Optimal Arbitrage Detection using Quantum Annealing

A Dissertation Submitted

for the Degree of

MASTER OF SCIENCE

in

Physics

by

Mehul Pandita

Roll No. 20191069

to

SCHOOL OF PHYSICS

INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH

PUNE - 411 008, INDIA

May 2024

CERTIFICATE

This is to certify that this dissertation entitled **Optimal Arbitrage Detection using Quantum Annealing** submitted towards the partial fulfillment of the BS-MS degree at the Indian Institute of Science Education and Research, Pune, represents original research carried out by Mehul Pandita under the supervision of Dr. Aniruddha Pant during academic year May 2023 to March 2024.

Pune - 411008

May 2024

Dr. Aniruddha Pant

DECLARATION

I, **Mehul Pandita, Roll No: 20191069**, hereby declare that, this report entitled **"Optimal Arbitrage Detection using Quantum Annealing"** submitted to Indian Institute of Science Education and Research Pune towards the partial requirement of **Master of Science** in **Physics**, is an original work carried out by me under the supervision of **Dr. Aniruddha Pant** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold academic ethics and honesty. Whenever a piece of external information or statement or result is used then, that has been duly acknowledged and cited.

Pune - 411 008 **Mehul Pandita**

May 2024

ACKNOWLEDGEMENT

I thank everyone who helped me see this project through to completion. I would like to first express my profound gratitude and deep regard to Dr. Aniruddha Pant and sincerely wish to acknowledge his vision, guidance, valuable feedback and constant support throughout the duration of this project.

I am indebted to my friends and family for their steadfast encouragement and time. I am lastly grateful to the Indian Institute of Science Education and Research Pune for providing the necessary resources and facilities to complete this project to the best of my ability.

Pune - 411008 **Mehul Pandita**

May 2024

ABSTRACT

This dissertation investigates the application of Quantum Annealing (QA) to the detection of optimal arbitrage opportunities, comparing its efficacy against classical approaches such as Simulated Annealing (SA). Arbitrage, a financial strategy exploiting price differences of identical or similar assets across markets, presents complex optimization challenges, traditionally tackled by classical computational methods. With the advent of quantum computing, QA emerges as a promising alternative, leveraging quantum mechanical principles to explore solution spaces more efficiently.

We formulate the Arbitrage Detection Problem (ADP) as a Binary Quadratic Model (BQM), employing D-Wave Systems' quantum processors for empirical evaluation. The study meticulously assesses the performance of QA and SA in terms of execution time and accuracy, with the latter benchmarked against brute-force methods where feasible.

Our findings indicate that while SA excels in computational speed, QA demonstrates a significant potential in navigating complex solution spaces to identify not only the most profitable arbitrage opportunities but also near-optimal solutions. This study not only highlights the current capabilities and limitations of quantum annealing in financial optimization but also sets the stage for future explorations into the scalability and practical applications of quantum algorithms in solving NP-hard problems prevalent in finance and beyond.

Keywords: Quantum Annealing, Simulated Annealing, Arbitrage Detection, Financial Optimization, D-Wave Systems, Quantum Computing, Binary Quadratic Model, Discrete Quadratic Model.

Contents

List of Figures

List of Tables

Notations and Abbreviations

Abbreviation	Meaning
BQM	Binary Quadratic Model
DQM	Discrete Quadratic Model
AD	Arbitrage Detection
E	Set of Edges
V	Set of Nodes (Assets, Currencies, Vertices)
QUBO	Quadratic Unconstrained Binary Optimisation
SA	Simulated Annealing
QA	Quantum Annealing

Chapter 1

Introduction

The topic of quantum computing is rapidly developing and has become quite popular in the previous several decades. Novel and enhanced approaches to numerous challenging issues can be achieved by harnessing the exceptional characteristics of quantum mechanics, like superposition and entanglement. In the broader area of quantum computing, a number of techniques and algorithms have emerged. One well-known example is the quantum annealing (QA) process, which is comparable to the simulated annealing (SA) technique. Like its classical equivalent, QA is an optimisation problem-solving metaheuristic. This is achieved by using quantum mechanic principles to find the lowest energy state over an energy landscape.

Theoretically, QA has been demonstrated to provide advantages, but their practical application is dubious due to the numerous constraints of the current state of quantum technology. The primary of those concerns would be its accuracy and performance in comparison to traditional methods.

1.1 Purpose

The aim of this study is to compare the use of QA to the classical approach of SA when solving a well-known NP-hard problem. The problem in question is the Arbitrage Problem. QA and SA are evaluated based on performance (defined by execution time) and accuracy (defined by comparison with Brute Force methods).

Evaluating and comparing QA and SA when solving the ADP has relevance for multiple reasons. We present data on the size of the prospective benefits of quantum computing at this point in time by comparing SA and quantum technology for real-world applications. The problem of classifying sets per specific rules that may be used in a wide range of applications is tackled by arbitrage detection. Therefore, determining better approaches to address this issue can be quite advantageous.

1.2 Scope

The objective of this research paper is to compare QA to SA when solving the ADP. This is accomplished by the implementation and measurement of one SA procedure and three distinct QA processes. The basis of the QA procedures is the translation of the ADP into three distinct models, which are executed on widely accessible D-Wave systems: the discrete quadratic model (DQM), the constrained quadratic model (CQM), and the binary quadratic model (BQM). The DQM is assessed on a D-Wave hybrid solver, which makes use of both classical and quantum resources, whereas the BQM is evaluated on the D-Wave Advantage QPU, which only uses quantum resources for executing the QA process.

This study will also cover the physics and math behind the QA and D-Wave systems. This work does not, however, include any theoretical study of time/space complexity or evidence of correctness for our implementations. When it comes to test cases, the limited number of qubits in D-Wave QPUs, as well as performance constraints, will limit the graphs in terms of both vertices and edges. Lastly, although we have carefully selected the values of the hyper-parameters in our implementations, we will not be talking about fine-tuning them.

Chapter 2

Background

2.1 Arbitrage Detection (AD)

Arbitrage is a financial strategy that capitalizes on price discrepancies of identical or similar assets across various markets to generate risk-free profits. The core principle underlying arbitrage is the simultaneous purchase and sale of an asset to exploit temporary price imbalances, theoretically ensuring a profit without exposure to risk. This strategy is predicated on the efficient market hypothesis, which postulates that asset prices should reflect all available information. However, market inefficiencies often lead to price disparities, creating windows of opportunity for arbitrageurs.

Arbitrage opportunities can be classified into several categories, each exploiting distinct market conditions:

- **Spatial Arbitrage** involves exploiting price differences of the same asset across

distinct geographical markets. These price discrepancies can arise due to variations in supply and demand, regulatory environments, or transaction costs between locations.

- **Temporal Arbitrage** takes advantage of price discrepancies of the same asset over time. Traders may purchase an asset expecting its price misalignment to correct in the near future, thereby realizing a profit.

- **Statistical Arbitrage** relies on mathematical models and quantitative strategies to identify price imbalances across correlated assets. This approach typically utilizes historical price data and statistical methods to predict future price movements and potential arbitrage opportunities.

- **Triangular Arbitrage**, particularly relevant in the foreign exchange market, exploits currency exchange rate discrepancies among three different currencies to realize a risk-free profit.

Identifying and exploiting arbitrage opportunities pose significant challenges in the modern financial landscape. The primary difficulty lies in the high-speed nature of contemporary markets, where price discrepancies often last for mere seconds before the market's natural mechanisms correct them. Consequently, successful arbitrage detection necessitates real-time data analysis and the ability to execute trades rapidly. This requirement underscores the need for sophisticated computational techniques capable of swiftly processing vast amounts of data to uncover potential arbitrage opportunities.

Moreover, the importance of arbitrage extends beyond the generation of profits for individual traders or firms. Arbitrageurs play a crucial role in enhancing market

efficiency by correcting price anomalies, thereby ensuring that asset prices accurately reflect their true value. This process contributes to the health and fairness of financial markets by eliminating unfounded price discrepancies and promoting price uniformity across different trading platforms.

Given the complexities and fast-paced nature of financial markets, traditional computational methods often fall short in effectively detecting arbitrage opportunities. This limitation has sparked interest in exploring advanced computational techniques capable of addressing these challenges. Among the most promising of these advanced methods is quantum computing, which offers the potential to improve the speed and efficiency of arbitrage detection significantly. The subsequent sections will delve into the role of quantum annealing and discrete optimization models in arbitrage detection, highlighting their advantages over traditional computational approaches.

The identification of arbitrage opportunities, a well-established problem within the field, involves determining whether at least one such opportunity exists and, if so, identifying it. A common approach to this problem involves defining a directed graph where nodes represent assets, and edge weights correspond to the negative logarithm of the conversion rate. Negative cycles in this graph signify the presence of arbitrage opportunities, which can be detected using algorithms such as Bellman-Ford or Floyd-Warshall. These methods terminate upon encountering the first negative cycle.

However, the task of finding the most profitable arbitrage opportunity, defined as the negative cycle with the most negative weight and of arbitrary length, is an **NP-hard problem**. Consequently, the time complexity of an algorithm designed to

solve this problem optimally is expected to be exponential. Practically, the distinction between merely detecting the existence of arbitrage opportunities and identifying the most profitable opportunity can be substantial. Consider a scenario where two arbitrage opportunities coexist, one offering a minuscule profit while the other presents a significant financial gain. While a trader would undoubtedly be interested in the larger profit, the aforementioned approaches (utilizing Bellman-Ford or Floyd-Warshall algorithms) would terminate upon identifying the first arbitrage opportunity, which could frequently be the less profitable option. Furthermore, our approach aims not only to identify the best (most profitable) arbitrage opportunity but also to uncover near-optimal solutions. This is achieved by leveraging the quantum annealer's capability to return a sample containing near-optimal solutions.

2.2 Simulated Annealing (SA)

Simulated Annealing (SA) is a metaheuristic optimization technique inspired by the annealing process in materials science, wherein temperature controls the structural changes in a material. As the temperature decreases, the material transitions from a state of high diffusivity, allowing significant alterations, to a state of low diffusivity, stabilizing its structure.

In computer science, this concept is applied to optimization problems by iteratively modifying a solution while systematically reducing the "system temperature." The process initiates from a starting state and gradually lowers the temperature to evaluate random solution adjustments. Every modification that improves the solution is accepted, while those of equal quality have a fifty percent acceptance rate.

Interestingly, deteriorations in the solution quality are conditionally accepted based on the current temperature, simulating the concept of "thermal energy." This mechanism aids in navigating out of local minima during higher temperatures and refines the solution as the temperature diminishes.

The conditional acceptance of suboptimal solutions is governed by a probability function, often expressed as $p(t) = e^{-\frac{1}{T}}$, where T represents the temperature. Early in the SA process, the algorithm is endowed with sufficient "thermal energy" to explore a broad range of solutions, including those less optimal, thereby avoiding premature convergence to local minima. As the temperature decreases, the algorithm becomes increasingly selective, honing in on the most promising solutions.

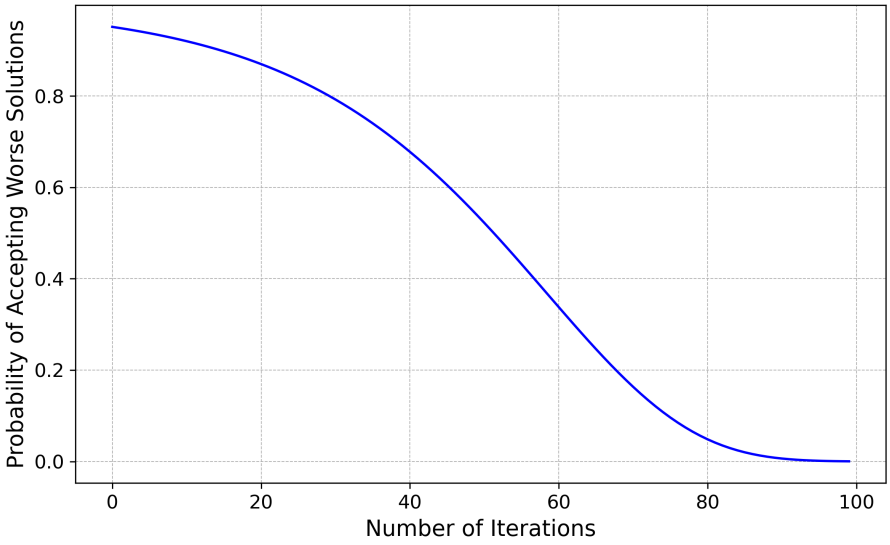

Figure 2.1: **The probability of accepting worse solutions decreases as the number of iterations increases**

In the context of arbitrage detection, SA's adaptive exploration strategy is particularly valuable. The algorithm's capacity to escape local optima and its probabilistic approach to solution acceptance make it adept at uncovering potentially profitable

arbitrage cycles within the complex, multidimensional space of financial exchanges. However, the performance of Simulated Annealing is contingent on the careful selection of parameters, such as the initial temperature and the cooling schedule. Furthermore, its probabilistic nature does not guarantee the identification of the absolute global optimum, and its efficiency may vary with the scale and specific characteristics of the optimization problem.

2.3 Quantum Annealing

Quantum annealing is a novel optimization technique inspired by the annealing process in materials science, where temperature controls the structural changes in a material. As a computational method, it employs the principles of quantum mechanics to explore the energy landscape of an optimization problem and identify low-energy solutions.

Quantum annealing is particularly well-suited for solving two broad classes of problems: **optimization** and **sampling problems**. In optimization problems, the objective is to find the best combination or configuration from a set of many possible solutions. Classic examples include scheduling challenges, such as determining the most efficient route for a traveling salesperson or optimizing the allocation of resources. By framing these problems as energy minimization tasks, quantum annealing can leverage the fundamental physical principle that systems tend to seek minimum energy states.

On the other hand, sampling problems involve characterizing the shape of the energy landscape and generating samples from low-energy states. This approach is

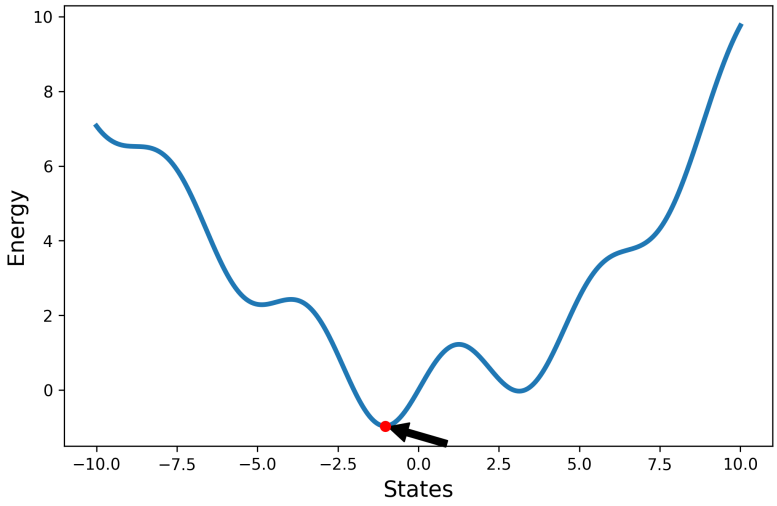

Figure 2.2: **Arbitrary energy-landscape with point of minimum energy highlighted by an arrow.**

valuable for machine learning applications, where probabilistic models are built to represent reality, accounting for uncertainties and noise in data sources. The samples obtained through quantum annealing provide insights into the model's state for a given set of parameters, enabling the refinement and improvement of the model.

2.3.1 Qubits

The core of the quantum annealing process lies in the behavior of quantum bits (qubits), which are the **fundamental units of information** in quantum computing. In D-Wave quantum processing units (QPUs), qubits are implemented as the lowest energy states of superconducting loops, with circulating currents representing the 0 and 1 states. These states have a circulating current and an associated magnetic field.

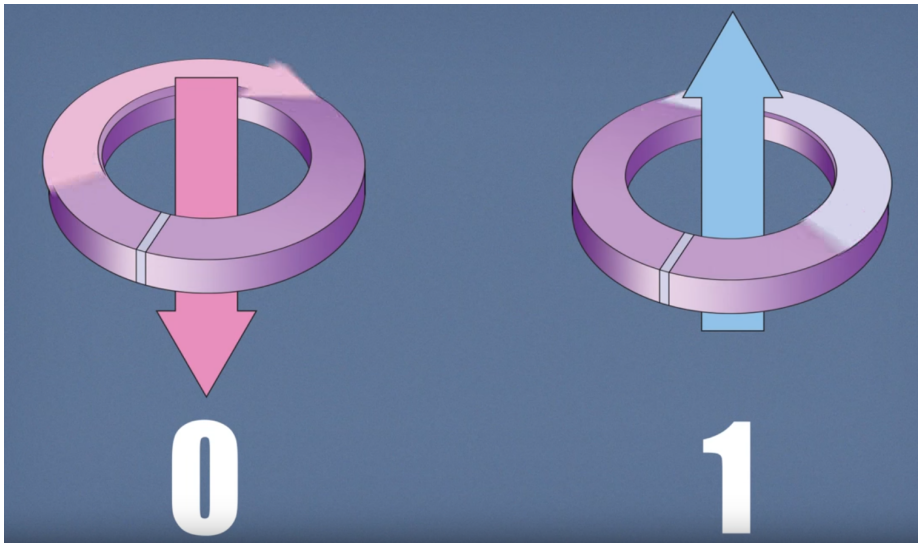

Figure 2.3: **Qubit states implemented as circulating currents (clockwise and anticlockwise) with corresponding magnetic fields**

Due to its quantum nature, a qubit can also be in a superposition of the 0 and 1 state at the same time. At the end of the quantum annealing process, each qubit collapses from a superposition state into either 0 or 1 (a classical state).

2.3.2 The Annealing Process

The foundational quantum mechanics governing the annealing mechanism can be articulated through the concept of the Hamiltonian. A classical Hamiltonian provides a mathematical framework for depicting a physical system by delineating its energies. This framework allows for the input of any specific state of the system, with the Hamiltonian yielding the corresponding energy for that state.

In the context of a quantum system, the Hamiltonian operates as a function that associates particular states, referred to as eigenstates, with their energies. The

energy of the system is precisely determined and termed as the eigenenergy exclusively when the system resides in an eigenstate of the Hamiltonian. Conversely, if the system occupies any state other than an eigenstate, the precision of its energy remains indeterminate. The assemblage of eigenstates that possess clearly defined eigenenergies constitutes the eigenspectrum.

The Hamiltonian of the D-Wave Quantum Processing Unit (QPU) is bifurcated into two distinct components: The initial component is known as the *initial Hamiltonian* (alternatively termed the tunneling Hamiltonian), which delineates the qubits' state of superposition. The subsequent component, referred to as the *final Hamiltonian* (or the problem Hamiltonian), encapsulates the optimization challenge intended to be addressed[8].

$$\mathcal{H} = \underbrace{-\frac{A(s)}{2}\left(\sum_i \sigma_x^{(i)}\right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i\sigma_z^{(i)} + \sum_{i>j} J_{i,j}\sigma_z^{(i)}\sigma_z^{(j)}\right)}_{\text{Final Hamiltonian}} \tag{2.1}$$

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

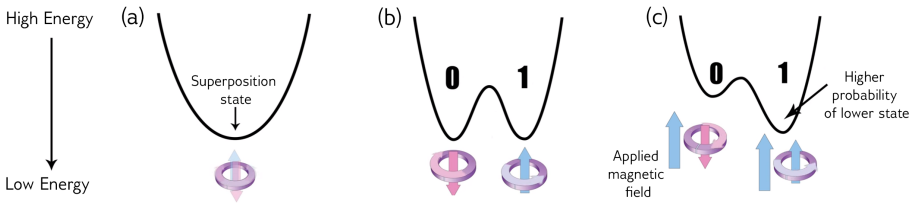

Figure 2.4: **Energy diagram changes over time as the quantum annealing process runs and a bias is applied.**

The underlying physics of quantum annealing can be depicted through an energy diagram. Three different configurations are marked as (a), (b), and (c). Initially, the

system displays a single valley (a), housing one minimum. As quantum annealing progresses, an energy barrier is introduced, morphing the landscape into a double-well potential (b). In this scenario, the minimum of the left well aligns with the quantum state 0, whereas the minimum of the right well aligns with state 1. Upon completion of the annealing process, the qubit will reside in one of these two valleys.

Assuming that all other conditions are the same, each qubit has an equal chance of settling into either the state 0 or the state 1, with a likelihood of 50% for each. Nevertheless, it is possible to influence this likelihood by applying an external magnetic field to the qubit. The effect of this field is to skew the qubit's double-well potential, thus altering the likelihood in favor of the qubit occupying the energetically more favorable well. The degree of this external influence is determined by a variable known as the bias, which enables the qubit to seek out a state of lower energy.

However, the utility of the bias is not fully realized in isolation. The essence of qubit functionality is unveiled when they are interconnected, thereby allowing for mutual influence. This interconnection is achieved using a component known as a coupler. A coupler's role is to predispose two qubits towards aligning in the same state, either both as 0s or both as 1s, or to predispose them to assume complementary states. Similar to the bias of an individual qubit, the extent of interaction between two coupled qubits is adjustable through what is termed as the coupling strength. It is the combination of these adjustable biases and coupling strengths that delineates a problem within the D-Wave quantum computational framework.

The annealing process begins with all qubits in a superposition state, governed by the initial Hamiltonian. As the process unfolds, a problem Hamiltonian is gradually

introduced, incorporating programmable biases and couplings that define the energy landscape of the specific problem being solved. Simultaneously, the influence of the initial Hamiltonian is reduced, allowing the qubits to become entangled and evolve toward the minimum energy state of the problem Hamiltonian. By the end of the anneal, each qubit collapses into a classical state, representing the optimal or near-optimal solution to the problem.

However, factors such as thermal fluctuations and rapid annealing rates can cause the system to deviate from the ground state, leading to excitations into higher energy states. The minimum gap between the ground state and the first excited state throughout the anneal is a crucial factor affecting the success of the quantum annealing process.

2.4 Quadratic Models

A Quadratic Model (QM) is a family of polynomials with one or two variables. Mathematically, they are defined as:

$$E(\mathbf{x}) = \sum_{\mathbf{i=1}} \mathbf{a_i x_i} + \sum_{\mathbf{i<j}} \mathbf{b_{i,j} x_i x_j} + \mathbf{c} \tag{2.2}$$

where a_i and $b_{i,j}$ are linear and quadratic coefficients respectively. x_i denotes the unknown variables of the polynomial.

It can be seen that the final Hamiltonian in equation 2.1, is a quadratic model with $\sigma_z^{(i)}$ as variables, h_i and $J_{i,j}$ as linear and quadratic coefficients respectively. There-

fore, to solve a problem with D-Wave QPU, it needs to be mapped to a quadratic model such that the specifics and constraints of the model are sufficiently represented.

2.4.1 Binary Quadratic Model (BQM)

A polynomial of the form (2.2) where variables x_i can take one of two values (binary)[2]. There are two further formulations for a BQM: Ising and QUBO. If $x_i \in \{-1, +1\}$, the BQM is said to be following the Ising formulation where the QUBO is defined by $x_i \in \{0, 1\}$. Both formulations of the BQM can be mathematically represented as

$$E(\mathbf{x}) = \sum_{\mathbf{i=1}} \mathbf{a_i x_i} + \sum_{\mathbf{i<j}} \mathbf{b_{i,j} x_i x_j} + \mathbf{c} \qquad \mathbf{x_i} \in \{\mathbf{-1}, \mathbf{+1}\} \text{ or } \{\mathbf{0}, \mathbf{1}\} \tag{2.3}$$

where c is a constant (offset).

The QUBO model can be conveniently represented as an upper-diagonal matrix Q, where the linear coefficients lie on the diagonal with quadratic coefficients being the off-diagonal ones.

$$\text{QUBO:} \qquad E(\mathbf{x}) = \sum_{\mathbf{i \leq j}} \mathbf{x_i Q_{i,j} x_j} \qquad \mathbf{x_i} \in \{\mathbf{0}, \mathbf{1}\} \tag{2.4}$$

2.4.2 Discrete Quadratic Model (DQM)

A polynomial of the form (2.2) where variables x_i can take any value from a set of discrete values[5]. The DQM is defined mathematically as:

$$H(\mathbf{d}) = \sum_{\mathbf{i}} \mathbf{a_i}(\mathbf{d_i}) + \sum_{\mathbf{i,j}} \mathbf{b_{i,j}}(\mathbf{d_i}, \mathbf{d_j}) + \mathbf{c} \tag{2.5}$$

where d_i are the discrete variables, $a_i()$ and $b_{i,j}()$ are real-valued functions, and c is a constant (offset).

Any DQM can be represented by an equivalent model over binary variables, one-hot encoding vectors of binary variable to replace each discrete variable d_i. In each vector, only one variable is True and all others are False. This changes the objective function to be minimized to this:

$$E(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{N}} \sum_{\mathbf{u=1}}^{\mathbf{n_i}} \mathbf{a_{i,u}} \mathbf{x_{i,u}} + \sum_{\mathbf{i=1}}^{\mathbf{N}} \sum_{\mathbf{j=i+1}}^{\mathbf{N}} \sum_{\mathbf{u=1}}^{\mathbf{n_i}} \sum_{\mathbf{v=1}}^{\mathbf{n_j}} \mathbf{b_{i,j,u,v}} \mathbf{x_{i,u}} \mathbf{x_{j,v}} + \mathbf{c} \tag{2.6}$$

where N is the number of discrete variables d_i, each with n_i cases. $x_{i,u}$ then indicates whether the discrete variable d_i is set to case u.

2.5 D-Wave Systems

Founded in 1999, D-Wave Systems Inc. is a leader in the development and delivery of quantum computing systems, software, and services. The company's focus on quantum annealing technology has enabled it to produce quantum computers designed to solve complex optimization problems that traditional computers cannot efficiently address.

D-Wave's technology, particularly its Quantum Processing Units (QPUs), uses quantum mechanics principles to perform computations. Unlike classical computers,

D-Wave's systems are optimized for problems characterized by a large number of variables and potential solutions, making them suitable for applications in logistics, machine learning, financial modeling, and material science.

As of now, the D-Wave collection contains two commonly accessible QPUs: D-Wave 2000Q, which follows the Chimera architecture, and Advantage, which follows the Pegasus architecture. The differences between the Chimera and Pegasus architectures are based on the number of qubits available and how they are stacked and connected.

D-Wave's python-based Ocean SDK [2] is used to run processes on these devices and may be accessible via the D-Wave Leap IDE or from their publicly available repository.

2.5.1 D-Wave Samplers

Samplers in the context of quantum computing are mechanisms that select from the low-energy states of a problem's objective function. Specifically, a Binary Quadratic Model (BQM) sampler is adept at identifying low-energy states within models defined either by Ising equations or Quadratic Unconstrained Binary Optimization (QUBO) frameworks. These samplers return a series of samples, organized by ascending energy levels.

The Ocean software suite, developed by D-Wave, offers an array of `dimod` samplers. These samplers are equipped with functionalities to sample from QUBO and Ising models through the `sample_qubo` and `sample_ising` methods, alongside a generic method for BQM sampling. Among the available tools is the `DWaveSampler()`,

alongside classical solvers designed for CPU or GPU execution. These classical solvers are particularly beneficial for code development or preliminary testing on smaller problem instances to ensure code accuracy.

Hybrid Quantum-Classical Samplers

The hybrid quantum-classical approach leverages both quantum and classical computational resources, drawing on the unique advantages of each. This method is particularly effective in tackling complex computational challenges.

D-Wave's Leap Quantum Application Environment enhances this strategy by offering advanced hybrid solvers capable of processing arbitrary BQMs. Additionally, the `dwave-hybrid` library provides a Python-based framework for crafting versatile hybrid workflows. These workflows combine quantum and classical resources, aiming to identify optimal solutions efficiently.

2.5.2 D-Wave QPU Architectures

Solving a problem using D-Wave QPUs requires the problem to be mapped to graph that represents the topology of the system's qubits. It may be the **Chimera** topology for D-Wave 2000Q systems, or the **Pegasus** topology for Advantage systems.

The QPUs consist of qubits and couplers. Couplers are shared resources among exactly two qubits which represents their entanglement. Depending on the specific problem representation of the form equation (2.2), the linear coefficients are assigned as the biases of the qubits whereas the quadratic coefficients form the bases for the coupler strengths. These biases and strengths govern the evolution of the QA process.

Chimera Architecture

The Chimera architecture is characterized by its unique assembly of connected unit cells, each consisting of four horizontal and four vertical qubits interconnected through couplers, achieving bipartite connectivity. These unit cells are systematically arranged in a grid, extending both vertically and horizontally, to form a sparse lattice network of qubits. Each unit cell can be visualized as either a cross or a column, facilitating an intuitive understanding of its structure.

Within the Chimera topology, qubits are organized into unit cells that create bipartite connections, leading to a standardized Chimera graph representation, such as C3, where qubits are sorted into nine unit cells. A notable feature of Chimera qubits is their nominal length of four, indicating that each qubit is internally connected to four orthogonal qubits. Additionally, each qubit has a degree of six, signifying its coupling with six distinct qubits.

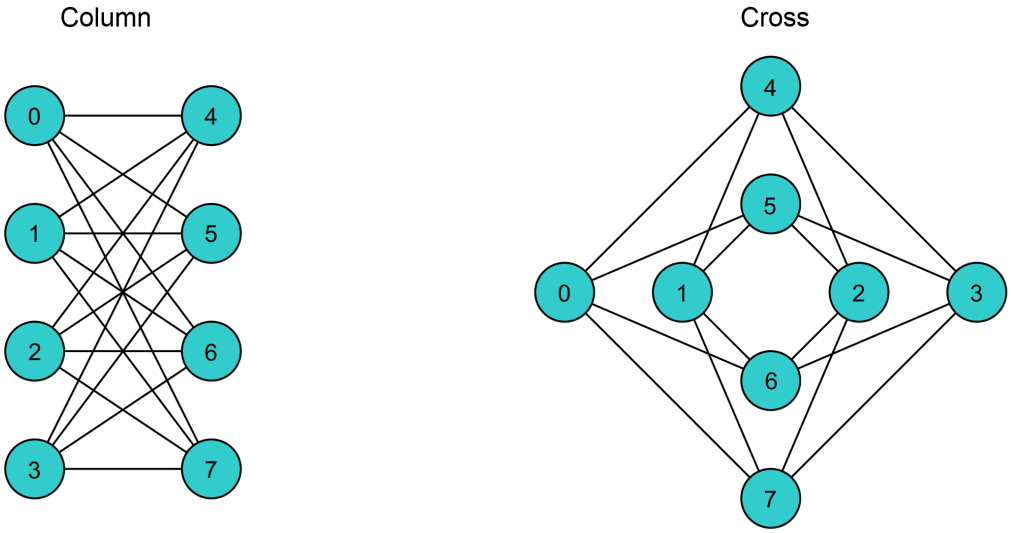

Figure 2.5: **Chimera Unit Cell**

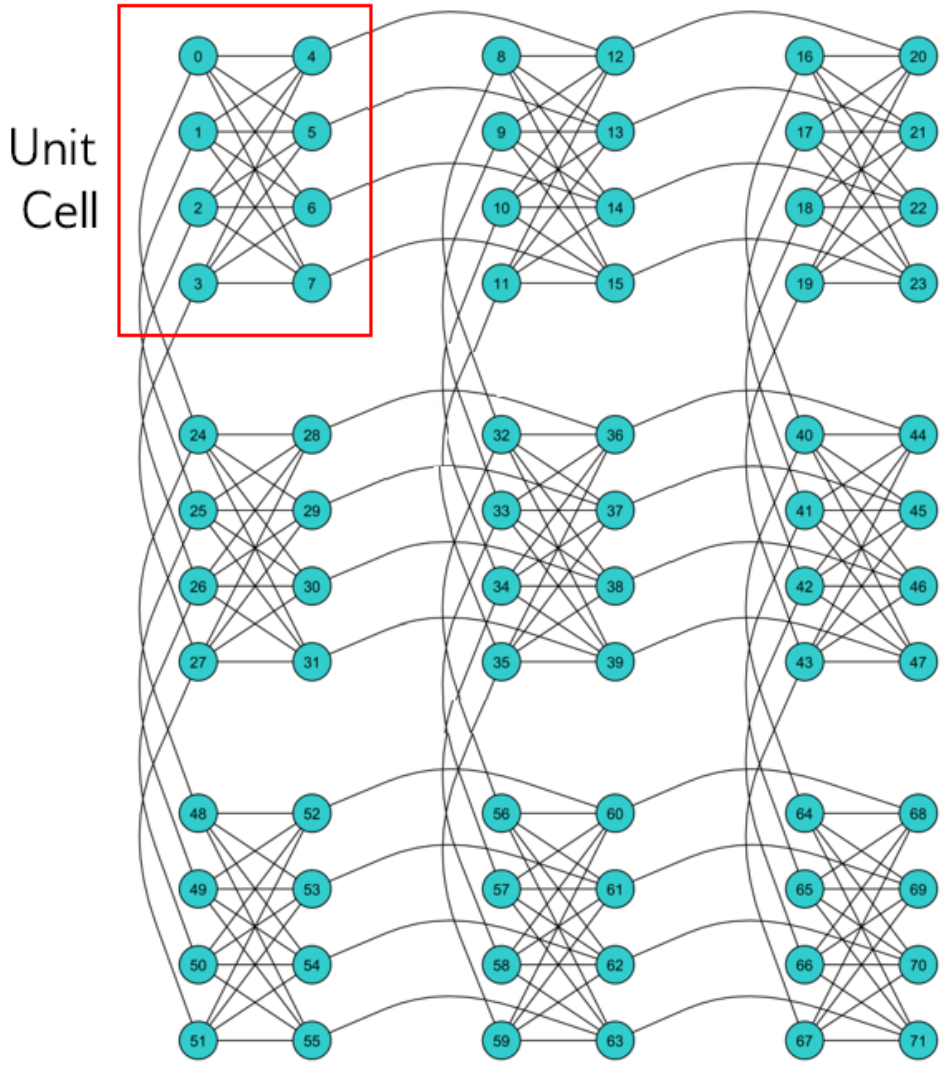

Figure 2.6: **C3: A 3x3 Chimera graph**

The term CN denotes a Chimera graph that consists of an N by N grid of unit cells. For instance, the D-Wave 2000Q QPU accommodates a C16 Chimera graph, incorporating 2048 qubits logically arranged into a 16 by 16 matrix of unit cells, each containing eight qubits.

Pegasus Architecture

Similar to the Chimera structure, Pegasus maintains vertical or horizontal orientations for qubits. However, Pegasus introduces variations in alignment shifts and groupings among similarly aligned qubits across different Pegasus families. This architecture enhances qubit connectivity and introduces three distinct types of couplers:

- **Internal couplers** connect qubits of opposite orientations. In the Pegasus architecture, each qubit is linked to 12 others through internal couplers, a notable increase from the four connections in Chimera.
- **External couplers** link vertical qubits to their vertical neighbors and horizontal qubits to their horizontal counterparts, with each qubit featuring one or two external couplers.
- **Odd couplers** enable connections between pairs of similarly aligned qubits, with each qubit associated with one odd coupler.

The Pegasus qubits, illustrated through various graphical representations, highlight the intricate connections facilitated by these couplers. Pegasus qubits exhibit a nominal length of 12, reflecting the internal connections to orthogonal qubits, and a degree of 15, indicating coupling to 15 different qubits.

To leverage a D-Wave QPU for solving a problem, the objective function's properties must be mapped onto the QPU's hardware. This prerequisite process, termed minor embedding, involves configuring the problem in a manner that aligns with the hardware's existing structure.

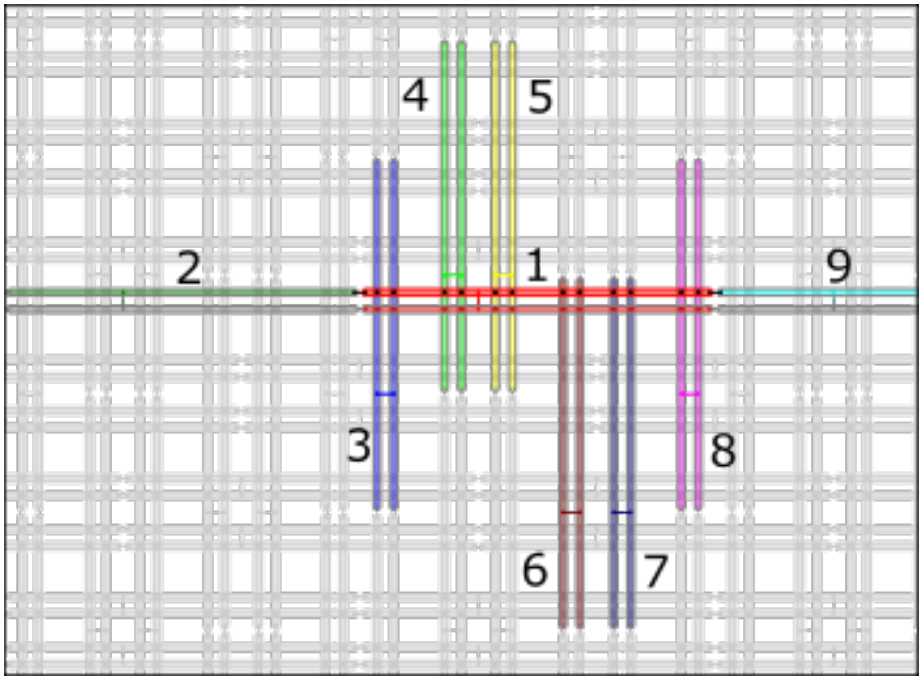

Figure 2.7: **Pegasus Qubits**[4]

Minor Embedding and Qubit Chains

To answer an arbitrarily posed binary quadratic problem directly on a D-Wave system, a mapping, known as minor embedding, to the quantum processing unit's (QPU) topology is required. This preprocessing can be performed by a constructed sampler that combines the DWaveSampler() with a composite that performs minor-embedding. (This phase is handled automatically by the LeapHybridSampler() and dwave-hybrid reference samplers.

In D-Wave's quantum processors, chain strength is a crucial parameter that directly impacts the coherence, entanglement, and overall performance of quantum computations. By setting an optimal chain strength, D-Wave aims to minimize chain breaks and ensure that qubits within chains can interact effectively to solve complex

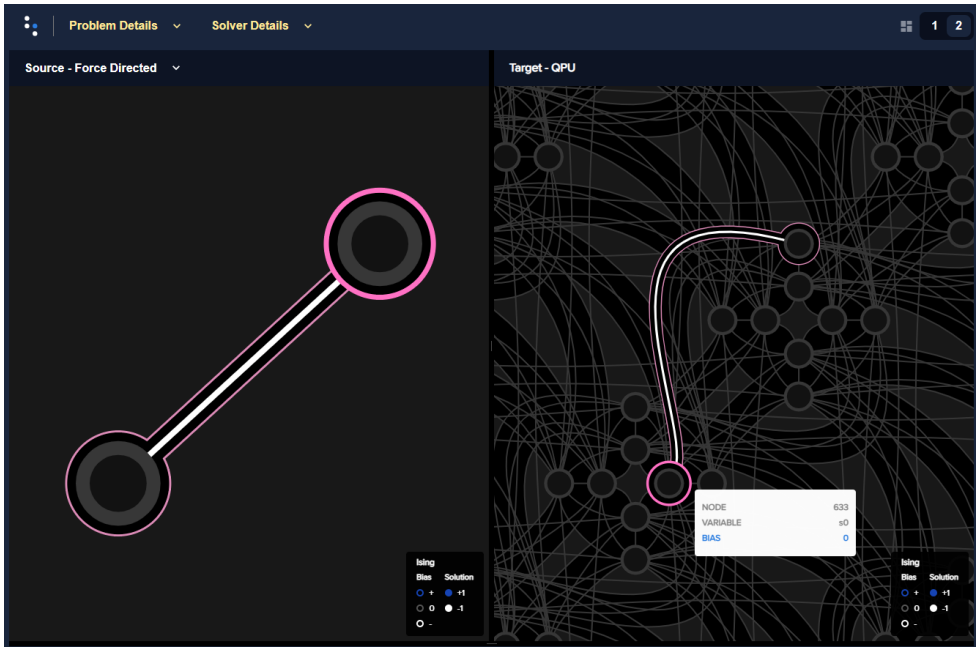

Figure 2.8: **The two-variable problem, represented as a graph on the left, is embedded in two connected qubits on an Advantage, represented on the right against the Pegasus topology.**[6]

optimization problems. Finding the right balance in chain strength is essential to maintain the integrity of the quantum problem representation and to reduce errors caused by noise and non-adiabatic dynamics, ultimately enhancing the stability and accuracy of quantum computations on their quantum processors.

2.5.3 Ising vs. QUBO in D-Wave Systems

The D-Wave systems are designed to work with models that follow the final Hamiltonian formulation of equation (2.2). That is the Ising formulation. This is due to Ising model being an accurate representation of the actual physical qubit spins (along withe their weights and interactions).

However, moving from a QUBO to Ising (and vice versa) is simple and just requires a variable transformation:

$$\textbf{Ising to QUBO: } x_i = \frac{1 + s_i}{2} \tag{2.7}$$

$$\textbf{QUBO to Ising: } s_i = 2x_i - 1 \tag{2.8}$$

where $i = 1..n$, $x_i \in 0, 1$ (QUBO) and $s_i \in -1, 1$ (Ising)

This transformation is automated when a QUBO formulation is passed to a D-Wave sampler.

2.6 Complete QA Process Workflow

To effectively tackle a problem using Quantum Annealing (QA), a sequence of steps is undertaken. A QM needs to be defined according to the problem instance. This results in a function whose low-energy states hold the solution to the problem. The QA process encompasses several key stages to identify and leverage these low-energy states:

1. **Conversion to Logical Graph:** Initially, the QM is transformed into a logical graph. In this representation, nodes symbolize variables, while edges represent the interaction terms between these variables.

2. **Minor Embedding:** The subsequent step involves embedding the logical interaction graph onto the physical Quantum Processing Unit (QPU), a process

detailed in Section 2.5.3. This adaptation ensures the problem's compatibility with the QPU's architecture.

3. **Programming and Initialization** This stage sets the parameters of the QM, culminating in the formulation of the final Hamiltonian, as discussed in Section 2.3.1. Parameter setting involves adjusting qubit biases to modulate the magnetic fields influencing the qubits and determining the strength of couplers. An initial state, subject to optimization, is established during this phase, a task efficiently managed by D-Wave's software suite.

4. **Annealing Process** At this juncture, the Ising model undergoes resolution through the system's self-transformation. By employing predefined annealing functions, the system endeavors to minimize the current state's energy, an operation performed autonomously by the D-Wave system.

5. **Readout** Following the annealing phase, the qubits attain an eigenstate or a superposition of eigenstates. These states can then be read out and subjected to post-processing to derive a solution.

6. **Resampling** To enhance the solution obtained, the processes detailed in steps 5 and 6 are iteratively repeated. Upon completing a predefined number of iterations, the solution is conclusively read out.

Through this structured approach, Quantum Annealing facilitates the identification of low-energy states that represent optimal or near-optimal solutions to complex problems, leveraging the unique capabilities of quantum computation.

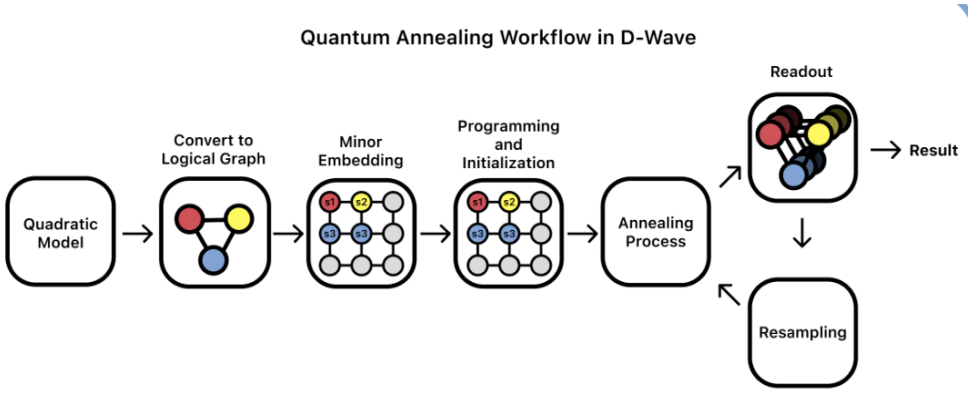

Figure 2.9: **QA Workflow**

Chapter 3

Methods

Before creating the various implementations, appropriate hardware and software had to be selected. To implement the two quantum annealing and simulated annealing processes, the systems provided by D-Wave were selected.

3.1 Problem Formulation

The formulation used here is built upon work done by Gill Rosenberg [1] demonstrating possible potential use-case of QA in the problem of Arbitrage Detection. The problem needs to be translated into a QM equation which can then be processed in the D-Wave software.

The meaning and representation of the linear and quadratic coefficients will differ for both formulations, i.e, BQM and DQM.

We construct a directed graph. Each node i denotes an asset and each edge is weighted by the conversion rate c_{ij}. A conversion rate of c_{ij} implies, each unit of asset i could be converted to c_{ij} units of j. In general, $c_{ij} \neq c_{ji}$.

The problem of finding the most profitable arbitrage path turns into the problem of maximising the product of conversion rates c_{ij} in a cycle.

3.1.1 BQM Formulation

Objective

Maximising the product of conversion rates is further simplified to maximising the logarithm of the product instead, making the problem linear.

Therefore, the objective is defined by

$$\max \sum_{(i,j)\in E} x_{ij} \log c_{ij}$$

where E is the set of edges.

Constraints

For finding a valid arbitrage cycle, there are two constraints that need to be followed:

1. **Exchange Validity Constraint:** Each currency can have at most one outgoing edge in the cycle.

2. **Cycle Completion Constraint:** The number of incoming edges must be equal to the number of outgoing edges for each currency (flow conservation).

Mathematically, the constraints are defined as:

$$\sum_{j,(i,j)\in E} x_{ij} = \sum_{j,(j,i)\in E} x_{ji} \qquad \text{for all } i \in V,$$

$$\sum_{j,(i,j)\in E} x_{ij} \leq 1 \qquad \text{for all } i \in V.$$

where V is the set of nodes (assets).

We convert this into a quantum unconstrained binary optimation (QUBO) format by putting the constraints in as penalty terms. On rearragment, squaring of first constraint and using non-negativity of the sum in the second constraint (allowing us to write it as a product), leads to the following function:

$$x = \underset{x}{\operatorname{argmax}} \left[\sum_{(i,j)\in E} x_{ij} \log c_{ij} - M_1 \sum_{i\in V} \left(\sum_{j,(i,j)\in E} x_{ij} - \sum_{j,(j,i)\in E} x_{ji} \right)^2 - M_2 \sum_{i\in V} \left(\sum_{j,(i,j)\in E} x_{ij} - 1 \right) \right],$$

where M_1 and M_2 are positive penalty coefficients.

3.1.2 DQM Formulation

Both formulations have the same fundamental objective and constraints. However, due to difference between allowed assignment of variables among the two, the way to encode them as QMs changes.

To incorporate these constraints in the DQM formulation, we can modify the definitions of variables, linear coefficients, and quadratic coefficients as follows:

Variables:

Let's define a set of variables $p \in P$, where each variable p represents a currency.

The value assigned to each variable p will represent the order or position of the corresponding currency in the cycle (or path). The possible values for each variable p will be integers from 0 to $n - 1$, where n is the total number of currencies.

Linear Coefficients:

The linear coefficients $a \in A$ in the DQM formulation will promote the assignment of lower values to the variables. We can define n linear coefficients, one for each possible value of the variables. The linear coefficient a_i for the value i can be set to i itself, i.e., $a_i = i$.

Quadratic Coefficients:

The quadratic coefficients $b \in B$ in the DQM formulation will enforce the constraints and objective of the Arbitrage Detection problem. For each pair of variables p_1 and p_2 representing currencies c_1 and c_2, respectively, we define a quadratic coefficient b based on the exchange rate between c_1 and c_2.

If there is an edge (currency exchange) between c_1 and c_2 with an exchange rate r, we set the quadratic coefficient b for the state where $|p_1 - p_2| = 1$ to $-log(r)$. This promotes the inclusion of profitable edges in the cycle.

For the state where $p_1 = p_2$, we set the quadratic coefficient b to a large positive value (or a penalty value) to prevent the same currency from appearing twice in the cycle (Exchange Validity Constraint).

For all other states of p_1 and p_2, we can set the quadratic coefficient b to a large positive value (or a penalty value) to discourage those assignments, as they violate the Cycle Completion Constraint.

By defining the variables, linear coefficients, and quadratic coefficients in this way, the minimal energy state of the DQM will correspond to a solution that represents a profitable cycle (or path) in the currency exchange network, if one exists.

3.1.3 SA Formulation

The `SimulatedAnnealingSampler` from D-Wave's `neal` software package applies the Simulated Annealing (SA) algorithm to the Arbitrage Detection problem. This problem is articulated through an Ising model or a Binary Quadratic Model (BQM), with the objective of identifying profitable arbitrage opportunities in currency exchange networks by minimizing a specifically defined energy function.

Algorithm Overview

The SA algorithm is a metaphorical equivalent to the metallurgical process of annealing. It is during this process that a material is thermally treated and then gradually cooled to attain a state of minimal energy. In optimization, the SA algorithm traverses the solution space, proposing and accepting or rejecting new solutions based on probabilistic rules.

SA Algorithmic Steps

The following steps detail the application of the SA algorithm:

1. Begin with an elevated temperature, $T_{\max}$, and initialize a randomly selected solution, referred to as *current_solution*.
2. Evaluate the energy, E_{current}, of *current_solution* via the problem's energy function.
3. Proceed iteratively until a stopping criterion is satisfied:
   (a) Generate *new_solution* by introducing small random changes to *current_solution*.
   (b) Calculate E_{new}, the energy of *new_solution*.
   (c) Establish $\Delta E = E_{\text{new}} - E_{\text{current}}$.
   (d) If $\Delta E \leq 0$, set *current_solution* = *new_solution*, indicating an energy reduction.
   (e) If $\Delta E > 0$, adopt *new_solution* with a probability of $\exp(-\Delta E/T)$, adhering to the Metropolis criterion.

(f) Update T by following the designated cooling schedule.

4. Output the optimal solution identified, indicative of a profitable arbitrage route.

The **cooling schedule** plays a crucial role in the performance of the Simulated Annealing algorithm. The `SimulatedAnnealingSampler()` supports three types of cooling schedules: linear, geometric, and custom. The linear schedule decreases the temperature linearly from T_{max} to a final temperature T_{min}, while the geometric schedule decreases the temperature exponentially. The custom schedule allows users to specify their own sequence of temperature values.

During the annealing process, the `SimulatedAnnealingSampler()` keeps track of the best solution found so far (the one with the lowest energy) and updates it whenever a better solution is encountered. The algorithm terminates when a specified number of sweeps (iterations) have been performed, or when a user-defined interrupt function (if provided) returns True.

The final result is a `dimod.SampleSet` object containing the best solution found (represented as a sample) and its corresponding energy value. The sample can be interpreted as the most profitable arbitrage cycle or path in the currency exchange network, satisfying the Exchange Validity and Cycle Completion Constraints.

3.2 Data Extraction

For the Arbitrage Detection problem in the realm of cryptocurrencies, obtaining real-time market data is pivotal. This section elucidates the process of extracting

ticker information from cryptocurrency exchanges, which includes bid and ask prices, as well as other relevant data points necessary for identifying potential arbitrage opportunities.

The data extraction was automated through a Python script utilizing the `ccxt` library, which provides a unified way of accessing different cryptocurrency exchange markets. A list of target currency pairs was pre-defined, including major cryptocurrencies traded against USD and each other, such as BTC/USD, ETH/BTC, and XRP/USDT among others.

A continuous extraction loop was implemented to fetch the latest market ticker data for these currency pairs. The script captures various attributes including the current bid price, ask price, last transaction price, high and low prices for the day, and the trading volume. This data is timestamped to maintain a record of market conditions at the time of data retrieval.

The following steps summarize the data extraction algorithm:

1. Initialize a connection to an exchange platform using the `ccxt.exchange()` method.
2. Retrieve ticker data for a predefined list of cryptocurrency pairs at regular intervals.
3. For each ticker, extract and store the bid price, ask price, last transaction price, daily high and low prices, and the trading volume.
4. Append the new data to a `pandas` DataFrame, ensuring time-series consistency.

5. Periodically save the accumulated data to a CSV file, appending new rows while avoiding duplicate header entries.

6. Employ error handling to manage and log potential connectivity issues or API limitations encountered during data fetching.

7. Utilize a sleep interval to manage the rate of API calls, adhering to the exchange's rate limit policy and ensuring data integrity.

Data extraction was executed over an extended period to gather a comprehensive dataset. The frequency of data retrieval and saving intervals were carefully selected to balance the need for up-to-date information against the operational constraints of API usage limits and system resources.

The accumulated data constitutes a historical record of market dynamics, serving as a primary input for the Arbitrage Detection algorithm. It encapsulates the fluctuations in cryptocurrency prices over time, forming the basis for subsequent analysis and decision-making processes in the identification of arbitrage opportunities.

Table 3.1: **Data Snippet from Kraken Exchange**

time	symbol	last	high	low	volume
1687413932	BTC/USD	30171.6	30794.9	28760.1	8971.21
1687413932	USDT/USD	0.99975	1.0004	0.99884	111685647.67
1687413932	BTC/USDT	30190.4	30758.9	28785.2	1224.09
1687413932	XRP/USDT	0.51162	0.52686	0.49105	956210.1124
1687413932	XRP/BTC	1.70E-05	1.75E-05	1.63E-05	950366.2115
1687413932	XRP/USD	0.51092	0.52638	0.49	10058898.46
1687413932	XRP/USDT	0.51162	0.52686	0.49105	956210.1124
1687413932	ETH/BTC	0.06313	0.0637	0.06168	5944.063477
1687413932	ETH/USD	1906.87	1930.88	1807.72	61078.06309
1687413932	ETH/USDT	1906.76	1932.38	1807.99	4970.165723
1687413932	LTC/USD	86.66	88.79	83.4	54935.574
1687413932	LTC/BTC	0.002876	0.002946	0.002817	5669.864227
1687413957	BTC/USD	30171.6	30794.9	28760.1	8892.329023
1687413957	USDT/USD	0.9997	1.0004	0.99884	111702202.3
1687413957	BTC/USDT	30175.8	30758.9	28785.2	1224.109709
1687413957	XRP/USDT	0.51162	0.52686	0.49105	956210.1124
1687413957	XRP/BTC	1.70E-05	1.75E-05	1.63E-05	950366.2115
1687413957	XRP/USD	0.51236	0.52638	0.49	10068680.18
1687413957	XRP/USDT	0.51162	0.52686	0.49105	956210.1124
1687413957	ETH/BTC	0.06313	0.0637	0.06168	5944.063477
1687413957	ETH/USD	1907.49	1930.88	1807.72	61083.46434
1687413957	ETH/USDT	1906.76	1932.38	1807.99	4970.165723
1687413957	LTC/USD	86.7	88.79	83.4	54939.3741
1687413957	LTC/BTC	0.002876	0.002946	0.002817	5669.864227

Chapter 4

Results and Analysis

This chapter presents the results obtained from applying the Arbitrage Detection algorithm on the extracted dataset, which comprises real-time bid, ask, and transaction prices for selected cryptocurrency pairs. The primary objective is to identify and analyze potential arbitrage opportunities within the cryptocurrency market.

4.1 Data Preprocessing

The extracted data was initially processed to construct a directed graph representation of the currency exchange network. Each node in this graph corresponds to a specific cryptocurrency, while directed edges represent exchange opportunities between currency pairs. The weight assigned to each edge denotes the logarithm of the corresponding exchange rate. This graph-based representation facilitates the algorithm's execution and the subsequent identification of profitable arbitrage cycles.

4.2 Computational Performance Evaluation

The analysis commences with a comparative evaluation of the computational efficiency between the Binary Quadratic Model (BQM) and Simulated Annealing (SA) algorithms, both employed for identifying arbitrage opportunities.

4.2.1 Computational Time

Figure 4.1 illustrates the average computational times required by each algorithm. The SA algorithm demonstrates a significant advantage in computational speed over BQM, highlighting its suitability for time-sensitive applications and real-time decision-making processes.

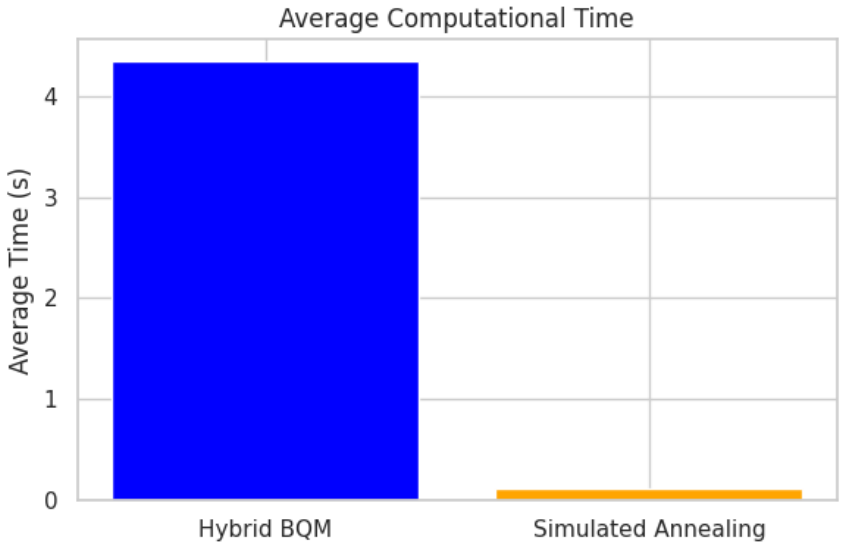

Figure 4.1: The average computational times for the BQM and SA algorithms.

Further analysis of the computational time distribution, as depicted in Figure 4.2, reveals a broader spread for the BQM algorithm, indicating greater variability in its

execution times. In contrast, the SA algorithm exhibits a more consistent and tightly clustered distribution, underscoring its reliability in maintaining low computational overhead.

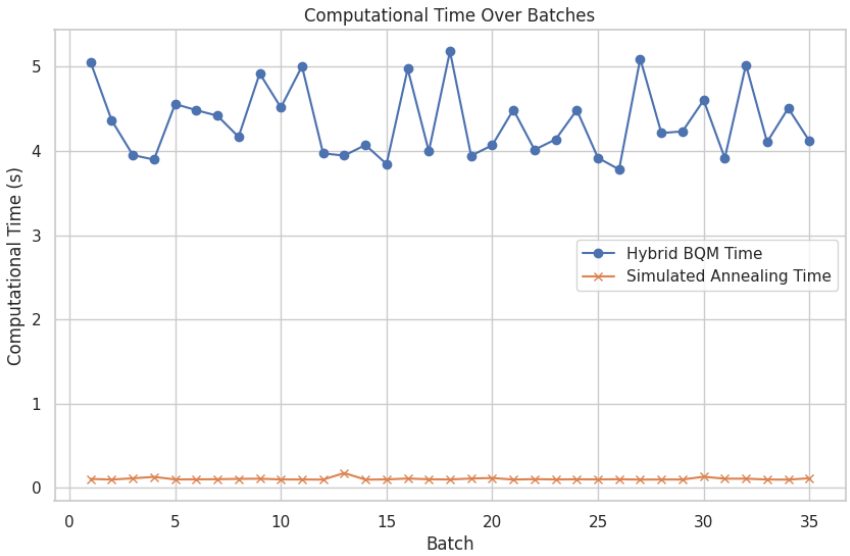

Figure 4.2: Computational time distribution for each algorithm, highlighting the consistency of the SA algorithm.

4.2.2 Profitability Analysis

The next aspect of the analysis focuses on the profitability of the arbitrage opportunities identified by each algorithm. Figure 4.3 presents a histogram displaying the distribution of profits generated by the two algorithms. Notably, both distributions exhibit a similar concentration of profitable outcomes above the break-even point, indicating their effectiveness in identifying lucrative arbitrage opportunities.

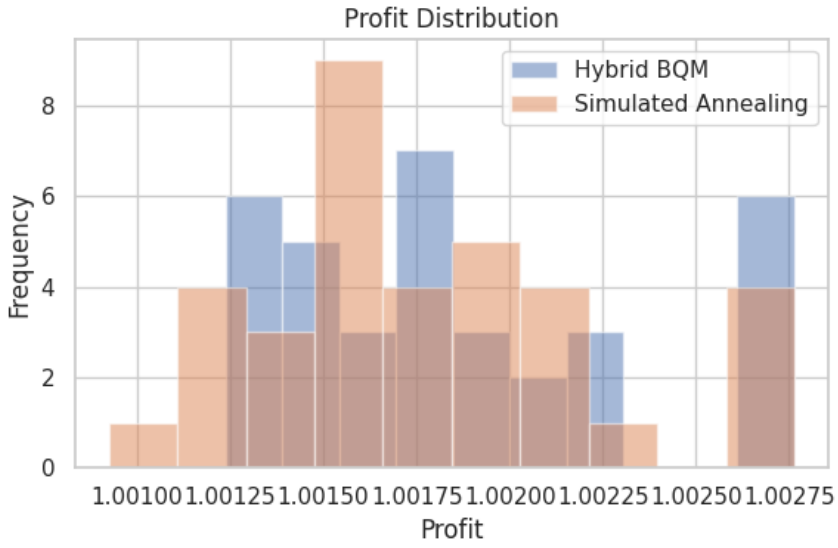

Figure 4.3: Histogram of profit distribution, comparing the performance of the BQM and SA algorithms.

4.2.3 Solution Complexity

To assess the complexity of the identified arbitrage opportunities, we analyze the cycle lengths of the solutions generated by each algorithm. Figure 4.4 presents the average cycle lengths, which exhibit little variation between the BQM and SA algorithms. This observation suggests that both algorithms are capable of identifying arbitrage opportunities with comparable levels of complexity, as measured by the number of currencies involved in the cycle.

4.2.4 Solution Quality

To evaluate the quality of the solutions obtained, we analyze the energy values associated with the identified arbitrage opportunities. The energy value serves as a

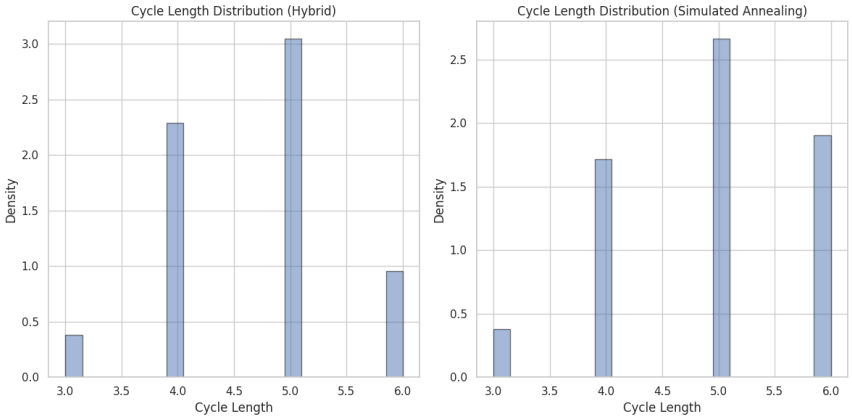

Figure 4.4: Average cycle lengths for the BQM and SA algorithms, indicating the complexity of the identified solutions.

measure of the optimization quality, with lower values indicating more optimal solutions. Figure 4.5 presents a line graph depicting the sorted energy values of the solutions generated by each algorithm. The results demonstrate a slight advantage for the BQM algorithm in finding solutions with lower energy values, suggesting its potential for identifying higher-quality arbitrage opportunities.

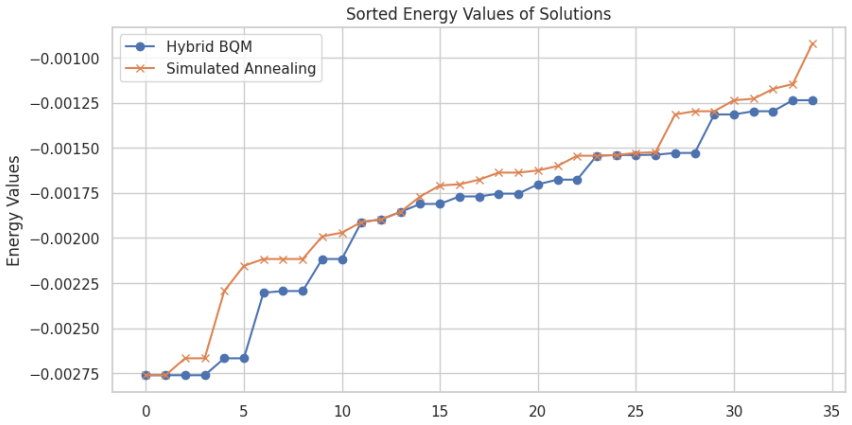

Figure 4.5: Sorted energy values of solutions from the BQM and SA algorithms, illustrating the optimization quality of each algorithm.

4.2.5 Efficiency Analysis

To further evaluate the performance trade-offs between the two algorithms, we introduce an efficiency ratio metric that balances the profitability of the identified arbitrage opportunities against the associated computational time. Figure 4.6 presents a scatter plot depicting the efficiency ratio for each solution generated by the BQM and SA algorithms. The results suggest that while the SA algorithm generally exhibits faster execution times, the BQM algorithm occasionally achieves higher efficiency ratios, particularly in more complex scenarios involving longer arbitrage cycles.

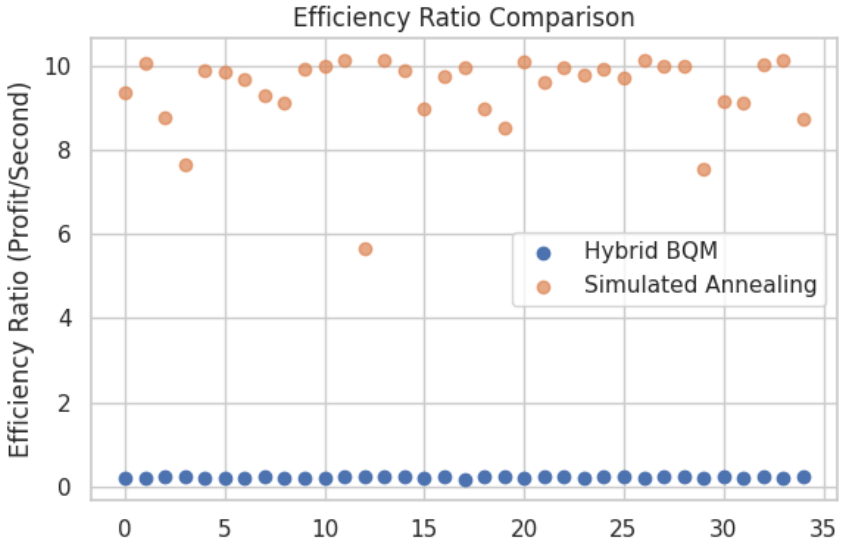

Figure 4.6: Efficiency ratio of the BQM and SA algorithms, balancing profit against computational time.

4.3 Summary and Insights

The results presented in this chapter provide valuable insights into the performance and trade-offs associated with the BQM and SA algorithms for detecting arbitrage opportunities in cryptocurrency markets. While the SA algorithm demonstrates superior computational speed and consistency, the BQM algorithm exhibits potential advantages in identifying higher-quality solutions and achieving greater efficiency in complex scenarios.

These findings underscore the importance of carefully evaluating the specific requirements and constraints of the application domain when selecting an appropriate algorithm for arbitrage detection. Time-sensitive applications may benefit from the computational efficiency of the SA algorithm, while scenarios where solution quality is of paramount importance may warrant the use of the BQM algorithm.

Furthermore, the analysis highlights the potential for combining the strengths of both algorithms or exploring hybrid approaches to leverage their respective advantages, ultimately enhancing the overall effectiveness of arbitrage detection in dynamic and rapidly evolving cryptocurrency markets.

Discussion

Our comparative analysis of the BQM and SA algorithms reveals nuanced strengths and limitations in their application to identifying arbitrage opportunities. SA's speed and consistency make it a valuable tool for rapid analysis. However, BQM's ability to find slightly more optimized solutions suggests its utility in scenarios where the

quality of the solution is paramount. Both algorithms effectively uncover profitable opportunities, with no significant difference in the complexity of solutions identified.

For the present generation of quantum annealers, the API call time-gap creates a substantial time delay as compared to local SA methods. This rules out its use on short-lived opportunities, which arise in highly liquid markets (such as crude oil or gasoline futures). However, even very-high-speed traders are frequently unable to access them since the time required to notice and respond to them is more than the time required for the market to correct itself.

Further research might explore the scalability of these algorithms, the impact of parameter tuning on their performance, and their application to a broader range of optimization problems. This study underscores the potential of quantum-inspired and classical algorithms in financial optimization, offering a foundation for future advancements in the field.

Bibliography

[1] Rosenberg, Gill. *Finding optimal arbitrage opportunities using a quantum annealer*, 1QBit Research

[2] D-Wave Systems. *Binary Quadratic Models*. https://docs.ocean.dwavesys.com/en/stable/concepts/bqm.html#bqm-sdk

[3] D-Wave Systems. *D-Wave Ocean Software Documentation*. https://docs.ocean.dwavesys.com/en/stable/

[4] D-Wave Systems. *D-Wave QPU Architecture: Topologies*. https://docs.dwavesys.com/docs/latest/c_gs_4.html

[5] D-Wave Systems. *Discrete Quadratic Models*. https://docs.ocean.dwavesys.com/en/stable/concepts/dqm.html

[6] D-Wave Systems. *Minor-Embedding*. https://docs.ocean.dwavesys.com/en/stable/concepts/embedding.html

[7] D-Wave Systems. *QPU Solvers: Decomposing Large Problems*. https://docs.dwavesys.com/docs/latest/handbook_decomposing.html

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

[9] D-Wave Systems. *Workflow: Formulation and Sampling.* https://docs.dwavesys.com/docs/latest/c_gs_workflow.html

[10] Gyongyosi, Laszlo and Imre, Sandor. *A Survey on quantum computing technology.* In: Computer Science Review 31 (2019), pp. 51–71. issn: 1574-0137. doi: 10.1016/j.cosrev.2018.11.002. https://doi.org/10.1016/j.cosrev.2018.11.002.

[11] Kadowaki, Tadashi and Nishimori, Hidetoshi. *Quantum annealing in the transverse Ising model.* In: Physical Review E 58.5 (Nov. 1998), pp. 5355–5363. doi: 10.1103/physreve.58.5355. https://doi.org/10.1103%2Fphysreve.58.5355.