Imperial College London

Department of Mathematics

Arbitrage Detection with Quantum Annealing

Author: Etienne Wallerich (CID: 02104133)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2021-2022

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Acknowledgements

Firstly, I would like to thank my thesis supervisor Dr Antoine Jacquier for his precious advice all along this research project. His teaching in Quantum Computing, along with Dr Alexei Kondratyev, and his experience were a source of inspiration and motivation from the beginning.

My sincere gratitude goes as well to all the Imperial College teaching staff of the MSc in Mathematics and Finance for this rewarding year and for all the resources provided, allowing us to achieve our goals and even more.

Finally, I would like to thank my parents Denis and Laurence for giving me the opportunity to study in such a leading university. They always believed in me and provided me comfort and strength to help me when I needed. I will always be grateful to have you in my life.

Abstract

NP-complete and NP-Hard problems cannot be solved in polynomial time and usually require the use of heuristics to get approximated solutions. The progresses in quantum computing and quantum optimization over the last decade have allowed the use of innovative search policies. Quantum Annealing is one of these heuristics, derived from Simulated Annealing, and relying on the exploitation of the Adiabatic Quantum Theorem and quantum tunneling effects. The arbitrage detection problem for European Call Options can be stated in a binary integer programming which is NP-Complete. It can be translated into a specific Quadratic Unconstrained Binary Optimization form suited for the D-Wave quantum annealers. Finally, the embedding performed by the D-Wave Advantage used in this thesis encodes the problem into a Hamiltonian with a ground energy level corresponding to the problem's solution. Samples with the lowest energy level can be examined as approximations of the optimal solution.

Keywords: *Quantum Annealing, Binary Integer Programming, NP-complete, Arbitrage Detection, QUBO, Hamiltonian, Adiabatic Quantum Computing.*

Contents

1 Arbitrage Detection for Call Options 9

1.1 Description of the Call Options Portfolio . . . 9

1.2 Arbitrage Definition and Example . . . 10

1.2.1 Arbitrage Definition . . . 10

1.2.2 Arbitrage Example . . . 10

1.2.3 Link with Asset Pricing Theory . . . 11

1.3 Integer Programming . . . 12

1.3.1 Optimization Problem . . . 12

1.3.2 Reduction of constraints number . . . 12

1.4 NP-completeness of the problem . . . 14

2 Quantum Annealing Theoretical Overview 16

2.1 Quantum Computing Basics . . . 16

2.1.1 Quantum Binary Digits . . . 16

2.1.2 Schrödinger Equation . . . 17

2.1.3 Measurement . . . 18

2.2 Quantum Annealing Principles . . . 18

2.2.1 Simulated Annealing . . . 18

2.2.2 Adiabatic Quantum Computing . . . 19

2.2.3 Quantum Annealing . . . 20

2.3 D-Wave Quantum Annealer . . . 22

2.3.1 Physical Description . . . 22

2.3.2 QPU Architecture . . . 23

2.3.3 D-Wave Advantage and Pegasus topology . . . 24

3 Problem Translation and Implementation 26

3.1 QUBO Representation . . . 26

3.1.1 QUBO Definition . . . 26

3.1.2 Introduction of Slack Variables . . . 27

3.1.3 Canonical form of QUBO . . . 29

3.2 Implementation . . . 31

3.2.1 Data Used . . . 31

3.2.2 Ocean package . . . 31

3.2.3 Setting QUBO Hyperparameters . . . 32

4 Results and Interpretation **33**
4.1 Sampling Solutions 33
4.2 Metrics for Quantum Annealing performance 34
4.3 Results 34
4.3.1 Accuracy of Solutions 34
4.3.2 Time Performance 40
4.3.3 QPU layout and limits 41

A Results **45**
A.1 Samples Distributions 46
A.2 Solution's payoff for $N = 92$ 47

Bibliography **48**

List of Figures

1.1 Payoff Diagrams for Option 1, Short (top left plot), Option 2, Long (top right plot) and portfolio **x** (bottom plot) . . . 11

2.1 Time Evolution of the Quantum State's Energy during an Adiabatic Process . . . 20
2.2 Adiabatic Evolution from Initial to Final Hamiltonian during Quantum Annealing 21
2.3 Quantum Annealing and Tunnel Effect . . . 21
2.4 D-Wave 2000Q (L:10', W:7', H:10') . . . 22
2.5 D-Wave 2000Q major components . . . 23
2.6 On the left, general grid architecture of qbits, on the right, the two possible representations for a chimera unit cell (vertical columns or cross) . . . 23
2.7 On the left, general grid architecture external coupling, on the right, zoom on unit cells external coupling . . . 24
2.8 On the left, a zoom on a 2×2 unit cells structure, On the right, chimera graph implemented on D-Wave 2000Q systems . . . 24
2.9 Pegasus Topology implemented on D-Wave Advantage system . . . 25

4.1 $N = 16$: Payoffs of the ground solutions find by Simulated Annealing (top) and Quantum Annealing (bottom). Respective prices of the corresponding portfolios are -0.8 and -0.8 . . . 35
4.2 $N = 20$: Payoffs of the ground solutions find by Simulated Annealing (top) and Quantum Annealing (bottom). Respective prices of the corresponding portfolios are -13.5 and -0.23. . . . 36
4.3 $N = 32$: Payoffs of the ground solutions find by Simulated Annealing (top) and Quantum Annealing (bottom). Respective prices of the corresponding portfolios are 5.5 and -3.79 . . . 37
4.4 $N = 52$: Payoffs of the ground solutions find by Simulated Annealing (top) and Quantum Annealing (bottom). Respective prices of the corresponding portfolios are 15.5 and 15.24 . . . 37
4.5 Samples Energy Distribution for $N = 16$ (upper), $N = 32$ (middle) and $N = 92$ (bottom). X-axis: energy level, Y-axis: sample occurrences . . . 39
4.6 Left plot: QPU Sampling times provided by the D-Wave Problem Inspector . . . 40
4.7 Right plot: Time Comparison between Quantum Annealing and Simulated Annealing 40
4.8 QPU layout with Pegasus topology for: $N = 16$ (top left), $N = 32$ (top right), $N = 52$ (bottom left), $N = 92$ (bottom right) . . . 41
4.9 Left plot: Number of Qbits used vs Number of problem variables . . . 42

4.10 Right plot: Square-root of Number of Qbits used vs Number of problem variables (with regression line) 42

A.1 Samples Energy Distribution, from top to bottom: $N = 16$, $N = 20$, $N = 32$, $N = 52$, $N = 92$. X-axis: energy level, Y axis: sample occurrences 46

A.2 $N = 92$: Payoffs of the ground solutions find by Simulated Annealing (top) and Quantum Annealing (bottom). Respective prices of the corresponding portfolios are 28.38 and 52.94 47

List of Tables

3.1 Data Used on European SPY Call Options from CBOE website 31
3.2 QUBO Hyperparameters 32

4.1 Global Results 35
4.2 Number of Qbits used vs Number of source/problem variables 41

Introduction

The Efficient Market Hypothesis stated by Eugene Fama in 1970 [1] assumes that securities have a fair price which reflects all information available at each moment. However, in some cases, this efficiency can show some weaknesses, especially when liquidity is low, leading to short-lived opportunities called arbitrages. An arbitrage can be simply defined as an investment which offers a loss with a null probability and a gain with a positive probability. Its detection therefore arouse strong interests of the investors and different methods have already been explored for arbitrage research on the market. The methods used usually depends on the nature of the arbitrage that aims to be detected and the underlying asset. Whereas merger arbitrage [2] would focus on trying to anticipate some corporate actions, currencies arbitrage detection would use more quantitative tools to search graphs and cycles in the FX market [3].

Indeed, the arbitrage detection can result into complex mathematical problems which could require sophisticated methods to be solved, when it is possible. This thesis deals with an arbitrage detection problem on a portfolio constituted exclusively with European Call Options that will be represented by a vector of binary decision variables associated to each Call Option. The full description of the mathematical variables and the optimization problem associated will be presented in Chapter 1. It is intuitive that the complexity of this kind of problem is likely to grow exponentially with its size. Results from [4] classifies it as NP-complete as a binary integer programming problem and justifies the use of stochastic search policies rather than deterministic methods.

Quantum Annealing is one example of meta-heuristic that could provide promising results with this kind of NP problems. It is a search policy derived from the classical Simulated Annealing, itself inspired of the thermal annealing processes, and enhanced with properties inherited from quantum mechanics. Recent progresses in quantum computing have seen companies like D-Wave releasing systems that are able to fulfill the physical conditions necessary to observe quantum effects and perform quantum annealing. The functioning of those quantum annealers will be detailed in Chapter 2.

To use D-Wave systems and Quantum Annealing to solve the arbitrage detection problem, it has to be stated into a particular quadratic, unconstrained and binary form that is well tolerated by the tools allowing a remote connection to quantum annealers. Then, an embedding is performed and encodes the constraints of the problem into a Hamiltonian. Finding the solution of the problem will become equivalent to finding the ground energy level of the Hamiltonian. Therefore, a translation work has to be done and will be detailed in Chapter 3 where the constraints will be carefully encoded and the implementation methods and parameters will be presented.

Finally, Chapter 4 will expose the different results, assessing especially whether the quantum algorithm manages to detect arbitrage or not. Its performance in terms of accuracy and speed will be compared to a benchmark using a classic search policy. The last step will be to discuss the potential limitations of Quantum Annealing and more generally of quantum computing.

To sum up, the main aim of the thesis is to translate the arbitrage detection problem for European Call Options and solve it with Quantum Annealing. The secondary goal is to give a more general example about how a NP-complete problem can be handled by this quantum meta-heuristic and what are the prerequisites to use it.

Chapter 1

Arbitrage Detection for Call Options

This first chapter aims to set up the mathematical formulation of the Arbitrage Detection problem to solve. Firstly, some definitions of arbitrage will be given as well as the notation used to describe and quantify it. Then some simple example will be provided to illustrate it in a practical application. Finally, from the starting point which is a financial problem, the arbitrage detection in a portfolio of call options will be translated into a mathematical version as an optimization problem.

1.1 Description of the Call Options Portfolio

Assume n European Call Options on a common underlying risky asset S, denoted S_T at maturity, are given with a same maturity T and different strikes K_i, with $1 \leq i \leq n$, such that the order $K_1 < K_2 < \cdots < K_n$ is assumed. A portfolio with binary variables can be built using a vector $\mathbf{x} = (x_1, ..., x_n)^\intercal$ constituted only with 0 and 1 values. If the Option i is kept in the portfolio, the i^{th} coordinate of the vector $\mathbf{x}$ is worth 1, and 0 otherwise. To model the distinction of long/short position, the classical payoff functions will include a multiplicative factor α_i whether the payoff corresponds to a long or short position for the Option. There will be no restriction associated to short-selling. The respective payoff functions at maturity are denoted $\psi_i(.)$ are:

$$\psi_i(S_T) = \alpha_i(S_T - K_i)^+$$

where α_i is equal to $+1$ or -1.
In this document, Call Options portfolios will have in scope both long and short position for each strike K_i. This implies that for n selected strikes, the total number of options in the scope is $2n$. For more clarity, n will be referred as the total number of options, which is also the number of decision variables in the following. This representation is used to keep the problem simple and to already have a binary formulation which will be useful later.
Thus, the portfolio's payoff at maturity $\Pi^{\mathbf{x}}$ can be easily computed by adding all payoff functions weighted by the x_i:

$$\Pi^{\mathbf{x}}(S_T) = \sum_{i=1}^{n} x_i \psi_i(S_T) = \mathbf{x}^\intercal \psi(S_T)$$

where ψ is the vector of the payoff functions ψ_i. $\Pi^{\mathbf{x}}$ is a function of S_T interpreted directly as the valuation at maturity of the portfolio.
Moreover, the initial cost $\Pi_0^{\mathbf{x}}$ of the portfolio is defined as the sum of the costs of each Option Π_0^i times the decision variable x_i:

$$\Pi_0^{\mathbf{x}} = \sum_{i=1}^{n} x_i \Pi_0^i = \mathbf{x}^\mathsf{T} \Pi_0$$

where Π_0 is the vector of all Call Options initial cost Π_0^i. It is noted here that the initial cost Π_0^i for a short position in an Option, with therefore a $\alpha_i = -1$ coefficient in its payoff function, would likely be negative

1.2 Arbitrage Definition and Example

1.2.1 Arbitrage Definition

From there, a definition can be given for the two main types of arbitrage that can be encountered:[5]

- Arbitrage of type A:

  A portfolio described by a binary vector $\mathbf{x}$ is an arbitrage of type A if $\Pi_0^{\mathbf{x}} = 0$ and $\Pi^{\mathbf{x}}(S_T) > 0$ for any S_T.

- Arbitrage of type B:

  A portfolio described by a binary vector $\mathbf{x}$ is an arbitrage of type B if $\Pi_0^{\mathbf{x}} < 0$ and $\Pi^{\mathbf{x}}(S_T) \geq 0$ for any S_T.

1.2.2 Arbitrage Example

Let's give now a concrete example about an arbitrage opportunity for a portfolio with size $n = 2$ composed of European Call Options and described with a vector $\mathbf{x}$ with only 0 and 1. This simple example only aims to illustrate how the arbitrage phenomenon can occur and does not aim to be realistic. It is important to recall that arbitrages are not frequent and natural for markets as they translate a short-lived lack of efficiency.

Let's assume an agent has access to a market with the two following European Call Options:

	Position	Price	Maturity	Strike	Underlying Stock
Option 1	Short	10	1 year	100	XYZ
Option 2	Long	10	1 year	90	XYZ

Here all transaction fees and differences between bid and ask prices are ignored to keep the example simple. In this example, both options have the same price, maturity and underlying asset but different strikes, which offers an arbitrage opportunity. Indeed, if the agent buys one unit of Option 2 and sells one unit of Option 1, or, in other words, if he builds a portfolio with the vector $\mathbf{x} = (1, 1)^\mathsf{T}$, it leads to a 0 cost portfolio and the two following respective payoffs at maturity:

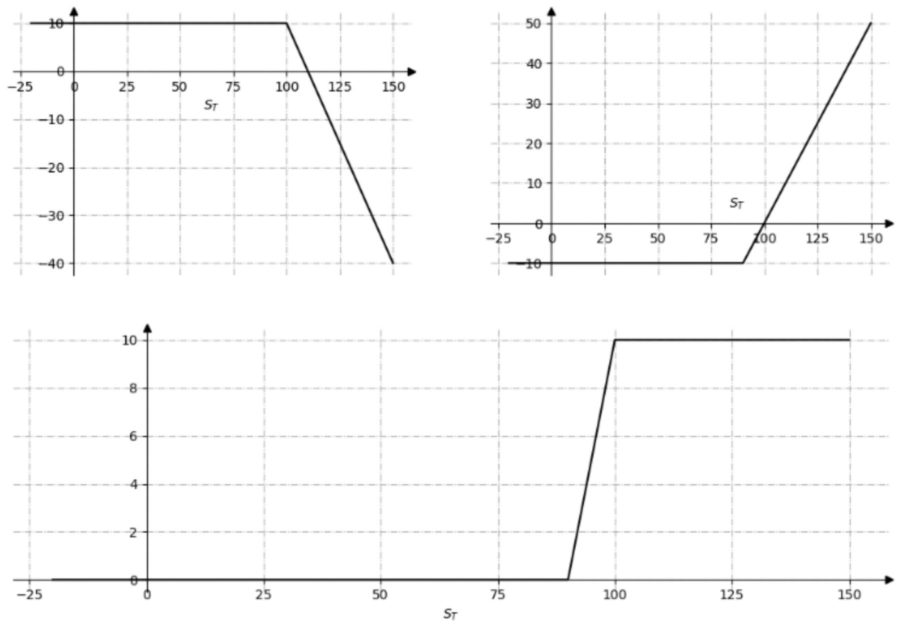


Figure 1.1: Payoff Diagrams for Option 1, Short (top left plot), Option 2, Long (top right plot) and portfolio **x** (bottom plot)

Here, the arbitrage opportunity is clearly illustrated in the bottom plot of the portfolio **x** payoff, in which there is a non-negative value anywhere for the payoff at maturity. Therefore, the portfolio never suffers a loss and offers gain in some scenarios, when the underlying asset is above 90. In other words, there is a null probability to suffer a loss and a positive probability to earn gains.

Remark 1.2.1. Finally, an important remark here is that, in the case one can only sell or buy 1 unit for each option, the value of the payoff function at $S_T = K_i$, where K_i is the strike of any option in the portfolio, will be a multiple of the gap $K_i - K_{i-1}$, considering this gap is the same among all the strikes. In a case where one would like to keep the values of the payoff function close to 0, a solution would be to re-scale the strikes to tighten the gap between them.

1.2.3 Link with Asset Pricing Theory

A parallel can be made with some results of the Asset Pricing theory, for example using definitions of super-replicating portfolio.
Let Y be a derivative with a certain payoff at maturity Y_T and **x** a portfolio with a payoff at maturity X_T which aims to replicate the derivative. It said that **x** super-replicates the derivative Y if for every possible outcome, $X_T \geq Y_T$.
Thus, this arbitrage detection problem has for solution a portfolio **x**, with a non-positive initial cost, which super-replicates a derivative Y_0 that has a null payoff at maturity.

1.3 Integer Programming

Now that the financial problem is clearly defined and that the different notations were introduced, it is possible to translate it as a mathematical problem. Indeed, the first step of the arbitrage detection solving with quantum annealing is to formulate the problem with a Binary Integer Programming problem, which is an optimization problem that will be consistent with the method used further. Notations and assumptions introduced in Section 1.2.1 will be used.

1.3.1 Optimization Problem

Let's first convert the arbitrage detection problem into a very basic constrained optimization problem. Let $\mathbf{x}$ be a vector of binary variables describing a portfolio of n European Call Options and with initial cost $\Pi_0^{\mathbf{x}} = \mathbf{x}^\mathsf{T}\Pi_0$. For remind, its valuation at maturity is:

$$\Pi^x(S_T) = \mathbf{x}^\mathsf{T}\psi(S_T) = \sum_{i=1}^{n} x_i\psi_i(S_T) = \sum_{i=1}^{n} x_i\alpha_i(S_T - K_i)^+$$

The binary optimization problem which directly translates the arbitrage of type B is the following:

$$\begin{aligned} &\text{minimize} && \mathbf{x}^\mathsf{T}\Pi_0 \\ &\text{subject to} && \Pi^{\mathbf{x}}(S_T) \geq 0, \quad S_T \in \mathcal{S} \end{aligned} \tag{1.3.1}$$

where $\mathcal{S}$ is the set of all possible values for S_T.

From here, and with the definitions provided in the previous section, it is clear that negative solution will represent an arbitrage of type B opportunity. Unfortunately, the set $\mathcal{S}$ can be infinite, which implies that the optimization problem above has infinitely many constraints. To convert it into a Binary Integer Programming problem, it is mandatory to convert it into a problem with a finite number of constraints. The next step is thus to shift from a problem with infinitely many constraints to a problem with a finite number of constraints.

1.3.2 Reduction of constraints number

To this aim, it is convenient to use the piece-wise linearity property of the European Calls payoff function. And, as each Call Option has a piece-wise linear payoff function, the portfolio's payoff function ψ is obviously piece-wise linear. It then allows the following proposition [5]:

Proposition 1.3.1. *For a portfolio, The payoff function of n European Call or Put Options with ordered strikes $(K_j)_{1\leq j\leq p}$. Its payoff function $\psi(.)$ is non-negative if and only if:*

$$\begin{cases} \psi(K_j) \geq 0, \quad 1 \leq j \leq p \\ \\ \psi(1 + K_p) \geq \psi(K_p) \end{cases}$$

Where p is the number of strikes. For recall, the corresponding number of options is $n = 2p$ (1 long and 1 short per strike).
This exactly means that, to ensure a non-negative payoff, the payoff diagram must be non-negative at each strike on the x axis and non-decreasing after the last strike on the right edge.

Therefore, the constraints of the optimization problem can be modified in the following way:

$$\Pi^{\mathbf{x}}(S_T) = \mathbf{x}^{\mathsf{T}}\psi(S_T) \geq 0, \quad S_T \in \mathcal{S} \quad \iff \quad \begin{cases} \mathbf{x}^{\mathsf{T}}\psi(K_j) \geq 0, \quad 1 \leq j \leq p \\ \\ \mathbf{x}^{\mathsf{T}}(\psi(1+K_p) - \psi(K_p)) \geq 0 \end{cases}$$

Which leads to the following Binary Integer Programming Problem, that has a finite number of constraints:

$$\begin{aligned} &\text{minimize} \quad \mathbf{x}^{\mathsf{T}}\Pi_0 \\ &\text{subject to} \quad \begin{cases} \mathbf{x}^{\mathsf{T}}\psi(K_j) \geq 0, \quad 1 \leq j \leq p \\ \\ \mathbf{x}^{\mathsf{T}}(\psi(1+K_p) - \psi(K_p)) \geq 0 \end{cases} \end{aligned} \tag{1.3.2}$$

This is a new optimization problem with this time $p+1$ constraints. Now we can finally formulate the Binary Integer Programming Problem in a very general form:

$$\inf_{\mathbf{x}} c^{\mathsf{T}}\mathbf{x}, \quad \text{subject to} \quad A\mathbf{x} \geq b \tag{1.3.3}$$

where b and c can easily be identified as $b = \mathcal{O}_{p+1}$, $c = \Pi_0$ and the matrix $A \in \mathcal{M}_{p+1,n}(\mathbb{R})$ which includes the $p+1$ constraints can be detailed as:

$$A = \begin{pmatrix} \psi_1(K_1) & \psi_2(K_1) & \cdots & \psi_n(K_1) \\ \psi_1(K_2) & \psi_2(K_2) & \cdots & \psi_n(K_2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(K_p) & \psi_2(K_p) & \cdots & \psi_n(K_p) \\ \psi_1(1+K_p) - \psi_1(K_p) & \psi_2(1+K_p) - \psi_2(K_p) & \cdots & \psi_n(1+K_p) - \psi_n(K_p) \end{pmatrix}$$

The expression of matrix A gets simpler using the Call Options payoff functions that read:

$$\psi_i(K_j) = \alpha_i(K_j - K_i)^+ = \begin{cases} \alpha_i(K_j - K_i) \quad \text{if} \quad j > i \\ \\ 0 \quad \text{otherwise} \end{cases}$$

Moreover, the first constraint and therefore the first line of the matrix can be deleted. Indeed, as $\psi_i(K_1) = 0$ for any $1 \leq i \leq n$, it will result in a $0 \geq 0$ condition that is always verified.

This leads to:

$$A = \begin{pmatrix} \alpha_1(K_2 - K_1) & 0 & 0 & \cdots & 0 \\ \alpha_1(K_3 - K_1) & \alpha_2(K_3 - K_2) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_1(K_p - K_1) & \alpha_2(K_p - K_2) & \alpha_3(K_p - K_3) & \cdots & 0 \\ \psi_1(1+K_p) - \psi_1(K_p) & \psi_2(1+K_p) - \psi_2(K_p) & \psi_3(1+K_p) - \psi_3(K_p) & \cdots & \psi_n(1+K_p) - \psi_n(K_p) \end{pmatrix}$$

From there, it is possible to interpret the solution of this Binary Integer Problem as an existence or not of an arbitrage in the portfolio, thanks to the following Theorem:

Theorem 1.3.2. *There is no arbitrage if and only if the binary integer programming problem admits a strictly positive solution.*

Now, let's make a few remarks about the matrix A that for now includes all the problem constraints. The matrix A gathers p constraints: p-1 constraint of the form $\mathbf{x}^\mathsf{T}\Psi(K_i) \geq 0$ translating the positiveness of the payoff function at each strike K_i and a p-th constraint of the form $\mathbf{x}^\mathsf{T}(\psi(1+K_p) - \psi(K_p)) \geq 0$. However, this p-th constraint is different from the n-1 previous, as it characterizes a monotonous direction as from some point K_p and not only a value at one point K_i. For this reason, it would be convenient for the following to encode the last constraint, corresponding to the last row of A, into a separate vector Φ. In this way, the problem would read:

$$\inf_{\mathbf{x}} c^\mathsf{T}\mathbf{x}, \quad \text{subject to} \quad A\mathbf{x} \geq \mathcal{O}_{p-1} \quad \text{and} \quad \mathbf{x}^\mathsf{T}\Phi \geq 0 \tag{1.3.4}$$

With the matrix A and vector Φ defined as:

$$\Phi^\mathsf{T} = (\psi_1(1+K_p) - \psi_1(K_p), \psi_2(1+K_p) - \psi_2(K_p), \cdots, \psi_n(1+K_p) - \psi_n(K_p)) \in \mathbb{R}^n$$

$$A = \begin{pmatrix} (K_2-K_1) & -(K_2-K_1) & 0 & 0 & 0 & \cdots & 0 \\ (K_3-K_1) & -(K_3-K_1) & (K_3-K_2) & -(K_3-K_2) & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \\ (K_p-K_1) & -(K_p-K_1) & (K_p-K_2) & -(K_p-K_2) & (K_p-K_3) & \cdots & 0 \end{pmatrix} \in \mathcal{M}_{p-1,n}(\mathbb{R}).$$

Thus, from the very first definition of an arbitrage in the beginning of this section, it is now possible to state its existence as a solution of a Binary Integer Programming Problem.

1.4 NP-completeness of the problem

Now that the Binary Integer Programming problem is stated, it is convenient to estimate its complexity and some other properties which may guide to adapted ways to solve it. In fact, this problem can be characterized as a special case of the decision version of Mixed-Integer Quadratic Programming (MIPQ), which is defined in [4] as an optimization problem with a quadratic objective function to be minimized over an integer-restricted polyhedron.

Furthermore, [4] categorizes MIPQ's decision versions as NP-complete. Indeed, they are also known as NP-complete combinatorial optimization problems. But what is NP-completeness?

In a simplified way, NP-complete problems are problem which do not have any Polynomial-Time algorithms, oppositely to P problems. It is important to recall that so far there is no proof that $P \neq NP$. One of the most famous NP-Complete problem is the Traveling Salesman Problem with Transportation [6]: If you are given a list of cities along with the distances between each others, what is the shortest possible route which allows you to pass by each city once and finally get back to your starting point?

Therefore, deterministic algorithms are not suitable for this arbitrage detection problem classified as NP-complete as they would not be able to find a solution in a reasonable time. For NP-complete problems, the time required to find a solution closed to the optimal one is likely to be exponential with the size of the problem, which explains why it is not convenient to use classical methods. It is recommended to rather use probabilistic methods, usually search policies named heuristic, to estimate an approximation of the solution. Quantum Annealing is one of these heuristics and will be the privileged solution here. But to understand precisely how does the quantum annealer processes, it is first convenient to recall basics of quantum computing.

Chapter 2

Quantum Annealing Theoretical Overview

This chapter aims to provide theoretical notions related to quantum annealing that are necessary for the understanding of the document. It is recommended to have knowledge of the basics of quantum mechanics that will not be covered.
The first part will focus on general principles of Quantum Computing, articulated around 3 postulates, whereas the second will look closer to the Quantum Annealing process that is used to solve the Arbitrage Detection problem.

2.1 Quantum Computing Basics

Classical computers and Quantum computers have different ways of managing memory. Indeed, classical computers store information coded in binary digits (bits) that can take two distinct and deterministic states: 0 and 1. Quantum computing uses the framework of states superposition, introduced by quantum mechanics, in which a general state is, before measurement, a combination of different 0 and 1 states, weighted by their probability of being measured. They are called Quantum Binary Digits, also known as qbits.

2.1.1 Quantum Binary Digits

As explained above, qbits are used to store and carry information in quantum computers. It must be adapted to the description of a closed system. To describe a quantum state ψ as a superposition of two 0 and 1 states, we use Dirac's notation $|.\rangle$ as below, for example:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1.1}$$

where α and β are complex numbers representing probability amplitudes and therefore verify:

$$|\alpha|^2 + |\beta|^2 = 1$$

When measuring the state $|\psi\rangle$, the state $|0\rangle$ is expected with a probability α^2 and the state $|1\rangle$ is expected with a probability β^2.

This is a direct result from the 1st postulate of quantum mechanics which allows describing the state of a closed system [7]:

Theorem 2.1.1 (1^{st} *Postulate*)**.** *The state space of a physical system is an associated complex inner product space (Hilbert Space). The system is fully described at any time t by its state vector, which is a unit vector belonging to its state space.*

The state 1-qbit $|\psi\rangle$ thus depends only on the two coefficients α and β and it is therefore convenient to represent it as a vector of two complex coefficients:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\begin{bmatrix}1\\0\end{bmatrix} + \beta\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha\\\beta\end{bmatrix}$$

Now that the notations are clear for a single qbit, let's look closer to quantum states with more than one qbit. An n-qbit system requires specifying 2^n probability amplitudes to be described. For example, a two-qbit system will require 4 probability amplitudes to be specified:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle = \alpha\begin{bmatrix}1\\0\\0\\0\end{bmatrix} + \beta\begin{bmatrix}0\\1\\0\\0\end{bmatrix} + \gamma\begin{bmatrix}0\\0\\1\\0\end{bmatrix} + \delta\begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}\alpha\\\beta\\\gamma\\\delta\end{bmatrix}$$

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$$

As information is stored as vectors, quantum computers can manipulate it and transform it easily with matrix operators named "quantum gates" that will not be detailed here.

2.1.2 Schrödinger Equation

Now that the framework to describe a quantum state has been presented, it is important to know about its dynamics and how is it likely to evolve through time in a closed environment [7].

Theorem 2.1.2 (2^{nd} postulate)**.** *A quantum system is composed of particles in a superposition of states, which each have an associated level of energy. It is likely to evolve through time and this time evolution is described by the Schrödinger Equation*

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

where $\hbar$ is Planck's constant and $\mathcal{H}$ is a Hermitian operator known as the Hamiltonian of the system

In the case of a closed system, the Hamiltonian is time independent. The Hamiltonian is a key operator as it allows understanding fully the dynamic of the system. It is an operator which maps the quantum state to its associated energy level. As it is Hermitian, it can be written with a spectral decomposition [8]:

$$\mathcal{H} = \sum_{\lambda\in\mathcal{E}} \lambda|e_\lambda\rangle\langle e_\lambda| \tag{2.1.2}$$

where $\mathcal{E}$ is the set of the eigenvalues λ associated to the normalized eigenvectors $|e_\lambda\rangle$. The Hamiltonian is a mathematical operator with a spectrum that can be interpreted as the collection of all possible energy levels measurable.

Its eigenvectors, called the eigenstates, are associated to their eigen-energy which is a well-defined level of energy. The lowest level of energy, that is the eigenvector associated to the lowest eigenvalue, is called the ground state.

2.1.3 Measurement

The Hermitian property allowing the spectral decomposition seen in equation 2.1.2 leads to the 3^{rd} postulate which translates the interaction of the quantum system with external system. The Hermitian operator $\mathcal{H}$ has a quantum state

Theorem 2.1.3 (3^{rd} postulate)**.** *Before measurement of the state $|\Psi\rangle$ of a Hermitian operator A with eigen functions $(|\psi_i\rangle)$ we have the following decomposition given the spectral theorem describing the uncertainty of the outcome:*

$$|\Psi\rangle = \sum_i \alpha_i |\psi_i\rangle$$

All the possible outcomes are its eigenvalues $(_i)$ associated to the eigenfunctions, with a probability $p_i = |\alpha_i^2|$ to be measured.
After measurement of the outcome λ_i, the state of the system immediately collapses to

$$|\Psi\rangle = |\psi_i\rangle$$

This postulate allows extracting classical information from qbits after measuring their quantum state. However, as the outcome is probabilistic, one needs to perform multiple measures to produce significant statistics.

2.2 Quantum Annealing Principles

Now that the basics of quantum computing have been introduced, it is possible to describe the process of Quantum Annealing used to solve the optimization problem stated.
Indeed, Quantum Annealing can be defined as meta-heuristic which is a search policy to find solutions of a problem. Those solutions are most of the time not global but are good approximations of the best one. There exists many different meta-heuristics to solve optimization problems that have their advantages and inconvenient. For example, gradient descent is a quite efficient search policy but could get easily trapped in local minima. The Quantum Annealing is derived from another famous search policy: the Simulated Annealing.

2.2.1 Simulated Annealing

Simulated Annealing is a meta-heuristic for bounded unconstrained optimization problems. The idea is to use thermal fluctuations modelling to avoid the search to be stuck in a local minimum and slowly cool down to stabilize the system after having explored enough. Let f be the objective function to minimize and define a convergence criterion that allows the search to stop, the algorithm below details the search process:

Algorithm 1 Simulated Annealing

```
Draw an initial point $x = x_0$
Compute $f(x)$
while convergence criterion is not attained do
    Choose a random neighbor $y$ of $x$ and compute $f(y)$
    if $f(y) < f(x)$ then
        Set $x \leftarrow y$
    else
        Set $x \leftarrow y$ with probability $p = \exp\left(-\frac{f(y)-f(x)}{T}\right)$
        Keep $x$ with probability $1 - p$
    end if
end while
```

The crucial step of this algorithm is when $f(y) > f(x)$ and the decision to set $x = y$ can be taken with the probability p defined above. Even if it can seem erratic to choose to switch for a less optimal solution, it actually allows avoiding being trapped into local minima as it favorites the exploration of other solutions. Now, let's look closer to the parameters on which depends the probability p. It is a decreasing function of the distance $|f(y) - f(x)|$. The further is $f(y)$ from the current solution, the less likely it will switch to it. Indeed, the exploration happens more likely with rather close solutions, in order to keep a quick convergence. And finally, p depends on a critical parameter which is T interpreted directly as the time-decaying temperature of the thermal annealing. At the beginning, the temperature is high, which brings p close to 1 and allows more exploration and large moves whereas, at the end, when it is low and when p is close to 0, x stays around its current solution. The high temperature at the beginning, provides energy to the system to make it able to cross the potential energy barriers between two minima.
This idea is at the origin of the quantum annealing meta-heuristic

2.2.2 Adiabatic Quantum Computing

The other result on which Quantum Annealing is founded is Adiabatic Quantum Computing.
A process is said to be adiabatic when is no energy exchanges between the system and the outside. Based on this property, Adiabatic Quantum Computing is a promising method to solve high complexity optimization problems. To use it, the prerequisite is to have encoded the objective function in a Hamiltonian $\mathcal{H}_T$, that for remind, is the mathematical operator describing the energy levels of a quantum closed system, in order to get its solution by measuring its ground state. There is then an equivalence between finding the lowest energy level and the solution of the optimization problem.
However, measuring the ground state of a Hamiltonian encoding an objective function can be tough and that is why one may use the quantum adiabatic process.
To clarify, the idea is to start from an initial Hamiltonian $\mathcal{H}_0$ which has a simpler form and where one can easily determine the ground state. Then the idea is to make the Hamiltonian of the system evolve from $\mathcal{H}_0$ to the wanted Hamiltonian $\mathcal{H}_T$:

$$\mathcal{H}(t) = r(t)\mathcal{H}_0 + (1 - r(t))\mathcal{H}_T \tag{2.2.1}$$

where the function $r(.)$ is the time decay with initial condition $r(0) = 1$ and terminal condition $r(T) = 0$. An example of time decaying path would be:

$$\mathcal{H}(t) = (1 - \frac{t}{T})\mathcal{H}_0 + \frac{t}{T}\mathcal{H}_T$$

From there, the Adiabatic Theorem can be used [9]:

Theorem 2.2.1 (Adiabatic Theorem)**.** *Let a quantum system start in the ground state of a Hamiltonian $\mathcal{H}_0$. If the Hamiltonian of the system evolves sufficiently slowly through time, the system remains in the lowest energy level until time T.*

Therefore, to find the ground state of a complicated Hamiltonian, representing the solution of an optimization problem, it is possible to start from the ground state of a simple Hamiltonian $\mathcal{H}_0$ and make an adiabatic evolution.
The energy levels graph below illustrates the process. As it is shown, there is an energy gap between the lowest energy state and the upper level. Therefore, to move up and leave the ground state, the system needs energy. In adiabatic conditions, when the system evolve slowly, the system doesn't receive any energy from the outside and stays in the ground state.

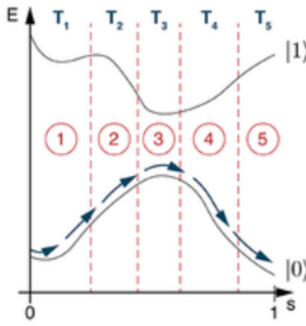


Figure 2.1: Time Evolution of the Quantum State's Energy during an Adiabatic Process

2.2.3 Quantum Annealing

Finally, Quantum Annealing is combining ideas of Simulated Annealing with a quantum adiabatic process. For remind, it is assumed that the optimization problem to solve can be encoded into a special Hamiltonian $\mathcal{H}_T$ in a way that finding the solution of the problem is equivalent to finding the ground state of this Hamiltonian operator.
Thus, if the starting point is a simple initial Hamiltonian $\mathcal{H}$ in its ground state, it can evolve slowly enough towards the final Hamiltonian $\mathcal{H}_T$ which encodes our problem, the adiabatic theorem ensures that one can measure the solution as the quantum system will remain in the ground state. The figure below from [9] well illustrates this idea:

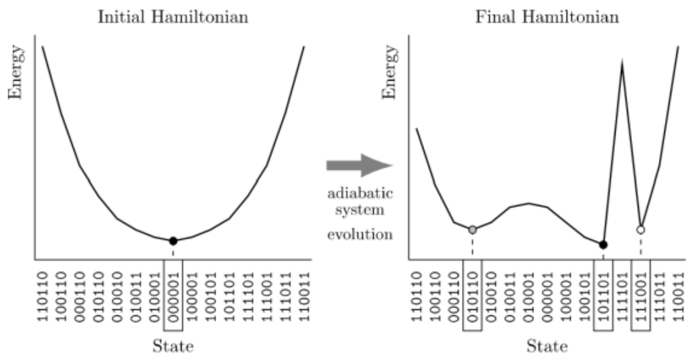


Figure 2.2: Adiabatic Evolution from Initial to Final Hamiltonian during Quantum Annealing

On this figure, the Hamiltonian is clearly expressed as a mapping from a state to an energy level. For some simple Hamiltonian, it is usually easy to know which state corresponds to the lowest energy level (black dot on the figure), but it may not be obvious when the mapping is more complicated and includes local minima. Adiabatic evolution would ensure staying in the global minimum.

Another major point of difference in comparison with the simulated annealing and enabled by the quantum properties is the local hills management. Whereas thermal annealing provided energy to jump other local hills, here the system will use the quantum tunneling phenomenon to get through the local hills as long as they are thin enough. The figure below illustrates well both phenomena [10]:

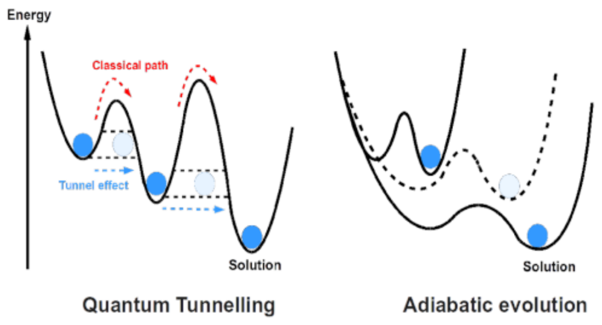


Figure 2.3: Quantum Annealing and Tunnel Effect

Quantum tunneling is a phenomenon proper to quantum particles [11]. It refers to how particles could simply pass through (tunnel), some potential energy hills for which they do not have the energy to climb. It would be impossible for any classical physical entity. But a quantum particle, which can be described as an oscillating wave, would not likely be stopped by an energy barrier. Instead, it will cross it but with some cost on its amplitude, that is directly interpreted as the probability of being found at a certain place. The thinner the barrier, the less the amplitude of the particle's wave will be affected and therefore, the more likely it could be found on the other side of the hill.

2.3 D-Wave Quantum Annealer

Recently, some of the biggest tech firms such as Google, IBM or Intel, have developed quantum chips, in order to be able to build entire quantum computers in the long run. D-wave is one of those firms and has developed components particularly suited for quantum annealing. Even if the size of the newly developed quantum computers is still limited, it has become popular with for example the D-Wave 2000Q quantum annealer, allowing 2048 qbits.

2.3.1 Physical Description

It is possible to find a few details about how D-Wave quantum annealers are built, especially in the D-Wave 2000Q Technology Overview [12]. Quantum annealers are very sophisticated systems, as they must ensure the control of several extreme physical constraints. That's why their size is important and could recall some of the first CPUs built in the XXth century as [12]:

Figure 2.4: D-Wave 2000Q (L:10', W:7', H:10')

The D-Wave 2000Q has to keep its Quantum Processing Unit (QPU) close to absolute zero ($0K$ which is merely equivalent to $-273°$ C). A temperature of $15mK$ is managed to be reached in the shielded enclosure thanks to refrigerator using liquid helium. The reason why it is necessary to keep this extremely low temperature is that the qbits constituting the QPU are made of metal loops which can become superconductors and demonstrate quantum effects only under those conditions. Moreover, the system has to erase any magnetic field noise which can pollute the process and is therefore using high-permeability and superconducting materials to make a magnetic shield. The figure below gives an idea about how are arranged the different components inside the black case [12]:

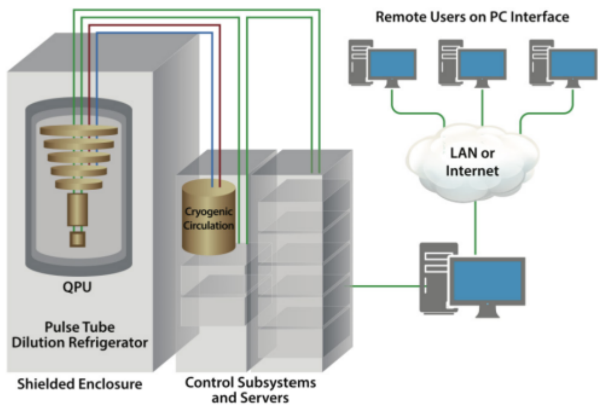

Remote Users on PC Interface
LAN or Internet
Cryogenic Circulation
QPU
Pulse Tube Dilution Refrigerator
Shielded Enclosure
Control Subsystems and Servers

Figure 2.5: D-Wave 2000Q major components

2.3.2 QPU Architecture

In addition to the restriction on the total number of qbits available on the machine, there are restrictions on how the qbits are interconnected with each other. Indeed, it is organized into a same uniform lattice scheme, sometimes called chimera unit cell, as it is represented on the figure below taken from the D-Wave documentation [13]:

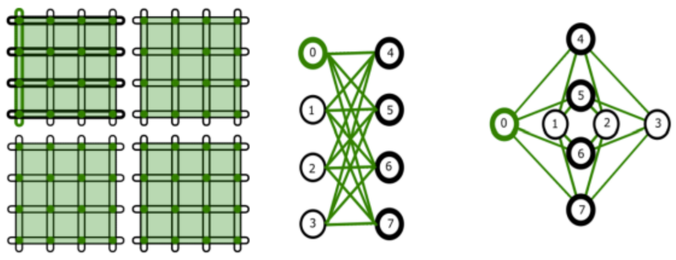


Figure 2.6: On the left, general grid architecture of qbits, on the right, the two possible representations for a chimera unit cell (vertical columns or cross)

The way qbits are interconnected within a same lattice is referred to as internal coupling. As it is shown on the figure, internal coupling is possible between two groups of 4 qbits in this scheme. Therefore, one would think it is impossible to couple a qbit with more than 4 other qbits which would be a huge constraint to solve most of the problems. Nevertheless, it is possible to couple more qbits using external coupling as well, as it is shown on the figure below [13]:

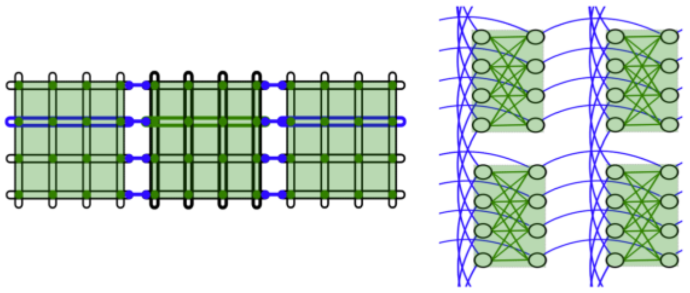


Figure 2.7: On the left, general grid architecture external coupling, on the right, zoom on unit cells external coupling

Thus, the adjacent lattices can have two qbits connected from their extremities. If they are coupled in a way such that they are always equal, it is equivalent with having enabled more possible external connections to one qbit, as it would benefit of the connections from both lattice. This more commonly called a qbit chain and also referred as external coupling.
The QPU architecture of quantum annealers have to respect these internal and external coupling constraints which lead, for the D-Wave to the following scheme [13]:

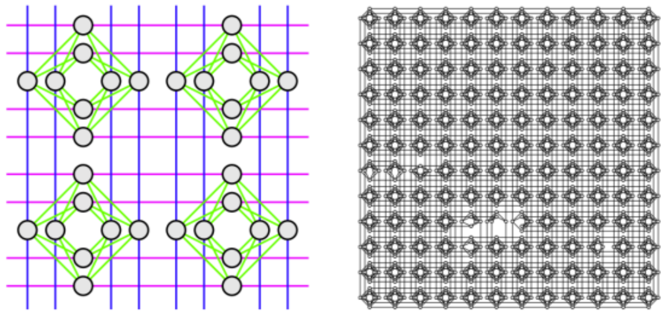


Figure 2.8: On the left, a zoom on a 2×2 unit cells structure, On the right, chimera graph implemented on D-Wave 2000Q systems

For remind, the physical realization of this QPU is possible only by keeping the system into cryogenic temperatures and erasing any magnetic field's interference.

2.3.3 D-Wave Advantage and Pegasus topology

Among the different versions of the D-Wave quantum annealers that have been released since 2011, the one that will be used in this document is the D-Wave Advantage, the latest up to date. It has been released in 2020 and has a capacity of 5640 qbits which is significantly more powerful than the 2048 qbits previously offered by the D-Wave 2000Q. One other major difference compared to the 2000Q is the topology, which refers to the QPU architecture, as the Chimera has been replaced by a Pegasus topology which corresponds to the following layout of the qbits [13]:

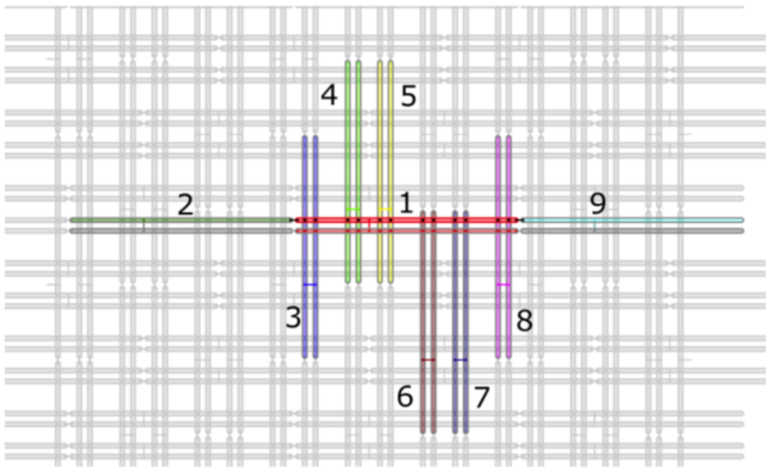


Figure 2.9: Pegasus Topology implemented on D-Wave Advantage system

As for the chimera graph, the qbits are still arranged within a grid, as vertical and horizontal bars but are shifted and no longer perfectly aligned. Here, in addition to the internal and external coupling, each qbit has some odd-coupling, which is why each qbit is represented by two aligned bars. In this way, each qbit is paired to another qbit with this odd coupling, which enables more internal and external coupling. On the figure above, the usual internal coupling of the qbit number 1 is represented by the vertical intersections. It is therefore internally coupled with qbits from 3 to 8. The external coupling is possible with horizontal adjacent qbits, as it is shown with qbit 2 and 9 on the figure above. With this new topology, the number of couplers has been increased from 6 to 15 [14], which enables wider usage of the QPU.

Chapter 3

Problem Translation and Implementation

This section aims to translate the basic mathematical formulation of arbitrage detection that has been stated in 1 to an unconstrained formulation adapted to the quantum annealer used. After having obtained a mathematical formulation adapted, the different tools and resources used for implementation will be presented

3.1 QUBO Representation

3.1.1 QUBO Definition

As explained, the quantum annealing is a search policy which aims to find minima in an objective loss function. This objective function has to represent fully the problem to solve as well as all its constraints. In Chapter 1, the arbitrage detection problem was stated as binary and quadratic but constrained. To convert it, it is convenient to use the Quadratic Unconstrained Binary Optimization (QUBO) formulation which is common for Quantum Annealing. Indeed, it is represented by a loss function $L(x)$ that is quadratic with respect to the binary components x_i. Its general form is:

$$L_{QUBO}(x) = \underbrace{\sum_{i=1}^{n} a_i x_i}_{linear} + \underbrace{\sum_{i=1}^{n} \sum_{j=i+1}^{n} b_{ij} x_i x_j}_{quadratic} \tag{3.1.1}$$

The loss function will include the maximization or minimization criteria of the optimization problem as well as its constraints. Its is built into additive penalties that will increase the overall loss whenever some constraint is not respected. For example, an equality constraint such as $u^{\mathsf{T}}\mathbf{x} = 10$ can be translated into the following penalty function:

$$l(x) = \lambda(\sum_{i=1}^{n} u_i x_i - 10)^2$$

It is here clear that the penalty will grow as soon as the constraint is not respect. Adding this penalty term to the overall loss function of a problem is equivalent to adding a constraint to this problem. Here, the positive coefficient λ is critical because it can be chosen by the user and is directly interpreted as the weight allocated to the constrained.

If it is a critical constraint, an important coefficient λ may be used for this penalty term, comparatively to the other terms weights, whereas if the constraint is more a preference that can be relaxed, a low coefficient may be used.

3.1.2 Introduction of Slack Variables

Thus, the Binary Integer Programming Problem stated in Chapter 1 of this paper and which is constrained, can be translated into a QUBO problem, which is unconstrained. For remind, the problem stated as in Chapter 1 is:

$$\inf_{\mathbf{x}} c^{\mathsf{T}}\mathbf{x}, \quad \text{subject to} \quad A\mathbf{x} \geq \mathcal{O}_{p-1} \quad \text{and} \quad \mathbf{x}^{\mathsf{T}}\Phi \geq 0$$

Where the matrix A and the vector Φ are detailed in section 1.3.2. Therefore, the loss function will be decomposed in three components:

$$L(\mathbf{x}) = L_1(\mathbf{x}) + L_2(\mathbf{x}) + L_3(\mathbf{x})$$

where $L_1(\mathbf{x})$ is the loss function associated to the minimization criteria of $c^{\mathsf{T}}\mathbf{x}$, $L_2(\mathbf{x})$ is the loss function associated to the p-1 constraints contained in the matrix A and $L_3(\mathbf{x})$ is the loss function associated to the p-th constraint contained in the vector Φ.

$L_1(\mathbf{x})$ can be easily set as:

$$L_1(\mathbf{x}) = \lambda_1 c^{\mathsf{T}}\mathbf{x} = \lambda_1 \sum_{i=1}^{n} \Pi_0^j x_j$$

Here, it is clear that the more expensive the initial portfolio is, the higher will be this penalty term. One import question that can arise from here is, what would arrive if this initial price is negative? Indeed, as all penalty terms are additive, it would offset the other penalties and confuse the results. However, here, it is assumed that by setting a high coefficient on the loss functions L_2 and L_3, a negative initial price will occur only in case of an arbitrage and therefore, rarely.

The other penalty terms $L_2(\mathbf{x})$ and $L_3(\mathbf{x})$ require however more attention. A naive formulation for $L_2(\mathbf{x})$ could be to set it as:

$$L_2(\mathbf{x}) = -\lambda_1(\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_j)$$

Indeed, the penalty would be positive if all the components of the product $A\mathbf{x}$ are negative and therefore increase the overall loss function. However, similarly as for the first term, some offsets could occur, for example when some components do respect the constraint and some do not. In some of these cases, the penalty could still be negative even if the constrained is not respected. From this issue, one can recognize that translating inequalities constraints into a QUBO loss function is not as straightforward as other constraints.

A solution to solve this issue is to use what is called "slack variables" that will allow translating an inequality into an equality. The main idea behind is that:

$$\text{If} \quad \sum_{i=1}^{n} x_i \leq h \quad \text{then there exists a positive binary expression } s \text{ such that} \quad \sum_{i=1}^{n} x_i + s = h$$

The binary expression s usually admits the following decomposition:

$$s = \sum_{i=0}^{k} 2^i s_i$$

where the s_i are the slack variables associated to the constraint. Their number k is chosen arbitrarily and is obviously related to the amplitude with which the constraint is likely to be respected. For this arbitrage detection problem, with n different options, there are $\frac{n}{2}$ inequality constraints which require slack variables that would increase the total number of variables.
However, it is preferable to keep this total number of variables as small as possible, as Quantum Annealers have limited capacities. To this aim, one would have to keep the amplitude relatively small and, as pointed out in the remark in section 1.2.2, this amplitude would be directly related to the gap between two consecutive strikes. For this reason, the strikes will be re-scaled to ensure $K_i - K_{i-1} = 1$ for $2 < i \leq p$. Thus, the amplitudes of the payoff function would stay relatively small, and it is possible to keep a small number of slack variables.

The choice is made to keep 2 slack variables for each inequality constraint, allowing therefore an amplitude $0 \leq h \leq 3$ and a total number of slack variables of $2 \times \frac{n}{2} = n$. This leads to the following expressions of $L_2(\mathbf{x})$ and $L_3(\mathbf{x})$:

$$L_2(\mathbf{x}) = \lambda_2 \sum_{i=1}^{p-1} (\sum_{j=1}^{n} A_{ij} x_j - s_i^0 - 2s_i^1)^2$$

where $p = \frac{n}{2}$ is the number of inequality constraints (equal to the number of different strikes).

$$L_3(\mathbf{x}) = \lambda_3 (\sum_{j=1}^{n} \Phi_j x_j - s_p^0 - 2s_p^1)^2$$

Here one can easily assess that as soon as $\sum_{i=1}^{n} A_{ij} x_i$ is negative, then $\sum_{i=1}^{n} A_{ij} x_i - \sum_{p=0}^{k} 2^p s_p^j$ will be negative for any positive binary expansion, and therefore it will generate a positive penalty. However, if $\sum_{i=1}^{n} A_{ij} x_i$ is positive and is within the amplitude given by then binary expansion, then there exists a value of the k slack variables which gives a null penalty. To sum up, the QUBO loss function for the problem is now:

$$L(\mathbf{x}) = \lambda_1 \sum_{i=1}^{n} \Pi_0^j x_j + \lambda_2 \sum_{i=1}^{p-1} (\sum_{j=1}^{n} A_{ij} x_j - s_i^0 - 2s_i^1)^2 + \lambda_3 (\sum_{j=1}^{n} \Phi_j x_j - s_p^0 - 2s_p^1)^2$$

Now, if one wants to use a representation where all the decision and slack variables lie in a same vector $\mathbf{X}$, it is possible to concatenate the initial vector $\mathbf{x}$ of n decision variables with all the slack variables pairs $(s_i^0, s_i^1)_{1 \leq i \leq p}$:

$$\mathbf{X} = [x_1, x_2, ..., x_n, s_1^0, s_1^1, s_2^0, s_2^1, ..., s_p^0, s_p^1] = [x_1, x_2, ..., x_{2n}]$$

$\mathbf{X}$ is a vector of size $N = 2n$. Using this notation, the objective function reads:

$$L(\mathbf{X}) = \lambda_1 \sum_{i=1}^{n} \Pi_0^j x_j + \lambda_2 \sum_{i=1}^{p-1} (\sum_{j=1}^{n} A_{ij} x_j - x_{n+2i} - 2x_{n+2i+1})^2 + \lambda_3 (\sum_{j=1}^{n} \Phi_j x_j - x_{2n-1} - 2x_{2n})^2$$

3.1.3 Canonical form of QUBO

Now that the constraints for the initial problem have been translated into a single objective function, it is now convenient to express it in the canonical form of QUBOs stated in 3.1.1, with a vector containing the linear part of the objective function and a triangular coupling matrix containing the quadratic part. This canonical form of the QUBO will then be straightforward to implement on a quantum annealer software interface which can perform the embedding from it. The best way is to proceed term by term, and the result is easily obtained from basic sum manipulations.
The first term gives:

$$L_1(\mathbf{x}) = \lambda_1 \sum_{i=1}^{n} \Pi_0^j x_j = \sum_{j=1}^{n} \lambda_1 \Pi_0^j x_j = \sum_{j=1}^{N} \lambda_1 \Pi_0^j \mathbb{1}_{j \leq n} x_j$$

Where $\mathbb{1}_{j\leq n}$ is an indicator function that is worth 1 when $j \leq n$ and 0 elsewhere.

The second term reads:

$$\begin{aligned}
L_2(\mathbf{X}) &= \lambda_2 \sum_{i=1}^{p-1} (\sum_{j=1}^{n} A_{ij} x_j - x_{n+2i} - 2x_{n+2i+1})^2 \\
&= \lambda_2 \sum_{i=1}^{p-1} \Big((\sum_{j=1}^{n} A_{ij} x_j)(\sum_{k=1}^{n} A_{ik} x_k) - 2(\sum_{j=1}^{n} A_{ij} x_j)(x_{n+2i} + 2x_{n+2i+1}) + (x_{n+2i} + 2x_{n+2i+1})^2 \Big) \\
&= \lambda_2 \sum_{i=1}^{p-1} \Big((\sum_{j=1}^{n} \sum_{k=1}^{n} A_{ij} x_j A_{ik} x_k - 2(\sum_{j=1}^{n} A_{ij} x_j)(x_{n+2i} + 2x_{n+2i+1}) + (x_{n+2i}^2 + 4x_{n+2i} x_{n+2i+1} + 4x_{n+2i+1}^2) \Big) \\
&= \lambda_2 \Big(\sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{i=1}^{p-1} A_{ij} A_{ik} x_j x_k - 2 \sum_{j=1}^{n} \sum_{i=1}^{p-1} A_{ij} x_j (x_{n+2i} + 2x_{n+2i+1}) + \sum_{i=1}^{p-1} (x_{n+2i}^2 + 4x_{n+2i} x_{n+2i+1} + 4x_{n+2i+1}^2) \Big) \\
&= \lambda_2 \Big(\sum_{j=1}^{N} \sum_{k=1}^{N} \Big(\mathbb{1}_{j \leq n, k \leq n} \sum_{i=1}^{p-1} A_{ij} A_{ik} \Big) x_j x_k + \sum_{j=1}^{N} \sum_{k=1}^{N} \Big(\mathbb{1}_{n < j \leq N-2} \mathbb{1}_{k \leq n} A_{j-n-int[\frac{j-n}{2}],k} (-2 \times \mathbb{1}_{j_{odd}} + -4 \times \mathbf{1}_{j_{even}}) x_j x \\
&+ \sum_{j=1}^{N} \sum_{k=1}^{N} \mathbb{1}_{n < j \leq N-2} \mathbb{1}_{n < k \leq N-2} (\mathbb{1}_{j=k, j_{odd}} + 4 \times \mathbb{1}_{j=k, j_{even}} + 4 \times \mathbb{1}_{j=k+1, j<N-3}) x_j x_k \Big) \\
&= \sum_{j=1}^{N} \sum_{k=1}^{N} \lambda_2 \Bigg[\underbrace{\Big(\mathbf{1}_{j \leq n, k \leq n} \sum_{i=1}^{p-1} A_{ij} A_{ik} \Big)}_{C_1} + \underbrace{\Big(\mathbb{1}_{n < j \leq N-2} \mathbb{1}_{k \leq n} A_{j-n-int[\frac{j-n}{2}],k} (-2 \times \mathbf{1}_{j_{odd}} - 4 \times \mathbb{1}_{j_{even}} \Big)}_{C_2} + \\
&+ \underbrace{\Big(\mathbb{1}_{n < j \leq N-2} \mathbb{1}_{n < k \leq N-2} (\mathbb{1}_{j=k, j_{odd}} + 4 \times \mathbb{1}_{j=k, j_{even}} + 4 \times \mathbb{1}_{j=k+1, j<N-3}) \Big)}_{C_3} \Bigg] x_j x_k \\
&= \sum_{j=1}^{N} \sum_{k=1}^{N} \lambda_2 (C_1 + C_2 + C_3) x_j x_k
\end{aligned}$$

With similar manipulations, it is possible explicit $L_3(\mathbf{X})$ into a canonical form:

$$\begin{aligned}
L_3(\mathbf{X}) &= \lambda_3\Big(\sum_{j=1}^{n}\Phi_j x_j - x_{2n-1} - 2x_{2n})^2\Big)\\
&= \lambda_3\Big(\sum_{j=1}^{n}\sum_{k=1}^{n}\Phi_j\Phi_k x_j x_k - 2(x_{2n-1}+2x_{2n})\sum_{j=1}^{n}\Phi_j x_j + (x_{2n-1}^2 + 4x_{2n-1}x_{2n} + 4x_{2n}^2)\Big)\\
&= \lambda_3\Big(\sum_{j=1}^{N}\sum_{k=1}^{N}\underbrace{(\mathbb{1}_{j\le n,k\le n}\Phi_j\Phi_k)}_{D_1}x_j x_k + \sum_{j=1}^{N}\sum_{k=1}^{N}\underbrace{(-2\mathbb{1}_{j=N-1,k\le n}\Phi_k - 4\mathbb{1}_{j=N,k\le n}\Phi_k)}_{D_2}x_j x_k +\\
&+ \sum_{j=1}^{N}\sum_{k=1}^{N}\underbrace{(4\mathbb{1}_{j=N,k\ge N-1} + \mathbb{1}_{j=N,k=N-1})}_{D_3}x_j x_k\Big)\\
&= \sum_{j=1}^{N}\sum_{k=1}^{N}\lambda_3(D_1+D_2+D_3)x_j x_k
\end{aligned}$$

Which leads to the following canonical QUBO representation of the entire objective function:

$$L(\mathbf{X}) = \underbrace{\sum_{j=1}^{N}\Big(\lambda_1\Pi_0^j\mathbf{1}_{j\le n}\Big)x_j}_{linear} + \underbrace{\sum_{j=1}^{N}\sum_{k=1}^{N}\Big(\lambda_2(C_1+C_2+C_3) + \lambda_3(D_1+D_2+D_3)\Big)x_j x_k}_{quadratic}$$

From this expression it is possible to explicit directly the triangular coupling matrix of the problem variables given by the quadratic part of the objective function:

$$M_{\text{coupling}} = \left(\begin{array}{c|c} M & \mathcal{O}_{n,n} \\ \hline N & P \end{array}\right)$$

where M, N and P are:

$$M_{(i,j)} = \lambda_2\sum_{k=1}^{p-1}A_{k,i}A_{k,j} + \lambda_3\Phi_i\Phi_j \quad \text{for} \quad 1\le i\le n \quad \text{and} \quad 1\le j\le n$$

$$N = \begin{pmatrix}
-2\lambda_2 A_{1,1} & -2\lambda_2 A_{1,2} & \cdots & -2\lambda_2 A_{1,n-1} & -2\lambda_2 A_{1,n}\\
-4\lambda_2 A_{1,1} & -4\lambda_2 A_{1,2} & \cdots & -4\lambda_2 A_{1,n-1} & -4\lambda_2 A_{1,n}\\
-2\lambda_2 A_{2,1} & -2\lambda_2 A_{2,2} & \cdots & -2\lambda_2 A_{2,n-1} & -2\lambda_2 A_{2,n}\\
-4\lambda_2 A_{2,1} & -4\lambda_2 A_{2,2} & \cdots & -4\lambda_2 A_{2,n-1} & -4\lambda_2 A_{2,n}\\
\vdots & \vdots & \vdots & \vdots & \vdots\\
-2\lambda_2 A_{p-1,1} & -2\lambda_2 A_{p-1,2} & \cdots & -2\lambda_2 A_{p-1,n-1} & -2\lambda_2 A_{p-1,n}\\
-4\lambda_2 A_{p-1,1} & -4\lambda_2 A_{p-1,2} & \cdots & -4\lambda_2 A_{p-1,n-1} & -p\lambda_2 A_{p-1,n}\\
-2\lambda_3\Phi_1 & -2\lambda_3\Phi_2 & \cdots & -2\lambda_3\Phi_{n-1} & -2\lambda_3\Phi_n\\
-4\lambda_3\Phi_1 & -4\lambda_3\Phi_2 & \cdots & -4\lambda_3\Phi_{n-1} & -4\lambda_3\Phi_n
\end{pmatrix}$$

$$P = \begin{pmatrix} \lambda_2 & 0 & \cdots & \cdots & \cdots & 0 \\ 4\lambda_2 & 4\lambda_2 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & 4\lambda_2 & 4\lambda_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 4\lambda_2 & 0 & 0 \\ 0 & \cdots & 0 & 0 & \lambda_3 & 0 \\ 0 & \cdots & 0 & 0 & 4\lambda_3 & 4\lambda_3 \end{pmatrix}$$

3.2 Implementation

This simulation will use the D-wave systems and its presented architecture to attempt to solve our QUBO problem stated in the previous section. To this end, it will be convenient to use the Ocean Python package, which is particularly suited for this.

3.2.1 Data Used

The data used for this arbitrage detection problem are prices of European SPY Call Options taken from the CBOE website [15]. The entire data set includes Options with strikes from 180 to 620, but not all the options will be considered as it is preferable that the size of the problem to remain within the limits of Quantum Processing Units.
Thus, the simulation will be run for the following number of options: $n = 8$, $n = 10$, $n = 16$, $n = 26$ and $n = 46$. As explained before, for one strike, two options are used: one corresponding to a long position, with an $\alpha_i = 1$ and relying on the ask price, and one corresponding to a short position, with an $\alpha_i = -1$ and relying on the bid price. For example, the head of the data set is shown below to illustrate the data used for n options (and therefore $\frac{n}{2}$ strikes):

Index	Price	Strike	Direction
0	218.32	180	-1
1	221.54	180	1
2	213.6	185	-1
3	216.81	185	1
...	...	...	...
n-1	0.25	600	-1
n	0.6	600	1

Table 3.1: Data Used on European SPY Call Options from CBOE website

3.2.2 Ocean package

D-Wave Ocean is a python library allowing to use D-Wave samplers for various type of problems, including Binary Quadratic Models, with the Dimod library. It is an open-source software that allows a connection to a solver which then will use a CPU, GPU or QPU depending on the sampling method specified.

The Problem Inspector is one of the popular and enjoyable feature of the Ocean Software as it directs the user to a special interface after having run the solver, to observe the results and analyze it.
In this implementation, two samplers will be used to solve the problem and will be compared:

- SimulatedAnnealingSampler : a sampler using the classical Simulated Annealing algorithm
- DWaveSampler: a sampler using D-Wave Advantage quantum annealer

3.2.3 Setting QUBO Hyperparameters

A final task before implementation is to set the hyperparameters λ_1, λ_2 and λ_3. These parameters have to be precisely calibrated in order for the samplers to be able to find arbitrage whenever it occurs. If ever it was not well calibrated, for example, if one of the three encoded constraints was over-weighted, then the two others will be under-weighted and therefore the solution found might not match the constraints defining arbitrage.
But, to assess whether some model is good or not at detecting arbitrage, it is necessary to know if there is indeed an arbitrage in the data used. As searching for a data set containing an arbitrage can be long and tedious, a good shortcut is to create an artificial arbitrage in the data used by manipulating some prices. Then the idea is to modify the parameters until the artificial arbitrage is found by a classical algorithm. The hyperparameters obtained by this method are:

λ_1	1.6
λ_2	30
λ_3	125

Table 3.2: QUBO Hyperparameters

Chapter 4

Results and Interpretation

This chapter aims to expose the results obtained from the simulations realized with the set-up described previously, using the D-Wave sampler provided by Ocean. Firstly, the different metrics used to assess the performance of Quantum Annealing for this arbitrage detection problem will be presented. Then comparisons will be made between simulations with different numbers of problem variables and each of this simulation will be compared to a benchmark using a classic meta-heuristic.

4.1 Sampling Solutions

The two samplers used provide as output all the samples found as solution for each read grouped by energy levels and ranked in an increasing order for energy level. Thus, at the top of the list it is possible to read the best solution found among all the trials as well as the number of times this solution was found. In the following, illustrative examples will be presented for the simulation with a number $n = 10$ of options, corresponding to a total number of variables of $N = 20$ and therefore offering 2^{20} possible states. Given that the number of possible states is very large relatively to the number of reads (which is limited to a maximum of 10 000), it is normal to expect that the number of occurrence for most of the energy level found is 1. But by gathering energy levels by buckets, it might be possible to observe a distribution of the solutions' energy levels.
For example, find below the first rows of the D-Wave sampler's output for the simulation with 10 options:

x_1	x_2	x_3	x_4	x_5	x_6	...	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	Energy
0	1	1	0	0	1	...	1	0	0	0	0	0	-21.6
0	1	1	1	1	1	...	1	0	0	0	0	0	-21.36
...	...	...	...	...	...	...	...	...	...	...	...	...	...
0	0	0	1	1	0	...	0	0	0	0	0	0	1937.2
0	1	0	1	0	1	...	0	1	1	0	0	1	2204.384

To access the best solution, one can extract the state in the first row corresponding to the best energy level. Then, it is necessary to remove the slack variables and keep only the decision variables, here from x_1 to x_{10}, which will indicate on which options the investor should take position to take an arbitrage, in the case an arbitrage is found. However, the Ocean package can provide various and precious post-processing information.

4.2 Metrics for Quantum Annealing performance

To assess the performance of the D-Wave Quantum Annealer for this problem, the first thing to assess is whether the solution provided is an arbitrage or not. Here, the arbitrage character is assessed with two elements: the payoff diagram of the portfolio corresponding to the solution that must be non-negative everywhere and the price of the replicating portfolio that must be negative. Nevertheless, other criteria can be used to evaluate the evolution of performance with respect to the number of the problem variables. Ocean comes with a Problem Inspector interface which provide several useful information and metrics about the Quantum Annealing performed. It is then possible to compare these data for different number of problem variables. Among the data that can be useful, the comparison done further will be focused on:

- Samples Distribution
- QPU Architecture
- QPU sampling time
- Number of Qbits used on the QPU comparatively to the initial number of problem variables
- Execution time

4.3 Results

The results provided below aim to compare the D-wave sampler's performance for the arbitrage detection problem with 8, 10, 16, 26 and 46 options, respectively corresponding to 16, 20, 32, 52 and 92 problem variables.

4.3.1 Accuracy of Solutions

The first critical assessment to do is, as explained previously, to check if the ground solutions found by the samplers are indeed arbitrages. Simulations were realized with 10000 reads per annealing for the D-Wave Sampler and 250 reads per annealing for the classic algorithm. The higher is the number of read, the higher is the probability to find a global solution. The reason why the number of reads is much lower for the Simulated Annealing is because it is way more time-consuming.
Below are the results of the simulations gathered in a table:

N	Sampler	Lower Energy Level	Non-negative payoff	Price	Arbitrage Found
16	D-Wave	-1.28	**Yes**	**-0.8**	**Yes**
	Simulated Annealing	-1.28	**Yes**	**-0.8**	**Yes**
20	D-Wave	-0.38	**Yes**	**-0.23**	**Yes**
	Simulated Annealing	-21.6	**Yes**	**-13.5**	**Yes**
32	D-Wave	53.94	No	**-3.79**	No
	Simulated Annealing	5.94	**Yes**	5.5	No
52	D-Wave	527.1	No	15.24	No
	Simulated Annealing	43.2	**Yes**	15.5	No
92	D-Wave	1140	**Yes**	52.94	No
	Simulated Annealing	45.4	**Yes**	28.38	No

Table 4.1: Global Results

Many comments can be made from this first table of results. Some solutions will be examined through their corresponding payoff diagram to illustrate the table's figure.

Payoff Diagrams of the Solutions

Firstly, for $N = 16$, the D-Wave sampler and Simulated Annealing sampler both converges to the same energy level. The quantum state measured associated to this energy level is a portfolio which has a non-negative payoff everywhere and a negative price of -0.8. It is therefore an arbitrage and both method did work quite well to detect it among the 2^{16} possible states. Below is the payoff diagram of this solution:

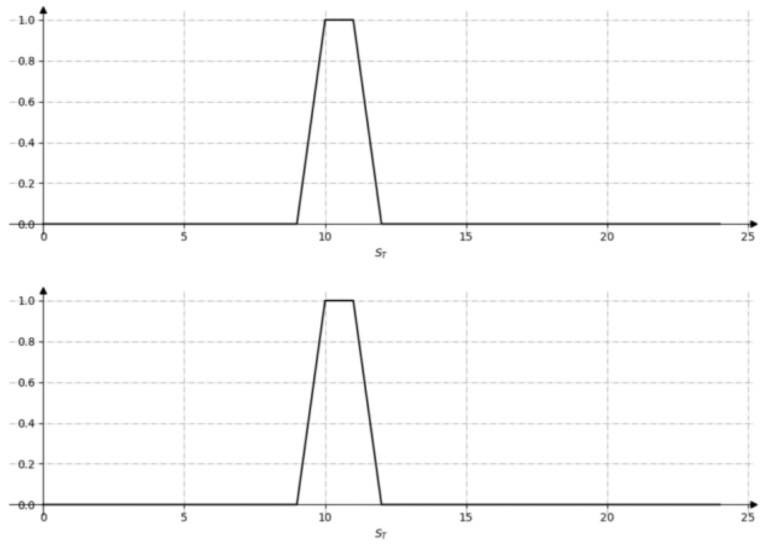


Figure 4.1: $N = 16$: Payoffs of the ground solutions find by Simulated Annealing (top) and Quantum Annealing (bottom). Respective prices of the corresponding portfolios are -0.8 and -0.8

Then, for $N = 20$, it is possible to observe some similar payoff diagrams as, again, both samplers manage to converge to some arbitrage portfolios with negative prices and non-negative payoff. However, there is a difference now between the two solutions found by the samplers, as it is possible to view it directly on the payoff diagrams which are slightly different:

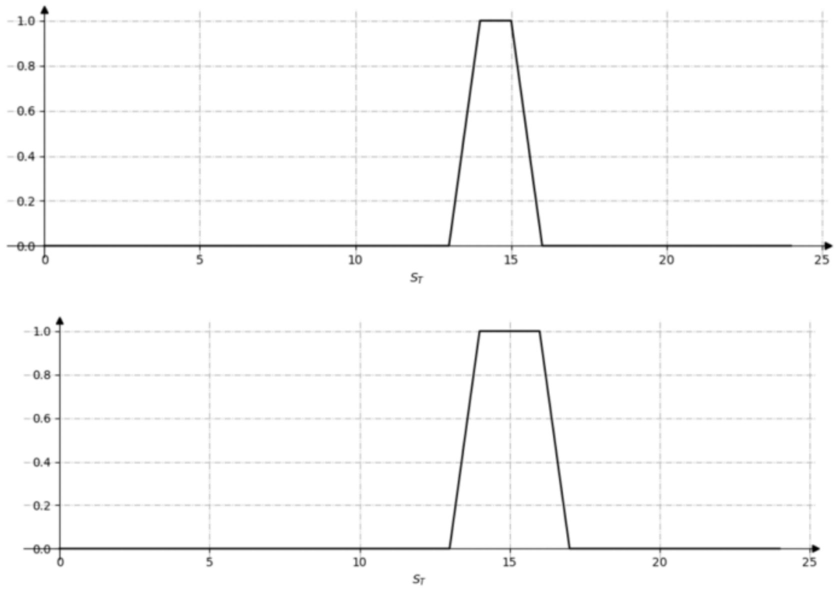


Figure 4.2: $N = 20$: Payoffs of the ground solutions find by Simulated Annealing (top) and Quantum Annealing (bottom). Respective prices of the corresponding portfolios are -13.5 and -0.23.

Indeed, the Simulated Annealing algorithm converges to a lower energy level than the Quantum annealing. Overall, both algorithms have well performed as they both found some arbitrage. The classic algorithm found a portfolio with a slightly narrower window of positive payoff but with a significantly cheaper price, which can explain why it is a "better" arbitrage. However, some investors could prefer portfolios of cost close to 0 but offering the largest window of positive payoff and would buy the portfolio found by the quantum algorithm rather than the other. Using the definitions given in Chapter 1, the solution provided by Simulated Annealing is an arbitrage of type B whereas the one provided by Quantum Annealing is an arbitrage of type B as well but closer to an arbitrage of type A.

Now, looking at results when the problem size increases to $N = 32$, $N = 52$ and $N = 92$, it seems that the main trend is that both algorithms do not converge to an arbitrage solution but that the Simulated Annealing seems more performing than the Quantum Annealing as each of its solution has a significantly smaller energy level. This gap of performance seems to increase with the size of the problem. Let's take a closer look at the cases $N = 32$ and N=52:

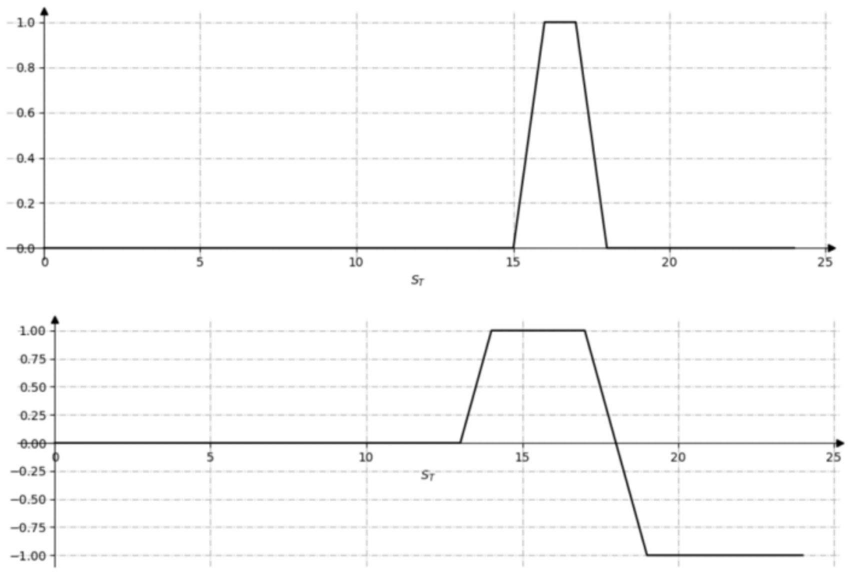


Figure 4.3: $N = 32$: Payoffs of the ground solutions find by Simulated Annealing (top) and Quantum Annealing (bottom). Respective prices of the corresponding portfolios are 5.5 and -3.79

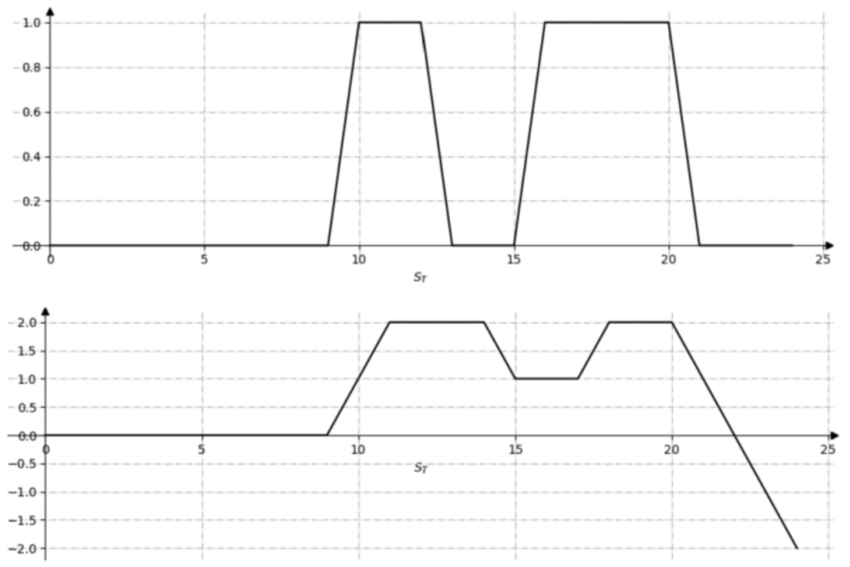


Figure 4.4: $N = 52$: Payoffs of the ground solutions find by Simulated Annealing (top) and Quantum Annealing (bottom). Respective prices of the corresponding portfolios are 15.5 and 15.24

On the top figure, the portfolio found by Simulated Annealing seems to be still a better solution as it respect all constraints on the payoff, whereas the one provided by Quantum Annealing shows some negative payoff as from $S_T = 17$. But one could argue that it is more complicated to tell which of the portfolio is the most erratic, as the first one has a positive cost and the second one a negative cost. The reason why the top solution has a lower energy level and is judged as "better" is because the setting of the QUBO hyperparameters done in subsection 3.2.3 has imposed a higher penalty to non-negative payoff comparatively to positive price.

For the case $N = 52$, it is more obvious that the quantum algorithm is less performing than the classical one. Indeed, now, both portfolios have a similar positive cost and respect the conditions to have a non-negative payoff at each strike K_i, (here the last and largest re-scaled strike is 21). Nevertheless, it is clear that the solution provided by Quantum Annealing does not respect the condition of a non-decreasing payoff on the right edge, imposed by the constraint related to vector Φ, that leads to negative payoff as from $S_T = 22$. Thus, one can conclude that the Simulated Annealing provides a less erratic solution even if it is not an arbitrage, which is consistent with the energy levels comparison.
The payoff diagrams of the solutions found for $N = 92$ are provided in the appendix A.2. They will not be detailed as the analysis could be similar as for $N = 52$.

Samples Distribution

Looking at the samples distribution is another illustration of how the Quantum Annealing struggles to converge toward a solution when the size increases. It maps each energy level to the number of times it is sampled by the algorithm during the annealing process:

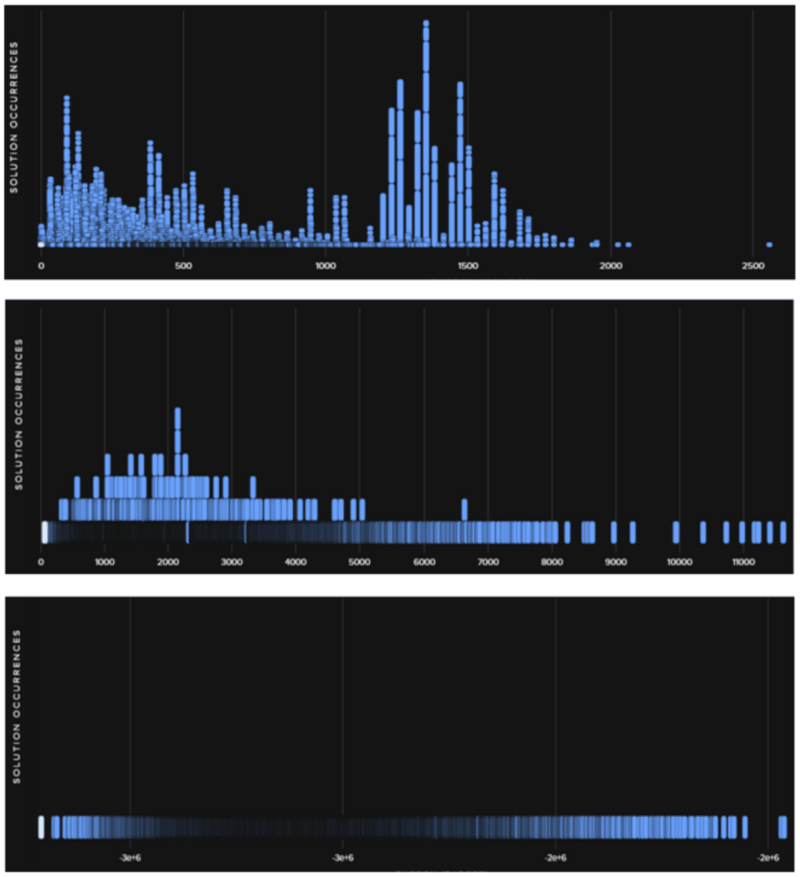


Figure 4.5: Samples Energy Distribution for $N = 16$ (upper), $N = 32$ (middle) and $N = 92$ (bottom). X-axis: energy level, Y-axis: sample occurrences

For simulations with the smaller number of problem variables, it is possible to observe a quite clear distribution of the samples with clear hills. Even if the hills do not correspond to a final solution but rather to local minima in which the search policy enters recurrently, it proves there is a decent trade-off between exploration and exploitation. This distribution is flattening when the number of problem variables increases, which shows that there are too many possible states to consider and the algorithm do not manage to converge to a global minimum. It would be still quite a good solver if the problem treated could accept some solutions that are not optimal but relatively close to the best one. But the arbitrage detection has a rather binary output, in which any solution that is not an arbitrage is considered as not profitable. Thus, without any time considerations, the Simulated Annealing seems more accurate than the Quantum Annealing for this problem.

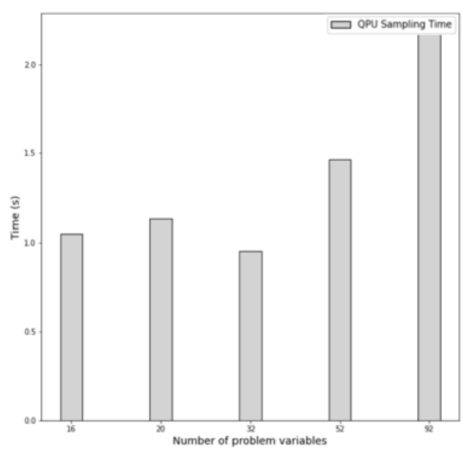
4.3.2 Time Performance

Now that the accuracy of both methods has been evaluated, let's focus on the QPU time performance and characteristics.

For this metric, two kind of comparisons can be made:

- The first one is how the Quantum Annealing time provided by the Problem Inspector evolves when the number of problem variables is increasing.
- The second one is a simple measure of the execution line time in the python environment, to compare how this time evolve for both samplers when the number of problem variables.

The results have been gathered on the two following bar plots:



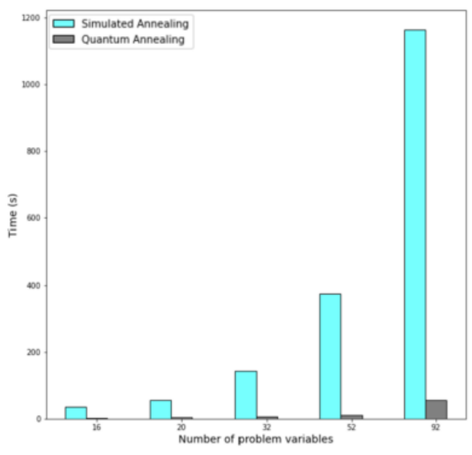


Figure 4.6: Left plot: QPU Sampling times provided by the D-Wave Problem Inspector

Figure 4.7: Right plot: Time Comparison between Quantum Annealing and Simulated Annealing

On the left-hand plot, one can assess the that the QPU sampling time for a simulation with 10000 reads is quick and rather stable when the number of variables increases, which is the most important.

On the right-hand plot, the time for the sampling execution line to be run in the python environment used. The values in absolute hardly have a meaning because the environment did not aim to give competitive times, but the critical information is in how these times evolves for the two sampler when the number of problem variables increases. It is clear that the execution time for the classic Simulated Annealing skyrockets and is likely to get too slow to be used in some competitive production environment such as algorithmic trading. On the opposite, the Quantum Annealing running time is always much faster and the performance remain much more stable with a light rise for $N = 92$.

As one could have expected, there seems to be a trade-off between the outstanding speed of Quantum Annealing and its lack of accuracy compared to classical methods.

4.3.3 QPU layout and limits

Finally, some last commentary can be made about some capacity limitations of the D-Wave Advantage quantum annealer.
As explained in Chapter 2, the QPU is subject to several constraints for the coupling of the qbits. That's why, the number of qbits used on the QPU, referred also as the number of target variables, is usually greater than the number of problem variables, also referred to as the number of source variables. Therefore, even if some QPUs now allow more than 5000 qbits, it may be impossible to use it with problems of 5000 source variables. Here below is the following number of qbits used on the QPU for each number of source variables used:

Number of source variables	**20**	**32**	**52**	**92**
Number of Qbits used	53	137	383	1128

Table 4.2: Number of Qbits used vs Number of source/problem variables

The pictures below represent the qbits used on the QPU and their connections to each other:

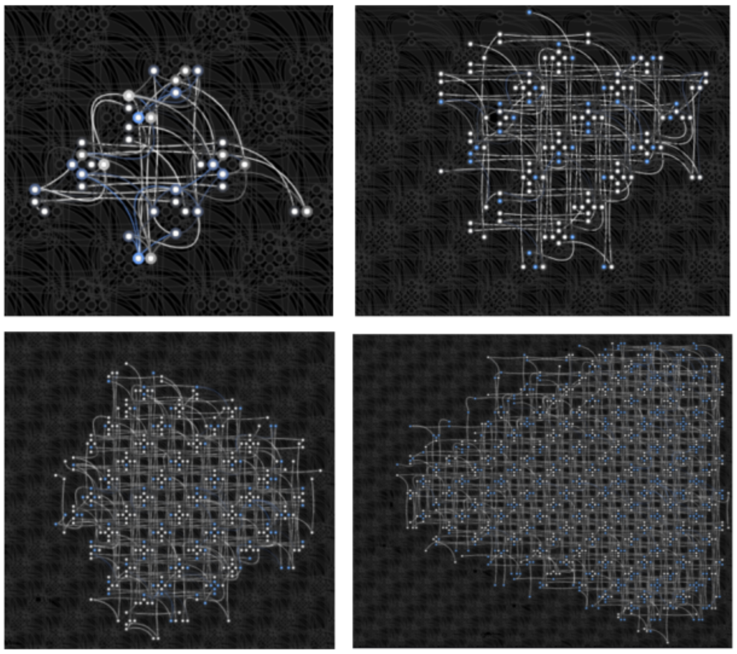
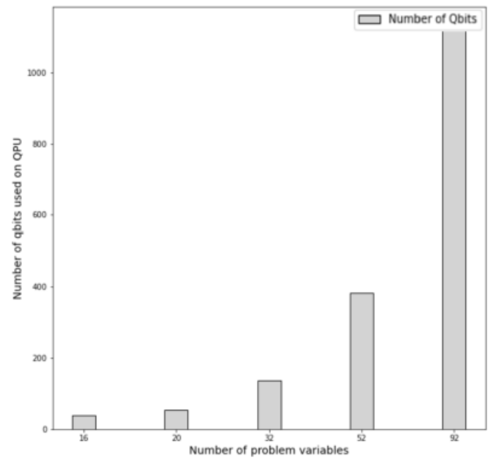


Figure 4.8: QPU layout with Pegasus topology for: $N = 16$ (top left), $N = 32$ (top right), $N = 52$ (bottom left), $N = 92$ (bottom right)

It confirms the figures of the previous table that were indicating that, due to the internal and external coupling constraints, the number of qbits used is significantly higher than the number of initial problem variables. It is easy to see that the evolution of the number of qbits is quadratic with the bar plots below:



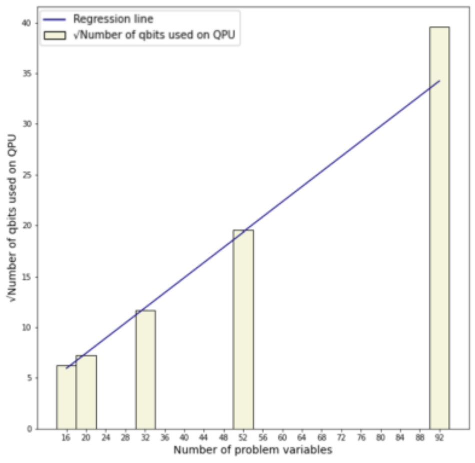


Figure 4.9: Left plot: Number of Qbits used vs Number of problem variables

Figure 4.10: Right plot: Square-root of Number of Qbits used vs Number of problem variables (with regression line)

Even if it is well known that QPU have limited capacity, one should not underestimate that the total number of qbits available, which can be above 5000, is far from indicating a limit for the number of problem variables that can be used. This limitation should be taken to account for any commercial or industrial application which would likely have large size problems to solve.

Conclusion

To conclude this thesis, let's summarize the main points that have been explored. For remind, the main goal of the paper was to propose an innovative solution to an arbitrage detection problem on a European Call Options portfolio. The secondary aim was to give an example of how an optimization problem can be translated and transformed into a problem that is suited for QPU use.

Firstly, it was convenient to use from the very beginning a description for the portfolio that would allow binary decision variables. The first arbitrage detection problem was stated as an optimization problem with an infinite number of constraints that had to be reduced using the piece-wise linearity of European Options payoff. The resulting Binary Integer Programming problem was qualified as NP-complete which justifies the use of some stochastic meta-heuristics and, in this case, of Quantum Annealing.

Quantum computing is a relatively recent topic which uses the framework of quantum mechanics to propose a new way of encoding information and perform computations. Quantum optimization has become popular thanks to companies such as D-Wave who have been successful in building reliable quantum annealers. Their latest D-Wave Advantage System, that has a capacity of over 5000 qbits, was used to solve the problem stated in Chapter 1.

To be able to use the QPU accessible through the Ocean Package which allows a remote connection to D-Wave systems, it was mandatory to translate the problem into a QUBO, which is a quadratic unconstrained binary objective function. It was necessary to introduce slack variables in order to handle inequality constraints that are not straightforward to translate into a QUBO. The QUBO was then translated into a Hamiltonian by the D-Wave system, and finding an optimal solution was equivalent to finding the ground state of this Hamiltonian. After having stated the problem into a canonical form, simulations have been run for a total number of decision variables of $N = 16$, $N = 20$, $N = 32$, $N = 52$ and $N = 92$. Data from the CBOE were used and modified in order to introduce some artificial arbitrage in it, to be able to assess how performing are the algorithms used at detecting it. The results were compared to the classical Simulated Annealing sampler used here as a benchmark.

The first result assessed was the arbitrage nature of the solutions found by the D-Wave Sampler and the Simulated Annealing Sampler. For small numbers of problem variables, both methods converge to the same energy level and therefore to the same solution, which is indeed an arbitrage. As the number of problem variables increases, both samplers' performance is degrading, but the classical Simulated Annealing seems to be more accurate as it always finds solution with lower energy levels, which are closer to the optimal one.

For $N = 20$, the two samplers both converge to arbitrage solutions but with different energy levels associated, the more optimal being the one provided by Simulated Annealing. For larger problem sizes, both algorithms do not converge to arbitrage, but again the Quantum Annealing seems to provide less optimal solutions according to the energy levels and payoff diagrams.
However, regarding time performance, the Quantum Annealing shows significant speed comparatively to its classical equivalent. The QPU sampling time and time to run the execution line in the Python environment remain low and quite stable when the size of the problem increases. Conversely, the execution time for Simulated Annealing is higher and grows exponentially with the problem size. There is therefore a trade-off between the accuracy and the speed for the two methods, which may be adapted to different kind of problem.

The arbitrage detection problem is a problem which requires strong accuracy, as the arbitrage nature of a solution is a binary output that does not allow approximated solutions. But it requires great speed as well as arbitrage comes from lacks of efficiency on the market which may not last more than several seconds. A good solution if one wants to keep quantum annealing as an arbitrage detector would be to keep the problem size reasonable, below $N = 20$. Thus, one could focus on Options with strikes strongly in the money and strikes strongly out of the money which are likely to be less traded, less liquid and which therefore would allow more easily arbitrage. In this case, it is hard to conceive that Quantum Annealing could be used for commercial or industrial applications, unless if it is for problems which allow approximated solutions. Quantum computing is evolving at a fast pace but still showing strong limitations such as qbits capacity, especially when considering the coupling constraints, and its restriction to specific kinds of mathematical problems only. For the first one, research on new QPU architectures is active, as for example the Zephyrus topology which aims to replace the Pegasus one for the next generation of D-Wave systems and will likely relax the coupling constraints allowing more connections between the qbits.

Appendix A

Results

A.1 Samples Distributions

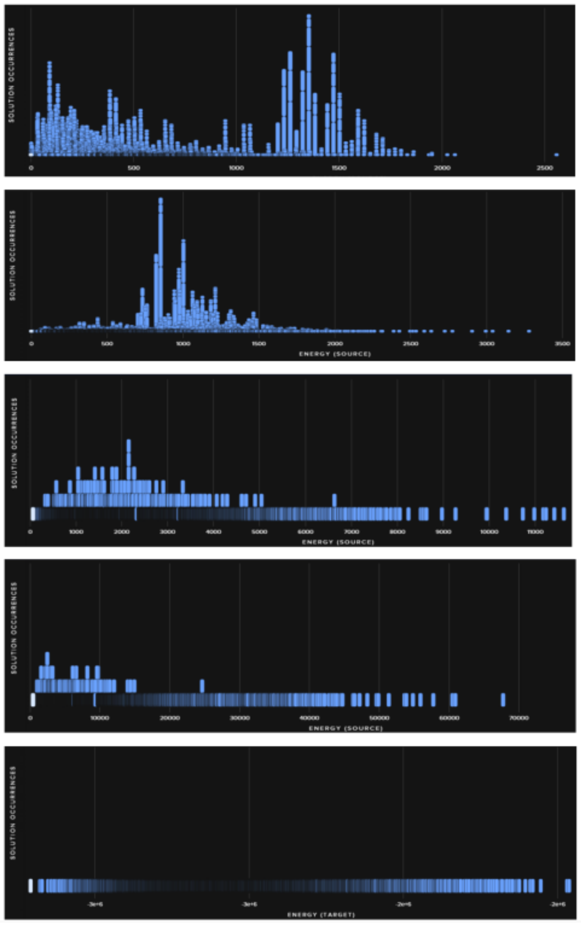


Figure A.1: Samples Energy Distribution, from top to bottom: $N = 16$, $N = 20$, $N = 32$, $N = 52$, $N = 92$. X-axis: energy level, Y axis: sample occurrences

A.2 Solution's payoff for $N = 92$

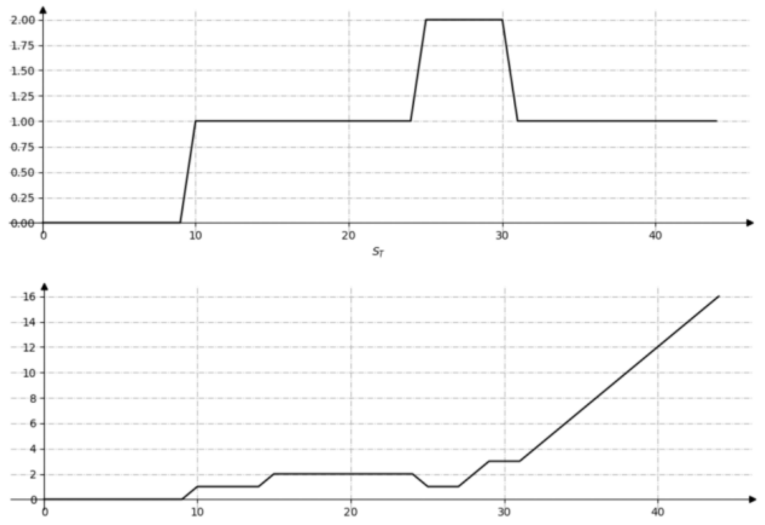


Figure A.2: $N = 92$: Payoffs of the ground solutions find by Simulated Annealing (top) and Quantum Annealing (bottom). Respective prices of the corresponding portfolios are 28.38 and 52.94

Bibliography

[1] Eugene Fama. *Efficient Capital Markets: a review of Theory and Empirical work*. 1970.

[2] Virtus Funds. *An Introduction To Merger Arbitrage*. 2021.

[3] Yagnesh Salian Pranit Mahajan, Vinayak Jadhav. *Currency Arbitrage Detection*. 2020.

[4] Marco Molinaro Alberto Del Pia, Santanu S. Dey. *Mixed-integer Quadratic Programming is NP*. 2018.

[5] Jack Jacquier. *Numerical Methods for Finance*. 2022.

[6] Valeriu Ungureanu. *Traveling Salesman Problem with Transportation*. 2006.

[7] Jack Jacquier. *The Principles of Quantum Mechanics*. 2022.

[8] Michael A. Nielsen Isaac L. Chuang. *Quantum Computation and Quantum Information*. 2000.

[9] Jack Jacquier. *The Principles of Adiabatic Quantum Computing*. 2022.

[10] Alba Cervera-Lierta. *Quantum Annealing*. 2018.

[11] Katrina Kramer. *Explainer: What is quantum tunnelling?* 2020.

[12] D-Wave. *The D-Wave 2000Q Quantum Computer*. 2017.

[13] D-Wave Systems Documentation. *D-Wave QPU Architecture: Topologies*.

[14] D-Wave. *Advantage Processor Overview*. 2022.

[15] CBOE. *https://www.cboe.com/delayed_quotes/spy/quote_table*. 2022.