutra. Quantum binary classification (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(18):15889–15890, May 2021.

[193] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv e-prints*, 2014.

[194] Andrea Skolik, Jarrod R McClean, Masoud Mohseni, Patrick van der Smagt, and Martin Leib. Layerwise learning for quantum neural networks. *arXiv e-prints*, 2020.

[195] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[196] Karthik Srinivasan, Saipriya Satyajit, Bikash K. Behera, and Prasanta K. Panigrahi. Efficient quantum algorithm for solving travelling salesman problem: An ibm quantum experience. *arXiv e-prints*, 2018.

[197] Michael Streif, Florian Neukart, and Martin Leib. Solving quantum chemistry problems with a d-wave quantum annealer. In *International Workshop on Quantum Technology and Optimization Problems*, pages 111–122. Springer, 2019.

[198] Bolesław Szafnicki. A unified approach for degree reduction of polynomials in the bernstein basis part i: Real polynomials. *Journal of Computational and Applied Mathematics*, 142 (2):287 – 312, 2002.

[199] Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1):26, 2019.

[200] Kay C. Tan, Huajin Tang, and Shuzhi S. Ge. On parameter settings of hopfield networks applied to traveling salesman problems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(5):994–1002, 2005.

[201] Kotaro Tanahashi, Shinichi Takayanagi, Tomomitsu Motohashi, and Shu Tanaka. Application of ising machines and a software development for ising machines. *Journal of the Physical Society of Japan*, 88(6):061010, 2019.

[202] X Tang and L Shu. Classification of electrocardiogram signals with rs and quantum neural networks. *International Journal of Multimedia and Ubiquitous Engineering*, 9(2):363–372, 2014.

[203] Prayag Tiwari and Massimo Melucci. Binary classifier inspired by quantum theory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):10051–10052, Jul. 2019.

[204] Tony T. Tran, Minh Do, Eleanor G. Rieffel, Jeremy Frank, Zhihui Wang, Bryan O'Gorman, Davide Venturelli, and J. Christopher Beck. A hybrid quantum-classical approach to solving scheduling problems. In *Proceedings of the Ninth Annual Symposium on Combinatorial Search, SOCS 2016, Tarrytown, NY, USA, July 6-8, 2016*, pages 98–106. AAAI Press, 2016.

[205] Dan Ventura and Tony Martinez. Quantum associative memory. *arXiv e-prints*, 1998.

[206] Davide Venturelli and Alexei Kondratyev. Reverse quantum annealing approach to portfolio optimization problems. *Quantum Machine Intelligence*, 1(1):17–30, May 2019.

[207] Tomáš Vyskočil, Scott Pakin, and Hristo N. Djidjev. Embedding inequality constraints for quantum annealing optimization. In Sebastian Feld and Claudia Linnhoff-Popien, editors, *Quantum Technology and Optimization Problems*, pages 11–22, Cham, 2019. Springer International Publishing. ISBN: 978-3-030-14082-3.

[208] Chi Wang, Huo Chen, and Edmond Jonckheere. Quantum versus simulated annealing in wireless interference network optimization. *Scientific Reports*, 6(1):25797, May 2016.

[209] Shui-Hua Wang, Shipeng Xie, Xianqing Chen, David S Guttery, Chaosheng Tang, Junding Sun, and Yu-Dong Zhang. Alcoholism identification based on an alexnet transfer learning model. *Frontiers in psychiatry*, 10:205, 2019.

[210] Yuanhao Wang, Ying Li, Zhang-qi Yin, and Bei Zeng. 16-qubit ibm universal quantum computer can be fully entangled. *npj Quantum Information*, 4(1):46, 2018.

[211] Richard H. Warren. Solving the traveling salesman problem on a quantum annealer. *SN Applied Sciences*, 2(1):75, 2019.

[212] Richard H Warren. A benchmark for quantum optimization: the traveling salesman problem. *QUANTUM INFORMATION & COMPUTATION*, 21(7-8):557–562, 2021.

[213] Maurice Weber, Nana Liu, Bo Li, Ce Zhang, and Zhikuan Zhao. Optimal provable robustness of quantum classification via quantum hypothesis testing. *npj Quantum Information*, 7(1): 76, May 2021.

[214] Dennis Willsch, Madita Willsch, Hans De Raedt, and Kristel Michielsen. Support vector machines on the d-wave quantum annealer. *Computer Physics Communications*, 248: 107006, 2020.

[215] Peter Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Academic Press, San Diego, 2014.

[216] Peter Wittek and Christian Gogolin. Quantum enhanced inference in markov logic networks. *Scientific reports*, 7(1):1–8, 2017.

[217] William K. Wootters and Wojciech H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, Oct 1982.

[218] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[219] Fengqi Yan, Xin Huang, Yao Yao, Mingming Lu, and Maozhen Li. Combining lstm and densenet for automatic annotation and classification of chest x-ray images. *IEEE Access*, 7: 74181–74189, 2019.

[220] Sheir Yarkoni, Andrii Kleshchonok, Yury Dzerin, Florian Neukart, and Marc Hilbert. Semi-supervised time series classification method for quantum computing. *Quantum Machine Intelligence*, 3(1), Apr 2021.

[221] Mashiyat Zaman, Kotaro Tanahashi, and Shu Tanaka. Pyqubo: Python library for mapping combinatorial optimization problems to qubo form. *arXiv e-prints*, 2021.

[222] Remmy Zen, Long My, Ryan Tan, Frédéric Hébert, Mario Gattobigio, Christian Miniatura, Dario Poletti, and Stéphane Bressan. Transfer learning for scalability of neural-network quantum states. *Physical Review E*, 101(5):053301, 2020.

[223] Kaining Zhang, Min-Hsiu Hsieh, Liu Liu, and Dacheng Tao. Toward trainability of quantum neural networks. *arXiv e-prints*, 2020.

[224] Lei Zhang, Guang Yang, and Xujiong Ye. Automatic skin lesion segmentation by coupling deep fully convolutional networks and shallow network with textons. *Journal of Medical Imaging*, 6(2):024001, 2019.

[225] Yi Zhang, Kai Lu, Kai Xu, Yinghui Gao, and Richard Wilson. Local feature point extraction for quantum images. *Quantum Information Processing*, 14(5):1573–1588, 2015.

[226] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.

[227] Neng-Fa Zhou, Håkan Kjellerstrand, and Jonathan Fruhman. Encodings for the traveling salesman problem. In *Constraint Solving and Planning with Picat*, pages 129–139. Springer, 2015.

[228] Bernard Zygelman. *A First Introduction to Quantum Computing and Information.* Springer, 2018.