FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Quantum Algorithms for Optimizing Urban Transportation

Bruno André Moreira Rosendo

Master in Informatics and Computing Engineering

Supervisor: Rui Maranhão

July 24, 2024

Quantum Algorithms for Optimizing Urban Transportation

Bruno André Moreira Rosendo

Master in Informatics and Computing Engineering

July 24, 2024

Abstract

Transportation is a fundamental aspect of modern urban life, profoundly influencing the daily experiences of countless individuals residing in major cities. Its pivotal role extends beyond mere convenience, as transportation systems significantly shape energy consumption patterns and substantially impact the environment. Our choices in optimizing transportation affect the efficiency of our daily commutes and play a critical role in determining the sustainability of our cities and the planet's well-being. As cities grow and face escalating congestion, energy usage, and environmental sustainability challenges, exploring innovative solutions becomes imperative. Within this context, the convergence of quantum computing stands out as a compelling pathway for transportation optimization.

This dissertation explores the application of quantum algorithms to optimize urban transportation through variations of the Vehicle Routing Problem (VRP). The study formulates four VRP variations using the Quadratic Unconstrained Binary Optimization (QUBO) framework, particularly addressing the base VRP, the Capacitated VRP (CVRP), Multi-Capacitated VRP (Multi-CVRP), and Ride Pooling Problem (RPP).

A comprehensive literature review provides context, covering quantum computing paradigms, hardware, and existing classical and quantum approaches to VRP. The practical implementation employs Quantum Approximate Optimization Algorithm (QAOA) and Quantum Annealing (QA), leveraging IBM's and D-Wave's platforms. A novel software architecture was developed, enabling flexible encoding and solving of QUBO formulations on both quantum and classical systems.

Experimental analysis demonstrates that quantum hardware has limitations, particularly with QAOA's overhead, but Quantum Annealing shows promising results. Real-world applicability is tested using data from Porto's urban transport network, highlighting the potential of hybrid quantum-classical methods for medium-sized problems. Although large-scale optimizations remain challenging due to hardware constraints, the findings lay a foundation for future research and practical applications in urban settings. The dissertation concludes with insights into hardware improvements, algorithmic enhancements, and further research directions that could significantly advance the field of quantum urban optimization.

Keywords: *Quantum Computing, Algorithms, Optimization, Urban Transportation, Vehicle Routing, Qubo, Qaoa, Quantum Annealing, CVRP, RPP, D-Wave, Porto, Energy Consumption, Carbon Emissions*

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Rui Maranhão, for his guidance and allowing me to explore this fascinating topic in my dissertation. His support was essential to the completion of this work.

I would also like to thank the professors at Technische Universität Wien, especially Professor Vincenzo De Maio, for providing me with a solid education in the fundamentals of quantum computing during my Erasmus semester. Their teaching laid the foundation upon which this dissertation was built.

Special thanks to Professor Hiu Yung Wong from San Jose State University for his dedication to education and for making a wealth of educational material freely available. This initiative is immensely beneficial to the community and greatly appreciated, as it significantly aided me in building this dissertation.

Finally, I am deeply grateful to my family and friends for their unwavering support and encouragement throughout the development of this dissertation. Their belief in me kept me motivated and focused during the challenges I faced. In particular, I want to thank my parents and sister for their constant support at home and in my personal life. I also extend my heartfelt gratitude to my colleagues and friends, João Mesquita and Rui Alves, who stood by me through numerous challenges during this Master's degree and provided invaluable support throughout the thesis development.

Bruno Rosendo

"When different experiments give you the same result, it is no longer subject to your opinion. That's the good thing about science: It's true whether or not you believe in it. That's why it works."

Neil deGrasse Tyson

Contents

List of Figures

List of Tables

Abbreviations and Symbols

2E-CVRP	Two-Echelon Capacitated Vehicle Routing Problem
ACVRP	Asymmetric Capacitated Vehicle Routing Problem
API	Application Programming Interface
BKS	Best Known Solution
BQM	Binary Quadratic Model
CQM	Constrained Quadratic Model
CSV	Comma-Separated Values
CVRP	Capacitated Vehicle Routing Problem
CVRPTW	Capacitated Vehicle Routing Problem with Time Windows
DCVRP	Distance-constrained Capacitated Vehicle Routing
HTML	HyperText Markup Language
HTTPS	Hypertext Transfer Protocol Secure
ILP	Integer Linear Programming
IQEA	Improved Quantum Evolution Algorithm
ISA	Instruction Set Architecture
JSON	JavaScript Object Notation
KP	Knapsack Problem
Multi-CVRP	Multi-Capacitated Vehicle Routing Problem
MTZ	Miller-Tucker-Zemlin
NF	Not Feasible
NISQ	Noisy Intermediate-Scale Quantum
NP	Nondeterministic Polynomial Time
P	Polynomial Time
PSO	Particle Swarm Optimization
SA	Simulated Annealing
QA	Quantum Annealing
QC	Quantum Computing
QAOA	Quantum Approximate Optimization Algorithm
QEA	Quantum Evolution Algorithm
QEC	Quantum Error Correction
QPU	Quantum Processing Unit
QUBO	Quadratic Unconstrained Binary Optimization
Qubit	Quantum Bit
RAM	Random Access Memory
RCESPP	Resource Constrained Elementary Shortest Path Problem
SDK	Software Development Kit
STCP	Sociedade de Transportes Coletivos do Porto
TSP	Traveling Salesman Problem
UML	Unified Modelling Language
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problem with Backhauls
VRPPD	Vehicle Routing Problem with Pickup and Delivery
VRPTW	Vehicle Routing Problem with Time Windows

Chapter 1

Introduction

Transportation is a fundamental pillar of urban life, profoundly influencing daily routines, shaping energy consumption patterns, and affecting environmental sustainability. As cities grow, the need for innovative solutions becomes increasingly apparent. While essential for facilitating mobility, current methodologies face limitations in addressing the complex issues associated with urban expansion. Traffic congestion, prolonged commuting times, escalating energy consumption, and environmental concerns are critical challenges that demand creative solutions. Transforming the transportation sector is crucial for enhancing efficiency and promoting environmental sustainability.

In the current landscape of urban transportation, efforts to optimize efficiency are evident in systems such as metro, tram and bus lines. These modes of transport have undergone optimization processes aimed at minimizing delays, reducing travel times, and maximizing overall system efficiency. Implementing algorithms for streamlining route planning, scheduling, and resource allocation has demonstrated notable improvements.

Amid this dynamic environment, the emergence of quantum computing presents a compelling frontier in urban transportation. Integrating quantum computing with transport optimization can further enhance the sector by reducing commuting times, energy consumption, and carbon emissions. The Vehicle Routing Problem (VRP) and its various variations are at the forefront of this convergence. Its recent exploration within the quantum domain reveals significant potential. This intersection of established transportation challenges and cutting-edge quantum technologies creates a fertile ground for developing innovative solutions to drive future efficiency and sustainability in urban transportation systems.

1.1 Motivation

The motivation for exploring quantum algorithms to solve routing problems stems from the potential quantum computers offer. Quantum or hybrid implementations of combinatorial problems promise significant speedups beyond the capabilities of classical computers, with the potential to significantly enhance transportation optimization.

The rapid progress in quantum technology spurs interest in this field. Recent breakthroughs and accelerated development in quantum computing require ongoing investigation and analysis to effectively harness the capabilities of quantum algorithms for complex optimization problems. This imperative drives the implementation of solutions based on current and expected near-future quantum technologies.

Moreover, the versatility of the routing problems extends across diverse logistical domains, including shipment services and emergency evacuations, underscoring its broad relevance. Quantum-enhanced VRP's adaptability to various logistical scenarios further amplifies its motivation for exploration and application.

1.2 Objectives

The foundation of this dissertation is built upon a set of specific goals aimed at exploring the capabilities of quantum algorithms in addressing vehicle routing problems (VRP). This study is motivated by the aspiration to enhance the urban transportation infrastructure and to identify potential enhancements or expansions for existing public transport networks, particularly in urban centres such as Porto. The approach extends beyond implementing quantum solutions for VRP, thoroughly evaluating their performance compared to classical technologies and examining their scalability on current quantum hardware.

To achieve the main goal, the following objectives have been defined:

1. **Quantum algorithm development for routing problems:** The primary objective is to design and implement quantum and hybrid algorithms tailored explicitly to solve the VRP and its variations. This entails harnessing quantum computational properties such as superposition and entanglement to address the inherent complexities of combinatorial problems.

2. **Performance evaluation on state-of-the-art quantum hardware:** The dissertation aims to rigorously assess the performance of the developed algorithms on various state-of-the-art quantum hardware platforms, including gate-based superconductors and quantum annealers. The study seeks to evaluate the scalability of current quantum technology and anticipate scalability with future advancements, considering the computational resources required for current problem sizes. Comparative analyses against classical computing counterparts will provide insights into the advantages and limitations of quantum solutions.

3. **Application to real-world scenarios:** To validate the practical feasibility of the developed algorithms, this dissertation aims to apply them to real-world scenarios in the urban city of Porto. Using actual data from Porto's urban transport system, the objective is to formulate problem instances that serve as inputs for the quantum algorithms. This practical application will provide valuable insights into quantum technology's current capabilities and potential applications in real-world settings.

1.3 Contributions

The primary contributions of this dissertation are as follows:

- Formulation of four variations of routing problems using the Quadratic Unconstrained Binary Optimization (QUBO) framework. These variations include the Vehicle Routing Problem (VRP), Capacitated Vehicle Routing Problem (CVRP), Multi-Capacitated Vehicle Routing Problem (Multi-CVRP), and Ride Pooling Problem (RPP).
- Development of a software framework that employs quantum algorithms to solve QUBO problems using the Quantum Approximate Optimization Algorithm (QAOA) and Quantum Annealing (QA). The algorithms are implemented with IBM's Qiskit and D-Wave's Ocean software kits, and the framework is designed to be extensible to other QUBO formulations and quantum computing platforms.
- Performance analysis of the quantum algorithms on state-of-the-art quantum hardware. This study evaluates the scalability of the algorithms on IBM's and D-Wave's platforms, comparing the results against benchmarks.
- Application of the quantum algorithms to real-world scenarios, utilizing data from Porto's urban transport system.

1.4 Document Structure

After the Introduction where the context, motivation, objectives, and contributions of the study are explained, this dissertation comprises six additional chapters.

The second chapter, Background & State of the Art, outlines the necessary background and related work. It includes three sections: Background covers urban transportation optimization and quantum computing basics; Classical Methods and Techniques reviews related classical work in the topic; and Quantum Methods and Techniques examines quantum computing approaches to similar problems.

The third chapter, Modelling Vehicle Routing Problems, describes various routing QUBO formulations suitable for quantum computing. It discusses four specific VRP variations, detailing their mathematical formulations and relevance to quantum algorithms.

The fourth chapter, Software Design for Quantum Routing Algorithms, details the integration and implementation of the quantum algorithms. It includes sections on the software architecture (Software Architecture), implementation of QAOA (QAOA Implementation) and Quantum Annealing (Quantum Annealing Implementation), use of classical tools for comparison (Classical Implementation), and handling of solutions (Solution Management and Visualization).

The fifth chapter, Results and Discussion, presents results from the developed algorithms using benchmark datasets. It compares quantum approaches to best-known solutions and introduces custom benchmarks for additional VRP variations. Sections include OR-Tools Results, QAOA

Results, Quantum Annealing Results, and a comparison of the different solutions (Overview of Benchmark Results).

The sixth chapter, Incorporating Porto's Urban Transports, demonstrates the practical application of the algorithms using real data from Porto's urban transport system. It details the processing of metro and bus system datasets (Understanding and Processing the Datasets) and presents solutions both visually and quantitatively (Applying VRP Models to Porto's Urban Transports).

The seventh and final chapter, Conclusions and Future Work, summarizes the main findings of the study and offers recommendations for future research.

Chapter 2

Background & State of the Art

This chapter aims to present a thorough and contemporary understanding of the theme explored in this dissertation. It is organized into three main sections.

The first one, Background, is structured into distinct topics, providing insights into the key concepts of the dissertation's theme. The second and third sections, Classical Methods and Techniques and Quantum Methods and Techniques, explore current methodologies addressing the issues outlined in this dissertation, utilizing classical and quantum computation, respectively. This analysis is based on research papers published in recent years.

2.1 Background

This section offers an overview of the key concepts related to the dissertation topic, providing the reader with a solid foundation for understanding the subsequent work. Specifically, the section is organized as follows:

- **Urban Transportation**: Explores the functioning of contemporary urban transportation in major cities and outlines existing challenges in these systems. It then explains the vehicle routing problem and a few of its variations.
- **Quantum Computing Principles**: Explains the fundamentals of quantum computing, detailing how it works, mainly focusing on the gate-based approach, and its benefits compared to classical computing.
- **Quantum Annealing**: Explores the principles of Quantum Annealing as a significant alternative to the gate-based model. It delves into its operational mechanisms and advantages for solving optimization problems.
- **Quantum Hardware**: Examines the current status of hardware for quantum computers in the NISQ era, highlighting available technologies and their constraints. Offers a detailed explanation of the hardware principles used in IBM and D-Wave's quantum computers.

- **Quantum Algorithms**: Explores established quantum algorithms tailored for addressing optimization tasks, such as optimizing urban transportation, and goes into detail on the QUBO model and the QAOA algorithm.

2.1.1 Urban Transportation

Urban transportation in major cities is a multifaceted system that involves a combination of private cars, public transport, and pickup services provided by modern smartphone applications, influencing the daily commutes of countless individuals. The intricate interplay of these transportation mechanisms significantly shapes energy consumption patterns and has substantial environmental impacts. As cities struggled with congestion, sustainability challenges, and the need to optimize daily travel, exploring innovative solutions became increasingly imperative [31, 15].

According to a study conducted in the United States on transit usage post-2020, if public transportation accounts for 29-34% of all urban trips, it is feasible to save 9.42-15.42 gigatons of carbon dioxide equivalent emissions from cars. This transition would also result in significant cost savings for consumers. With the population of urban areas projected to grow until 2050, addressing this issue becomes increasingly crucial [31, 59]. Technological innovation is essential to overcoming this challenge, as advancements in the area are crucial to creating more sustainable and efficient transportation systems [6].

Since it focuses on optimizing public transportation, the core issue addressed in this research aligns with the **Vehicle Routing Problem (VRP)**, a well-established optimization challenge in the field. VRP involves efficiently allocating resources, particularly public vehicles, to pick up passengers from various locations within a defined area. While the primary aim is to minimize commuting times, optimization can be pursued with alternative metrics, such as energy consumption or carbon emissions. A detailed explanation of this problem is provided in the following section.

2.1.1.1 Vehicle Routing Problems

The Vehicle Routing Problem (VRP) involves the deployment of a fleet of vehicles to collect goods or passengers from diverse locations. The challenge inherent in VRP lies in identifying the most optimal set of routes for these vehicles to achieve an objective function, as shown in Figure 2.1, often focused on optimizing factors such as travel distance and time. Due to its NP-hard nature, finding the globally optimal solution with classical algorithms is considered inefficient, making the quest for high-quality approximate solutions complex and demanding [7].

If vehicles are constrained by limited capacity, as is often the case, the problem is referred to as the **Capacitated Vehicle Routing Problem** (CVRP). The CVRP encompasses the Bin Packing Problem and the Traveling Salesman Problem (TSP) as specific instances, representing a conceptual convergence of these well-studied problems. The integer programming formulation of this routing model introduces capacity constraints that serve as the crucial link connecting the fundamental routing and packing structures underlying the problem [80].

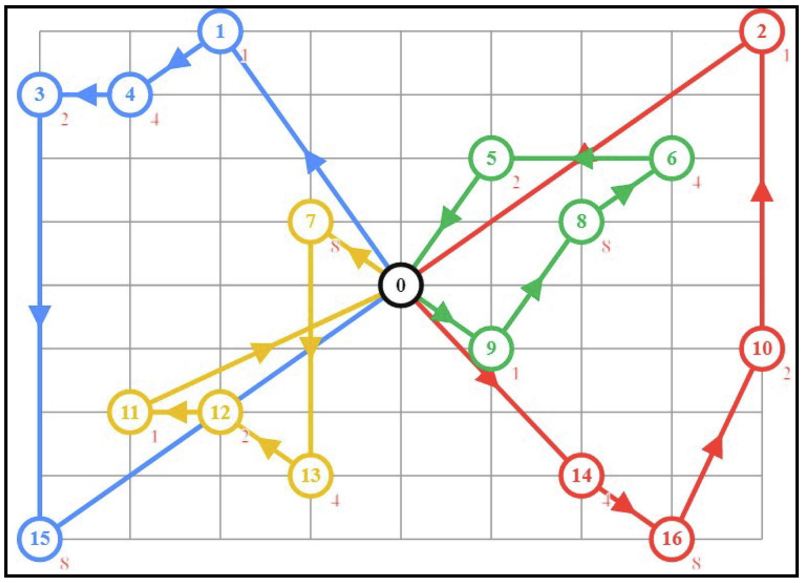

Figure 2.1: Example of an output of the VRP [75].

To extend the scope of the CVRP, various constraints can be integrated into the optimization problem. These constraints may include time windows (VRPTW), pickups and deliveries (VRPPD), stock replenishment (VRPB), and route maximum distance (DCVRP), resulting in different variants of the original problem (as depicted in Figure 2.2) [75]. These additional constraints enhance the algorithm's capability to dynamically allocate resources based on real-time demand and address flexible requirements at the cost of additional complexity.

Another closely related routing problem, albeit with slight differences, is the **Ride Pooling Problem** (RPP), which involves the absence of a central depot and allows multiple customers to request on-demand pickups and drop-off points from a shared fleet of vehicles [13]. This variation more accurately mirrors transportation systems like metro or bus lines and ride-sharing mobile applications. An illustrative example of a solution to this problem can be seen in Figure 2.3.

While exact solutions for CVRP can be obtained through advanced solvers for problem sizes of up to approximately 150 passenger locations with multiple vehicles, execution times may become impractical depending on the problem specifications. The most effective heuristic solutions consistently yield solutions within 1% of the optimal or best-known solutions, requiring an average of 10 minutes of CPU time for problems involving 50–199 passengers [7].

In addition to its applications in transportation optimization, the VRP finds utility in various contexts, including first-mile/last-mile services, emergency evacuation procedures, and shipment services. The versatility of the problem extends its relevance beyond conventional transportation scenarios, making it a valuable tool in addressing diverse logistical challenges.

2.1.2 Quantum Computing Principles

Quantum computing represents a novel paradigm in computation, drawing on the principles of quantum mechanics. Unlike classical computing, which relies on bits as binary units, quantum

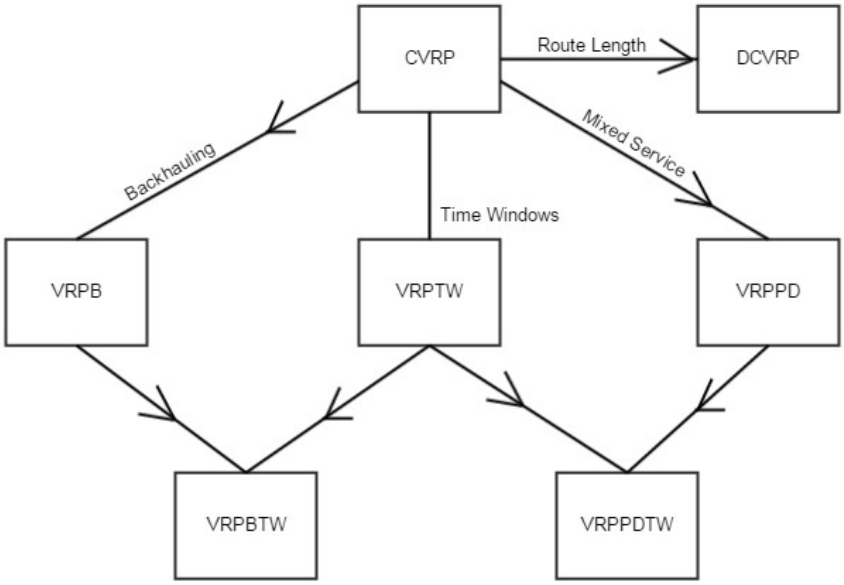

Figure 2.2: Variations of the Capacitated Vehicle Routing Problem [55].

computers utilize qubits capable of existing in multiple states simultaneously. This unique property, known as superposition, enables quantum computers to conduct parallel computations. Additionally, the phenomenon of entanglement allows qubits to be interconnected, facilitating more efficient problem-solving. This section provides an in-depth exploration of quantum computing, clarifying its essential attributes, potential applications, and inherent limitations.

2.1.2.1 Quantum Bits

A quantum bit, or qubit, is a fundamental unit of information in quantum computing. It is depicted as a unit vector within a two-dimensional complex vector space, with a specific basis denoted by $\{|0\rangle, |1\rangle\}$. In the domain of quantum computation, the basis states $|0\rangle$ and $|1\rangle$ are utilized to represent classical bit values 0 and 1, respectively [83]. Nevertheless, quantum bits can also be expressed in an alternative arbitrary basis, for example, $\{|+\rangle, |-\rangle\}$.

The qubit's basis representation relies on the **bra-ket notation**, a concept introduced by Dirac for expressing a Hilbert space vector in quantum mechanics [26]. Alternatively, the basis can be conveyed in matrix form:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.1}$$

2.1.2.2 Superposition

In contrast to classical bits, which can only be in states 0 or 1, qubits have the unique ability to exist in a superposition of $|0\rangle$ and $|1\rangle$. This characteristic enables qubits to occupy multiple states concurrently, represented as a linear combination of all possible states.

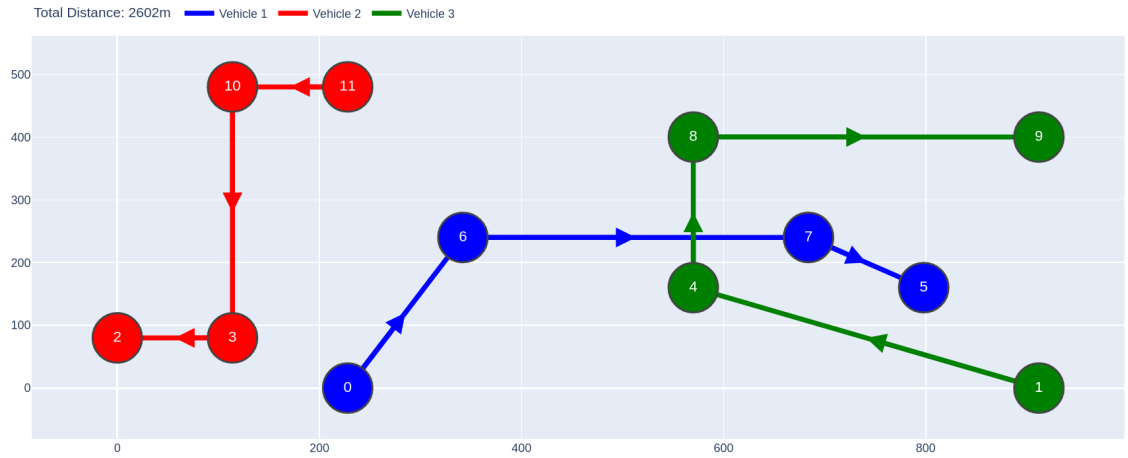

Figure 2.3: Example of an output of the RPP.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \equiv \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \tag{2.2}$$

where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$ [83]. Intuitively, $|\alpha|^2$ and $|\beta|^2$ denote the probabilities of the quantum position collapsing to that state when measured, as further explained in Section 2.1.2.4. It is important to note that superposition is basis-dependent, signifying that a qubit may exist in a superposition state on one basis but not in another [82].

As only two numbers are required to represent a qubit, it can be mapped onto an arrow originating from the origin and extending to the three-dimensional sphere in $\mathbb{R}^3$ with a radius of 1, commonly referred to as the **Bloch sphere**. Each qubit can be depicted using two angles that define the orientation of this arrow, as shown in Figure 2.4.

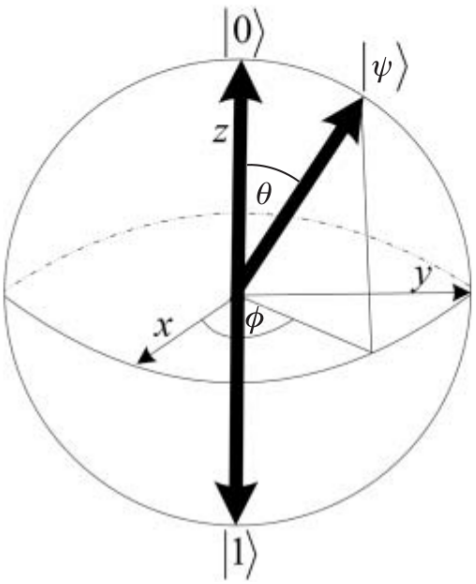

Figure 2.4: Bloch sphere representation [98].

where $0 \leq \phi < 2\pi$ and $0 \leq \theta < \frac{\pi}{2}$. That said, the superposition can be written as shown in Equation 2.3 [32].

$$|\psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\phi}sin\frac{\theta}{2}|1\rangle \tag{2.3}$$

2.1.2.3 Multi-qubit States

A system comprising n qubits can exist in a superposition of all 2^n possible states, demonstrating the true potential of quantum computation [83]. Consider a system with two qubits, where there are four possible states: $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, similar to a classical computer. However, in quantum systems, the state can be a superposition of all these possibilities:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \sigma|11\rangle \tag{2.4}$$

Here, α, β, γ, and σ are complex numbers, and the sum of their squares equals 1. Mathematically, a basis state with n qubits is expressed as the tensor product of its individual qubits:

$$\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\} = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\} \tag{2.5}$$

2.1.2.4 Measurement

In quantum computing, the measurement process stands out as a particularly counter-intuitive aspect. In contrast to classical computers, wherein the state of a system can be repeatedly determined at will, quantum systems, particularly qubits, only allow the determination of a single characteristic at a time. For instance, it is feasible to ascertain whether the system is in the state $|0\rangle$ or $|1\rangle$. However, this measurement is inherently random and alters the system's state, thus making further measurements impossible [32].

In the measurement process, the likelihood of the system being in a given state corresponds to the square of the respective coefficients, as shown in Equation 2.3. The measurement of systems comprising multiple qubits can be approached by conducting a sequence of individual bit measurements within the chosen basis [83]. Alternatively, certain qubits can be measured through the use of entanglement, as detailed in Section 2.1.2.7. The outcome can then be stored in a classical register.

From a mathematical perspective, the measurement process is described by the observable $\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ [32]. An observable represents a measurable physical quantity defined by a Hermitian operator A ($A^\dagger = A$) acting on a Hilbert space [63]. The potential outcomes of the measurement correspond to the eigenvalues of the state's matrix.

The measurement outcome depends on the selected basis, and the result will vary accordingly. While conducting measurements on an arbitrary orthonormal basis is possible, only one basis can be employed. This limitation arises due to the destruction of the initial superposition state

following the measurement, a characteristic sharply contrasting with the behaviour of classical systems.

2.1.2.5 Quantum Gates

We have observed quantum states that change solely upon measurement. Nevertheless, the most valuable aspect of a quantum system lies in its ability to undergo dynamic changes prior to measurement, facilitated by the application of **quantum gates**. In mathematical terms, quantum gates are linear transformations that maintain orthogonality, referred to as **unitary** transformations. These transformations can be represented by unitary matrices whose conjugate transpose equals its inverse ($U^{\dagger}U = I$). Unitary transformations can also be conceptualized as rotations within a complex vector space. A crucial implication of these transformations is their reversibility; hence, quantum gates must inherently be reversible [83].

Quantum gates perform operations analogous to classical logical gates, such as AND, OR, XOR, and NOT. In this context, the input and output of the gates are quantum states, and they are characterized by matrices of dimensions 2^k, where k denotes the number of qubits in the state. While quantum gates can be constructed arbitrarily, a standard set of gates is commonly employed, particularly those detailed in table 2.1, which operate on either 1 or 2 qubits [69].

2.1.2.6 Quantum Circuits

Quantum circuits are employed to visually depict a sequence of quantum gates and measurements applied to qubits. These circuits serve as a model of quantum computation based on classical circuits. Each qubit has its quantum wire, upon which corresponding quantum gates are applied. Additionally, a wire is commonly used to denote the classical register for storing measurement values. An illustrative example is presented in Figure 2.5.

Quantum circuits should be interpreted chronologically from left to right. The vertical axis corresponds to the qubits, with their order subject to variation in different implementations. In this context, the top qubit is considered the most significant. Multi-qubit gates are typically defined by connecting the relevant qubits with a vertical line.

2.1.2.7 Entanglement

Quantum entanglement is a physical phenomenon manifested when multiple qubits exhibit correlated states. The implications of superposition and entanglement suggest the potential for quantum computers to outperform classical computers, achieving notable speedup. Through entanglement, qubits become correlated, revealing hidden information. This property is a crucial advantage in quantum computing and is harnessed by numerous algorithms [45, 50].

In intuitive terms, if one qubit's measurement influences the measurement of another, then they are considered entangled [83]. A straightforward illustration of this phenomenon is the following state, recognized as one of the *Bell* states:

Gate	**Description**	**Matrix**
Identity (I)	Simplest gate, does not alter the state. Can be adapted to multiple qubits.	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Hadamard (H)	Creates an equal superposition when given a basis state (e.g. $\|0\rangle$).	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Pauli-X (X)	Quantum equivalent of the NOT classical gate for the basis $\{\|0\rangle, \|1\rangle\}$. When applied, rotates the x axis of the Bloch sphere by π radians.	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	When applied, rotates the y axis of the Bloch sphere by π radians.	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	When applied, rotates the z axis of the Bloch sphere by π radians. Leaves the basis state $\|0\rangle$ and maps $\|1\rangle$ to $-\|1\rangle$, thus being also called phase-flip.	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Phase (S, P)	Family of single-qubit gates that leave the basis state $\|0\rangle$ and map $\|1\rangle$ to $e^{i\phi}\|1\rangle$. The probability of measurement is unchanged after this gate, but the phase of the state is modified.	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$
CNOT	Applies the X gate on the second qubit if the first one is $\|1\rangle$. Otherwise, both are unchanged.	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
SWAP	Swaps two qubits, with respect to the $\{\|0\rangle, \|1\rangle\}$ basis.	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table 2.1: Most common quantum gates.

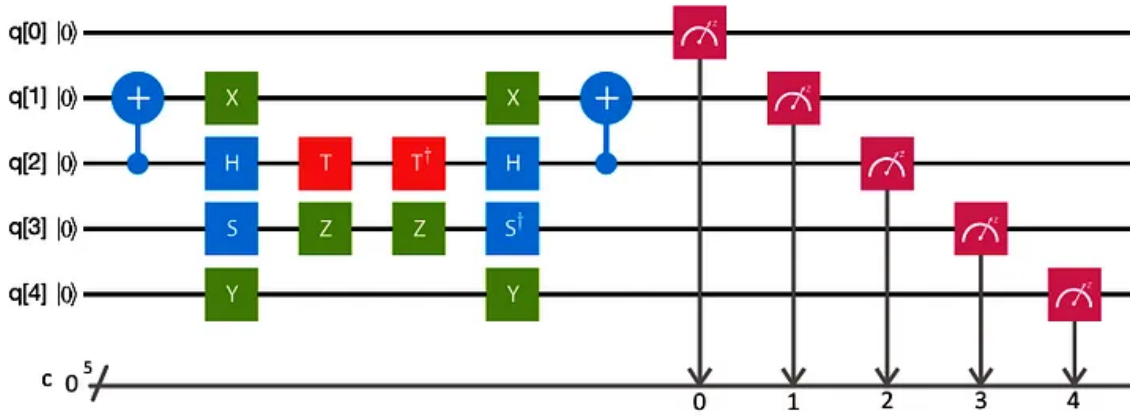

Figure 2.5: Example of a quantum circuit [85].

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.6}$$

In this scenario, measuring one of the qubits immediately discloses the state of the other, even without directly measuring it. Before any measurement, the probabilities of both qubits being in states $|0\rangle$ or $|1\rangle$ are each 50%. If, for instance, the first qubit is measured and found in the state $|0\rangle$, then the second qubit is guaranteed to be in the state $|0\rangle$ with 100% probability. This highlights the concept that entanglement unveils concealed information, a capability beyond the reach of classical computers.

From a mathematical perspective, a multi-qubit state is deemed entangled if it cannot be represented as a product state, i.e., it is not separable. A separable state can be expressed as a tensor product of basis states:

$$|\psi_{\text{separable}}\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \tag{2.7}$$

Nevertheless, establishing whether a general multi-qubit state can be expressed as a product state can be challenging. A more straightforward test for entanglement involves checking whether measuring one qubit alters the probability distribution of the second qubit. If the test is affirmative, the system is considered entangled. However, a negative result does not necessarily rule out entanglement, as hidden information might be associated with the coefficients' signs, which do not impact the probability distribution [45].

Creating entanglement between two qubits can be achieved quite straightforwardly. A simple method involves applying a Hadamard gate to the first qubit and subsequently applying a CNOT gate between the first and second qubits. This procedure generates the previously mentioned *Bell* state, as depicted in the circuit illustrated in Figure 2.6.

Including the previously provided example, four *Bell* states hold fundamental significance in quantum computing. These states are crucial as they represent the most straightforward instances of entanglement and play a key role in essential algorithms such as quantum teleportation and

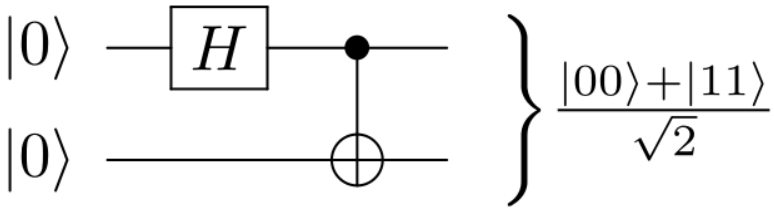

Figure 2.6: Quantum circuit generating an entangled state [43].

superdense coding [82]:

$$\begin{aligned}
|\Phi^+\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
|\Phi^-\rangle &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\
|\Psi^+\rangle &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\
|\Psi^-\rangle &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)
\end{aligned} \tag{2.8}$$

2.1.2.8 Benefits of Using Quantum Computation

As explained earlier, the power of quantum computation lies in the non-classical encoding of information in qubits, allowing computations to harness quantum effects like entanglement that are not accessible in classical computing [8]. Numerous algorithms leverage these effects; however, quantum computers are not intended to replace classical counterparts. Instead, quantum computation aims to complement classical computing, addressing tasks inherently inefficient for classical systems or enhancing the efficiency of existing operations. Consequently, hybrid systems have gained prominence by combining classical and quantum components. This approach allows real-world applications to capitalize on the strengths of each paradigm, employing quantum algorithms when they provide a speedup over classical counterparts (quantum advantage).

In Section 2.1.5, we explore examples of prominent algorithms pertinent to transport optimization. Additionally, Section 2.3 delves into current applications within the broader context of related work.

2.1.3 Quantum Annealing

The two leading paradigms in quantum computing are the gate-based model, explored in Section 2.1.2, and Quantum Annealing, based on an adiabatic time evolution of the system [99]. In contrast to gate-based machines, which are universal and applicable across a wide range of tasks, quantum annealers specialize in optimization tasks [18].

Quantum Annealing is a metaheuristic designed for solving or sampling intricate optimization problems, similar to simulated annealing in classical computing [35]. While traditional simulated annealing uses thermal fluctuations to converge on solutions, Quantum Annealing relies on a different mechanism, quantum tunnelling [11, 51]. This effect is illustrated in Figure 2.7.

Quantum Tunnelling: A phenomenon in quantum systems where particles pass through a potential energy barrier they classically should not be able to cross, due to their wave-like properties [67].

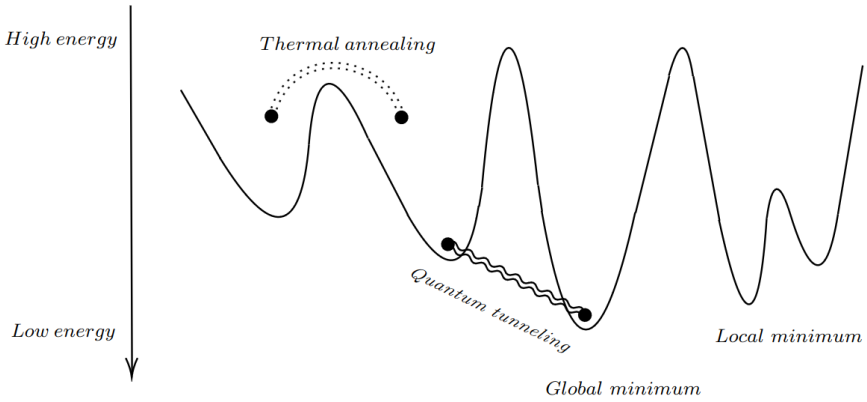

Figure 2.7: Quantum tunnelling used to find the global minimum in Quantum Annealing [74].

Quantum Annealing processors naturally return low-energy solutions [48]. They operate by manipulating a set of qubits with a time-dependent Hamiltonian $\hat{H}(t)$, which represents the total energy of a system in quantum mechanics. The initial Hamiltonian $\hat{H}_s$ harbours an easily attainable ground state, while the final Hamiltonian $\hat{H}_f$ encodes the cost function for the problem. Suppose the qubits start in the ground state and $\hat{H}(t)$ evolves slowly enough. In that case, the qubits remain in the ground state upon reaching the final Hamiltonian with probability close to 1, thereby circumventing local minima and discovering the solution [52, 62]. In the Equation 2.9, it is possible to see this adiabatic evolution represented by a function $s(t)$ that decreases from 1 to 0 overtime [35]. Figure 2.8 illustrates the evolution of a single qubit.

$$\hat{H}(t) = s(t)\hat{H}_s + (1 - s(t))\hat{H}_f \tag{2.9}$$

As the problem Hamiltonian is applied to the system, some energy levels may approach the ground state, as shown in Figure 2.9. The closer these levels get, the greater the probability that the system will transition from the lowest energy state to one of the excited states. The system may or may not return to the base state due to quantum tunnelling. During the annealing process, there is a point where the first excited state is at its closest to the ground state, known as the **minimum gap**, before diverging away [48].

Under ideal, adiabatic conditions, the system remains in the ground state without interference. However, achieving such conditions is virtually impossible in the real world. The primary factors contributing to this behaviour include:

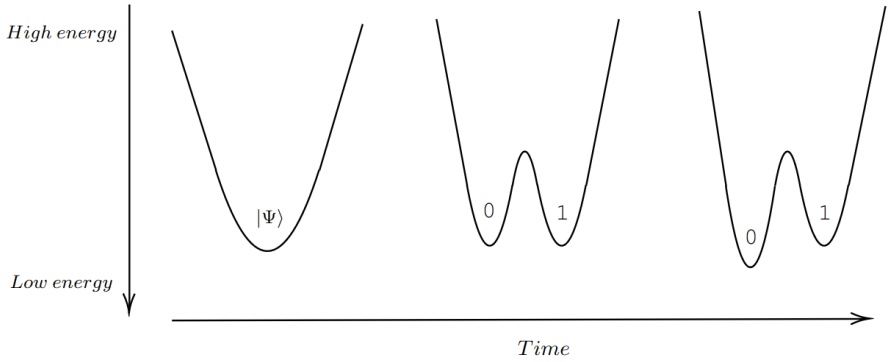

Figure 2.8: Quantum Annealing time evolution for one qubit (double-well potential) [74].

- The annealing process is not slow enough, leading to mixtures with nearby energy states [39].
- The thermal fluctuations present in any physical system [48].
- The quantum annealer's non-zero temperature precludes the description of a pure state [39].

As a result, analyzing results from Quantum Annealing requires statistical sampling to ensure confidence in the outcomes. Nevertheless, the solutions obtained are close to optimal and highly beneficial.

2.1.3.1 Hamiltonian as an Ising Model

The Ising model was initially formulated to characterize the physical interactions between spin pairs in a magnetic material. However, it has been applied to various systems involving interacting binary variables [39]. The final Hamiltonian encapsulates the problem and can be expressed in the Ising form.

$$\hat{H}_f = \sum_{i,j} J_{ij} \hat{\sigma}_z^i \hat{\sigma}_z^j + \sum_i h_i \hat{\sigma}_z^i \tag{2.10}$$

In Equation 2.10, h_i represents a bias on qubit i, $\hat{\sigma}_z$ is the Pauli-Z operator, and J_{ij} describes the coupling strength between the two qubits i and j. This formulation can be used to model a wide variety of combinatorial optimization problems, which is then used to find the ground state of the problem [39, 74]. For most non-convex Hamiltonians, determining the minimum energy state is an NP-hard problem that classical computers cannot solve efficiently [48].

2.1.3.2 D-Wave's Quantum Annealing

D-Wave Systems is a leading provider of Quantum Annealing technology, offering quantum processors designed to solve optimization problems. The company provides a range of quantum and hybrid computers accessible through the cloud.

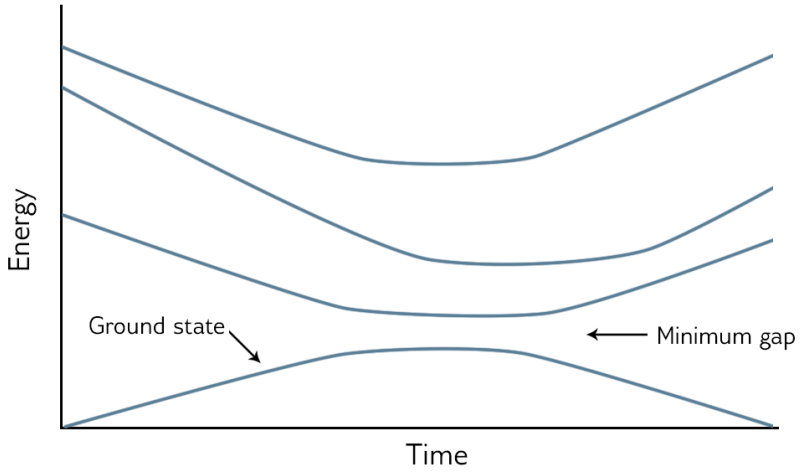

Figure 2.9: Energy spectrum showing the ground and excited states [48].

In D-Wave's systems, qubits are represented by superconducting loops forming the Quantum Processing Unit (QPU). These loops sustain a circulating current and an associated magnetic field, enabling them to maintain an equal superposition. External magnetic fields can be applied to the qubits to introduce biases, thereby favouring particular states (0 or 1) and creating a double-well potential akin to the one depicted in Figure 2.8 [48]. Details about their hardware can be found in Section 2.1.4.2.

As elucidated in Section 2.1.2.7, entanglement is another critical aspect to address. In Quantum Annealing, this is achieved via couplers, which makes two qubits tend to end up in the same or opposite states (0 or 1). Like biases, coupling strength can be adjusted programmatically [48].

The energy distribution among states is determined by the biases of qubits and the strength of coupling between them. At the end of the annealing process, each qubit settles into a classical state representing either the minimum energy configuration of the problem or one closely approximating it. Take the example in Figure 2.10, where two qubits yield four possible energy states.

Moreover, in the context of D-Wave's annealers, the initial Hamiltonian is represented by Equation 2.11, comprising a straightforward summation of Pauli-X operators [48].

$$\hat{H}_i = \sum_i \hat{\sigma}_x^i \tag{2.11}$$

2.1.3.3 Time Complexity

In computational complexity theory, it is common to use the **Big O**, $O(f(n))$, and **Big** Ω, $\Omega(f(n))$, notations to denote the upper and lower bounds of a function f, respectively, based on its asymptotic analysis for large inputs [12, 74]. The time complexity of classical problems generally falls into one of the major classes: P (Polynomial Time) or NP (Nondeterministic Polynomial Time). Problems in the NP class can be verified efficiently, but it remains an open question whether they

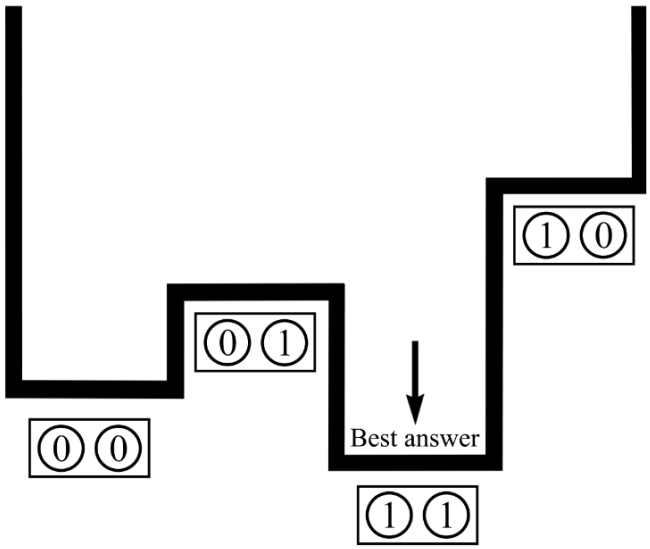

Figure 2.10: Diagram showing the energy of all possible states with two qubits [48].

can be solved efficiently on a classical computer. Among NP problems, the NP-complete problems are the hardest, meaning that if any NP-complete problem can be solved efficiently, then every problem in NP can be solved efficiently [88, 12].

Regarding adiabatic quantum computation, which QA is based on, its time complexity remains an open problem. The crucial aspect is to ensure that the Hamiltonian maintains a significant energy gap between its ground and excited states. This allows the Hamiltonian to vary more rapidly while preventing substantial deviations from the ground state, thus preserving high-quality solutions [52].

Typically, the minimum gap tends to decrease as the input n increases, and the rate of this decrease significantly impacts the computational time complexity [39, 42]. Although there are approximations regarding the scaling of the minimum gap with size, tackling the overall problem is computationally challenging [39, 52], or in some cases, unattainable [19].

One approach to analyzing the time complexity of Quantum Annealing (QA) involves comparing it to its classical counterpart, simulated annealing, which is expected to be inherently faster due to quantum tunnelling [71, 70]. The argument asserts that for a system to surpass an energy barrier (local optimum) of height Δv, simulated annealing (SA) has a probability of escaping the barrier on the order of $e^{-\frac{\Delta v}{T}}$, where T denotes the system temperature. Conversely, Quantum Annealing (QA) exhibits a probability on the order of $e^{-\frac{w\sqrt{\Delta v}}{\Gamma}}$, where w represents the width of the barrier and Γ signifies the strength of the quantum fluctuation. When generalized to n variables, the annealing time of SA can be approximated as e^n, whereas QA is approximated as $e^{\sqrt{n}}$ [71].

2.1.4 Quantum Hardware

Quantum hardware is currently in the Noisy Intermediate-Scale Quantum (NISQ) era and is expected to remain in this phase in the near future. While quantum computers with 50–100 qubits

may outperform today's classical digital computers in specific tasks, noise in quantum gates limits the reliable execution of larger quantum circuits. Although NISQ devices hold significant value for research purposes, a 100-qubit quantum computer will not immediately revolutionize the world [76].

In the short term, quantum computers will serve as specialized tools, with users accessing them predominantly through the cloud for research purposes or to attain some speedup. Presently, IBM provides gate-based quantum computers featuring 127 qubits, and their roadmap outlines projections of reaching up to 1000 qubits in the coming years, though their performance remains uncertain.

While achieving quantum computers with over 100 qubits marks a significant milestone in quantum technology, it is important to note that a quantum annealer with more than 5000 qubits, the D-Wave Advantage QPU, has been available since 2020, and an annealer with 2048 qubits, the D-Wave 2000Q, since 2017 [65]. However, it is essential to mention that no conclusive evidence demonstrates that quantum annealers can significantly speed up the time to solution compared to the most advanced classical hardware. The proven speedup only applies to the theoretical noiseless adiabatic computing, which is shown to be equivalent to gate-based quantum computation. For practical quantum annealers, the ability to scale with a growing number of qubits is still unknown [76]. Details about the Quantum Annealing theory can be found in Section 2.1.3.

For a fault-tolerant quantum computer, surpassing the limitations of NISQ, to be viable, it must satisfy the following five criteria. Although these criteria have yet to be fully met, they serve as guiding principles to enhance and refine quantum hardware progressively over time [56].

1. A scalable physical system containing well-defined qubits.
2. Capability for deterministic initialization of the system into a well-defined initial state.
3. Availability of a set of universal quantum gates, including single-qubit and entangling two-qubit gates.
4. Qubit decoherence times significantly exceeding gate times.
5. Ability to perform measurements on the qubit state with high accuracy.

2.1.4.1 IBM's Superconductors

Superconducting quantum computing is a type of quantum computing that uses superconducting circuits and superconducting qubits. Unlike Quantum Annealing systems, the concept here is to freely control the circuit by applying quantum gates within these circuits. These circuits must be kept at extremely low temperatures, on the order of 10mK, to display quantum properties [5].

In addition to this, Josephson junctions, nanoscale insulating layers positioned between two superconducting metals, are necessary. These junctions enable a more accurate simulation of an actual atom's behaviour, which is crucial for quantum computations. A gate capacitor is also incorporated, allowing the circuit to receive external signals, such as photons, for control purposes

[89, 5]. These artificial atoms, known as **transmons**, are a type of charge qubit and have their circuit depicted in Figure 2.11 [5, 97, 54].

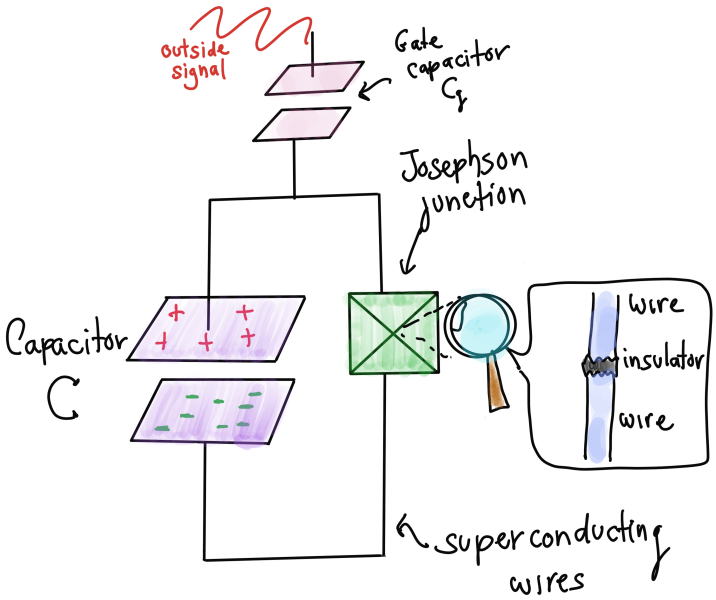

Figure 2.11: Transmon circuit with a Josephson junction and a gate capacitor [5].

Despite significant advancements in the field of superconducting qubits, the implementation of quantum error correction (QEC) remains a prerequisite for the realization of large-scale quantum computers. A majority of QEC strategies employ a form of redundancy, using multiple qubits and leveraging parity measurements. For example, Figure 2.12 illustrates a scenario where an auxiliary qubit $|A\rangle$ is employed to deduce the bit parity of the circuit. Similarly, Figure 2.13 presents a case for phase parity. Operators Z_1Z_2 and X_1X_2 can determine the occurrence and location of a bit or phase flip error without causing the collapse of the underlying quantum state [54].

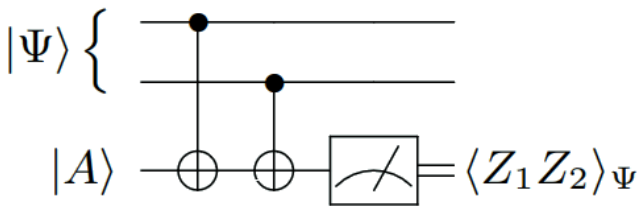

Figure 2.12: Bit parity error correction using an auxiliary qubit [54].

Among the companies using superconducting qubits for quantum computation, IBM has been a pioneer since 2016 [54] and leads the field regarding scale and the number of qubits in their machines. IBM currently offers 127-qubit machines to the public [5], the **IBM Eagle**. They have also released a development roadmap, depicted in Figure 2.14, highlighting their past achievements and plans for hardware advancements [36].

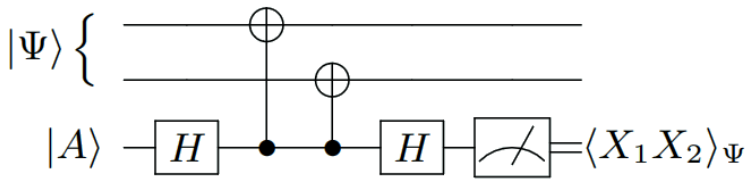

Figure 2.13: Phase parity error correction using an auxiliary qubit [54].

The roadmap includes **IBM Heron**, the next-generation processor presented in December 2023 at the IBM Quantum Summit. This QPU has 133 qubits and improved error rates, reportedly achieving a fivefold improvement over IBM Eagle's execution records [66], and is currently available for premium users. Another processor anticipated in the coming years is **IBM Flamingo**, which will provide 156 qubits and up to 15 thousand gates (up from 5 thousand in Heron). After 2029, the plan is less certain, but IBM aims to reach 200 qubits by 2029 and 2000 after 2033, with functional error-corrected modularity [77].

IBM's quantum computers are accessible via the cloud through their Quantum Platform. The hardware can be controlled, and circuits can be executed using the Python framework Qiskit, along with tools developed by the Qiskit Community.

2.1.4.2 D-Wave's Quantum Annealers

As previously explained in Section 2.1.3.2, D-Wave Systems is the current leading provider of Quantum Annealing technology. The company has released multiple iterations of hardware over the years, each with an increasing number of qubits and improved result quality. Due to the promising results for the project and the availability of extensive educational material, this dissertation focuses on their technology to explain the hardware of quantum annealers. The main architectures discussed in this section are Chimera, which underpins the D-Wave 2000Q; Pegasus, used in the D-Wave Advantage; and Zephyr, the next-generation QPU currently under development and expected to reach more than 7000 qubits [10].

For a clear understanding of the concept, consider the task of mapping a triangle graph onto a hexagon graph, as illustrated in Figure 2.15. The triangle graph forms a 3-loop using all its vertices, whereas the hexagon graph forms a 6-loop. To embed the 3-loop within the 6-loop, we can pair the vertices of the hexagon, such as (q_0, q_1), (q_2, q_3), and (q_4, q_5). This method, known as **embedding**, allows us to map graphs with differing topologies [74].

Similarly, depicting the optimization problem's graph must be integrated into the QPU graph topology. The QPU consists of a lattice of interconnected qubits linked to others via couplers, though it is not fully connected. In the Chimera topology, a unit cell comprises two sets (one vertical and one horizontal) of four qubits. Each qubit is connected to all qubits in the other set but none to its own, as depicted in Figure 2.16. Depending on the architecture, the lattice is formed by assembling these unit cells [48, 65].

The qubits within the QPU are interconnected using different types of couplers [48]:

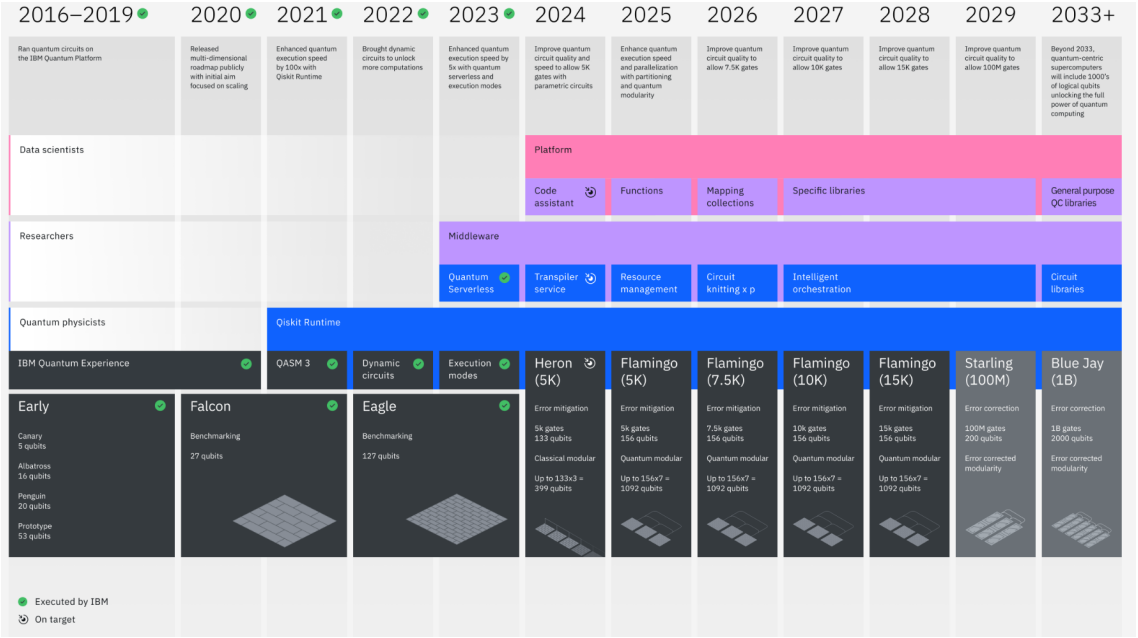

Figure 2.14: IBM Quantum Development Roadmap in 2024 [77].

- **Internal couplers**: Connect pairs of qubits with opposing directions, as illustrated in Figure 2.16.
- **External couplers**: Connect pairs of adjacent qubits within the same row or column.
- **Odd couplers**: Connect pairs of qubits aligned in a similar manner. These are only present in the Pegasus and Zephyr topologies.

Each unit cell contained eight qubits in the older Chimaera topology, as illustrated in Figure 2.16. Each qubit had six couplers (internal and external). These units were part of a simple lattice, depicted in the cut in Figure 2.17 [48]. The Pegasus topology, on the other hand, features a more complex structure rather than a simple lattice of unit cells. In this case, shown in Figure 2.18, each qubit has 15 couplers (including odd couplers) and forms a larger lattice [9, 65]. The Zephyr topology aims to combine multiple chimera graphs while incorporating odd couplers, with each qubit having 20 couplers, as demonstrated in Figure 2.19 [10]. It is important to note that, due to a small percentage of defective qubits or couplers, the **working graph** is a subgraph of the initial graph [99].

The evolution of the architectures is driven by their scalability with the number of qubits, achieved by enabling larger lattices, unit cells, more couplers between them, and better error correction. Additionally, it is influenced by the ability to embed problem graphs compactly, for instance, by facilitating shorter (and thus more versatile) chains of nodes [65, 10, 9].

Material-wise, these QPUs use superconducting materials and are known as flux qubits. Flux qubits create a quantized magnetic flux in a superposition of states using Josephson junctions, which are nanoscale insulating layers within the superconducting loop [97]. This setup permits

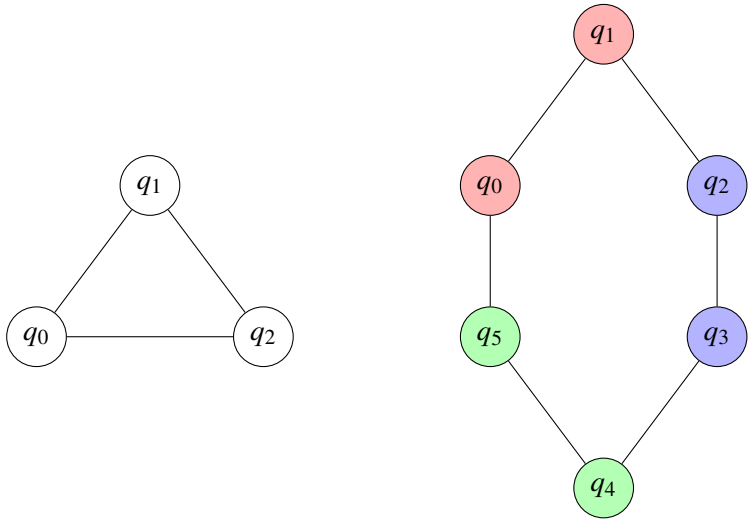

Figure 2.15: Illustration of embedding using graphs with triangular and hexagonal topologies [74].

the manipulation of potential energy landscapes via external magnetic fields, allowing qubits to interact through magnetic interference [39]. An illustration of this can be seen in Figure 2.20.

D-Wave QPUs are accessible via the cloud through their Leap platform. The hardware can be controlled, and problems can be submitted using the Python framework Ocean [47].

2.1.5 Quantum Algorithms

This section delves into prominent quantum algorithms capable of navigating the complexities of transportation system optimization. Grounded in the principles of quantum computing, these algorithms provide solutions for addressing optimization tasks, such as vehicle routing problems.

2.1.5.1 Quadratic Unconstrained Binary Optimization

The Quadratic Unconstrained Binary Optimization (QUBO) model is a mathematical formulation for solving quadratic combinatorial optimization problems. It involves finding the optimal configuration of binary variables to minimize or maximize a quadratic objective function. In simpler terms, it is a way to represent and solve optimization problems where the goal is to make binary choices that lead to the best overall outcome based on a quadratic function [69].

The QUBO model has successfully tackled major NP-hard challenges, including multiple knapsack problems, clique problems, graph colouring, and warehouse location problems. This success led to the creation of well-known optimization tools for classical optimizers, such as IBM CPLEX. QUBO remains highly relevant due to its applications in quantum computing algorithms like Quantum Annealing and QAOA, as well as in neuromorphic computation [37].

A **formal formulation** of the QUBO model is presented as the optimization problem:

$$\text{minimize/maximize } y = x^T Q x, \quad x \in \{0,1\}^n \tag{2.12}$$

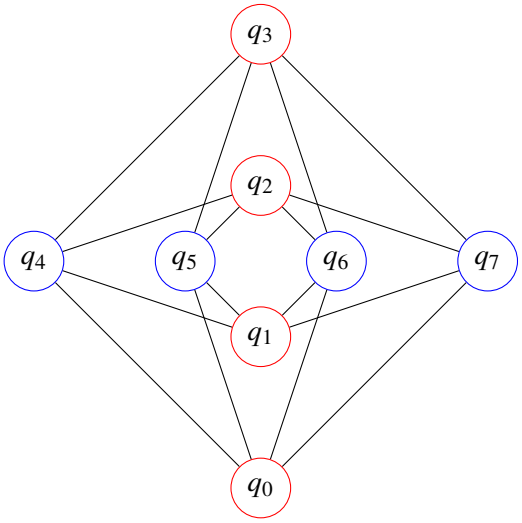

Figure 2.16: Illustration of D-Wave Chimera unit cell graph [48, 74].

where x is a vector of binary decision variables, Q is a square matrix of constants representing a cost function, and x^T is the transpose of the vector x. The Q matrix is commonly assumed to be symmetric or in upper triangular form, depending on the framework implementing the model [37, 13].

The pure form of the QUBO model lacks constraints, which poses a challenge when dealing with problems involving real-world situations. Many practical scenarios need additional conditions to be met during the optimization process. To address this, **quadratic penalties** are introduced into the objective function as an alternative to explicitly enforcing classical constraints. These penalties are structured to be zero for feasible solutions and assume a significant value for infeasible solutions. Examples of such penalties for commonly encountered constraints are outlined in Table 2.2 [37].

Classical Constraint	**QUBO Penalty**
$x+y \leq 1$	$P(xy)$
$x+y \geq 1$	$P(1-x-y+xy)$
$x+y = 1$	$P(1-x-y+2xy)$
$x \leq y$	$P(x-xy)$
$x_1+x_2+x_3 \leq 1$	$P(x_1x_2+x_1x_3+x_2x_3)$
$x = y$	$P(x+y-2xy)$

Table 2.2: Table of a few known constrain/penalty pairs [37].

To illustrate the idea, consider a problem of the form $Min\ y = f(x)$ subject to the constraint $x_1+x_2 \leq 1$, where x_1 and x_2 are binary variables. As long as $f(x)$ is linear or quadratic, then this will be the QUBO form of the problem: $minimize\ y = f(x)+P(x_1x_2)$.

A significant attribute of the QUBO model is its equivalence to the Ising model, explained in Section 2.1.3.1. This equivalence allows the QUBO model to be solved using algorithms such as Quantum Annealing and Quantum Approximate Optimization Algorithm (QAOA), detailed in

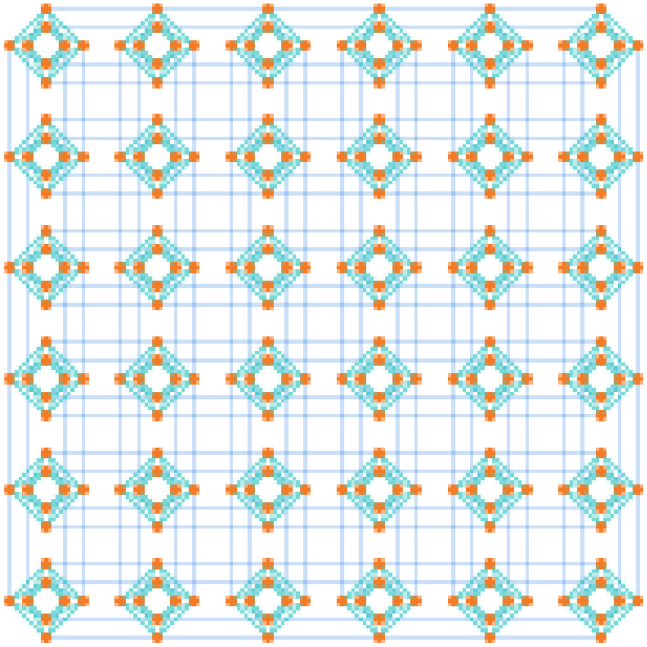

Figure 2.17: Chimera lattice cut with 36 unit cells. The internal couplers are shown in cyan, while the external couplers are depicted in blue [65].

sections 2.1.3 and 2.1.5.2, respectively. The transformation $x_i = \frac{1+s_i}{2}$ facilitates the conversion between the two models, where x_i represents a QUBO variable and s_i denotes a qubit state in the Ising model [41, 13].

When deciding between the QUBO model and the Ising model, the QUBO model is often preferred due to its ease of problem definition, especially when including constraints. For example, if we need to apply a one-hot constraint to a group of binary variables $x_i \in x$, we can express it in the QUBO framework as shown in Equation 2.13. Then, the right-hand side of the equation is incorporated into the objective function with a designated penalty to ensure that the constraint is satisfied when minimized [13], as shown in Equation 2.14. This process is much more practical than manually encoding the Hamiltonian.

$$\sum_i x_i = 1 \Leftrightarrow min(1 - \sum_i x_i)^2 = 0 \tag{2.13}$$

$$\text{minimize } y = x^T Q x + \lambda (1 - \sum_i x_i)^2, \quad \lambda \geq 1 \tag{2.14}$$

It should be noted that although QUBO accepts integer or continuous variables in its model, they are not natively supported by quantum computers. For this reason, they need to be converted to binary variables before the algorithm is executed. Depending on the variable bounds, this will naturally increase the problem size [53], so these variables should be carefully analyzed when formulating a model. Similar considerations apply to the use of inequality constraints. To integrate them into the objective function, integer auxiliary variables, known as **slack variables**, must be

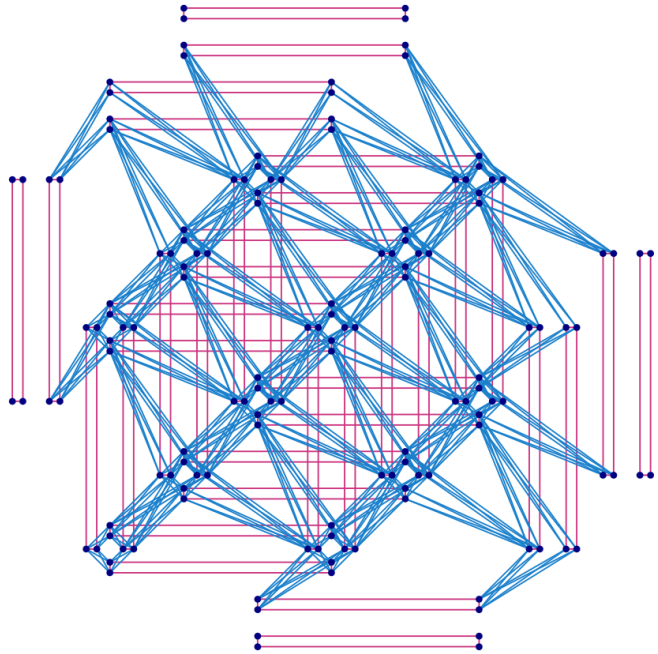

Figure 2.18: Pegasus lattice cut. The internal couplers are represented in blue, the external couplers are illustrated as long red lines, and the odd couplers are short red lines [9].

introduced into the model. For example, this transforms the inequality $Ax \leq b$ into an equality $Ax + Ds = b$, where s represents the slack variable and D is a coefficient ensuring the validity of the conversion [92].

2.1.5.2 Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid optimization algorithm that integrates classical and quantum computing components, as illustrated in Figure 2.21. As an approximation algorithm, it is employed to achieve near-optimal solutions to complex combinatorial problems [38], such as vehicle routing problems. The QAOA framework efficiently employs QUBO problems, with each binary variable mapped to a single qubit [7].

The primary advantage of QAOA is its low depth, meaning the number of time steps (or operations) from initialization to final measurement is relatively small. Consequently, it is less affected by decoherence, making it quite robust to errors. Regarding width (the number of qubits in the circuit), QAOA requires only enough qubits to represent the problem. This is ideal, although solving large problems remains unfeasible in the NISQ era. This presents a significant advantage in the near term compared to more traditional quantum algorithms, such as Shor's or Grover's algorithms [87].

QAOA employs a quantum circuit consisting of two interleaved modules, whose circuits are illustrated in Figure 2.22 [7]:

- A problem or cost layer, which applies a phase shift to solutions proportional to their quality.

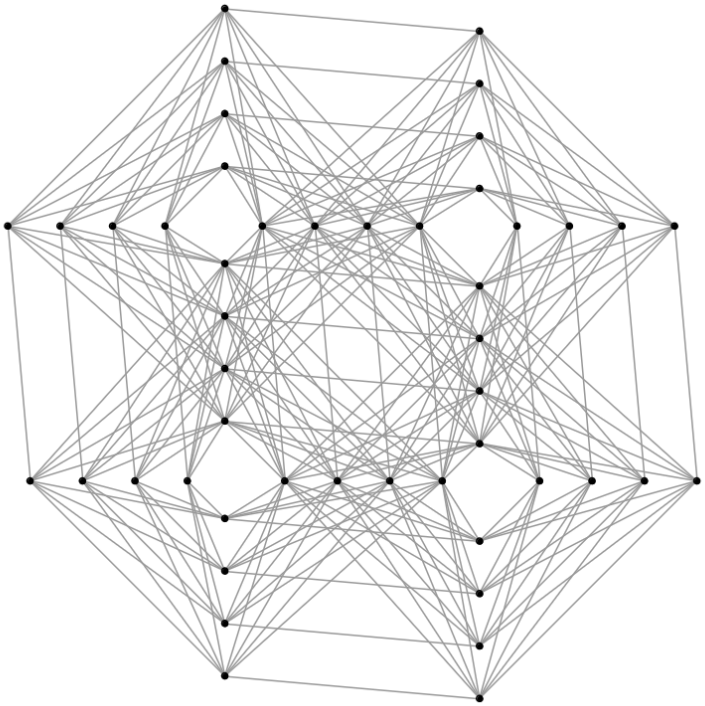

Figure 2.19: Zephyr lattice cut. The internal couplers are depicted as gray circles, while the odd and external couplers are represented by lines [10].

- A mixer layer, which mixes quantum amplitudes between solutions to explore the search space.

A classical loop iteratively adjusts the parameterized modules, ensuring that the combined effects of exploration and quality-dependent phase shifts enhance the likelihood of measuring high-quality outputs. The quantum circuit undergoes multiple runs to generate a set of measured states (candidate solutions). The classical optimizer refines the variational quantum circuit parameters until convergence or a predefined stopping condition is met. Ultimately, the algorithm returns the best-obtained candidate solution [7, 34]. The initial parameters for these modules are called **ground state** or **ansatz** [86].

QAOA as an adiabatic schedule

Similar to the explanation provided for Quantum Annealing in Section 2.1.3, particularly in Equation 2.9, QAOA can also be associated with adiabatic computation. In this context, the equation becomes 2.15, where $s(t)$ is the same function transitioning from 1 to 0 over time, H_M is the mixer Hamiltonian, H_C is the problem Hamiltonian, and X is a mixer parameter [96, 34].

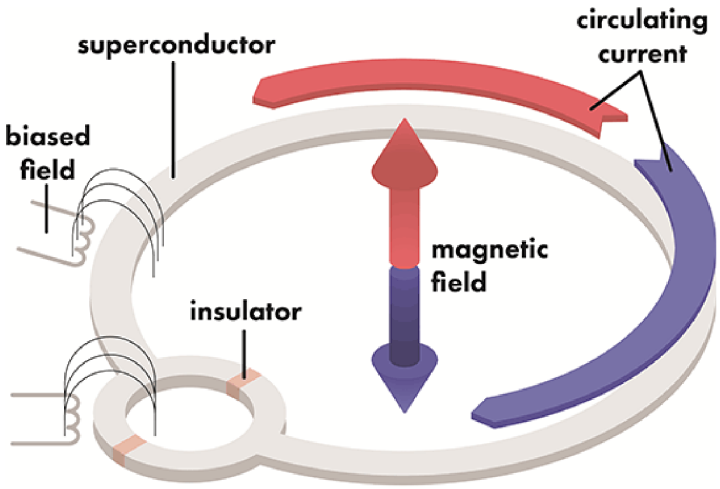

Figure 2.20: The design of a flux qubit using superconducting materials with magnetic fields [39].

$$\hat{H}(t) = s(t)\hat{H}_M + (1 - s(t))\hat{H}_C, \qquad \hat{H}_M = \sum_i X_i \tag{2.15}$$

The QAOA algorithm essentially approximates a quantum adiabatic process. The mixer and problem layers are segmented over time through a process known as **trotterization**, where they alternate multiple times, and the sum of the applied angles represents the total run time. For a good approximation, it is preferable for these angles to be small and for the run time to be long, which can be generalized by a parameter p. The larger the value of p, the closer the solution will be to the optimum. Theoretically, an infinite p would always yield the best result, similar to an adiabatic algorithm (see Equation 2.16, where M represents the quality of a solution and C_{max} is the optimum) [34, 96].

$$M_{p+1} \geq M_p \qquad \lim_{p\to\infty} M_p = C_{max} \tag{2.16}$$

Warm Starting QAOA

The concept of a warm start involves initializing the QAOA with a solution derived from a classical optimizer, replacing the standard initial state. Consequently, the mixer Hamiltonian is adjusted so that this new state serves as its corresponding ansatz [96]. This approach typically yields better results than standard QAOA, particularly on noisy quantum computers and at lower depths. However, it requires running an additional optimizer at the beginning [33].

In Figure 2.23, a simulation demonstrates that warm starting achieves better solutions than regular QAOA at lower depth values p. Additionally, the importance of replacing the mixer is evident, as failure to do so decreases the probability of measuring the optimal result from the

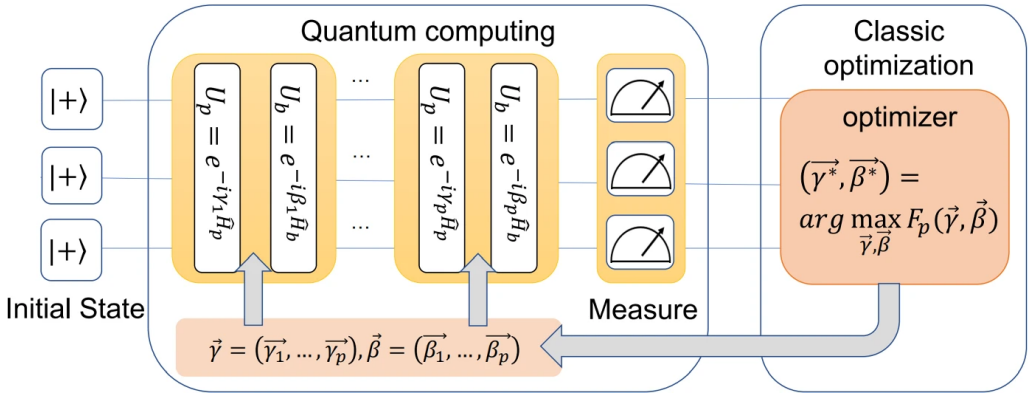

Figure 2.21: Diagram showing the principles of QAOA [38].

initial state and prevents QAOA from converging, even for large p. The simulation was conducted ten times with six qubits.

This technique involves applying a continuous-valued relaxation to the original problem, simplifying it before determining the initial state for QAOA. The optimal solution of this relaxation, which is easier to obtain, is used as the initial state, providing a good approximation for the initial problem. Optionally, the optimal solution of the relaxation can be rounded to produce a feasible discrete solution [33].

2.2 Classical Methods and Techniques

This section explores the current literature on vehicle routing problems, emphasizing implementations utilizing classical computation. The Vehicle Routing Problem (VRP) and its various variations represent longstanding challenges within the NP group, leading to an extensive exploration of solutions and proposals.

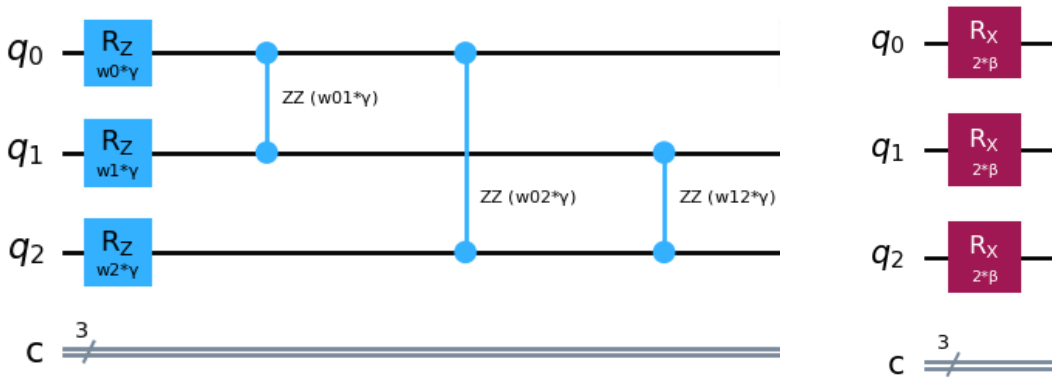

Figure 2.22: Circuits for the cost (left) and mixer (right) layers, with parameterized rotation gates (R_Z, R_{ZZ}, and R_X) [96]

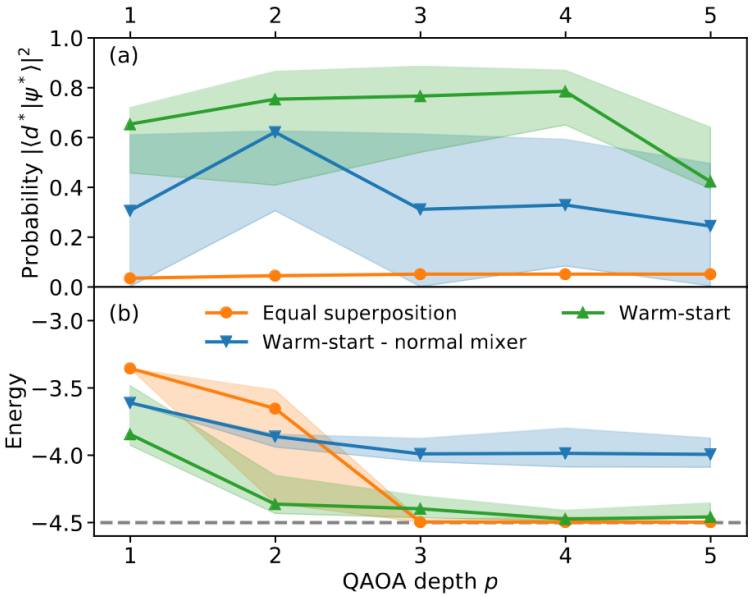

Figure 2.23: (a) Probability to sample the optimal result from the initial state and (b) Energy of the system for warm-start and standard QAOA at different depths p [33].

2.2.1 Survey on Classical Algorithms

In 2022, Praveen et al. [75] surveyed research papers presenting algorithms aimed at solving the CVRP. Most of this subsection will be grounded in the key solutions highlighted by their comprehensive survey.

The Clarke and Wright algorithm (1964) is one of the most renowned Vehicle Routing Problem (VRP) heuristics. It is notable for its ability to rapidly generate a near-optimal solution rather than striving to find the absolute best one [17]. However, numerous algorithms have been developed since its inception that offer improvements, particularly in handling constraints like limited capacity.

In 2017, Ali Asghar Rahmani Hosseinabadi et al. [44] developed the CVRP_GELS algorithm, which is based on principles from physics concerning searching and random search. This method performs well for larger problem sizes and fewer vehicles, achieving shorter computation times than the previous state-of-the-art, except in a few instances.

In 2017, M.A. Hannan et al. [40] devised a route optimization solution for waste pickup using a modified Particle Swarm Optimization (PSO) version. While the results were positive, the approach's feasibility for real-world scenarios remains to be validated, suggesting the potential for future work to involve the development of smart bins.

In 2017, Kangzhou Wanga et al. [94] employed integer programming and Variable Neighborhood Search (VNS) algorithms to address the Two-Echelon Capacitated Vehicle Routing Problem (2E-CVRP) variant, which uses satellites as secondary depots. Their approach led to a significant reduction in both the number of required drivers and fuel costs.

In 2018, Mario Marinelli et al. [64] successfully employed a Fuzzy C-means approach to cluster end users to streamline civic carriage distribution in urban logistics. This innovative approach contributes to developing green routing strategies, primarily enhancing traffic flow and reducing discharge costs.

In 2018, Valeria Leggieri and Haouari [57] introduced metaheuristics for the Asymmetric Capacitated Vehicle Routing Problem (ACVRP), which involves different travel costs between locations. Their approach comprises three stages: reducing the problem size by eliminating unwanted arcs, deriving a reasonable solution, and finally generating the optimal solution. The study suggests that improved formulations can alleviate the computational burden and emphasizes the need for future research to develop faster methods for obtaining initial solutions.

In 2014, Wassan and Nagy [95] presented an Integer Linear Programming (ILP) formulation for the Vehicle Routing Problem with Pickup and Delivery (VRPPD), demonstrating its adaptability for addressing analogous problems and facilitating future extensions of this model.

2.2.2 QUBO-Compatible Formulations

The literature sources most relevant to this study are the formulations that easily translate into a Quadratic Unconstrained Binary Optimization (QUBO) model, elaborated in Section 2.1.5.1. Fortunately, much of the research concerning the Capacitated Vehicle Routing Problem (CVRP) relies on linear formulations, which seamlessly align with QUBO. While these formulations may need to be finely tuned for quantum computation, particularly with regard to handling a low number of qubits, they offer a solid foundation. For this reason, this subsection centres on research papers featuring such formulations.

2.2.2.1 Two-Index Vehicle Flow Formulation

In the two-index vehicle flow formulation, Munari et al. [72] define binary decision variables x_{ij} that equal 1 if a route directly connects node i to node j, for $i, j \in \mathbb{N}$. Additionally, y_i is an integer variable representing the cumulated demand on a route up to node $j \in \mathbb{N}$. With these parameters, the objective function is specified by Equation 2.17, where c_{ij} is the cost of travelling from node i to node j, K is the number of available vehicles, Q is the maximum vehicle capacity, and q_i is the demand at node i. Similar formulations have been proposed by Çağrı Koç et al. [100] and Rieck and Zimmermann [81].

$$\min \quad \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} c_{ij} x_{ij} \tag{2.17}$$

Subject to:

$$\sum_{j=1}^{n+1} x_{ij} = 1, \quad i = 1, \ldots, n \tag{2.18}$$

$$\sum_{i=0}^{n} x_{ih} = \sum_{j=1}^{n+1} x_{hj}, \quad h = 1, \ldots, n \tag{2.19}$$

$$\sum_{i=1}^{n} x_{0i} \leq K \tag{2.20}$$

$$y_j \geq y_i + q_j x_{ij} - Q(1 - x_{ij}), \quad i, j = 0, \ldots, n+1 \tag{2.21}$$

$$q_i \leq y_i \leq Q, \quad i = 0, \ldots, n+1 \tag{2.22}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 0, \ldots, n+1 \tag{2.23}$$

Constraints 2.18 and 2.19 ensure that each node is visited exactly once. Constraint 2.20 limits the number of routes (vehicles) to at most K [72]. Constraints 2.21 and 2.22 implement the Miller-Tucker-Zemlin (MTZ) formulation, which uses integer variables to prevent subtours (routes not connected to the depot) and ensure vehicle capacity is not exceeded [2, 68]. The final constraint, 2.23, bounds the binary variables.

2.2.2.2 Two-Index Commodity Flow Formulation

The two-index commodity flow formulation, proposed by Çağrı Koç et al. [100] and Dell'Amico et al. [24], is very similar to the vehicle flow formulation in 2.2.2.1, with the difference of not using MTZ subtour elimination. Instead of controlling the flow route-wise, this model analyses the commodity picked up and delivered at each node, hence the name.

In this case, x_{ij} are the same binary decision variables with value 1 if a route connects node i to node j, for $i, j \in \mathbb{N}$. Other than that, y_{ij} and z_{ij} are integer decision variables representing the amount of commodity picked up and delivered on a given edge (i, j), respectively. In terms of constants, c_{ij} is the cost of travelling from node i to node j, K is the number of available vehicles, Q is the maximum vehicle capacity, and p_i and d_i are the pickup and delivery demands at node i. The objective function can be found in Equation 2.24.

$$\min \quad \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} c_{ij} x_{ij} \tag{2.24}$$

Subject to:

$$\sum_{j=1}^{n+1} x_{ij} = 1, \quad i = 1, \ldots, n \tag{2.25}$$

$$\sum_{i=0}^{n} x_{ih} = \sum_{j=1}^{n+1} x_{hj}, \quad h = 1, \ldots, n \tag{2.26}$$

$$\sum_{i=1}^{n} x_{0i} \leq K \tag{2.27}$$

$$\sum_{i=0}^{n+1} y_{hi} - \sum_{j=0}^{n+1} y_{jh} = p_h, \quad h = 1,\ldots,n+1 \tag{2.28}$$

$$\sum_{i=0}^{n+1} z_{ih} - \sum_{j=0}^{n+1} z_{hj} = d_h, \quad h = 1,\ldots,n+1 \tag{2.29}$$

$$y_{ij} + z_{ij} \leq Q x_{ij}, \quad i,j = 0,\ldots,n+1 \tag{2.30}$$

$$y_{ij}, z_{ij} \geq 0, \quad i,j = 0,\ldots,n+1 \tag{2.31}$$

$$x_{ij} \in \{0,1\}, \quad i,j = 0,\ldots,n+1 \tag{2.32}$$

Constraints 2.25 - 2.27 and 2.32 are the same as in 2.2.2.1. Constraints 2.28 and 2.29 control the route flows at each node. Specifically, these constraints ensure that the amount of commodity picked up and delivered is balanced according to the demands at each node. Constraint 2.30 ensures the vehicle capacity is not exceeded, while 2.31 imposes limits on the auxiliary variables.

2.2.2.3 Three-Index Commodity Flow Formulation

The three-index commodity flow formulation, proposed by Çağrı Koç et al. [100] and Montané and Galvão [3], is very similar to the formulation in 2.2.2.2 but incorporates an additional vehicle index. In this model, the binary variables x_{ijk} are equal to 1 if the edge $(i,j) \in \mathbb{N}$ is traversed by vehicle k (Equation 2.33). Desaulniers et al. [25] also presented a formulation with three indices similar to that one, although using different constraints for flow control.

$$\min \quad \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} \sum_{k=0}^{K-1} c_{ij} x_{ijk} \tag{2.33}$$

Subject to:

$$\sum_{j=1}^{n+1} \sum_{k=0}^{K-1} x_{ijk} = 1, \quad i = 1,\ldots,n \tag{2.34}$$

$$\sum_{i=0}^{n} x_{ihk} = \sum_{j=1}^{n+1} x_{hjk}, \quad h = 1,\ldots,n, \quad k = 0,\ldots,K-1 \tag{2.35}$$

$$\sum_{i=1}^{n} x_{0ik} \leq 1, \quad k = 0,\ldots,K-1 \tag{2.36}$$

$$\sum_{i=0}^{n+1} y_{hi} - \sum_{j=0}^{n+1} y_{jh} = p_h, \quad h = 1,\ldots,n+1 \tag{2.37}$$

$$\sum_{i=0}^{n+1} z_{ih} - \sum_{j=0}^{n+1} z_{hj} = d_h, \quad h = 1,\ldots,n+1 \tag{2.38}$$

$$y_{ij} + z_{ij} \leq Q \sum_{k=0}^{K-1} x_{ijk}, \quad i,j = 0,\ldots,n+1 \tag{2.39}$$

$$y_{ij}, z_{ij} \geq 0, \quad i,j = 0,\ldots,n+1 \tag{2.40}$$

$$x_{ij} \in \{0,1\}, \quad i,j = 0,\ldots,n+1 \tag{2.41}$$

The constraints (2.34-2.41) serve the same purpose as those in 2.2.2.2, but they have been adapted to account for the additional vehicle index. In this case, the constraint 2.27 is removed.

2.2.2.4 Set Partitioning Formulation

Munari et al. [72] proposed another formulation, set partitioning, which takes a set $\mathbb{R}$ of routes satisfying the problem constraints as input. For instance, in the CVRP, a route must begin and end at the depot, visit each node at most once, stay within the vehicle's capacity, and fulfil the node's demand.

Let λ_r denote the binary decision variable, equal to 1 if route $r \in \mathbb{R}$ is chosen, thus formulating Equation 2.42. Also, consider the constant column $a_r = (a_{r0},\ldots,a_{rn})$, a binary vector where a_{ri} is 1 if route r visits node i.

$$\min \quad \sum_{r\in\mathbb{R}} c_r \lambda_r \tag{2.42}$$

Subject to:

$$\sum_{r\in\mathbb{R}} a_{ri}\lambda_r = 1, \quad i = 1,\ldots,n \tag{2.43}$$

$$\sum_{r\in\mathbb{R}} \lambda_r \leq K \tag{2.44}$$

$$\lambda_r \in \{0,1\}, \quad r \in \mathbb{R} \tag{2.45}$$

Constraint 2.43 ensures that each node is visited exactly once, while constraint 2.44 sets the maximum number of available vehicles. Lastly, constraint 2.45 places bounds on the binary variables λ_r.

The main challenge of this approach lies in computing the set of routes $\mathbb{R}$, as its size grows exponentially with the number of nodes. Hence, set partitioning uses a column generation technique that addresses a linear relaxation of the original model with a reduced subset $\overline{\mathbb{R}} \subset \mathbb{R}$. This iterative method tackles a subproblem, the Resource Constrained Elementary Shortest Path Problem (RCESPP), depicted in Equation 2.46.

$$\min_{r \in \mathbb{R}} rc(\overline{u}, \overline{\sigma}) = \sum_{j=0}^{n+1} \sum_{i=0}^{n+1} (c_{ij} - \overline{u}_i) x_{rij} - \overline{\sigma} \tag{2.46}$$

Here, $\overline{u}$ and $\overline{\sigma}$ represent variables associated with constraints 2.43 and 2.44, respectively, while x_{rij} are binary variables whose values are set to 1 if route r visits node i and then proceeds directly to node j. Although integer programming formulations exist for RCESPP, they are not efficiently solved by current solvers. Hence, current solutions resort to approximate methods [72].

2.2.3 Google OR-Tools

OR-Tools is an open-source software tool for combinatorial optimization tasks to solve large instances of problems such as constraint programming, linear programming, and vehicle routing. OR-Tools won gold in the International Constraint Programming Competition every year since 2013 [61], establishing it as a very capable tool that implements state-of-the-art techniques for combinatorial problems, including the VRP.

Besides allowing developers to model their problems in various programming languages, OR-Tools offers access to both commercial solvers, such as Gurobi or CPLEX, and open-source solvers, including CP-SAT or GLPK [61].

Within OR-Tools, the vehicle routing library lets developers model and solve generic vehicle routing problems, ranging from the simplest Traveling Salesman Problem (TSP) to more complex variations, such as the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW). This library is built on top of the more generic constraint programming library, giving it a robust foundation [20, 61].

2.3 Quantum Methods and Techniques

This section explores the current literature on vehicle routing problems, specifically emphasizing quantum computation implementations. It introduces various quantum approaches, along with the outcomes already achieved or anticipated in the near future, considering the constraints of the currently limited quantum hardware.

2.3.1 QAOA Implementations

The first part of this section reviews and explains previous work on developing implementations of the Quantum Approximate Optimization Algorithm (QAOA), as described in Section 2.1.5.2. These studies focus specifically on applying QAOA to solve vehicle routing problems.

2.3.1.1 QAOA Framework for CVRP

Bentley et al. [7] developed a framework to achieve potential performance enhancement through quantum algorithms tailored for transport optimization, explicitly addressing the Capacitated Vehicle Routing Problem (CVRP).

In their approach, the Quantum Approximate Optimization Algorithm (QAOA) model is employed to generate near-optimum approximate solutions for the CVRP. The authors introduced an innovative binary method for encoding the CVRP for quantum algorithms, thereby reducing the qubit resource requirements. This method, initially applied to the problem of clique partitioning, was extended from a QUBO representation of the Traveling Salesperson Problem (TSP) by incorporating a vehicle index into the binary decision variables and an additional constraint related to the capacity of each vehicle into each decision variable.

Their formulation adopted a more compact permutation encoding by using integer variables that are binary encoded, resulting in the use of $n \log n$ qubits instead of the usual n^2. Each node is assigned a visit-cycle position i. This binary encoding offers the advantage of reducing qubit complexity, albeit at the cost of introducing a problem Hamiltonian with multi-qubit operations. These operations must be decomposed into one and two-qubit operations in the final circuit, requiring careful transpilation to avoid significant overhead.

Apart from the high-level algorithm, the authors introduced two advancements to improve the circuit's success on small-scale quantum computers. Firstly, they incorporated a hard mixer designed to limit the algorithm's exploration to the valid solution space, unlike the conventional QUBO approach, where penalties are added to discourage invalid solutions. Secondly, they utilized Q-CTRL's Fire Opal software for efficient hardware-aware custom compilation. To exemplify, they applied a circuit for routing a vehicle between a depot and two passenger nodes, which involved four two-qubit gates.

The authors showcased over 20X error reduction in their example compared to the standard QAOA implementation despite employing a smaller but non-scalable circuit. The result is plotted in Figure 2.24. According to their solution, addressing a problem involving seven vehicles with a maximum capacity of 20 passengers needs approximately 1000 qubits, a demand exceeding the capabilities of current hardware (with 127-qubit devices currently available). However, this capacity aligns with IBM's quantum systems roadmap, indicating a potential feasibility in the near future.

2.3.1.2 VRP QUBO Formulation

Utkarsh et al. [4] proposed a QUBO formulation for the fundamental Vehicle Routing Problem (VRP) and described a method to transform it into the Ising form and solve it using Qiskit simulations.

Let x_{ij} be the binary decision variable, where $x_{ij} = 1$ indicates the presence of an edge between nodes i and j, $i, j \in \mathbb{N}$, with a corresponding cost c_{ij}. The formulation is given by the cost function 2.47 and needs $n(n-1)$ qubits to encode it.

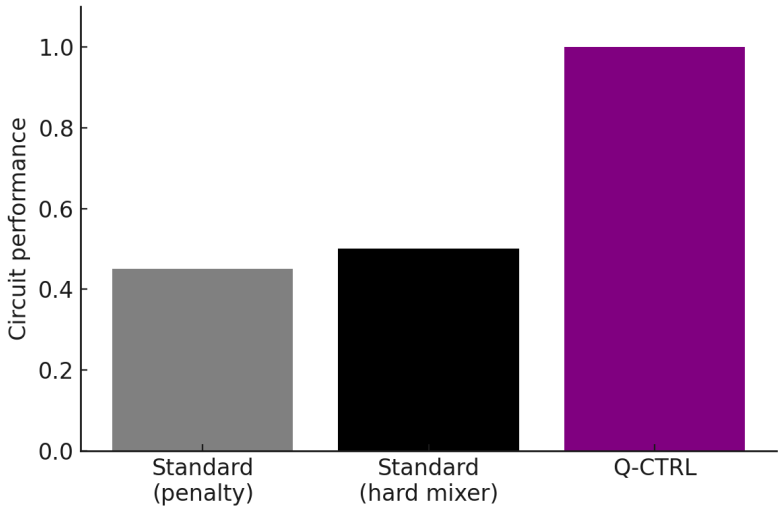

Figure 2.24: Comparison of circuit performance (probability of achieving an error-free routing result) between standard implementation, hard mixer approach, and custom circuit compilation. The exact values are 43%, 46% and 97% [7].

$$\min \quad \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} c_{ij} x_{ij} \tag{2.47}$$

Subject to:

$$\sum_{j=1}^{n+1} x_{ij} = 1, \quad i = 1, \ldots, n \tag{2.48}$$

$$\sum_{j=0}^{n} x_{ji} = 1, \quad i = 1, \ldots, n+1 \tag{2.49}$$

$$\sum_{i=1}^{n} x_{0i} = K \tag{2.50}$$

$$\sum_{i=1}^{n} x_{i0} = K \tag{2.51}$$

$$x_{ij} \in \{0,1\}, \quad i,j = 0, \ldots, n+1 \tag{2.52}$$

Constraints 2.48 and 2.49 guarantee that each node is visited exactly once. Constraints 2.50 and 2.51 ensure that all vehicles start and finish at the depot.

In their article, they show three examples with up to 20 qubits. For the example with five nodes and three vehicles, the probability distribution of the solutions can be seen in Figure 2.25 and the best solution in Figure 2.26. The study concludes by saying that, although some reasonable solutions were obtained, the optimal solution has a low chance of being measured. They give a

few justifications, such as the *p-step* value not being optimized or the hardware noise still being too high.

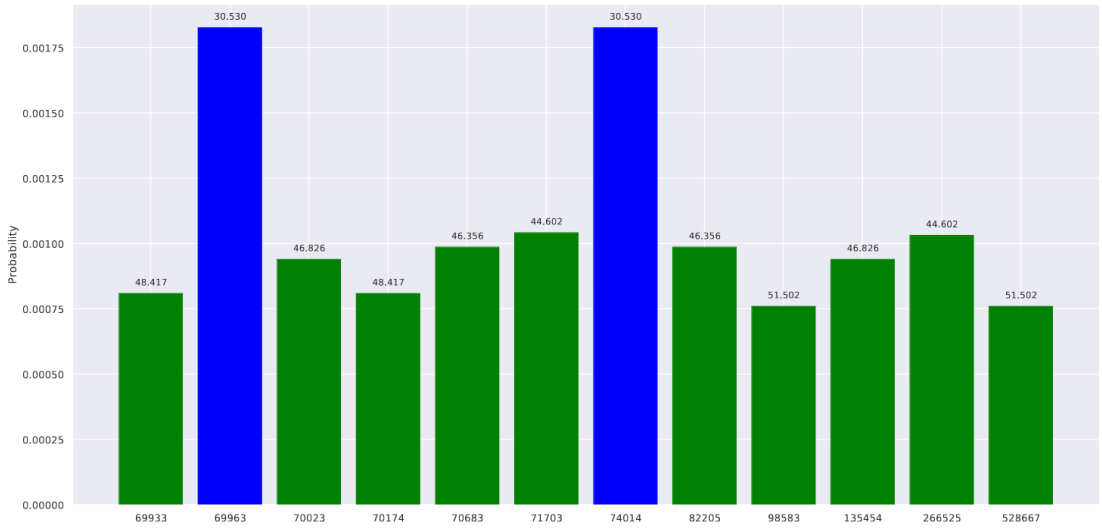

Figure 2.25: Probability distribution for the leading 12 solutions in the example featuring 5 nodes and 3 vehicles, with a *p-step* setting of 24. Optimal states are highlighted in blue. The numerical values atop the bars represent the respective costs [4].

2.3.2 Quantum Annealing Implementations

Following the review of QAOA implementations, this section examines the implementations of the Quantum Annealing (QA) algorithm, as detailed in Section 2.1.3. These studies also concentrate on solving vehicle routing problems.

2.3.2.1 2-Phase Hybrid Solution for CVRP

Feld et al. [35] present three methodologies encompassing quantum and hybrid strategies structured around two distinct algorithmic phases. The initial phase entails a clustering procedure employing the Knapsack Problem (KP), succeeded by a routing phase employing the Travelling Salesman Problem (TSP). Figure 2.27 elucidates these three methodologies. They encompass two discrete QUBOs, each representing an individual algorithm, a singular QUBO matrix comprising both stages and a hybrid approach wherein the TSP segment is executed solely on the quantum annealer. The hybrid approach emerged as the most efficacious throughout the experiments, potentially owing to the early state of quantum hardware.

Because of issues encountered during the clustering phase, the hybrid approach emerged as the most effective among the three, demonstrating competitive performance compared to other classical methodologies relative to the best known solution (BKS). Regarding execution time in a scenario involving 5 clusters, the hybrid algorithms required 1.24 seconds to execute locally in their classical version, while the remote execution took 15.792 seconds. This discrepancy can be attributed to overhead resulting from problem embedding, internet latency, and queuing for the

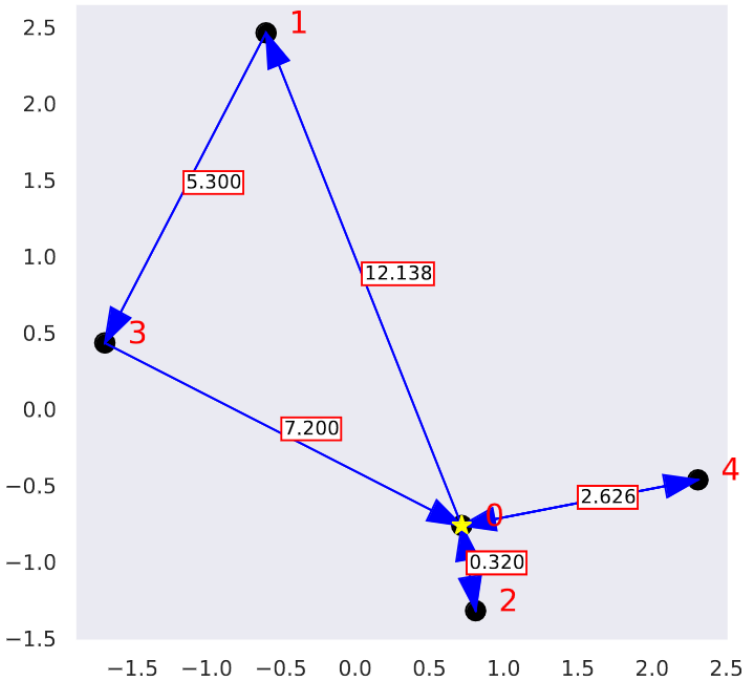

Figure 2.26: Graphical representation of the best solution for the example with 5 nodes and 3 vehicles, by Utkarsh et al. [4].

quantum hardware, given that the actual processing time on the Quantum Processing Unit (QPU) was merely 110 milliseconds, compared to 234 milliseconds in the classical version.

Given the restricted qubit availability on D-Wave's Quantum Processing Unit (QPU), the hybrid solution method outlined in the study did not yet demonstrate clear advantages in solution quality or computational time, offering only reasonable approximations for large problem instances compared to the BKS. Nevertheless, the authors have introduced a technique to embed and partition intricate problems into subproblems solvable by the annealer. This approach lays the groundwork for future investigations, anticipating enhancements in hardware that will diminish the overhead in problem embedding.

2.3.2.2 CVRP with Time and State

Irie et al. [49] introduced a new QUBO formulation of the CVRP, incorporating a timetable that describes the time evolution of the vehicles and their states. This feature enables the implementation of various travel rules based on the state of each vehicle, offering a flexible approach to modelling different VRP problems by adjusting these state rules.

The **concept of time** in this study diverges significantly from the conventional notion of time steps used in many other formulations. For instance, if a location is far from others, travelling there will take longer than to nearer locations, a factor not considered in traditional time-step formulations. Integrating time into the QUBO formulation facilitates modelling time windows, non-simultaneous arrivals, and chronological variations in the cost function.

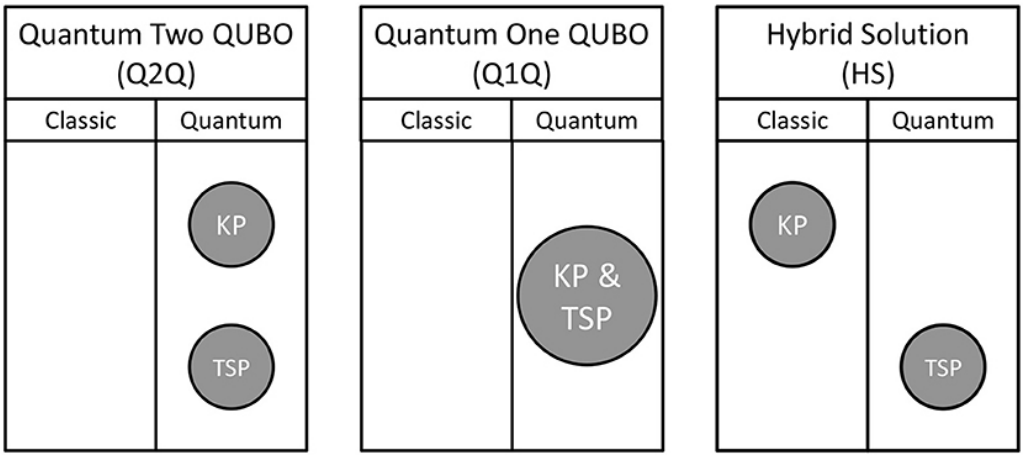

Figure 2.27: Assignment of the clustering and routing phases in 3 different approaches [35].

To introduce time, the study begins by defining binary variables $x_{\tau,i,k}$, where τ represents a time interval in the timetable, i is a location such that $i \in \mathbb{N}$, and k is one of the vehicles. The time interval divides the total time T into several units $\Delta t = t_{\tau+1} - t_{\tau}$. The binary variable is 1 if the vehicle arrives at that location during the specific time interval. The cost function (and respective matrix) must be adjusted to accommodate this formulation.

Regarding constraints, Irie et al. diverge from the conventional penalties typically added in QUBO to the objective function. Instead, they introduce additional parameters that negatively adjust the cost based on the restrictions. Examples of constraints used include:

- No vehicle will arrive at two different locations within the same time interval.
- If a vehicle arrives at location i during a time interval, it will not arrive at location i in any other time interval.
- If a vehicle arrives at location i during a time interval, then other vehicles will not arrive at location i in any time interval.

To incorporate **capacity** restrictions, the study introduces a capacity index into the binary variables, $x_{\tau,i,k,c}$, where $q_m \leq c_m \leq Q_m$, defining the indices between the minimum and maximum vehicle capacities. Additionally, a new constraint ensures that a vehicle arriving at any city during a given time interval cannot be assigned more than one capacity status.

The same technique used for capacity can be applied to any other measurement, leading to the concept of **state**. By adding another index, the state of the vehicles can be extended, and new rules (constraints) can be incorporated by modifying the cost function.

The study validated its formulation using the D-Wave 2000Q quantum annealer, with an example utilizing 83 qubits. However, the result included overtime travelling, highlighting that time windows sometimes prevent the shortest trips. For real-world applications, the required number of qubits increases rapidly. For approximately 30 locations and 20-minute scheduling over half a day, more than 2000 qubits would be necessary.

2.3.3 Ride Pooling Problem

The Ride Pooling Problem (RPP) is a VRP variant where multiple customers can request pickups and drop-offs from shared vehicles, similar to flexible bus routes or pickup services through mobile applications. Cattelan and Yarkoni [13] proposed a QUBO formulation for the RPP for quantum optimization algorithms, such as QAOA or Quantum Annealing. RPP has been used to solve many problems but has gained recent interest due to services such as Lyft and Uber. This variant's most distinctive characteristic is that vehicles neither start nor end at a common depot. The problem is described in the paper as:

Given a fleet of vehicles A and a set of requests (i, j) with pickup points i and drop-off points j for n customers, assign routes to vehicles in A to pick up and drop off all customers such that the cost of the target (objective) function is minimized.

In this study, the binary decision variables are defined using a step-based approach, where $x_{k,i,s}$ takes a value of 1 if vehicle k is at location $i \in \mathbb{N}$ at step s. Since every customer has a pickup and drop-off point, the maximum number of steps is $2C+1$ for each vehicle, where C is the number of ride requests. The objective function is given by Equation 2.53, where ε is a small positive constant used to prevent division by zero.

$$\min \quad \sum_{i=0}^{n}\sum_{j=1}^{n}\sum_{s=0}^{S-1}\sum_{k=1}^{K}\frac{c_{i,j}}{W}x_{k,i,s}x_{k,j,s+1}+\sum_{i=1}^{n}\sum_{s=0}^{S-2}\sum_{k=1}^{K}\frac{c_{i,j}}{W}x_{k,i,s}x_{0,k,s+1}, \quad W=\varepsilon+\max_{i,j=0,\ldots,n}c_{i,j} \tag{2.53}$$

Subject to:

$$\sum_{k=1}^{K}\sum_{s=1}^{S}x_{k,i,s}=1, \quad i=1,\ldots,n \tag{2.54}$$

$$\sum_{i=0}^{n}x_{k,i,s}=1, \quad k=1,\ldots,K, \quad s=1,\ldots,S-1 \tag{2.55}$$

$$\sum_{i=1}^{n}x_{k,i,S}\leq 1, \quad k=1,\ldots,K \tag{2.56}$$

$$\sum_{i=1}^{n}\sum_{s2=1}^{s1}q_i x_{k,i,s2}\leq Q_k, \quad k=1,\ldots,K, \quad s1=1,\ldots,S \tag{2.57}$$

$$x_{k,i,s}\in\{0,1\}, \quad i=0,\ldots,n, \quad k=1,\ldots,K, \quad s=0,\ldots,S \tag{2.58}$$

And subject to the following incentive (negated penalty):

$$\sum_{k=1}^{K}\sum_{(i,j)\in\mathbb{C}}\sum_{\substack{s_1,s_2=1\\ s_1<s_2}}^{S}-x_{k,i,s_1}x_{k,j,s_2} \tag{2.59}$$

The first term of the objective function 2.53 sums the distances between each location (including the starting point) and all other locations across all steps and vehicles (which always start at step 0). The second term adds the distance from all locations to the starting point, except for the last step, in which the start is removed in order to meet constraints 2.55 and 2.56. Note that the cost of staying at the starting point is zero.

Constraint 2.54 ensures that each location, except for the starting point, is visited only once. Constraints 2.55 and 2.56 ensure that each vehicle visits only one location at any given step. Since at least one location is visited, it is possible to reduce the route length of all vehicles by 1, except when one vehicle handles all requests. Thus, the last step can be transformed into a half-hot constraint 2.56, where possibly no location is visited at the last step. This reduces the number of variables in the QUBO and slightly modifies the objective function.

Constraint 2.57 enforces the vehicle capacity limit (Q_k), given that p_i is the number of customers picked up (or dropped off), while this value may be negative. Finally, the incentive (a negative coefficient that reduces the objective function) in 2.59 ensures that trip requests are fulfilled. This approach is used instead of a penalty, achieving the same result with a more straightforward formulation.

The study demonstrates that this formulation performs better than standard edge-based formulations, such as those presented in Section 2.2.2, for quantum optimization due to its reduced qubit demand. While the step-based formulation requires $O(kN^2)$ variables, the edge-based formulation would require $O(kN^3)$ variables.

Finally, the paper demonstrates how to reduce the number of qubits used in the problem by eliminating some redundant variables and pre-fixing their values before running the algorithm. This reduction is essential for current quantum hardware, as it decreases overhead and enables the solution of problem instances that might otherwise be too large. For example, all vehicles must start their path from the starting point, meaning they are not present at any other location in the first step. The study provides additional examples like this one.

Despite making a valuable contribution, this study remains theoretical, as it does not present any practical results, either through simulations or actual quantum hardware experiments. This limitation opens a promising avenue for future research in the realm of the RPP. Subsequent studies could focus on implementing and testing the proposed methods in practical settings, yielding insights into their scalability and real-world applicability, further advancing the field.

Chapter 3

Modelling Vehicle Routing Problems

This chapter elaborates on the variations of the Vehicle Routing Problem (VRP) introduced in Section 2.1.1.1. It provides a detailed explanation of their formulation for this dissertation using the Quadratic Unconstrained Binary Optimization (QUBO) model detailed in Section 2.1.5.1. The VRP variations explored in this study are:

- **Vehicle Routing Problem (VRP)**: The essential vehicle routing problem, with multiple vehicles starting and ending at a joint depot, without capacity constraints.
- **Capacitated Vehicle Routing Problem (CVRP)**: Similar to VRP, but with all vehicles sharing an equal capacity limit.
- **Multi-Capacitated Vehicle Routing Problem (Multi-CVRP)**: Similar to CVRP, but with each vehicle having a different capacity.
- **Ride Pooling Problem (RPP)**: A VRP variation where the joint depot is removed and the input includes customer requests with pickup and drop-off points.

For each of these variations, we present a QUBO formulation largely based on the related work discussed in Sections 2.2 and 2.3. Special attention is given to formulating these problems with the minimum number of variables possible, considering the development of quantum algorithms and the limited number of qubits available in current quantum hardware. Each section subsequently examines the number of variables and qubits used in the formulation, analyzing the complexity to predict how the formulation will scale on actual hardware.

Furthermore, for each variation, we provide an analysis that identifies and removes redundant variables, thereby optimizing the problem and reducing the number of qubits needed. This approach not only optimizes the use of quantum resources but also enhances the feasibility of implementing these algorithms on current and near-future quantum devices.

3.1 Vehicle Routing Problem

The basis model for the Vehicle Routing Problem (VRP) is a two-index formulation, as described in Section 2.2.2.1 [72, 100]. The primary distinction of this formulation from previous studies lies in reusing the starting point at index 0 as both the initial and final joint depot for all vehicles, thereby eliminating the need for an additional index $N+1$ for the final stop. This modification moderately reduces the number of variables and affects the constraints, as detailed in Equations 3.2-3.8. This simplification was inspired by the formulation proposed by Qiskit Optimization [90], a part of the Qiskit Community's vehicle routing documentation.

Here, we define binary decision variables x_{ij} that equal 1 if a route directly connects node i to node j, for $i, j \in \mathbb{N}$. Moreover, u_i is an auxiliary integer variable part of the Miller-Tucker-Zemlin (MTZ) subtour elimination constraint. These extra variables are assigned a value for each node, except for the depot. If a vehicle travels from i to j, then the value of u_j must be higher than u_i, thus controlling the route flow [73, 2].

Equation 3.1 specifies the objective function for this formulation, where c_{ij} is the cost of travelling from node i to node j, K is the number of available vehicles, and n is the number of nodes in the problem, with 0 representing the depot.

$$\min \quad \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} \tag{3.1}$$

Subject to:

$$\sum_{j=0}^{n} x_{ji} = 1, \quad i = 1, \ldots, n, \text{ with } i \neq j \tag{3.2}$$

$$\sum_{j=0}^{n} x_{ij} = 1, \quad i = 1, \ldots, n, \text{ with } i \neq j \tag{3.3}$$

$$\sum_{i=1}^{n} x_{0i} = K \tag{3.4}$$

$$\sum_{i=1}^{n} x_{i0} = K \tag{3.5}$$

$$u_i - u_j + n x_{ij} \leq n - 1, \quad i, j = 1, \ldots, n, \text{ with } i \neq j \tag{3.6}$$

$$1 \leq u_i \leq n, \quad i = 1, \ldots, n \tag{3.7}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 0, \ldots, n \tag{3.8}$$

Constraints 3.2 and 3.3 ensure each node is visited exactly once without comparing edges connected to the same node. Constraints 3.4 and 3.5 ensure that the exact number of vehicles

leave from and return to the depot, thereby forming the correct number of routes. Constraints 3.6 and 3.7 implement the MTZ subtour elimination. Here, the value of n (the number of nodes) is used as the upper limit for the variables since an upper bound is required for the formulation, and the vehicles have infinite capacity. The final constraint, 3.8, bounds the binary variables.

The number of variables directly impacts the number of qubits required to execute algorithms, as each binary variable maps to a qubit in QAOA and Quantum Annealing. Moreover, the limited number of qubits in current quantum hardware may render it impossible to execute algorithms with many variables. Therefore, analysing how the formulation scales relative to the input size is crucial. A complexity analysis based on **Big O** notation [12] can be used. It is also worth noting that integer variables incur higher costs than binary ones due to the subsequent conversion into binary variables. Similarly, inequalities require the introduction of integer slack variables when transformed into equalities. This transformation process is elaborated upon in Section 2.1.5.1.

Given these considerations, the VRP formulation requires $O(n^2)$ binary variables to represent all edges, along with $O(n)$ integer variables for subtour elimination and additional $O(n^2)$ slack variables arising from the inequalities used in subtour elimination. In sum, the formulation involves $O(n^2)$ variables, with $O(n^2)$ integer variables used.

3.1.1 Optimization Step

As discussed in Section 2.1.4, quantum computers currently face severe limitations in both the number of available qubits and the challenge of quantum noise as qubit count increases, leading to higher error rates and degraded solution quality. Therefore, even polynomial scaling of qubits (such as the quadratic complexity mentioned earlier) imposes significant overhead on quantum computing.

Nevertheless, strategically fixing variables based on logical deduction and intuition can eliminate redundant variables before executing the algorithm. While these simplifications may not decrease the necessary space complexity, they can enhance the performance of quantum computers, enabling the execution of larger problem instances [13].

In this case, it is possible to eliminate n binary variables by recognizing that a vehicle should never travel from the same location to itself, implying there are no edges between identical nodes. This simplification is expressed in Equation 3.9.

$$x_{ii} = 0, \quad i = 0, \ldots, n \tag{3.9}$$

3.2 Capacitated Vehicle Routing Problem

The Capacitated Vehicle Routing Problem (CVRP) closely resembles the problem discussed in Section 3.1, except that all vehicles share a fixed maximum capacity. Therefore, each location is associated with a demand that must not exceed this capacity when serviced by a vehicle. This results in a formulation that closely mirrors the VRP, with the capacity constraint integrated into the Miller-Tucker-Zemlin (MTZ) subtour elimination constraint.

In terms of variables, we use the same x_{ij} binary variables and the auxiliary u_i integer variable, where $i, j \in \mathbb{N}$. Regarding constants, we maintain c_{ij} as the cost of edges, K as the number of vehicles, and n as the number of nodes. Additionally, we introduce Q as the maximum vehicle capacity and q_i as the demand at node i. It is important to note that, in this problem, the demand is always positive. Therefore, Equation 3.1 continues to define the objective function for this problem, with constraints 3.6 and 3.7 replaced by Equations 3.10 and 3.11.

$$u_i - u_j + Qx_{ij} \leq Q - q_j, \quad i, j = 1, \ldots, n, \text{ with } i \neq j \tag{3.10}$$

$$q_i \leq u_i \leq Q, \quad i = 1, \ldots, n \tag{3.11}$$

To gain insight into the new constraints, consider the scenarios where a vehicle travels or does not travel from node i to node j:

- If a vehicle travels from i to j, then $x_{ij} = 1$, and Equation 3.10 can be rewritten as $u_j - u_i \geq q_j$. This inequality ensures that u_j is at least q_j larger than u_i, accommodating the demand q_j when traveling from node i to node j. These variables effectively accumulate load and enforce the vehicle capacity limit, reflected in the upper bound of Equation 3.11, which is set to Q.
- If no vehicle travels from i to j, then $x_{ij} = 0$, and the constraint remains valid as $u_i - Q \leq u_j - q_j$. Here, Q represents the maximum possible value for u_i, and q_j represents the minimum possible value for u_j. Therefore, $u_i - Q \leq 0$ and $u_j - q_j \geq 0$, ensuring the constraint's satisfaction [2].

In this formulation, the count of binary variables remains $O(n^2)$, while integer variables, including slack variables, are $O(n^2)$. The redundant variables removed are identical to those detailed in Section 3.1.1, eliminating n binary variables.

3.3 Multi-Capacitated Vehicle Routing Problem

In terms of requirements, the Multi-Capacitated Vehicle Routing Problem (Multi-CVRP) is similar to the CVRP discussed in Section 3.2, except that each vehicle has its own maximum capacity. Despite this similarity, this distinction requires a completely different formulation. We must now track each route with a specific capacity, unlike before when we only needed to ensure the correct number of vehicles departing from and arriving at the depot. Consequently, we can no longer use the two-index formulation applied in the VRP and CVRP.

For the Multi-CVRP, we adopt a step-based approach described by Cattelan and Yarkoni [13] based on a TSP formulation proposed by A. Lucas [62]. In this formulation, the binary variables with three indices $x_{k,i,s}$ indicate whether vehicle k visits location $i \in \mathbb{N}$ at time step s. We consider the number of steps $n+1$, allowing for the worst-case scenario where a single vehicle visits all nodes. Note that all vehicles start at the depot at step 0, requiring the additional time step.

Equation 3.12 defines the objective function for this formulation, where $c_{i,j}$ represents the cost of traveling from node i to node j, with $i, j \in \mathbb{N}$. Furthermore, K is the number of available vehicles, n is the number of nodes, S is the number of steps, Q_k is the capacity of vehicle k, q_i is the demand of location i, and W is a normalization factor that bounds the edge cost between 0 and 1. ε is a small positive constant used to avoid division by zero.

$$\min \quad \sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{s=0}^{S-1} \frac{c_{i,j}}{W} x_{k,i,s} x_{k,j,s+1}, \quad W = \varepsilon + \max_{i,j=0,\ldots,n} c_{i,j} \tag{3.12}$$

Subject to:

$$\sum_{k=1}^{K} \sum_{s=0}^{S} x_{k,i,s} = 1, \quad i = 1, \ldots, n \tag{3.13}$$

$$\sum_{i=0}^{n} x_{k,i,s} = 1, \quad k = 1, \ldots, K, \quad s = 0, \ldots, S \tag{3.14}$$

$$\sum_{i=1}^{n} \sum_{s=0}^{S} q_i x_{k,i,s} \leq Q_k, \quad k = 1, \ldots, K \tag{3.15}$$

$$x_{k,i,s} \in \{0,1\}, \quad i = 0, \ldots, n, \quad k = 1, \ldots, K, \quad s = 0, \ldots, S \tag{3.16}$$

Constraint 3.13 ensures that each node is visited only once across all vehicles and time steps. Constraint 3.14 guarantees that each vehicle is in only one location at each time step. Constraint 3.15 restricts the capacity limit of each vehicle by monitoring its accumulated load at the last step. This constraint assumes that the demand at each node is always greater or equal to zero. Lastly, constraint 3.16 defines the binary decision variables with three indices.

Regarding qubit usage, this formulation requires $O(Kn^2)$ binary decision variables since the number of steps is $n+1$. Additionally, $O(K)$ integer slack variables are needed due to the capacity constraints for each vehicle. Notably, if negative node demands were to be supported, we would need one capacity constraint per vehicle and time step, increasing the number of slack variables to $O(Kn)$.

To demonstrate that this step-based formulation uses fewer qubits than any edge-based formulation, such as the three-index commodity flow approach shown in Section 2.2.2.3, let's informally deduce the smallest possible formulation. We start by defining the three-index decision variables $x_{i,j,k}$, resulting in $O(Kn^2)$ binary variables, identical to the current step-based model. We can use constraints like MTZ or the commodity flow approach to restrict the maximum capacity for each vehicle. We need at least $O(n)$ auxiliary integer variables regardless of the choice. That said, we would require an inequality flow constraint for each pair of nodes and constraints to link each vehicle to the appropriate flow constraints, amounting to at least $O(n^2)$ slack variables. In total, the best-case scenario involves $O(Kn^2)$ binary and $O(n^2)$ integer variables, which is asymptotically worse than the current formulation, especially considering that $K \ll n$ in any practical situation.

3.3.1 Optimization Step

As with previous formulations, there are redundant variables in the Multi-CVRP that can be fixed to reduce the number of qubits used. In this problem, we observe that all vehicles must start and end at the depot, allowing us to fix these values in advance, as shown in Equation 3.17. This reduction eliminates $2K$ binary variables.

$$x_{k,0,0} = 1, \quad x_{k,0,S} = 1, \quad k = 1, \ldots, K \tag{3.17}$$

Furthermore, if the vehicles start and end at the depot, they cannot start or end at any other location. Therefore, we can fix these variables at the first and last step, as shown in Equation 3.18. This reduces up to $2Kn$ binary variables.

$$x_{k,i,0} = 0, \quad x_{k,i,S} = 0, \quad k = 1, \ldots, K, \quad i = 1, \ldots, n \tag{3.18}$$

3.4 Ride Pooling Problem

As previously mentioned in Sections 2.1.1.1 and 2.3.3, the Ride Pooling Problem (RPP) involves transporting multiple customers based on their pickup and drop-off points. Vehicles, such as flexible buses or shared cars, determine optimal routes based on these trip requests. Unlike the other variations analyzed earlier, RPP does not rely on a joint depot, allowing for entirely disjoint vehicle routes.

This formulation is inspired by the work done by Cattelan and Yarkoni [13] and explained in Section 2.3.3. However, this dissertation makes a crucial change. In Cattelan and Yarkoni's paper, vehicles have a pre-defined starting point since their study mainly focused on shared car services. In this dissertation, we are eliminating that starting point and letting the algorithm decide the complete path of each vehicle. This allows for the optimization of flexible bus and metro routes (or any other similar transport method), which is precisely the dissertation's goal.

To achieve this, rather than defining a specific starting point, we treat index 0 as a state where vehicles remain stationary for one or multiple steps without incurring additional costs. The algorithm subsequently determines the optimal location for each vehicle to initiate its journey, considering that the cost from this fictitious starting point to the actual starting point is always 0. This approach involves adding a node at the beginning and adjusting the objective function accordingly.

For RPP, the input includes the number of vehicles, K, a set of customer requests $(i, j) \in \mathtt{C}$ where $i, j \in \mathbb{N}$, the number of nodes, n, the capacity of each vehicle k, Q_k, and the demand at location i, q_i, which might be negative when passengers are being dropped off. Similar to Multi-CVRP, we define binary decision variables $x_{k,i,s}$, indicating whether vehicle k visits location $i \in \mathbb{N}$ at time step s. In this case, the number of steps S is defined as $2C+1$ since the worst-case scenario involves a single vehicle satisfying each trip individually, plus the initial step representing the idle vehicle.

Equation 3.19 defines the objective function for this formulation, where $c_{i,j}$ represents the cost of travelling from node i to node j, and W is the same normalization factor as in Section 3.3.

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=0}^{S-1}\sum_{k=1}^{K}\frac{c_{i,j}}{W}x_{k,i,s}x_{k,j,s+1}+\sum_{i=1}^{n}\sum_{s=0}^{S-1}\sum_{k=1}^{K}x_{k,i,s}x_{k,0,s+1}, \quad W=\varepsilon+\max_{i,j=1,\ldots,n}c_{i,j} \tag{3.19}$$

Subject to:

$$\sum_{k=1}^{K}\sum_{s=1}^{S}x_{k,i,s}=1, \quad i=1,\ldots,n \tag{3.20}$$

$$\sum_{i=0}^{n}x_{k,i,s}=1, \quad k=1,\ldots,K, \quad s=0,\ldots,S-1 \tag{3.21}$$

$$\sum_{i=1}^{n}x_{k,i,S}\leq 1, \quad k=1,\ldots,K \tag{3.22}$$

$$\sum_{i=1}^{n}\sum_{s2=1}^{s1}q_i x_{k,i,s2}\leq Q_k, \quad k=1,\ldots,K, \quad s1=1,\ldots,S \tag{3.23}$$

$$x_{k,i,s}\in\{0,1\}, \quad i=0,\ldots,n, \quad k=1,\ldots,K, \quad s=0,\ldots,S \tag{3.24}$$

And subject to the following incentive (negated penalty):

$$\sum_{k=1}^{K}\sum_{(i,j)\in\mathbb{C}}\sum_{\substack{s_1,s_2=1\\ s_1<s_2}}^{S}-x_{k,i,s_1}x_{k,j,s_2} \tag{3.25}$$

The first term of the objective function 3.19 sums the distances between all locations, excluding the starting point, across all steps for each vehicle. The second term adds a penalty equal to 1 if any vehicle travels back to the fictional starting point, aiming to discourage such movements. As usual, the cost of staying at the starting point is zero.

Constraint 3.20 ensures that each location, excluding the starting point, is visited only once. Constraints 3.21 and 3.22 ensure that each vehicle visits only one location at any given step. Like the original study, the last step can be transformed into a half-hot constraint 3.22 to reduce variables, where it's possible that no location is visited at the last step.

Constraint 3.23 is optional and enforces the vehicle capacity limit, applicable when vehicles have limited capacities, which may be the same or varied among vehicles. The constraint is evaluated at every step due to potential negative demands. Lastly, the incentive in Equation 3.25 ensures that trip requests are fulfilled. This approach replaces a penalty, achieving the same goal with a simpler formulation.

Regarding qubit usage, this RPP formulation requires $O(nKC)$ binary decision variables, given that the number of steps is $2C+1$. Additionally, $O(KC)$ integer slack variables may be needed

due to the capacity constraints. If the vehicles in a problem instance have unlimited capacity, then no integer variables are required.

3.4.1 Optimization Step

This RPP formulation allows for several simplifications by fixing redundant variable values. For instance, vehicles must begin at their designated starting points and nowhere else. This reduction eliminates $K(n+1)$ binary variables, as demonstrated in Equations 3.26 and 3.27.

$$x_{k,0,0} = 1, \quad k = 1,\ldots,K \tag{3.26}$$

$$x_{k,i,0} = 0, \quad k = 1,\ldots,K, \quad i = 1,\ldots,n \tag{3.27}$$

Another aspect is that no vehicle can be at the starting point at the last step. This relates to the earlier modification for the half-hot constraint 3.22, which allows vehicles not to visit any location at the last step. Therefore, they do not need to remain at the depot even if they do not visit any locations. This simplification reduces K variables, as shown in Equation 3.28.

$$x_{k,0,S} = 0, \quad k = 1,\ldots,K \tag{3.28}$$

Lastly, given the nature of the trips, it is logically impossible to go from the starting point to any drop-off location at step 1 or to end at any pickup point in the last step. Otherwise, the customers' trips would not be fulfilled. This eliminates $2KC$ binary variables, as shown in Equation 3.29.

$$x_{k,j,1} = 0, \quad x_{k,i,S} = 0, \quad k = 1,\ldots,K, \quad (i,j) \in \mathtt{C} \tag{3.29}$$

3.5 Overview

In summary, this chapter detailed the modelling of four variations of the vehicle routing problem using the Quadratic Unconstrained Binary Optimization (QUBO) framework: VRP, CVRP, Multi-CVRP, and RPP. Consequently, these problems can be addressed using the Quantum Approximate Optimization Algorithm (QAOA) and Quantum Annealing (QA) algorithms. The chapter also demonstrated methods to optimize each formulation for minimal qubit usage. The implementation of these algorithms is discussed in Chapter 4, while the performance and efficiency of the models are evaluated in Chapter 5.

Chapter 4

Software Design for Quantum Routing Algorithms

This chapter elaborates on the software implementation of the quantum algorithms capable of solving the formulations discussed in Chapter 3. The code is entirely developed in Python and is openly available on GitHub.

The project's architecture is designed with extensibility in mind, allowing for the easy addition of new QUBO formulations or solvers for entirely different algorithms or quantum computing providers. The architecture is detailed in Section 4.1. It currently supports the QAOA algorithm with IBM systems, as explained in Section 4.2, Quantum Annealing with D-Wave systems, as described in Section 4.3, and a classical implementation for state-of-the-art comparison, detailed in Section 4.4.

Finally, the chapter describes how the obtained solutions are handled in Section 4.5, including how they are processed, stored, and displayed.

4.1 Software Architecture

In quantum computation, it is common to see implementations tailored to a specific problem and a single system or algorithm, such as implementing the CVRP exclusively for D-Wave's quantum annealers. However, the requirements for this dissertation are more complex, encompassing multiple formulations (VRP, CVRP, Multi-CVRP, and RPP) and multiple algorithms (QAOA and Quantum Annealing). Therefore, the project focuses on creating a flexible framework for solving routing QUBO problems, designed to be easily extensible with additional formulations, algorithms, and quantum providers. This section will detail the implementation of this framework, specifying its architecture.

The initial challenge was to model each formulation without needing to re-implement them for every different system, each of which typically has its own method for modelling QUBO or Ising models. For instance, IBM's quantum software development kit (SDK), Qiskit, uses *Quadratic*

Programs as developed by the Qiskit Optimization library [90], while D-Wave's SDK, Ocean, uses *Binary* or *Constrained Quadratic Models* [47].

This problem was addressed by incorporating the ***docplex*** library [46], developed by IBM, to model all the formulations. *Docplex* is a decision optimization library initially inspired by the *simplex* method [21]. It has since evolved to support modelling constraint programming models in Python, amongst other features. Besides being a mature and widely used library, it is straightforward to convert a *docplex* model into the *Quadratic Programs* used by Qiskit, given that IBM also developed it. This makes it an excellent choice for the use case.

The next step in the process was to create **adapter** classes. These classes are designed to convert *docplex* models into their respective system or SDK formats. The essential information typically converted includes the variables, objective function, and problem constraints. These adapted models can then be used by **solvers**, which are top-level classes responsible for receiving problem inputs and parameters, constructing the appropriate formulations using the models and adapters, and executing the respective algorithms.

The ultimate result produced by all solvers is a **VRP Solution**, an object containing all the information and methods necessary to analyze, store, visualize, or process the solution to the problem. Further details on this implementation are provided in Section 4.5.

Figure 4.1 presents a UML diagram illustrating all the previously mentioned classes. Several key points should be noted:

- *VRP* serves as the base class holding the input parameters for the problem. It is directly used by the *ClassicSolver*, while the other solvers use QUBO models that extend from it.
- Despite having different names, *InfiniteCVRP* corresponds to the VRP formulation discussed in Section 3.1, while *InfiniteRPP* and *CapacityRPP* represent the RPP formulations described in Section 3.4 without and with capacity constraints, respectively.
- The diagram is simplified by excluding some classes or source files that are not essential for overall comprehension. For example, the file containing cost functions, a parameter of VRP, is omitted.

4.1.1 Cost Functions

As explained in Chapter 3, all VRP formulations take as input a cost function that contains information on the travel costs between every pair of locations in the problem, where the costs for travelling in each direction may differ. In this implementation, it is possible to directly provide the cost matrix as input associated with a unit such as meters or seconds. However, calculating this matrix each time we want to run an experiment or solve a problem instance would be inconvenient.

Therefore, several predefined cost functions can be selected as input. These functions take the provided locations and calculate the cost (usually distance or time) between every pair of nodes. The currently implemented functions are detailed below.

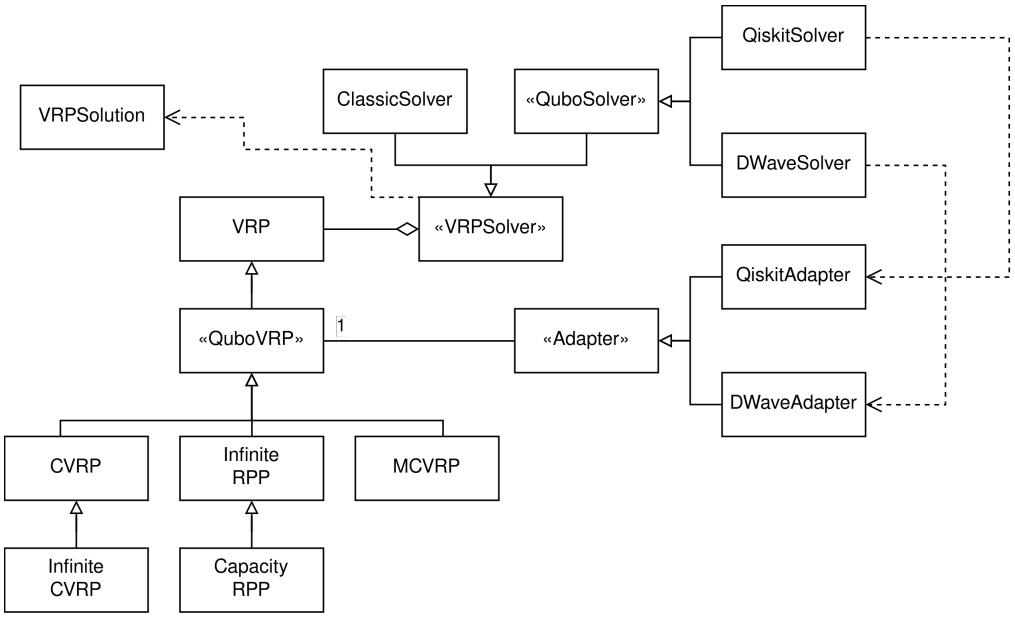

Figure 4.1: Simplified UML Class Diagram illustrating the software framework developed for solving quantum VRP algorithms.

Manhattan Distance

Using the Manhattan distance as the cost function is a straightforward and efficient approach that works well for most experiments. Therefore, it is set as the default function if none is specified. Equation 4.1 shows the formula for the distance between a pair of nodes.

$$c_{i,j} = |x_i - x_j| + |y_i - y_j| \tag{4.1}$$

Euclidean Distance

A similar approach to the Manhattan distance is the Euclidean distance. Although almost as simple as the Manhattan distance, the Euclidean distance is preferable in situations where it is essential to consider the straight-line path between locations rather than just the relative distance between the nodes. Equation 4.2 presents the formula for the Euclidean distance.

$$c_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{4.2}$$

Google Distance Matrix API

Google offers a service based on its Maps platform called the Distance Matrix API. This API accepts HTTPS requests containing origin and destination coordinates and returns the travel distance or duration for each combination [60]. This is useful for calculating realistic cost matrices when

real locations are defined by latitude and longitude. The main disadvantage of this service is its pay-as-you-go pricing model, although the free plan suffices for a few experiments.

In addition to origins and destinations, the API accepts several other parameters, such as the unit of measurement (e.g., metric or imperial) and the travel mode (e.g., driving, walking, public transportation). Consider the following example request:

```
https://maps.googleapis.com/maps/api/distancematrix/json?
origins=41.157944,-8.629105&destinations=41.14557,-8.61056&
mode=driving&units=metric&key={YOUR_API_KEY}
```

The responses are given in the JSON format, as shown in the example below:

```
{
   "destination_addresses": [
      "Praça de Mouzinho de Albuquerque, 4050-115 Porto, Portugal"
   ],
   "origin_addresses": [
      "Praça de Almeida Garrett, 4000-069 Porto, Portugal"
   ],
  "rows": [
    {
      "elements": [
        {
          "distance": {
            "text": "3.3 km",
            "value": 3300
          },
          "duration": {
            "text": "8 mins",
            "value": 480
          },
          "status": "OK"
        }
      ]
    }
  ],
  "status": "OK"
}
```

As we observe, the API returns a list of rows with elements that can be easily transformed into a cost matrix. However, the API is limited to 100 elements per request [61]. Therefore, we

need to divide the calculations into multiple chunks and merge them one by one. This approach is practical, but it is essential to be mindful of the number of elements and requests generated to avoid unexpected service costs. The distance function also accepts a unit parameter, allowing it to return either the distance (in meters) or the duration (in seconds).

4.2 QAOA Implementation

The Quantum Approximate Optimization Algorithm (QAOA), detailed in Section 2.1.5.2, optimizes QUBO formulations. In this project, the algorithm is implemented using the Qiskit ecosystem, an open-source quantum framework developed by IBM. As outlined in Section 4.1, two primary components were developed: *QiskitAdapter* and *QiskitSolver*.

Implementing the Qiskit adapter is straightforward since problems are formulated using the *docplex* model. The Qiskit Optimization library provides a *from_docplex_mp()* function that converts this model into a *QuadraticProgram* usable by Qiskit, handling most of the required tasks.

In contrast, the Qiskit solver presents a more intricate setup. Most importantly, it uses the Qiskit Algorithms and Qiskit Optimization libraries, specifically incorporating the *QAOA* algorithm along with the *MinimumEigenOptimizer* and *WarmStartQAOAOptimizer* optimizers. To delve deeper into this solver, let us examine its specific parameters (in addition to the problem's input), detailed in Table 4.1.

If *classical_solver* is set to True, IBM's CPLEX optimizer [46] is employed to generate a solution. This approach may help test formulations or compare the performance of quantum optimizers against a classical one. Otherwise, the program converts all constraints into penalties and integer variables into binary equivalents. This encoding ensures compatibility with QAOA. Subsequently, the algorithm executes with the appropriate optimizer, determined by whether the warm start is enabled.

It's important to note that the default value of 'sampler' is a perfect quantum simulator, which computes the probabilities of qubit states rather than sampling them (therefore, the number of shots is not a required parameter). Quantum simulators are computationally intensive, so they should only be used for testing with a small number of qubits. Larger problem instances require a real quantum computer or a more robust cloud simulator. Alternatively, Qiskit provides access to noisy simulators, known as fake providers, designed to emulate the behavior of real quantum systems using snapshots [78]. Examples include providers like FakePrague. They help evaluate the algorithm's performance on quantum machines without consuming resources on actual systems.

If a feasible solution is found, we extract the final variables and timing information from the results after the algorithm executes. We then parse these into an instance of *VRPSolution*, which we use to display, store, and analyze the solution.

4.2.1 IBM Sessions and Circuit Transpilation

In Qiskit, sessions enable the execution of iterative workflows involving multiple jobs on quantum computers. Using these sessions helps avoid delays caused by queuing each job separately, which

Table 4.1: Parameters available in the Qiskit QAOA solver.

Parameter	**Type**	**Description**	**Default**
classical_solver	boolean	Whether to use a classical optimizer (CPLEX) instead of a quantum optimizer.	False
simplify	boolean	Whether to simplify the problem by removing redundant variables.	True
track_progress	boolean	Whether to track progress by printing logs every iteration of the algorithm.	True
sampler	Sampler	The sampler used in the quantum optimizer. Can either be a simulator or a real quantum computer.	Sampler
classical_optimizer	Optimizer	Classical optimizer used to decide new parameters in between QAOA iterations.	COBYLA
warm_start	boolean	Whether to run QAOA with a warm start.	False
pre_solver	OptimizationAlgorithm	Classical optimizer used to pre-solve the problem if warm start is used.	CplexOptimizer

can take several hours depending on machine demand [78]. This capability is particularly crucial for QAOA, as it frequently exchanges information between classical and quantum computers, initiating a new quantum job with each iteration. This algorithm is therefore only feasible with sessions.

By initiating a session, we can submit new jobs without waiting in the entire queue as long as the session remains active. The session will automatically time out after a period of inactivity following the last submitted job. Sessions can be easily defined using the Qiskit Runtime Service to select the appropriate backend.

Despite the theoretical advantages of sessions, a significant implementation challenge emerged. Effective March 1st, 2024, Qiskit Runtime services require circuits to be transformed to use only instructions supported by the system's Instruction Set Architecture (ISA) [78]. This requires circuit transpilation and layout operations before submission to the quantum computer. However, as of June 2024, the Qiskit library implementing QAOA optimizers still needs to be updated to accommodate this requirement. Consequently, sessions cannot be used on IBM's quantum computers, rendering the execution of QAOA virtually impossible.

To address this issue, a solution was implemented by creating a fork of the Qiskit Algorithms library and working with a local copy. Within this modified version, the circuit undergoes transpilation into ISA instructions, and the ansatz's layout is applied before sending it to the quantum computer in cases where a remote service is used. One way to verify the system is working as intended is by checking the IBM Quantum dashboard, as shown Figure 4.2. The fork of the repository can be consulted on GitHub.

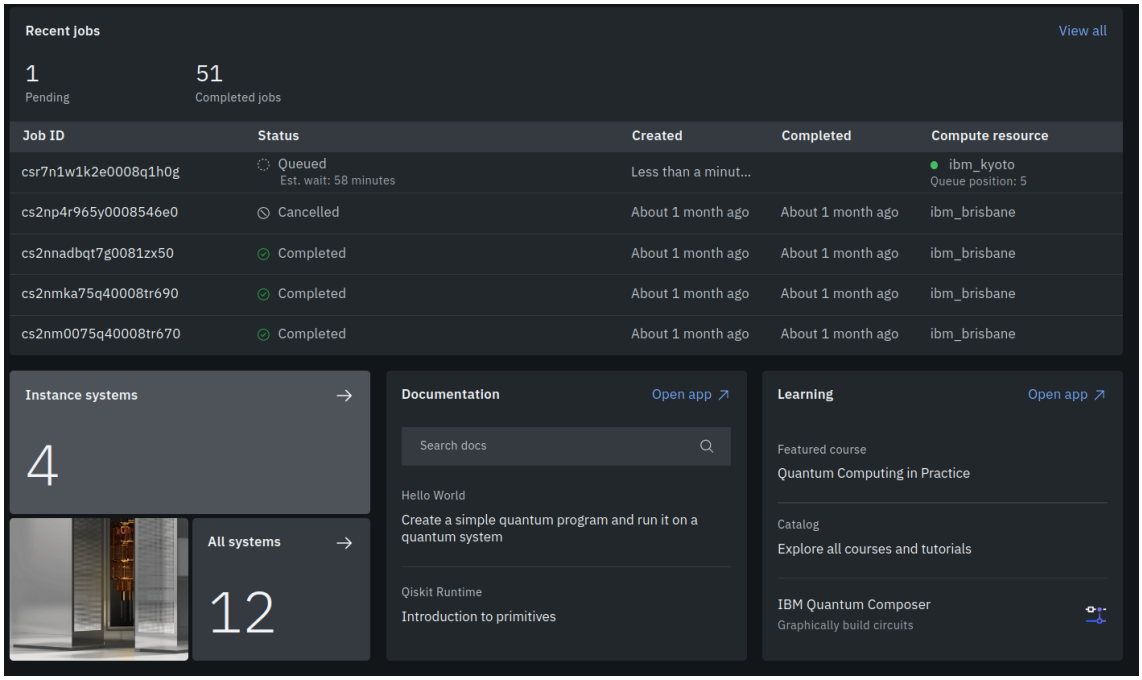

Figure 4.2: Screenshot of the IBM Quantum Dashboard after submitting a QAOA iteration.

4.3 Quantum Annealing Implementation

Quantum Annealing (QA) is another algorithm that optimizes QUBO formulations, detailed in Section 2.1.3. In this case, the algorithm is implemented using the Ocean SDK, an open-source Python framework developed by D-Wave. The hardware currently available for developers is the D-Wave Advantage, based on the Pegasus architecture explained in Section 2.1.4.2. As outlined in Section 4.1, the primary components of the implementation are *DWaveAdapter* and *DWaveSolver*.

Implementing the adapter for Ocean is more complex than in the Qiskit case, as D-Wave uses its own implementation for quadratic models. This is done by parsing *docplex*'s relevant fields (e.g., variables, objective, constraints) and adding them to a *ConstrainedQuadraticModel*.

The solver operates by sampling the solution multiple times using a specified annealer and selecting the best solution from all the samples obtained. The number of samples done varies significantly based on the parameters provided to the solver. The detailed differences are outlined in Table 4.2.

Before discussing the parameters, it is important to understand the difference between a Binary Quadratic Model (BQM) and a Constrained Quadratic Model (CQM):

BQM: The binary quadratic model is designed to encode Ising and QUBO problems representing binary decision problems (yes or no question). BQMs consist of binary variables with linear or quadratic interactions but do not support explicit constraints. Any constraints must be encoded as penalties within the model.

CQM: The constrained quadratic model extends BQM by explicitly incorporating constraints and supporting integer variables, making it a more versatile framework for handling constrained problems. CQMs can also be converted into BQMs by applying penalties, although converting them into BQMs significantly increases the number of binary variables when they use integer variables or constraints with slack variables. The *DWaveAdapter* uses CQM formulations, which are then provided to the solver.

As shown in Table 4.2, the parameter ‘sampler‘ allows us to select a sampler supported by Ocean. It is important to note that some samplers only support BQMs, while others support CQMs (e.g., *ExactSolver* and *ExactCQMSolver*). Consequently, although the initial model is a CQM, if a BQM sampler is passed to the solver, the model is converted into a BQM. After the algorithm runs, an inverter function converts the BQM samples back into the CQM format, excluding slack variables and re-adding integer variables.

Samplers can be classical, quantum or even hybrid (e.g. *SimulatedAnnealingSampler*, *DWaveSampler* and *LeapHybridCQMSampler*). While classical samplers are meant for testing purposes, the quantum and hybrid ones have distinct applications. While the quantum optimizers only support BQM models, are great for research and yield reasonable solutions up to some number of qubits, the hybrid solvers are more powerful, supporting CQM models and bringing the best of the classical and quantum optimizers to retrieve great solutions capable of competing with state-of-the-art classical approaches.

Table 4.2: Parameters available in the D-Wave Quantum Annealing solver.

Parameter	**Type**	**Description**	**Default**
simplify	boolean	Whether to simplify the problem by removing redundant variables.	True
track_progress	boolean	Whether to track progress by printing logs during the annealing.	True
sampler	Sampler	The sampler for the annealing. Can be a classical or quantum sampler.	ExactCQMSolver
embedding	Embedding	Embedding class used to embed the problem into the quantum hardware.	EmbeddingComposite
embed_bqm	boolean	If using a BQM sampler, locally embeds the problem before sampling.	True
num_reads	int	If using a BQM sampler, sets a fixed number of reads.	null
time_limit	int	Optionally sets a time limit for the annealing process, in seconds. If not specified, the sampler usually has its own default value.	null
embedding_timeout	int	Optionally sets a time limit for the embedding process, in seconds. If not specified, the sampler usually has the default value of 1 000 seconds.	null

For the ‘embedding‘ parameter, it is necessary only for BQM quantum-only samplers, as hybrid samplers handle this process automatically. When using a quantum annealer, the problem must be embedded before submission. In contrast, the BQM hybrid sampler (*LeapHybridSampler*) does not require embedding, which can be controlled by the ‘embed_bqm‘ parameter. Various types of embeddings are available, including custom options, but the *EmbeddingComposite* is the most versatile, suitable for most problems.

Another parameter specific to quantum-only samplers is ‘num_reads‘, which indicates the number of times the annealing process is performed, equating to the number of samples in the result. Hybrid samplers do not require this parameter, as they automatically determine the number of samples needed for near-optimal results, reducing manual intervention.

At the end of the annealing process, unfeasible solutions are filtered out, and the solution with the lowest energy (i.e. the least cost) is selected. If BQM conversion was used, it is inverted, and the result is converted into a *VRPSolution*.

After submitting the problem, it can be monitored on the D-Wave dashboard, shown in Figure 4.3. This dashboard provides insights into the problem’s status, parameters, timings, samples, and usage.

4.4 Classical Implementation

As mentioned in Section 2.2.3, Google OR-Tools is a state-of-the-art tool for solving combinatorial optimization tasks, including routing problems. Given its proven capabilities and results, this dissertation uses OR-Tools as the classical benchmark to compare the developed algorithms regarding efficiency and solution quality.

While this implementation also addresses all the problems formulated in Chapter 3, it operates independently of the QUBO models. This design is beneficial because it allows the classical implementation to test and validate the results obtained from the formulations. OR-Tools automatically determines many parameters used to solve the problems, but there are a few that this *ClassicalSolver* implements, shown in Table 4.3.

4.5 Solution Management and Visualization

The final component in this architecture is the *VRPSolution*. As described in Section 4.1, this class encompasses all the information needed to reconstruct a problem’s solution. It includes the initial problem input necessary to recreate the scenario, the path for each vehicle in the solution, respective distances, loads at each location, and timing statistics (e.g., optimization time, QPU usage, local execution time).

An essential feature of this component is its ability to display the solution in an interactive and visually appealing manner. While it is possible to print the information to the console, the optimal experience is achieved using the display feature. This functionality leverages the Plotly library to represent the solution in a high-quality, interactive graph.

Table 4.3: Parameters available in the Classical OR-Tools solver.

Parameter	**Type**	**Description**	**Default**
track_progress	boolean	Whether to track progress by printing logs every iteration of the algorithm.	True
solution_strategy	FirstSolution (enum)	First solution strategy. The method used to find an initial solution.	Automatic
local_search_metaheuristic	LocalSearch (enum)	Local search strategy (metaheuristic) used by the solver.	Automatic
distance_global_coefficient	int	The coefficient is multiplied by each vehicle's travelled distance and is helpful to distribute distance between routes.	1
time_limit_seconds	int	Maximum execution time, in seconds	10

In addition to displaying each vehicle's path in different colours on a graph, the generated webpage offers various interactive features. Zooming, scaling, exporting the image, and hovering over nodes to reveal load information is possible. We can also remove and re-add routes for specific visualization, check the solution's total distance or time and specify whether distance or time should be used as the cost unit (metric system). Location names (e.g., bus stops) can be optionally displayed. An example plot is shown in Figure 4.4.

Another crucial function of this class is to save the result for later analysis or visualization. While directly storing the HTML page is an option, it consumes unnecessary storage space and is useless for further result processing. Therefore, the solution can be converted to a JSON file by dumping the object's fields, which can later be loaded to recover the solution. A file example of a simple CVRP with one vehicle and two locations is shown below.


```
{
    "num_vehicles": 1,
    "locations": [[456, 320],[228, 0]],
    "objective": 1096,
    "total_distance": 1096,
    "routes": [[0, 1, 0]],
    "distances": [1096],
    "loads": [[1, 2, 2]],
    "depot": 0,
    "run_time": 636,
    "qpu_access_time": null,
    "local_run_time": 3115,
```

```
    "location_names": ["0", "1"],
    "distance_unit": "METERS",
    "use_depot": true,
    "capacities": 2,
    "use_capacity": true
}
```

4.6 Overview

This chapter presented the software architecture and implementation of quantum algorithms designed to solve various Vehicle Routing Problem (VRP) formulations. The extensible framework, developed entirely in Python, supports multiple algorithms, including the Quantum Approximate Optimization Algorithm (QAOA), Quantum Annealing (QA) and classical solvers for comparison.

The framework's design allows easy integration of new QUBO formulations and quantum computing providers, ensuring adaptability and future-proofing for developments in quantum routing algorithms. By creating a flexible and extensible software framework, this chapter lays the groundwork for effectively deploying quantum algorithms to tackle complex routing problems.

The results and discussion of the implemented algorithms are presented in the following Chapter 5, where their performance and efficiency are analyzed using benchmarks. Furthermore, the applicability of these algorithms to the real-world scenario of Porto is examined in Chapter 6, using actual data from the city's urban transport system.

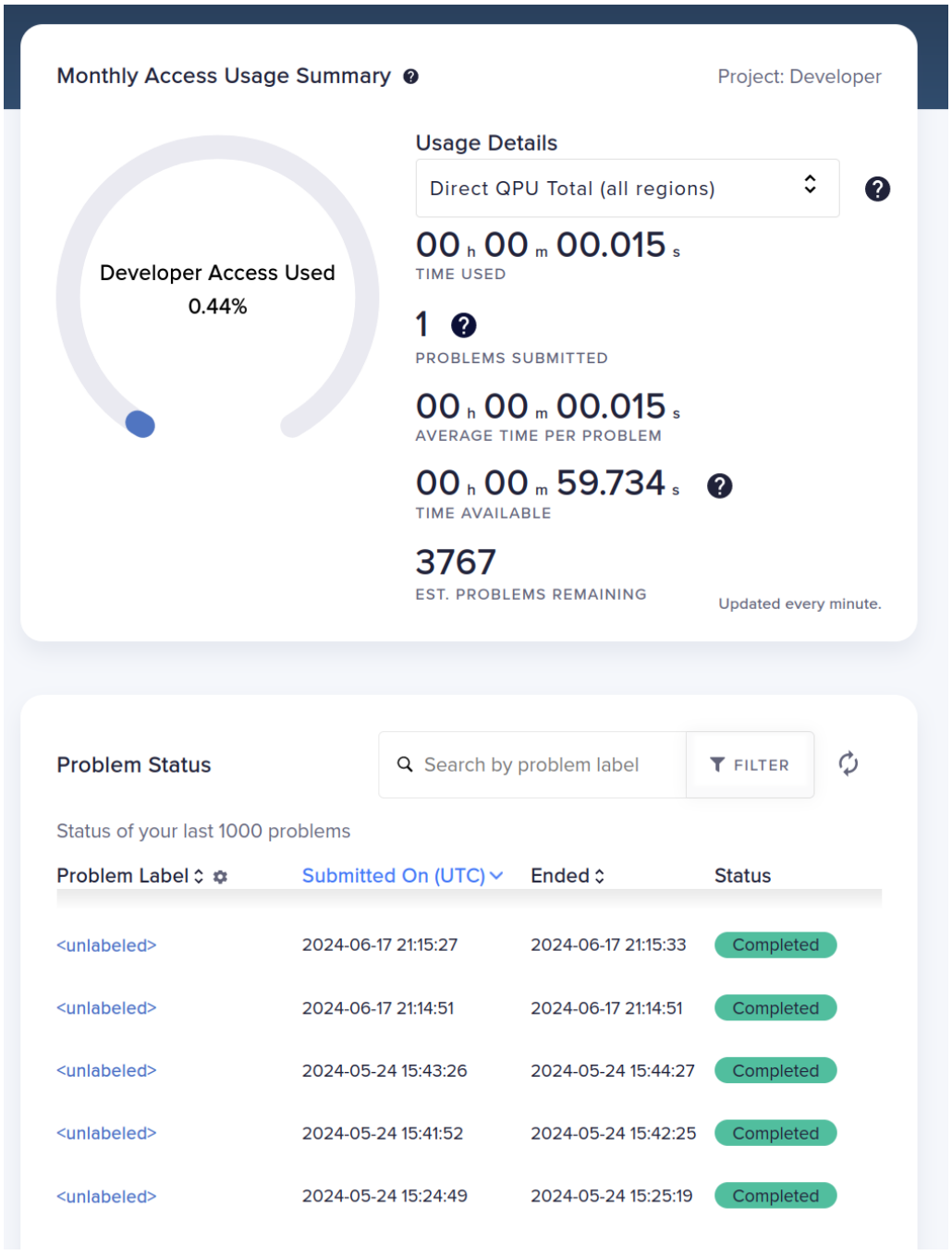

Figure 4.3: Screenshot of the D-Wave Leap Dashboard after submitting a few problems.

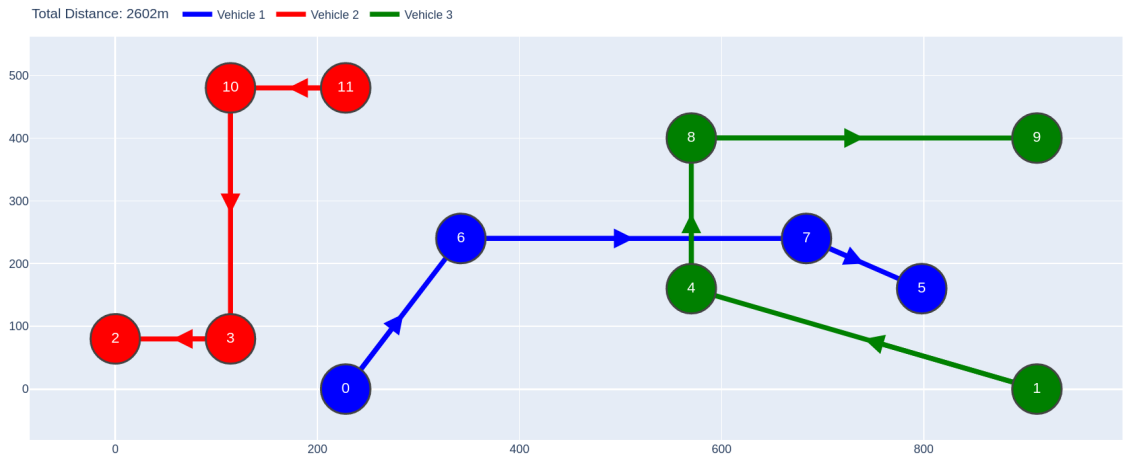

Figure 4.4: Example of an RPP solution visualization using the Plotly library.

Chapter 5

Results and Discussion

This chapter presents the results obtained using the QUBO formulations and algorithms implemented in Chapters 3 and 4, respectively. To effectively demonstrate the capability of these algorithms to produce quality solutions, we use benchmark instances from datasets constructed by the community over the years [93, 79, 58, 91]. These benchmarks allow us to compare our results to best-known solutions (BKS) and compute the optimal gap to evaluate the effectiveness of the different algorithms and solvers [1]. The benchmarks used in this dissertation are listed and labelled in Table 5.1, with all instances assuming using the Euclidean distance as the cost function. The best-known solutions presented are all optimal, extracted from the datasets Augerat (Set A), Augerat (Set P), and Christofides & Eilon (Set E) [16, 79, 58].

Table 5.1: Benchmark instances displaying best-known solutions for the CVRP problem. Here, n represents the number of locations, K denotes the number of vehicles, Q indicates the maximum capacity per vehicle, and *tightness* is defined as the ratio of total demand across nodes to the overall vehicle capacity $\left(\frac{demand}{QK}\right)$.

Instance	n	K	Q	Tightness	BKS
P-n16-k8	16	8	35	0.88	450
P-n19-k2	19	2	160	0.97	212
A-n32-k5	32	5	100	0.82	784
A-n53-k7	53	7	100	0.95	1010
E-n101-k14	101	14	112	0.93	1067

However, significant issues exist with the VRP benchmark instances available in the community:

- While these datasets include data for various routing problem variations, only the CVRP variation is helpful in the context of this dissertation. This limitation arises because all instances have the same finite capacity for all vehicles, thus excluding the VRP and Multi-CVRP variations.
- The RPP is a less popular variation, so no benchmark datasets are available.

- The size of the problem inputs poses a challenge. Since the benchmarks were computed using classical means, the limitation on the number of variables (which are translated to qubits) did not exist, resulting in many instances having too many locations for this use case.

We complement the benchmark data with custom instances to address these issues, using Google OR-Tools as the state-of-the-art comparison, as explained in Section 4.4. The classical computer used in all experiments has an AMD Ryzen 7 7840HS processor, 32GB of RAM, and runs on the Linux Mint 21.2 operating system. The cost function used in all instances is the Manhattan distance. The custom benchmarks are listed and labelled in Table 5.2. Note that *tightness* might exceed 1 in the case of RPP since customers are also dropped off.

Table 5.2: Custom benchmark instances for VRP, Multi-CVRP, and RPP problems. Here, n denotes the number of locations, K denotes the number of vehicles, Q denotes the vehicles' maximum capacity, and *tightness* represents the ratio between the total demand of all nodes and the overall vehicle capacity $\left(\frac{\text{demand}}{QK}\right)$.

Instance	**Problem**	**n**	**K**	**Q**	**Trips**	**Tightness**
vrp-n2-k1	VRP	2	1	-	-	-
vrp-n5-k2	VRP	5	2	-	-	-
mcvrp-n8-k2	Multi-CVRP	8	2	[10, 8]	-	1.0
mcvrp-n16-k2	Multi-CVRP	16	2	[10, 8]	-	1.0
rpp-n2-k1	RPP	2	1	-	1	-
rpp-n4-k2	RPP	4	2	-	2	-
rpp-n8-k2	RPP	8	2	[12, 10]	5	0.91
rpp-n14-k4	RPP	14	4	8	7	1.13
rpp-n16-k2	RPP	16	2	-	8	-
rpp-n26-k6	RPP	26	6	15	13	0.72

In Section 5.1, we present the results obtained with OR-Tools, comparing them to the community benchmark instances and showing the new custom benchmarks generated with the tool. Sections 5.2 and 5.3 explore the results of the QAOA and Quantum Annealing algorithms, respectively, highlighting their capabilities and limitations with different parameters, simulators, and actual quantum or hybrid computations. Finally, in Section 5.4, we discuss the overall results of the benchmarks, showcasing the best solvers of each class from the previous sections.

5.1 OR-Tools Results

As detailed in Section 4.4, Google OR-Tools is a leading tool for solving combinatorial tasks such as vehicle routing problems, serving as a comparison for the algorithms developed in this dissertation. We present the results of employing OR-Tools on the earlier benchmark instances from the best-known solution (BKS) datasets, as displayed in Table 5.3. The optimal gap represents the percentage difference between the new and optimal BKS solutions and is calculated following the formula $\frac{solution}{BKS} - 1$.

Table 5.3: Results obtained from running the OR-Tools implementation to solve benchmark instances from the BKS datasets.

Instance	BKS	OR-Tools	Exec. Time (ms)	Optimal Gap
P-n16-k8	450	450	8.40	0.00%
P-n19-k2	212	212	1 000	0.00%
A-n32-k5	784	784	28.0	0.00%
A-n53-k7	1010	1010	7 000	0.00%
E-n101-k14	1067	1103	58 000	3.37%

Upon examining the low values of the optimal gap for each benchmark instance, most of which are 0, it is evident that the OR-Tools classical implementation is an excellent approximation of the best-known solutions for problems of the scale considered in this study. These results also suggest the implementation is a suitable benchmark for the custom instances defined in Table 5.2. The results obtained from OR-Tools for the custom benchmarks are presented in Table 5.4.

Table 5.4: Results obtained from running the OR-Tools implementation to solve the custom benchmark instances.

Instance	OR-Tools	Exec. Time (ms)
vrp-n2-k1	30	0.868
vrp-n5-k2	214	1.41
mcvrp-n8-k2	352	4.69
mcvrp-n16-k2	458	11.0
rpp-n2-k1	15	0.899
rpp-n4-k2	138	1.69
rpp-n8-k2	306	8.59
rpp-n14-k4	387	15.5
rpp-n16-k2	394	52.9
rpp-n26-k6	586	2 000

For illustrative purposes, example plots for the `A-n32-k5` and `rpp-n26-k6` benchmark results are shown in Figures 5.1 and 5.2, respectively.

5.2 QAOA Results

In this section, we present the results of implementing the QUBO formulations detailed in Chapter 3, specifically related to the Quantum Approximate Optimization Algorithm (QAOA) as described in Section 4.2. In addition to comparing the performance and efficiency of the algorithm, we also examine the impact of modifying specific QAOA parameters, such as using a warm start.

First, we validated and tested the QUBO formulations by running the benchmarks with the classical CPLEX optimizer, as described in 5.2.1. Next, we analyzed the results of the smallest instances using a simulator, detailed in 5.2.2. Finally, we tried executing the benchmarks on a real IBM quantum computer, as presented in 5.2.3.

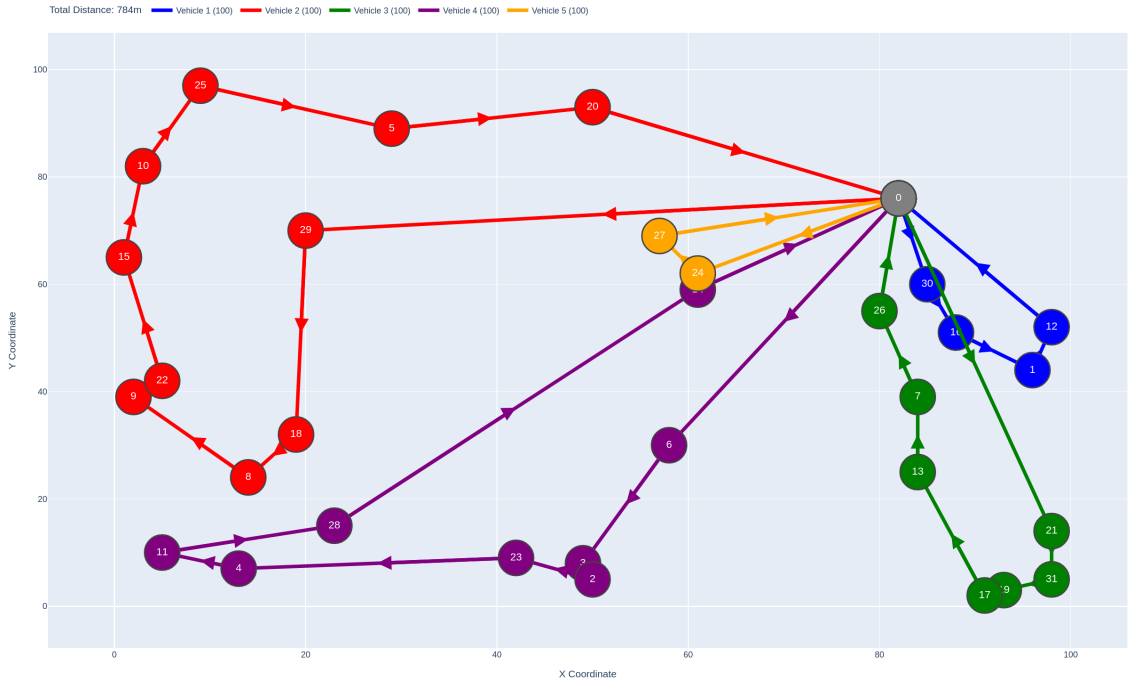

Figure 5.1: Graphical result for the `A-n32-k5` instance obtained with the Google OR-Tools implementation.

5.2.1 CPLEX Results

As mentioned in Section 4.2, the IBM CPLEX optimizer can solve problems formulated as QUBOs. This capability allows us to validate and test our models before running them on a quantum computer, which is limited in problem size and consumes more resources. Additionally, it is helpful to compare the performance of these formulations with the best-known solutions or the solutions obtained in the previous section using OR-Tools. However, the most significant limitation of this optimizer is the restriction to a maximum of 1000 variables in the open version of CPLEX. Consequently, the more considerable benchmark instances could not be executed.

The results for the benchmark instances from the BKS datasets, `P-n16-k8` and `P-n19-k2`, are presented in Table 5.5, including the number of variables needed. The other BKS instances required more than 1000 variables, making it impossible to obtain their results. The outcomes for the custom benchmark instances, except for the sizeable `rpp-n26-k6` instance, are shown in Table 5.6. This time, the solution gap represents the percentage difference between the CPLEX and OR-Tools solutions and is calculated using the formula $\frac{solution}{ORTools} - 1$.

Table 5.5: Results obtained from running the CPLEX optimizer to solve benchmark instances from the BKS datasets.

Instance	BKS	CPLEX	Variables	Exec. Time (s)	Optimal Gap
P-n16-k8	450	450	255	0.167	0.00%
P-n19-k2	212	212	360	4.83	0.00%

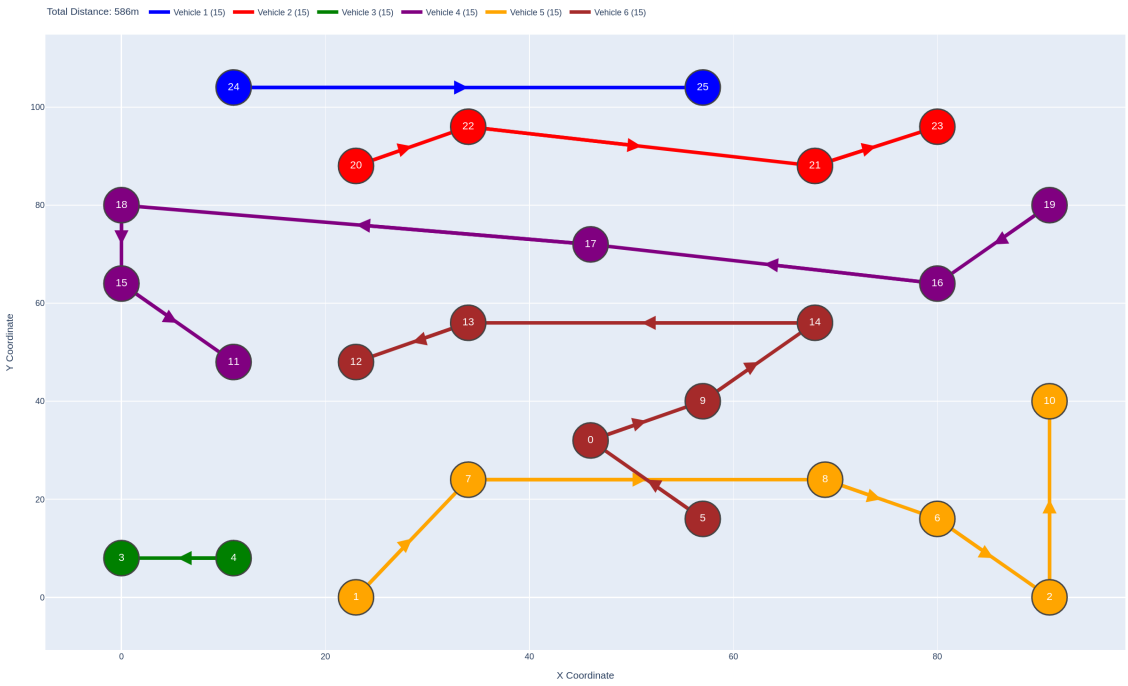

Figure 5.2: Graphical result for the `rpp-n26-k6` instance obtained with the Google OR-Tools implementation.

Although CPLEX can still obtain near-optimal solutions in a timely manner, it is considerably slower and consumes more resources than OR-Tools. Many of the instances executed were able to match the best-known solution (either from the dataset or OR-Tools). However, as problem size increased, CPLEX tended to use significantly more time and memory, sometimes resulting in timeouts or crashes. An interesting case was observed with the instance `rpp-n8-k2`, where CPLEX found a better solution than the previous solver, albeit with a higher execution time. That is why the solution gap for this instance has a negative value.

These results also validate the QUBO models formulated in Chapter 3, demonstrating their effectiveness for the respective variations and ability to provide suitable solutions for routing problems. An example of the rpp-n8-k2 instance can be seen in Figure 5.3.

5.2.2 Qiskit Simulations

Qiskit provides simulators for its developers to test and analyze quantum algorithms without submitting them to a real quantum computer. These simulators allow us to observe the results obtained by an ideal (or, if preferred, noisy) quantum computer. This is highly beneficial as it enables us to test algorithms and formulations without consuming limited resources or waiting in long queues to use the actual machines.

Despite their usefulness, simulators have a significant drawback: they are computationally demanding, with execution times increasing exponentially with the number of qubits. Furthermore, they are limited to a few qubits, meaning only tiny problems can be executed on these simulators.

Table 5.6: Results obtained from running the CPLEX optimizer with the custom benchmark instances. Negative gaps represent improvements.

Instance	OR-Tools	CPLEX	Variables	Exec. Time (s)	Solution Gap
vrp-n2-k1	30	30	3	0.0031	0.00%
vrp-n5-k2	214	214	24	0.0053	0.00%
mcvrp-n8-k2	352	352	112	1.52	0.00%
mcvrp-n16-k2	458	506	480	33.9	10.5%
rpp-n2-k1	15	15	3	0.0031	0.00%
rpp-n4-k2	138	138	30	0.0018	0.00%
rpp-n8-k2	306	297	160	26.1	-2.94%
rpp-n14-k4	387	484	780	451.2	25.1%
rpp-n16-k2	394	597	510	492.1	51.5%

Since the BKS dataset instances are too large, we present the results of the smallest benchmark instances from the custom benchmarks in Table 5.7 and Table 5.8, without and with a warm start, respectively. These instances were assessed using a Sampler simulator, which employs probabilistic methods to compute results, in combination with the Cobyla classical optimizer. For the warm start version of the algorithm, the pre-solver used was CplexOptimizer. This setup constitutes a simulated QAOA framework.

Table 5.7: Results obtained from running the Qiskit simulator with the smallest custom benchmark instances, without warm start.

Instance	OR-Tools	Simulator	Qubits	Exec. Time (s)	Solution Gap
vrp-n2-k1	30	30	3	2.72	0.00%
rpp-n2-k1	15	15	4	0.294	0.00%

Table 5.8: Results obtained from running the Qiskit simulator with the smallest custom benchmark instances, with warm start.

Instance	OR-Tools	Simulator	Qubits	Exec. Time (s)	Solution Gap
vrp-n2-k1	30	30	3	1.00	0.00%
rpp-n2-k1	15	15	4	0.176	0.00%

Only two benchmark instances could be executed with the simulator, as the others exceeded its qubit capacity. For example, the `vrp-n5-k2` instance would require 80 qubits after adding slack variables and encoding integer variables. Additionally, while the `rpp-n4-k2` problem requires only 32 qubits, which is supported by the simulator, it attempts to allocate 64GiB of memory, which is not doable in the current experimental environment.

Despite being limited to two benchmark instances, the simulator demonstrated that the formulations were effective and produced correct solutions within the QAOA framework. It is also possible to see a difference when using a warm start, since the execution times were consistently lower.

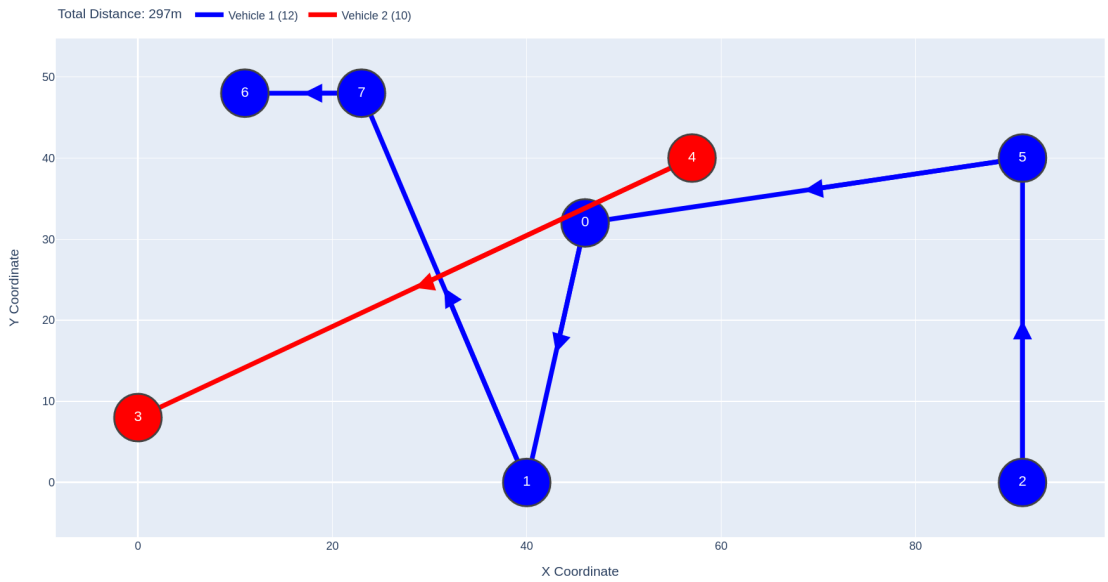

Figure 5.3: Graphical result for the `rpp-n8-k2` instance obtained with the CPLEX classical solver.

5.2.3 IBM Quantum Results

The next step involved conducting QAOA benchmark experiments on real quantum computers. For this dissertation, we used IBM's latest model available to developers, the IBM Eagle, featuring a maximum of 127 qubits. Detailed information regarding IBM's superconducting hardware and their quantum cloud platform can be found in Section 2.1.4.1. Due to the qubit limitation, not all benchmark instances can be executed, as some exceeded this threshold significantly. The experimental parameters are aligned with those used in the simulations discussed in Section 5.2.2.

Having said that, the performance of IBM's quantum computers did not meet expectations, as no benchmark instance could be solved using QAOA on a real IBM machine. The main issues encountered were:

- Running complex circuits on current quantum computers involves substantial overhead due to waveform generation, qubit resets, state preparation, and especially error mitigation [78]. We attempted to reduce this overhead by disabling error mitigation, thus setting the resilience level of the sampler to 0. Although this significantly reduced the overhead, each iteration of QAOA for the smallest instances still took around 3 seconds (down from approximately 20 seconds). This is too long for an algorithm that typically requires hundreds of iterations. A few instances could be run on the Qiskit simulator, as this overhead does not apply.

- The queue waiting time for quantum computers is generally too high, often extending up to a few hours. This can be mitigated using sessions, as explained in Section 4.2.1. However, the priority given by a session does not persist for the entire duration required by QAOA.

After a while, the session re-enters the queue and eventually times out, wasting the resources expended previously.

- The free open plan provided by IBM is limited to 10 minutes of execution per month, significantly limiting the experiments. We attempted to obtain extra credits for the research project but were unsuccessful.
- The QAOA algorithm employs a sampler that runs the same circuit 4000 times per iteration. While this slightly increases the execution time, it is negligible compared to the overhead. Adjusting the number of shots in the sampler does not enhance the algorithm's execution feasibility.

Therefore, instead of providing the usual tables with execution results, we present Table 5.9, which shows each iteration's approximate time for the respective benchmark instances. This at least gives an idea of how the algorithm scales as the number of qubits used increases. The missing instances were too large to be solved on a 127-qubit computer, including all instances from the BKS datasets.

Table 5.9: Number of qubits used and execution time per QAOA iteration on IBM Eagle.

Instance	**Qubits**	**Time p/ Iteration (s)**
vrp-n2-k1	3	3
vrp-n5-k2	80	7
mcvrp-n8-k2	120	14
rpp-n2-k1	4	3
rpp-n4-k2	32	4

As illustrated in Table 5.9, the execution time per iteration increases steadily with the number of qubits. This trend is expected since state preparation takes longer as more qubits are used. This increase in execution time is due to overhead rather than circuit execution because reducing the number of shots does not significantly improve the execution time.

5.3 Quantum Annealing Results

In this section, we present the results of implementing the QUBO formulations detailed in Chapter 3, specifically related to the Quantum Annealing (QA) implementation described in Section 4.3. Besides comparing the performance and efficiency of the algorithm, we also examine the impact of using different samplers and parameters.

Firstly, the implementation was validated and tested by running benchmarks with a classical solver, as detailed in 5.3.1. The next step involved executing them solely on a quantum annealer, as described in 5.3.2, where we analyze the effect of varying the number of reads parameter. Subsequently, we analyzed the results using hybrid samplers, which combine classical and quantum optimization techniques. In 5.3.3, we present the results using a Constrained Quadratic Model

(CQM) formulation. In contrast, the results after converting to a Binary Quadratic Model (BQM) formulation are detailed in 5.3.4.

5.3.1 Classical Samplers

Similar to the approach with Qiskit simulators, D-Wave offers tools within their Ocean framework to test formulations for quantum annealers before submitting them to a real quantum machine. This is highly beneficial as it allows developers to test algorithms without consuming limited resources. Instead of simulators, D-Wave offers classical solvers or samplers that deliver results comparable to a quantum annealer's.

One of the classical solvers provided is the ExactCQMSolver. This sampler, intended for testing and debugging, calculates the energy of every possible combination of variable assignments [47], ensuring the best solution is always found. However, it is computationally intensive and becomes slow as the problem size increases. This solver supports problems formulated as a CQM, which is advantageous as it avoids conversions to a BQM formulation (the two models' differences are explained in Section 4.3).

Another classical sampler provided is the SimulatedAnnealingSampler. Although also used for testing and debugging, this sampler employs a classical implementation of simulated annealing, which is less computationally intensive than the exact solver but does not always find the optimal solution. The 'num_reads' parameter must be defined to determine the number of samples done. Increasing the number of reads raises both the execution time and the likelihood of finding the best solutions. Additionally, this sampler only supports BQM formulations, so the problems need to be converted, which increases the number of variables as they are now all binary.

Due to the large size of the BKS dataset instances, the results of the smallest benchmark instances from the custom benchmarks are presented in Table 5.10 and Table 5.11, for the exact solver and simulated annealing sampler, respectively. For the exact solver, the best result from the final sample set in each experiment always equals the optimal solution.

Table 5.10: Results obtained from running Quantum Annealing with the exact CQM solver on the custom benchmark instances.

Instance	OR-Tools	Exact Solver	Variables	Exec. Time (ms)	Solution Gap
vrp-n2-k1	30	30	3	0.60	0.00%
rpp-n2-k1	15	15	3	0.53	0.00%

Table 5.11: Results obtained from running D-Wave's simulated annealing sampler on the custom benchmark instances.

Instance	OR-Tools	SA	Variables	Reads	Exec. Time (ms)	Solution Gap
vrp-n2-k1	30	30	5	1	4.94	0.00%
vrp-n5-k2	214	214	92	10	3 674	0.00%
rpp-n2-k1	15	15	3	2	7.22	0.00%
rpp-n4-k2	138	138	32	10	869	0.00%

The results illustrate the slow scalability of these classical samplers with the number of variables in the problems, particularly for the exact solver. This is expected, as they are intended primarily for testing purposes. The exact solver becomes infeasible and times out when approaching just 30 variables. Simulated annealing, on the other hand, is hindered by the conversion to BQM, which significantly increases the number of binary variables. Despite these limitations, the experiments demonstrate that the Quantum Annealing formulation can correctly solve routing problems, even though only a few small instances were executed at this stage.

5.3.2 Quantum Sampler

After validating the implementation with classical samplers, the next phase was to analyze the results using a real quantum annealer. Detailed in Section 2.1.4.2, the hardware used was D-Wave Advantage, the latest model developed by D-Wave openly available to developers. This quantum annealer supports more than 5000 qubits per problem (the exact number depends on the operational qubits), which should be enough to cover most of our benchmark instances in terms of qubit capacity.

The solver used for this section was the DWaveSampler, a quantum sampler designed for the BQM model. Therefore, our formulations had to be converted from CQM to BQM, as explained in Section 4.3. This conversion impacts the number of qubits used in each problem by introducing more binary variables. Additionally, this sampler requires manually setting the ‘num_reads‘ parameter, which specifies how many samples are retrieved from the quantum annealer (each sample represents one Quantum Annealing execution). The final sample set is examined, and the sample with the lowest energy is selected as the final solution. The ‘num_reads‘ parameter was increased for more complex problems, resulting in longer execution times. Without this increase, it would be unlikely to obtain suitable solutions. To better understand the impact of this parameter, Figure 5.4 illustrates the relationship between the parameter value and the frequency of getting the BKS solution. The results were obtained by running the instance `vrp-n5-k2` 10 times for each parameter value.

A significant bottleneck of this sampler is the fixed maximum execution time of 1 second. Consequently, the number of samples cannot be arbitrarily set, as it directly impacts the annealer's execution time. Instances where this time limit was reached without finding feasible solutions are marked as *NF*. A solution is only considered feasible if it satisfies all constraints.

Before submitting the problem to the quantum annealer, an important step is embedding the problem to run on the hardware, as described in Section 2.1.4.2. The embedding process used for the results was the EmbeddingComposite. This local process is time-consuming, so the embedding time is included as an extra column in the results. Additionally, we set a timeout value for the embedding of 5 minutes.

For the benchmarks from the best-known solutions datasets, the experiments could not embed any of those problems locally. Therefore, Table 5.12 presents the number of qubits required to solve these problems after conversion to the BQM format, assuming successful embedding. Even

Table 5.12: Number of qubits required for each benchmark instance from the BKS dataset when using the quantum annealer after conversion to a BQM formulation.

Instance	No. of Qubits
P-n16-k8	1 672
P-n19-k2	3 384
A-n32-k5	8 866
A-n53-k7	24 700
E-n101-k14	90 700

if embedding were possible, only the `P-n16-k8` and `P-n19-k2` instances would not exceed the quantum annealer's qubit capacity.

In contrast, some custom benchmark instances were successfully solved and are detailed in Table 5.13. Problems with up to 120 qubits were locally embedded, whereas those with 294 or more timed out. A notable case involved the instance `mcvrp-n8-k2`, where embedding was successful, but the execution time of 1 second was insufficient to find a feasible solution, with the maximum number of reads attempted being 3500.

The inefficiency of the quantum annealer in producing practical solutions can be attributed to some key factors:

- Embedding the problem locally, such as on a personal computer, poses significant limitations. This results, in many instances, timing out during the embedding process. Enhanced capabilities in embedding, possibly through more powerful hardware, would be highly beneficial.
- The current state of the quantum hardware causes the annealing process to deviate substantially from the idealized adiabatic process. Consequently, obtaining the optimal or near-optimal solution in each execution remains challenging. While increasing the number of reads helps mitigate this issue, more is needed to tackle complex problems.
- While the number of qubits remains a limiting factor, it is much less restrictive than with IBM's quantum computers.

Currently, D-Wave's approach to mitigating these challenges involves using hybrid samplers, which will be elaborated upon in the following sections.

5.3.3 Hybrid CQM Sampler

D-Wave's hybrid samplers combine cutting-edge classical algorithms with dynamic utilization of the quantum annealer, focusing its computational power where it can offer the most significant benefit. These samplers are specifically engineered to tackle large problem instances, allowing developers and researchers to achieve competitive solutions while advancing formulations for Quantum Annealing [47].

Table 5.13: Results obtained from running D-Wave's Quantum Annealing sampler on the custom benchmark instances.

Instance	OR	QA	Qubits	Samples	Ex. Time (ms)	Embedding (ms)	Gap
vrp-n2-k1	30	30	5	1	15.9	0.025	0.00%
vrp-n5-k2	214	214	92	500	88.6	11.8	0.00%
mcvrp-n8-k2	352	*NF*	120	3500	945.6	47.4	*NF*
mcvrp-n16-k2	458	-	488	-	-	*timeout*	-
rpp-n2-k1	15	15	3	1	15.9	0.019	0.00%
rpp-n4-k2	138	138	32	100	27.0	0.318	0.00%
rpp-n8-k2	306	-	294	-	-	*timeout*	-
rpp-n14-k4	387	-	1228	-	-	*timeout*	-
rpp-n16-k2	394	-	512	-	-	*timeout*	-
rpp-n26-k6	586	-	5574	-	-	*timeout*	-

This section presents the results obtained using the LeapHybridCQMSampler, a robust hybrid solver designed for constrained quadratic models (CQM), which includes the routing problems formulated in this study. In addition to automatically allocating portions of the problem to the quantum annealer, this solver handles problem embedding independently and automatically manages the number of reads, eliminating the need for manual configuration.

The primary limitation of this sampler lies in the predetermined time limit set before submitting the problem, which has a minimum duration of 5 seconds. Despite being defined as a limit, the solver mostly uses the entire allotted time, requiring careful consideration of resource allocation, especially under the constraints of the open developer plan.

The results for the best-known solutions and custom benchmarks are detailed in Tables 5.14 and 5.15, respectively. We report the number of variables instead of qubits to clarify that not all variables may be encoded on the quantum annealer. Instances yielding infeasible solutions are denoted as *NF*, as previously described. Additionally, an extra column shows the number of samples/reads performed by the solver.

Table 5.14: Results obtained from running D-Wave's hybrid CQM sampler on the BKS benchmark instances.

Instance	BKS	QA	Vars	Samples	Exec. Time (s)	QPU Time (ms)	Gap
P-n16-k8	450	450	255	81	5.00	16.1	0.00%
P-n19-k2	212	318	360	126	60.04	31.7	50.0%
A-n32-k5	784	895	1 023	131	60.51	32.0	14.2%
A-n53-k7	1010	2939	2 808	110	120.38	47.9	191.0%
E-n101-k14	1067	3573	10 200	125	120.39	47.8	234.9%

The results obtained with the hybrid sampler represent a significant improvement over those obtained using the quantum annealer alone. Firstly, with the embedding process conducted remotely, all problems from the custom and BKS benchmarks were successfully executed on the solver. Additionally, only one instance, the `rpp-n26-k6`, failed to produce a feasible solution

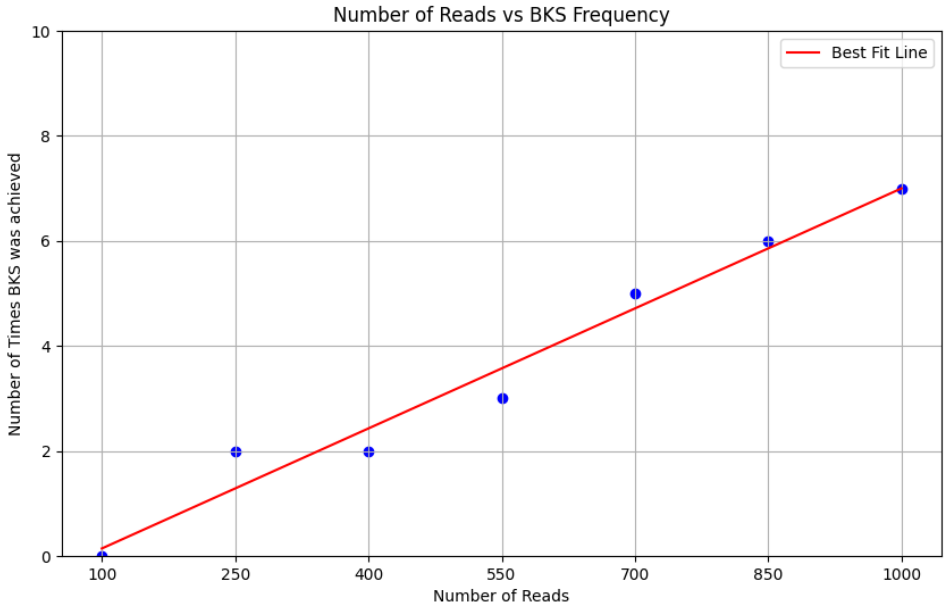

Figure 5.4: Plot with linear regression depicting the influence of the number of reads in the quantum annealer on the frequency of achieving the BKS solution for the instance `vrp-n5-k2`.

after reaching the maximum execution time of 2 minutes. The custom benchmarks achieved excellent solutions, all matching or surpassing those calculated using Google OR-Tools. Regarding the BKS dataset benchmarks, the solutions for problems requiring multiple thousands of qubits exhibited high gaps. They were far from optimal, but these results are still impressive, given the enormous size of the inputs. For a visual comparison, the plots for the `A-n32-k5` and `rpp-n16-k2` results are shown in Figures 5.5 and 5.6, respectively.

An analysis of D-Wave's intelligent resource allocation can be conducted by examining several metrics related to the quantum annealer. For instance, the number of samples remains relatively constant as the input size increases, which is unexpected considering the previous approach with the quantum solver, where the number of samples was increased to counteract rising errors. This stability is likely due to the hybrid solver running only portions of the problem on the quantum annealer while utilizing classical methods for the remainder. Another notable metric is the QPU usage time. Although it is well-controlled and typically within a few tens of milliseconds, there is a noticeable increase in usage as the number of variables grows. This indicates that the quantum annealer is employed more intensively as the input size increases. Another illustrative case is the `E-n101-k14` instance, where the number of variables exceeds the available qubits, yet a valid solution involving QPU usage is still obtained, demonstrating effective problem partitioning.

5.3.4 Hybrid BQM Sampler

In the previous section, we analyzed the results obtained with D-Wave's hybrid CQM sampler. Although it is the most suitable solver for the problems at hand, other hybrid options exist. An

Table 5.15: Results obtained from running D-Wave's hybrid CQM sampler on the custom benchmark instances. Negative gaps represent improvements.

Instance	OR	QA	Vars	Samples	Exec. Time (s)	QPU Time (ms)	Gap
vrp-n2-k1	30	30	3	97	4.30	15.8	0.00%
vrp-n5-k2	214	214	24	127	5.08	31.8	0.00%
mcvrp-n8-k2	352	352	112	127	5.05	32.0	0.00%
mcvrp-n16-k2	458	458	480	126	5.07	31.9	0.00%
rpp-n2-k1	15	15	3	1	5.6×10^{-5}	0.0	0.00%
rpp-n4-k2	138	138	30	127	5.06	31.8	0.00%
rpp-n8-k2	306	297	160	126	5.34	32.0	-2.94%
rpp-n14-k4	387	387	780	122	10.20	32.1	0.00%
rpp-n16-k2	394	383	510	126	5.06	31.8	-2.79%
rpp-n26-k6	586	*NF*	4 050	134	119.4	48.1	*NF*

example is the hybrid BQM sampler, examined in this section. As explained earlier (refer to 2.1.4.2), our models use CQM formulations, and converting to a BQM significantly increases the number of variables required. Therefore, we expect to obtain worse results with the BQM sampler. Nonetheless, this analysis is conducted to understand the impact of this conversion, which is similar to the conversion performed for the direct quantum annealer approach.

In this case, we must also set a maximum time limit when submitting a problem. Unlike before, the minimum value for this parameter is 3 seconds instead of 5. The results for the best-known solutions and custom benchmarks are detailed in Tables 5.16 and 5.17, respectively. Infeasible solutions are once again marked with *NF*.

Table 5.16: Results obtained from running D-Wave's hybrid BQM sampler on the BKS benchmark instances.

Instance	BKS	QA	Vars	Samples	Exec. Time (s)	QPU Time (ms)	Gap
P-n16-k8	450	*NF*	1 672	1	60.0	1 324	*NF*
P-n19-k2	212	*NF*	3 384	1	60.0	897	*NF*
A-n32-k5	784	*NF*	8 866	1	60.0	342	*NF*
A-n53-k7	1010	*NF*	24 700	1	157.9	2 563	*NF*
E-n101-k14	1067	*NF*	90 700	1	546.9	2 991	*NF*

As expected, the results obtained with the hybrid BQM sampler were less promising than the CQM version. We ceased to achieve feasible solutions when nearing 300 binary variables, which is insufficient for solving any of the benchmarks from the BKS datasets. This leads us to conclude that converting problems into a BQM formulation is not advisable when they benefit from the additional features provided by CQM, such as explicit constraints and integer variables.

An interesting observation from these results is the significantly higher QPU usage compared to the CQM hybrid sampler. This indicates that the BQM hybrid sampler relies more heavily on the quantum annealer, resulting in fewer viable solutions and worse variable scaling than the CQM version. Generally, this sampler's allocation between quantum and classical resources appears less optimized. Another noteworthy metric is that all examples use only one sample from the quantum

Figure 5.5: Graphical result for the `A-n32-k5` instance obtained with Quantum Annealing using the hybrid CQM sampler.

annealer. Combined with the high QPU usage, the quantum annealer is likely not being utilized to its full potential. It is important to note that these assumptions are based on the analysis of the results, as there is no corroboration from D-Wave's official materials.

5.4 Overview of Benchmark Results

This section compiles the results from all previous sections of this chapter to facilitate a practical comparison of the benchmarks used and to discuss the various outcomes. We include all solvers, both quantum and classical, that produced significant results during the experiments. The results for the custom benchmark instances are presented in Table 5.18, while the benchmarks from the best-known solutions datasets are shown in Table 5.19. Here, *NF* indicates that the solver only reached an infeasible solution, while a dash (-) means it was not possible to run the problem instance.

Analyzing the two tables, overall **Google OR-Tools** was the best optimizer out of all the options experimented with, which was to be expected at the start of the study. Out of the quantum approaches, **Quantum Annealing (QA)** with the hybrid CQM sampler was the best option, achieving way better results than other QA samplers and even surpassing OR-Tools in some custom benchmarks. The only custom benchmark where QA failed to meet OR-Tools was `rpp-n26-k6`, where no feasible solution was found. QA was more distant from OR-Tools in the benchmarks from the best-known solutions datasets, but it still got valid results with large instances.

As mentioned before, **QAOA** was the approach with the most disappointing results, with the quantum computer unable to achieve any solutions. In this case, the simulator validated the implementation by solving the two smallest benchmark instances, while **CPLEX**, a classic optimizer,

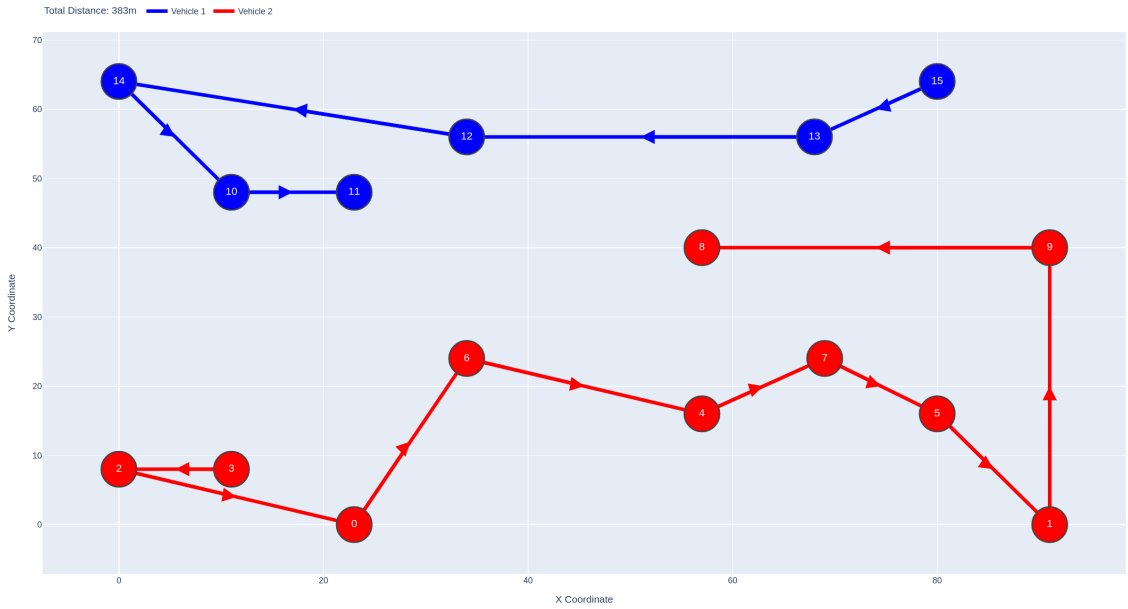

Figure 5.6: Graphical result for the `rpp-n16-k2` custom instance obtained with Quantum Annealing using the hybrid CQM sampler.

managed good results using the same QUBO formulations.

Regarding computational efficiency, OR-Tools demonstrated superior performance, consistently achieving the shortest execution times across all instances. While CPLEX produced commendable results, its execution times were generally much longer than OR-Tools. Regarding QAOA, both simulators and quantum computers exhibited significantly prolonged execution times, resulting in limited outcomes. The classical samplers from D-Wave, designed for testing purposes, exhibited poor scalability, requiring several seconds for even small instances and often timing out as the number of variables increased.

The standalone quantum annealer showed efficient execution times, but the time required for problem embedding and hardware noise hindered it, necessitating numerous samples to find feasible solutions. On the other hand, hybrid solvers took up to a couple of minutes to deliver excellent solutions, which, while somewhat lengthy, was deemed acceptable given the near-optimal solutions returned by the CQM hybrid sampler. Overall, the hybrid CQM solver demonstrated robust scalability with many variables. The timing information for each solver can be found in the previous sections of this chapter.

Table 5.17: Results obtained from running D-Wave's hybrid BQM sampler on the custom benchmark instances.

Instance	OR	QA	Vars	Samples	Exec. Time (s)	QPU Time (ms)	Gap
vrp-n2-k1	30	30	5	1	2.98	128.7	0.00%
vrp-n5-k2	214	214	92	1	2.99	151.0	0.00%
mcvrp-n8-k2	352	352	120	1	10.0	567.0	0.00%
mcvrp-n16-k2	458	*NF*	488	1	30.0	679.4	*NF*
rpp-n2-k1	15	15	3	1	3.00	111.6	0.00%
rpp-n4-k2	138	138	32	1	2.99	151.0	0.00%
rpp-n8-k2	306	*NF*	294	1	30.0	727.0	*NF*
rpp-n14-k4	387	*NF*	1 228	1	30.0	513.0	*NF*
rpp-n16-k2	394	*NF*	512	1	30.0	724.0	*NF*
rpp-n26-k6	586	*NF*	5 574	1	60.0	427.0	*NF*

Table 5.18: Results for the custom benchmark instances from all the relevant solvers experimented with. The results are displayed as solution gaps, representing the percentage difference between the solvers and OR-Tools solutions. Negative gaps represent improvements.

Instance	OR	QAOA		Quantum Annealing				
		CPLEX	Sim.	Exact	SA	QA	H. CQM	H. BQM
vrp-n2-k1	30	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
vrp-n5-k2	214	0.00%	-	-	0.00%	0.00%	0.00%	0.00%
mcvrp-n8-k2	352	0.00%	-	-	-	*NF*	0.00%	0.00%
mcvrp-n16-k2	458	10.5%	-	-	-	-	0.00%	*NF*
rpp-n2-k1	15	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
rpp-n4-k2	138	0.00%	-	-	0.00%	0.00%	0.00%	0.00%
rpp-n8-k2	306	-2.94%	-	-	-	-	-2.94%	*NF*
rpp-n14-k4	387	25.1%	-	-	-	-	0.00%	*NF*
rpp-n16-k2	394	51.5%	-	-	-	-	-2.79%	*NF*
rpp-n26-k6	586	-	-	-	-	-	*NF*	*NF*

Table 5.19: Results for the BKS benchmark instances from all the relevant solvers experimented with. The results are displayed as solution gaps, representing the percentage difference between the solvers and best-known solutions.

Instance	BKS	OR-Tools	CPLEX	QA Hybrid CQM
P-n16-k8	450	0.00%	0.00%	0.00%
P-n19-k2	212	0.00%	0.00%	50.0%
A-n32-k5	784	0.00%	-	14.2%
A-n53-k7	1010	0.00%	-	191.0%
E-n101-k14	1067	3.37%	-	234.9%

Chapter 6

Incorporating Porto's Urban Transports

This chapter demonstrates how the algorithms developed in this dissertation can be applied in a practical setting using actual data from the urban transport system of Porto, Portugal. For this purpose, we employed datasets containing information about the city's metro system (Metro do Porto [28]) and buses (STCP [23]), which are openly available on Porto's digital open data portal.

The chapter begins by detailing the datasets' origin in Section 6.1. We discuss the specific types of information they contain, such as stops, schedules, and routes, and explain why this data benefits the study. Additionally, we outline the processing steps to clean and format the data, ensuring it is suitable for input to the vehicle routing problem algorithms.

In Section 6.2, we describe how the models were applied to the extracted data and the experiments conducted to showcase the applicability of the algorithms. Various examples illustrate different VRP variations explored for both buses and metro lines, providing details on execution time, solution quality, and the impact and potential revealed by the results. Additionally, we include an analysis comparing the scalability of the qubits with the amount of real-world data the algorithms can optimize, such as the number of lines, stops, and vehicles.

6.1 Understanding and Processing the Datasets

The datasets used are openly accessible via the digital open data portal of Porto. The datasets provided by Sociedade de Transportes Coletivos do Porto (STCP), concerning buses, include data from August 2022 and September 2023 [23]. Additionally, datasets from the metro system cover July 2022, September 2023, and April 2024. The dataset for April 2024 is sourced directly from Metro do Porto's official website rather than Porto Digital [28, 30].

The datasets consist of text files with CSV data concerning different parts of the transports' details. Excluding non-important information, such as calendar dates and fare details, the relevant files for each month are explained in Table 6.1. The dataset's primary limitation lies in the absence of customer counts for entries and exits at each location. This information would significantly

enhance our experiments, allowing for realistic route optimizations based on demand and vehicle capacities. As a result, we are faced with optimizing without demand considerations or relying on rough estimations.

Table 6.1: Important files within Porto's urban transport datasets.

File	Description
stops.txt	Includes all stops within the transport system (metro or bus), making it highly valuable for populating the locations list in the input of the problem instances.
trips.txt	Includes details of all trips made on each bus or metro route throughout the month, including their respective directions.
stop_times.txt	Provides vehicles' arrival and departure times at each stop throughout the day, every day of the month. Each entry specifies the trip the vehicle is on.
routes.txt	Provides a list of all routes within the transport system (metro or bus). This information is valuable for selecting routes used in problem instances, which can be associated with their respective trips and stops.

Further subsections, 6.1.1 and 6.1.2, will detail the process of filtering and condensing the datasets, as well as preparing the information for input into vehicle routing models.

6.1.1 Methods for Data Filtering

Before converting the data into problem inputs, we first clean data to remove unwanted entries and optionally reduce information to minimize the number of qubits required for algorithm execution. Data cleaning involves removing invalid stops lacking geographical coordinates (latitude and longitude) and eliminating express routes, shorter versions of existing routes with fewer stops. We avoid using these express routes since we handle our reduction procedures as needed.

The data reduction process is controlled by a constant parameter named 'STOP_GAP'. Approximately only one every 'STOP_GAP' stops from the selected routes will be used, while always retaining the first and last stops for consistency. Consequently, during the data processing detailed in Section 6.1.2, the functions will not add stops until reaching the specified gap, thereby reducing the number of locations by approximately 'STOP_GAP' times. This adjustment is crucial because the number of qubits scales quadratically with the number of locations in all formulations presented in Chapter 3.

6.1.2 Preparing Data for Problem Modeling

We prepare the problem's input in two distinct ways, depending on whether we use circular routes (for VRP, CVRP, or Multi-CVRP formulations, depending on capacity constraints) or disjoint routes (for RPP formulations). External parameters not sourced from the dataset include information such as selected routes, number of vehicles, and their respective capacities. These capacity values can be based on the metros and buses used in Porto. For example, metros in Porto typically

accommodate between 216 and 244 passengers [27, 29], while buses typically have a capacity of up to 80 passengers [22].

To prepare the input for the circular routes, we follow these steps:

1. Select the first trip for each selected route and extract the corresponding locations.
2. If there are multiple circular routes, ensure at least one common stop; if not, halt the process.
3. Identify the stop most frequently in common among the selected routes and designate it as the depot by moving it to the beginning of the locations list.
4. Remove duplicate depots (from other routes) to guarantee a possible solution. The number of duplicates removed is the minimum between its occurrences (excluding the depot itself) and the number of vehicles minus one.
5. Compute the distance matrix by revisiting the same trips and determining the time cost between locations based on arrival and departure times at each stop. Stops without connections are assigned an infinite cost (a very high value).

To prepare the input for the disjoint routes, we follow these steps:

1. Set the locations list to include all stops in the dataset, as the RPP algorithm only considers locations involved in customers' trips.
2. Initialize the distance matrix with infinite distances (very high values) for all entries.
3. Select the first departure and return trips for each route in the dataset.
4. For each trip, complete the distance matrix by calculating the time cost between locations based on the arrival and departure times at each stop in both directions.
5. Determine the customers' trip requests by revisiting the same trips in one direction only, considering all connections as trips based on the assumption that passengers enter and leave at every stop. To estimate the number of customers for each trip, assign a demand of 1 to all or use the number of trips on that route as an indicator of demand.
6. Determine if any common stops exist between the selected routes. If they do, terminate the program because RPP does not permit such scenarios.

Alternatively, instead of calculating the distance matrix based on the dataset, the Google Distance Matrix API can be used as described in Section 4.1.1. By using the coordinates of the locations, we retrieve the distance or travel time between them and add it to the distance matrix. This approach allows for more extensive and compelling result analysis by comparing different cost functions.

6.2 Applying VRP Models to Porto's Urban Transports

In this section, the experimental results obtained from the Porto datasets discussed in Section 6.1 are examined. This analysis focuses primarily on the qualitative outcomes, evaluating the impact and potential of the algorithms in optimizing real transport systems. Examples are presented and discussed, showing execution times, the number of qubits (or variables) used, and different cost functions (namely, the dataset timetable and the Google Distance Matrix API). Following Chapter 5, **all examples were executed using D-Wave's hybrid CQM sampler** (Quantum Annealing) since it consistently yielded the best solutions out of all quantum solvers.

6.2.1 Circular Routes

For circular routes, the suitable formulations from those presented in Chapter 3 are the Vehicle Routing Problem (VRP), the Capacitated Vehicle Routing Problem (CVRP), and the Multi-Capacitated Vehicle Routing Problem (Multi-CVRP), depending on the desired capacity constraints. This is appropriate because circular routes inherently start and end at a joint depot (the route's starting point). In this scenario, since passenger demand cannot be extracted from the data, the decision is made to forgo capacity constraints and use only the VRP formulation in the examples.

To begin, we examine the example of a single vehicle traversing a circular route, specifically Porto's bus line 302, including all its stops. The schedule times from the dataset, in relation to September 2023, were used to calculate the distance matrix. With 26 locations, the problem required encoding 675 variables. The maximum execution time allocated to the sampler was 5 seconds, the minimum allowed. The result is shown in Figure 6.1, where the X coordinate represents latitude, and the Y coordinate represents longitude.

As expected, the algorithm found the exact route followed by bus line 302, which logically represents the optimal path among the given locations. The estimated travel time for the vehicle along this route is 43 minutes. The sampler successfully identified the optimal solution after 86 samples, with a total runtime of 5.3 seconds, out of which 32 milliseconds were spent on the quantum annealer. While the outcome was predicted, it demonstrates the feasibility of modeling and optimizing a real bus line using quantum or hybrid formulations.

In the following example, we introduce another route, bus line 300, which shares a few stops with the previous line 302. To reduce the number of variables, the 'STOP_GAP' parameter is set to 2, effectively halving the number of locations to 37 and encoding them using 1368 variables. Consequently, the maximum execution time was increased to 30 seconds. The result is depicted in Figure 6.2.

This time, we simulated a new scenario for the transportation system. Considering two routes covered by a single vehicle, the algorithm successfully computed a path circling all locations in approximately 1 hour and 46 minutes. The sampler executed 102 samples, with a runtime of 30.1 seconds and 32 milliseconds of QPU usage.

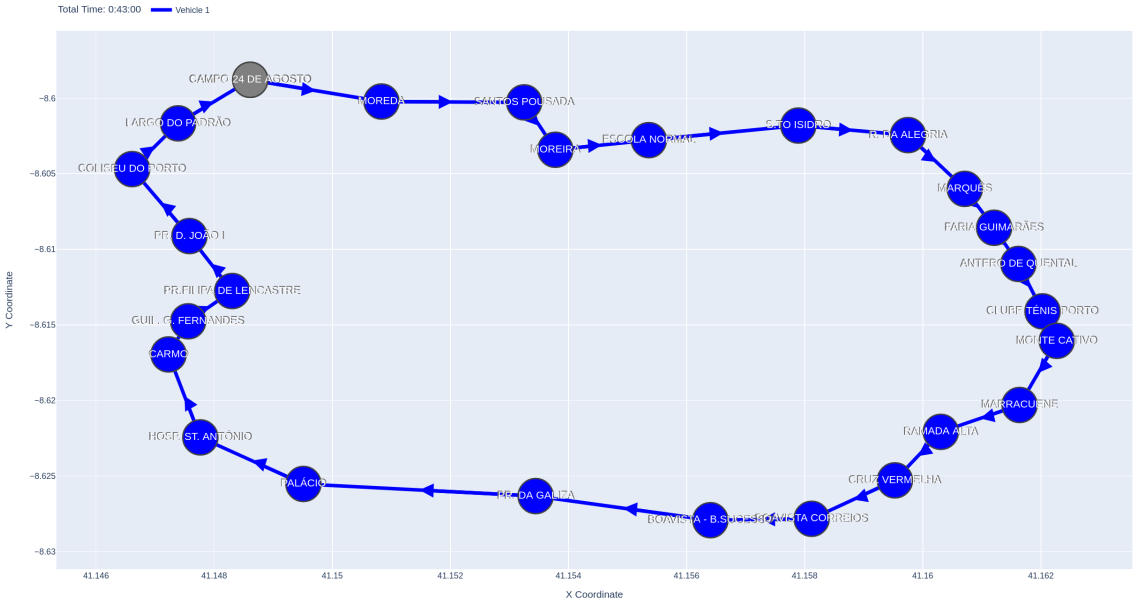

Figure 6.1: Example of a circular route for Porto's bus line 302 with one vehicle. The total traversal time is 43 minutes.

However, the previous example used the distance matrix calculated from the dataset, which only accounts for connections supported by the current bus system. Therefore, we changed the cost function to use the Google Distance Matrix API, resulting in the solution shown in Figure 6.3. In this example, the 'STOP_GAP' was increased to 3 to reduce API usage, which is limited. However, the structure of the locations remains unchanged from the previous example, and the travel time would have been the same even if there were fewer stops (since arrival and departure times do not change, see Appendix A). Hence, the input involves 25 locations encoded in 624 variables, with a maximum execution time of 5 seconds.

Interestingly, adhering to the current bus routes with these locations no longer makes sense. Using this cost function shows that it is possible to cover the stops in just 1 hour and 22 minutes, 24 minutes faster than the previous iteration using the dataset's timetable. In this instance, the sampler performed 72 samples, with a runtime of 4.8 seconds and 32 milliseconds of QPU usage.

Another approach involves using these two routes to explore the results achievable by running the algorithm with multiple vehicles. Figure 6.4 depicts an example with two vehicles. The distance matrix was computed using the dataset, and the 'STOP_GAP' was set to 3. In this scenario, 24 locations (the depot is not duplicated since we have two vehicles) require 575 variables, with a maximum execution time set to 5 seconds.

The total traversal time of the solution was 1 hour 46 minutes, with 1 hour and 3 minutes for one vehicle and 43 minutes for the other. Regarding timing, the sampling was completed in 4.9 seconds with 32 milliseconds of QPU usage, running 87 samples. Increasing the number of vehicles resulted in the same solution, indicating that adding more vehicles does not yield any improvements (see Appendix A).

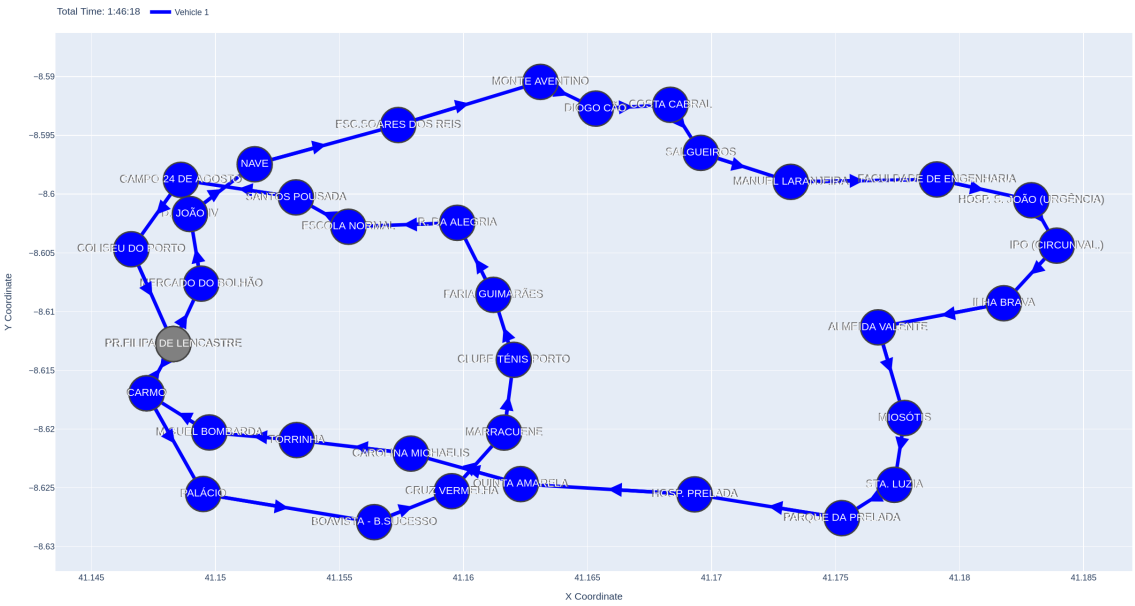

Figure 6.2: Example of multiple circular routes for Porto's bus lines 300 and 302 with one vehicle. The total traversal time is approximately 1 hour 46 minutes.

6.2.2 Disjoint Routes

The Ride Pooling Problem (RPP) formulation presented in Section 3.4 is used for disjoint routes. This is the most suitable approach, as the output will be a set of routes matching the available vehicles, ensuring all customers' trips are accommodated. Like circular routes, the dataset lacks customer information, so most examples will not employ capacity constraints. Instead, trip requests are designed to cover every connected stop in the current transportation system, with at least one customer entering and leaving at each location. There is one exception in Appendix A where capacity constraints are applied for illustrative purposes.

The first example involves a single vehicle traversing one route: Porto's metro line D, including all its stops. The timetable from the dataset, dated April 2024, was used to calculate the distance matrix. With 16 locations and 15 trips, the problem required encoding 479 variables, and the maximum execution time for the sampler was set to 5 seconds, the minimum allowed. The result is illustrated in Figure 6.5.

As expected, the algorithm found the exact route of metro line D, which logically represents the optimal path among the given locations. The estimated travel time for the vehicle along this route is 27 minutes. The sampler successfully identified the solution after 127 samples, with a total runtime of 5.1 seconds, including 32 milliseconds on the quantum annealer. Although anticipated, the outcome demonstrates the feasibility of modelling and optimizing an actual metro line using the RPP formulation.

The following example introduces two disjoint routes, bus lines 18 and 304, including all stops, with no shared locations (based on data from September 2023). This scenario involves 32 locations and 30 customer trip requests, requiring the encoding of 3838 variables. Due to the

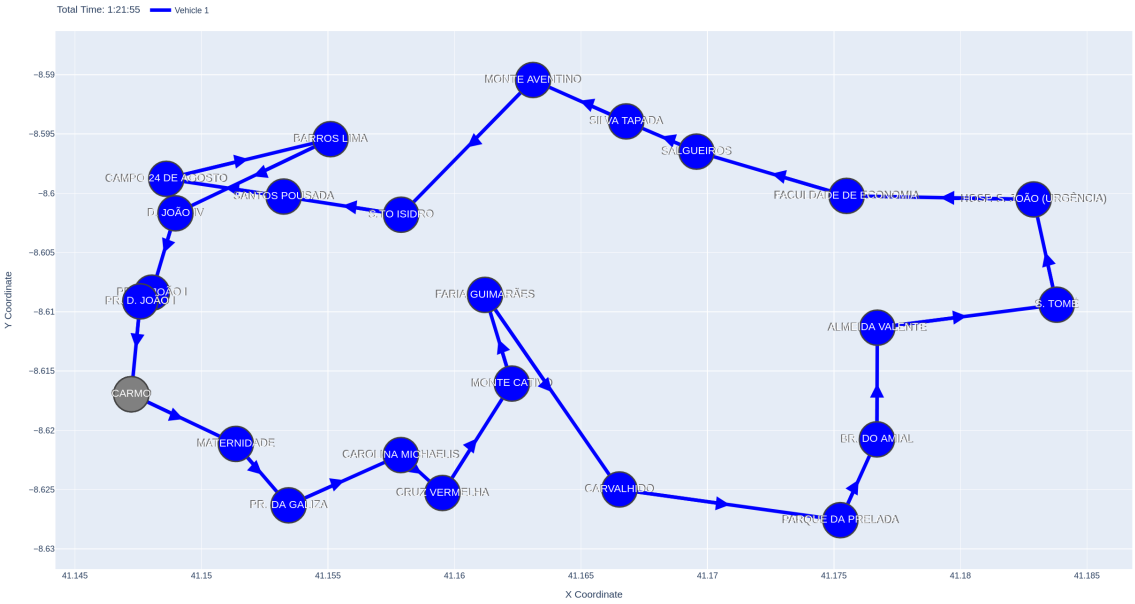

Figure 6.3: Example of multiple circular routes for Porto's bus lines 300 and 302 with one vehicle, using Google Distance Matrix API as the cost function. The total traversal time is approximately 1 hour 22 minutes.

increased complexity of this problem, the maximum execution time was increased to 30 seconds. The result is shown in Figure 6.6.

In this example, the two vehicles correspond to the two encoded bus lines, which is expected. Unlike the circular routes, using the Google Distance Matrix API does not yield more interesting solutions here, as the critical factor is the customer trip requests. Without accurate data for this parameter, the final routes will follow the initial assumptions. Nonetheless, the solution obtained covers both routes in 39 minutes, split into 14 minutes and 25 minutes. The sampler took 30.1 seconds to process 130 samples, with 32 milliseconds on the quantum device.

As long as the problem complexity remains manageable for the hybrid sampler, we can increase the number of routes and vehicles used. In the following example, in Figure 6.7, we add another route, Porto's bus line 200, to the previous scenario. This time, the 'STOP_GAP' is set to 4 to avoid overwhelming the solver. With this adjustment, there are 21 locations and 18 trip requests, resulting in 1821 variables. The maximum execution time remains 30 seconds due to the large search space with three vehicles.

In this version, the three vehicles take 1 hour and 2 minutes, split into 30, 17, and 16 minutes. The sampler processed 131 samples in a total runtime of 30 seconds, with 32 milliseconds of QPU usage.

6.3 Overview

In summary, this chapter demonstrated the practical application of Quantum Annealing, using D-Wave's hybrid CQM solver with data from Porto's urban transportation system, specifically metro

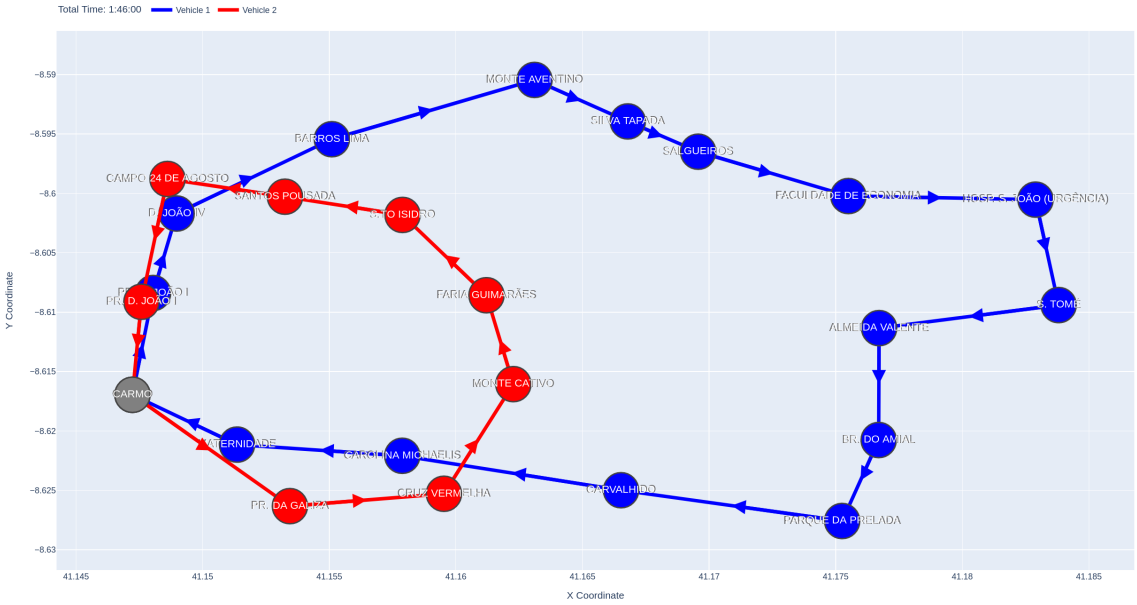

Figure 6.4: Example of multiple circular routes for Porto's bus lines 300 and 302 with two vehicles. The total traversal time is 1 hour 46 minutes, split into 1 hour 3 minutes and 43 minutes.

and bus services. The initial section explained the extraction and processing of data to convert it into inputs for various vehicle routing problems. The following section provided examples of modelling different metro and bus lines, including circular routes modelled with the VRP formulation and disjoint routes with the RPP formulation. Although the data was sufficient to show promising results, the lack of passenger information limited the realism of location demand and customer trips.

The results were promising for small and medium-scale inputs, such as optimizing up to three bus routes. However, for more significant problems, like optimizing an entire bus or metro system, current quantum hardware still faces challenges, particularly with noise, error rates, and execution time. The number of qubits is also a limitation, though this is mitigated by the hybrid sampler, which also incorporates classical optimizers. Nonetheless, the chapter presented examples with up to 32 locations and 30 customer requests, supported by visual plots and metric analysis. Additional examples can be found in Appendix A.

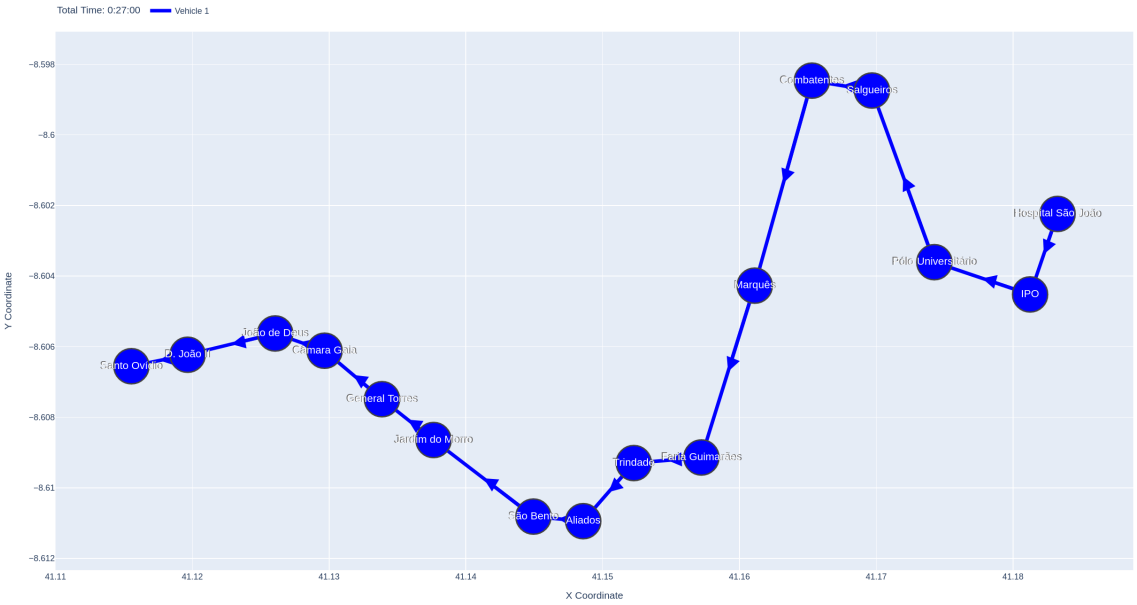

Figure 6.5: Example of a single route for Porto's metro line D with one vehicle. The total traversal time is 27 minutes.

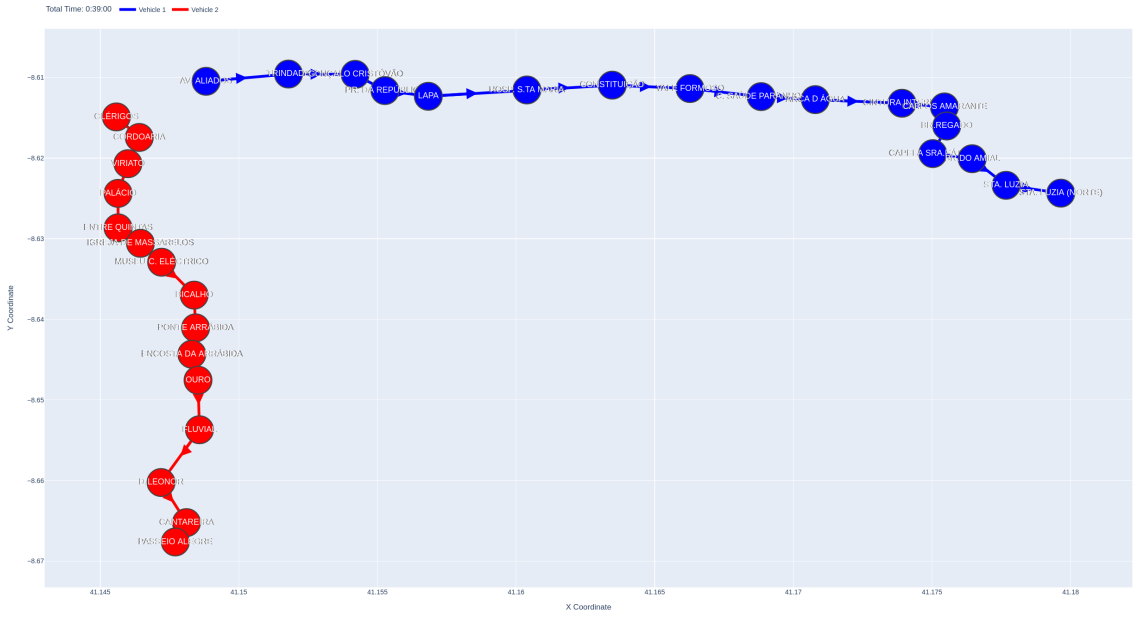

Figure 6.6: Example of two disjoint routes for Porto's bus lines 18 and 304 with two vehicles. The total traversal time is 39 minutes, split into 14 and 25 minutes.

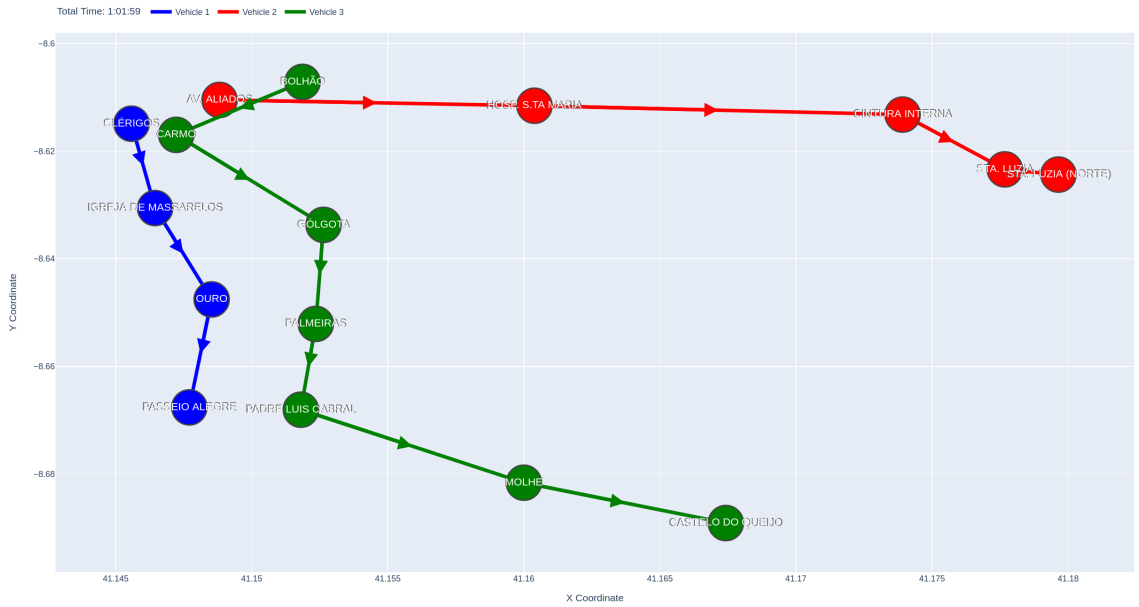

Figure 6.7: Example of three disjoint routes for Porto's bus lines 18, 304, and 200 with three vehicles. The total traversal time is 1 hour 2 minutes, split into 30, 17, and 16 minutes.

Chapter 7

Conclusions and Future Work

This chapter presents the conclusions of the work developed in this dissertation and recommendations for future research. Section 7.1 provides an overview of the main achievements in the study, considering the initial goals. Section 7.2 suggests potential directions for future work, building on the work done in this dissertation, particularly in the areas of vehicle routing problems and quantum computing.

7.1 Conclusions

In this dissertation, we investigated the application of quantum algorithms to optimize urban transportation by addressing variations of the Vehicle Routing Problem (VRP). In addition to developing theoretical formulations and algorithms for these problems, this study tackled the practical challenges of implementing quantum algorithms on actual quantum hardware. Additionally, it conducted thorough efficiency and performance analyses and applied the developments to a real-world scenario in Porto.

Prior to implementation, Chapter 2 of this dissertation presents a thorough literature review covering the background of urban transportation, quantum computing paradigms, hardware, and algorithms. It also includes an examination of related work utilizing both classical and quantum approaches to the topic. This chapter is an excellent introduction for readers interested in the field, offering a comprehensive overview of the current state-of-the-art.

Chapter 3 formulated four VRP variations using the Quadratic Unconstrained Binary Optimization (QUBO) framework. These variations include the Vehicle Routing Problem (VRP) without capacity restrictions, the Capacitated Vehicle Routing Problem (CVRP), the Multi-Capacitated Vehicle Routing Problem (Multi-CVRP), and the Ride Pooling Problem (RPP). The study focused on designing models suitable for quantum computing and optimizing them to minimize qubit requirements while considering current hardware limitations.

This dissertation implemented algorithms capable of solving the aforementioned QUBO formulations using quantum and hybrid approaches, detailed in Chapter 4. The Quantum Approximate Optimization Algorithm (QAOA) and Quantum Annealing (QA) algorithms were imple-

mented, utilizing IBM's and D-Wave's software development kits. A classical implementation was built using Google OR-Tools as a benchmark for comparison. The chapter also details the extensible software architecture designed to accommodate additional platforms and problem variations.

In Chapter 5, we evaluated the performance and efficiency of the implemented algorithms using instances from CVRP problems from datasets with best-known solutions. Custom benchmarks, calculated with OR-Tools, complemented these evaluations for other problem formulations. Quantum Annealing technology from D-Wave consistently outperformed QAOA running on IBM's gate-based computers, producing valid solutions for large instances, albeit often not optimal, and sometimes surpassing the classical implementation for medium-sized problems. The hybrid sampler proved particularly effective for Quantum Annealing, achieving superior results. However, challenges with overhead hindered QAOA's effectiveness, where execution proved impractical despite successful simulation for smaller cases with 2 locations.

Moreover, practical applications of Quantum Annealing with the hybrid solver were demonstrated on Porto's urban transport networks in Chapter 6. Publicly available data about Porto's metro and bus routes, including circular and disjoint routes, was used. The chapter features examples where up to 32 locations and 30 customer requests were successfully encoded and executed, accompanied by visual plots and metric analyses. While the current hardware scale limits the optimization of entire metro or bus networks, the algorithms show promise, particularly in smaller-scale applications such as optimizing a few routes.

From the experiments done, our findings indicate that significant advancements are needed within the quantum computing community, particularly in hardware development, specifically in qubit count, overhead reduction, noise mitigation, and error rates, which impact result accuracy. Regarding QAOA, designed for limited hardware with few qubits, current gate-based quantum computers still face challenges related to overhead and availability, rendering them impractical in solving the problems addressed in this study. On the other hand, Quantum Annealing shows promise with large qubit capacities, yet noise and error rates continue to hinder its performance, deviating from ideal adiabatic processes. Hybrid techniques, which leverage quantum annealers for specific subproblems and integrate them with classical optimizers, demonstrate significant potential and are recommended for tackling medium-sized problems in current applications.

In conclusion, this dissertation establishes a promising foundation for QUBO formulations of vehicle routing problems tailored for quantum algorithms alongside an innovative software architecture capable of encoding and solving these formulations across multiple quantum and classical platforms. The study also contributes to the analysis of current quantum hardware, particularly concerning the QAOA and QA algorithms, offering valuable insights for researchers exploring their application to other problem domains. While no immediate urban impact has been demonstrated, this dissertation illustrates the practical application of these algorithms and their potential influence on urban optimization strategies, with the example of Porto.

7.2 Future Work

The results of this dissertation are promising, but there are several areas for potential improvement and expansion. One constraint during the study was the limited access to quantum hardware from companies like D-Wave and IBM. Running experiments without execution time limitations, access to the latest machines, and queue priority would be highly beneficial.

Expanding the research could involve developing formulations for additional VRP variations by incorporating constraints such as time windows, stock replenishment, or multiple depots. Another avenue is implementing different optimization algorithms or using alternative quantum platforms, such as QAOA with photonic quantum computation [14, 84].

Further exploration could focus on applying the algorithms to real-world data. In the case of Porto, accessing customer data to understand typical travel patterns, including entry and exit points, would allow for more realistic transport optimization. Additionally, modifying the cost function to consider factors like energy consumption, carbon emissions, or customer satisfaction instead of just distance or time could enhance the impact of the results. This approach could also be applied to data from other urban areas, which might interest some researchers.

As quantum hardware advances, revisiting the experiments conducted in this dissertation and comparing future results with current findings would be intriguing.

Bibliography

[1] Ibrahim A.A, Lo N., Abdulaziz R.O, and Ishaya J.A. Capacitated vehicle routing problem. *International Journal of Research - Granthaalayah*, 7(3), April 2019. Available at https://doi.org/10.5281/zenodo.2636820.

[2] AIMMS. Capacitated vehicle routing problem formulation, August 2020. Available at https://how-to.aimms.com/Articles/332/332-Formulation-CVRP.html.

[3] Fermín Alfredo Tang Montané and Roberto Diéguez Galvão. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research*, 33(3):595–619, May 2006. Available at https://doi.org/10.1016/j.cor.2004.07.009.

[4] Utkarsh Azad, Bikash K. Behera, Emad A. Ahmed, Prasanta K. Panigrahi, and Ahmed Farouk. Solving vehicle routing problem using quantum approximate optimization algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 24(7):7564–7573, July 2023. Available at http://dx.doi.org/10.1109/TITS.2022.3172241.

[5] Alvaro Ballon. Quantum computing with superconducting qubits, March 2022. Available at https://pennylane.ai/qml/demos/tutorial_sc_qubits/.

[6] David Banister. The sustainable mobility paradigm. *Transport Policy*, 15(2):73–80, November 2007. Available at https://doi.org/10.1016/j.tranpol.2007.10.005.

[7] Christopher D. B. Bentley, Samuel Marsh, André R. R. Carvalho, Philip Kilby, and Michael J. Biercuk. Quantum computing for transport optimization. June 2022. Available at https://arxiv.org/abs/2206.07313.

[8] Rupak Biswas, Zhang Jiang, Kostya Kechezhi, Sergey Knysh, Salvatore Mandrà, Bryan O'Gorman, Alejandro Perdomo-Ortiz, Andre Petukhov, John Realpe-Gómez, Eleanor Rieffel, Davide Venturelli, Fedir Vasko, and Zhihui Wang. A nasa perspective on quantum computing: Opportunities and challenges. *Parallel Computing*, 64:81–98, November 2016. Available at https://doi.org/10.1016/j.parco.2016.11.002.

[9] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. Next-generation topology of d-wave quantum processors, February 2019. Available at https://www.dwavesys.com/media/jwwj5z3z/14-1026a-c_next-generation-topology-of-dw-quantum-processors.pdf.

[10] Kelly Boothby, Andrew D. King, and Jack Raymond. Zephyr topology of d-wave quantum processors, September 2021. Available at https://www.dwavesys.com/media/2uznec4s/14-1056a-a_zephyr_topology_of_d-wave_quantum_processors.pdf.

[11] J. Brooke, D. Bitko, T. F. Rosenbaum, and G. Aeppli. Quantum annealing of a disordered magnet, May 2001. Available at https://arxiv.org/abs/cond-mat/0105238.

[12] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*. Springer Berlin, Heidelberg, 1st edition, 2000. Available at https://doi.org/10.1007/978-3-662-04179-6.

[13] Michele Cattelan and Sheir Yarkoni. Modeling routing problems in qubo with application to ride-hailing, December 2022. Available at https://arxiv.org/abs/2212.04894.

[14] Jack Ceroni. Continuous qaoa optimization with photonic quantum computation: A tutorial, July 2019. Available at https://lucaman99.github.io/blog/2019/07/06/Continuous-QAOA.html.

[15] Mikhail V Chester and Arpad Horvath. Environmental assessment of passenger transportation should include infrastructure and supply chains. *Environmental Research Letters*, 4(2):024008, June 2009. Available at http://stacks.iop.org/ERL/4/024008.

[16] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20(3):309–318, September 1969. Available at https://doi.org/10.1057/jors.1969.75.

[17] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, August 1964. Available at https://doi.org/10.1287/opre.12.4.568.

[18] E. J. Crosson and D. A. Lidar. Prospects for quantum enhancement with diabatic quantum annealing. *Nature Reviews Physics*, 3(7):466–489, July 2021. Available at https://doi.org/10.1038/s42254-021-00313-6.

[19] Toby Cubitt, David Perez-Garcia, and Michael M. Wolf. Undecidability of the spectral gap. *Forum of Mathematics, Pi*, 10, July 2022. Available at http://dx.doi.org/10.1017/fmp.2021.15.

[20] Thibaut Cuvelier. Or-tools' vehicle routing solver, February 2023. Available at https://hal.science/hal-04015496v1/file/ROADEF_2023_ORTools_slides.pdf.

[21] George B. Dantzig. Origins of the simplex method. In *A History of Scientific Computing*, page 141–151. Association for Computing Machinery, New York, NY, USA, 1990. Available at https://dl.acm.org/doi/pdf/10.1145/87252.88081.

[22] Sociedade de Transportes Colectivos do Porto (STCP). Stcp modernize a frota com 188 novos autocarros, August 2017. Available at https://www.stcp.pt/pt/noticias/stcp-moderniza-a-frota-com-188-novos-autocarros.

[23] Sociedade de Transportes Colectivos do Porto (STCP). Horários, paragens e rotas em formato gtfs (stcp), September 2023. Available at https://opendata.porto.digital/dataset/horarios-paragens-e-rotas-em-formato-gtfs-stcp.

[24] Mauro Dell'Amico, Giovanni Righini, and Matteo Salani. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40:235–247, May 2006. Available at https://doi.org/10.1287/trsc.1050.0118.

[25] Guy Desaulniers, Jacques Desrosiers, Andreas Erdmann, Marius Solomon, and F. Soumis. *VRP with Pickup and Delivery*, pages 225–242. January 2002. Available at http://dx.doi.org/10.1137/1.9780898718515.ch9.

[26] P. A. M. Dirac. A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35(3):416–418, April 1939. Available at https://doi.org/10.1017/S0305004100021162.

[27] Metro do Porto. Flexity swift, May 2016. Available at https://en.metrodoporto.pt/pages/406.

[28] Metro do Porto. Horários, paragens e rotas em formato gtfs, September 2023. Available at https://opendata.porto.digital/dataset/horarios-paragens-e-rotas-em-formato-gtfs.

[29] Metro do Porto. O novo membro da família, February 2023. Available at https://www.metrodoporto.pt/pages/771?news_id=457.

[30] Metro do Porto. Mapas e horários, April 2024. Available at https://www.metrodoporto.pt/pages/337.

[31] Project Drawdown. Public transit, 2020. Available at https://drawdown.org/solutions/public-transit.

[32] W. Dür and S. Heusler. What we can learn about quantum physics from a single qubit, December 2013. Available at https://arxiv.org/abs/1312.1463v1.

[33] Daniel J. Egger, Jakub Mareček, and Stefan Woerner. Warm-starting quantum optimization. *Quantum*, 5:479, June 2021. Available at http://www.arxiv.org/abs/2009.10095.

[34] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, November 2014. Available at https://arxiv.org/abs/1411.4028.

[35] Sebastian Feld, Christoph Roch, Thomas Gabor, Christian Seidel, Florian Neukart, Isabella Galter, Wolfgang Mauerer, and Claudia Linnhoff-Popien. A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer. *Frontiers in ICT*, 6, June 2019. Available at https://www.frontiersin.org/articles/10.3389/fict.2019.00013/full.

[36] Jay Gambetta. Ibm quantum blog - the hardware and software for the era of quantum utility is here, December 2023. Available at https://www.ibm.com/quantum/blog/quantum-roadmap-2033.

[37] Fred Glover, Gary Kochenberger, and Yu Du. A tutorial on formulating and using qubo models, November 2019. Available at https://arxiv.org/abs/1811.11538.

[38] Changqing Gong, Ting Wang, Wanying He, and Han Qi. A quantum approximate optimization algorithm for solving hamilton path problem. *The Journal of Supercomputing*, 78(13):15381–15403, April 2022. Available at https://doi.org/10.1007/s11227-022-04462-y.

[39] Erica K. Grant and Travis S. Humble. Adiabatic quantum computing and quantum annealing, July 2020. Available at https://doi.org/10.1093/acrefore/9780190871994.013.32.

[40] M.A. Hannan, Mahmuda Akhtar, R.A. Begum, H. Basri, A. Hussain, and Edgar Scavino. Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using pso algorithm. *Waste Management*, 71:31–41, January 2018. Available at https://www.sciencedirect.com/science/article/pii/S0956053X17307675/pdfft?md5=4dbae25767b6ae36f4bda498f62c472b&pid=1-s2.0-S0956053X17307675-main.pdf.

[41] Ramkumar Harikrishnakumar, Saideep Nannapaneni, Nam H. Nguyen, James E. Steck, and Elizabeth C. Behrman. A quantum annealing approach for dynamic multi-depot capacitated vehicle routing problem, May 2020. Available at https://arxiv.org/abs/2005.12478.

[42] Itay Hen and Federico M. Spedalieri. Quantum annealing for constrained optimization. *Physical Review Applied*, 5(3), March 2016. Available at http://dx.doi.org/10.1103/PhysRevApplied.5.034007.

[43] Omnissiahs Hierophant. The hadamard-cnot transform on the zero-state, December 2012. Available at https://commons.wikimedia.org/w/index.php?curid=85279984.

[44] Ali Asghar Rahmani Hosseinabadi, Najmeh Sadat Hosseini Rostami, Maryam Kardgar, Seyedsaeid Mirkamali, and Ajith Abraham. A new efficient approach for solving the capacitated vehicle routing problem using the gravitational emulation local search algorithm. *Applied Mathematical Modelling*, 49:663–679, March 2017. Available at https://doi.org/10.1016/j.apm.2017.02.042.

[45] Ciaran Hughes, Joshua Isaacson, Anastasia Perry, Ranbel F. Sun, and Jessica Turner. *Quantum Computing for the Quantum Curious*. Springer, 1st edition, 2021.

[46] IBM. Ibm® decision optimization cplex® modeling for python, 2022. Available at http://ibmdecisionoptimization.github.io/docplex-doc/.

[47] D-Wave Quantum Systems Inc. D-wave ocean software documentation, June 2024. Available at https://docs.ocean.dwavesys.com/en/stable/index.html.

[48] D-Wave Quantum Systems Inc. Quantum annealing solver official documentation, June 2024. Available at https://docs.dwavesys.com/docs/latest/guides_solvers.html.

[49] Hirotaka Irie, Goragot Wongpaisarnsin, Masayoshi Terabe, Akira Miki, and Shinichirou Taguchi. Quantum annealing of vehicle routing problem with time, state and capacity, March 2019. Available at https://arxiv.org/abs/1903.06322.

[50] Richard Jozsa and Noah Linden. On the role of entanglement in quantum-computational speed-up. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 459(2036):2011–2032, August 2003. Available at http://dx.doi.org/10.1098/rspa.2002.1097.

[51] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355–5363, November 1998. Available at http://dx.doi.org/10.1103/PhysRevE.58.5355.

[52] William M. Kaminsky and Seth Lloyd. Scalable architecture for adiabatic quantum computing of np-hard problems, November 2002. Available at https://arxiv.org/abs/quant-ph/0211152.

[53] Sahar Karimi and Pooya Ronagh. Practical integer-to-binary mapping for quantum annealers. *Quantum Information Processing*, 18(4), July 2017. Available at http://dx.doi.org/10.1007/s11128-019-2213-x.

[54] Morten Kjaergaard, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I.-J. Wang, Simon Gustavsson, and William D. Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11(1):369–395, March 2020. Available at http://dx.doi.org/10.1146/annurev-conmatphys-031119-050605.

[55] Lady-shirakawa. Map of vrp subproblems, November 2014. Available at https://commons.wikimedia.org/wiki/File:Map_of_vrp_subproblems.jpg.

[56] Jonathan Wei Zhong Lau, Kian Hwee Lim, Harshank Shrotriya, and Leong Chuan Kwek. Nisq computing: where are we and where do we go? *AAPPS Bulletin*, 32(1):27, September 2022. Available at https://doi.org/10.1007/s43673-022-00058-z.

[57] Valeria Leggieri and Mohamed Haouari. A matheuristic for the asymmetric capacitated vehicle routing problem. *Discrete Applied Mathematics*, 234:139–150, January 2018. Available at https://doi.org/10.1016/j.dam.2016.03.019.

[58] Ivan Lima, Uchoa, Eduardo, Pecin, Diego, Pessoa, Artur, Poggi, Marcus, Vidal, Thibaut, Subramanian, Anand, Daniel Oliveira, and Eduardo Queiroga. Capacitated vehicle routing problem library, 2021. Available at http://vrp.galgos.inf.puc-rio.br/index.php/en/.

[59] Haifeng Lin and Chengpei Tang. Intelligent bus operation optimization by integrating cases and data driven based on business chain and enhanced quantum genetic algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):9869–9882, November 2022. Available at https://doi.org/10.1109/TITS.2021.3121289.

[60] Google LLC. Google maps platform official developer documentation, June 2024. Available at https://developers.google.com/maps/documentation.

[61] Google LLC. Google or-tools official developer documentation, June 2024. Available at https://developers.google.com/optimization.

[62] Andrew Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2, January 2014. Available at http://dx.doi.org/10.3389/fphy.2014.00005.

[63] W.A.J. Luxemburg and A.C. Zaanen. Chapter 8 hermitian operators in hilbert space. In *Riesz Spaces*, volume 1 of *North-Holland Mathematical Library*, pages 369–418. Elsevier, 1971. Available at https://www.sciencedirect.com/science/article/abs/pii/S0924650908704859.

[64] Mario Marinelli, Aleksandra Colovic, and Mauro Dell'Orco. A novel dynamic programming approach for two-echelon capacitated vehicle routing problem in city logistics with environmental considerations. *Transportation Research Procedia*, 30:147–156, 2018. Available at https://doi.org/10.1016/j.trpro.2018.09.017.

[65] Catherine McGeoch and Pau Farré. Advantage processor overview, January 2022. Available at https://www.dwavesys.com/media/3xvdipcn/14-1058a-a_advantage_processor_overview.pdf.

[66] David C. McKay, Ian Hincks, Emily J. Pritchett, Malcolm Carroll, Luke C. G. Govia, and Seth T. Merkel. Benchmarking quantum processor performance at scale, November 2023. Available at https://arxiv.org/abs/2311.05933.

[67] Eugen Merzbacher. The early history of quantum tunneling. *Physics Today - PHYS TODAY*, 55:44–50, August 2002. Available at http://dx.doi.org/10.1063/1.1510281.

[68] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, October 1960. Available at https://doi.org/10.1145/321043.321046.

[69] André Mamprin Mori. Replanning flight schedules using quantum computing. Faculdade de Engenharia da Universidade do Porto, July 2022. Available at https://hdl.handle.net/10216/142739.

[70] Satoshi Morita and Hidetoshi Nishimori. Mathematical foundation of quantum annealing. *Journal of Mathematical Physics*, 49(12), December 2008. Available at http://dx.doi.org/10.1063/1.2995837.

[71] S. Mukherjee and B.K. Chakrabarti. Multivariable optimization: Quantum annealing and computation. *The European Physical Journal Special Topics*, 224(1):17–24, February 2015. Available at https://arxiv.org/abs/1408.3262.

[72] Pedro Munari, Twan Dollevoet, and Remy Spliet. A generalized formulation for vehicle routing problems, September 2017. Available at https://arxiv.org/abs/1606.01935.

[73] Gábor Pataki. Teaching integer programming formulations using the traveling salesman problem. *SIAM Review*, 45(1):116–123, January 2003. Available at https://doi.org/10.1137/S00361445023685.

[74] Rafael Pereira da Silva. Annealing quantum computing: An overview, July 2023. Available at https://www.researchgate.net/publication/372134305_Annealing_Quantum_Computing_An_Overview.

[75] V. Praveen, Dr.P. Keerthika, G. Sivapriya, A. Sarankumar, and Boddu Bhasker. Vehicle routing optimization problem: A study on capacitated vehicle routing problem. *Materials Today: Proceedings*, 64:670–674, July 2022. Available at https://doi.org/10.1016/j.matpr.2022.05.185.

[76] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018. Available at https://doi.org/10.22331/q-2018-08-06-79.

[77] IBM Quantum. Development & innovation roadmap, 2024. Available at https://www.ibm.com/quantum/assets/IBM_Quantum_Development_&_Innovation_Roadmap.pdf.

[78] IBM Quantum. Ibm quantum documentation, 2024. Available at https://docs.quantum.ibm.com.

[79] Ted Ralphs. Vehicle routing data sets, October 2003. Available at https://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/index.htm.old#A.

[80] T.K. Ralphs, L. Kopman, W.R. Pulleyblank, and L.E. Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2):343–359, January 2003. Available at https://doi.org/10.1007/s10107-002-0323-0.

[81] Julia Rieck and Jürgen Zimmermann. Exact solutions to the symmetric and asymmetric vehicle routing problem with simultaneous delivery and pick-up. *BuR Business Research Journal*, 6(1), May 2013. Available at https://ssrn.com/abstract=2266430.

[82] E.G. Rieffel and W.H. Polak. *Quantum Computing: A Gentle Introduction*. Scientific and Engineering Computation. MIT Press, 2011.

[83] Eleanor G. Rieffel and Wolfgang Polak. An introduction to quantum computing for non-physicists, January 2000. Available at https://arxiv.org/abs/quant-ph/9809016.

[84] Jacquiline Romero and Gerard Milburn. Photonic quantum computing, April 2024. Available at https://arxiv.org/abs/2404.03367.

[85] Amelie Schreiber. What is a quantum circuit transpiler? *Towards Data Science*, December 2019. Available at https://towardsdatascience.com/what-is-a-quantum-circuit-transpiler-ba9a7853e6f9.

[86] Ruslan Shaydulin. Combinatorial optimization on quantum computers (lecture notes), September 2021. Available at https://github.com/rsln-s/QAOA_tutorial/blob/cabcb8190035c51116358da8c4f039bdf1ba094f/Slides.pdf.

[87] Peter Shor. Quantum approximate optimization algorithm - isca 2018 (lecture notes), June 2018. Available at https://static1.squarespace.com/static/5a56fca2c027d8a33aa5c48f/t/5b2050918a922d6f5dabb935/1528844433868/QAOA-talk.pdf.

[88] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition, 2012.

[89] M. Steffen, D. P. DiVincenzo, J. M. Chow, T. N. Theis, and M. B. Ketchen. Quantum computing: An ibm perspective, October 2011. Available at https://ieeexplore.ieee.org/abstract/document/6032777.

[90] Qiskit Optimization Development Team. iskit optimization official documentation, May 2024. Available at https://qiskit-community.github.io/qiskit-optimization/index.html.

[91] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257, August 2016. Available at doi.org/10.1016/j.ejor.2016.08.012.

[92] Amit Verma and Mark Lewis. Penalty and partitioning techniques to improve performance of qubo solvers. *Discrete Optimization*, 44:100594, June 2022. Available at https://doi.org/10.1016/j.disopt.2020.100594.

[93] VRP-REP. Vrp-rep: The vehicle routing problem repository, 2014. Available at http://vrp-rep.org/.

[94] Kangzhou Wang, Yeming Shao, and Weihua Zhou. Matheuristic for a two-echelon capacitated vehicle routing problem with environmental considerations in city logistics service. *Transportation Research Part D: Transport and Environment*, 57:262–276, December 2017. Available at https://doi.org/10.1016/j.trd.2017.09.018.

[95] Niaz A. Wassan and Gábor Nagy. Vehicle routing problem with deliveries and pickups: Modelling issues and meta-heuristics solution approaches. *International Journal of Transportation*, 2(1):95–110, 2014. Available at http://dx.doi.org/10.14257/ijt.2014.2.1.06.

[96] Johannes Weidenfeller. Introduction to the quantum approximate optimization algorithm and applications - qiskit global summer school 2021 (lecture notes), 2021. Available at https://github.com/Qiskit/platypus/blob/41b1209dde46a27305d4fd4900acde78c2545fff/notebooks/summer-school/2021/resources/lecture-notes/Lecture5.2.pdf.

[97] G Wendin. Quantum information processing with superconducting circuits: a review. *Reports on Progress in Physics*, 80(10):106001, September 2017. Available at http://dx.doi.org/10.1088/1361-6633/aa7e1a.

[98] Noson S. Yanofsky and Mirco A. Mannucci. *Quantum Computing for Computer Scientists*. Cambridge University Press, 1st edition, 2008.

[99] Sheir Yarkoni, Elena Raponi, Thomas Bäck, and Sebastian Schmitt. Quantum annealing for industry applications: introduction and review. *Reports on Progress in Physics*, 85(10):104001, September 2022. Available at https://dx.doi.org/10.1088/1361-6633/ac8c54.

[100] Çağrı Koç, Gilbert Laporte, and İlknur Tükenmez. A review of vehicle routing with simultaneous pickup and delivery. *Computers & Operations Research*, 122:104987, May 2020. Available at https://www.sciencedirect.com/science/article/pii/S0305054820301040.

Appendix A

Additional Examples of Applying Models to Porto's Urban Transports

In this appendix, we present additional examples of applying the models to Porto's urban transport systems, as introduced in Chapter 6. While these examples do not provide new conclusions beyond what is covered in the main text, they offer interesting insights. These supplementary examples further illustrate the practical application of the discussed methodologies, highlighting their effectiveness in optimizing transportation routes in real-world scenarios.

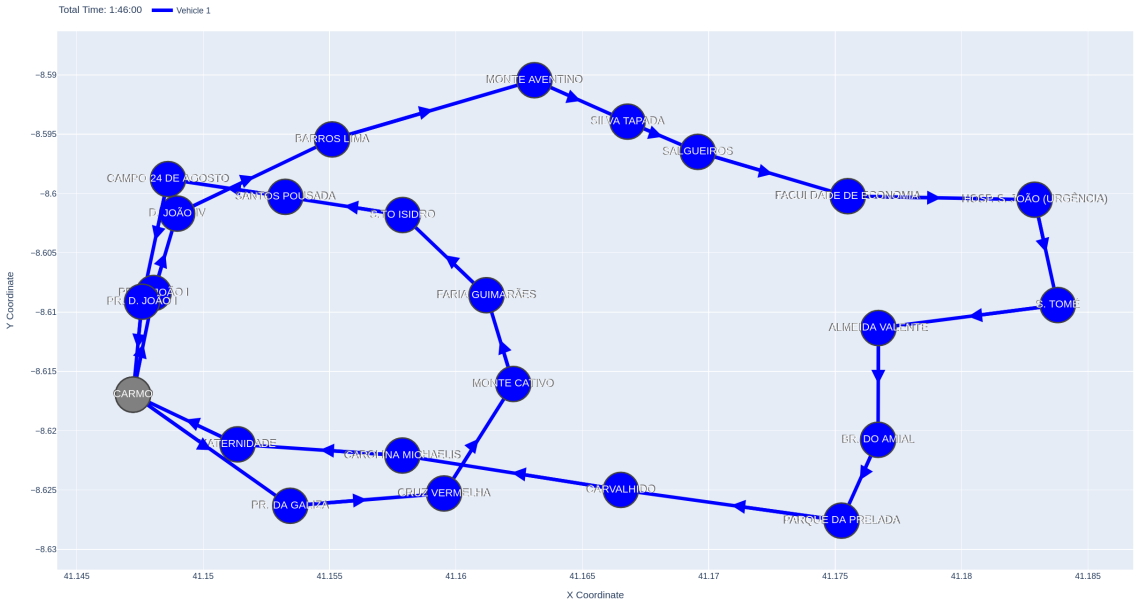

Figure A.1: Example of multiple circular routes for Porto's bus lines 300 and 302 with one vehicle and 'STOP_GAP' set to 3, using data from September 2023. The total traversal time is approximately 1 hour and 46 minutes. With 25 locations, the problem was encoded with 624 variables.

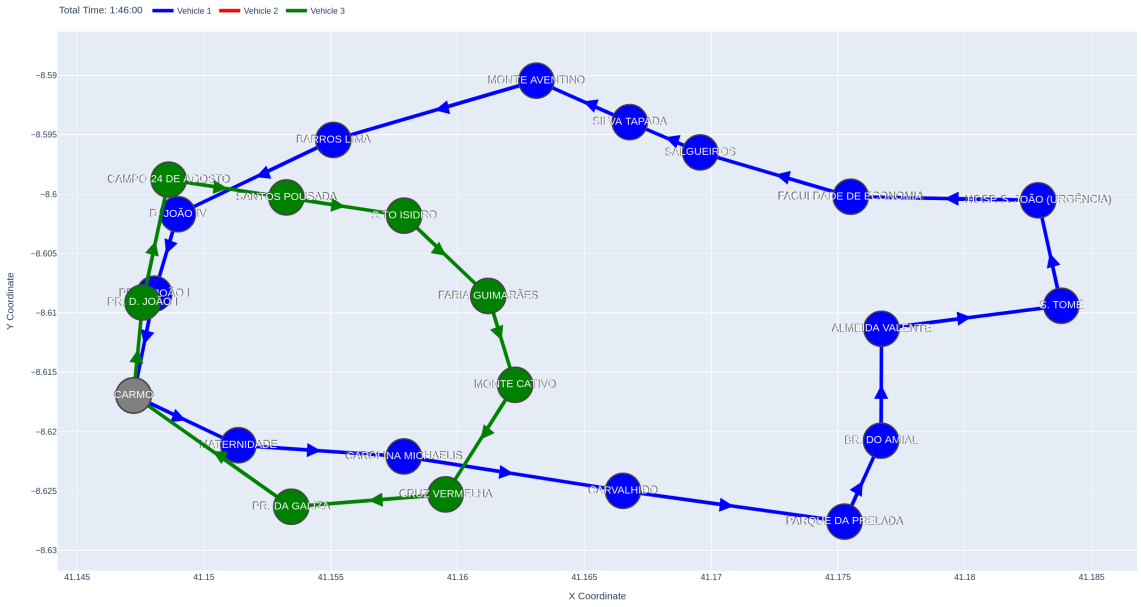

Figure A.2: Example of multiple circular routes for Porto's bus lines 300 and 302 with three vehicles and 'STOP_GAP' set to 3, using data from September 2023. The total traversal time is approximately 1 hour and 46 minutes. With 24 locations, the problem was encoded with 575 variables. Compared to Figure 6.4, this example demonstrates that adding more vehicles does not improve the solution.

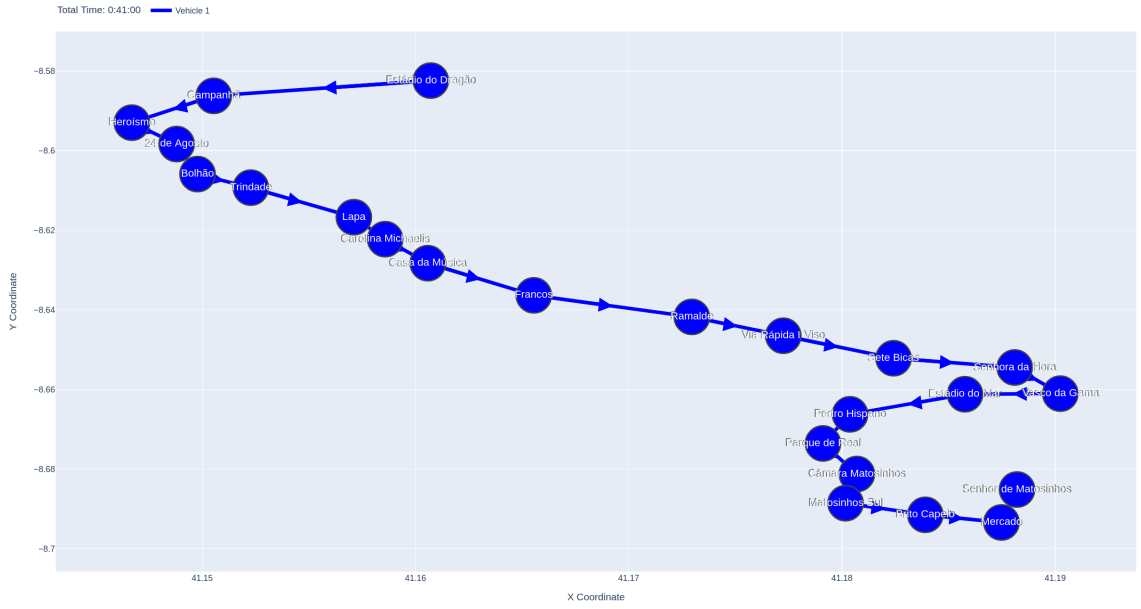

Figure A.3: Example of a single route representing Porto's metro line A with one vehicle, using data from April 2024. The total traversal time is 41 minutes. With 23 locations and 22 trip requests, the model encoded 1011 variables.

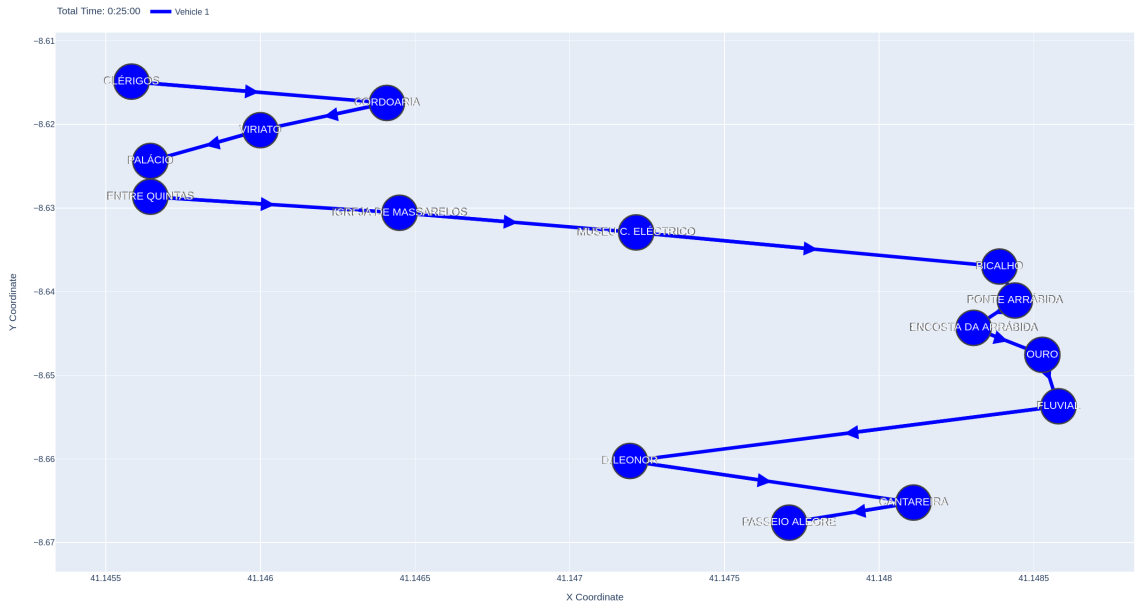

Figure A.4: Example of a single route representing Porto's bus line 18 with one vehicle, using data from September 2023. The total traversal time is 25 minutes. With 15 locations and 14 trip requests, the model encoded 419 variables.

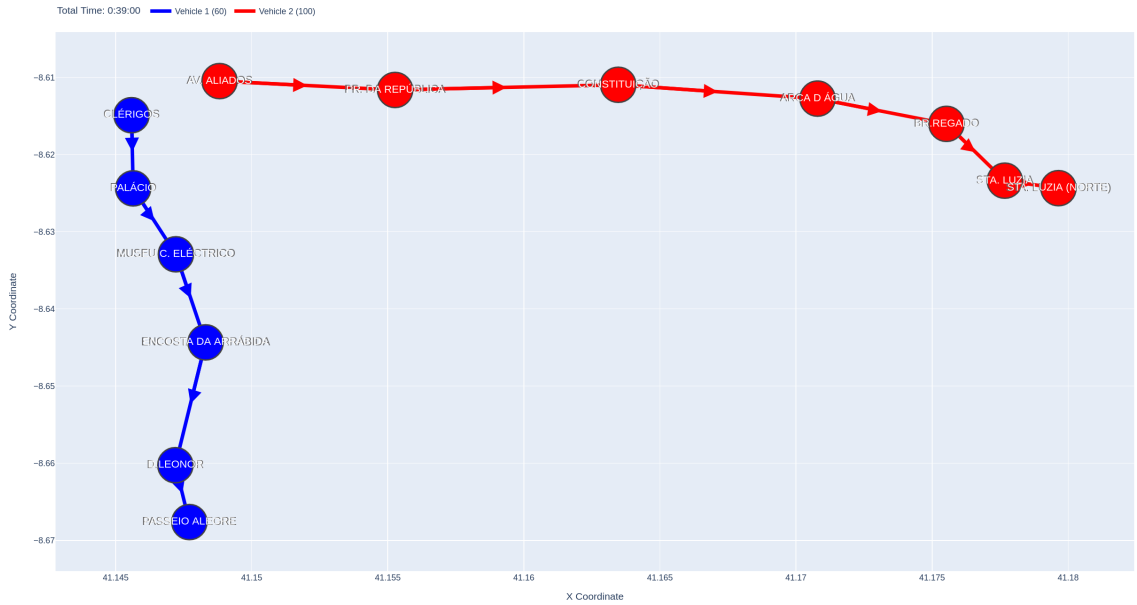

Figure A.5: Example of two disjoint routes representing Porto's bus lines 18 and 304 with two vehicles of different capacities (Multi-CVRP formulation), using data from September 2023. The total traversal time is 39 minutes, divided into 14 and 25 minutes. With 13 locations and 11 trip requests, the model encoded 570 variables.