



UNIVERSIDADE D  
COIMBRA

Cláudio Filipe Prata Gomes

**PORTFOLIO OPTIMIZATION IN FINANCIAL MARKETS USING  
QUANTUM COMPUTING: AN EXPERIMENTAL STUDY**

Dissertation in the context of the Master in Informatics Engineering, Specialization in  
Intelligent Systems, advised by Professor João Paulo Fernandes,  
Professor Gabriel Falcão, and Professor Luís Paquete, and presented to  
Faculty of Sciences and Technology / Department of Informatics Engineering.

June 2021

Faculty of Sciences and Technology  
Department of Informatics Engineering

# Portfolio Optimization in Financial Markets using Quantum Computing: An Experimental Study

Cláudio Filipe Prata Gomes

Dissertation in the context of the Master in Informatics Engineering, Specialization in  
Intelligent Systems, advised by Prof. João Paulo Fernandes, Prof. Gabriel Falcão, and  
Prof. Luís Paquete, and presented to the  
Faculty of Sciences and Technology / Department of Informatics Engineering.

June 2021



UNIVERSIDADE D  
COIMBRA

This page is intentionally left blank.

---

## Acknowledgements

The dissertation that I am now delivering is the culmination of more than two years of work. From the first conversation at a summer school in Budapest, to the exploration of potential areas of contribution, to the first step in a new field of research, to the first contribution to a book, to the scholarships, and then to this thesis, I can only say that this journey has been incredible and that I look forward to what the future holds for me!

To my supervisors, Prof. João Paulo Fernandes, Prof. Gabriel Falcão, and Prof. Luís Paquete, I am very grateful for your continuous support, guidance, directions, and knowledge that helped me execute this dissertation with the quality and scientific rigor that I would not be able to provide by myself. I genuinely enjoyed learning a lot with you, and I am very grateful for this opportunity.

To my family and friends, thank you very much for being with me at my best and my worst moments. Your support has been and will always be fundamental. I hope that your patience never depletes and that I can count on you for many more great moments!

I also want to thank the CISUC-DEI for making me part of the SSE group and providing the working environment that helped me complete my dissertation. I also want to express my full gratitude for everyone at DEI-FCTUC, for providing me with everything that I needed to feel included, especially in the current pandemic situation.

The dissertation was conducted with the support from Programme New Talents in Quantum Technologies of the Gulbenkian Foundation (Portugal), to which I thank very much for supporting me in this dissertation and providing me a path to new opportunities!

Thank you all very much, I can almost hear the champagne corks popping at this moment!

Cláudio Filipe Prata Gomes,

Coimbra, June 2021

This page is intentionally left blank.

---

## Abstract

Quantum computing is bound to change the world as we know it. By exploring the properties of quantum theory for computational purposes, it is expected to substantially reduce the amount of problems that are nowadays considered computationally intractable. This means that quantum computers have the power of providing solutions for some of the problems of practical interest for which a classical computer cannot, at least in a timely manner. This is even more revolutionary and remarkable given the fact these problems range from multidisciplinary domains such as Chemistry, Medicine, and, most relevant in the context of this dissertation, Finance.

In this work, we will focus on leveraging quantum computing to addressing a relevant and timely problem within the financial domain. We will target a combinatorial optimization problem, the portfolio optimization problem, which consists of selecting the best portfolio (combination of assets) among all possible portfolios, according to some objective function, whether to maximize return or minimize risk. Due to the high number of parameters, such as the expected return per asset and market conditions, this problem attains an exponential complexity and is an NP-hard problem, intractable in the context of classical computing.

We designed and conducted an empirical study on the effect of parameters on solutions to the portfolio optimization problem given by a quantum computer. In particular, we use a quantum computer from D-Wave Systems, Inc. and vary the parameters related to not only the quantum computer, but also to the portfolio optimization problem itself. We believe that our findings are useful not only for those using adiabatic quantum computers in the context of portfolio optimization problem, and also in other application domains.

Our findings suggest that the parameters do have an effect on the results, whether they are related to the portfolio optimization problem or to the quantum computer. Moreover, we found that some of the parameters have a great impact, such as the chain strength, which defines the strength associated to the couplings between qubits that represent a variable, and that other parameters have no statistically significant effect, such as the anneal schedule or embedding used.

## Keywords

Portfolio Optimization, Quantum Computing, Quadratic Unconstrained Binary Optimization, Combinatorial Optimization

This page is intentionally left blank.

---

## Resumo

A computação quântica está prestes a mudar o mundo tal como o conhecemos. Através da exploração das propriedades da teoria quântica para fins computacionais, é esperada uma redução substancial na quantidade de problemas que hoje são considerados intratáveis. Isto significa que os computadores quânticos têm a capacidade de devolver soluções para alguns problemas de interesse prático para os quais um computador clássico não consegue devolver, pelo menos em tempo útil. Isto é ainda mais revolucionário e notável pelo facto de que esses problemas abrangem domínios multidisciplinares como Química, Medicina e, mais relevante no contexto desta dissertação, Finanças.

Neste trabalho, vamos focar-nos na utilização da computação quântica para abordar um problema relevante e atual no domínio financeiro. Mais especificamente, um problema de otimização combinatorial, o problema de otimização de portfólios, que consiste em selecionar o melhor portfólio financeiro (combinação de ativos) entre um conjunto de todos os portfólios possíveis, de acordo com uma certa função objetivo, comumente de forma a maximizar o retorno esperado ou minimizar o risco. Devido ao grande número de parâmetros, como o retorno esperado por ativo e as condições de mercado, este problema atinge uma complexidade exponencial e é um problema *NP-hard*, intratável no contexto da computação clássica.

Nós desenvolvemos um estudo empírico acerca da influência dos parâmetros nas soluções devolvidas por um computador quântico para o problema de otimização de portfólios. Em particular, utilizamos um computador quântico da D-Wave Systems, Inc. e variamos os parâmetros relacionados não só com o computador quântico, mas também com o problema de otimização de portfólios. Acreditamos que as conclusões do estudo são contribuições úteis para qualquer investigador que deseje utilizar computadores quânticos adiabáticos no contexto do problema de otimização de portfólios e também noutros domínios de aplicação.

As nossas descobertas sugerem que os parâmetros têm efeito nos resultados, quer sejam relacionados com o problema de otimização de portfólios ou com o computador quântico. Além disso, também descobrimos que alguns dos parâmetros têm um grande impacto, tal como o *chain strength*, que define a força com a qual os qubits que representam uma variável estão correlacionados, e que outros não têm nenhum efeito estatisticamente significativo, tais como o *anneal schedule* ou o *embedding*.

## Palavras-Chave

Otimização de Portfólios, Computação Quântica, Otimização Binária Quadrática Irrestrita, Otimização Combinatória



This page is intentionally left blank.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Hypothesis and Objectives . . . . .	2
1.2	Contributions and Results . . . . .	2
1.3	Outline . . . . .	3
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Linear Algebra . . . . .	5
2.2	Quantum Mechanics . . . . .	7
2.2.1	The Hilbert Space and the Quantum Bit . . . . .	7
2.2.2	Evolution of a Quantum System . . . . .	9
2.2.3	Measurement on Quantum Systems . . . . .	10
2.2.4	Systems with Multiple Quantum Bits . . . . .	11
2.3	Gate-based Quantum Computers . . . . .	12
2.4	Adiabatic Quantum Computers . . . . .	14
2.5	Combinatorial Optimization . . . . .	20
2.5.1	Ising Model . . . . .	21
2.5.2	Quadratic Unconstrained Binary Optimization Formulation . . . . .	21
2.5.3	Graph Expression . . . . .	22
2.5.4	Summary of Problem Expressions . . . . .	23
<b>3</b>	<b>Portfolio Optimization</b>	<b>25</b>
3.1	Terminology and Motivation . . . . .	25
3.2	The Portfolio Optimization Problem . . . . .	25
3.2.1	Multiobjective Optimization and Efficient Frontier . . . . .	26
3.2.2	Parameters . . . . .	27
3.2.3	Quadratic Unconstrained Binary Optimization formulation . . . . .	28
3.3	Quantum Computing applied to POP . . . . .	30
<b>4</b>	<b>Approach and Methodology</b>	<b>33</b>
4.1	General Structure . . . . .	33
4.1.1	Universe Size . . . . .	34
4.1.2	Chain Strength . . . . .	35
4.1.3	Number of Reads . . . . .	35
4.1.4	Directions . . . . .	36
4.1.5	Budget . . . . .	37
4.1.6	Embedding . . . . .	38
4.1.7	Anneal Schedule . . . . .	38
4.1.8	Dataset . . . . .	39
4.1.9	Annealing System . . . . .	40
4.2	Quality of Solutions . . . . .	41
4.2.1	Outlining the Approximation Frontier . . . . .	41

4.2.2	Solving the Problem Instance to Optimality . . . . .	43
4.2.3	$\epsilon$ -indicator . . . . .	43
4.3	Statistical Analysis . . . . .	44
<b>5</b>	<b>Workflow Implementation</b>	<b>47</b>
5.1	Dataset Generation . . . . .	48
5.2	Solving with MILP Solver . . . . .	49
5.3	Solving Scenarios with D-Wave . . . . .	50
5.4	$\epsilon$ -indicator Calculation . . . . .	51
5.5	Statistical Analysis . . . . .	52
<b>6</b>	<b>Results and Discussion</b>	<b>53</b>
6.1	Universe Size . . . . .	53
6.2	Chain Strength . . . . .	54
6.3	Number of Reads . . . . .	56
6.4	Directions . . . . .	58
6.5	Budget . . . . .	59
6.6	Embedding . . . . .	61
6.7	Anneal Schedule . . . . .	64
6.8	Dataset . . . . .	65
6.9	Annealing System . . . . .	68
6.10	Main Takeaways . . . . .	69
<b>7</b>	<b>Conclusions and Future Work</b>	<b>73</b>

# Acronyms

- BQM** Binary Quadratic Model. xi, 18, 21
- D-Wave** D-Wave Systems, Inc.. iii, v, viii, xii, 2, 14, 16–19, 21, 28–31, 33, 34, 38, 41, 43, 47, 50, 51, 53, 54, 58–60, 62, 63, 65, 66, 69–71, 73, 74, 87, 88
- GICS** Global Industry Classification Standard. 48
- GLPK** GNU Linear Programming Kit. 50
- IBM** International Business Machines Corporation. 12, 14, 19, 30
- JSON** JavaScript Object Notation. 49
- MILP** Mixed Integer Linear Program. viii, 2, 34, 36, 41, 43, 44, 47, 49, 50
- NaN** Not a Number. 48
- POP** Portfolio Optimization Problem. vii, xi, xiii, 1–3, 25–31, 33, 43, 47, 50, 53, 73
- QPU** Quantum Processing Unit. xi–xiii, 16–19, 23, 30, 31, 73, 87, 88
- QUBO** Quadratic Unconstrained Binary Optimization. vii, xi–xiii, 18, 21–23, 25, 28, 29, 33–35, 50, 51, 87, 88
- S&P** Standard & Poor’s. 48
- SCIP** Solving Constraint Integer Programs. 49, 50, 52
- SDK** Software Development Kit. 14

This page is intentionally left blank.

# List of Figures

2.1	Bloch sphere visualization of the states $ 0\rangle$ , $ 1\rangle$ , and $\frac{ 0\rangle- 1\rangle}{\sqrt{2}}$ , respectively, from left to right . . . . .	9
2.2	Circuit that describes a 2-qubits system with entanglement and measurement	13
2.3	Illustration of the evolution of a system under the adiabatic theorem . . . . .	15
2.4	Energy diagrams of a qubit in the quantum annealing process . . . . .	16
2.5	Energy diagram of a entangled two-qubit system in the quantum annealing process . . . . .	17
2.6	Illustration of part of the Pegasus topology, adapted from [1] . . . . .	17
2.7	Illustration of an embedding of a BQM problem on a QPU . . . . .	18
2.8	Illustration of a piecewise-linear curve followed by a QPU . . . . .	19
2.9	Graph expression of a Quadratic Unconstrained Binary Optimization (QUBO) formulated problem . . . . .	22
3.1	Scatter plot of points associated to each of the feasible solutions of example POP . . . . .	27
4.1	Scatter plot of points from a set of solutions returned by an execution of $\mathcal{S}_{8a}$	42
4.2	Illustration of the approximation frontier of a set of points from a set of solutions returned by an execution of $\mathcal{S}_{8a}$ . . . . .	42
4.3	Illustration of the representation of the efficient frontier for an instance executed in $\mathcal{S}_{8a}$ . . . . .	43
5.1	Diagram of the implemented pipeline of our study . . . . .	47
5.2	MathProg file template of the POP ( <code>IPL_linearized_template.ampl</code> ) . . . . .	50
5.3	Python code to create a QUBO matrix for each direction specified for $q$ ( <code>pop_solver_annealer.py</code> ) . . . . .	51
6.1	Boxplot of $\mathcal{S}_1$ 's results . . . . .	54
6.2	Line chart of $\mathcal{S}_2$ 's results for $n = 16$ . . . . .	55
6.3	Line chart of $\mathcal{S}_2$ 's results for $n = 32$ . . . . .	55
6.4	Line chart of $\mathcal{S}_2$ 's results for $n = 64$ . . . . .	56
6.5	Boxplot of $\mathcal{S}_3$ 's results for $n = 16$ . . . . .	56
6.6	Boxplot of $\mathcal{S}_3$ 's results for $n = 32$ . . . . .	57
6.7	Boxplot of $\mathcal{S}_3$ 's results for $n = 64$ . . . . .	57
6.8	Boxplot of $\mathcal{S}_4$ 's results for $n = 32$ . . . . .	58
6.9	Boxplot of $\mathcal{S}_4$ 's results for $n = 64$ . . . . .	59
6.10	Boxplot of $\mathcal{S}_5$ 's results for $n = 32$ . . . . .	60
6.11	Boxplot of $\mathcal{S}_5$ 's results for $n = 64$ . . . . .	60
6.12	Boxplot of $\mathcal{S}_6$ 's results for $n = 16$ . . . . .	61
6.13	Boxplot of $\mathcal{S}_6$ 's results for $n = 32$ . . . . .	62
6.14	Boxplot of $\mathcal{S}_6$ 's results for $n = 64$ . . . . .	62
6.15	Boxplot of $\mathcal{S}_7$ 's results for $n = 16$ . . . . .	64

6.16	Boxplot of $\mathcal{S}_7$ 's results for $n = 32$ . . . . .	64
6.17	Boxplot of $\mathcal{S}_7$ 's results for $n = 64$ . . . . .	65
6.18	Boxplot of $\mathcal{S}_8$ 's results for $n = 16$ . . . . .	66
6.19	Boxplot of $\mathcal{S}_8$ 's results for $n = 32$ . . . . .	66
6.20	Boxplot of $\mathcal{S}_8$ 's results for $n = 64$ . . . . .	67
6.21	Boxplot of $\mathcal{S}_9$ 's results for $n = 16$ . . . . .	68
6.22	Boxplot of $\mathcal{S}_9$ 's results for $n = 32$ . . . . .	69
6.23	Boxplot of $\mathcal{S}_9$ 's results for $n = 64$ . . . . .	69
1	Python code to import the necessary tools and load the IBM Q account . . .	83
2	Python code to setup a quantum circuit . . . . .	83
3	Python code to draw the quantum circuit . . . . .	83
4	Drawing of the quantum circuit implemented in Qiskit . . . . .	84
5	Python code to run the circuit on a real quantum device and plot the results	84
6	Results of the quantum circuit implemented in Qiskit . . . . .	84
7	Python code to import the necessary tools to create a QUBO and gain access to the D-Wave QPU . . . . .	87
8	Python code to prepare a QUBO matrix . . . . .	87
9	Python code to execute a QUBO matrix on the D-Wave QPU . . . . .	88

# List of Tables

2.1	Table of the most common quantum gates with their operator name, abbreviation, circuit form, and corresponding unitary matrices . . . . .	12
2.2	Table of equivalence between Quantum Processing Unit (QPU) terminology, Ising models, QUBO formulations, and graph expressions . . . . .	23
3.1	Expected return and volatility of each of the feasible solutions of example POP . . . . .	27
4.1	Values of parameters for each strategy in $\mathcal{S}_1$ . . . . .	34
4.2	Values of parameters for each strategy in $\mathcal{S}_2$ . . . . .	35
4.3	Values of parameters for each strategy in $\mathcal{S}_3$ . . . . .	36
4.4	Values of parameters for each strategy in $\mathcal{S}_4$ . . . . .	37
4.5	Values of parameters for each strategy in $\mathcal{S}_5$ . . . . .	37
4.6	Values of parameters for each strategy in $\mathcal{S}_6$ . . . . .	38
4.7	Values of parameters for each strategy in $\mathcal{S}_7$ . . . . .	39
4.8	Values of parameters for each strategy in $\mathcal{S}_8$ . . . . .	39
4.9	Values of parameters for each strategy in $\mathcal{S}_9$ . . . . .	40
6.1	Output of the <i>Conover-Iman test</i> on $\mathcal{S}_1$ 's results . . . . .	53
6.2	Output of the pairwise <i>t-test</i> on $\mathcal{S}_5$ 's results for $n = 32$ . . . . .	61
6.3	Output of the <i>Conover-Iman test</i> on $\mathcal{S}_6$ 's results for $n = 16$ . . . . .	63
6.4	Output of the pairwise <i>t-test</i> on $\mathcal{S}_6$ 's results for $n = 32$ . . . . .	63
6.5	Output of the <i>Conover-Iman test</i> on $\mathcal{S}_8$ 's results for $n = 16$ . . . . .	67
6.6	Output of the <i>Conover-Iman test</i> on $\mathcal{S}_8$ 's results for $n = 32$ . . . . .	67
6.7	Output of the <i>Conover-Iman test</i> on $\mathcal{S}_8$ 's results for $n = 64$ . . . . .	68



This page is intentionally left blank.

# Chapter 1

## Introduction

There are many problems of practical interest that are known to be computationally intractable using classical computers [2]. This includes, and is not limited to, knapsack with integer weights, graph coloring, traveling salesman, and generating true random numbers [3].

While solutions (or algorithms) for solving such problems may exist, classical computers are not able to execute them for realistic inputs and produce the desired results in a timely manner. In practice, classical computers are “only” able to produce results, given sufficiently large inputs, for some of the programs whose execution time grows polynomially with the size of the inputs.

While the physical implementation of quantum computers has been studied for decades already, the advent of promising and publicly available quantum computers has brought a renewed focus and expectation regarding Quantum Computing. In fact, quantum computers have already been demonstrated in practice to be able to solve concrete problems that classical computers never could in reasonable time — i.e., that quantum advantage has been achieved in practice.

In 2019, Google was the first company to claim this impressive feat by using a processor with 53 qubits to prove that a random-number generator was truly random [4]. More recently, in December 2020, the Hefei team, from China, demonstrated quantum advantage by solving a problem that is virtually unassailable by any classical computer [5]. This team calculated the probability distribution of many bosons in 200 seconds, which is faster than an execution on a classical computer by a factor of around  $10^{14}$ ! Moreover, Accenture has published a whitepaper that claims that RSA, which is foundational to modern cryptography, is going to be broken by quantum computers by around 2025 [6]. All in all, quantum computing is getting more and more promising each day.

In this work, we will focus on a problem from the financial domain, where optimization problems abound, and where a quantum speedup is highly expected and desired. Specifically, we will focus on a combinatorial optimization problem, the Portfolio Optimization Problem (POP), which constitutes one of the main study objects of our work. This is such a relevant problem that it granted Harry Markowitz a Nobel prize in Economic Sciences for his work on it in 1990 [7]!

The POP consists of selecting the best financial portfolio (set of assets) out of the set of all possible portfolios, according to some objective function, whether to maximize the expected return or to minimize the risk associated with the investment. Due to the high number of parameters, such as the expected return per asset and market conditions, this

problem attains an exponential complexity and becomes NP-hard [8], intractable in the context of classical computing.

The exhaustive analysis of the state of the art in quantum computing applied to POP, both for gate-based quantum computers and for adiabatic quantum computers, allowed identifying a potential contribution that was not covered before. We designed and conducted an empirical study on the effect of parameters on solutions returned by an adiabatic quantum computer to the POP. In particular, we will use a quantum computer from D-Wave Systems, Inc. (D-Wave) and vary the parameters related not only to the quantum computer, but also to the POP. The parameters include the chain strength, the anneal schedule, the annealing system, the embedding, and the number of reads, which relate to the quantum computer, as well as the universe size, the budget size, the directions, and the dataset, which relate to the POP.

We believe that our findings are useful not only for those using adiabatic quantum computers in the context of portfolio optimization problem, but also for those using these computers in other application domains. In the next sections, we describe the main hypothesis supporting our work, as well as the contributions we made.

## 1.1 Hypothesis and Objectives

Our work seeks to shed light whether the following hypothesis is confirmed or dismissed:

*“The configuration of the parameters from both the portfolio optimization problem and the quantum computer influences the quality of the solutions returned by the quantum computer to the portfolio optimization problem.”*

If we confirm the hypothesis, we then wish to find which parameters in particular have the largest influence, and how can we leverage such influences to improve the quality of the solutions returned by the quantum computer.

## 1.2 Contributions and Results

In the course of our work, a series of contributions were made, together with findings. Our main contributions are the following:

- We conducted an extensive review and analysis of the state of the art in quantum computing, with a focus on its application on the POP.
- We implemented a quantum algorithm to solve POP instances on a D-Wave computer.
- We designed and conducted an empirical study on the effect that parameters from both the POP and the quantum computer have on the quality of the solutions returned by the quantum computer to the POP.
- We developed and implemented a workflow that solves a series of POP instances in a MILP solver and in a D-Wave computer. The MILP solver results are then used as a baseline to which we compare D-Wave’s results, assessing the performance of the adiabatic quantum computer in these instances.

- We performed a systematic analysis of the results we obtained, drawing insights that can be useful for both practitioners and researchers.

Regarding our findings, they suggest that certain parameters do have an effect on the results, particularly when considering the POP-related parameters universe size, budget size, and dataset, which have a great impact, since they directly affect the complexity of the problem. On the other hand, there are some parameters related to the quantum computer that also have a great impact on the quality of the solutions, such as the number of solutions that are read in each execution of the problem instance, as well as the strength between different qubits representing a variable. These parameters will all be explained in depth in Section 2.4 and Section 3.2.2.

### 1.3 Outline

The remainder of this document is aligned with the following structure:

- **Chapter 2 — Background** contains an introductory review on concepts necessary to understand the thesis work, including a review on the state of the art in quantum computers.
- **Chapter 3 — Portfolio Optimization** describes the POP, as well as the concepts associated with it.
- **Chapter 4 — Approach and Methodology** contains our approach and the methodology we developed for our work.
- **Chapter 5 — Workflow Implementation** explains our implementation of the methodology, including a pointer to the code repository of the work.
- **Chapter 6 — Results and Discussion** contains the results from our empirical study and the main takeaways from those results.
- **Chapter 7 — Conclusions and Future Work** presents the conclusions from our work and indicates open paths that can be explored for future work.

This page is intentionally left blank.

# Chapter 2

## Background

This chapter will cover the elementary concepts and the state of the art of the scope and domain of this thesis, outlined as follows:

- **Section 2.1** presents a review on linear algebra that is necessary to understand the concepts behind quantum mechanics.
- **Section 2.2** introduces the postulates and main concepts of quantum mechanics.
- **Section 2.3** describes a category of quantum computers that follow the circuit model to solve problems.
- **Section 2.4** describes a category of quantum computers that follow the adiabatic theorem to solve problems.
- **Section 2.5** introduces and illustrates combinatorial optimization with definitions and examples of models that are used throughout the work.

### 2.1 Linear Algebra

Linear algebra is the language used to describe quantum computations. In this sense, we start by doing an introductory review on linear algebra and by becoming acquainted with the bra-ket notation. For the sake of simplicity, let us assume that  $z$  is a complex number described by a vector  $e + fi$  and that  $A$  is a matrix  $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ , where  $e$  and  $f$  are real numbers, and  $a$ ,  $b$ ,  $c$  and  $d$  are complex numbers.

**Complex Conjugate** The complex conjugate of a complex number  $z$  is denoted by  $z^*$ , such that  $z^* = e - fi$ . The complex conjugate of a matrix  $A$  is denoted by  $A^*$ , such that  $A^* = \begin{bmatrix} a^* & b^* \\ c^* & d^* \end{bmatrix}$ .

For example, if  $A = \begin{bmatrix} 3 & i \\ 8i & 3 + 7i \end{bmatrix}$ , then  $A^* = \begin{bmatrix} 3 & -i \\ -8i & 3 - 7i \end{bmatrix}$ .

**Transpose** The transpose of a matrix  $A$  is denoted by  $A^T$ , such that  $A^T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$ .

For example, if  $A = \begin{bmatrix} 3 & i \\ 8i & 3 + 7i \end{bmatrix}$ , then  $A^T = \begin{bmatrix} 3 & 8i \\ i & 3 + 7i \end{bmatrix}$ .

**Hermitian Conjugate** The Hermitian conjugate of a matrix  $A$  is denoted by  $A^\dagger$ , such that  $A^\dagger = \begin{bmatrix} a^* & c^* \\ b^* & d^* \end{bmatrix}$ . Note that  $A^\dagger = (A^T)^*$  and that  $A^\dagger$  can also be called the *adjoint* of  $A$ .

For example, if  $A = \begin{bmatrix} 3 & i \\ 8i & 3 + 7i \end{bmatrix}$ , then  $A^\dagger = \begin{bmatrix} 3 & -8i \\ -i & 3 - 7i \end{bmatrix}$ .

**Vector Space** Vector spaces are the building blocks of linear algebra. Each element of a vector space is called *vector*. In the context of this thesis,  $C^n$  is the vector space of most interest, comprising all  $n$ -tuples of complex numbers  $(z_1, \dots, z_n)$ .

**Ket Vector** The *ket* is a column vector of complex numbers such that  $|\psi\rangle = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}$ .

**Bra Vector** For a ket vector  $|\psi\rangle = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}$ , its *bra* vector is a row vector of complex conjugate numbers such that  $\langle\psi| = \begin{bmatrix} z_1^* & \dots & z_n^* \end{bmatrix}$ .

**Inner Product** The inner product between vectors  $|\psi\rangle = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$  and  $|\phi\rangle = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$  is

denoted by  $\langle\psi|\phi\rangle$ , such that  $\langle\psi|\phi\rangle = \begin{bmatrix} a_1^* & \dots & a_n^* \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = a_1^*b_1 + \dots + a_n^*b_n$ . Note that

$\langle\psi|\phi\rangle = \langle\phi|\psi\rangle^*$  and that the inner product between  $|\psi\rangle$  and  $A|\phi\rangle$  or between  $A^\dagger|\psi\rangle$  and  $|\phi\rangle$  is denoted by  $\langle\psi|A|\phi\rangle$ .

For example, if  $|\psi\rangle = \begin{bmatrix} 3 \\ 8i \end{bmatrix}$  and  $|\phi\rangle = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ , then  $\langle\psi|\phi\rangle = \begin{bmatrix} 3 & -8i \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = 3 - 24i$ .

**Tensor Product** The tensor product between vectors  $|\psi\rangle$  and  $|\phi\rangle$  is denoted by  $|\psi\rangle|\phi\rangle$

$$\text{or } |\psi\rangle \otimes |\phi\rangle, \text{ such that } |\psi\rangle \otimes |\phi\rangle = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ \vdots \\ a_1 b_n \\ \vdots \\ a_n b_1 \\ \vdots \\ a_n b_n \end{bmatrix}. \text{ Note that this operation can}$$

also be extended to matrices.

$$\text{For example, if } |\psi\rangle = \begin{bmatrix} 3 \\ 8i \end{bmatrix} \text{ and } |\phi\rangle = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \text{ then } |\psi\rangle \otimes |\phi\rangle = \begin{bmatrix} 3 \\ 8i \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \times 1 \\ 3 \times 3 \\ 8i \times 1 \\ 8i \times 3 \end{bmatrix} = \begin{bmatrix} 3 \\ 9 \\ 8i \\ 24i \end{bmatrix}.$$

**Norm** The norm of a vector  $|\psi\rangle$  is denoted by  $\| |\psi\rangle \|$ , and is calculated as  $\| |\psi\rangle \| = \sqrt{\langle \psi | \psi \rangle}$ . The normalization of a vector  $|\psi\rangle$  is  $\frac{|\psi\rangle}{\| |\psi\rangle \|}$ , which transforms it into a unit vector — a vector of size 1.

$$\text{For example, if } |\psi\rangle = \begin{bmatrix} 6 \\ 8i \end{bmatrix}, \text{ then its norm is } \| |\psi\rangle \| = \sqrt{\langle \psi | \psi \rangle} = \sqrt{6 \times 6 - 8i \times 8i} = \sqrt{36 + 64} = \sqrt{100} = 10. \text{ Thus, if we normalize } |\psi\rangle, \text{ we obtain the unit vector } \frac{|\psi\rangle}{\| |\psi\rangle \|} = \frac{1}{10} \begin{bmatrix} 6 \\ 8i \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.8i \end{bmatrix}.$$

## 2.2 Quantum Mechanics

Classical computing is ubiquitous and can be found almost everywhere, in our personal computers, smartphones, televisions, and many more devices. These computers obey to a series of classical properties that can be described as Boolean algebra, i.e. as variables comprised of truth values *true* and *false*.

Quantum computing follows an entirely different mathematical framework, called quantum mechanics. During a long process of trial and error by the creators of the quantum physics theory, four postulates were iterated over, and are presented next.

### 2.2.1 The Hilbert Space and the Quantum Bit

**First postulate** In [3], we know that “associated to any isolated physical system is a complex vector space with inner product (that is, 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.”

In the context of quantum computers, this postulate relates to the notion of a *quantum bit*, a two-dimensional state space called *qubit* in the remaining of this thesis. The qubit is



the simplest quantum mechanical system whose state space has an orthonormal basis that can be formed by  $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$  and  $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ . Hence, any arbitrary state vector  $|\psi\rangle$  can be defined in a linear combination such that:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad (2.1)$$

where the coefficients  $\alpha$  and  $\beta$  are complex numbers that are also denoted *probability amplitudes*, for reasons that will be explained in depth in Section 2.2.3. From the first postulate,  $|\psi\rangle$  must be a unit vector, since the inner product of  $|\psi\rangle$  with itself is 1,  $\langle\psi|\psi\rangle = 1$ , and this is equivalent to the *normalization condition* for state vectors:

$$|\alpha|^2 + |\beta|^2 = 1 \quad (2.2)$$

Note that the qubit may have an orthonormal set of basis vectors other than  $|0\rangle$  and  $|1\rangle$ . Since this is a formalism that does not affect any property from quantum mechanics, it is fixed in advance that the dissertation uses this set.

The states  $|0\rangle$  and  $|1\rangle$  are analogous to the *classical bit's* truth values *False* and *True* from the Boolean algebra used by classical computing. The qubit differentiates itself from the bit by making it possible to have *superpositions* of these two states, in which a qubit is neither certainly in the state  $|0\rangle$  nor certainly in the state  $|1\rangle$ . Superpositions are defined by the linear combination:

$$\begin{aligned} & \sum_i \alpha_i |\psi_i\rangle \\ & \text{subject to } \sum \alpha_i = 1 \end{aligned} \quad (2.3)$$

where  $\alpha_i$  is the probability amplitude of the state  $|\psi_i\rangle$ . For example, one superposition could be defined as:

$$\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \quad (2.4)$$

where the probability amplitudes are  $\alpha_1 = \frac{1}{\sqrt{2}}$ ,  $\alpha_2 = -\frac{1}{\sqrt{2}}$ , and the states are  $|\psi_1\rangle = |0\rangle$ , and  $|\psi_2\rangle = |1\rangle$ .

The Hilbert space can be visualized via the Bloch sphere, a *unit sphere* with three axes:  $x$ ,  $y$ , and  $z$ . The points in the surface of this unit sphere represent every possible state that a single qubit may have. Figure 2.1 shows three Bloch spheres with a red arrow representing the familiar states  $|0\rangle$ ,  $|1\rangle$ , and  $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ , respectively.

It is important to note that, despite the fact that the Hilbert space tells us how to describe the state of any system, quantum mechanics does not tell us the specific state of a particular system. Finding the specific state of a particular system is a difficult problem that is still in research by physicists and is out of the scope of this thesis.

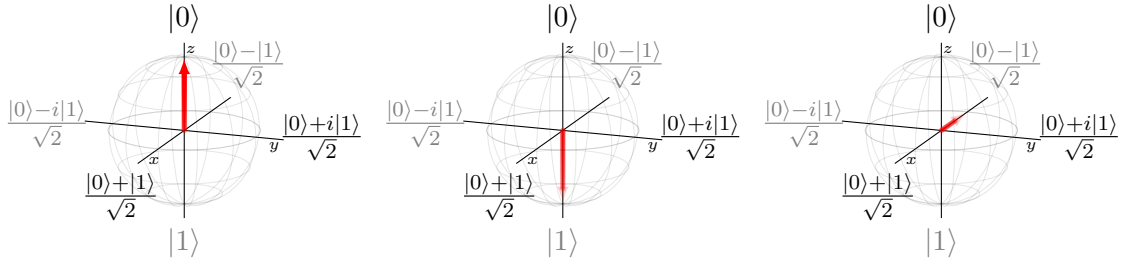


Figure 2.1: Bloch sphere visualization of the states  $|0\rangle$ ,  $|1\rangle$ , and  $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ , respectively, from left to right

## 2.2.2 Evolution of a Quantum System

**Second postulate** As defined in [3], “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  $|\psi'\rangle$  of the system at time  $t_2$  by a unitary operator  $U$  which depends only on the times  $t_1$  and  $t_2$ ,”

$$|\psi'\rangle = U |\psi\rangle \quad (2.5)$$

The second postulate specifies that quantum systems are closed systems, i.e. that do not interact with other systems in any way, and that the transformation of the system’s state between any two given points in time ( $t_1$  and  $t_2$ ) can be described by a single unitary operator.

Note that this unitary operator cannot depend on the state  $|\psi\rangle$ , which means that we cannot conditionally apply it. In other words, we cannot apply an operator on specific states, the transformation is always applied regardless of the system’s state.

For example, let us assume that a transformation  $U$  was applied to a system with the state  $|\psi\rangle = |0\rangle$ , which evolved to the state  $|\psi'\rangle = |1\rangle$ . This transformation can be described by the following unitary operator, also known as the Pauli  $\sigma_x$  operator:

$$U = \sigma_x = X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (2.6)$$

Calculating the state that is reached can be done as follows:

$$U |\psi\rangle = \sigma_x |0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle = |\psi'\rangle \quad (2.7)$$

Similarly to quantum states, quantum mechanics also does not tell us which unitary operators describe particular real-world phenomena. Therefore, finding the specific unitary operator that describes a real-world phenomenon is also a physics problem out of the scope of this thesis.

### 2.2.3 Measurement on Quantum Systems

**Third postulate** Once again, in [3], we know that “quantum measurements are described by a collection  $\{M_m\}$  of *measurement operators*. These are operators acting on the state space of the system being measured. The index  $m$  refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is  $|\psi\rangle$  immediately before the measurement then the probability that result  $m$  occurs is given by

$$p(m) = \langle \psi | M_m^\dagger M_m | \psi \rangle, \quad (2.8)$$

and the state of the system after the measurement is

$$\frac{M_m |\psi\rangle}{\sqrt{\langle \psi | M_m^\dagger M_m | \psi \rangle}}. \quad (2.9)$$

The measurement operators satisfy the *completeness equation*,

$$\sum_m M_m^\dagger M_m = I. \quad (2.10)$$

Let us assume that we are measuring a qubit in the computational basis —  $|0\rangle$  and  $|1\rangle$ . The measurement operators are:

$$M_0 = |0\rangle \langle 0| = \begin{bmatrix} 1 & \\ & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad (2.11)$$

$$M_1 = |1\rangle \langle 1| = \begin{bmatrix} & \\ & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (2.12)$$

which can be verified to obey the completeness equation since  $M_0^\dagger M_0 + M_1^\dagger M_1 = I$ . It is also useful to note that  $M_0^\dagger M_0 = M_0$ , hence for a state  $|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$ :

$$p(0) = \langle \psi | M_0^\dagger M_0 | \psi \rangle = \langle \psi | M_0 | \psi \rangle = \begin{bmatrix} a^* & b^* \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a^* & b^* \end{bmatrix} \begin{bmatrix} a \\ 0 \end{bmatrix} = |a|^2 \quad (2.13)$$

which means that the probability of measuring  $|0\rangle$  is the square of its probability amplitude,  $|a|^2$ , which is why this coefficient is denoted *probability amplitude*. This property can be generalized to any of the basis states comprising the computational basis.

To calculate the state of the system after measuring it, we use Expression (2.9):

$$|\psi'\rangle = \frac{M_0 |\psi\rangle}{\sqrt{\langle \psi | M_0^\dagger M_0 | \psi \rangle}} = \frac{\begin{bmatrix} a \\ 0 \end{bmatrix}}{\sqrt{|a|^2}} = \frac{a}{|a|} |0\rangle \implies |0\rangle \quad (2.14)$$

which can also be generalized to any of the basis states in the computational basis. That is, after measuring a system in the computational basis, the system's state collapses to the measured basis state. Note that this is because multipliers such as  $\frac{a}{|a|}$ , that have a modulus of 1, can be ignored in practice, so the post-measurement states are precisely the measured basis states. The reasoning behind this is out of scope for this thesis.

There is an open debate whether the third postulate can be derived from the second postulate [3]. Since the measuring system and the quantum system being measured comprise a larger isolated quantum system, it should be possible to describe the evolution of this larger isolated system as a unitary operator, according to the second postulate. This debate is out of the scope of this thesis and in practice it is possible to apply both the second and third postulates without worrying about deriving one from the other.

## 2.2.4 Systems with Multiple Quantum Bits

**Fourth postulate** Finally, as described in [3], “the state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through  $n$ , and system number  $i$  is prepared in the state  $\rho_i$ , then the joint state of the total system is  $\rho_1 \otimes \rho_2 \otimes \dots \otimes \rho_n$ .”

The fourth postulate is the easiest one to understand and apply. Let us assume that we have a system with state  $|\psi_1\rangle = |1\rangle$  and another system with state  $|\psi_2\rangle = |0\rangle$ . The composite system  $|\phi\rangle$  that is comprised of those two systems has its state described as:

$$|\phi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (2.15)$$

which is also denoted by  $|10\rangle$  (or  $|2\rangle$  in decimal notation). In the same way, a composite system  $|\phi\rangle$  that is comprised of  $n$  qubits, all in the state  $|\psi\rangle$  can be described as:

$$|\phi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \dots \otimes |\psi_n\rangle = |\psi\rangle^{\otimes n} \quad (2.16)$$

Composite systems enable another property that has no classical analog, *quantum entanglement*. This property lets practitioners correlate two or more qubits in such a way that it becomes impossible to describe each individual qubit of the composite system by itself, and that by measuring one qubit they are affecting the state of the other qubits. For example, for a composite system of two qubits, one of the simplest and maximal states of entanglement is:

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad (2.17)$$

In practice, if we measure a single qubit of a system in the state  $|\Phi^+\rangle$ , the result is indeterminate — i.e. random. However, upon measuring it, the result of measuring the second qubit is guaranteed to yield the same value. For example, if measuring the first qubit yields zero, measuring the second qubit will also yield zero (00), and if measuring the first qubit yields one, measuring the second qubit will also yield one (11). Both situations

have the same probability of happening. Hence, the two qubits are correlated in a way that allows quantum computers to process information beyond what is possible in classical computers [3].

## 2.3 Gate-based Quantum Computers

In this section, we describe gate-based quantum computers, such as the quantum computers from International Business Machines Corporation (Hereinafter denoted IBM), openly accessible through the IBM Quantum Experience platform [9]. These computers follow the quantum circuit model, in which a computation is a sequence of quantum gates. That is, the evolution of a quantum system is described and programmed by a series of quantum gates that represent unitary operators.

These quantum gates vary from computer to computer, but every full-fledged computer has a set of quantum gates such that it is *universal* — i.e., a set of quantum gates that can be combined to represent every possible unitary operator from quantum mechanics. Table 2.1 lists the most common quantum gates with their operator name, abbreviation, circuit form, and corresponding unitary matrices.

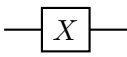
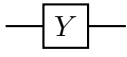
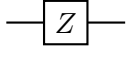
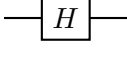
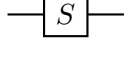
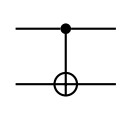
Operator name	Abbreviation	Circuit form	Unitary matrix
Pauli- $X$	$\sigma_x, X$		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli- $Y$	$\sigma_y, Y$		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli- $Z$	$\sigma_z, Z$		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard	$H$		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase	$S$		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
Controlled $X$	$CX$		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Table 2.1: Table of the most common quantum gates with their operator name, abbreviation, circuit form, and corresponding unitary matrices

For example, if we wanted to describe a quantum system with 2 qubits that are entangled and measured, we would apply the following circuit:

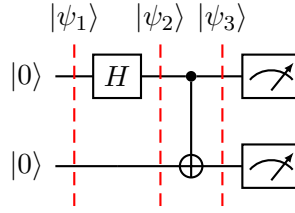


Figure 2.2: Circuit that describes a 2-qubits system with entanglement and measurement

The circuit describes a system that evolves along the following states.

The original state is:

$$\begin{aligned}
 |\psi_1\rangle &= |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
 &= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |00\rangle,
 \end{aligned} \tag{2.18}$$

and, after applying the Hadamard gate, we get:

$$\begin{aligned}
 |\psi_2\rangle &= H |0\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
 &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \frac{|00\rangle + |10\rangle}{\sqrt{2}}
 \end{aligned} \tag{2.19}$$

Finally, by applying the controlled  $X$  gate, we reach the final state:

$$\begin{aligned}
 |\psi_3\rangle &= CX |\psi_2\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\
 &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}
 \end{aligned} \tag{2.20}$$

which is the maximal example of entanglement mentioned in Section 2.2.4 that, upon measurement, will return 00 or 11, with equal probabilities.

The IBM Quantum Experience platform lets practitioners run quantum circuits on real quantum hardware with an open access model. That is, practitioners can submit any number of circuits to IBM’s open access quantum computers, subject to a fair use policy and waiting queues. The circuits can be developed both via a simple interface in IBM Quantum Experience’s website and via Qiskit programs. Qiskit is an open-source Software Development Kit (SDK) founded by IBM Research, the research and development division for IBM [10]. This SDK lets users implement hybrid (classical and quantum) programs with a high-level of detail and control. Appendix A shows an implementation of the circuit from Figure 2.2 in Qiskit, as well as its execution results on a real quantum computer, IBM Q Athens.

At the moment, IBM’s most advanced quantum computer is IBM Q Kolkata, with a quantum volume level of 128, achieved by reliably executing a circuit with 7 qubits ( $2^7 = 128$ ) [9]. However, the best openly accessible quantum computer is IBM Q Manila, with a quantum volume level of 32, achieved by reliably executing a circuit with 5 qubits ( $2^5 = 32$ ) [9]. Quantum volume  $V_Q$  is a performance metric that measures the biggest possible circuit a quantum computer can execute with regards to its width (number of qubits)  $N$  and its depth  $d(N)$  (number of steps that can be executed with a low rate of error) [11]:

$$\log_2 V_Q = \arg \max_N \min(N, d(N)) \quad (2.21)$$

That is,  $V_Q$  is the largest square-shaped circuit (i.e.,  $N = d(N)$ ) that a quantum computer can execute with a low rate of error. For example, if we have a quantum computer that successfully implements circuits with  $N = d(N) = 10$ , but fails to implement circuits with  $N = d(N) = 11$ , then its  $V_Q$  is  $2^N = 2^{10} = 1024$ . In short, the quantum volume tells us that the state-of-the-art gate-based quantum computer from IBM can reliably execute problems up to a maximum of seven variables ( $V_Q = 2^7$ ), whereas the best openly accessible gate-based quantum computer from IBM can reliably execute problems up to a maximum of five variables ( $V_Q = 2^5$ ).

## 2.4 Adiabatic Quantum Computers

In this section, we describe adiabatic quantum computers, a category of quantum computers that follow the adiabatic theorem, such as the computers from D-Wave Systems, Inc., hereinafter denoted D-Wave. To understand this theorem, it is important to revise the second postulate, explained in depth in Section 2.2.2, to first understand another concept in quantum mechanics: the *Hamiltonian* of a system. The Hamiltonian  $H$  is an operator that describes the total energy of the system  $|\psi\rangle$ , which includes both kinetic energy and potential energy [3]. This operator is part of the *Schrödinger equation*:

$$i\hbar \frac{d|\psi\rangle}{dt} = H |\psi\rangle \quad (2.22)$$

The equation is another way to describe the time evolution of the state of a closed quantum system, equivalent to Expression 2.5. This equation uses an experimentally determined physical constant, *Planck’s constant*, denoted  $\hbar$ . In this sense, knowing the Hamiltonian is equivalent to knowing the dynamics of the system as it evolves. The Hamiltonian can be decomposed as follows:

$$H = \sum_E E |E\rangle \langle E|, \quad (2.23)$$

where  $E$  are the eigenvalues of the system and  $|E\rangle$  are the corresponding eigenstates. The eigenvalues are the possible value outcomes from a measurement of the system, each associated to an eigenstate. The eigenstates are states of the system that cause a specific outcome upon measurement, their associated eigenvalues. We present an analogous example to understand this concept. Let us assume that we have a tennis ball with a mass of  $1\text{kg}$ . In this case, the mass is an observable quantity that can be measured and is associated to an operator. This operator would state that the tennis ball has an eigenstate with a mass of  $1\text{kg}$ , and that the corresponding outcome would be a result of  $1\text{kg}$ . While in the case of the tennis ball the outcome is guaranteed to be  $1\text{kg}$ , in the case of systems in a superposition of states, the operator would state that there are different eigenstates, each with its associated eigenvalue, that has its own probability of outcome. The Hamiltonian is equivalent to that operator, but relative to the total energy of the system.

The eigenstates of an Hamiltonian are also called stationary states, since a system in an eigenstate remains in that state if no external perturbation occurs. The stationary state with the lowest energy is called ground state.

The adiabatic theorem states that a system with an initial Hamiltonian remains in its stationary state as long as the external conditions change slowly [12]. Therefore, the system will end in the corresponding eigenstate of the final Hamiltonian, whose outcome is usually designed to be a solution to a problem, in the context of adiabatic quantum computing. It is important to note that there must be large enough gaps in the energy levels between different eigenstates such that the changing conditions do not cause jumps between neighbor states. In fact, the enlargement of these gaps is one of the main problems of interest in the area of adiabatic quantum computing [13]. Figure 2.3 illustrates the evolution of a system with two stationary states while slowly transforming its initial Hamiltonian into a final Hamiltonian.

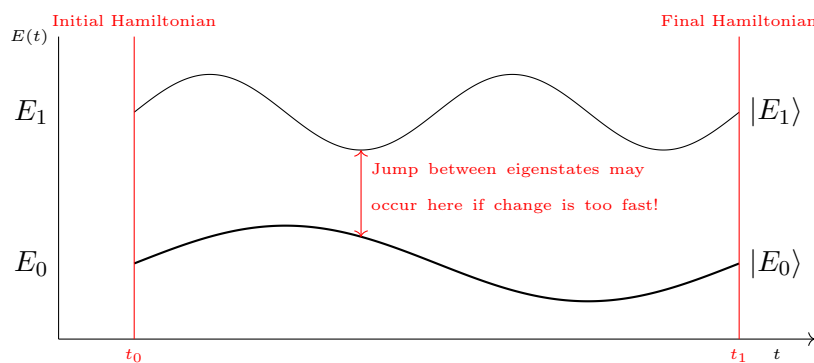


Figure 2.3: Illustration of the evolution of a system under the adiabatic theorem

Adiabatic quantum computers take advantage of this theorem by preparing a simple Hamiltonian and initializing it to the ground state ( $|E_0\rangle$  in the illustration). Afterwards, practitioners find a Hamiltonian whose ground state describes the solution to the problem of interest. Finally, the simple Hamiltonian is adiabatically evolved to the desired final Hamiltonian. If the evolution is slow enough and the gaps between energy levels are sufficiently large, the system remains in the ground state during the entire evolution. Hence, at the end of the evolution, the state of the system corresponds to the ground state of the final Hamiltonian — the solution to the problem of interest!



In the case of D-Wave, the Quantum Processing Unit (QPU) present in every quantum computer takes advantage from adiabatic evolution by implementing quantum annealing in order to solve a specific set of problems. This annealing process applies the idea illustrated in Figure 2.3, by seeking low-energy states while trying to keep external interference at a minimum in order to minimize the probability of jumps between stationary states. Nonetheless, even if jumps do occur, the final state may still provide a useful enough solution. When measuring qubits, the QPU observes their spin direction to determine the final value. The measurement on each qubit returns the value 0 if the qubit is in spin up ( $s^\uparrow$ ), and returns the value 1 if the qubit is in spin down ( $s^\downarrow$ ). Let us suppose that we have a qubit as shown in the energy diagram of Figure 2.4.

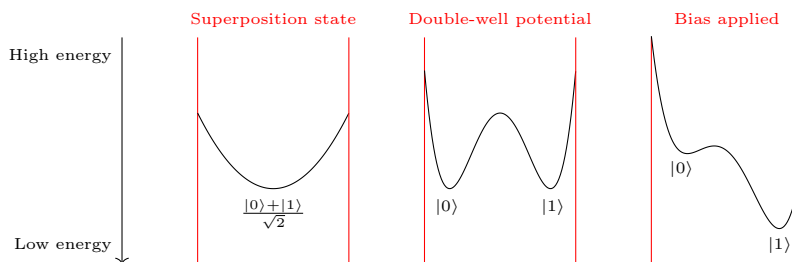


Figure 2.4: Energy diagrams of a qubit in the quantum annealing process

Initially, the qubit is in a superposition state, with a single minimum. Once the quantum annealing process begins, the energy diagram transforms into a double-well potential, where both states  $|0\rangle$  and  $|1\rangle$  have a valley with the same depth — if no more transformations are applied, the probability of the qubit ending in the state  $|0\rangle$  or  $|1\rangle$  is equal. However, the computer can control the depth of both valleys, by applying an external magnetic field to the qubit — a *bias*. Once applied, the bias shifts the probability of the qubit ending in one of the states  $|0\rangle$  and  $|1\rangle$ . Figure 2.4 illustrates a bias on the state  $|1\rangle$ , which lowers the energy level associated to this state, and thus also increases the probability of the qubit ending in state  $|1\rangle$ . In this sense, the programmable bias is equivalent to applying an operator.

However, this programmable bias is not useful by itself. There is also a programmable *coupler* in the system. The coupler links two qubits together such that they become entangled and the correlation weight between coupled qubits can be adjusted by the computer. Several couplers can work together to form a *chain* of qubits that are all linked together. Entangled qubits can be thought of as a single object with  $2^N$  possible states, being  $N$  the number of qubits. Hence, the energy of each state is the sum of the biases of the qubits and the coupling between them. As an example, we have the energy diagram of a entangled two-qubit system in Figure 2.5, where the state  $|00\rangle$  is the most probable one to end up with —  $00$  is the solution for the designed problem! In short, we need to specify values for the biases and couplers of the computer, in order to define an energy landscape such that its minimum value is the solution for the problem that we want to solve.

D-Wave computers' qubits are implemented by following a specific topology of qubits and couplers. That is, the topology of a computer describes the number and pattern of the qubits and couplers in the system. For example, the Pegasus topology, currently implemented in the state-of-the-art computers from D-Wave, is illustrated in Figure 2.6. Each line represents a qubit and its orientation (vertical or horizontal). To understand this topology, let us focus on the green qubit. There are three types of couplers: internal, external, and odd. First, internal couplers connect each qubit to its overlapping qubits, that have a different orientation — the green qubit is linked to the twelve dark gray qubits via internal couplers. Next, external couplers connect each qubit to its front and back

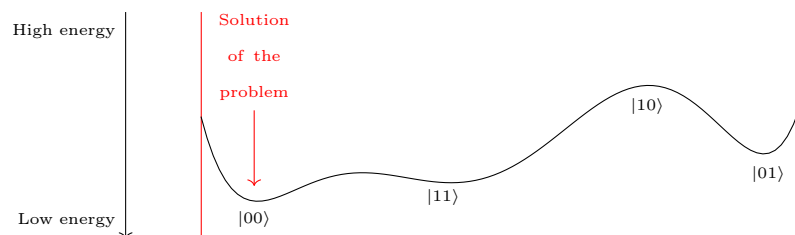


Figure 2.5: Energy diagram of an entangled two-qubit system in the quantum annealing process

neighbor qubits with the same orientation — the green qubit is linked to the blue qubits via external couplers. Last, odd couplers link each qubit to its side neighbor qubit with the same orientation — the green qubit is linked to the red qubit via an odd coupler.

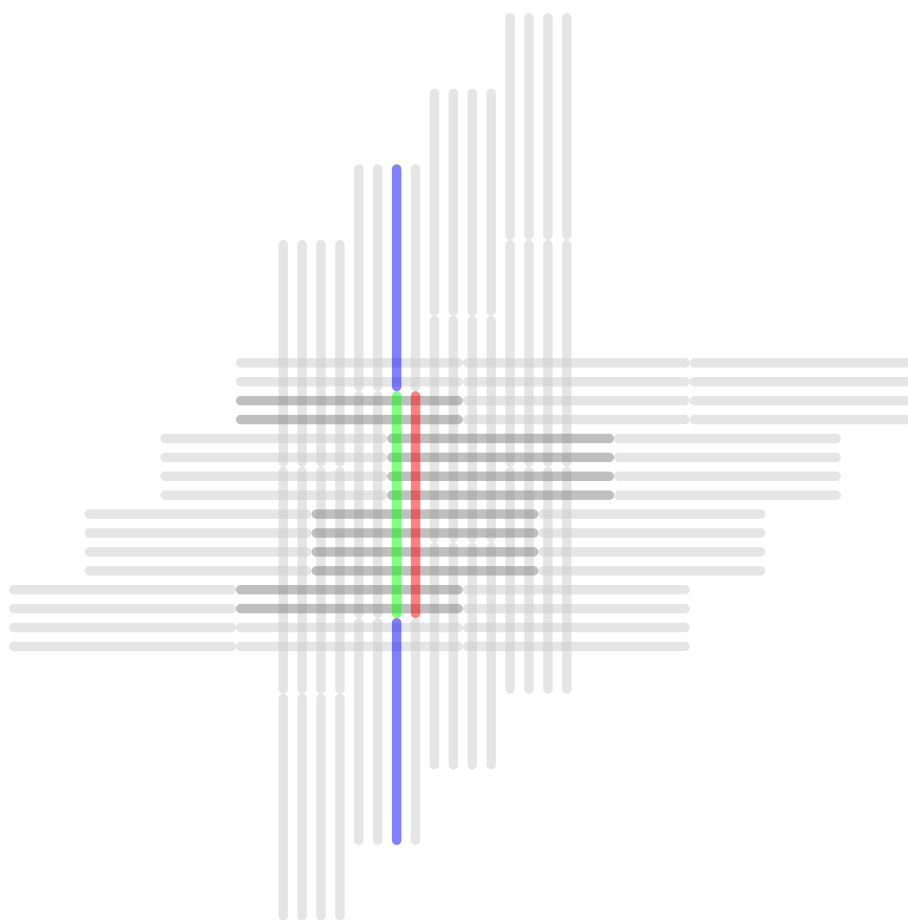


Figure 2.6: Illustration of part of the Pegasus topology, adapted from [1]

In short, each qubit is connected to other 15 qubits and the computers that implement this topology have at most 5760 qubits, and, since the qubits in the margin of the chip of the QPU are not as connected as those in the middle, the computers have at most 5640 qubits with 15 couplers [1]. However, there is a high probability of having defects in the chip, and thus the total number of qubits may be lower. This means that, in the case of state-of-the-art D-Wave computers, the theoretical maximum size of problems that they can solve is 180 variables, since we can directly map a fully connected graph of 180 nodes [14].

D-Wave computers can be accessed via D-Wave's Leap platform. At the time of writing,

new users, upon registering, obtain a free minute of access time to their QPU as a trial time, for one month. After spending their trial access time, or after having passed a month, users can agree to open-source any software they develop using the Leap platform, and obtain an additional minute of QPU access time per month.

In order to input problems to D-Wave QPU, we have to transform the problem to one of two binary quadratic models: Ising model and Quadratic Unconstrained Binary Optimization (QUBO). A Binary Quadratic Model (BQM) is a representation of problems that has no constraints and uses quadratic and linear coefficients that represent the coupler links and the programmable bias, respectively. These models will be explained in depth in Section 2.5. To perform those transformations, and to execute, solve and tune problems, D-Wave provides a framework called Ocean Tools. Appendix B shows an implementation of a problem on a D-Wave computer, as well as its execution.

In the following paragraphs, we present the D-Wave system's parameters that are studied in our work. These parameters influence the evolution of the QPU when solving a problem.

**Embedding** The embedding parameter represents the mapping used to translate the problem to solve to the topology of the QPU [15]. The QPU follows a topology that is not a fully connected graph, with each qubit connected to only a part of the remaining qubits. This characteristic makes it impossible to directly map the majority of the problems onto the QPU. Therefore, we need to map each of the binary variables of the problem to a set of qubits in the QPU. In this sense, a *chain* is a set of qubits associated to a single variable of the problem. At the end of the execution of the problem, each chain in the QPU should, in theory, have all its qubits in the same state. In practice, however, this does not always happen, and some of the qubits may be in a different state. This situation is called *chain break* and is, by default, solved with a majority voting algorithm. Figure 2.7 illustrates the embedding of a BQM problem with three fully connected binary variables on a QPU with four qubits, each connected to other two, forming a square topology.

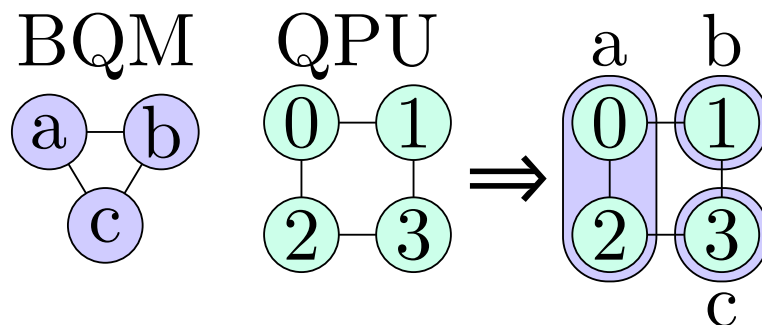


Figure 2.7: Illustration of an embedding of a BQM problem on a QPU

In the illustration, the variable  $a$  is mapped to qubits 0 and 2, while the variables  $b$  and  $c$  are mapped to qubits 1 and 3, respectively. After executing the problem, qubits 0 and 2 should end in the same state. The embedding is very important to minimize the number of chain breaks when executing a problem.

**Chain strength** The chain strength parameter defines the strength of the couplings between qubits. A higher strength reduces the number of chain breaks that might occur in a chain. Note that setting too strong couplings between qubits is not beneficial, since it scales down the bias of each qubit in comparison to the couplings between qubits, degrading the definition of the problem [15, 16].

**Number of reads** The number of reads parameter sets the number of solutions to read when executing a problem. Due to the non-deterministic nature of D-Wave computers, a single read is not sufficient for most problems, and several reads are performed, returning a set of solutions for each execution [15].

**Anneal schedule** The anneal schedule parameter defines how the QPU evolves from the initial Hamiltonian to the problem Hamiltonian [15]. The schedule is defined by a series of time-current points. Each time-current point  $(t, s)$  is a pair of two numbers, one that specifies a point in time  $t$  in microseconds, and other that specifies the normalized persistent current  $s$  in the range  $[0, 1]$ . A normalized persistent current value of 0 is analogous to the system being in the initial Hamiltonian, while a value of 1 is analogous to the system being in the final Hamiltonian, and the values in-between are analogous to the system being in a transitional Hamiltonian. Given the series of time-current points, the QPU evolves the system along these points, following a piecewise-linear curve that connects them. Figure 2.8 illustrates the curve that is followed by the system for a series of four time-current points:  $(0\mu s, 0)$ ,  $(10\mu s, 0.5)$ ,  $(90\mu s, 0.5)$ , and  $(100\mu s, 1)$ . Using this series, the QPU evolves the Hamiltonian with a pause of  $80\mu s$  at the middle of the evolution.



Figure 2.8: Illustration of a piecewise-linear curve followed by a QPU

**Annealing system** D-Wave provides two types of quantum computers, according to their topology [15]. The Advantage systems are the most recent ones, following the Pegasus topology. At the time of writing, the Advantage system accessible for open-source users, `Advantage_system1.1`, has 5436 working qubits. The D-Wave 2000Q are the other type, following an older topology, called Chimera. At the time of writing, the D-Wave 2000Q system accessible for open-source users, `DW_2000Q_6`, has 2041 working qubits. Moreover, in the older topology, Chimera, each qubit only has four internal couplers and two external couplers associated, and thus each qubit is only connected to other six qubits.

In the short-term, adiabatic quantum computers such as D-Wave's are more promising than gate-based quantum computers such as IBM's, since they accept significantly bigger problems (180 variables versus 7 variables). For this reason, we will be using D-Wave computers in our work, studying the impact that the different parameters identified have on the quality of the solutions for a specific problem.

## 2.5 Combinatorial Optimization

In this section, we present a specific case of optimization problems, *combinatorial optimization* problems. Mathematical optimization is the field of research that studies and develops techniques for finding the maximum or minimum of an objective function, given a set of constraints — i.e. optimization problems [17]. For example, let us suppose that we have a function that tells us the amount of power generated by a hydroelectric power station, or the failure rate of a vehicle, or other dependent variables of significant interest. In such cases, it is in our best interest to find the set of values that give us the best possible value from the function. For example, in the case of the power station, one of the variables is the rate of water passing by the generators, which is a continuous variable, inserting the problem in the subset of *continuous optimization*.

The objective function may have constraints that limit the variables. For example, one possible constraint for the power station problem is that the amount of water in the reservoir cannot exceed a specific volume. In practice, optimization problems have a high number of variables and constraints such that it becomes infeasible to simulate every possible combination of values, due to classical computing not being able to solve these problems in polynomial time. The general definition of a continuous optimization problem is as follows:

$$\begin{aligned} & \min f(x) \\ & \text{subject to } g(x) \geq 0 \\ & \quad h(x) = 0 \\ & \text{and } x \in \mathbb{R} \end{aligned} \tag{2.24}$$

Here,  $f(x)$  is the objective function and  $g(x)$  and  $h(x)$  are the inequality and equality constraints, respectively. Note that the objective function can also be maximized, since it is equivalent to minimizing  $-f(x)$ .

Among the set of optimization problems, there is a subset tackled by *discrete optimization*, where the variables are restricted to be *discrete variables*, i.e. that have a discrete set of values, such as the binary values 0 and 1 or integer values [17]. For example, if we defined  $x$  in expression 2.24 such that  $x \in \{0, 1\}^n$ , the optimization problem would fall under the subset of discrete optimization problems. The thesis work focuses on a problem that inserts in one of the branches of discrete optimization, *combinatorial optimization*.

In combinatorial problems, we are seeking for the best object from a finite set of objects, such as integers, permutations, and graphs. That is, we are finding the optimal *combination* or *configuration* of variables that minimizes the objective function. An example is the 0–1 knapsack problem — let us suppose that we have a knapsack where we intend to store the max amount of value in objects while keeping the total weight under a fixed limit, such that:

$$\begin{aligned} & \max \sum_{i=1}^n v_i x_i \\ & \text{subject to } \sum_{i=1}^n w_i x_i \leq W \\ & \quad x \in \{0, 1\}^n \end{aligned} \tag{2.25}$$

Here,  $n$  is the number of objects,  $x_i$  indicates whether object  $i$  goes into the knapsack,  $v_i$  is

the value of object  $i$ ,  $w_i$  is the weight of object  $i$ , and  $W$  is the maximum limit of the total weight. This is a model that we use as part of the means to solve this problem and we could have used others. In the following subsections, three ways to express combinatorial problems are explained, since they are of special interest for the thesis work. Afterwards, a list of the most common used classical and quantum algorithms is described.

### 2.5.1 Ising Model

The Ising model is one of the supported BQM that D-Wave computers accept, as mentioned in Section 2.4. A classical Ising model is the following quadratic objective function of a set of  $N$  spins  $s_i \in \{-1, +1\}$  [2]:

$$E_{ising}(s) = - \sum_{i=1}^N \sum_{j=i+1}^N J_{i,j} s_i s_j - \sum_{i=1}^N h_i s_i \quad (2.26)$$

Here,  $h_i$  is the magnetic field strength applied on the spin  $i$  (i.e. the bias of the variable  $i$ ),  $J_{i,j}$  is the coupling strength between spins  $i$  and  $j$ . To obtain the quantum version of the same model, we substitute the spins by Pauli-Z operators [18], which act as variables of the problem, and thus get the following Hamiltonian:

$$H = - \sum_{i=1}^N \sum_{j=i+1}^N J_{i,j} J_{i,j} \sigma_z^i \sigma_z^j - \sum_{i=1}^N h_i \sigma_z^i \quad (2.27)$$

For example, let us suppose that we want to produce one Ising model of the 0-1 knapsack problem. This process is difficult since we need to translate the constraints into terms of an unconstrained objective function. To ease this translation, we let  $H = H_A + H_B$ , where the Hamiltonian  $H_A$  ensures that the total weight limit is satisfied and the Hamiltonian  $H_B$  ensures that the total value is maximized:

$$H_A = A \left( 1 - \sum_{j=1}^W y_j \right)^2 + A \left( \sum_{j=1}^W j y_j - \sum_{i=1}^n w_i x_i \right)^2 \quad (2.28)$$

$$H_B = -B \sum_{i=1}^n v_i x_i \quad (2.29)$$

For  $H_A$ , we introduce the binary variable  $y_j$  for  $1 \leq j \leq W$ , which is 1 if the final weight of the knapsack is  $j$ , and 0 otherwise. Note that  $A$  and  $B$  are two real positive constants that are adjusted such that adding one item to the knapsack in excess of weight will always be numerical worse than the potential value gained by the item. Hence, it is important to define the condition  $0 < B \max(v_i) < A$ , such that the model is correct.

### 2.5.2 Quadratic Unconstrained Binary Optimization Formulation

Another supported BQM that D-Wave computers accept is the QUBO formulation. This formulation is actually directly translated to the Ising model in quantum computers, since

both models are isomorphic [19]. In fact, to translate we just need to do the map  $s = 2q - 1$ , with  $q \in \{0, 1\}^N$ , which leads us to the following quadratic objective function:

$$E_{qubo}(a_i, b_{i,j}; q_i) = - \sum_{i=1}^N \sum_{j=i+1}^N b_{i,j} q_i q_j - \sum_{i=1}^N a_i q_i. \quad (2.30)$$

In other words, the QUBO formulation replaces the Ising model terminology with linear  $a_i$  coefficients and quadratic  $b_{i,j}$  coefficients. Linear coefficients correspond to the bias applied to each qubit, while quadratic coefficients correspond to the coupling between qubits. It is also very common to find this formulation in a matrix form, where the coefficients are described by an upper-diagonal matrix  $Q$  of linear  $Q_{i,i}$  and quadratic  $Q_{i,k}$  real coefficients.

### 2.5.3 Graph Expression

Both the Ising model and the QUBO formulation can be represented by graphs, where nodes represent the binary variables, node weights represent the biases, and edge weights represent the coupling strengths. For example, if we have a QUBO formulation represented by the following upper-diagonal matrix:

$$Q = \begin{bmatrix} 1 & 7 & 3 & 1 \\ & 2 & 7 & 0 \\ & & 8 & 4 \\ & & & 9 \end{bmatrix} \quad (2.31)$$

Then the following graph expresses the problem formulated by this matrix QUBO:

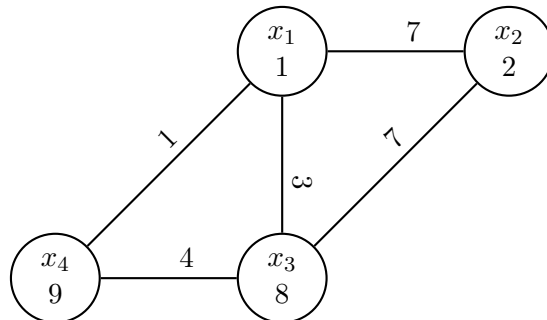


Figure 2.9: Graph expression of a QUBO formulated problem

### 2.5.4 Summary of Problem Expressions

In the end, we can express the problem in several different, but equivalent, ways. Table 2.2 shows the equivalence between them.

Expression	Variable	States	Linear coefficient	Quadratic coefficient
QPU	Qubit state	$\{s^\uparrow, s^\downarrow\}$	Qubit bias	Coupling strength
Ising	$s_i$	$\{-1, 1\}$	$h_i$	$J_{i,j}$
Scalar QUBO	$q_i$	$\{0, 1\}$	$a_i$	$b_i$
Matrix QUBO	$x_i$	$\{0, 1\}$	$Q_{i,i}$	$Q_{i,j}$
Graph	Node		Node weight	Edge weight

Table 2.2: Table of equivalence between QPU terminology, Ising models, QUBO formulations, and graph expressions

All the expressions can be directly converted between each other. Nonetheless, there are some advantages in choosing a particular expression, depending on the problem that is being tackled. For example, the QUBO formulation is more suited for problems where we need to choose a combination of elements among a set, since we can express this combination as binary variables with the value 0 meaning “element not chosen” and with the value 1 meaning “element chosen”.



This page is intentionally left blank.

## Chapter 3

# Portfolio Optimization

In this chapter, we introduce the Portfolio Optimization Problem (POP), as well as the concepts that are relevant to understand the motivation and goal of this optimization problem. We also present two additional formulations of the POP, one for Linear Programming and other for QUBO. The chapter is outlined as follows:

- **Section 3.1** introduces the terminology required to understand the POP, as well as the motivation behind this optimization problem.
- **Section 3.2** presents the POP, its parameters in the context of our work, as well as a QUBO formulation.
- **Section 3.3** introduces an overview on the state of the art in quantum computing applied to POP.

### 3.1 Terminology and Motivation

In the context of financial markets, an asset is an item of value owned by an individual or a company [20]. This item can be any tangible or intangible resource that could produce an economic value. An investment is the purchase of a resource as a whole or parts thereof to achieve long-term returns. To support their decisions, investors draw their attention to several market factors, such as the asset's intrinsic value, the company's earnings, or industry trends. A portfolio is the collection of investments of an individual or a company. In light of the above, investors aim to select the best possible portfolio — by definition, this consists of the POP [7].

### 3.2 The Portfolio Optimization Problem

Our work consists on tackling the combinatorial application of the POP, in which investors seek to allocate capital to a subset of a universe of assets. Concretely, our work solves the following combinatorial problem:

$$\begin{aligned} \min_{x \in \{0,1\}^n} \quad & qx^T \Sigma x - \mu^T x \\ \text{subject to} \quad & 1^T x = B \end{aligned} \tag{3.1}$$

where  $x$  is the portfolio, which is a vector of binary decision variables that indicate which assets are selected ( $x_i = 1$ ) and which are not ( $x_i = 0$ );  $n$  is the universe size, which indicates the number of assets to select from ( $n \in \mathbb{N}$ );  $q$  is the risk appetite of the investor ( $q \in \mathbb{R}_{\geq 0}$ );  $\Sigma$  is the matrix of the covariance between assets ( $\Sigma \in \mathbb{R}^{n \times n}$ );  $\mu$  is the vector of the expected returns for the assets ( $\mu \in \mathbb{R}^n$ ); and  $B$  is the budget size, which indicates the number of assets to select ( $B \in \{1, \dots, n\}$ ). Note that this objective function is a weighted sum of two individual objective functions, the expected return ( $\mu^T x$ ) and the volatility ( $x^T \Sigma x$ ).

### 3.2.1 Multiobjective Optimization and Efficient Frontier

The risk appetite  $q$  of the investor, one of the parameters of POP, is related with the multiobjective nature of this combinatorial problem. This means that, for nontrivial cases, there is no single solution that maximizes the expected return and minimizes the volatility, two terms that can be considered as two individual objective functions. In this sense, the investor has to decide the trade-off between the expected return and the volatility. For any trade-off, the investor should choose a solution whose associated point in the objective space belongs to the efficient frontier, a set of points in which the value of one of the objective functions cannot be improved without degrading the value of the other objective function. These points are said to be *non-dominated*, since there is no other single point that has a better value for both objective functions.

We will introduce an example to illustrate the POP and the efficient frontier. Let us assume that we have a universe of 4 assets, and a budget size of 2. Thus, we have  $n = 4$  and  $B = 2$ , and the expected return of the assets and the covariance matrix are:

$$\begin{aligned} \mu &= \begin{bmatrix} 0.5 \\ -0.75 \\ 1 \\ 1.5 \end{bmatrix}, \\ \Sigma &= \begin{bmatrix} 0.333 & -0.167 & 0.667 & 0 \\ -0.167 & 0.917 & -0.333 & 0.833 \\ 0.667 & -0.333 & 1.333 & 0 \\ 0 & 0.833 & 0 & 1.667 \end{bmatrix}. \end{aligned} \tag{3.2}$$

In this case, there are 6 feasible solutions or portfolios, which is the same number as the number of pairs that exists in a set of 4 assets. Table 3.1 shows the expected return and the volatility of each of these solutions. Figure 3.1 shows the scatter plot of the points associated to each of the feasible solutions in the objective space.

The scatter plot enables us to visually identify the efficient frontier, which is the subset containing the points associated to solutions  $\mathcal{P}_1$ ,  $\mathcal{P}_3$ ,  $\mathcal{P}_4$ , and  $\mathcal{P}_6$ , shown in green. These solutions are denoted *efficient solutions*. The dominated points are shown in red and, in the context of POP, we should never select the portfolios associated to these points,  $\mathcal{P}_2$  and  $\mathcal{P}_5$ , since they both are dominated by portfolios  $\mathcal{P}_3$  and  $\mathcal{P}_6$ . Note that the solution  $\mathcal{P}_1$  has a negative expected return, which means that this portfolio, in practice, should not be selected, despite its associated point being part of the efficient frontier.

When solving the POP, we consider the weighted sum formulation for different values of

Table 3.1: Expected return and volatility of each of the feasible solutions of example POP

<i>Portfolio ID</i>	$\mathbf{x}$	<i>Expected return</i>	<i>Volatility</i>
$\mathcal{P}_1$	[1 1 0 0]	-0.250	0.917
$\mathcal{P}_2$	[1 0 1 0]	1.500	3.000
$\mathcal{P}_3$	[1 0 0 1]	2.000	2.000
$\mathcal{P}_4$	[0 1 1 0]	0.250	1.583
$\mathcal{P}_5$	[0 1 0 1]	0.750	4.250
$\mathcal{P}_6$	[0 0 1 1]	2.500	3.000

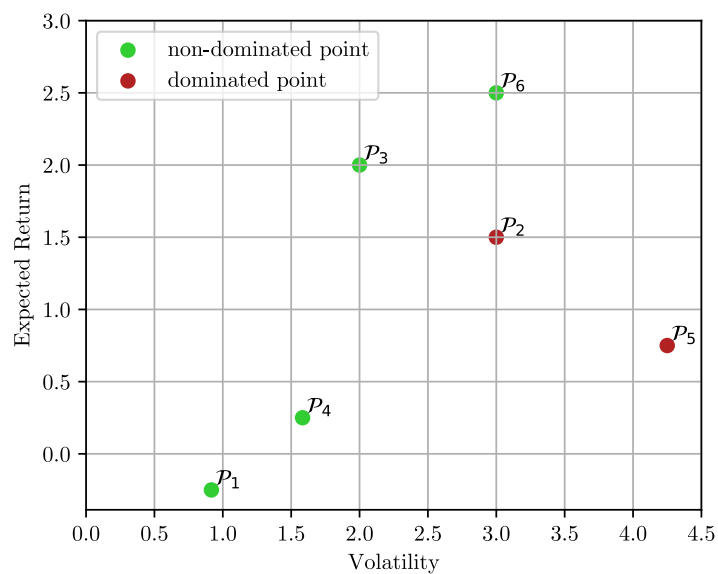


Figure 3.1: Scatter plot of points associated to each of the feasible solutions of example POP

$q$ , such that we obtain a set of solutions that encompasses different risk appetites, which keeps our work suited to any investor profile, whether they are risk-prone, risk-averse, or in-between. That is, we try to obtain a frontier that ranges from solutions with the lowest volatility to solutions with the highest volatility, providing different options for investors to choose from.

### 3.2.2 Parameters

In the context of our work, we look at the POP while taking into account the different trade-offs investors may choose. In this sense, we present a different set of parameters that we will refer to hereinafter. The set will contain most of the parameters already presented in Expression 3.1, and new ones that will directly or indirectly influence the remaining.

**Universe size** The universe size, denoted  $n$ , is the number of assets that constitute the universe of the problem instance. Larger values of  $n$  translate to larger instances, since

larger universes have a larger number of subsets to select.

**Budget size** The budget size, denoted  $B$ , is the number of assets to be selected among the universe of the problem instance. In other words, it is the size of the portfolio to be selected.

**Directions** The directions parameter is related to the risk appetite  $q$  of the formulation. Since  $q$  represents the trade-off between the expected return and the volatility of the investor, it can be viewed as a pointer of the direction in the objective space to which the solver targets. Hereinafter, when saying direction, we refer to a value for the parameter  $q$  in the POP formulation. Therefore, we provide a set of different directions to a given solver, which solves the problem instance for each direction in this set. In return, we obtain a set of solutions that encompasses different risk appetites, which keeps our work suited to any investor profile, whether they are risk-prone, risk-averse, or in-between.

The specific directions to provide to the solver are problem dependent, since it depends on the domain of both objective functions, the expected return and the volatility. To understand how to choose a set of directions, let us assume that we want a set of directions for the example shown in Table 3.1 and in Figure 3.1, such that each of the feasible solutions maximizes the objective function of the problem for one of the directions in the set. Starting with the direction 0, the solution that maximizes the objective function for this direction is  $\mathcal{P}_6$ , since it has the highest expected return. For the next solution,  $\mathcal{P}_3$ , a good direction would be 1, since, for this direction,  $\mathcal{P}_3$  minimizes the objective function, with a value of 0, while  $\mathcal{P}_6$  has a value of 0.5 and  $\mathcal{P}_4$  has a value of 1.333. Applying the same reasoning, we obtain one more direction, 3, which is minimized by  $\mathcal{P}_1$ .  $\mathcal{P}_4$  does not minimize the objective function for any direction, despite being part of the efficient frontier, due to the fact that this portfolio is not a *supported solution*. A supported solution is an efficient solution that can be obtained by solving a weighted sum problem of the objective functions [21], which is the case of portfolios  $\mathcal{P}_6$ ,  $\mathcal{P}_3$ , and  $\mathcal{P}_1$ . In the end, for this example, the set of directions that a solver needs to find the supported solutions is  $\{0, 1, 3\}$ . Note that, generally, it is not guaranteed to find an optimal solution for direction  $q = 0$ , since there may exist two or more solutions with the same expected return and different volatilities that would be found in this direction.

**Dataset type** The dataset parameter specifies which assets take part of the universe, as well as their expected return and volatility. In other words, this parameter defines the POP parameters  $\mu$  and  $\Sigma$ . Hence, different datasets can have different levels of correlation between assets, when considering their market performance.

### 3.2.3 Quadratic Unconstrained Binary Optimization formulation

We present the Quadratic Unconstrained Binary Optimization (QUBO) formulation of the POP. This formulation is important to be able to input this problem on D-Wave machines, which accept this matrix to solve the problem [22]. First, we need to move the cardinality constraint  $1^T x = B$  to the objective function:

$$\min_{x \in \{0,1\}^n} qx^T \Sigma x - \mu^T x + P(1^T x - B)^2, \quad (3.3)$$

where  $P$  is the penalization factor, whose value has to guarantee that, for any solution that is not feasible (i.e., that does not comply with the cardinality constraint), its objective function value is higher than the worst objective function value that a feasible solution can have. To understand what are the linear and quadratic coefficients of this objective function, we transform the function such that:

$$\min_{x \in \{0,1\}^n} \left( \sum_{i=1}^n \sum_{j=1}^n x_i x_j \Sigma_{i,j} q \right) - \left( \sum_{i=1}^n x_i \mu_i \right) + P \left( \left( \sum_{i=1}^n x_i \right) - B \right)^2, \quad (3.4)$$

and then transform the penalization term such that [22]:

$$P \left( \left( \sum_{i=1}^n x_i \right) - B \right)^2 = P \left( \sum_{i=1}^n x_i^2 \right) + 2P \left( \sum_{i=1}^n \sum_{j>i}^n x_i x_j \right) - 2BP \left( \sum_{i=1}^n x_i \right) + PB^2. \quad (3.5)$$

Since  $x$  is a vector of binary decision variables, we know that  $x_i^2 = x_i$ . Moreover, we can remove any constant value from the objective function. With these changes, the penalization term becomes:

$$P \left( \sum_{i=1}^n x_i \right) + 2P \left( \sum_{i=1}^n \sum_{j>i}^n x_i x_j \right) - 2BP \left( \sum_{i=1}^n x_i \right). \quad (3.6)$$

The linear and quadratic coefficients are now identified, and we can define the QUBO matrix as follows:

$$Q = q\Sigma - \begin{bmatrix} \mu_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mu_n \end{bmatrix} + PI_{n \times n} + \begin{bmatrix} 0 & P & \cdots & P \\ P & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & P \\ P & \cdots & P & 0 \end{bmatrix}_{n \times n} - 2BPI_{n \times n}, \quad (3.7)$$

where the first matrix is related to the volatility term, the second matrix is related to the expected return term, and the last three matrices are related to the penalization term. Since this QUBO matrix is symmetric, we transform it such that it becomes an upper triangular matrix that can be input on D-Wave machines to solve the POP:

$$\begin{aligned}
Q = & \begin{bmatrix} q\Sigma_{1,1} & 2q\Sigma_{1,2} & \cdots & 2q\Sigma_{1,n} \\ & \ddots & \ddots & \vdots \\ & & \ddots & 2q\Sigma_{n-1,n} \\ & & & q\Sigma_{n,n} \end{bmatrix} - \begin{bmatrix} \mu_1 & 0 & \cdots & 0 \\ & \ddots & \ddots & \vdots \\ & & \ddots & 0 \\ & & & \mu_n \end{bmatrix} + \\
& +PI_{n \times n} + \begin{bmatrix} 0 & 2P & \cdots & 2P \\ & \ddots & \ddots & \vdots \\ & & \ddots & 2P \\ & & & 0 \end{bmatrix}_{n \times n} - 2BPI_{n \times n},
\end{aligned} \tag{3.8}$$

### 3.3 Quantum Computing applied to POP

In this section, a review on the state of the art in quantum computing applied to POP is presented.

In [23], our formulation was solved with two quantum algorithms on a simulator of a quantum computer from IBM. First, they solved the problem for a universe size of 6, a budget size of 3, and a specific direction  $q$ , obtaining optimal and near-optimal solutions with both algorithms. Afterwards, they repeated the same problem without the cardinality constraint, and for different directions, solving it with one of the algorithms, obtaining a set of solutions that closely follows the efficient frontier. Our work is based on real adiabatic quantum computers from D-Wave, without resorting to a simulator, and attempts to solve the problem with a fixed budget constraint for different directions. In this sense, we believe that those findings can be complemented with the extra insight that our work can provide about real quantum hardware that otherwise would not exist with a simulator. Moreover, our work also attempts to provide a large set of options for investors that are limited to their budget, irrespective of their risk appetite.

In [24], different methods are used to solve our formulation of the POP, ranging from quantum algorithms in gate-based quantum computers to a classical-quantum strategy denoted D-Wave Hybrid, as well as to tensor networks. In [24], the risk appetite is also fixed to a single value, not accounting for different investor profiles, which is what our work attempts to do. Our work also uses a non-hybrid approach on D-Wave computers, using both Pegasus and Chimera topologies without any classical processing used whether to improve the obtained solutions or to be able to solve larger problems. Part of the findings in that work suggest that D-Wave's algorithm not only is remarkably fast, but also is capable of handling large problems, indicating that, in the short term, D-Wave computers are well suited for the purposes of our work.

In [25], a different formulation of POP is solved, again using D-Wave's hybrid algorithm. The results of this algorithm were compared with classical solvers and heuristics, considering the objective function value of their best solution. Our approach distinguishes itself from this work not only by considering a POP that takes into account different investor profiles, but also by considering a different methodology to assess the performance of D-Wave's QPU (see in Section 4.2), and also by using D-Wave's QPU without any processing to improve its solutions.

The work described in [26] has a very similar objective as ours. It tries to understand the effect that some of the QPU-based parameters that we identified in Section 2.4 have on

the results returned by the D-Wave QPU to a different formulation of POP. Concretely, the authors vary the embedding, the spin reversal (which is not studied in our work), and the annealing schedules. In the formulation used by the authors, the expected return is maximized with budget and risk constraints. The metric that they use to assess the quality of the results is a probability of success, which represents the probability of the ground truth value being returned by the QPU. Our work contrasts with this metric, since it compares an approximation frontier derived from the results of the QPU with a representation of the efficient frontier, for each problem instance (see Section 4.2). In this sense, our metric is useful to assess the performance of the QPU across different directions, which we believe to provide insight that is valuable for any investor profile. Moreover, our work also focuses on the effect of POP-related parameters. Their findings suggest little difference between *general* and *clique* embeddings (these embeddings are provided by D-Wave and are better explained in Chapter 4), and also suggest no statistically significant difference between any of the explored annealing times.

All in all, we believe that our work is a contribution that was not covered before, by studying the effect that different parameters, both POP-related and QPU-related, have on the quality of the solutions returned by a D-Wave's QPU. Moreover, we also use a metric that we believe provides a good assessment on the quality of the solutions, since it takes into account the entire efficient frontier and, thus, provides good insight for the different risk appetites an investor may assume.



This page is intentionally left blank.

# Chapter 4

## Approach and Methodology

In our work, we formulated the Portfolio Optimization Problem (POP) as a QUBO and executed several instances of this problem on D-Wave’s quantum annealer. We have considered a significant set of varying parameters from both the quantum annealer and the problem, that we hypothesize can have an impact on the quality of the solutions that one can obtain. Our motivation is then to understand in a precise way how parameters that relate not only to POP itself but also to the D-Wave system can affect and ideally improve the quality of the obtained solutions.

In this Chapter, we describe in detail the empirical study that we have designed for us to be able of achieving our goal, outlined as follows:

- **Section 4.1** describes the general structure of the empirical study.
- **Section 4.2** explains how we measure the quality of the solutions returned by the quantum annealer.
- **Section 4.3** indicates how results are analyzed with respect to statistical methods.

### 4.1 General Structure

In the course of our empirical study, we execute a number of POP instances with different parameters and assess the impact caused by these parameters. We designate as a *strategy* a POP execution with a specific set of parameters. Moreover, as a rule, each strategy is executed 10 times, in order to collect a sufficient amount of solutions to perform a statistical analysis when comparing different strategies. This statistical analysis is explained in depth in Section 4.3. Generally, when assessing the influence of a specific parameter, we are performing a comparison between strategies that are identical except on that specific parameter. We designate as a *scenario* the set of very similar strategies that are compared in order to assess the influence of a particular parameter. Hereinafter the scenarios are identified in the form  $\mathcal{S}_N$ , where  $N$  is a number, and the strategies are identified in the form  $\mathcal{S}_{Nz}$ , where  $z$  is an alphabet letter and  $N$  is the number associated to the scenario to which the strategy belongs. Based on the list of parameters related to both POP and to D-Wave’s system, we designed a series of scenarios whose execution will help us to understand which parameters have a significant impact on the quality of the obtained solutions.

### 4.1.1 Universe Size

In this section, we study the scenario in which we vary the universe size parameter (see in Section 3.2.2), designated as  $\mathcal{S}_1$ . Our goal is to understand how the universe size affects the quality of the obtained solutions. For this, we consider four different strategies for universe sizes 8, 16, 32, and 64, respectively. The reasoning behind those numbers is explained in depth in Section 4.2.2.

Table 4.1: Values of parameters for each strategy in  $\mathcal{S}_1$

Parameter	$\mathcal{S}_{1a}$	$\mathcal{S}_{1b}$	$\mathcal{S}_{1c}$	$\mathcal{S}_{1d}$
Universe size $n$	8	16	32	64
Chain strength		default		
Number of reads		$1000 \times  \text{directions} $		
Directions		<i>minimal set</i>		
Budget size $B$		$\lfloor 0.5 \times n \rfloor$		
Embedding		<i>general</i>		
Anneal schedule		<i>standard</i>		
Dataset		<i>industry diversified</i>		
Annealing system		<i>Pegasus</i>		

Table 4.1 lists the four strategies that are part of this scenario, as well as the values for their parameters. We have considered the default chain strength established by the D-Wave system, which is calculated via the *uniform\_torque\_compensation* function (see 2<sup>nd</sup> row of Table 4.1). We have configured the quantum annealer to read 1000 solutions for each direction specified (see Section 3.2.2). When executing a strategy, we are sending a queue of different QUBO matrices to the D-Wave system, one for each direction specified. Therefore, we need to specify a number of reads to the D-Wave system to carry out for each QUBO matrix. We decided on 1000 reads, since it provides a good probability of finding good solutions to the problem (see 3<sup>rd</sup> row of Table 4.1). We have set the directions to the set with the minimum number of directions that leads the MILP solver to the same representation of the efficient frontier. This set will be explained in depth in Section 4.1.4. Hereinafter, for brevity, we define this set as *minimal set* in the context of directions (see 4<sup>th</sup> row of Table 4.1). Concretely, for  $n = 8$ , the set is  $\{0, 11, 20, 54\}$ ; for  $n = 16$ , the set is  $\{0, 2, 6, 100, 500\}$ ; for  $n = 32$ , the set is  $\{0, 0.4, 0.9, 2, 3, 9, 100\}$ ; and for  $n = 64$ , the set is  $\{0, 0.2, 0.4, 0.6, 1.1, 1.3, 1.5, 2, 5, 6, 7, 8, 10, 100, 500\}$ . We have set the budget size to half of the universe size,  $\lfloor 0.5 \times n \rfloor$  (see 5<sup>th</sup> row of Table 4.1). We have set the embedding parameter to the *general* embedding provided by D-Wave’s *minorminer* package (see 6<sup>th</sup> row of Table 4.1). According to this package, the *general* embedding aims to be useful for any type of problem. We have set the anneal schedule parameter to the default schedule according to the D-Wave system, which is hereinafter designated *standard*. This schedule evolves the initial Hamiltonian to the problem Hamiltonian in  $20\mu s$ , in a way such that it follows a piecewise-linear curve with two time-current points, one at  $(0\mu s, 0)$  and the other at  $(20\mu s, 1)$ . We have set the dataset parameter to the *industry diversified* dataset (see 8<sup>th</sup> row of Table 4.1). This dataset is generated by choosing assets in such a way that the assets cover the maximum amount of different industry sectors. This method is explained in depth in Section 4.1.8. The annealing system used by the strategies is the

*Pegasus* system (see 9<sup>th</sup> row of Table 4.1).

### 4.1.2 Chain Strength

In this section, we study the scenario in which we vary the chain strength parameter (see in Section 2.4), designated as  $\mathcal{S}_2$ . For this, we consider 13 different strategies not only for the default value for this parameter but also for a range of values from  $0.125M_Q$  to  $1.5M_Q$ , with incrementing steps of 0.125.  $M_Q$  corresponds to the element with the maximum absolute value in the QUBO matrix of the problem instance.

Table 4.2: Values of parameters for each strategy in  $\mathcal{S}_2$

Parameter	$\mathcal{S}_{2a}$	$\mathcal{S}_{2b} \cdots \mathcal{S}_{2m}$
Universe size $n$		16, 32, and 64
Chain strength	default	$0.125M_Q$ to $1.500M_Q$
Number of reads		$1000 \times  \text{directions} $
Directions		<i>minimal set</i>
Budget size $B$		$\lfloor 0.5 \times n \rfloor$
Embedding		<i>general</i>
Anneal schedule		<i>standard</i>
Dataset		<i>industry diversified</i>
Annealing system		<i>Pegasus</i>

Table 4.2 lists the 13 strategies that are part of this scenario, as well as the values for their parameters. This scenario will be an exception compared to the other scenarios, when considering not only the number of executions, but also the statistical analysis of its results. Due to the large number of strategies, strategies  $\mathcal{S}_{2b}$  to  $\mathcal{S}_{2m}$  are each executed three times, while  $\mathcal{S}_{2a}$  is executed 12 times. We are using results from previous scenarios as part of the 12 executions of  $\mathcal{S}_{2a}$ , hence the number. The results will be compared with line charts, instead of the boxplots and statistical tests that are used for all the other scenarios explained in depth in Section 4.3.

Regarding the idea that different strategies share the same values for all but one of their parameters, there is another parameter that is executed with more than one value: the universe size parameter. In our work, every strategy is always executed for all the specified universe sizes. For example, in this scenario,  $\mathcal{S}_{2a}$  is executed 12 times for each of the following values of  $n$ : 16, 32, and 64, while strategies  $\mathcal{S}_{2b}$  to  $\mathcal{S}_{2m}$  are executed 3 times for each of the same values of  $n$ .

The universe size 8 will not be included for this and the following scenarios, such that we can avoid expending time from our budget. More concretely, the execution of a single strategy from this scenario for universe size 8 takes around  $423ms$  (see Section 2.4).

### 4.1.3 Number of Reads

In this section, we study the scenario in which we vary the number of reads (see in Section 2.4), designated as  $\mathcal{S}_3$ . Our goal is to understand how the number of reads affects the

quality of the obtained solutions. For this, we consider two different strategies, one that reads 1000 solutions per direction, and other that reads a total of 15000 solutions.

Table 4.3: Values of parameters for each strategy in  $\mathcal{S}_3$ 

Parameter	$\mathcal{S}_{3a}$	$\mathcal{S}_{3b}$
Universe size $n$	16, 32, and 64	
Chain strength	$1.000M_Q$	
Number of reads	$1000 \times  \text{directions} $	15000
Directions	<i>minimal set</i>	
Budget size $B$	$\lfloor 0.5 \times n \rfloor$	
Embedding	<i>general</i>	
Anneal schedule	<i>standard</i>	
Dataset	<i>industry diversified</i>	
Annealing system	<i>Pegasus</i>	

Table 4.3 lists the two strategies that are part of this scenario, as well as the values for their parameters. The total of 15000 solutions that are read as part of  $\mathcal{S}_{3b}$  is equally distributed among the directions specified for that execution. For example, if an execution has 10 directions, then each direction will have 1500 solutions read. To ensure that different universe sizes read the same number of solutions, the remaining scenarios will read 15000 solutions in each execution of their strategies.

#### 4.1.4 Directions

In this section, we study the scenario in which we vary the directions (see in Section 3.2.2), designated as  $\mathcal{S}_4$ . For this, we consider four different strategies, one for the smallest set of directions that leads the MILP solver to the same representation of the efficient frontier, other for a set with a low number of directions (6), another for a set with an intermediate number of directions (15), and one more for a set with a high number of directions (30).

Table 4.4 lists the four strategies that are part of this scenario, as well as the values for their parameters. Regarding the *minimal set* of directions, this set is obtained with the help of the MILP solver. This solver is used to obtain a representation of the efficient frontier of a given dataset. As part of this obtainment, the solver solves the problem instance for a large range of directions. For each direction, the solver returns the optimal portfolio. In the end, we obtain a representation of the efficient frontier. For each solution in this representation, we select the direction associated to it with the lowest numerical value. The set of the selected directions is the *minimal set*. This process of solving a problem instance to optimality is described in depth in Section 4.2.2.

Concretely, the set used in  $\mathcal{S}_{4b}$ , with a low number of directions, has the following values for  $q$ : 0, 0.1, 1, 10, 100, and 1000. The set used in  $\mathcal{S}_{4c}$ , with an intermediate number of directions, has the following values for  $q$ : 0, 0.2, 0.4, 0.6, 0.8, 1, 2, 3, 4, 5, 7.5, 10, 50, 100, and 1000. The set used in  $\mathcal{S}_{4d}$ , with a high number of directions, has the following values for  $q$ : 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, and 1000.

Table 4.4: Values of parameters for each strategy in  $\mathcal{S}_4$ 

Parameter	$\mathcal{S}_{4a}$	$\mathcal{S}_{4b}$	$\mathcal{S}_{4c}$	$\mathcal{S}_{4d}$
Universe size $n$		32 and 64		
Chain strength		$1.000M_Q$		
Number of reads		15000		
Directions	<i>minimal set</i>	6	15	30
Budget size $B$		$[0.5 \times n]$		
Embedding		<i>general</i>		
Anneal schedule		<i>standard</i>		
Dataset		<i>industry diversified</i>		
Annealing system		<i>Pegasus</i>		

The universe size 16 will not be included for this and the next scenarios, such that we can avoid expending time from our budget. More concretely, the execution of  $\mathcal{S}_{4d}$  for universe size 16 takes around  $1864ms$ .

#### 4.1.5 Budget

In this section, we study the scenario in which we vary the budget size parameter (see in Section 3.2.2), designated as  $\mathcal{S}_5$ . For this, we consider three different strategies for budget sizes  $[0.2 \times n]$ ,  $[0.5 \times n]$ , and  $[0.8 \times n]$ , respectively.

 Table 4.5: Values of parameters for each strategy in  $\mathcal{S}_5$ 

Parameter	$\mathcal{S}_{5a}$	$\mathcal{S}_{5b}$	$\mathcal{S}_{5c}$
Universe size $n$		32 and 64	
Chain strength		$1.000M_Q$	
Number of reads		15000	
Directions		15	
Budget size $B$	$[0.2 \times n]$	$[0.5 \times n]$	$[0.8 \times n]$
Embedding		<i>general</i>	
Anneal schedule		<i>standard</i>	
Dataset		<i>industry diversified</i>	
Annealing system		<i>Pegasus</i>	

Table 4.4 lists the three strategies that are part of this scenario, as well as the values for their parameters. In this scenario, the directions are the same as in  $\mathcal{S}_{4c}$ .

### 4.1.6 Embedding

In this section, we study the scenario in which we vary the embedding parameter (see in Section 2.4), designated as  $\mathcal{S}_6$ . For this, we consider three different strategies for embeddings *general*, *clique*, and *layout*, respectively.

Table 4.6: Values of parameters for each strategy in  $\mathcal{S}_6$

Parameter	$\mathcal{S}_{6a}$	$\mathcal{S}_{6b}$	$\mathcal{S}_{6c}$
Universe size $n$		16, 32, and 64	
Chain strength		$1.000M_Q$	
Number of reads		15000	
Directions		<i>minimal set</i>	
Budget size $B$		$[0.5 \times n]$	
Embedding	<i>general</i>	<i>clique</i>	<i>layout</i>
Anneal schedule		<i>standard</i>	
Dataset		<i>industry diversified</i>	
Annealing system		<i>Pegasus</i>	

Table 4.6 lists the three strategies that are part of this scenario, as well as the values for their parameters. Embeddings *clique* and *layout* are both provided in D-Wave’s `minorminer` package. According to this package, the *clique* embedding is targeted to problems whose graph is a clique, while the *layout* embedding is targeted to problems whose graph has nodes of low degrees or its underlying data is spatial.

### 4.1.7 Anneal Schedule

In this section, we study the scenario in which we vary the anneal schedule parameter (see in Section 2.4), designated as  $\mathcal{S}_7$ . For this, we consider four different strategies for anneal schedules *standard*, *long*, *pause*, and *quench*, respectively.

Table 4.7 lists the four strategies that are part of this scenario, as well as the values for their parameters. Anneal schedule *long* is similar to the default schedule, but instead of taking a total of  $20\mu s$ , it takes a total of  $100\mu s$ . By including this schedule in the analysis, we can understand the impact of a significantly longer anneal schedule on the quality of the obtained solutions. Concretely, this schedule evolves the initial Hamiltonian to the problem Hamiltonian in  $100\mu s$ , in a way such that it follows a piecewise-linear curve with two time-current points, one at  $(0\mu s, 0)$  and the other at  $(100\mu s, 1)$ . Anneal schedule *pause* is a schedule that pauses the evolution of the initial Hamiltonian to the problem Hamiltonian. In our case, we are introducing a  $100\mu s$  pause in the middle point of the default schedule. By including this schedule in the analysis, we can understand the impact of a long pause in the schedule on the quality of the obtained solutions. Concretely, this schedule follows a piecewise-linear curve with four time-current points, the first at  $(0\mu s, 0)$ , the second at  $(10\mu s, 0.5)$ , the next at  $(110\mu s, 0.5)$ , and the last at  $(120\mu s, 1)$ . Anneal schedule *quench* is a schedule that abruptly terminates the evolution of the initial Hamiltonian to the problem Hamiltonian. In our case, we are introducing a quench in the middle point of the default schedule, which evolves the remaining schedule in  $2\mu s$ . By including this schedule in the

Table 4.7: Values of parameters for each strategy in  $\mathcal{S}_7$ 

Parameter	$\mathcal{S}_{7a}$	$\mathcal{S}_{7b}$	$\mathcal{S}_{7c}$	$\mathcal{S}_{7d}$
Universe size $n$		16, 32, and 64		
Chain strength		$1.000M_Q$		
Number of reads		15000		
Directions		<i>minimal set</i>		
Budget size $B$		$\lfloor 0.5 \times n \rfloor$		
Embedding		<i>layout</i>		
Anneal schedule	<i>standard</i>	<i>long</i>	<i>quench</i>	<i>pause</i>
Dataset		<i>industry diversified</i>		
Annealing system		<i>Pegasus</i>		

analysis, we can understand the impact of a quench in the schedule on the quality of the obtained solutions. Concretely, this schedule follows a piecewise-linear curve with three time-current points, the first at  $(0\mu s, 0)$ , the second at  $(10\mu s, 0.5)$ , and the last at  $(12\mu s, 1)$ .

#### 4.1.8 Dataset

In this section, we study the scenario in which we vary the dataset parameter (see in Section 3.2.2), designated as  $\mathcal{S}_8$ . Our goal is to understand how the dataset affects the quality of the obtained solutions. For this, we consider four different strategies for datasets *diversified*, *correlated*, *industry diversified*, and *industry correlated*, respectively.

 Table 4.8: Values of parameters for each strategy in  $\mathcal{S}_8$ 

Parameter	$\mathcal{S}_{8a}$	$\mathcal{S}_{8b}$	$\mathcal{S}_{8c}$	$\mathcal{S}_{8d}$
Universe size $n$		16, 32, and 64		
Chain strength		$1.000M_Q$		
Number of reads		15000		
Directions		<i>minimal set</i>		
Budget size $B$		$\lfloor 0.5 \times n \rfloor$		
Embedding		<i>layout</i>		
Anneal schedule		<i>standard</i>		
Dataset	<i>diversified</i>	<i>correlated</i>	<i>industry diversified</i>	<i>industry correlated</i>
Annealing system		<i>Pegasus</i>		

Table 4.8 lists the four strategies that are part of this scenario, as well as the values for their parameters. When choosing assets from a market index to constitute a dataset, we know which industry sector they are associated with, as well as their volatility and expected return, among other information. This information can be leveraged to generate a dataset



with a particular structure or pattern.

The *industry diversified* dataset is generated by choosing assets from as many industry sectors as possible. For example, let us assume that we wanted to generate a dataset of 8 assets and that we have a market index of 64 assets, with 8 industry sectors and 8 assets associated to each. We would choose 1 asset from each industry sector, such that we would end with a dataset of 8 assets, each from a different industry sector.

The *industry correlated* dataset is generated by choosing assets from as few industry sectors as possible. For example, let us assume that we wanted to generate a dataset of 8 assets and that we have a market index of 64 assets, with 8 industry sectors and 8 assets associated to each. We would choose 8 assets from a single industry sector, such that we would end with a dataset of 8 assets, all from the same industry sector.

The *diversified* dataset is generated by choosing assets in such a way that they are as little statistically correlated as possible, when considering their market performance. For example, let us assume that we wanted to generate a dataset of 8 assets and that we have a market index of 64 assets. We would choose the 8 assets from this index that have the minimum sum of pairwise correlation among all the possible subsets of 8 assets, when considering their market performance.

The *correlated* dataset is generated by choosing assets in such a way that they are as much statistically correlated as possible, when considering their market performance. For example, let us assume that we wanted to generate a dataset of 8 assets and that we have a market index of 64 assets. We would choose the 8 assets from this index that have the maximum sum of pairwise correlation among all the possible subsets of 8 assets, when considering their market performance.

#### 4.1.9 Annealing System

In this section, we study the scenario in which we vary the annealing system parameter (see in Section 2.4), designated as  $\mathcal{S}_9$ . For this, we consider two different strategies for annealing systems *Pegasus* and *Chimera*, respectively.

Table 4.9: Values of parameters for each strategy in  $\mathcal{S}_9$

Parameter	$\mathcal{S}_{9a}$	$\mathcal{S}_{9b}$
Universe size $n$	16, 32, and 64	
Chain strength	$1.000M_Q$	
Number of reads	15000	
Directions	<i>minimal set</i>	
Budget size $B$	$[0.5 \times n]$	
Embedding	<i>general</i>	
Anneal schedule	<i>standard</i>	
Dataset	<i>industry diversified</i>	
Annealing system	<i>Pegasus</i>	<i>Chimera</i>

Table 4.9 lists the two strategies that are part of this scenario, as well as the values for

their parameters.

Now that we have presented the series of scenarios, we have to indicate how are the results from each strategy in a scenario evaluated with regards to their quality. The following section describes the quality indicator that we use and the method to calculate it.

## 4.2 Quality of Solutions

After executing any of the above strategies with a D-Wave system, we obtain a set of solutions, which is compared with other sets from the other strategies of the same scenario. In order to compare sets of solutions from different strategies, we measure the quality of the set with respect to a quality indicator, which will be associated to each strategy. This indicator is measured in three steps: first, an approximation frontier is identified from the results returned by an execution of a strategy; next, a representation of the efficient frontier is obtained with a MILP solver; last, both the approximation frontier and the representation of the efficient frontier are used to calculate the quality indicator.

In this Section, we introduce the method that we use to calculate an indicator of the quality of the results from a strategy. This method is described in three sections, outlined as follows:

- **Section 4.2.1** describes the step in which an approximation frontier is identified from the set of solutions returned by an execution of a strategy.
- **Section 4.2.2** explains the step in which we obtain a representation of the efficient frontier, using a MILP solver.
- **Section 4.2.3** indicates the step in which the approximation frontier and the representation of the efficient frontier are used to measure the  $\epsilon$ -indicator, an indicator of the quality of the results returned by an execution of a strategy.

### 4.2.1 Outlining the Approximation Frontier

When placing the feasible solutions (i.e., the solutions that comply with the cardinality constraint) on a chart, we get a set of points in the objective space such as the example shown in Figure 4.1. In this sense, a point is a representation of a solution in a two-dimensional objective space, where the expected return and the volatility are the two objectives.

After plotting this set of points, the approximation frontier is identified by selecting the subset of non-dominated points that have a positive expected return. Figure 4.2 shows in red the approximation frontier of the set of points from the example.

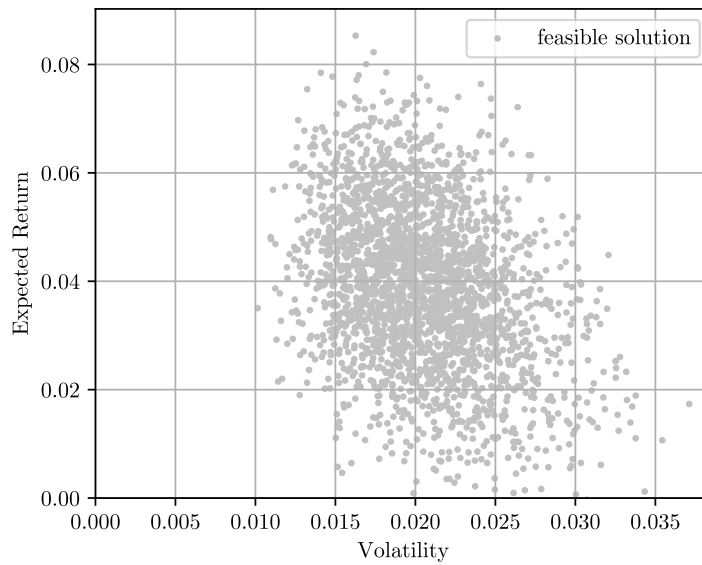


Figure 4.1: Scatter plot of points from a set of solutions returned by an execution of  $\mathcal{S}_{8a}$

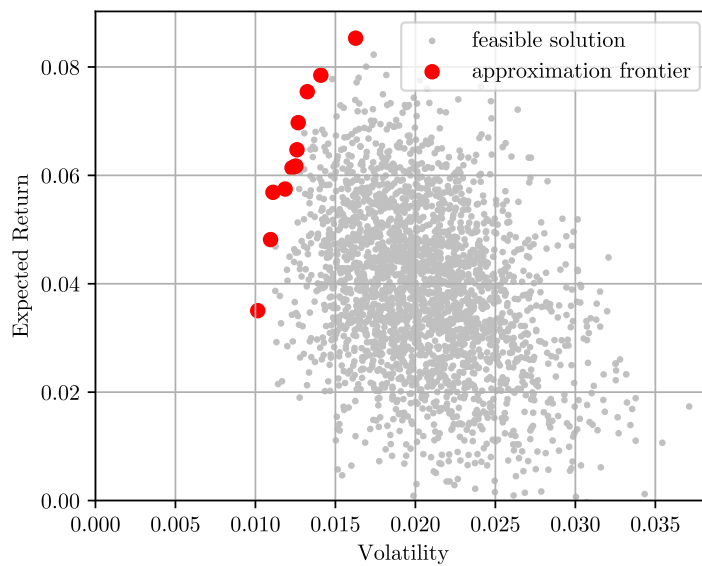


Figure 4.2: Illustration of the approximation frontier of a set of points from a set of solutions returned by an execution of  $\mathcal{S}_{8a}$

### 4.2.2 Solving the Problem Instance to Optimality

Having outlined the approximation frontier from the results of the D-Wave system for a strategy, we will compare it against a representation of the efficient frontier. With a MILP solver, we solve the problem instance to optimality for a large range of directions, such that it obtains not only the solutions associated to the extrema of the efficient frontier, but also a series of solutions associated to points between the extrema. The exact range is problem dependent, since it depends on the magnitude and range of both the expected return and the volatility of the assets (see in Section 3.2.2).

Given a range of directions, the MILP solver obtains the optimal solution associated to each direction. After obtaining all the optimal solutions, we end with a set of optimal solutions whose associated points are a subset of the efficient frontier. These associated points constitute the representation of the efficient frontier upon which the efficient frontiers obtained by executions of strategies are compared. Figure 4.3 shows in blue the representation of the efficient frontier of the problem instance from the example.

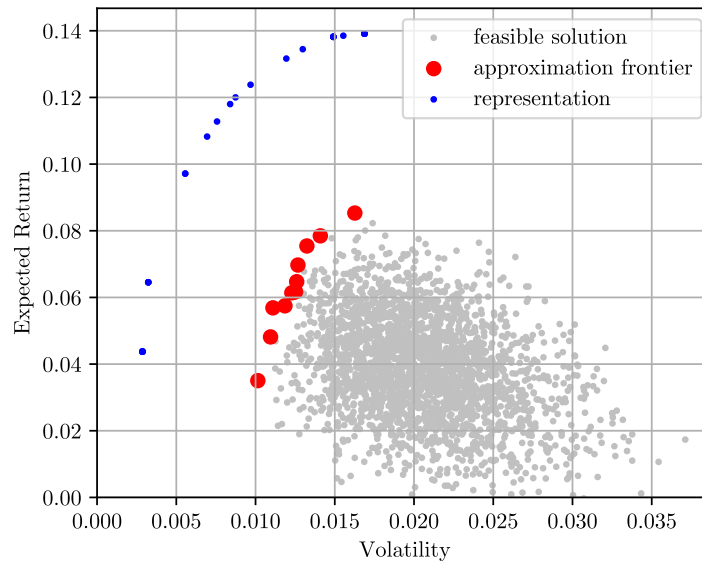


Figure 4.3: Illustration of the representation of the efficient frontier for an instance executed in  $\mathcal{S}_{8a}$

The largest universe size in which we solved instances of the POP to optimality was 64, since the MILP solver already takes over two hours for certain strategies, such as  $\mathcal{S}_{8a}$ . This is the reason why we do not explore universe sizes larger than 64 in the designed strategies and scenarios.

### 4.2.3 $\epsilon$ -indicator

The quality of the approximation frontier obtained by the execution of a strategy is measured with the  $\epsilon$ -indicator [27]. In the context of our work, the  $\epsilon$ -indicator corresponds to the smallest factor by which a set of points is multiplied in order to *weakly dominate* the efficient frontier. The smallest the  $\epsilon$ -indicator of a set of points, the closer is it to the efficient frontier, and thus the better is its quality.

We consider that a set of points  $A$  is weakly dominating another set  $B$  if every point in  $A$

is not dominated by any of the points in  $B$ . In other words, a set of points that is weakly dominating another set is at least as good as the other set, and there may exist points in it that dominate points in the other set.

Let us suppose that  $O$  is the representation of the efficient frontier, and  $S$  is the approximation frontier obtained by the execution of a strategy, then this indicator is defined as:

$$I_\epsilon(S, O) = \max_{o \in O} \min_{s \in S} \epsilon(s, o), \quad (4.1)$$

where

$$\epsilon(s, o) = \max \left( \frac{o^{\text{return}}}{s^{\text{return}}}, \frac{M_\Sigma - o^{\text{volatility}}}{M_\Sigma - s^{\text{volatility}}} \right), \quad (4.2)$$

which implies that  $\epsilon(s, o)$  is the smallest factor such that  $o$  is weakly dominated by  $\epsilon(s, o) \cdot s$ . That is, such that  $\epsilon(s, o) \cdot s$  is guaranteed to be at least as good as  $o$  when considering both their objectives.  $M_\Sigma$  is an upper bound on the volatility.

Since we are minimizing volatility (as opposed to return, which is maximized), we transformed the volatility in the formulation of the  $\epsilon$ -indicator such that it is evaluating a bi-objective maximization problem that is equivalent to the original formulation. In the end, instead of minimizing volatility, we are actually maximizing the distance between the solution's volatility  $o^{\text{volatility}}$  and the upper bound on volatility  $M_\Sigma$ . Put differently, we want to keep the volatility as far as possible from an upper bound, and thus minimize it.

The upper bound on volatility,  $M_\Sigma$ , is obtained with the MILP solver by solving the problem instance with the single objective of maximizing volatility. Once the solution is found,  $M_\Sigma$  is set to the volatility of this solution.

The  $\epsilon$ -indicator is used to assess the quality of the approximation frontiers obtained from results of strategies executed in the course of our work. It should be emphasized that an approximation frontier obtained by the execution of a strategy having  $I_\epsilon(S, O) = 1$  implies that it is a representation of the efficient frontier. Therefore, in the context of our work, this indicator has a lower bound of 1, which means that the closer the  $\epsilon$ -indicator of an approximation frontier is to 1, the better is its quality.

### 4.3 Statistical Analysis

When executing a strategy more than once, we observe that the results are not deterministic, since they vary from execution to execution. Due to this variability, we have to resort to statistical analysis in order to take conclusions.

After executing a strategy 10 times, we get 10 sets of solutions, each of which associated to each execution. For each set, we measure its  $\epsilon$ -indicator. After those measurements, we have a group of 10  $\epsilon$ -indicator values, each of which associated to that strategy.

When comparing different strategies, we are comparing the groups of 10  $\epsilon$ -indicator values associated to each strategy. These groups are graphically depicted on a boxplot that follows Tukey's original definition of boxplots, and that has notches representing 95% confidence intervals [28].

Next, we perform a *one-way analysis of variance* in order to test the null hypothesis that all the groups have equal means. First, we check if the assumptions to reliably execute this test are met. For each group, we use the *Shapiro-Wilk test* to test the null hypothesis that the group is normally distributed. Afterwards, we use the *Levene's test* to test the null hypothesis that all the group have equal variances.

If both null hypotheses are not rejected, then it indicates that we have met the assumptions. Consequently, we perform a *one-way analysis of variance* on the groups. If the null hypothesis that the groups have equal means is rejected, we advance to a post-hoc analysis, by performing a pairwise *t-test* with a Bonferroni correction for multiple comparisons. In case the assumptions are not met, we perform a non-parametric alternative, the *Kruskal-Wallis H test*. If the null hypothesis that the groups are significantly different is rejected, we perform a *Conover-Iman test* [29] with a Bonferroni correction for multiple comparisons. We consider a significance value of 0.05. Moreover, the reader should take into account that our work does not assess the impact of any potential interaction between two parameters. In this sense, our work focuses on the main effect of each parameter.

This page is intentionally left blank.

## Chapter 5

# Workflow Implementation

In this chapter, we describe the pipeline we implemented to generate the dataset of assets that constitute the universe of the POP, to solve the problem with a MILP solver, to solve the strategies and scenarios with a D-Wave computer, to calculate the  $\epsilon$ -indicator, and to perform the statistical analysis on the results. Our pipeline, which is depicted in Figure 5.1, was mainly implemented in Python.

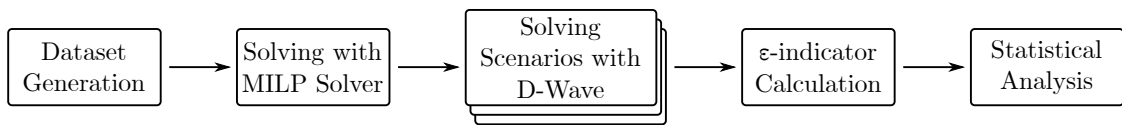


Figure 5.1: Diagram of the implemented pipeline of our study

To facilitate reuse and help with reproducibility, our entire code base is publicly hosted at:

<https://github.com/TofuLynx/embedding>

The steps in our pipeline follow the methodology we described in the previous chapter. The implementation of each step is going to be described as follows:

- **Section 5.1** describes how we retrieve the market information and how we generate the datasets.
- **Section 5.2** explains how we solve a dataset to optimality using a MILP solver to obtain a representation of the efficient frontier.
- **Section 5.3** describes the functions and Python packages used to solve instances of POP with a D-Wave's system, taking into account the different strategies and scenarios that use the dataset.
- **Section 5.4** explains our strategy for calculating the  $\epsilon$ -indicator for a set of solutions of an execution of a strategy.
- **Section 5.5** explains the Python packages used to perform the statistical analysis on the results of the strategies previously identified.



## 5.1 Dataset Generation

We had to generate different datasets for the different strategies and scenarios we designed. The market information used to generate our datasets was retrieved with the use of two Python packages: `pandas` [30] and `yfinance` [31]. With the function `read_html` of `pandas`, we download from a Wikipedia page<sup>1</sup> an updated table of Standard & Poor’s (S&P)’s 500, a stock market index that comprises stocks issued by 500 of the largest companies listed on stock exchanges in the United States [32]. This table includes the symbol associated to each stock, as well as their respective company and industry sector, among other information. The industry sectors are categorized according to a taxonomy, the Global Industry Classification Standard (GICS), also developed by S&P [33].

To generate an *industry diversified* or an *industry correlated* dataset, we consider the GICS sector associated to each of the assets from S&P’s 500. Concretely, for *industry diversified*, we first select the maximum number of assets of different GICS sectors possible. If, after the initial selection, we are still under the required number of assets to generate the dataset, we repeat this step until the number of assets necessary to generate the dataset is satisfied. For *industry correlated*, we first select the maximum number of assets of a single GICS sector as possible. If, after selecting, we are still under the required number of assets to generate the dataset, we repeat this step for another GICS sector until the number of assets necessary to generate the dataset is satisfied.

To generate a *diversified* or a *correlated* dataset, we consider the market performance associated to each of the assets from S&P’s 500 of the last period of a month, with 1-day intervals. This period could also be a year or a day but we opted for a month, since we believe that a month provides sufficient information about the asset’s performance. To download this information for all the assets from S&P’s 500, we use the function `download` of `yfinance`, which returns the market performance associated to each asset in the form of a `pandas DataFrame`. We use the functions `dropna` and `isnull` of `pandas` to remove from this `DataFrame` every NaN value and every asset that has no market performance associated. Afterwards, we use the function `pct_change` of `pandas` to transform the `DataFrame` such that it lists the percentage value change between two consecutive days. This transformation creates a line of NaN values that is also removed. From this transformed `DataFrame`, we call the function `corr` to obtain a pairwise correlation matrix of the assets’ market performance. We use this matrix to choose which assets are selected to generate the dataset, as described in Algorithm 1.

What differentiates the algorithm from generating a *diversified* dataset or a *correlated* dataset is the  $Score(\mathcal{M}, a, b)$  function, where  $\mathcal{M}$  is the pairwise correlation matrix, and  $a$  and  $b$  are assets. Concretely, if we generate a *diversified* dataset, this function is  $Score(\mathcal{M}, a, b) = -|\mathcal{M}_{a,b}|$ , favoring pairwise correlations whose value is closest to 0. In the case of a *correlated* dataset, this function is  $Score(\mathcal{M}, a, b) = \mathcal{M}_{a,b}$ , favoring pairwise correlations with higher values.

After selecting the assets to generate the desired dataset, we use the function `download` of `yfinance` to retrieve the market performance of the assets, in the same way we do to obtain the pairwise correlation matrix. Once the `DataFrame` is retrieved, we remove every NaN value and every asset that has no market performance associated and transform the `DataFrame` such that it lists the percentage value change between two consecutive days. Again, we remove the resulting line of NaN values from this `DataFrame`. From this transformed `DataFrame`, we call the functions `mean` and `cov` to obtain the parameters  $\mu$

<sup>1</sup>[https://en.wikipedia.org/wiki/List\\_of\\_S%26P\\_500\\_companies](https://en.wikipedia.org/wiki/List_of_S%26P_500_companies)

and  $\Sigma$  associated to the assets' market performance, respectively. The generated dataset is stored in a JSON file, containing the universe size  $n$ , the symbols of the selected assets, and the  $\mu$  and  $\Sigma$  associated to the assets' market performance.

---

**Algorithm 1:** Select assets to generate a *diversified* or *correlated* dataset

---

**Data:** Set of assets  $\mathcal{A}$   
Number of assets to select  $n$   
Pairwise correlation matrix  $\mathcal{M}$   
**Result:** Set of selected assets  $\mathcal{S}$

```

begin
  copy the first  $n$  assets of  $\mathcal{A}$  to  $\mathcal{S}$ ;
   $\mathcal{S}_{\text{score}} \leftarrow -\infty$ 
  foreach asset  $a$  of the set  $\mathcal{A}$  do
     $\mathcal{L} \leftarrow \{a\}$ ;
    while  $|\mathcal{L}| < n$  do
       $t \leftarrow \emptyset$ ;
       $t_{\text{score}} \leftarrow -\infty$ ;
      foreach asset  $b$  of the set  $\mathcal{A}$  not in  $\mathcal{L}$  do
         $b_{\text{score}} \leftarrow 0$ ;
        foreach asset  $l$  of the set  $\mathcal{L}$  do
           $b_{\text{score}} \leftarrow b_{\text{score}} + \text{Score}(\mathcal{M}, b, l)$ ;
        end
        if  $b_{\text{score}} > t_{\text{score}}$  then
           $t \leftarrow b$ ;
           $t_{\text{score}} \leftarrow b_{\text{score}}$ ;
        end
      end
      append  $t$  to  $\mathcal{L}$ ;
    end
     $\mathcal{L}_{\text{score}} \leftarrow 0$ ;
    foreach pair of assets  $x$  and  $y$  in  $\mathcal{L}$  do
       $\mathcal{L}_{\text{score}} \leftarrow \mathcal{L}_{\text{score}} + \text{Score}(\mathcal{M}, x, y)$ ;
    end
    if  $\mathcal{L}_{\text{score}} > \mathcal{S}_{\text{score}}$  then
       $\mathcal{S} \leftarrow \mathcal{L}$ ;
       $\mathcal{S}_{\text{score}} \leftarrow \mathcal{L}_{\text{score}}$ ;
    end
  end
end
end

```

---

## 5.2 Solving with MILP Solver

We use the MILP solver from Solving Constraint Integer Programs (SCIP) [34, 35] to solve a given dataset to optimality for each of the specified directions, such that we obtain a representation of the efficient frontier associated to the dataset. The directions that are solved to optimality are based on initial tests with the datasets, and cover both extrema of the efficient frontier:  $i \times 10^j$  for  $i = 1, \dots, 9, j = -1, 0, 1$ ; as well as 0, 100, 500, and 1000. Apart from the information contained in the dataset, the budget size also needs to be specified.

The SCIP solver uses a specific input file format, which is difficult to generate. For this reason, we use a model written in MathProg which is then converted by another MILP solver, from the GNU Linear Programming Kit (GLPK) [36], to the specific file format required by the SCIP solver. Figure 5.2 shows a template of the LP model of the Portfolio Optimization Problem (POP) written in MathProg (see LP formulation at Section 3.2). This MathProg file template has all the necessary information about the parameters before its conversion to the file format required by the SCIP solver.

```

2 #parameters
3 param n, integer, > 0; /* number of assets */
4 param B, >= 0;
5 param mu{i in 1..n};
6 param sigma{i in 1..n, j in 1..n};
7 param q;
8
9 #variables
10 var y{i in 1..n, j in 1..n}, binary;
11 var x{i in 1..n}, binary;
12
13 #capacity constraint
14 s.t. card0: sum{i in 1..n} x[i] = B ;
15 #constraints to link x and y
16 s.t. card1{i in 1..n, j in 1..n}: y[i,j] <= x[i] ;
17 s.t. card2{i in 1..n, j in 1..n}: y[i,j] <= x[j] ;
18 s.t. card3{i in 1..n, j in 1..n}: y[i,j] >= x[i]+x[j]-1 ;
19
20 #objective function
21 minimize obj: sum{i in 1..n, j in 1..n} y[i,j] * sigma[i,j] * q - sum{i in
    1..n} mu[i]*x[i];
22
23 solve;

```

Figure 5.2: MathProg file template of the POP (`IPL_linearized_template.ampl`)

The SCIP solver is also used to obtain the solution of the dataset with the highest volatility. In this case, we use a different model, which maximizes the volatility objective without any influence from the expected return objective (the model does not include the expected return term present in the line 21 of Figure 5.2). The volatility of the obtained solution serves as the upper bound on the volatility of the given dataset (see Section 4.2.3).

### 5.3 Solving Scenarios with D-Wave

To solve a given dataset, D-Wave’s system needs an embedding and a Quadratic Unconstrained Binary Optimization (QUBO) matrix to execute a strategy. We use the set of Python packages provided by D-Wave’s Ocean Tools to meet those needs. In particular, we use `minorminer` to generate embeddings, and `dwave-system` to execute the QUBO matrix as a problem instance. Note that the QUBO matrix is unique to each strategy (any change in the parameters causes a different QUBO matrix to be developed for that strategy) and therefore a new matrix is always developed for each execution.

After retrieving the parameters from the given strategy, we use three `minorminer` functions to generate embeddings: `find_embedding`, `busclique.find_clique_embedding`, and `layout.find_embedding`, for embeddings *general*, *clique*, and *layout*, respectively (see in Section 2.4). These functions return a mapping between the variables and the physical qubits, which is necessary for the D-Wave’s system to solve any problem instance. After-

wards, we generate in an incremental fashion a QUBO matrix for each of the directions specified in the given strategy, as described in Figure 5.3.

```

46 for q in q_values:
47     # Step 2: Formulate QUBO
48     Q = defaultdict(float)
49
50     P = -q * min_sigma + max_mu
51
52     # There are three terms in the objective function: Volatility, Return,
53     # and Budget
54
55     # Volatility term
56     for i in range(N):
57         Q[(i, i)] = float(q * sigma[tickers[i]][tickers[i]])
58         for j in range(i+1, N):
59             Q[(i, j)] = float(2 * q * sigma[tickers[i]][tickers[j]])
60
61     # Return term
62     for i in range(N):
63         Q[(i, i)] += float(-mu[tickers[i]])
64
65     # Budget term is decomposed into four terms, per the formula ((sum^{n
66     # -1}_{i=0} x_i) - B)^2
67     for i in range(N):
68         Q[(i, i)] += float(P)
69
70     for i in range(N):
71         for j in range(i + 1, N):
72             Q[(i, j)] += float(2*P)
73
74     for i in range(N):
75         Q[(i, i)] += float(-2 * B * P)

```

Figure 5.3: Python code to create a QUBO matrix for each direction specified for  $q$  (`pop_solver_annealer.py`)

For each of the generated QUBO matrices for a given strategy, we call the function `sample_qubo` from `dwave-system`'s `FixedEmbeddingComposite`, a composite that maps problems to a system, with a specified embedding. In the context of our work, this function maps the QUBO matrix of a strategy to the D-Wave's system, with the embedding associated to that strategy. The function can also be invoked with many more arguments, such as the chain strength, the number of samples, and the annealing schedule, all parameters that depend on the given strategy. A set of solutions is returned for each call of this function, comprising a larger set of solutions that is associated to an execution of the given strategy.

## 5.4 $\epsilon$ -indicator Calculation

The set of solutions associated to an execution of a given strategy presents both feasible and unfeasible solutions. In order to identify the approximation frontier associated to this set, we first exclude the unfeasible solutions, i.e. solutions that do not comply with the cardinality constraint (budget size). Next, we also exclude solutions with a non-positive expected return. Last, we identify the approximation frontier by selecting the subset of solutions that are not dominated.

We use both the representation of the efficient frontier obtained by the SCIP solver and the approximation frontier of the strategy's set of solutions to calculate the  $\epsilon$ -indicator value associated to the set of solutions, as described in Section 4.2.3.

## 5.5 Statistical Analysis

We use a series of tools from different Python packages as part of our statistical analysis. Concretely, we use the `matplotlib` package [37] to create boxplots and line charts; the `scipy` package [38] to perform the *Shapiro-Wilk test*, the *Levene's test*, the *one-way analysis of variance* and the *Kruskal-Wallis H test*; and the `scikit-posthocs` package to perform the pairwise *t-test* and the *Conover-Iman test*, both with a Bonferroni correction for multiple comparisons.

# Chapter 6

## Results and Discussion

The goal of our work is to identify the influence that different parameters, either D-Wave’s system parameters or parameters that are specific to the Portfolio Optimization Problem (POP), have on the quality of the solutions computed by a quantum annealer.

In the previous chapters, we described the parameters we are considering and the experimental methodology we have designed and implemented to support our work.

In this Chapter, we present and analyze the results we obtained from executing the scenarios and strategies we have designed. In each of the following sections, we describe and analyze the results obtained for one of the scenarios. In the last section, Section 6.10, we compile a list of the main takeaways of our work.

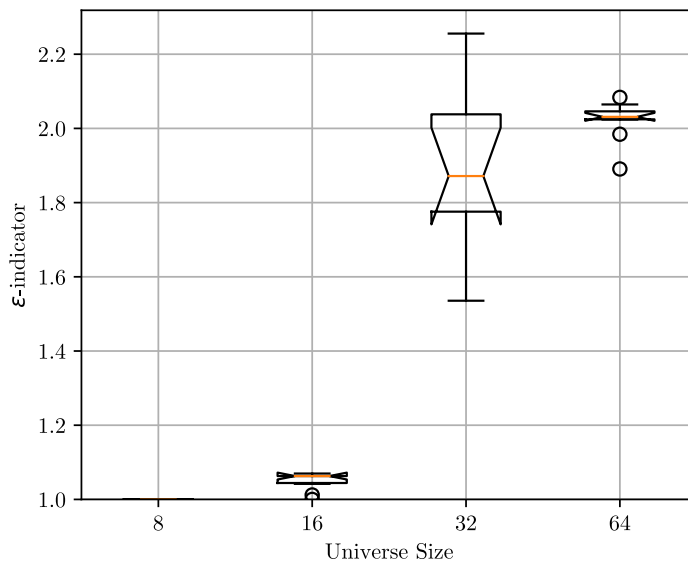
### 6.1 Universe Size

In this section, we present and analyze the results of Scenario  $\mathcal{S}_1$ , which focuses on the influence of the universe size parameter (see in Section 4.1.1). Figure 6.1 shows the boxplot of the  $\epsilon$ -indicator values associated to the results of the strategies that are part of this scenario.

The *Shapiro-Wilk test* did not preserve the null hypothesis for strategies  $\mathcal{S}_{1b}$  and  $\mathcal{S}_{1d}$ . The *Levene’s test* rejected the null hypothesis for all strategies. Hence, the assumptions to reliably perform a *one-way analysis of variance* are not met. The *Kruskal-Wallis H test* on the results returned a  $p$ -value of  $< 0.001$ , which rejects the null hypothesis that the strategies’ results originate from the same distribution. Table 6.1 shows the  $p$ -values returned by the *Conover-Iman test* with a Bonferroni correction for multiple comparisons on the results of this scenario.

Table 6.1: Output of the *Conover-Iman test* on  $\mathcal{S}_1$ ’s results

	<b>8</b>	<b>16</b>	<b>32</b>	<b>64</b>
<b>8</b>	—	$< 0.001$	$< 0.001$	$< 0.001$
<b>16</b>	—	—	$< 0.001$	$< 0.001$
<b>32</b>	—	—	—	0.411
<b>64</b>	—	—	—	—

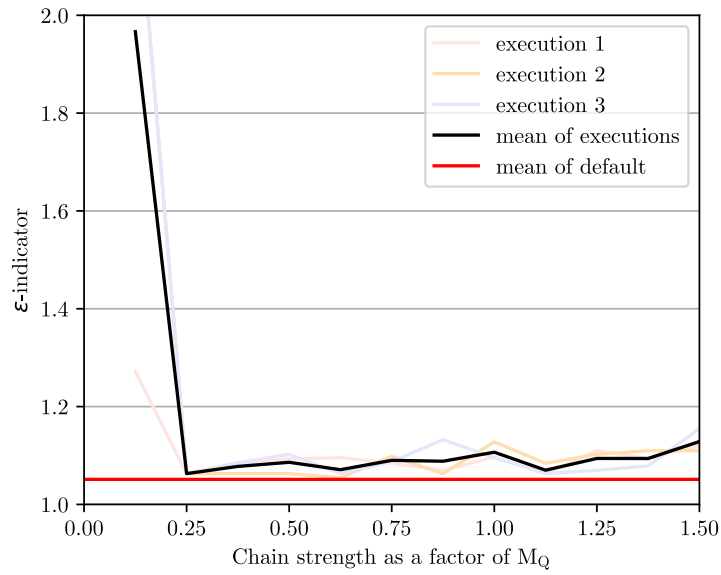
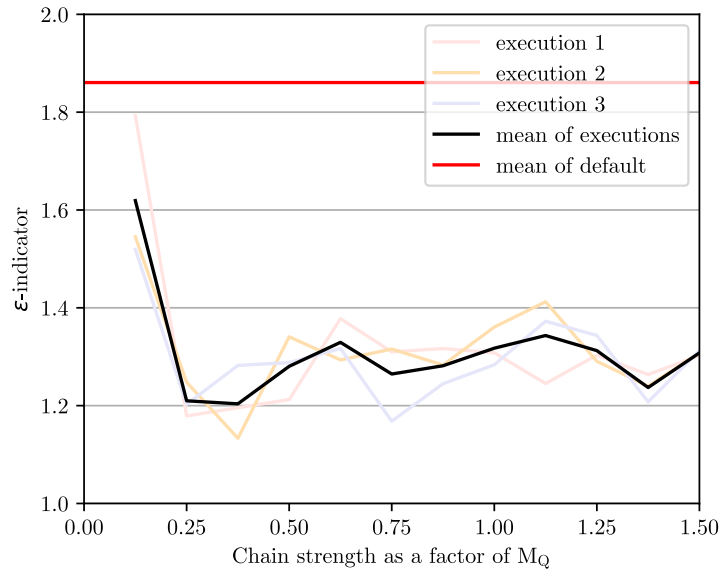
Figure 6.1: Boxplot of  $\mathcal{S}_1$ 's results

According to the statistical tests, there exists a significant difference between all strategies in terms of  $\epsilon$ -indicator, except between  $\mathcal{S}_{1c}$  and  $\mathcal{S}_{1d}$ .  $\mathcal{S}_{1a}$ , with  $n = 8$ , got a median  $\epsilon$ -indicator of 1.000, which is the lower bound of the indicator. This suggests that D-Wave's system consistently obtains a part of the optimal efficient frontier for this universe size.  $\mathcal{S}_{1b}$ , with  $n = 16$ , got a median  $\epsilon$ -indicator of 1.063, very close to the lower bound of the  $\epsilon$ -indicator.  $\mathcal{S}_{1c}$ , with  $n = 32$ , got a median  $\epsilon$ -indicator of 1.872 in its executions, which is a large difference to strategies  $\mathcal{S}_{1a}$  and  $\mathcal{S}_{1b}$ . This suggests that the quality of the solutions obtained by D-Wave's system gets significantly worse at some point between universe sizes 16 and 32.  $\mathcal{S}_{1d}$ , with  $n = 64$ , got a median  $\epsilon$ -indicator of 2.031 in its executions, the worst median obtained in this scenario. This value suggests that universe size 64 is the hardest to solve in this scenario. These results suggest that the quality of the solutions obtained by D-Wave's system deteriorates with the growth of the universe size.

## 6.2 Chain Strength

In this section, we present and analyze the results of Scenario  $\mathcal{S}_2$ , which focuses on the influence of the chain strength parameter (see in Section 4.1.2). Figures 6.2, 6.3, and 6.4 show the line charts of the  $\epsilon$ -indicator values associated to the results of the strategies that are part of this scenario. The line charts show in red the mean  $\epsilon$ -indicator of 12 executions with the default chain strength, and in black the mean  $\epsilon$ -indicator for the three executions at each value of chain strength as a factor of  $M_Q$ . In pale colors are shown the values of the individual executions.

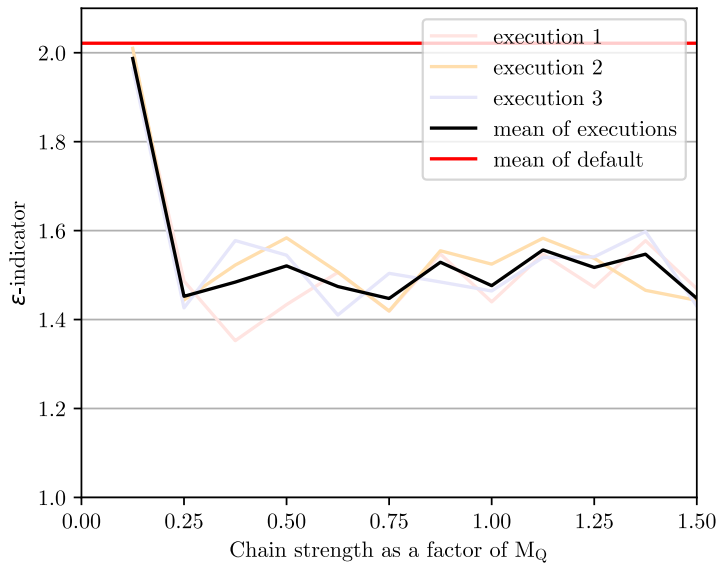
For  $n = 16$ , the quality of the solutions did not see neither a meaningful improvement nor a meaningful deterioration when comparing  $\mathcal{S}_{2a}$  to any of the other strategies, except for a significant deterioration at  $\mathcal{S}_{2b}$ , which has a chain strength of  $0.125M_Q$ . For  $n = 32$ , the quality of the solutions saw a meaningful improvement when comparing  $\mathcal{S}_{2a}$  to any of the other strategies, except for  $\mathcal{S}_{2b}$ , which got a mean  $\epsilon$ -indicator just slightly lower than the mean  $\epsilon$ -indicator of  $\mathcal{S}_{2a}$ . In general, the improvement was from an  $\epsilon$ -indicator of just over 1.8 to around 1.3. For  $n = 64$ , the quality of the solutions saw a meaningful

Figure 6.2: Line chart of  $\mathcal{S}_2$ 's results for  $n = 16$ Figure 6.3: Line chart of  $\mathcal{S}_2$ 's results for  $n = 32$ 

improvement when comparing  $\mathcal{S}_{2a}$  to any of the other strategies, except for  $\mathcal{S}_{2b}$ , which got a mean  $\epsilon$ -indicator similar to the mean  $\epsilon$ -indicator of  $\mathcal{S}_{2a}$ . In general, the improvement was from an  $\epsilon$ -indicator of just over 2.0 to around 1.5.

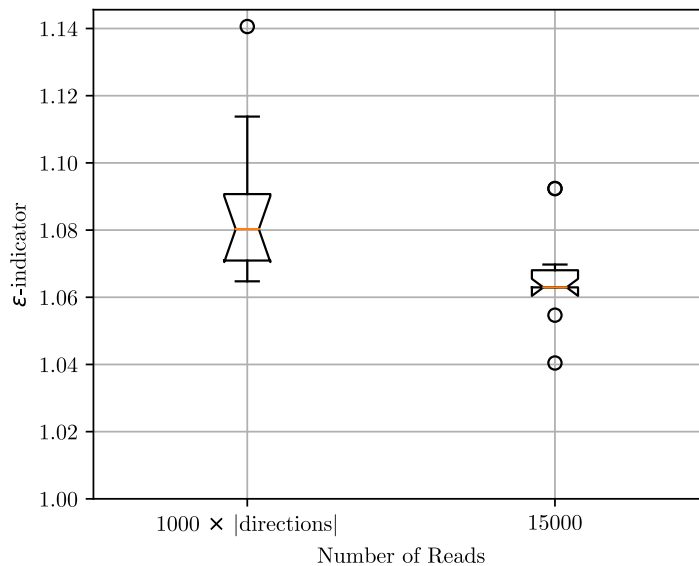
These results suggest, by looking at the disparity between the red line and the black line, that the benefit of setting a chain strength value is larger for  $n = 32$  and  $n = 64$ , also suggesting that larger problem instances benefit significantly more from a good value of chain strength. Moreover, the line charts also suggest that the range of chain strength between  $0.250M_Q$  and  $1.500M_Q$  consistently obtains the solutions with best quality.



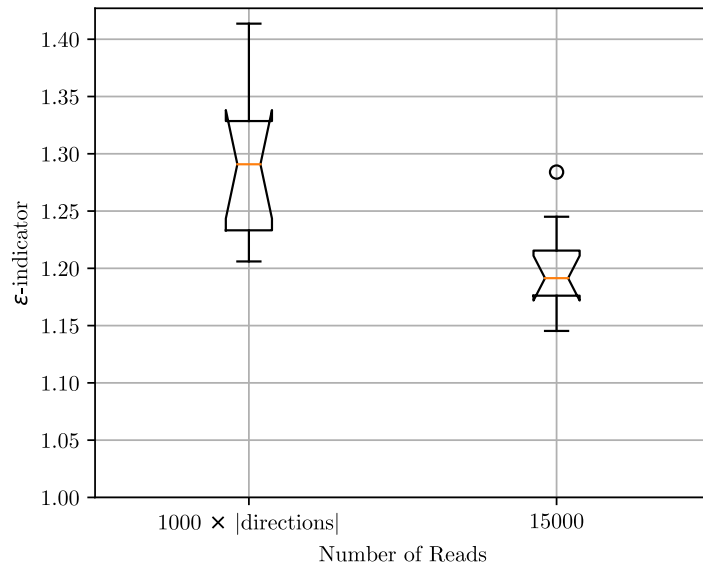
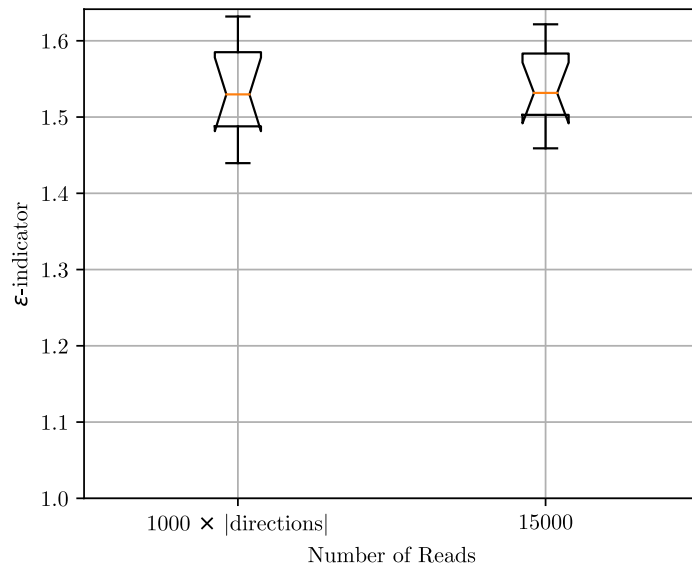
Figure 6.4: Line chart of  $\mathcal{S}_2$ 's results for  $n = 64$ 

### 6.3 Number of Reads

In this section, we present and analyze the results of Scenario  $\mathcal{S}_3$ , which focuses on the influence of the number of reads parameter (see in Section 4.1.3). Figures 6.5, 6.6, and 6.7 show the boxplots of the  $\epsilon$ -indicator values associated to the results of the strategies that are part of this scenario, for universe sizes 16, 32, and 64, respectively.

Figure 6.5: Boxplot of  $\mathcal{S}_3$ 's results for  $n = 16$ 

The *Shapiro-Wilk test* preserved the null hypothesis for both strategies for all universe size, except  $\mathcal{S}_{3a}$  for  $n = 16$ . The *Levene's test* preserved the null hypothesis for all strategies for all universe sizes. Hence, the assumptions to reliably perform a *one-way analysis of*

Figure 6.6: Boxplot of  $\mathcal{S}_3$ 's results for  $n = 32$ Figure 6.7: Boxplot of  $\mathcal{S}_3$ 's results for  $n = 64$ 

*variance* are met for universe sizes  $n = 32$  and  $n = 64$ . The *Kruskal-Wallis H test* on the results returned a  $p$ -value of 0.0105, which rejects the null hypothesis that the strategies' results originate from the same distribution. The *one-way analysis of variance* on the results returned a  $p$ -value of 0.002 for  $n = 32$ , and 0.917 for  $n = 64$ , which rejects the null hypothesis that the strategies' results have equal means for universe size 32, and preserves the null hypothesis for universe size 64.

For  $n = 16$ ,  $\mathcal{S}_{3a}$ , which reads 1000 solutions for each of the 5 specified directions, totaling 5000 reads, got a median  $\epsilon$ -indicator of 1.080. For the same universe size,  $\mathcal{S}_{3b}$ , which reads a total of 15000 reads, distributed among the 5 specified directions, got a median  $\epsilon$ -indicator of 1.063. These results suggest that  $\mathcal{S}_{3b}$  is a significantly better strategy than  $\mathcal{S}_{3a}$ , when

considering the quality of the solutions returned by D-Wave’s system for  $n = 16$ . For  $n = 32$ ,  $\mathcal{S}_{3a}$ , which reads 1000 solutions for each of the 7 specified directions, totaling 7000 reads, got a median  $\epsilon$ -indicator of 1.291. For the same universe size,  $\mathcal{S}_{3b}$ , which reads a total of 15000 reads, distributed among the 7 specified directions, got a median  $\epsilon$ -indicator of 1.191. These results suggest that  $\mathcal{S}_{3b}$  is a significantly better strategy than  $\mathcal{S}_{3a}$ , when considering the quality of the solutions returned by D-Wave’s system for  $n = 32$ . For  $n = 64$ ,  $\mathcal{S}_{3a}$ , which reads 1000 solutions for each of the 15 specified directions, totaling 15000 reads, got a median  $\epsilon$ -indicator of 1.530. For the same universe size,  $\mathcal{S}_{3b}$ , which reads a total of 15000 reads, distributed among the 7 specified directions, got a median  $\epsilon$ -indicator of 1.532. These results suggest that  $\mathcal{S}_{3b}$  is neither significantly better nor significantly worse than  $\mathcal{S}_{3a}$ , when considering the quality of the solutions returned by D-Wave’s system for  $n = 64$ .

These results suggest that the quality of the solutions obtained by D-Wave’s system improves with the increase of the number of reads. In fact, for  $n = 16$  and  $n = 32$ , the universe sizes where  $\mathcal{S}_{3b}$  reads more solutions than  $\mathcal{S}_{3a}$  show a significant improvement in the quality of the solutions when changing from  $\mathcal{S}_{3a}$  to  $\mathcal{S}_{3b}$ .

## 6.4 Directions

In this section, we present and analyze the results of Scenario  $\mathcal{S}_4$ , which focuses on the influence of the directions parameter (see in Section 4.1.4). Figures 6.8 and 6.9 show the boxplots of the  $\epsilon$ -indicator values associated to the results of the strategies that are part of this scenario, for universe sizes 32 and 64, respectively.

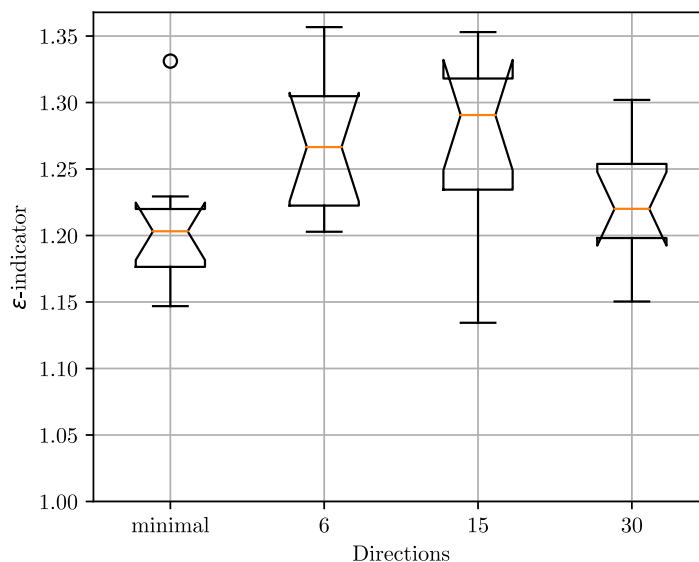
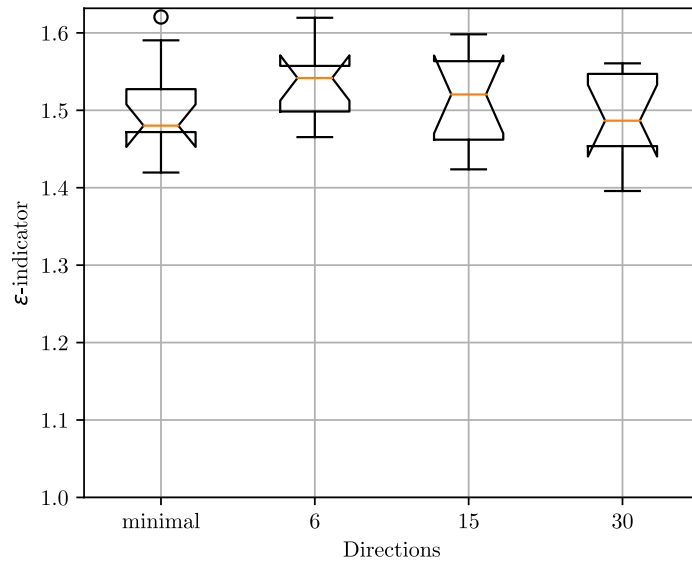


Figure 6.8: Boxplot of  $\mathcal{S}_4$ 's results for  $n = 32$

The *Shapiro-Wilk test* preserved the null hypothesis for each strategy for all universe sizes. The *Levene's test* preserved the null hypothesis for all strategies for all universe sizes. Hence, the assumptions to reliably perform a *one-way analysis of variance* are met for all universe sizes. The *one-way analysis of variance* on the results returned a  $p$ -value of 0.032 for  $n = 32$  and 0.355 for  $n = 64$ , which rejects the the null hypothesis that the strategies' results have equal means for universe size 32 and preserves the same hypothesis for universe

Figure 6.9: Boxplot of  $\mathcal{S}_4$ 's results for  $n = 64$ 

size 64. The pairwise  $t$ -test with a Bonferroni correction for multiple comparisons did not find any significant difference between any pair of strategies for universe size 32. These results suggest that the number of directions has no significant impact on the quality of the solutions obtained by D-Wave's system, for  $n = 32$  and  $n = 64$ .

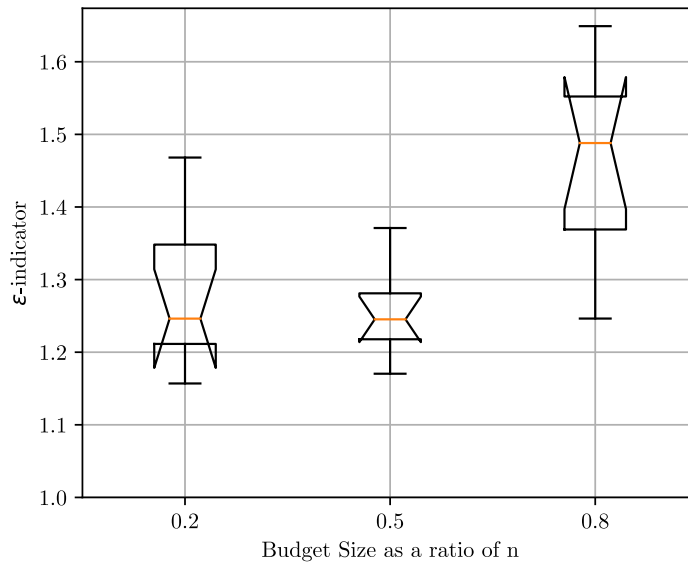
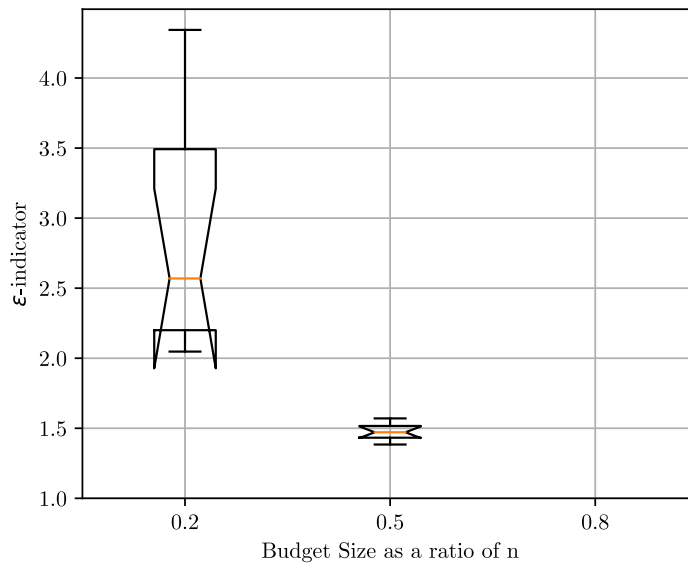
## 6.5 Budget

In this section, we present and analyze the results of Scenario  $\mathcal{S}_5$ , which focuses on the influence of the budget size parameter (see in Section 4.1.5). Figures 6.10 and 6.11 show the boxplots of the  $\epsilon$ -indicator values associated to the results of the strategies that are part of this scenario, for universe sizes 32 and 64, respectively.

The *Shapiro-Wilk test* preserved the null hypothesis for each strategy for all universe sizes. The *Levene's test* rejected the null hypothesis for all strategies for universe size  $n = 64$ . Hence, the assumptions to reliably perform a *one-way analysis of variance* are only met for universe size  $n = 32$ . The *one-way analysis of variance* on the results returned a  $p$ -value of  $< 0.001$  for  $n = 32$ , which rejects the null hypothesis that the strategies' results have equal means. The *Kruskal-Wallis H test* on the results returned a  $p$ -value of  $< 0.001$  for  $n = 64$ , which rejects the null hypothesis that the strategies' results originate from the same distribution for all universe sizes. Table 6.2 shows the  $p$ -values returned by the pairwise  $t$ -test with a Bonferroni correction for multiple comparisons on the results of this scenario, for  $n = 32$ .

For  $n = 32$ ,  $\mathcal{S}_{5a}$ , which has a budget size of  $\lfloor 0.2 \times n \rfloor$ , 6 for this universe size, got a median  $\epsilon$ -indicator of 1.246. For the same universe size,  $\mathcal{S}_{5b}$ , which has a budget size of  $\lfloor 0.5 \times n \rfloor$ , 16 for this universe size, got a median  $\epsilon$ -indicator of 1.245. For the same universe size,  $\mathcal{S}_{5c}$ , which has a budget size of  $\lfloor 0.8 \times n \rfloor$ , 25 for this universe size, got a median  $\epsilon$ -indicator of 1.488. These results suggest that  $\mathcal{S}_{5a}$  and  $\mathcal{S}_{5b}$  are significantly better strategies than  $\mathcal{S}_{5c}$ , when considering the quality of the solutions returned by D-Wave's system for  $n = 32$ .

For  $n = 64$ ,  $\mathcal{S}_{5a}$ , which has a budget size of  $\lfloor 0.2 \times n \rfloor$ , 12 for this universe size, got a

Figure 6.10: Boxplot of  $\mathcal{S}_5$ 's results for  $n = 32$ Figure 6.11: Boxplot of  $\mathcal{S}_5$ 's results for  $n = 64$ 

median  $\epsilon$ -indicator of 2.569. For the same universe size,  $\mathcal{S}_{5b}$ , which has a budget size of  $[0.5 \times n]$ , 32 for this universe size, got a median  $\epsilon$ -indicator of 1.471. For the same universe size,  $\mathcal{S}_{5c}$ , which has a budget size of  $[0.8 \times n]$ , 51 for this universe size, did not get any  $\epsilon$ -indicator, since no feasible solutions were returned. These results suggest that  $\mathcal{S}_{5b}$  is a significantly better strategy than both  $\mathcal{S}_{5a}$  and  $\mathcal{S}_{5c}$ , when considering the quality of the solutions returned by D-Wave's system for  $n = 64$ .

These results suggest that the quality of the solutions obtained by D-Wave's system is best when the budget size is half of the universe size, and deteriorates with the increase of the difference between the budget size and half of the universe size. In fact, for  $n = 64$ , strategies  $\mathcal{S}_{5a}$  and  $\mathcal{S}_{5c}$  show a significant deterioration in the quality of the solutions

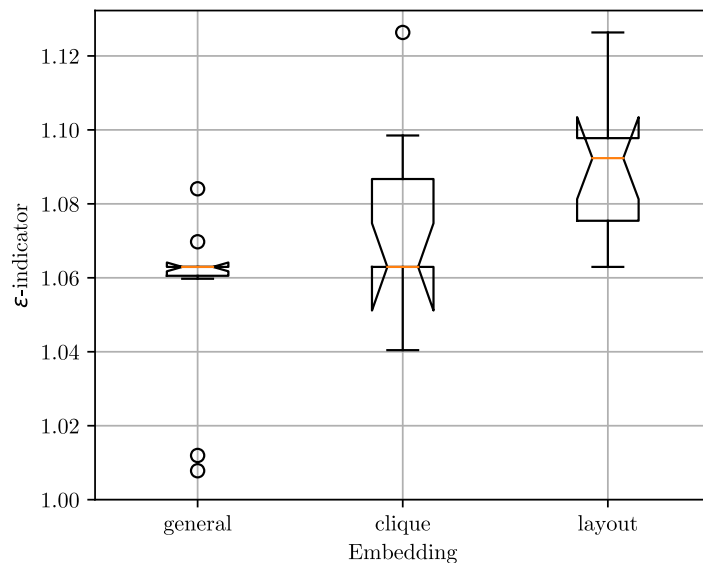
Table 6.2: Output of the pairwise  $t$ -test on  $\mathcal{S}_5$ 's results for  $n = 32$ 

	0.2	0.5	0.8
0.2	—	1.000	0.008
0.5	—	—	< 0.001
0.8	—	—	—

compared to  $\mathcal{S}_{5b}$ . The results also suggest that  $\mathcal{S}_{5c}$  is the worst strategy in this scenario, particularly due to the lack of feasible solutions in this strategy for  $n = 64$ .

## 6.6 Embedding

In this section, we present and analyze the results of Scenario  $\mathcal{S}_6$ , which focuses on the influence of the embedding parameter (see in Section 4.1.6). Figures 6.12, 6.13, and 6.14 show the boxplots of the  $\epsilon$ -indicator values associated to the results of the strategies that are part of this scenario, for universe sizes 16, 32, and 64, respectively.

Figure 6.12: Boxplot of  $\mathcal{S}_6$ 's results for  $n = 16$ 

The *Shapiro-Wilk test* only preserved the null hypothesis for all strategies for universe size 32. The *Levene's test* preserved the null hypothesis for all strategies for all universe sizes. Hence, the assumptions to reliably perform a *one-way analysis of variance* are only met for universe size 32. The *Kruskal-Wallis H test* on the results returned a  $p$ -value of 0.009 for  $n = 16$ , and 0.212 for  $n = 64$ , which rejects the null hypothesis that the strategies' results originate from the same distribution for for  $n = 16$ , and preserves the same hypothesis for  $n = 64$ . The *one-way analysis of variance* on the results returned a  $p$ -value of < 0.001 for  $n = 32$ , which rejects the null hypothesis that the strategies' results have equal means for universe size 32. Table 6.3 shows the  $p$ -values returned by the *Conover-Iman test* with a Bonferroni correction for multiple comparisons on the results of this scenario, for  $n = 16$ .

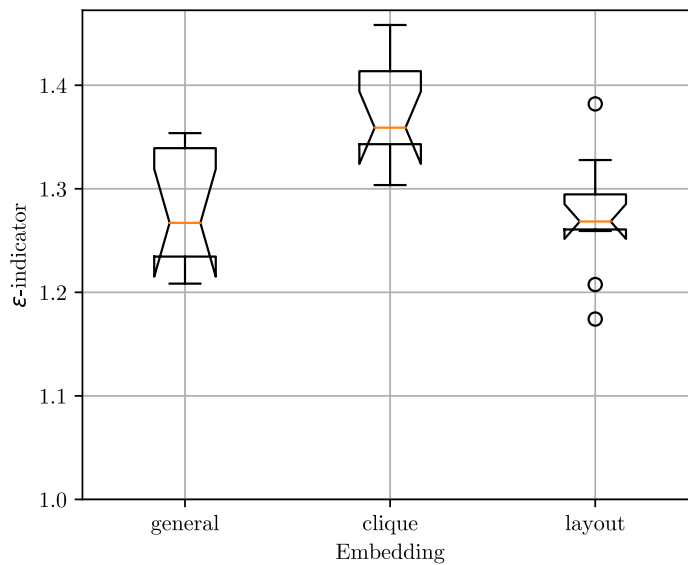
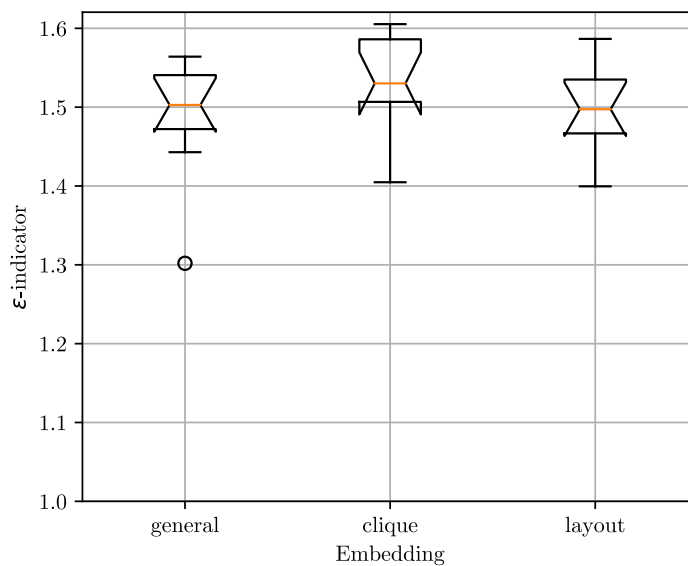
Figure 6.13: Boxplot of  $\mathcal{S}_6$ 's results for  $n = 32$ Figure 6.14: Boxplot of  $\mathcal{S}_6$ 's results for  $n = 64$ 

Table 6.4 shows the  $p$ -values returned by the pairwise  $t$ -test with a Bonferroni correction for multiple comparisons on the results of this scenario, for  $n = 32$ .

For  $n = 16$ ,  $\mathcal{S}_{6a}$ , which uses *general* embedding, got a median  $\epsilon$ -indicator of 1.063. For the same universe size,  $\mathcal{S}_{6b}$ , which uses *clique* embedding, got a median  $\epsilon$ -indicator of 1.063. For the same universe size,  $\mathcal{S}_{6c}$ , which uses *layout* embedding, got a median  $\epsilon$ -indicator of 1.092. These results suggest that strategies  $\mathcal{S}_{6a}$  and  $\mathcal{S}_{6b}$  are significantly better than  $\mathcal{S}_{6c}$ , when considering the quality of the solutions returned by D-Wave's system for  $n = 16$ . For  $n = 32$ ,  $\mathcal{S}_{6a}$ , which uses *general* embedding, got a median  $\epsilon$ -indicator of 1.267. For the same universe size,  $\mathcal{S}_{6b}$ , which uses *clique* embedding, got a median  $\epsilon$ -indicator of 1.359. For the same universe size,  $\mathcal{S}_{6c}$ , which uses *layout* embedding, got a median  $\epsilon$ -indicator

Table 6.3: Output of the *Conover-Iman test* on  $\mathcal{S}_6$ 's results for  $n = 16$ 

	<i>general</i>	<i>clique</i>	<i>layout</i>
<i>general</i>	—	0.457	0.004
<i>clique</i>	—	—	0.131
<i>layout</i>	—	—	—

Table 6.4: Output of the pairwise *t-test* on  $\mathcal{S}_6$ 's results for  $n = 32$ 

	<i>general</i>	<i>clique</i>	<i>layout</i>
<i>general</i>	—	0.004	1.000
<i>clique</i>	—	—	0.002
<i>layout</i>	—	—	—

of 1.268. These results suggest that  $\mathcal{S}_{6b}$  is worse than any of the other strategies, when considering the quality of the solutions returned by D-Wave's system for  $n = 32$ . For  $n = 64$ ,  $\mathcal{S}_{6a}$ , which uses *general* embedding, got a median  $\epsilon$ -indicator of 1.503. For the same universe size,  $\mathcal{S}_{6b}$ , which uses *clique* embedding, got a median  $\epsilon$ -indicator of 1.530. For the same universe size,  $\mathcal{S}_{6c}$ , which uses *layout* embedding, got a median  $\epsilon$ -indicator of 1.497. These results suggest that no specific strategy is better than any of the other strategies, when considering the quality of the solutions returned by D-Wave's system for  $n = 64$ .

These results suggest that the embedding parameter has no significant impact on the quality of the solutions obtained by D-Wave's system for  $n = 64$ . For  $n = 16$  and  $n = 32$ , there exists a significant impact, with the results suggesting that *general* embedding is the best suited for these universe sizes, closely followed by *layout* embedding.



## 6.7 Anneal Schedule

In this section, we present and analyze the results of Scenario  $\mathcal{S}_7$ , which focuses on the influence of the anneal schedule parameter (see in Section 4.1.7). Figures 6.15, 6.16, and 6.17 show the boxplots of the  $\epsilon$ -indicator values associated to the results of the strategies that are part of this scenario, for universe sizes 16, 32, and 64, respectively.

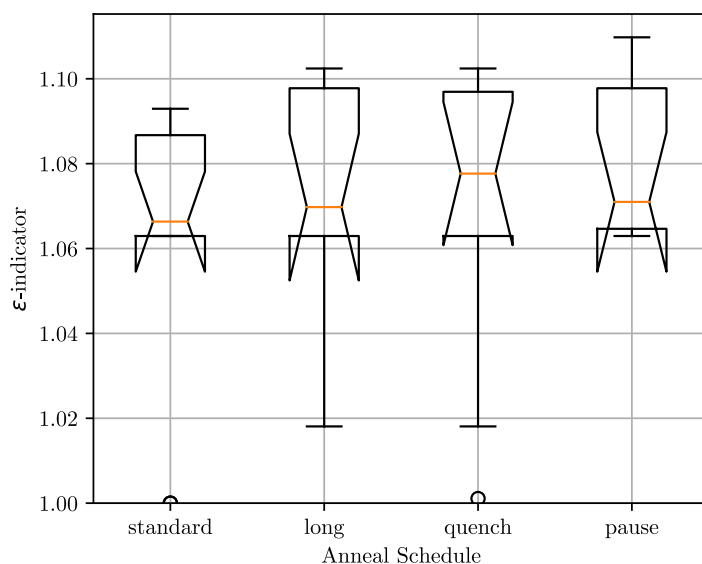


Figure 6.15: Boxplot of  $\mathcal{S}_7$ 's results for  $n = 16$

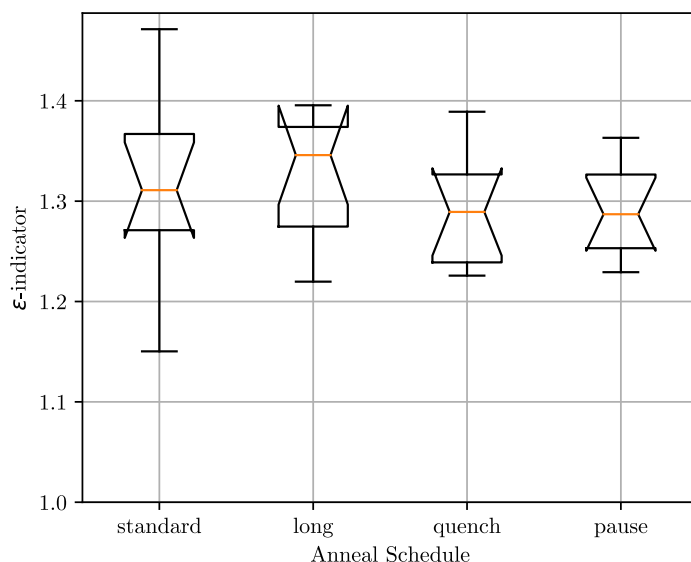
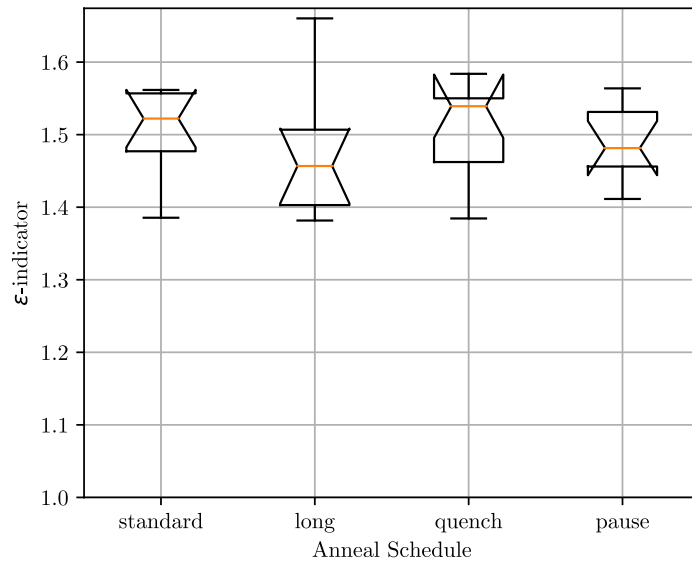


Figure 6.16: Boxplot of  $\mathcal{S}_7$ 's results for  $n = 32$

The *Shapiro-Wilk test* preserved the null hypothesis for each strategy for  $n = 32$ , and rejected the same hypothesis for  $n = 16$  and  $n = 64$ . The *Levene's test* preserved the null hypothesis for all strategies for all universe sizes. Hence, the assumptions to reliably perform a *one-way analysis of variance* are only met for universe size 32. The *Kruskal-*

Figure 6.17: Boxplot of  $\mathcal{S}_7$ 's results for  $n = 64$ 

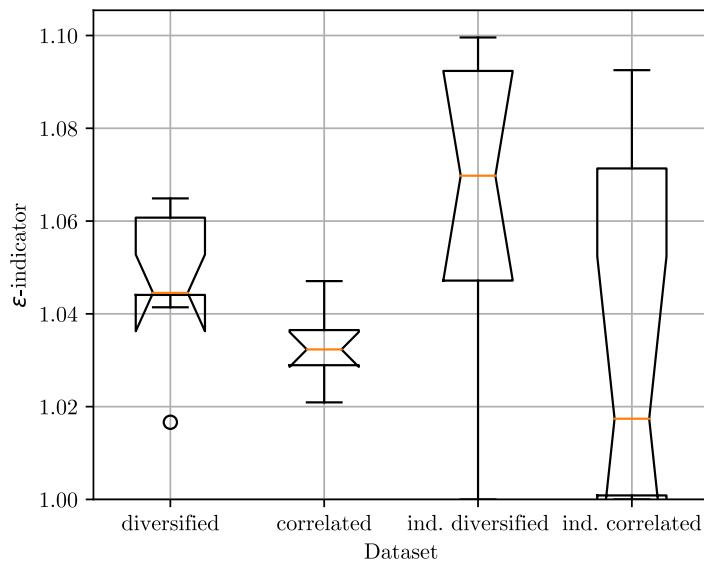
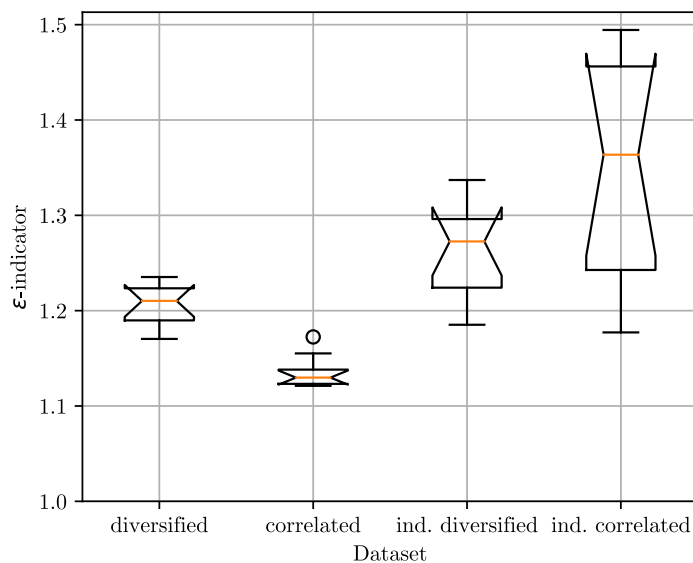
*Wallis H test* on the results returned a  $p$ -value of 0.568 for  $n = 16$  and 0.377 for  $n = 64$ , which preserves the null hypothesis that the strategies' results originate from the same distribution for these universe sizes. The *one-way analysis of variance* on the results returned a  $p$ -value of 0.608 for  $n = 32$ , which preserves the null hypothesis that the strategies' results have equal means for this universe size. These results suggest that varying the anneal schedule parameter has no significant impact on the quality of the solutions obtained by D-Wave's system, for all universe sizes.

## 6.8 Dataset

In this section, we present and analyze the results of Scenario  $\mathcal{S}_8$ , which focuses on the influence of the dataset parameter (see in Section 4.1.8). Figures 6.18, 6.19 and 6.20 show the boxplots of the  $\epsilon$ -indicator values associated to the results of the strategies that are part of this scenario, for universe sizes 16, 32, and 64, respectively.

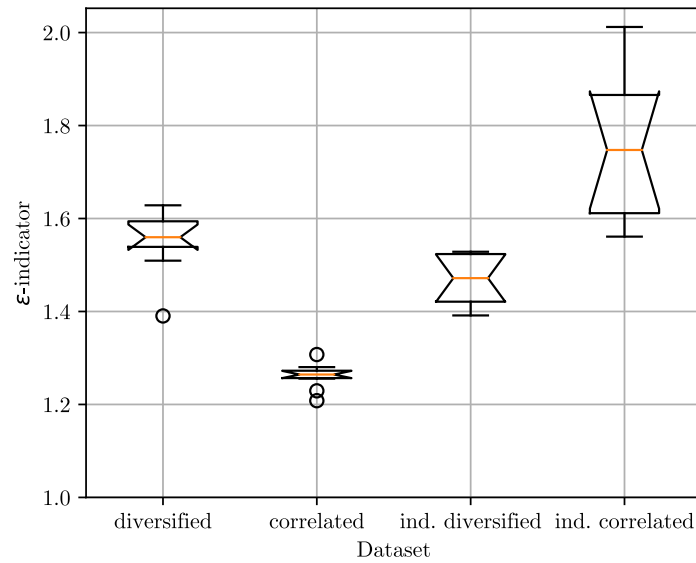
The *Shapiro-Wilk test* did not preserve the null hypothesis for all strategies for all universe sizes. The *Levene's test* rejected the null hypothesis for all strategies for all universe sizes. Hence, the assumptions to reliably perform a *one-way analysis of variance* are not met for all universe sizes. The *Kruskal-Wallis H test* on the results returned a  $p$ -value of 0.034 for  $n = 16$ ,  $< 0.001$  for  $n = 32$ , and  $< 0.001$  for  $n = 64$ , which rejects the null hypothesis that the strategies' results originate from the same distribution for all universe sizes. Tables 6.5, 6.6, and 6.7 show the  $p$ -values returned by the *Conover-Iman test* with a Bonferroni correction for multiple comparisons on the results of this scenario, for universe sizes 16, 32, and 64, respectively.

For  $n = 16$ , no significant difference exists between any pair of strategies. For  $n = 32$ , a significant difference exists between all strategies except between  $\mathcal{S}_{8a}$  and  $\mathcal{S}_{8c}$ , and between  $\mathcal{S}_{8c}$  and  $\mathcal{S}_{8d}$ . These results suggest that D-Wave's system returns solutions with better quality for the datasets used in  $\mathcal{S}_{8a}$  and  $\mathcal{S}_{8b}$ , *diversified* and *correlated*, than for the datasets used in  $\mathcal{S}_{8c}$  and  $\mathcal{S}_{8d}$ , *industry diversified* and *industry correlated*, for  $n = 32$ . Moreover,

Figure 6.18: Boxplot of  $\mathcal{S}_8$ 's results for  $n = 16$ Figure 6.19: Boxplot of  $\mathcal{S}_8$ 's results for  $n = 32$ 

these results also suggest that D-Wave's system returns solutions with better quality for the dataset used in  $\mathcal{S}_{8b}$ , *correlated*, than for the dataset used in  $\mathcal{S}_{8a}$ , *diversified*, for  $n = 32$ . For  $n = 64$ , a significant difference exists between all strategies. These results suggest that, for universes larger than  $n = 32$ , D-Wave's system returns solutions with better quality for  $\mathcal{S}_{8b}$ , followed by  $\mathcal{S}_{8c}$ , followed by  $\mathcal{S}_{8a}$  and finally  $\mathcal{S}_{8d}$ . If we order the datasets by the quality of their associated solutions, they are *correlated*, *industry diversified*, *diversified*, and *industry correlated*.

These results suggest that, when looking at the strategies generated via a statistical method, *diversified* and *correlated*, versus the strategies generated via a industry-based method, *industry diversified* and *industry correlated*, their behaviors are not equal. Gener-

Figure 6.20: Boxplot of  $\mathcal{S}_8$ 's results for  $n = 64$ Table 6.5: Output of the *Conover-Iman test* on  $\mathcal{S}_8$ 's results for  $n = 16$ 

	<i>diversified</i>	<i>correlated</i>	<i>industry diversified</i>	<i>industry correlated</i>
<i>diversified</i>	—	0.552	1.000	0.541
<i>correlated</i>	—	—	0.077	1.000
<i>industry diversified</i>	—	—	—	0.075
<i>industry correlated</i>	—	—	—	—

Table 6.6: Output of the *Conover-Iman test* on  $\mathcal{S}_8$ 's results for  $n = 32$ 

	<i>diversified</i>	<i>correlated</i>	<i>industry diversified</i>	<i>industry correlated</i>
<i>diversified</i>	—	< 0.001	0.064	0.001
<i>correlated</i>	—	—	< 0.001	< 0.001
<i>industry diversified</i>	—	—	—	0.940
<i>industry correlated</i>	—	—	—	—

ally, *diversified* has a consistently higher median  $\epsilon$ -indicator compared to *correlated*, while *industry diversified* has a similar or lower median  $\epsilon$ -indicator compared to *industry correlated*. Overall,  $\mathcal{S}_{8b}$ , which uses *correlated* dataset, is consistently the strategy with the

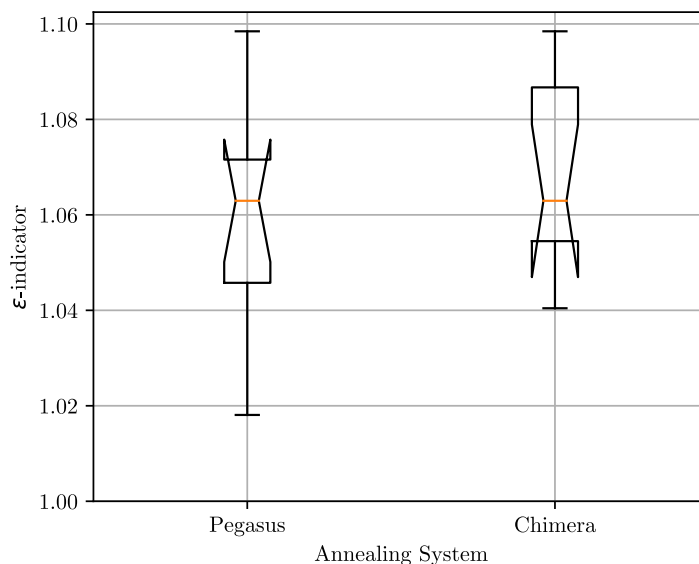
Table 6.7: Output of the *Conover-Iman test* on  $\mathcal{S}_8$ 's results for  $n = 64$ 

	<i>diversified</i>	<i>correlated</i>	<i>industry diversified</i>	<i>industry correlated</i>
<i>diversified</i>	—	< 0.001	0.002	0.001
<i>correlated</i>	—	—	< 0.001	< 0.001
<i>industry diversified</i>	—	—	—	< 0.001
<i>industry correlated</i>	—	—	—	—

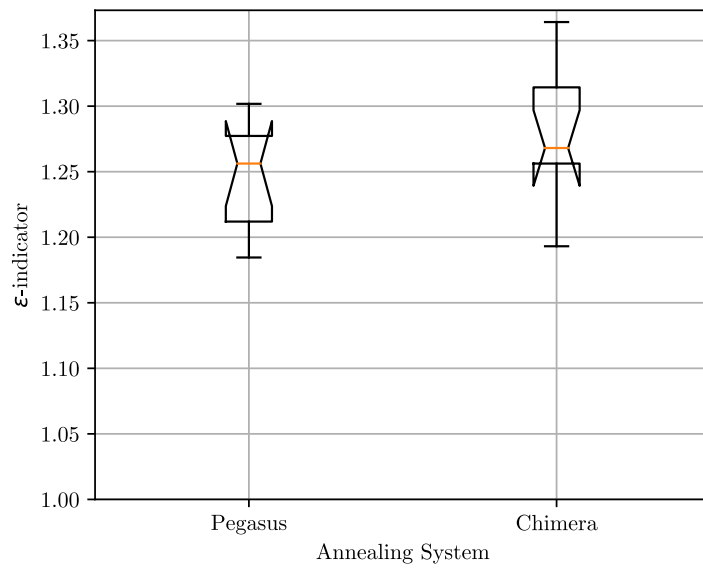
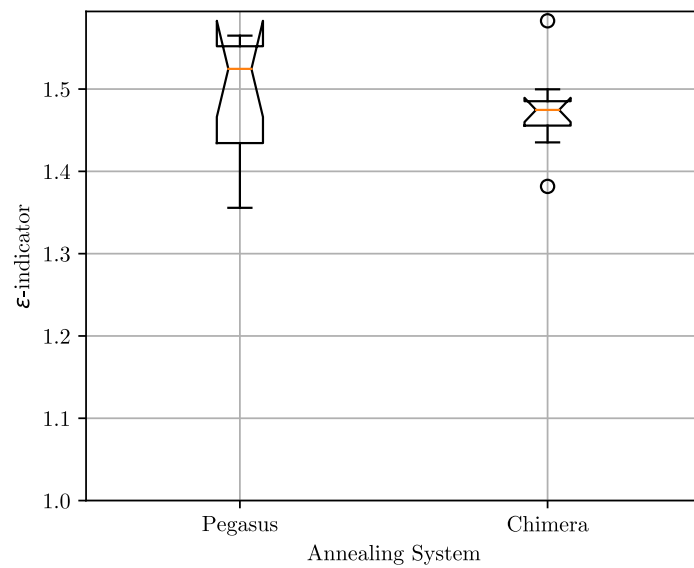
lowest median  $\epsilon$ -indicator.

## 6.9 Annealing System

In this section, we present and analyze the results of Scenario  $\mathcal{S}_9$ , which focuses on the influence of the annealing system parameter (see in Section 4.1.9). Figures 6.21, 6.22 and 6.23 show the boxplots of the  $\epsilon$ -indicator values associated to the results of the strategies that are part of this scenario, for universe sizes 16, 32, and 64, respectively.

Figure 6.21: Boxplot of  $\mathcal{S}_9$ 's results for  $n = 16$ 

The *Shapiro-Wilk test* preserved the null hypothesis for each strategy for all universe sizes, except for  $\mathcal{S}_{9a}$  for  $n = 64$ . The *Levene's test* preserved the null hypothesis for all strategies for all universe sizes. Hence, the assumptions to reliably perform a *one-way analysis of variance* are met for universe sizes 16 and 32. The *one-way analysis of variance* on the results returned a  $p$ -value of 0.515 for  $n = 16$ , and 0.154 for  $n = 32$ , which preserves the null hypothesis that the strategies' results have equal means for universe sizes 32 and 64. The *Kruskal-Wallis H test* on the results returned a  $p$ -value of 0.290 for  $n = 64$ , which preserves

Figure 6.22: Boxplot of  $\mathcal{S}_9$ 's results for  $n = 32$ Figure 6.23: Boxplot of  $\mathcal{S}_9$ 's results for  $n = 64$ 

the null hypothesis that the strategies' results originate from the same distribution for this universe size. These results suggest that the annealing system parameter does not have any significant impact on the quality of the solutions obtained by D-Wave's system, for all universe sizes.

## 6.10 Main Takeaways

With all the scenarios executed and analyzed, we achieved the main objective of our empirical study. We have a clear idea of which parameters have a significant impact

on the quality of the solutions. This section is going to wrap up all the analysis on several paragraphs, one for each parameter, ordered by their impact.

**Varying the universe size has a very strong effect** The universe size is the parameter with the most significant impact we have noticed so far, with  $n = 8$  consistently achieving optimal solutions for some directions and the following universe sizes being increasingly worse in quality. This pattern suggests that an increase in the universe size is accompanied with a problem that is harder to solve. In practical terms, a practitioner should strive for the shortest universe that can solve their problem.

**Varying the chain strength has a very strong effect** In any case, and especially for larger universe sizes, a practitioner needs to take into consideration the chain strength. This parameter, if wrongly set, can significantly lower the quality of the solutions, a reduction that is sometimes of several orders of magnitude. According to our results, we believe that there is a value after which the chain strength is good, with no significant changes expected in the quality of the solutions. Objectively, we found this value to be  $0.125M_Q$ , but there is a possibility that this also varies with the dataset used. We should note that if we go further than  $1.000M_Q$ , we are scaling down the problem, which may not be ideal [16].

**Varying the dataset has a very strong effect** Regarding the dataset parameter, we have determined that the choice of the dataset has a great impact on the quality of the solutions. More specifically, our results suggest that statistically correlated datasets are associated to better quality solutions when compared to statistically diversified datasets. This information can be useful for practitioners to achieve an expected performance given they know how correlated is their universe.

**Varying the budget size has a very strong effect** Another parameter that practitioners should consider is the budget size. Our results suggest that practitioners should expect a reduction in quality from budget sizes that narrow the space of feasible solutions. That is, the larger the difference between the budget size and half the universe size, the harder is the problem to solve. Particularly, D-Wave's system was not able to return any feasible solution for  $n = 64$  and  $B = \lfloor 0.8 \times n \rfloor$ , reflecting the complexity of solving under this cardinality constraint.

**Varying the number of reads has a significant effect** When considering the total number of solutions to read when solving a problem, we believe that practitioners should take as many solutions as possible, without disregarding budget constraints. Our results suggest that increasing the number of reads improves the quality of the solutions, albeit with diminishing returns expected.

**Varying the embedding has close to no effect** One more parameter that did show an effect, albeit weak, particularly for  $n = 16$  and  $n = 32$ , according to our results, is the embedding. We believe that practitioners are well served by D-Wave's default algorithm to generate an embedding for a given annealing system. In fact, for  $n = 16$  and  $n = 32$ , our results suggest that *general* embedding is the best suited. For the largest universe size, our results did not suggest any meaningful improvement or deterioration for the situations where other embedding algorithms are used. Nevertheless, practitioners can quickly assess whether one of the other embedding algorithms can improve their solutions.

**Varying the number of directions has no effect** Another important assessment that practitioners should also do before advancing on a full-fledged execution suite is which and how many directions to take. Although our results do not suggest a significant impact when changing the number of directions, we did not assess the impact of changing the value of the directions. Therefore, this parameter still has the unexamined potential to provoke a significant bias on the solver towards one direction, deteriorating the quality of solutions in opposite directions, and thus the overall performance of the solver.

**Varying the anneal schedule has no effect** There is a parameter that also did not show any significant impact, the anneal schedule. Again, we believe that practitioners are also well served by D-Wave's default anneal schedule. Our results did not suggest any meaningful improvement or deterioration for the situations where other anneal schedules are used. In fact, according to our results, we believe that it is more cost-effective to first increase the number of reads instead of first increasing the annealing time.

**Varying the annealing system has no effect** Last, our findings suggest that there is no particular advantage in choosing one system over other when considering the quality of the solutions. However, before advancing on a full-fledged execution suite to solve their problem, practitioners should quickly assess which annealing system is best suited for the problem, since a *Pegasus* system supports larger problems compared to a *Chimera* system.



This page is intentionally left blank.

## Chapter 7

# Conclusions and Future Work

In the context of Finance, Quantum Computing has applications on many domains. There are contributions demonstrating the use of quantum computers, whether gate-based or adiabatic, to solve the combinatorial optimization problem in which our work focuses.

In our work, we seek to shed light on the hypothesis that parameters related not only to the weighted-sum formulation of the POP, but also to the adiabatic quantum computer, have an effect on the results returned by a D-Wave QPU in problem instances across the different risk appetites that investors may assume. Therefore, in Chapters 4 and 5, we designed and implemented a workflow that establishes scenarios and strategies to individually study each of the identified parameters. In this workflow, we executed each strategy, as a rule, 10 times, obtaining a group of 10 sets of solutions for each strategy. For each set of solutions, we calculated the  $\epsilon$ -indicator value associated with it, ending up with a group of 10  $\epsilon$ -indicator values associated to each strategy. A statistical analysis was performed on those results, studying the main effect of each parameter.

Considering the POP-related parameters, our findings from the results indicate that the universe size, the budget size, and the dataset, which directly change the complexity of the problem, have a significant effect on the results. The results also suggest that the remaining POP-related parameter, directions, does not have a significant effect, when considering the number of directions.

Considering the QPU-related parameters, the chain strength was a parameter that stood out when considering its impact on the quality of the solutions. In this regard, our findings indicate that this parameter, if wrongly set, causes a major performance hit, that cannot be compensated with any other parameter. The findings also suggest that the number of reads and the embedding are parameters with a significant effect on the results: the higher the number of reads, the better the quality of the solutions returned by the QPU; and the general embedding can present the best performance, especially for smaller problem instances ( $n = 16$  and  $n = 32$ ). The remaining parameters, annealing system and anneal schedule, did not present any significant effect, results that support the findings from [26].

One potential direction of future work that can complement our work is related with the existence of any potential interaction between two or more parameters. In this sense, there may exist some interaction between parameters that has a significant effect on the quality of the solutions, which is a hypothesis that is open for future research.

Another direction of future research is related with the impact of the directions parameter. In our work, we studied the impact that the number of directions has on the results. However, we did not study the impact that different values for directions can have. In this

---

sense, there is an opportunity to find if the results have a better quality for any direction in particular, such as risk-prone directions or risk-averse directions.

An additional effect that was not studied in our work was that of the reverse annealing, a procedure that can be performed in D-Wave computers, and that is related to the anneal schedule parameter. This procedure was studied in [26], which found some improvements over the annealing used in our work, suggesting that there may exist room for further improvement with reverse annealing in our work.

# References

- [1] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. Next-Generation Topology of D-Wave Quantum Processors, 2020.
- [2] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2, 2014.
- [3] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.
- [4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villedo, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019.
- [5] 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.
- [6] Accenture. Cryptography in a post-quantum world. Retrieved June 29, 2021, from <https://www.accenture.com/nz-en/insights/technology/quantum-cryptography>.
- [7] Harry Markowitz. Portfolio selection\*. *The Journal of Finance*, 7(1):77–91, 1952.
- [8] Jana Doering, Renatas Kizys, Angel A. Juan, Àngels Fitó, and Onur Polat. Metaheuristics for rich portfolio optimisation and risk management: Current state and future trends. *Operations Research Perspectives*, 6:100121, 2019.
- [9] IBM Quantum Experience. Retrieved June 29, 2021, from <https://quantum-computing.ibm.com/>.

- 
- [10] Héctor Abraham, AduOffei, Rochisha Agarwal, Ismail Yunus Akhalwaya, Gadi Aleksandrowicz, Thomas Alexander, Matthew Amy, Eli Arbel, Arijit02, Abraham Asfaw, Artur Avkhadiev, Carlos Azaustre, AzizNgoueya, Abhik Banerjee, Aman Bansal, Panagiotis Barkoutsos, Ashish Barnawal, George Barron, George S. Barron, Luciano Bello, Yael Ben-Haim, Daniel Bevenius, Arjun Bhubhe, Lev S. Bishop, Carsten Blank, Sorin Bolos, Samuel Bosch, Brandon, Sergey Bravyi, Bryce-Fuller, David Bucher, Artemiy Burov, Fran Cabrera, Padraic Calpin, Lauren Capelluto, Jorge Carballo, Ginés Carrascal, Adrian Chen, Chun-Fu Chen, Edward Chen, Jielun (Chris) Chen, Richard Chen, Jerry M. Chow, Spencer Churchill, Christian Claus, Christian Clauss, Romilly Cocking, Filipe Correa, Abigail J. Cross, Andrew W. Cross, Simon Cross, Juan Cruz-Benito, Chris Culver, Antonio D. Córcoles-Gonzales, Sean Dague, Tareq El Dandachi, Marcus Daniels, Matthieu Dartiailh, DavideFrr, Abdón Rodríguez Davila, Anton Dekusar, Delton Ding, Jun Doi, Eric Drechsler, Drew, Eugene Dumitrescu, Karel Dumon, Ivan Duran, Kareem EL-Safty, Eric Eastman, Grant Eberle, Pieter Eendebak, Daniel Egger, Mark Everitt, Paco Martín Fernández, Axel Hernández Ferrera, Romain Fouilland, FranckChevallier, Albert Frisch, Andreas Fuhrer, Bryce Fuller, MELVIN GEORGE, Julien Gacon, Borja Godoy Gago, Claudio Gambella, Jay M. Gambetta, Adhisha Gammanpila, Luis Garcia, Tanya Garg, Shelly Garion, Austin Gilliam, Aditya Giridharan, Juan Gomez-Mosquera, Gonzalo, Salvador de la Puente González, Jesse Gorzinski, Ian Gould, Donny Greenberg, Dmitry Grinko, Wen Guan, John A. Gunnels, Mikael Haglund, Isabel Haide, Ikko Hamamura, Omar Costa Hamido, Frank Harkins, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Stefan Hillmich, Hiroshi Horii, Connor Howington, Shaohan Hu, Wei Hu, Junye Huang, Rolf Huisman, Haruki Imai, Takashi Imamichi, Kazuaki Ishizaki, Raban Iten, Toshinari Itoko, JamesSeaward, Ali Javadi, Ali Javadi-Abhari, Wahaj Javed, Jessica, Madhav Jivrajani, Kiran Johns, Scott Johnstun, Jonathan-Shoemaker, Vismai K, Tal Kachmann, Akshay Kale, Naoki Kanazawa, Kang-Bae, Anton Karazeev, Paul Kassebaum, Josh Kelso, Spencer King, Knabberjoe, Yuri Kobayashi, Arseny Kovyrshin, Rajiv Krishnakumar, Vivek Krishnan, Kevin Krsulich, Prasad Kumkar, Gawel Kus, Ryan LaRose, Enrique Lacal, Raphaël Lambert, John Lapeyre, Joe Latone, Scott Lawrence, Christina Lee, Gushu Li, Dennis Liu, Peng Liu, Yunho Maeng, Kahan Majmudar, Aleksei Malyshev, Joshua Manela, Jakub Marecek, Manoel Marques, Dmitri Maslov, Dolph Mathews, Atsushi Matsuo, Douglas T. McClure, Cameron McGarry, David McKay, Dan McPherson, Srujan Meesala, Thomas Metcalfe, Martin Mevisen, Andrew Meyer, Antonio Mezzacapo, Rohit Midha, Zlatko Minev, Abby Mitchell, Nikolaj Moll, Jhon Montanez, Gabriel Monteiro, Michael Duane Mooring, Renier Morales, Niall Moran, Mario Motta, MrF, Prakash Murali, Jan Müggenburg, David Nadlinger, Ken Nakanishi, Giacomo Nannicini, Paul Nation, Edwin Navarro, Yehuda Naveh, Scott Wyman Neagle, Patrick Neuweiler, Johan Nicander, Pradeep Niroula, Hassi Norlen, NuoWenLei, Lee James O’Riordan, Oluwatobi Ogunbayo, Pauline Ollitrault, Raul Otaolea, Steven Oud, Dan Padilha, Hanhee Paik, Soham Pal, Yuchen Pang, Vincent R. Pascuzzi, Simone Perriello, Anna Phan, Francesco Piro, Marco Pistoia, Christophe Piveteau, Pierre Pocreau, Alejandro Pozas-Kerstjens, Milos Prokop, Viktor Prutyaynov, Daniel Puzzuoli, Jesús Pérez, Quintiii, Rafey Iqbal Rahman, Arun Raja, Nipun Ramagiri, Anirudh Rao, Rudy Raymond, Rafael Martín-Cuevas Redondo, Max Reuter, Julia Rice, Matt Riedemann, Marcello La Rocca, Diego M. Rodríguez, RohithKarur, Max Rossmannek, Mingi Ryu, Tharmashastha SAPV, SamFerracin, Martin Sandberg, Hirmay Sandesara, Ritvik Sapra, Hayk Sargsyan, Anirudha Sarkar, Ninad Sathaye, Bruno Schmitt, Chris Schnabel, Zachary Schoenfeld, Travis L. Scholten, Eddie Schoute, Joachim Schwarm, Ismael Faro Sertage, Kanav Setia, Nathan Shammah, Yunong Shi, Adenilton Silva, Andrea Simonetto, Nick Sing-

- stock, Yukio Siraichi, Iskandar Sitdikov, Seyon Sivarajah, Magnus Berg Sletfjerding, John A. Smolin, Mathias Soeken, Igor Olegovich Sokolov, Igor Sokolov, SooluThomas, Starfish, Dominik Steenken, Matt Stypulkoski, Shaojun Sun, Kevin J. Sung, Hitomi Takahashi, Tanvesh Takawale, Ivano Tavernelli, Charles Taylor, Pete Taylour, Soolu Thomas, Mathieu Tillet, Maddy Tod, Miroslav Tomasik, Enrique de la Torre, Kenso Trabing, Matthew Treinish, TrishaPe, Davindra Tulsi, Wes Turner, Yotam Vaknin, Carmen Recio Valcarce, Francois Varchon, Almudena Carrera Vazquez, Victor Villar, Desiree Vogt-Lee, Christophe Vuillot, James Weaver, Johannes Weidenfeller, Rafal Wieczorek, Jonathan A. Wildstrom, Erick Winston, Jack J. Woehr, Stefan Woerner, Ryan Woo, Christopher J. Wood, Ryan Wood, Stephen Wood, Steve Wood, James Wootton, Daniyar Yeralin, David Yonge-Mallo, Richard Young, Jessie Yu, Christopher Zachow, Laura Zdanski, Helena Zhang, Christa Zoufal, Zoufal, a kapila, a matsuo, bcamorrison, brandhsn, nick bronn, brosand, chlorophyll zz, cs-seifms, dekel.meirom, dekelmeirom, dekool, dime10, drholmie, dtrenev, ehchen, elfrocampeador, faisaldebouni, fanizzamarco, gabrieleagl, gadial, galeinston, georgios ts, gruu, hhorii, hykavitha, jagunther, jliu45, jscott2, kanejess, klinvill, krutik2966, kurarr, lerongil, ma5x, merav aharoni, michelle4654, ordmoj, sagar pahwa, rmoyard, saswati qiskit, scottkelso, sethmerkel, shaashwat, sternparky, strickroman, sumitpuri, tigerjack, toural, tsura crisaldo, vvilpas, welien, willhbang, yang.luh, yotamvakninibm, and Mantas Čepulkovskis. Qiskit: An open-source framework for quantum computing, 2019.
- [11] Petar Jurcevic, Ali Javadi-Abhari, Lev S. Bishop, Isaac Lauer, Daniela F. Bogorin, Markus Brink, Lauren Capelluto, Oktay Günlük, Toshinari Itoko, Naoki Kanazawa, Abhinav Kandala, George A. Keefe, Kevin Krsulich, William Landers, Eric P. Lewandowski, Douglas T. McClure, Giacomo Nannicini, Adinath Narasgond, Hasan M. Nayfeh, Emily Pritchett, Mary Beth Rothwell, Srikanth Srinivasan, Neereja Sundaresan, Cindy Wang, Ken X. Wei, Christopher J. Wood, Jeng-Bang Yau, Eric J. Zhang, Oliver E. Dial, Jerry M. Chow, and Jay M. Gambetta. Demonstration of quantum volume 64 on a superconducting quantum computing system, 2020.
- [12] Tosio Kato. On the Adiabatic Theorem of Quantum Mechanics. *Journal of the Physical Society of Japan*, 5(6):435–439, 1950.
- [13] Raouf Dridi, Hedayat Alghassi, and Sridhar Tayur. Enhancing the efficiency of adiabatic quantum computations, 2019.
- [14] Elijah Pelofske, Georg Hahn, and Hristo Djidjev. Decomposition Algorithms for Solving NP-hard Problems on a Quantum Annealer. *Journal of Signal Processing Systems*, Jun 2020.
- [15] D-Wave Systems, Inc. D-Wave System Documentation documentation. Retrieved June 29, 2021, from [https://docs.dwavesys.com/docs/latest/c\\_gs\\_1.html](https://docs.dwavesys.com/docs/latest/c_gs_1.html).
- [16] D-Wave Systems, Inc. Programming the D-Wave QPU: Setting the Chain Strength. Retrieved May 29, 2021, from [https://www.dwavesys.com/sites/default/files/14-1041A-A\\_Setting\\_The\\_Chain\\_Strength.pdf](https://www.dwavesys.com/sites/default/files/14-1041A-A_Setting_The_Chain_Strength.pdf).
- [17] A. Gaspar-Cunha, R. Takahashi, and C.H. Antunes. *Manual de computação evolutiva e metaheurística*. Ensino. Imprensa da Universidade de Coimbra / Coimbra University Press, 2012.
- [18] Mark W. Coffey. Adiabatic quantum computing solution of the knapsack problem, 2017.

- 
- [19] Frank Phillipson and Harshil Singh Bhatia. Portfolio Optimisation Using the D-Wave Quantum Annealer, 2020.
- [20] Jonathan Law. *A Dictionary of Finance and Banking*. Oxford University Press, 2014.
- [21] Matthias Ehrgott. Multiobjective Optimization. *AI Magazine*, 29:47–57, 12 2008.
- [22] D-Wave Systems, Inc. Quantum Programming 101: Solving a Problem From End to End | D-Wave Webinar.
- [23] Daniel J. Egger, Claudio Gambella, Jakub Marecek, Scott McFaddin, Martin Mevisen, Rudy Raymond, Andrea Simonetto, Stefan Woerner, and Elena Yndurain. Quantum Computing for Finance: State-of-the-Art and Future Prospects. *IEEE Transactions on Quantum Engineering*, 1:1–24, 2020.
- [24] Samuel Mugel, Carlos Kuchkovsky, Escolastico Sanchez, Samuel Fernandez-Lorenzo, Jorge Luis-Hita, Enrique Lizaso, and Roman Orus. Dynamic Portfolio Optimization with Real Datasets Using Quantum Processors and Quantum-Inspired Tensor Networks, 2020.
- [25] Frank Phillipson and Harshil Singh Bhatia. Portfolio Optimisation Using the D-Wave Quantum Annealer, 2020.
- [26] Erica Grant, Travis S. Humble, and Benjamin Stump. Benchmarking Quantum Annealing Controls with Portfolio Optimization. *Phys. Rev. Applied*, 15:014012, Jan 2021.
- [27] Miqing Li and Xin Yao. Quality Evaluation of Solution Sets in Multiobjective Optimisation: A Survey. *ACM Computing Surveys (CSUR)*, 52(2):1–38, 2019.
- [28] Robert McGill, John W. Tukey, and Wayne A. Larsen. Variations of Box Plots. *The American Statistician*, 32(1):12–16, 1978.
- [29] Ronald L. Iman and W. J. Conover. The Use of the Rank Transform in Regression. *Technometrics*, 21(4):499–509, 1979.
- [30] The pandas development team. pandas-dev/pandas: Pandas 1.2.4. <https://doi.org/10.5281/zenodo.4681666>, April 2021.
- [31] yfinance: Yahoo! Finance market data downloader. Retrieved June 29, 2021, from <https://github.com/ranaroussi/yfinance>.
- [32] S&P 500® - S&P Dow Jones Indices. Retrieved May 25, 2021, from <https://www.spglobal.com/spdji/en/indices/equity/sp-500/>.
- [33] GICS - Global Industry Classification Standard. Retrieved May 25, 2021, from <https://www.msci.com/gics>.
- [34] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. Technical report, Optimization Online, March 2020.

- 
- [35] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin, March 2020.
- [36] Andrew Makhorin. GLPK (GNU Linear Programming Kit). Retrieved May 29, 2021, from <https://www.gnu.org/software/glpk/>.
- [37] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.
- [38] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.



This page is intentionally left blank.

# Appendices

This page is intentionally left blank.

---

## Appendix A: Implementation and Execution of Quantum Entanglement using Qiskit

In this appendix, we will implement and execute the quantum circuit described in Figure 2.2 using the Qiskit framework [10]. Before following the next steps, we need to create an account in the IBM Quantum Experience platform [9] and open a Jupyter Notebook there. The Jupyter Notebook is a special Python environment in the cloud that has most of the necessary tools installed, such as Qiskit.

**Step 1** We need to import the necessary tools and load the IBM Q account in order to gain access to real quantum devices. Usually, this step is automatically done when opening a new Jupyter Notebook. Figure 1 shows the code to import such tools and load the IBM Q account.

```
1 # Import necessary tools:
2 from qiskit import QuantumCircuit, execute, Aer, IBMQ
3 from qiskit.compiler import transpile, assemble
4 from qiskit.tools.jupyter import *
5 from qiskit.visualization import *
6 from iqx import *
7
8 # Load IBM Q account:
9 provider = IBMQ.load_account()
```

Figure 1: Python code to import the necessary tools and load the IBM Q account

**Step 2** Now, we can setup a quantum circuit to be executed. First, we create a quantum circuit with two quantum registers and two classical registers. This means that we have two qubits whose measurement can be registered in two classical bits. Next, we add a Hadamard gate into the first qubit. Afterwards, we add a controlled  $X$  gate which is controlled by the first qubit and targets the second qubit. Finally, we measure both qubits and save the result into the classical registers. Figure 2 shows the code to setup this circuit.

```
1 # Setup a circuit:
2 circuit = QuantumCircuit(2, 2)
3 circuit.h(0)
4 circuit.cx(0, 1)
5 circuit.measure(0, 0)
6 circuit.measure(1, 1)
```

Figure 2: Python code to setup a quantum circuit

**Step 3** This step is optional. We can draw the circuit using the code in Figure 3. Figure 4 shows the result of this function in our example code.

```
1 # Print the circuit:
2 circuit.draw()
```

Figure 3: Python code to draw the quantum circuit

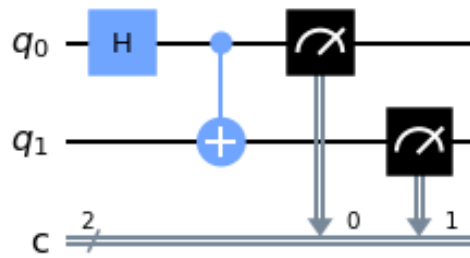


Figure 4: Drawing of the quantum circuit implemented in Qiskit

**Step 4** To execute the quantum circuit we first need to retrieve a quantum device from the provider created in Step 1. IBM provides open access to several devices, as listed in their platform. Each device has a unique string identifier which must be used to retrieve it. In this example, we will use the IBM Q Athens computer, identified as `ibmq_athens`. Afterwards, we must specify the number of *shots*, i.e., the number of times the device will execute the quantum circuit. We can then pass the quantum circuit and the number of shots to the device, which will then enqueue this job and execute it on its turn. Once the job is executed, its results are collected and can be processed as we wish. In this example, we plot the results on a histogram. Figure 5 shows the code for this step and Figure 6 shows the histogram for the results of this example.

```

1 # Execute the circuit and collect results:
2 real_device = provider.get_backend('ibmq_athens')
3 shots = 4096
4 results = execute(circuit, backend=real_device, shots=shots).result()
5 answer = results.get_counts()
6 plot_histogram(answer)

```

Figure 5: Python code to run the circuit on a real quantum device and plot the results

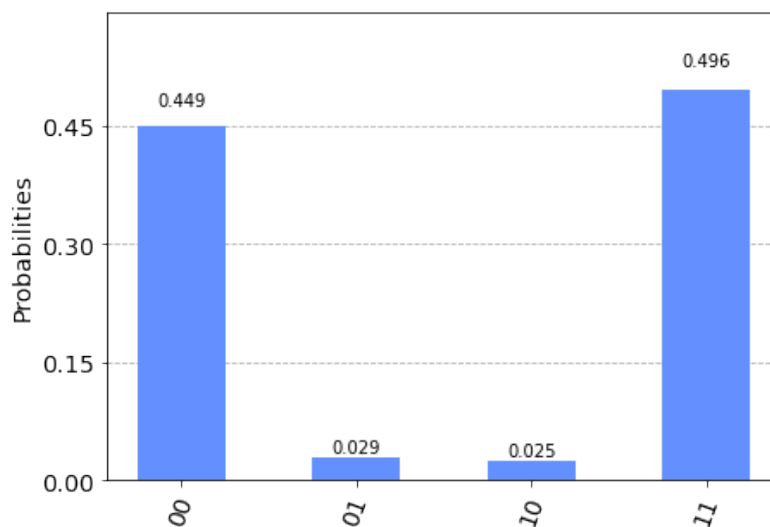


Figure 6: Results of the quantum circuit implemented in Qiskit

And that is it. With this example, we have successfully demonstrated quantum entanglement in a real quantum device!

---

An attentive reader may notice that in Figure 6, the results are very close to what is expected, but find odd that unexpected results also appear (only 00 and 11 should be measured). At the moment, quantum computers have a high error rate, which translates into some outliers when executing jobs. Hence, advanced and complex algorithms must take into account error mitigation strategies in order to get the maximum effectiveness from the hardware.

Those five steps were sufficient for this small circuit. Bigger circuits would have more steps and complicated functions, which requires access to Qiskit's documentation. Nonetheless, this example should give a grasp of the general procedure to implement and execute a quantum circuit.

This page is intentionally left blank.

---

## Appendix B: Implementation and Execution of a Quadratic Unconstrained Binary Optimization in a D-Wave Computer

In this appendix, we will implement and execute a problem formulated in QUBO in a D-Wave computer. the Qiskit framework [10]. Before following the next steps, we need to create an account in the D-Wave Leap platform and open an IDE workspace. The workspace is a virtual machine in the cloud that has most of the necessary tools installed, such as D-Wave Ocean Tools. In this workspace, we create a Python file where our code will be stored.

**Step 1** We need to import the necessary tools in order to create a QUBO and gain access to the D-Wave QPU. Figure 7 shows the tool necessary for this example.

```
1 from collections import defaultdict
2 from dwave.system import DWaveSampler, AutoEmbeddingComposite
```

Figure 7: Python code to import the necessary tools to create a QUBO and gain access to the D-Wave QPU

**Step 2** The next step is to prepare the QUBO matrix of the problem. In this example, we are preparing the matrix shown in Section 2.5.3. Figure 8 shows the code to prepare this matrix.

```
1 # Prepare QUBO
2 Q = defaultdict(float)
3
4 Q[(1,1)] = 1
5 Q[(1,2)] = 7
6 Q[(1,3)] = 3
7 Q[(1,4)] = 1
8
9 Q[(2,2)] = 2
10 Q[(2,3)] = 7
11 Q[(2,4)] = 0
12
13 Q[(3,3)] = 8
14 Q[(3,4)] = 4
15
16 Q[(4,4)] = 9
```

Figure 8: Python code to prepare a QUBO matrix

**Step 3** The last step is to execute the QUBO on the D-Wave QPU. Figure 9 shows the code to initialize a QPU and use it to execute the QUBO.



---

```

1 # Prepare QPU solver
2 composite = AutoEmbeddingComposite(DWaveSampler())
3
4 # Execute QUBO on solver
5 sampleset = composite.sample_qubo(Q, num_reads=1000)
6
7 # Print results
8 print(sampleset)

```

Figure 9: Python code to execute a QUBO matrix on the D-Wave QPU

The `AutoEmbeddingComposite` is responsible for embedding, if needed, the problem to the QPU. Afterwards, we use function `sample_qubo` to execute the QUBO matrix and read 1000 solutions. The last line of the code shows the results of the execution. One possible output of the print is

```

  1  2  3  4 energy num_oc. chain_.
0  0  0  0  0   0.0    797   0.0
1  1  0  0  0   1.0    176   0.0
2  0  1  0  0   2.0     27   0.0
['BINARY', 3 rows, 1000 samples, 4 variables]

```

where each row shows a solution that was found by the QPU, as well as its associated energy, number of occurrences and chain breaks. In this example, assuming that we want to minimize the energy, as the QPU does, the best solution found corresponds to setting all the four decision variables to zero ( $x_i = 0$ ), and was read 797 times.

These three steps were sufficient for this small problem. Bigger circuits would have more steps and complicated functions, which requires access to D-Wave's documentation. Nonetheless, this example should give a grasp of the general procedure to implement and execute a problem on D-Wave machines.