Department of Mathematics
TUM School of Computation, Information and Technology

Solving the CVRPTW: Quantum Annealing vs. Exact Optimization

Master's Thesis by Oscar Axel Vargas Salomón

Examiner:	Prof. Dr. Stefan Weltge.
Advisor:	Prof. Dr. Martin Bichler, M.Sc. Matthias Oberlechner.
Submission Date:	27^{th} October 2022.

I hereby confirm that this is my own work, and that I used only the cited sources and materials.

Munich, 27^{th} October 2022.

Oscar Axel Vargas Salomón

Acknowledgments

I would like to express my gratitude to Dr. Paul Sutterer for inviting me to join this project and for his guidance regarding its initial state. His ideas were vital for a proper formulation of the problem and project continuity. I am very grateful to Dr. Andreas Jäger for sharing his knowledge of Quantum Computing; his support was critical. Without his help, this thesis would not have been possible in the same way.

I want to thank Professor Richard Allmendinger deeply; his expertise in the application of Quantum Annealing was highly valued. His continuous assistance and alternative solution paths were vital. Dr. Pradyumn Kumar Shukla's enhancements in mathematical modeling strongly enriched the project. His support and revision of the problem setting were precious.

My deeper thanks go to Professor Martin Bichler for making this joint project of the Technical University of Munich and Siemens Advanta Consulting possible. Especial gratitude to doctoral candidate Matthias Oberlechner for his supervision and outstanding support throughout the whole thesis.

Furthermore, I would like to thank the entire project team and all the people involved from Siemens for the fruitful discussions, insights regarding business understanding, and for not letting this study perish.

Finally, I would like to thank my beloved family for their continuous encouragement and unconditional support throughout my studies and life.

Abstract

Quantum Computing (QC) has proven to be a promising technology in several business applications. Notably, this technology is worth studying on problems whose complexity demands an overwhelming amount of solving time. Additionally, due to Moore's law, the increase in computing capability is starting to be limited. Alternative algorithms for solving complex problems must be evaluated.

Quantum Annealing (QA) is a meta-heuristic quantum algorithm designed for solving combinatorial optimization problems. A well-known combinatorial optimization problem many companies face nowadays is distribution logistics. A fleet of vehicles picks up or delivers goods optimally from one depot to various customer locations. Truck-specific capacity and schedule constraints must be fulfilled. This problem is known as the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW), classified as an NP-hard problem. Throughout this work, CVRPTW was adapted to solve a Siemens AG operational process, expecting to find a real-time QA solution response required for potential industrial applicability. Furthermore, the modeling approach is based on a binary programming formulation of the CVRPTW in a time-expanded network. A time-expanded network allows discretizing time by expanding the location nodes with time dependencies, enabling the utilization of indicator decision variables.

This work introduces a performance comparison of an adjusted CVRPTW on a four-month simulation task solved via an exact optimization algorithm (CBC) and QA, respectively. D-Wave's quantum annealer Advantage 6.1 (5,640 qubits) with the CBC solver as a baseline was employed. Both solvers were run remotely. The four-month simulation was split into pieces of three days, run sequentially. Each of these three-day instances included a different number of nodes. QA handled up to 12 nodes instances, plus two exceptions that could not be solved with seven and nine nodes. It was found that the transformations of the initial problem into an eligible QA intake hinder the solving time advantages of this algorithm, such as parameter tuning and quantum embedding for each instance. Therefore, a real-world application of the CVRPTW via QA is not yet recommended. However, QA outperformed the CBC, by an order of about two of magnitude, in finding the optimal solution once all parameters and transformations were completed; demonstrating the potential of this technology in the nearer future.

"...nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum-mechanical...."

- Richard Feynman, 1981.

Contents

Chapter 1

Introduction

Nowadays, technology has helped us to find more useful organizational structures. However, there are still topics that current technologies cannot yet adapt to. One sort of challenges faced by enterprises regards goods distribution. For instance, the EU assessed the impact of this sector because of problems related to the COVID-19 pandemic. Contingency strategies to address these problems include reducing the number of vehicles required for cross-border transport, thereby reducing the economic risk of this sector.

On the other hand, the European Union envisions a net-zero greenhouse gas emissions economy by 2050; nonetheless, future trends show that without additional actions to curb CO_2 emissions, the share of CO_2 from Heavy Goods Vehicle (HGV)[1] transports will increase from 27% in 2016 to 32% in 2030, [1]. Consequently, testing new technologies that help us to address this problem is vital to foster a global-economic development without sacrificing certain services. Vehicle Routing Problem (VRP) is a class of combinatorial optimization problems that could help allay these concerns. Unfortunately, the solving time required for large instances increases exponentially as more customers or visiting points are added; therefore, the industry is not encouraged to employ it in its operational processes. Quantum Annealing (QA) is developed to solve optimization problems making it a proper candidate for overcoming VRPs implementation difficulties.

1.1 Problem Definition

This research explores a meta-heuristic algorithm based on Quantum Computing (QC) principles, QA, designed to solve combinatorial optimization problems. Notably, this work is interested in solving a well-known combinatorial optimization problem called the Vehicle Routing Problem (VRP). The VRP is classified as an NP-hard[2] class problem; therefore, solving a problem using standard optimization techniques in an acceptable

[1] HGV is any goods vehicle over 3.5 tonnes.

[2] A problem is NP-hard if all problems in NP can be reduced to it [2].

time is challenging [3]. Moreover, a mathematical routing optimization formulation based on the requirements of Siemens AG[3] is presented. The requirements included scheduling hours, specific-truck parameters such as capacity and volume and driving hours. Further details regarding the model assumptions will be explain in Chapter 4.

The main idea is to simulate the distribution of products. The simulation was derived using almost four months of data from a logistic operational process of Siemens. In order to obtain possible routing solutions from the simulation, two approaches were employed, a classical and a quantum solver, seeking to compare their performances. The classical solver work as a benchmark model that will be run remotely on a classical computer[4]. Throughout the thesis, the classical solver will also be referred to as an exact solver.

1.2 Theoretical Background

The following section introduces two central topics of this thesis. The first subsection defines the VRP, while the second subsection explains the basic concepts related to Quantum Mechanics helpful to understanding the QA algorithm.

1.2.1 Vehicle Routing Problem

The VRP can be defined as the problem of optimally designing routes for goods delivery, starting and ending at a warehouse, through multiple customers at distinct geographical locations. This problem also considers a loading capacity, contrary to a similar problem known as the Traveling Salesman Problem (TSP). Further information concerning this class of problems will be explained in Chapter 2. VRPs are subject to certain constraints, giving them new names accordingly. For instance, scheduling constraints generate a new branch called VRP with Time Windows. Further discussions regarding VRP variants are studied in Chapter 2. The set of constraints employed to address the problem is described in Chapter 3.

1.2.2 Quantum Mechanical Theory

Quantum annealing uses quantum physics to find low-energy states of a problem; this algorithm helps find an optimal or near-optimal solution to a combinatorial problem [4]. QA relies on a physics compass that can be observed in nature's behavior. Nature always attempts to utilize the minimal available energy. For example, it is possible to

[3]Multinational company with headquarters in Munich, Germany.
[4]Current John von Neumann computers architecture.

observe this pattern in the straight line drawn when an object falls to earth and on the refraction in the light when it is going through a glass of water. QA uses this principle to find optimal possible solutions through *quantum tunneling*[5], which is considered a valuable property to reaching a global minimum solution faster and of better quality than classical annealers[6].

Hamiltonian

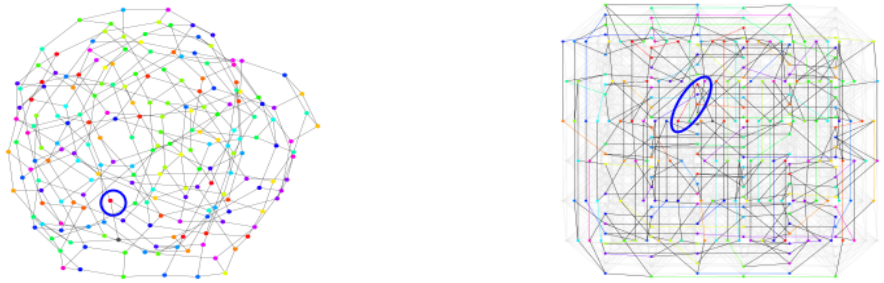
The Hamiltonian is a mathematical formulation of a physical system in terms of its energy; under certain circumstances, it accurately describes a physical system. By designing the Hamiltonian model to mimic an optimization problem, we can prepare a final quantum state that represents the solution to the original optimization problem [6]. Chapter 3 defines the Hamiltonian corresponding to the problem that is solved in this thesis.

Qubit

In Classical Computing (CC), a bit is the most basic unit of information. This information is handled through transistors, which either let energy pass through or not, giving a 0 or a 1. The evolution in the capacity of classical processing has been given by its ability to house more significant numbers of transistors in smaller spaces. Unfortunately, this evolution is being threatened due to the physical property of the electrons. The barriers established by the transistors are not enough to stop the path of energy due to a quantum effect. QC has its own basic unit of information called *qubit*. A qubit represents the energy state of a system, and contrary to a bit, it can be in a superposition of the state 0 and state 1. Once the final state is achieved, the qubit collapses from its quantum state into a classical one, 0 or 1. Qubits are particularly useful since a series of them can be connected through a physical property called *quantum entanglement*. In quantum mechanics, quantum entanglement is referred to as *coupling* due to the short distance between the qubits. The following Subsection describes how the connection through coupling between qubits is handled for using QA.

[5]Quantum tunneling refers to the nonzero probability that a particle in quantum mechanics can be measured to be in a state that is forbidden in classical mechanics [5]; this can also be described as a particle ability to transfer through possible energy barriers.

[6]In metallurgy, *annealing* is the process of material heating so its temperature can be stabilized by cooling it down slowly.



(a) QUBO graph (Logical problem graph).

(b) Minor-embedding graph (Physical qubit graph).

Figure 1.1: Example of a QUBO graph (a) onto a graph (b). Each node in (a) maps to a chain (qubit coupling) in (b). Chain nodes and edges are colored to match their source nodes, and logical edges from the original graph are black. Qubits and edges that are unused in this embedding are gray. The blue circle highlights an example of a node mapped onto a two qubits chain [4].

Embedding

Formulating the original objective function in a format such that QA can be employed is necessary. QA starts with the *embedding* process. Firstly, the problem of interest needs to be defined in its Quadratic Unconstrained Binary Optimization (QUBO) form, so the translation from CC to QC can be performed through the embedding. For example, the value 11 in the binary number system is $x = 1011$, representing the 4-qubit $|1011\rangle$. The general mapping process is called *minor-embedding*, consisting of mapping the QUBO problem onto a sub-graph of the quantum graph[7]. Figure 1.1 dispatches an example of a minor-embedding of a QUBO map onto a minor-embeding graph. Information from "The D-Wave[8] Advantage2 Prototype" report [4].

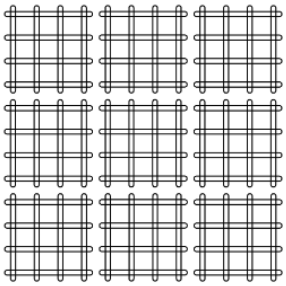
D-Wave Systems developed the quantum graphs that were considered for this work. As explained, qubits connect to others via couplers; nonetheless, D-Wave QPU[9] graphs are not fully connected; rather, qubits connect according to the selected *topology*[10]. The subgraph of qubits and couples available for computing is a *working graph*. Figure 1.2 depicts some of the topologies developed by D-Wave. For example, Chimera can be employed under 2,048 and 6,016 couplers, whereas Pegasus is the latest and most powerful structure by the time this work is written, with 5,640 and 40,484; Pegasus'

[7] The quantum graph represents the qubits and their connections.

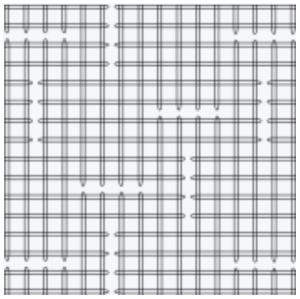
[8] Quantum computing company with headquarters in British Columbia, Canada. D-Wave Systems Official Webpage: https://www.dwavesys.com/

[9] QPU is a lattice of connected qubits.

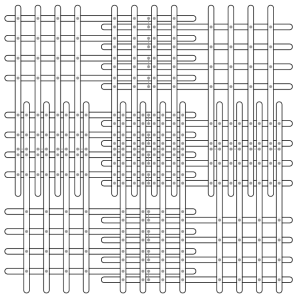
[10] D-Wave's graph architectures. In D-Wave Systems, the topology is the structure where the qubits and couplers are set up on the chip.



(a) Chimera Topology. Released 2017.



(b) Pegasus Topology. Released 2020.



(c) Zephyr Topology (D-Waves' next generation QPUs, release planned for 2023-2024).

Figure 1.2: D-Wave most import topologies. Qubits are tiled vertically or horizontally. a) D-Wave 2000Q QPU, supports 2,048 qubits mapped into a 16×16 matrix of unit cells of 8 qubits. Graph degree of 6 [4]. b) Advantage QPUs, Pegasus qubits are shifted allowing each qubit to have a degree of 15 [7]. c) Zephyr graph, contrary to Pegasus, achieves a degree of 20. By June 2022 an experimental prototype is available in D-Waves Leap, quantum cloud service from D-Wave [8]. Figure 1.1, b) depicts an example of Zephyr's degree 3 connectivity topology.

latest version was released in 2021 under the name *Advantage QPUs* [4]. Siemens provided financial support to test the experiments on the D-Wave Advantage QPUs. Further details concerning the applicability of this technology to our problem are explained in Chapter 4 4.

Holding a more extensive topology structure facilitates the mapping. However, finding minor-embeddings is already considered an NP-hard problem if the QUBO and the quantum graph are not fixed [9]. Different heuristic methods are used to solve the problem. Throughout this project, we must work with binary decision variables precisely due to the embedding procedure. Finally, once the quantum graph is embedded, quantum mechanics is used to finding the minimal energy through quantum tunneling. Quantum tunneling is achieved via qubits superposition and coupling. Figure 1.3 depicts a simplified effect of quantum tunneling in a system, in which depending on

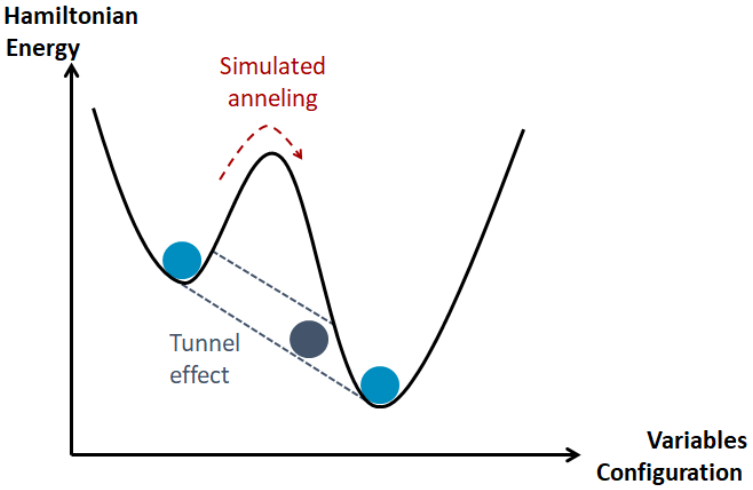


Figure 1.3: Comparison diagram between simulated annealing and using the tunneling effect in quantum annealing. Adapted from Wang et al. [10].

the variable's configuration, coupling affects the system's energy.

1.3 Contributions

The main contributions from this study are as follows:

1. A VRP optimization model was designed according to a real-world operation.
2. Implementation of a four-month simulation and a performance evaluation of an adjusted VRP; model based on Siemens' logistics requirement operation.
3. Evaluate the QA scope applied to a vehicle routing class optimization problem variant.
4. Finally, an extension of the VRP-solving between the classical approach and quantum annealing is presented, providing an overview of the state-of-the-art performance in solving these combinatorial optimization problems in terms of the quality of the solution and time.

1.4 Outline

The remainder of this thesis is structured as follows: **Chapter 2** provides an overview of the evolution of VRP and its approach to solving real-world problems. In there, variants of the VRP and its solution strategies, such as heuristics methods employed

to address this problem's complexity, are reviewed. Subsequently, modeling results through quantum computing and quantum annealing with a particular focus on the applicability of this technology for solving VRPs are studied. In **Chapter 3** , three sections are presented. The first one studies the mathematical framework for the classical Capacitated Vehicle Routing Problem with Time Windows (CVRPTW). Afterward, a modification of the CVRPTW formulation is presented, and lastly, the mathematical framework required for the application of QA is defined. **Chapter 4** is divided into two main components. The first describes the experimental framework, from the procedure for modeling the adjusted CVRPTW to a detailed explanation of QA's implementation. The last part presents the experiments conducted for the adjusted CVRPTW. The classic approach is evaluated as a benchmark for results obtained when utilizing QA. Both in terms of time and solution quality. The final **Chapter 5** summarizes the insights gained throughout this work and assesses the limitations and scope of using QA technology for solving a class of VRPs.

Chapter 2

Literature Review

This chapter outlines essential studies regarding the topics studied in the thesis found in the literature: VRP and its variants, as well as QC coupled with the QA meta-heuristic. The organization of this chapter is divided into two major parts. Section 2.1 presents several models related to the Vehicle Routing Problem, whereas, Section 2.2 explains quantum computing technology and its applications.

2.1 The Vehicle Routing Problem

In a Vehicle Routing Problem, the manufacturer must optimize its routes' distribution to deliver its products to a group of customers. The first formal record for this problem was by Dantzig et al. [11] in 1954. The authors introduced a solution for the TSP for 48 cities and one vehicle. The primary objective of the TSP is to find the shortest tour[1] to visit n cities a single time, except the origin city, which is the starting and returning point. The TSP is one of the most widely studied NP-hard combinatorial optimization problems [12]. Some authors consented that the first introduction of an additional *salesman* was accomplished in Clarke et al. [13], opening the path for a multi-Traveling Salesman Problem (mTSP). Different versions for addressing the mTSP were developed afterward, e.g., Gillett et al. [14] introduced a heuristics method in 1974, and Husban [15] presented a solution approach using branch-and-bound in 1989.

In the taxonomy review from Eksioglu et al. [16], is mentioned that the first time the word *vehicle* was used in a research paper was in 1977 by Golden et al. [17]. This work is considered the first time the VRP was introduced in the literature as we know it today. The introduction of the *vehicle* concept opened space for different notions, such as *vehicle capacity* in terms of volume or weight, since the TSP or mTSP is formulated with no capacity constraints. Therefore, during this research, it may be used the term Capacitated Vehicle Routing Problem (CVRP) for the extended mTSP with m homogeneous trucks [18]–[20]. Conversely, it is also possible to find studies considering a heterogeneous fleet of vehicles [21], [22]. Additional studies introduce a

[1] Route driven by each vehicle, from the first supplier until the arrival at the depot.

probabilistic guideline to the VRP to address real-life challenges of this problem, such as probabilistic capacity constraints or locations due to parking availability [23], [24].

In 1988, Solomon [25] introduced one of the most important variants of the classical VRP, according to the current focus problem setting, namely the *window constraints*. The VRP with Time Window constraints is its generalization, in which a customer's service can begin within the time window defined by the earliest and the latest times allowed by the customer [26]. Solomon [25] remarked that time window constraints alter the computational complexity of even *easy* problems.

A distinct class of optimization problem is called Vehicle Routing Problem with Pick-up and Delivery Problem (VRPPD). VRPPDs belong to the VRP family, where goods must be transported from an origin to a destination; the sole distinction is that the goods are allowed to leave or enter the vehicle at any node [27]–[29]. One typical example of a VRPPD is courier transportation. The class of VRPPDs can be classified into three different groups [30]. The first group consists of *many-to-many* problems, inhere any node can serve as a source (pick-up) or as a destination (deliver) for any commodity. The second group is known as the *one-to-many-to-one*. Here, commodities are initially picked up at the depot and are destined for the customer; in addition, commodities available at the customers for pick-up may be destined to the depot. This setting is highly similar to the combination of an inbound and outbound logistic service. Finally, the third group are *one-to-one* problems; each commodity is picked up and delivered to a given destination. The problem described in this study corresponds to a *one-to-one* service, in which each supplier requests a commodity pick-up whose final destination is one common depot. The pick-up and delivery problem can be extended into a Vehicle Routing Problem with Backhauls (VRPB), which includes a set of customers and vendors whose products are delivered and picked up back to the distribution center, respectively [31].

Several algorithms such as *meta-heuristics*, and problem-independent heuristics, were developed to address this problem's complexity. For instance, one of the earliest publications that used Simulated Annealing (SA)[2] to solve this problem can be found in Alfa et al. [33]; the most known papers include Osman [34] and Lim et al. [35]. Another well-known meta-heuristics technique is the *genetic algorithm*, a biological evolution optimization model, introduced by Filipec et al. [36]. The *ant colony systems*, an optimization model approach that consists of finding optimal paths through a graph [37]. Although nowadays neural networks are a hot topic, in 1996, Ghaziri [38]

[2] A probabilistic method for solving optimization problems. Simulated Annealing "works by emulating the physical process whereby a solid is slowly cooled so that when eventually its structure is *frozen*, this happens at a minimum energy configuration" [32].

proposed utilizing this approach. The development of the VRP addressed with QA will be reviewed in 2.1.

The frameworks presented above considered operating with a single depot and multiple suppliers. However, a more convenient strategy relies upon the assumption of more than one manufacturer in the goods distribution, namely Multi-Depot Vehicle Routing Problem (MDVRP). One of the earliest studies of this branch can be found in Fleischmann [39]. This guideline presents an advantage for large enterprises with several manufacturers and customers in various cities. A proposed two-step approach, where a customer is first assigned to a cluster of depots and secondly performs the route optimization, is introduced in Giosa et al. [40]. A comparable heuristic procedure based on the problem decomposition was done by Min et al. [31] in 1992. It allowed solving the problem with three depots and 134 customers. As reviewed, several authors employed multi-level composite heuristics to address this problem setting, e.g., for a heterogeneous fleet [41]. Alternatively, Lim et al. [35] involved SA for solving MDVRP with a fixed distribution of vehicles. Compared to the best-known results published at that time, the SA yielded a reduction of 12% in the total distance. One of the most robust model proposals for the VRP family reviewed in this project was found in Pisinger et al. [42]; which studied the Multi-Depot Vehicle Routing Problem with Pick-up and Delivery (MDVRPPD), together with the variants of time windows and vehicle capacity MDVRPPDTW. The lector may recall Roman numerals. There are less exact algorithms for solving the MDVRP, [43]–[46] proposed exact branch-and-bound algorithms, performing well only in relatively small settings. Further studies were performed extending this problem, for example, the variant with pick up and deliveries and heterogeneous fleet [47], or one of the most studied problem setting, the Capacitated MDVRP [48]. Fig. 2.1 depicts a hierarchy of the VRP variants discussed.

Finally, the last type of problem studied in this literature review regards a *static* and *dynamic* approach. It will be considered a *static* VRP when all the input data of the problem are known before the routes are constructed. Conversely, it is called *dynamic* VRP if only part of the data involved in constructing the routes is known, and the rest of the information is updated throughout the process. Bianchi [49] defines a problem as *static* if the decision maker can predict the updates for the input data, i.e., although the data may not be deterministic, it is still possible to solve the problem based on probabilistic assumptions. The problem is *not dynamic*. A Dynamic TSP is introduced in Bianchi [49]. Nowadays, *dynamic* routing planning is needed due to the possibility to foreseen traffic congestion [50]; moreover, fast inter-communication allows updating the available information to optimize the route in real-time [51].

In recent years, the development of machine learning algorithms, such as deep learning, promises to leverage challenges when solving vehicle routing optimization

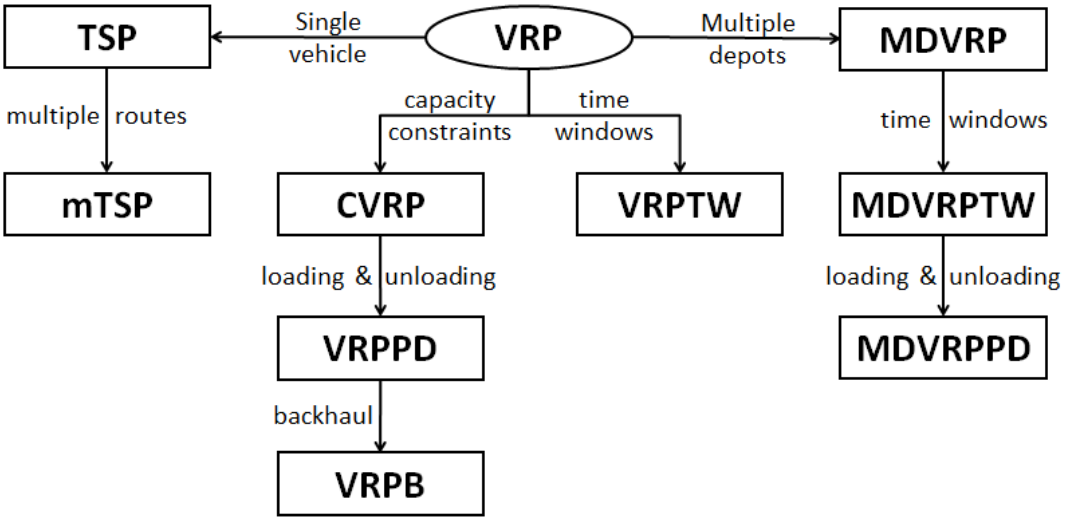


Figure 2.1: Subset of reviewed VRP variants (adapted from Eksioglu et al. [16]).

problems. However, these methods need a vast amount of training data before achieving satisfactory performance. To address the need for training data, Lucas et al. [52] proposed a two-step approach: create a decision tree based on feature characterization from previously generated solutions and then use them as an input for the solver. Due to this, deep reinforcement learning has been demonstrated to be a viable option for addressing the VRP, since the training dataset is not required. For instance, employing a neural network-integrated with an attention mechanism empowers the policy in deep reinforcement learning to select the subsequent nodes automatically [53].

The VRP is one of the most studied combinatorial problems, and as reviewed, it has always been a trial benchmark for applying new developments. The following section provides a literature review of a technology that could be employed to speed up current VRPs solvers.

2.2 Quantum Computing

An additional part of this project consisted of testing the behavior of the VRP with non-conventional computing. This section overviews significant developments concerning quantum computing, especially QA.

Quantum Computing

Quantum Computing is mainly distinct from CC, since it leverages specific properties described by quantum mechanics to perform the computation [54]. It is acknowledged that Richard Feynman introduced the concept of QC. However, in 1979 Benioff [55] presented the theoretical basis for QC and how such a computer could be built. Shor's factorization algorithm is one of the first and most famous algorithms demonstrating QC's better performance than CC [54]. Factoring large numbers is intractable on CC. Prime factorization forms the basis for several proposed cryptosystems [56]. QA performance is also demonstrated in the breaking speed of the Rivest-Shamir-Adleman (RSA) algorithm [57], which proves the optimality of QC for solving this type of algorithm more than CC [56]. Further applications of QC have been studied in the literature, such as quantum speed-up for Monte-Carlo sampling [58] or in chemistry [59]. Several benefits encountered throughout this technology review are its application in communication [60]–[62]. Biamonte et al. [63] studied software implementation paths that may enable machine learning to run faster in QC than in CC. The superposition of qubits, for example, enables QC to be a suitable option for handling large tensors [64]. An evaluation of quantum advantages can be found in Harrow et al. [65].

Quantum Annealing

It was in 2011 when D-Wave Systems announced "the world's first commercially available quantum computer" [3]. In contrast, IBM 2016 was the first company to release quantum computer services onto the cloud with a five-qubit quantum processor [66]. Real-world applications of QA are related to finding a solution to combinatorial optimization problems. The notion of using the quantum tunnel effect to include QA rather than thermal into optimization modeling appears to have been independently proposed in McGeoch [67]. Well-known research performed by Kadowaki et al. [68]used the superiority of QA over SA on a particular system. Further research regarding QA technology has been performed; for example, Fujitsu developed an algorithm inspired by QC technology called Digital Annealing (DA), which represents a more affordable alternative to QA. It is designed to solve fully connected QUBO problems. DA proved to be two orders of magnitude faster than SA in solving dense problems, whereas it does not exhibit a speed-up for sparse ones [69]. Ayodele et al. [70] proposed an extension of the single-objective DA algorithm to a multi-objective one.

Successful quantum optimization stories from D-Wave's customers can be found in D-Wave [71], including portfolio risk management, protein design, and CO_2 reduction caused by waste control optimization. Likewise, quantum annealers proved to be helpful in finance [58] and weather forecasting [72]. Further problems solved using QA in the

[3]D-Wave Systems Official Webpage: https://www.dwavesys.com/

industry are addressed by Yarkoni et al. [73].

The TSP is one of the most studied problems addressed using QA from the VRP family [74]–[76], VRP and its variants [77], [78] or a dynamic approach of the MD-VRP, see Harikrishnakumar et al. [79]. However, QA does not always present better performance than CC; the algorithm additionally encounters time delay issues for embedding onto the qubit graph, the latency of the Internet connection, and queuing at the quantum hardware [6]. Borowski et al. [80] introduced a hybrid quantum algorithm solver called DBSCAN. This solver's idea is to disjointly split the problem in minor instances, proven to find faster results and can find solutions of similar or even better quality than the tested classical algorithms. Feld et al. [6] proposed a hybrid guideline of the Capacitated VRP, which studied the behavior of the QBSolv using QA and a classical solver. The authors observe that QBSolv is currently more advantageous when using classical methods, since finding a proper embedding of the QUBO problem to the hardware generates overhead costs.

Currently, research work has emerged that solves machine learning methods using QA. For example, Nath et al. [81] presented a review of D-wave QA for optimizing machine learning algorithms in classification problems. On the other hand, the limitation of QA due to a need for flawless control of the processor environment or hyperparameter tuning motivated a research line; in which machine learning algorithms improve QA Brence et al. [82]). An application of deep learning Monte-Carlo simulations technique for speeding up sampling [83], and similarly, the use of reinforcement learning to adapt the penalty term of unsatisfied constraints and adjust the given problem into a new Hamiltonian was studied by Ayanzadeh et al. [84].

Contrary to VRP's literature, this technology has fewer tested applications, making it an excellent study object. There exist findings demonstrating that QA has confirmed usefulness in particular scenarios, such as encryption and combinatorial optimization problems. Several developments in this technology are expected, such as D-Wave's Advantage2 with 7,000 qubits and a new qubit design (Zephyr) [4], given its endorsement by companies and governments in recent years [85]. The following chapter explains the mathematical framework employed to model the VRP and construct the QUBO problem required to use D-Wave's QA software.

Chapter 3

Problem Formulation

This chapter contains the mathematical formulations employed during this study. After the Literature Review from Chapter 2, a selected a baseline of the route optimization model that suits the project in Subsection 3.1.1 was chosen; nonetheless, one can find assumptions in this model that do not entirely align with the scheme requirements. In order to address the particular demands of the operational logistics process from Siemens, the adjustments performed are outlined in Subsection 3.1.2.

On the other hand, this work aims to understand the scope of Quantum Annealing applied to the VRP optimization problem. Therefore, a modification of the MIP formulation into a legible structure for Quantum Annealing is needed. This modification consists of mapping the optimization problem onto a QUBO problem Feld et al. [6]. The latter type of problem is solved with Ising machines[1], where the energy function of the Ising model or QUBO form corresponds to the objective function [87]. The physical process for solving this instance is known as Quantum Annealing based on the principles of Adiabatic Quantum Computing (AQC)[2]. Therefore, Section 3.2 describes the steps to obtain its Hamiltonian formulation and then proposes a solution for reaching the Quantum Annealing final formulation, namely QUBO.

3.1 Multi-Vehicle Routing Optimization

This section consists of two components. According to the literature reviewed, the first one defines the VRP that describes reasonable and practically the faced problem. Afterward, an adjustment that fulfills additional prerequisites of the selected VRP is presented.

[1] Ising machines are hardware solvers whose objective is to locate an approximate ground state of the Ising model [86].

[2] "An adiabatic quantum computing is a model of computation that uses quantum mechanical processes operating under adiabatic conditions" [88].

3.1.1 The Capacitated Vehicle Routing Problem with Time Window

In Chapter 2, the limitations and advantages of members from the family of VRPs were discussed. Their objective is to determine a set of vehicle routes to perform the transportation requests; in particular, to decide which vehicle handles each sequence request so that all routes can be feasibly executed as proposed by Irnich et al. [89].

The CVRPTW is described in Kallehauge et al. [90] as a fleet of identical vehicles $\mathcal{V} = \{1, ..., k\} \subsetneq \mathbb{N}$ located at a central depot, represented by the set $\{0, n+1\} \subsetneq \mathbb{N}$, that must be optimally routed to supply a set of customers $\mathcal{C} = \{1, ..., n\} \subsetneq \mathbb{N}$. Each customer is served exactly once. In this setting, there exists a known demand $q \in \mathbb{R}^n$ of goods of each customer. The set of all nodes is denoted by $\mathcal{N} = \{0, 1, ..., n, n+1\} \subsetneq \mathbb{N}$. This relation is described as the complete and undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{A}$ is the set of arcs between nodes i and j. Each arc $(i, j) \in \mathcal{A}$, with $i \neq j$, reflects a time τ_{ij} associated with the distance between the nodes, $d_{ij} \in \mathbb{R}^+$. Additionally, $i \neq n+1$ since $n+1$ must be the last stop for all the vehicles, and similarly, $j \neq 0$ indicates that at the starting point, no vehicle is allowed to arrive at the depot, when is node 0. In other words, every route originates at node 0 and finishes at node $n+1$.

Each customer has a scheduled service time assigned to only one vehicle. The service for any customer starts within a given time interval, called a time window. There are two types of time window classification: *i*) they are called *soft* when the vehicles are allowed to arrive to the customer before the earliest and after the latest predefined time window; and *ii*) hard, if a vehicle must wait until the customer opens when arriving too early and not reach the customer if the arrival time is after the upper bound. In this problem description, the latter is modeled.

The objective of the CVRPTW is to serve several customers, each within a predefined time window $[e_i, l_i] \subsetneq \mathbb{N}$, minimizing the distance traveled. Every vehicle must arrive at the customer before b_i, and wait if it arrives before e_i to serve the customer. The vehicle may not leave the depot before e_0 and the last vehicle must return to the depot before l_{n+1}. Note that the interval from e_0 to l_{n+1} describes the *scheduling horizon*. For simplicity, assume $e_0 = 0$. All requirements must be fulfilled without exceeding the truck's maximum capacity c, homogeneous fleet, such that the demand q_i of customer i is bounded by $q_i \leq c$. Note that the minimum number of vehicles needed to serve all customers is $\lceil \frac{\sum q_i}{c} \rceil$ [3].

The formulation considers two decision variables. The indicator variable:

$$x_{i,j,k} = \begin{cases} 1, & \text{if vehicle } k \text{ drives from } i \text{ to } j, \\ 0, & \text{otherwise,} \end{cases} \tag{3.1}$$

establishes the connections in the graph $\mathcal{G}$. In addition, for each node i and vehicle k, the decision variable $s_{ik} \in \mathbb{R}^+$ specifies vehicle $k's$ starting service time at a customer i; and no vehicle arrives at more than one customer. Therefore, s_{ik} only is considered when $x_{i,j,k} = 1$ and it is irrelevant otherwise. Regarding the depot, without loss of generality, assume $s_{0k} = 0$.

Finally, the mathematical formulation for the *VRP with homogeneous truck Capacity and hard Time-Windows* (CVRPTW) is the following:

$$\min_{x} \quad \sum_{k\in\mathcal{V}}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} d_{ij}x_{ijk} \tag{3.2a}$$

$$\text{subject to} \quad \sum_{k\in\mathcal{V}}\sum_{j\in\mathcal{N}} x_{ijk} = 1, \qquad \forall \quad i \in \mathcal{C}, \tag{3.2b}$$

$$\sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{N}} q_i x_{ijk} \leq c, \qquad \forall \quad k \in \mathcal{V}, \tag{3.2c}$$

$$\sum_{i\in\mathcal{N}} x_{0jk} = 1, \qquad \forall \quad k \in \mathcal{V}, \tag{3.2d}$$

$$\sum_{i\in\mathcal{N}} x_{ijk} - \sum_{i\in\mathcal{N}} x_{jik} = 0, \qquad \forall \quad j \in \mathcal{C}, \quad \forall \quad k \in \mathcal{V}, \tag{3.2e}$$

$$\sum_{i\in\mathcal{N}} x_{ijk}x_{i,n+1,k} = 1, \qquad \forall \quad k \in \mathcal{V}, \tag{3.2f}$$

$$x_{ijk}(s_{ik} + \tau_{ij} - s_{jk}) \leq 0, \qquad \forall \quad i,j \in \mathcal{N}, \quad \forall \quad k \in \mathcal{V}, \tag{3.2g}$$

$$e_i \leq s_{ik} \leq l_i, \qquad \forall \quad i \in \mathcal{N}, \quad \forall \quad k \in \mathcal{V}, \tag{3.2h}$$

$$s_{ik} \in \mathbb{R}^+, \qquad \forall \quad i \in \mathcal{N}, \quad \forall \quad k \in \mathcal{V}, \tag{3.2i}$$

$$x_{ijk} \in \{0,1\}, \qquad \forall \quad i,j \in \mathcal{N}, \quad \forall \quad k \in \mathcal{V}. \tag{3.2j}$$

The objective function (3.2a) minimizes the total travel distance. Constraint (3.2b) guarantees that each customer is visited only once. Conversely, (3.2c) assures that a vehicle can only be loaded to its maximum capacity. Equation (3.2d) ensures that each vehicle must leave the depot 0, following equation (3.2e), known as the *balance constraint*, since each vehicle that arrives at a customer must leave it. Then, (3.2f) indicates that all vehicles must finish at the depot $n+1$. The inequality (3.2g) specifies the relationship between the vehicle departure time from a customer and the vehicle's next customer. Finally, constraint (3.2h) attaches the service time within the allowed

time window. Finally, (3.2i) and (3.2j) defined the decision variables description. The formulation employed during the initial testing substituted equation 3.2g with 3.2k.

$$s_{ik} + \tau_{ij} - M(1 - x_{ijk}) \quad \leq \quad s_{jk}, \qquad \forall \quad i, j \in \mathcal{N}, \quad \forall \quad k \in \mathcal{V} \tag{3.2k}$$

for a natural number M big enough, such that $M \geq \max_{ij \in \mathcal{N}}\{l_i - e_j + t_{ij}\}$ [91], known in the literature as the *Subtour Elimination Constraint*. The service start variables impose a unique route direction for each vehicle, thereby eliminating any subtours. Note that $s_{ik} \leq s_{jk}$ if $x_{ijk} = 1$ [90]. A study regarding other variants of this problem can be found in Nanda Kumar et al. [3]; where variable s_{ik} is replaced with another two variables u_i and w_i representing the arrival and waiting time at customer i respectively. Note that this model approach considers the soft time windows described above.

Due to the type of variables used, an entire procedure with this formulation was not selected. The problem setting described in this Section utilizes non-binary variables. Recall that binary variable to solve the problem using Quantum Annealing is needed. In addition, this study is interested in utilizing volume constraints and minimizing the number of vehicles employed.

3.1.2 CVRPTW Adjusted Formulation

Parallel to the problem setting presented previously, a set $\mathcal{N} = \{1, ..., n\} \subsetneq \mathbb{N}$ of suppliers distributed in different locations is considered. In the same manner, a unique depot is used. Nevertheless, throughout this work, it will be referred to it only as $\{0\} \subsetneq \mathbb{N}$ without using the index $n + 1$. The notation utilized for the complete set of nodes is now represented as $\mathcal{N}_0 := \mathcal{N} \cup \{0\}$.

An initial set of vehicles $\mathcal{V} = \{1, ..., n\} \subsetneq \mathbb{N}$ to transport the commodities is considered in both problem settings. Similarly, these vehicles had the same loading capacity, $c \in \mathbb{R}^+$, and respective volume, $m \in \mathbb{R}^+$. All the vehicles depart from only one supplier and are allowed to visit each location within a predefined time window once. Recall that, unlike in the last Subsection, all the vehicles depart only from the depot. Finally, each of these vehicles must end its tour at the depot. Throughout the Section, a rigorous description of the performed changes will be provided.

Time-indexed framework

The framework is constructed following a Service Network Design (SND) problem [92]–[94] due to its advantages when using binary variables. Here, the time is not

modeled as a positive real number, (3.2i), but is discretized and assigned to an element of the network, namely a time-expanded network. The implemented model employs a 5-index formulation (3.4), which consists in describing the route with 5 elements in a decision variable $x^k_{itjt'}$, such that:

- k represents the vehicle used
- i represents the departure node
- t represents the arriving time at node i
- j represents the arrival node
- t' represents the arriving time at node j

In Subsection 3.1.1, an introduction to a 3-index formulation was explained. Additional strategies that use the time-index framework can be found. Albiński et al. [95] studied a 4-index formulation; this approach reduces the computational effort when solving this problem. Related studies proposed different time-extended network analyses by considering additional decision variables, such as *clock variables*, to follow the truck's driving time. They achieve it by counting the time after breaks or daily breaks [91], [95], e.g., 8 hours of driving with breaks of 30 minutes after 4 hours of continuous driving.

Firstly, it is constructed as a standard connection network $\mathcal{G} = (\mathcal{N}_0, \mathcal{A})$, where $\mathcal{N}_0$ denotes the nodes of graph G representing the suppliers and the depot. Then, the arrival time windows $[e_i, l_i] := \{e_i, e_i + 1, ..., l_i\}$ for each supplier $i \in \mathcal{N}$ using the requested pick-up hour from the data received is created. In this regard, it is assumed that the arrival times follow the principle $First - In$, $First - Out$, which states that a later arrival l_i cannot lead to an earlier arrival e_i and that the travel time is a piece-wise linear function of departure time [96]. All possible options within this interval using the opening hours of the supplier and a predefined padding time t_w, giving an interval of $\pm\frac{t_w}{2}$ hours around the requested pick-up time are given. For instance, $t_w = 12$ hours would give an additional buffer of ± 6 from the requested supplier's pick-up time if these hours were elements of the supplier's opening hours. The same situation for the depot's arrival time window $[e^i_0, l^i_0]$ is considered; using the requested depot delivery hour, the arrival time in the depot according to each supplier's requested order deadline is generated. Note that all the vehicles must be in the depot at a time no later than $\max_{\forall i \in \mathcal{N}_0}\{l^i_0\}$.

As mentioned at the beginning of this chapter, the CVRPTW is modeled with various modifications. One adjustment is related to vehicles starting positions. In the classic CVRPTW, all vehicles start in the depot and return to it at the end of the tour. However, this problem setting has the advantage that any transport can start at the most convenient supplier. Therefore, the time matrix $\mathcal{T}$ is modified by converting all

entries of the first row to zero. This is a valid transformation since this row can be associated with the transit from the depot to each supplier. Let $\tau_{ij} \in \mathcal{T} \subsetneq \mathbb{R}^{+^{n \times n}}$ be the transition time for going from i to j in an HGV according to the suppliers' locations.

Then, the latter modification is given by:

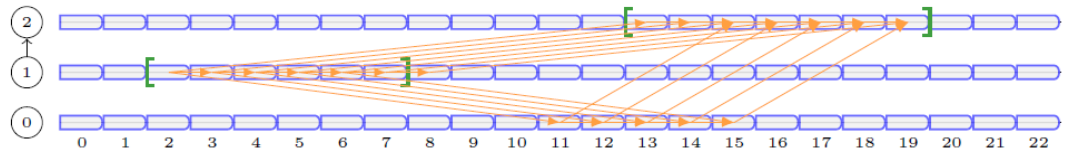
$$\tilde{\mathcal{T}} := \begin{pmatrix} 0 & 0 & \dots & 0 \\ \tau_{10} & \tau_{11} & \dots & \tau_{1n} \\ \vdots & & \ddots & \vdots \\ \tau_{n0} & \tau_{n1} & \dots & \tau_{nn} \end{pmatrix} = \begin{pmatrix} \tau_{00} & \tau_{01} & \dots & \tau_{0n} \\ \tau_{10} & \tau_{11} & \dots & \tau_{1n} \\ \vdots & & \ddots & \vdots \\ \tau_{n0} & \tau_{n1} & \dots & \tau_{nn} \end{pmatrix} \circ \begin{pmatrix} 0 & 0 & \dots & 0 \\ 1 & 1 & \dots & 1 \\ \vdots & & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}$$

where $\circ$ represents the Hadamard product, also known as the element-wise product.

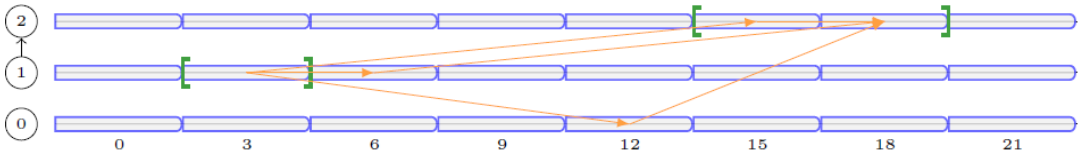
Furthermore, the minimization function depends on the distance between nodes, which also follows $d_{ij} = 0$ if and only if $i = 0$. Similarly, the distance between the nodes is determined by their locations. This modification on the $\tilde{\mathcal{T}}$ matrix also impacts the connections from the graph $\mathcal{G}$. Recall that $\mathcal{A}$ represents the set of arcs between nodes. It can be considered these arcs as travel costs where $\mathcal{A} \subsetneq \mathcal{N} \times \mathcal{N}$ indicates if traveling between two nodes is possible. Note that each arc $a = (i, j) \in \mathcal{A}$ is associated with the travel time between two nodes τ_a.

Assuming vehicle k arrives to supplier i at time t, the arrival time t' at supplier j is given by $t' = t + \tau_{ij}$. Thus, t' is considered feasible if $t' \in [e_j, l_j]$ and added to the set of arcs in $\mathcal{A}$. Furthermore, a *service time* s_i at each supplier is added to the model, obtaining the arrival time at j, $t' = t + s_i + \tau_{ij} \geq 0$. Note that in this formulation the *service time* is a parameter and not a decision variable as in Subsection 3.1.1. This view helps to prune connection options, i.e., if t' turns out to be an element of the supplier's closing hours, and not be considered this connection. Finally, the feasible solution would be a subset of the node-time pairs $(i, t) \in \mathcal{N} \times \mathcal{T}$, where $\mathcal{T}$ represent the time domain.

However, considering all possible options may increase the computational complexity, since it must be given all the options to the network as inputs. Several authors have introduced the concept of *interval discretization* in this setting, among the ones studied in this research were [92], [95], [96]. The idea is to divide the interval into equally distributed time partitions. Selecting an adequate discretization time is challenging, since it can strongly impact the solving time and quality of the solution [96]. For instance, choosing a relatively big *discretization constant* could imply a much faster



Discretized arrival time window interval for $\Delta = 1$ hour.



Discretized arrival time window interval for $\Delta = 3$ hours.

Figure 3.1: Arrival time window length according to two different discretization constants Duc Minh et al. [94].

solution with very low-quality results; on the other side, a small one describes the continuous space better and increases the quality of the result at the expense of running time.

Assuming that the time-space starts in 0, without loss of generality, it is assigned the earliest possible arrival to a supplier j, $\min_{\forall i \in \mathcal{N}_0}\{e_i\} = 0$, and an upper bound $\mathcal{H} := \max_{\forall i \in \mathcal{N}_0}\{l_0^i\}$. The *discretization constant* in this problem is computed using the time matrix T. The idea is to create a trade-off between solution quality and optimal solving time. For instance, in some scenarios studied during the experimental phase, it was encountered situations where each node was 6 hours from the other, i.e., $\min \tau_{ij} = 6$. In this case, a *discretization constant* of less than 6 would only increase the computation cost without increasing the quality of the result. At the same time, in scenarios where the nodes were too near each other, i.e., in the same city and only minutes were needed to reach each other, a relatively big *discretization constant* would substantially decrease the solution quality. Therefore, it is considered a *discretization constant* $\Delta = \max\{\alpha, \min \tau_{ij}\} \in \mathbb{R}^+$, with $\alpha \in \mathbb{N}$ found by trial and error in the testing period. Throughout the experimental phase, Chapter Chapter 4, it was selected an $\alpha = 4$ hours. Assume that the original time interval is $\mathcal{T} = [0, 12]$, then it can be discretized such that $\mathcal{T}_\Delta := \{0, 1, ..., \lceil \frac{12}{\Delta} \rceil\}$. It can be redefined the *discretized interval* $\mathcal{T}_\Delta := \{0, 1, ..., \lceil \frac{\mathcal{H}}{\Delta} \rceil\}$ using a conservative rounding scheme, with $\mathcal{H}$ the planning. Figure 3.1 depicts the difference between different $\mathcal{T}_\Delta$ according to the discretization time given.

Thus far, the index formulation using a standard connection network $\mathcal{G}$ consisted

of location nodes and travel time arcs. From now on refer to a time-node as the union of the time with the location nodes, i.e., instead of considering a node $i \in \mathcal{N}_0$, expanding the nodes to $(i,t) \in \mathcal{N}_{\mathcal{T}_\Delta} := \mathcal{N} \times \mathcal{T}_\Delta$. Therefore, the time-arcs are $((i,t),(j,t')) \in \mathcal{A}_{\mathcal{T}_\Delta} := \mathcal{N}_{\mathcal{T}_\Delta} \times \mathcal{N}_{\mathcal{T}_\Delta}$, such that $t' = t + \lceil \frac{s_i}{\Delta} \rceil + \lceil \frac{\tau_{ij}}{\Delta} \rceil$ and $i \neq j$. As an example, assume vehicle k needs 12 hours for reaching j from i, i.e., $\tau_{ij} = 12$. If the discretization constant is $\Delta = 4$; the adjusted time τ_{ij} would be represented as a $3-$*steps* in time within the network. Similar to the *service time*, assuming 4 hours, it would become a $1-$*step* in time with the given discretization parameter. For simplicity, refer now to $\mathcal{T}_\Delta$ only as $\mathcal{T}$. Both time-nodes and time-arcs define our time-expanded network $G_\mathcal{T} = (\mathcal{N}_{0_\mathcal{T}}, \mathcal{A}_\mathcal{T})$.

Mathematical Formulation

The formulation is firmly based on two studies: regarding the time-expanded network, it was helpful the model solution for the TSPTW proposed by Boland et al. [93]; due to its advantage when constructing the time-expanded matrix. This formulation grants the freedom to add the service time to the time-network. Another practical usage of this setting is the ability to operate only with binary variables. Nevertheless, as explained in Section 2, the TSPTW only addresses the problem of a single vehicle. Applying the transformations presented in Korablev et al. [97] can generalize the TSPTW model to a multi-vehicle problem. The necessary notations used for the mathematical formulation are listed in Table 5.1.

Firstly, define a binary variable, indicating whether vehicle k arrives to pick up material at supplier i at time t, and afterward arrives at supplier j at time $t' = t + \lceil \frac{s_i}{\Delta} \rceil + \lceil \frac{\tau_{ij}}{\Delta} \rceil$. The parameter s_i may include the waiting time needed until attendance and the loading time. As depicted in 3.4:

$$x^k_{(i,t),(j,t')} = \begin{cases} 1, & \text{if vehicle } k \text{ drives from } i \text{ at time } t \text{ and arrives to } j \text{ at time } t', \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

The objective is to minimize the distance traveled by all the vehicles along the suppliers until the depot, through the following function:

$$z_x \quad := \quad \sum_{k \in \mathcal{V},\, ((i,t),(j,t')) \in \mathcal{A}_\mathcal{T} : i \neq j} d_{ij} x^k_{((i,t),(j,t'))}, \tag{3.5}$$

where $d_{ij} \in \mathbb{R}^+$ represents the distance from node i to j.

During the testing phase, the number of given vehicles $k = |\mathcal{V}|$ significantly impacted the time and quality of the solution. Therefore, it was introduced an additional binary variable y_k indicating whether or not vehicle k is used:

$$y_k = \begin{cases} 1, & \text{if vehicle } k \text{ is used,} \\ 0, & \text{otherwise,} \end{cases} \tag{3.6}$$

and its function (3.7), which reduces the number of vehicles.

$$z_y \quad := \quad \sum_{k \in \mathcal{V}} y_k \tag{3.7}$$

Currently, the amount of starting vehicles provided to the model is precisely the number of nodes, seeking not to use them all. This approach also aims to reduce the monetary capital invested during these processes due to the lower need for drivers—not to mention the CO_2 emissions.

Finally, the new minimization objective function (3.8), is formulated as the sum of the costs from the selected arcs and routes plus a penalization for the number of vehicles employed.

$$z_{xy} \quad = \quad \min_{xy} \quad w z_x \quad + \quad (1 - w) \mathcal{P} z_y \tag{3.8}$$

where $w \in [0, 1]$ is the relative weight of costs, i.e. total distance traveled, and the number of vehicles used; w is assumed to be known. A benefit of (3.8) is the opportunity to replace the original objective function, single distance minimization, with a weighted sum of objective functions as proposed by Korablev et al. [97].

Additionally, the penalization parameter $\mathcal{P}$ defined in (3.9) showed an adequate performance during the testing period.

$$\mathcal{P} \quad := \quad \frac{\sum_{j \in \mathcal{N}_0} d_{j0}}{|\mathcal{V}|} \tag{3.9}$$

The foundation of P relies upon the result in which every vehicle visit only the starting supplier and drives directly to the depot. Refer to this as the *trivial solution*. Note that the penalization parameter P is the average distance driven in the *trivial solution*. As a side note, the *trivial solution* serves as a benchmark to evaluate the solution quality from any model tested during the study.

The following equations describe the constraints utilized in the model. Equation (3.10) ensures that exactly one vehicle arrives at each location only one time during its time window.

$$\sum_{k \in \mathcal{V},\, ((i,t),(j,t')) \in \mathcal{A}_{\mathcal{T}} : i \neq j} x^k_{((i,t),(j,t'))} \quad = \quad 1, \qquad \forall \quad (j,t) \in \mathcal{N}_{0_{\mathcal{T}}} \tag{3.10}$$

Balance constraints guarantee that the number of vehicles that reach a timed node equals the number of transports going out of the node; for all suppliers except the depot.

$$\sum_{((i,t),(j,t')) \in \mathcal{A}_{\mathcal{T}}} x^k_{((i,t),(j,t'))} \quad = \sum_{((j,\widetilde{t}),(i,t)) \in \mathcal{A}_{\mathcal{T}}} x^k_{((j,\widetilde{t}),(i,t))}, \qquad \forall \quad k \in \mathcal{V}, \quad \forall \quad (i,t) \in \mathcal{N}_{0_{\mathcal{T}}} \tag{3.11}$$

The following equation designates the number of vehicles allowed to return to the depot.

$$\sum_{((i,t),(0,t')) \in \mathcal{A}_{\mathcal{T}}} x^k_{((i,t),(0,t'))} \quad = \quad y_k, \qquad \forall \quad k \in \mathcal{V} \tag{3.12}$$

Let $q_i \in \mathbb{R}^+$ be the weight of the goods requested to be picked up at supplier i. The next equation maintains the truck's load less or equal than its maximum weight capacity $c \in \mathbb{R}^+$. If this constraint is not fulfilled, the truck will not load any more goods, going straight to the depot.

$$\sum_{((i,t),(j,t')) \in \mathcal{A}_{\mathcal{T}}} q_i x^k_{((i,t),(j,t'))} \quad \leq \quad c y_k, \qquad \forall \quad k \in \mathcal{V} \tag{3.13}$$

Similarly, let $v_i \in \mathbb{R}^+$ be the volume of the goods requested to be picked up at supplier i. The following equation tracks down the volume loaded into the truck, meaning that the vehicle will go directly to the depot if the truck's maximum volume $m \in \mathbb{R}^+$ is reached.

$$\sum_{((i,t),(j,t'))\in\mathcal{A}_\mathcal{T}} v_i x^k_{((i,t),(j,t'))} \quad \leq \quad m y_k, \qquad \forall \quad k \in \mathcal{V} \tag{3.14}$$

As stated above, the decision variables describe the connections between the time-expanded network and the vehicle usage indicator function:

$$\begin{aligned} x^k_{((i,t),(j,t'))} &\in \{0,1\}, \qquad \forall \quad ((i,t),(j,t')) \in \mathcal{A}_\mathcal{T} \\ y_k &\in \{0,1\}, \qquad \forall \quad k \in \mathcal{V} \end{aligned} \tag{3.15}$$

Chapter 4 will explain in further detail the steps used to use this formulation according to Siemens' operational parameters, e.g., the depot's opening and suppliers opening hours. Afterward, Chapter 3 presents the results using an Amazon Web Service (AWS) instance.

3.2 Quantum Annealing

3.2.1 Framework

This section will explain the approach used to construct the Hamiltonian function of the optimization problem. The need for a Hamiltonian formulation to apply quantum mechanics rules and its QUBO form for using quantum annealing is described in Subsection 2.2.

Following the Hamiltonian formulation from Lucas [98], suppose a quantum Hamiltonian H_F whose ground state encodes the solution to a problem of interest on one side. And on the other, an initial state H_I, whose lowest-energy state happens when the qubits are in a superposition state of 0 and 1 [99], i.e., a trivial state to achieve. Then, if a quantum system can gradually evolve from the initial state at $t = 0$ to the final one at $t = \mathrm{T}$, it is proved that the last state will represent the lowest energy, i.e., the solution to the problem of interest. Equation 3.16 describes the evolution function.

$$H(t) \quad = \quad (1 - \tfrac{t}{\mathrm{T}})H_I \quad + \quad \tfrac{t}{\mathrm{T}}H_F \tag{3.16}$$

where $t \in [0, \mathrm{T}]$.

In other words, if it is measured the quantum system at a large enough time T, with T, by the adiabatic theorem [3], the quantum state will return a solution to the problem Lucas [98]. The Hamiltonian function from the mathematical formulation of Section 3.1 is constructed from the principles studied in Harikrishnakumar et al. [79]. In the first place, let us consider the following simple optimization problem.

$$\min_{x} \quad f(x) \tag{3.17a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_x} A_i x_i \quad = \quad b, \tag{3.17b}$$

$$\sum_{i=1}^{n_x} D_i x_i \quad \leq \quad c. \tag{3.17c}$$

where $x \in \mathbb{Z}_2^{n_x}$ represents a binary decision variable, $A, D \in \mathbb{R}^{n_x \times n_x}$ the coefficient matrices and $b, c \in \mathbb{Z}$. According to Lucas [98], the equality constraints with the penalty terms are depicted in 3.17d.

$$\sum_{i=1}^{n_x} A_i x_i = b \quad \Longleftrightarrow \quad (\sum_{i=1}^{n_x} A_i x_i - b)^2 = 0 \tag{3.17d}$$

Quantum annealing uses the QUBO representation of the formulation; thus, it is needed first to transform the inequality constraints into equality ones. This can be done with the aid of slack variables, as in 3.17e.

$$\sum_{i=1}^{n_x} D_i x_i \leq c \quad \Longleftrightarrow \quad (\sum_{i=1}^{n_x} D_i x_i + \sum_{j=1}^{n_\lambda} 2^j \lambda_j - c)^2 = 0 \tag{3.17e}$$

[3] The adiabatic theorem states that a system remains in its state even when being perturbed, as long as that perturbation is slow and gradual enough and there is a gap between the energy of that state and the rest of the energy of the system [100].

where $n_\lambda \in \mathbb{R}^+{}_0$ represents the number of slack variables calculated as $n_\lambda = \lceil 1 + log_2 c \rceil$.

The purpose of introducing the slack variables is to transform inequality constraints into equality ones; by representing all potential values of the decision variable x from 0 to the upper-bound c (3.17c). Nevertheless, the introduction of such variables increases the model's complexity. A proposal to decrease the number of slack variables can be found at Verma et al. [101], where scalar transformations are employed. Notice that the options available in (3.17c) start from $\min D_i$ until c. This motivated to adjust the predefined formula $n_\lambda = \lceil 1 + log_2 c \rceil$ to $n_\lambda = \lceil 1 + log_2(c - \min D_i) \rceil$ in order to reduce the number of possible options. To obtain the final formulation is necessary to add (3.17d) and (3.17e) to $f(x)$.

Lastly, the Hamiltonian needs to be transformed into its QUBO form. The QUBO formulation represents the cost function when using binary variables with a linear and a quadratic term. Even though the QUBO matrix can be computed directly by using the Hamiltonian terms (since various elements go to zero); employed in the python package, PyQUBO [87]. The PyQUBO package considers the following: assume the Hamiltonian function from $\mathcal{G} = ((\mathcal{N}, \mathcal{A})$, undirected graph explained in Subsection 3.1.1, is equation 3.18. And let $x_i \in \{0, 1\}$ be the decision variable of the problem of interest. Then:

$$\mathcal{H}_{test}(x) \quad = \quad \sum_{i \in \mathcal{N}} c_i x_i \quad + \quad \sum_{(i,j) \in \mathcal{A}} d_{ij} x_i x_j \tag{3.18}$$

where $a_i, b_{ij} \in \mathbb{R}$.

According to Zaman et al. [87], the QUBO formulation can be represented in a matrix form $Q \in \mathbb{R}^{|\mathcal{N}| \times |\mathcal{N}|}$, with elements:

$$Q_{i,j} = \begin{cases} c_i, & |i = j, \quad \forall \quad i \in \mathcal{V}|, \\ d_{ij}, & |i < j, \quad \forall \quad (i,j) \in \mathcal{A}|, \\ 0, & \text{otherwise.} \end{cases} \tag{3.19}$$

Then, equation 3.18 is transformed to its QUBO form by using the coefficients Q_{ij}, jointly with the vector form representation of the decision variable $x \in \mathbb{Z}$ and its transport x^T. See equation 3.20.

$$\mathcal{H}_{test}(x) \quad = \quad \sum_{i \leq j} Q_{ij} x_i x_j \quad = \quad x^T Q x, \tag{3.20}$$

3.2.2 Quantum Annealing Input Construction

In accordance with equation 3.16, our Hamiltonian term $\mathcal{H}_0$ is defined from the objective function (3.8). Therefore:

$$\mathcal{H}_0 \quad = \quad \sum_{k\in\mathcal{V},\,((i,t),(j,t'))\in\mathcal{A}} w d_{ij} x^k_{((i,t),(j,t'))} \quad + \quad (1-w)\frac{P}{|V|}\sum_{k\in\mathcal{V}} y_k \tag{3.21}$$

Let C_i, with i $\in [\![1,5]\!]$ be the five constraints given in Subsection 3.1.2, and let $\mathcal{H}_i$ represent its corresponding Hamiltonian terms.

$$\mathcal{H}_1 \quad = \quad \sum_{(j,t)\in\mathcal{N}_{0_\mathcal{T}}} \left(\sum_{k\in\mathcal{V},\,((i,t),(j,t'))\in\mathcal{A}_\mathcal{T}:i\neq j} x^k_{((i,t),(j,t'))} \quad - \quad 1 \right)^2 \tag{3.22}$$

$$\mathcal{H}_2 \quad = \quad \sum_{k\in\mathcal{V},(i,t)\in\mathcal{N}_{0_\mathcal{T}}} \left(\sum_{((i,t),(j,t'))\in\mathcal{A}_\mathcal{T}} x^k_{((i,t),(j,t'))} \quad - \sum_{((j,\widetilde{t}),(i,t))\in\mathcal{A}_\mathcal{T}} x^k_{((j,\widetilde{t}),(i,t))} \right)^2 \tag{3.23}$$

$$\mathcal{H}_3 \quad = \quad \sum_{k\in\mathcal{V}} \left(\sum_{((i,t),(0,t'))\in\mathcal{A}_\mathcal{T}} x^k_{((i,t),(0,t'))} \quad - \quad y_k \right)^2 \tag{3.24}$$

$$\mathcal{H}_4 \quad = \quad \sum_{k\in\mathcal{V}} \left(\sum_{((i,t),(j,t'))\in\mathcal{A}_\mathcal{T}} q_i x^k_{((i,t),(j,t'))} \quad + \quad \sum_{l=0}^{n_q} 2^l \lambda_{lk} \quad - \quad c y_k \right)^2 \tag{3.25}$$

$$\mathcal{H}_5 \quad = \quad \sum_{k\in\mathcal{V}} \left(\sum_{((i,t),(j,t'))\in\mathcal{A}_\mathcal{T}} m_i x^k_{((i,t),(j,t'))} \quad + \quad \sum_{l=0}^{n_m} 2^l \mu_{lk} \quad - \quad m y_k \right)^2 \tag{3.26}$$

$$\begin{aligned} \lambda_{lk} &\in \{0,1\}, \quad \forall \quad l \in [\![0, n_q]\!], \quad k \in \mathcal{V} \\ \mu_{lk} &\in \{0,1\}, \quad \forall \quad l \in [\![0, n_m]\!], \quad k \in \mathcal{V} \end{aligned} \tag{3.27}$$

Here, it was introduced the slack variable $\lambda_{lk} \in \mathbb{Z}_2^{n_q \times |V|}$ in equation 3.25 where $n_q := \lceil 1 + \log_2(c - \min q_i) \rceil$. Similarly, in 3.26 the slack variable $\mu_{lk} \in \mathbb{Z}_2^{n_m \times |V|}$ where $n_m := \lceil 1 + \log_2(v - \min m_i) \rceil$. Then, for ρ, a considerably large positive constant, the overall Hamiltonian is equal to the sum of the individual Hamiltonian terms:

$$\mathcal{H} \quad = \quad \mathcal{H}_0 \quad + \quad \rho \sum_{i=1}^{5} \mathcal{H}_i \tag{3.28}$$

Chapter 4 will discuss how the Ocean package from D-Wave is employed based on the QUBO matrix coefficients, given by the PyQUBO package using equation 3.28. The decision variables used for the final QUBO solution are the ones described in equation 3.15 and 3.27.

Chapter 4

Experimental Study

Throughout this project, the focus was only based on the inbound logistics process, assuming that there were no restrictions regarding the number of drivers or trucks available and no connection between routes traveling to different depots. The studied process corresponds to four months, in which one consignee or depot requests goods from n suppliers. This chapter will describe the methodology developed from the data prepossessing to the solution through our instance's classical and quantum approach. Due to confidentiality, the preliminary Exploratory Data Analysis (EDA) results will be excluded such that this chapter starts with the pre-processing details.

4.1 Experimental Setup

4.1.1 Data Pre-procesing

As mentioned in Chapter 1, CVRPTW proved to be the most helpful framework for solving our project. In addition to the adjustments explained in Subsection 3.1.2, new fields were created to the data received. Figure 4.1 contains the fields used in our study.

String Data Homologation

The first pre-processing step regards character homologation due to the german characters in some cells in the data. In order to do that, a function that helped to change three types of characters: *ä* for *ae*, *ü* for *ue* and *ß* for *ss* was created. Later, as depicted in Figure 4.1, the column Supplier Address based on the supplier street, postcode, and city was added; the same process was done to obtain the column *Depot Address*. Both supplier and depot addresses were used to obtain their coordinates by using them as input for Google Maps API. Finally, two new columns were constructed: the first was the *Requested loading date* using the pick-up date and time, and the second, was the *Requested delivery date* using delivery date and time.

The creation of the geographic coordinates consisted of the following: opening an account in Google Developer Console in order to create a key that allowed to make use

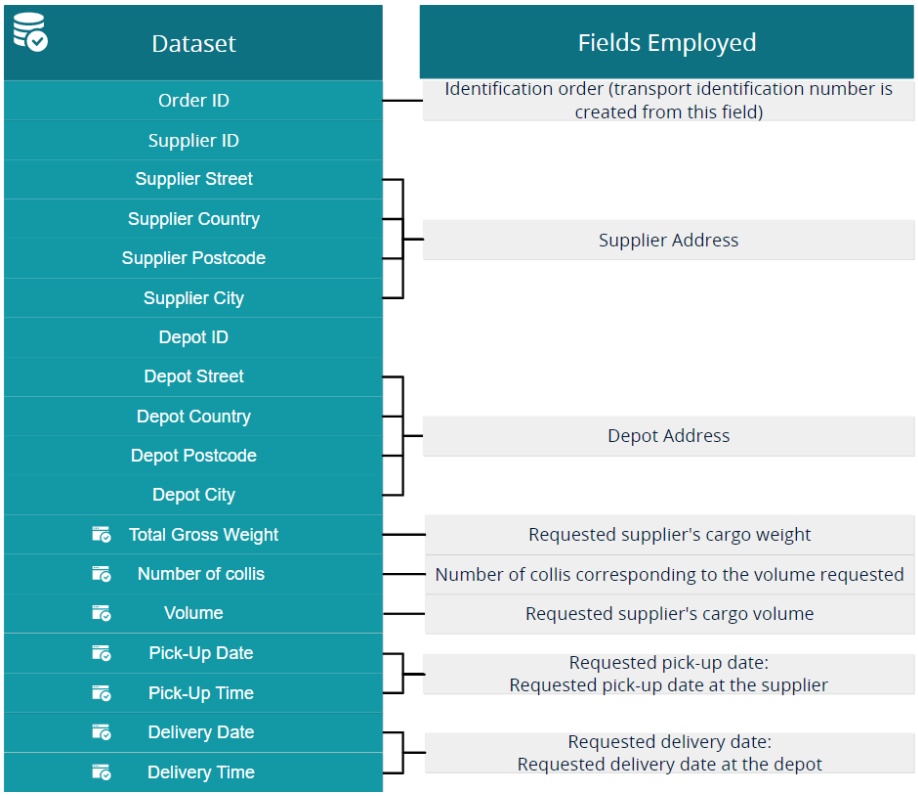


Figure 4.1: Dataset received from Siemens on the left, and on the right, the columns added during the pre-processing step.

of their geocoding[1] tool. Through the package google-maps [102], the latitude and the longitude from these addresses were retrieved. In some cases, the address was insufficient for Google to extract the coordinates. Therefore, only the postcode and city field were used to obtain the supplier address. This alternative is considered feasible since the distance driven in the model was mainly between European cities; the difference generated from these changes would not impact the optimization results. Finally, the located coordinates were plotted to validate their geolocalization, as dispatched in Figure 4.2.

[1]Process known for taking location text-based elements, such as addresses, and returning geographic coordinates.

Figure 4.2: All suppliers included in our dataset are possible to be reached via ground (simulated suppliers' locations).

Data Expansion

In some scenarios, the goods requested to be transported from the depot to the suppliers were heavier than the truck's capacity. The same situation occurred with the requested cargo's volume, in which the volume of the overall goods, per request, was bigger than the truck's volume capacity. These misleading data values are caused due to how the data was recorded in the company's system.

The model reads tabular data in which each row corresponds to one request (per node). Comprehending that some nodes will not fulfill the volume constraints established in the model, a plain copy of each row r_i was created in the following way: Let us say we had a row (requested cargo) with volume v_i requested to be picked-up at supplier i on a predefined date; in this case, we can assume from the data that $v_i \geq m$. Let $cl_i \in \mathbb{N}$ be the *number of collis* [2] corresponding to that good be bigger than one; otherwise, this pick-up would not be viable, since one truck would not be enough for one single colli. Table 4.1 describes the steps used in order to adapt the data to the model's input format.

Throughout the data cleaning phase, two exceptions (aprox.0.02%) were encountered where the respective volume value for one colli was bigger than the complete vehicle's capacity c. This could be due to two main reasons, it was entered wrongly into the system, or the truck's chosen dimension could not handle this cargo. In such

[2] One colli designates the minor packaging units of a consignment of goods.

Nodes Expansion

Step 1: Compute the respective cargo's proportion volume per colli: $p_v = \frac{v_i}{cl_i}$, i.e., p_v represents one colli's volume.

Step 2: Calculate the *maximum number of collis*, according to the maximum volume c, denoted as $m_c = \lfloor \frac{m}{p_v} \rfloor$. Note that $\lfloor . \rfloor$ gives the minimum *number of collis* needed in order to transport the complete volume v_i. This last operation helped us know how many collis we could load into one truck. The main idea is to split one request that exceeds the truck capacity into multiple smaller ones.

Step 3: Given the total *number of collis* requested cl_i and the *maximum number of collis* that can be loaded into one truck m_c, define $r_v := \lceil \frac{cl_i}{m_c} \rceil$ as the number of vehicles needed. Then, create a new **volume** vector $v_r \in \mathbb{R}^{r_v}$, such that $r_v - 1$ elements of v_r are $m_c \cdot p_v$ (maximum volume that can be loaded into one truck, note that $m_c \cdot p_v + \varepsilon = c$ with $\varepsilon \ll$), and one more element equal to $m_c \cdot res$, $res \equiv v_i \mod m$, being the volume corresponding to the remaining collis needed to be transported.

Step 4: In Step 2: we created a new vector with the objective of splitting the cargo's volume. Note that is possible to split the cargo's volume according to the **number of collis**. Therefore, repeat the step in order to create a new vector *Number of collis* $cl_r \in \mathbb{R}^{r_v}$, such that $r_v - 1$ elements of cl_r are m_c and *one* element has value $res \equiv cl_i \mod m_c$.

Step 5: Similarly, the cargo's **weight** needs to be substituted into a new weight vector:

1. Compute the proportion of this weight per colli: $p_w = \frac{w_i}{cl_i}$ as in Step 1:.
2. Create a new *weight* vector $w_r \in \mathbb{R}^{r_v}$, such that $r_v - 1$ elements of w_r are $m_c \cdot w_v$ and *one* more element equal $m_c \cdot res$, $res \equiv w_i \mod m$.

Step 6: Expand the original row r_v with all the original values, only substituting the volume field with v_r, the number of collis with cl_r. and the total weight w_r. See Table 4.2 for a concrete example.

Table 4.1: Performed steps to expand the requested orders into smaller ones according to the maximum truckload volume m and maximum weight capacity c.

scenarios, we assigned one colli to the volume c. Another data quality issue regards the field *delivery date*, in which sometimes the information was not entered into the system, causing missing values in the dataset. Therefore, in order to fill those gaps, it was assumed that the delivery date was *three-days* after the requested picked-up cargo appointment. Table 4.3 summarizes the data pre-processing steps.

4.1.2 Classical Modeling

In the previous section, we described the data modifications needed before being used for solving the VRP. The testing data from Siemens corresponds to almost four months

Example:

	Supplier Address	Pick-up Date	Num. Colli	Volume (m^3)	Total Gross Weight (kg)	...
0	P.Shermann	15.10.2022	350	504	9,000	...

↓

	Supplier Address	Pick-up Date	Num. Colli	Volume (m^3)	Total Gross Weight (kg)	...
0	P.Shermann	15.10.2022	69	99.36	1,774.28	...
1	P.Shermann	15.10.2022	69	99.36	1,774.28	...
2	P.Shermann	15.10.2022	69	99.36	1,774.28	...
3	P.Shermann	15.10.2022	69	99.36	1,774.28	...
4	P.Shermann	15.10.2022	69	99.36	1,774.28	...
5	P.Shermann	15.10.2022	5	7.2	128.57	...

Table 4.2: Expanded dataset according to the maximum volume capacity, $c = 100$. We initially had one row with a volume of $504 >$ c. Therefore, the row was expanded into six rows such that its total volume is still 504; the same matter occurs for the number of collis and the weight.

with multiple consignees and their corresponding suppliers. Even though an MDVRP could be used to propose a solution, it was decided to reduce the problem to only one Consignee. A single depot vehicle routing problem could be seen as a lower optimization bound, since running the algorithm for multiple depots independently would perform poorer than a Multi-Depot VRP. A lower bound that can benefit Siemens' logistics distribution by decreasing the amount of kilometers driven would be sufficient.

The number of suppliers for the selected consignee was almost 150, distributed in four months. Each supplier could have more than one requested pick-up cargo; consequently, throughout the complete evaluation, the same supplier (in location terms) might be deemed a different time node. In order to simulate a realistic scenario, it was decided to split the four months into *sub-simulations* of three days according to the pick-up date, i.e., it was considered to have information three days in advance every time the model was run. Notice that the number of periods evaluated was $\lceil \frac{H}{3} \rceil$, where H is the planning horizon. From now on we will refer to a sub-simulated period as S_i.

Pre-processing Steps
Step 1: Data standardization.
Step 2: Creation of four new columns: *Supplier Address*, *Consignee Address*, *Requested Pick-up Date* and *Requested Delivery Date*.
Step 3: Creation of two new columns corresponding to the supplier and the consignee addresses, *Supplier Coordinates* and *Consignee Coordinates*.
Step 4: Expansion of the dataset in order to fulfill the maximum capacity constraints.
Step 5: If the *Requested Delivery Date* presents missing values, set a three-days time window after the *Requested Pick-up Date* for filling them.

Table 4.3: Subsection summary, data transformation into a suitable input for the MIP model.

Connection Network

The first step was to create its respective distance and time matrix. This was achieved through one function, *matrix routing calculations*, from the Open Route Service [103] package. Among the arguments used in this function is the *driving-hgv*, which returns the duration τ_{ij} between nodes when driving an HGV. The output of the function uses highway routes and not geodesic distances to calculate the distance between nodes, Figure 4.3. Afterward, as explained in Subsection 3.1.2, the first row of the time and distance matrices were transformed to zero, in other words, traveling from the consignee to any supplier did not add costs in the objective function.

The following step regards the construction of $\mathcal{T}_\Delta = \{0, 1, ..., \frac{H}{\Delta}\}$, presented in Subsection 3.1.2. First, consider the following example. Let us say a piece S_i of the entire period is being evaluated and assume we have a set of $n = 3$ suppliers with their respective cargo requests. Without loss of generality, assume that supplier 1 has the earliest pick-up request date, 21.10.2022 07:30, and supplier 3 the latest, 22.10.2022 12:45. Therefore, if we consider a discretization constant of one hour, we assign 0 to the earliest pick-up value at the supplier and 17 to the latest delivery date at the consignee. Note that the latest value corresponds to supplier 3 cargo's latest arrival time at the consignee. The described example is further depicted in Table 4.4.

The time-index set $\mathcal{T}$ contains all possible arrival options according to the given time windows. However, these options are not always feasible, e.g., opening or driving hours. Accordingly, let us create three sets. The first two, $\mathcal{C}_s$ and $\mathcal{C}_d$, correspond to the closing suppliers' and depot's hours; similarly, let $\mathcal{F}_d$ be the forbidden driving hours or driver's testing time. Likewise, as displayed in Table 4.4 the forbidden hours are assigned an integer number between 0 and H, so our final time-index matrix is

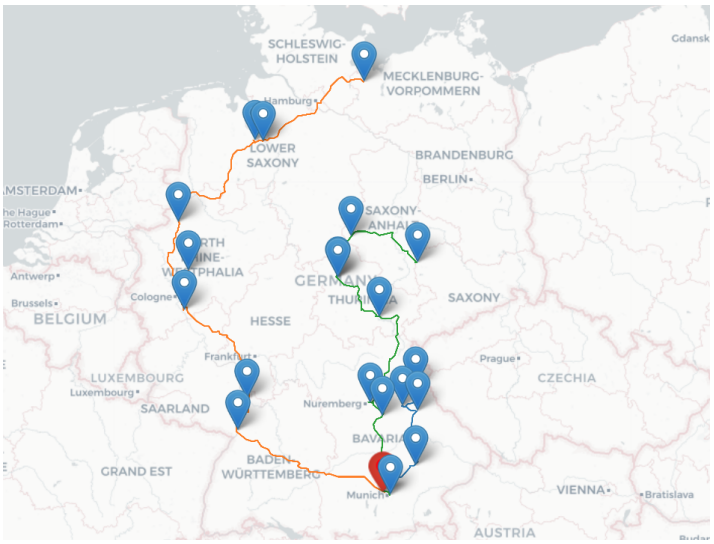


Figure 4.3: Example of simulated routes from S_i using Open Route Service with one depot in Munich, Germany.

$\mathcal{T} \setminus \{\mathcal{C}_s \bigcup \mathcal{C}_d \cup \mathcal{F}_d\}$. For instance, assume supplier i opens at 6 am and its requested pick-up hour is also at 6 am, then allowing the vehicle to arrive at 5 am, from ± 1 hour time window, would not be feasible and thus removed from $\mathcal{T}_\Delta$. Finally, based on the assumptions from Table 4.5, the time-expanded network $\mathcal{G}$ can be created. Recall from Subsection 3.1.1 that $\mathcal{G} = \mathcal{N}_{0_\mathcal{T}} \times \mathcal{A}$ with the nodes $\mathcal{N}_{0_\mathcal{T}} = (N_0, \mathcal{T}_\Delta)$ and $\mathcal{A} = \mathcal{N}_{0_\mathcal{T}} \times \mathcal{N}_{0_\mathcal{T}}$.

Model Solver

The package used to have a CC benchmark against QA is called *OR-Tools*. It is an open-source software developed by Google for combinatorial optimization problems, which seeks to find the best solution out of a large set of options [104]. OR-Tools was chosen at the beginning due to a set of pre-loaded VRP solvers, TSP or CVRP, for naming some. However, specific requests from Siemens were unsuitable to add when using the pre-loaded functions. Additionally, it became challenging to model the constraints these functions were using. Therefore, without a mathematical formulation, it was not possible to make a fair comparison between CC and QA. The solution was to adapt Siemens' requirements by defining a mathematical formulation of the problem based on the CVRPTW, as explained in Subsection 3.1.2. Nevertheless, the package OR-Tools was still used to call the solver Coin-or branch and cut (CBC), an open-source mixed integer linear programming in C++ [105]. The CBC solver was run

Suppliers	Supplier' Pick-up Date
1	21.10.2022 07:30
1	21.10.2022 08:30
1	21.10.2022 09:30
2	21.10.2022 11:00
2	21.10.2022 12:00
2	21.10.2022 13:00
3	22.10.2022 10:45
3	22.10.2022 11:45
3	22.10.2022 12:45

$\Longrightarrow$

$\mathcal{T}_1$	$\mathcal{T}_2$
0	0
1	1
2	1
3	2
4	2
5	3
6	3
7	4
8	4

$\Longrightarrow$

$\mathcal{N}_{\mathcal{T}_1}$	$\mathcal{N}_{\mathcal{T}_2}$
(1,0)	(1,0)
(1,1)	(1,1)
(1,2)	(1,1)
(2,3)	(2,2)
(2,4)	(2,2)
(2,5)	(2,3)
(3,6)	(3,3)
(3,7)	(3,4)
(3,8)	(3,4)

a) Supplier pick-up time-index according to its time window:

Suppliers	Consignee's Delivery Date
1	24.10.2022 07:30
1	24.10.2022 08:30
1	24.10.2022 09:30
2	24.10.2022 11:00
2	24.10.2022 12:00
2	24.10.2022 13:00
3	25.10.2022 10:45
3	25.10.2022 11:45
3	25.10.2022 12:45

$\Longrightarrow$

$\mathcal{T}_1$	$\mathcal{T}_2$
9	5
10	5
11	6
12	6
13	7
14	7
15	8
16	8
17	9

$\Longrightarrow$

$\mathcal{N}_{\mathcal{T}_1}$	$\mathcal{N}_{\mathcal{T}_2}$
(0,9)	(0,5)
(0,10)	(0,5)
(0,11)	(0,6)
(0,12)	(0,6)
(0,13)	(0,7)
(0,14)	(0,7)
(0,15)	(0,8)
(0,16)	(0,8)
(0,17)	(0,9)

b) Consignee delivery time-index according to suppliers' cargo time window, recall that the depot is represented by the index 0.

Table 4.4: Time-index construction. Example for three suppliers with ± 1 arrival time window. Presenting two possible options: $\mathcal{T}_1 := \{0, ..., 17\}$ or $\mathcal{T}_2 := \{0, ..., \lceil \frac{17}{2} \rceil\}$ with their time-expanded nodes. The value of $\alpha = 4$ was found by trial and error for the complete testing period, nevertheless, Δ changes according to S_i.

in the AWS EC2 instance *m6g.8xlarge*[3]. A relative MIP GAP tolerance of 0.1% was set, and no instance S_i was allowed to run more than 90 minutes; if the maximum time was reached, the solver returned the best solution until then.

[3]AWS EC2 instance description: https://aws.amazon.com/ec2/instance-types/m6g/?nc1=h$_l s$.

Network Assumptions
Depot Opening: 7 am
Depot Closing: 4 pm
Supplier Opening: 7
Supplier End: 3 pm
Early arrival: 6 hours (time window)
Late arrival: 6 hours (time window)
Loading time: 2 hours
Driver Starts: 7 am
Driver Stops: 5 pm
Max Weight: 22,000 kg
Max Volume: 196 m^3

Table 4.5: Parameters considered for creating the time-expanded network

4.1.3 QA Modeling

Like the classical solver, QA employed the network explained in Subsection 4.1.2. In order to solve the VRP explained throughout this study with QA, a five-step approach depicted in Table 4.6 was performed.

1) The first relies on the MIP's Hamiltonian formulation, as explained in Subsection 3.2.2. Afterward, it is required to select a proper hyperparameter ρ. Several authors proposed different rules of thumb for selecting ρ. For example, Lucas [98] established the parameter selection based on the constraints' contribution to the change in energy. On the other hand, Feld et al. [6] selected ρ utilizing the costs values, i.e., $\rho = n \cdot max d_{ij}$, with n being the number of suppliers in S_i. The approach used in this work employs $max_c := 2 \cdot max_{i,j \in N_0}(d_{ij})$, testing for different values within the set:

$$l := \frac{1.5 max_c - 0.7 max_c}{10} \tag{4.1a}$$

$$\rho \in I_\rho := \{ \quad 0.7 \cdot max_c + l \cdot m \quad | \quad m \in [\![0, 10]\!] \quad \}, \tag{4.1b}$$

which can be considered as having taken 70% to 150% of twice the maximum distance between nodes.

2) The parameter ρ is problem-dependent, and since QA is non-deterministic, it was necessary to run each S_i several times to find an acceptable solution. Unfortunately, it was not always sufficient to fulfill the solution requirements.

3) The function *to_qubo* from the PyQUBO package was used for obtaining the QUBO coefficients from the Hamiltonian model. Additionally, *to_qubo* has an argument,

VRP solving procedure with QA
Step 1: Convert the MIP formulation into Hamiltonian, Subsection 3.2.2.
Step 2: Select a value for the parameter ρ, equation 3.28.
Step 3: Formulate the QUBO problem according to the output from Step Step 2:.
Step 4: Find an embedding.
Step 5: Solve and evaluate whether or not the solution is acceptable, if not return to Step 2:. It is consider an acceptable solution if non-constraint is violated and its value is less or equal than the trivial solution, $\sum_{j \in \mathcal{N}_0} d_{j0}$.

Table 4.6: Steps followed for solving an S_i instance using QA.

Placeholder, which allows using different ρ values in the same instance without needing to create a new Hamiltonian each time.

4) After different embedding approaches attempts, the function selected was *find_embedding* from the package *minorminer*. This function heuristically attempts to find a minor-embedding of a source graph S into a target graph T [106].

5) The size of the resulting QUBO from Step Step 3: was significant and, on some occasions, exceeded the limit of available qubits on the QPU. The problem was addressed using the function *QBSolv* from D-Wave. It split the QUBO problem into pieces, solved them sequentially, and returned its minimum value. Each piece can be solved using either a classical solver or a D-Wave system solver; Feld et al. [6] addressed NP problems with a similar procedure. In our case, the selected solver was *Advantage_system6*, a physical lattice of qubits and couplers known as Pegasus. The *Advantage QPU* from D-Wave contains 5,640 qubits and more than 35,000 couplers [7]. All the experiments were run in AWS-Braket[4], located in London, UK.

Both procedures share the time-expanded network explained in Subsection 4.1.2 to create either the classical model or the Hamiltonian. The complete simulation is run in sequences of S_i. The first part exclusively solves the Adjusted CVRPTW employing OR_Tools; afterward, the Hamiltonian is constructed and utilized for defining the QUBO problem. Then, the QBSolv function is employed for solving the QUBO problem with D-Wave as the solver, and finally, it is reviewed whether or not the solution is acceptable. Algorithm 1 summarizes the process in this section.

[4] "Amazon Braket is a fully managed quantum computing service designed to help speed up scientific research and software development for quantum computing." [107]

Algorithm 1 General Solver

procedure (N_0, H, A)▷ Where: N_0 - nodes; H - planning horizon; A - model assumptions
  Create a partition of the complete time set $S = \{S_1, ..., S_{\lceil \frac{H}{3} \rceil}\}$
  for S_i in S **do**
    Create the Distance and Time matrix $d_{ij}, \tau_{ij} \leftarrow f_4(N_0)$
    Create Time space $\mathcal{T} \leftarrow f_5(d_{ij}, \tau_{ij}, \mathcal{C}_s, \mathcal{C}_d, \mathcal{F}_d)$
    Discretize the time-space $\mathcal{T}_\Delta = \frac{\mathcal{T}}{\Delta}$
    Create time-expanded network $\mathcal{G} \leftarrow f_7(\mathcal{N}_0, \mathcal{T}_\Delta)$
    Create classical model $\mathcal{M} \leftarrow f_8(\mathcal{G}, A)$
    Solve $\mathcal{M}$
    Store CC solution
    Create Hamiltonian $\mathcal{H} \leftarrow f_{11}(\mathcal{G}, A)$ ▷ Starting QA phase
    for ρ in I_ρ **do**
      while 5 trials are executed **do** ▷ Due to QA's probability nature, more than one trial is preferable
        Define a random seed r
        Get the QUBO matrix $\mathcal{Q}_{ij} \leftarrow f_{15}(\mathcal{H}, \rho)$
        Perform embedding $\mathcal{E} \leftarrow f_{16}(\mathcal{Q}_{ij}, r)$
        Solve QA model $\mathcal{E}$
        if (QA solution $\leq$ trivial solution) **and** (Constraints are fulfilled) **then**
          Store QA solution
        end if
      end while
    end for
  end for

4.2 Experimental Results

This section presents the insights and results obtained after solving the VRP with CC and QA. Intuitively, one might assume that as more nodes are in an instance, the solution time would increase in the same proportion. However, we encountered scenarios where the solver needed less time for instances with more nodes. Recall that a node in the time matrix represents an order to be picked-up at a specific location and arrival time. Therefore, placing multiple orders at the same date and time did not increase the number of connections in the time-expanded network. Figure 4.4 illustrates the number of connections, on average, according to the number of nodes.

For instances S_i with nodes larger than 11 elements, the classical algorithm required more than 90 minutes to obtain the optimal solution. Recall that if an instance S_i

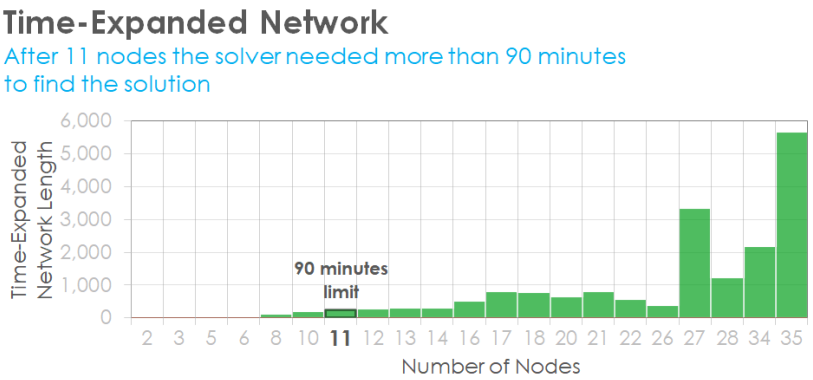


Figure 4.4: Average number of connections (length) in the time-expanded network according to different instances S_i. For example, instances with 26 nodes required less connection than instances with 17 nodes due to time window constraints. The average connections' number of the time-expanded network with 11 nodes is 262 elements. The set of 11 nodes refers to 10 suppliers plus 1 depot.

reached the 90 minutes solving limit, the best result obtained until then was accepted. Throughout this chapter, we will refer to *solving time* as the time required since the model is constructed until a minimal solution is found. The exponential solving time increased when employing the classical solver as depicted in Figure 4.5.

Regarding the QA solver, up to 12 nodes could be solved. It is worth mentioning that there were two instances with seven and nine nodes, respectively, in which the procedures mentioned in this study were insufficient to find a solution through QA.

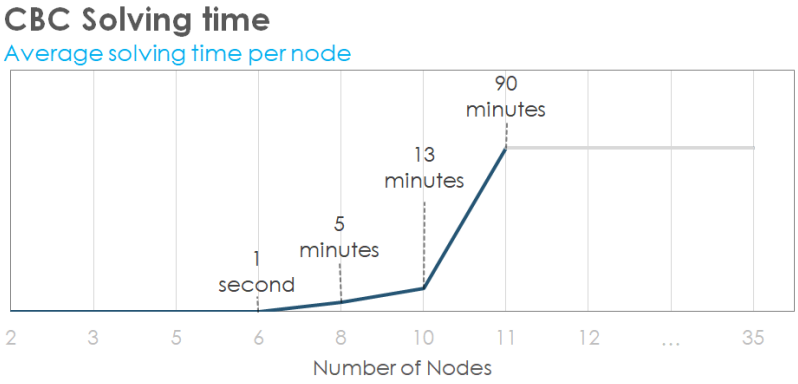


Figure 4.5: Classical solver. Average solving time according to the number of nodes in each instance S_i. The average solving time after 11 nodes exceeded the 90-minute time limit.

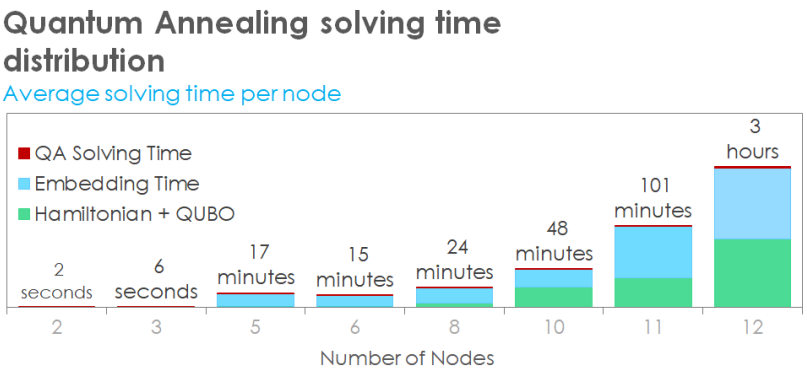


Figure 4.6: Quantum annealing solver. Average solving time according to the number of nodes in each instance S_i. The maximum number of nodes capable of finding a solution through QA was 12 elements. Hamiltonian + QUBO refers to the time required to construct the Hamiltonian and its conversion into a QUBO problem. The QA solving time represents the time it held for the QA solver to find a solution after completing all steps.

Figure 4.6 depicts the overall average solving time according to the number of nodes. The total solving time is split into three groups: the construction of the Hamiltonian and the QUBO in the same group, followed by the embedding time, and finishing with the solely QA solving time. Note that the embedding time represented a significant part of the time needed for solving the problem, followed by the construction of the Hamiltonian and the QUBO.

Generally, QA required more time to solve the complete sample, up to 12 nodes, Figure 4.7. The VRP must be adapted to an eligible input, substantially decreasing the

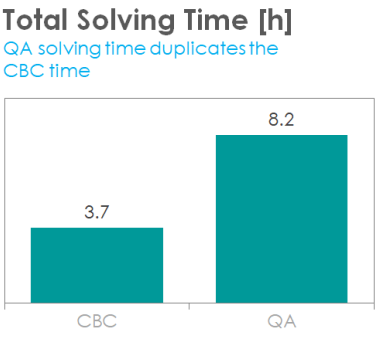


Figure 4.7: CBC solving time considers the model construction and finding the minimum solution. QA total solving time encloses the Hamiltonian constructions, QUBO transformation, embedding, and finding the minimum solution. CBC represents 45% of the QA required time.

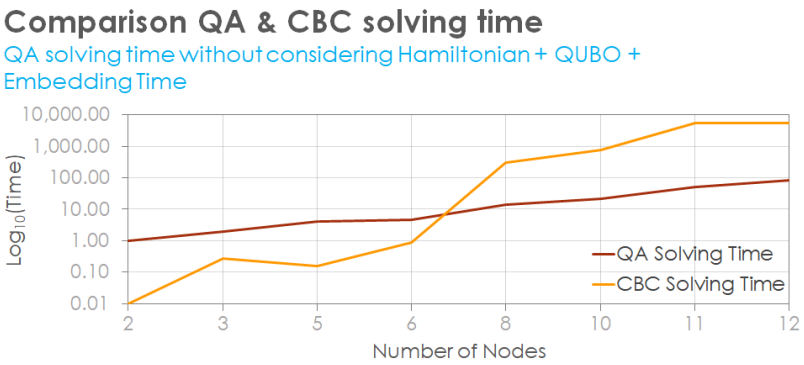


Figure 4.8: Average solving time over the reduced sample, i.e., instances with less than 12 elements. Comparison using Log_10 time after all pre-processing requirements for an eligible QA intake have been performed.

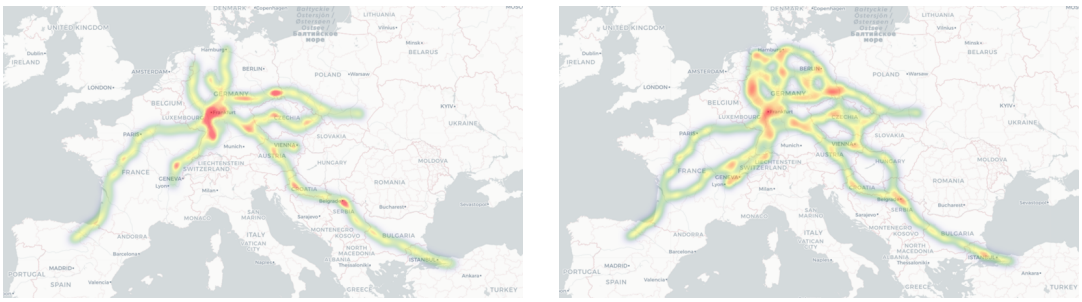
total solving time using QA. Furthermore, due to the nature of the problem, a formal follow-up of the full time needed for obtaining the parameter ρ in the Hamiltonian is not presented.

Experimental Results

Solving approach	**Sample [%]**	**Sample distance driven [km]**	**Pre-processing time [h]**	**Total calculation time [h]**	**Optimization[5] [%]**
CBC	84[6]	455,919	∽ 0.03	27	11.7
CBC	16[7]	64,226	∽ 0.002	3.7	8.3
Advantage 6.1	16[8]	64,226	8.17	8.23	8.1

Table 4.7: Comparison results. Instances S_i solved with CBC were 34, while 13 were solved with QA.

[5]Percentage reduction in kilometers driven, employing the trivial solution as a baseline. Derived from almost four months of simulated data from a logistic operational process of Siemens.
[6]Sample solved using CBC solver.
[7]Sample solved with CBC solver, smaller sub-sample selected for appropriate comparison with QA.
[8]Sample solved with QA.



(a) Routes designed according to the trivial solution.

(b) CBC routing optimization solution.

Figure 4.9: Transports Heatmaps. Routing optimization simulated solution, one depot located in Frankfurt, Germany. Reduction of 18% in the number of km driven using the trivial solution as a baseline. Notice the creation of new paths to visit more suppliers on the course to Frankfurt, resulting in fewer driven vehicles.

Even though the embedding was not achieved for instances with more than 12 nodes, and two more cases, with seven and nine nodes. QA proved to be more efficient than a classical solver once all the necessary pre-processing steps, i.e., the Hamiltonian construction, QUBO transformation, embedding, and parameter testing, were accomplished. Figure 4.8 illustrates a significant difference in computation time after all the necessary pre-processing steps for both solvers are completed. QA and CBC were tested remotely.

Table 4.7 presents the optimization results for both methods regarding data quality and solving time. The column Optimization [%] indicates the reduction in km against the trivial solution, considering different data sample lengths.

The pre-processing time utilizing the Advantage 6.1 architecture from D-Wave represented 99.3% of the total QA solving time. On the other hand, the CBC algorithm demonstrated slightly better solution quality, a reduction in the total distance driven of 0.2%, over the instances that QA managed to solve. Figure 4.9 illustrates an example of the routes optimized with the CBC solver. Chapter 5 will present the final outstanding points from this research.

Chapter 5

Conclusion

5.1 Conclusion

The applicability of QC and CC was assessed throughout this thesis. Based on the VRP literature, a new model benchmark according to Siemens requirements was defined. In the tested sample, both solvers (QA and CBC) presented a better solution quality than the trivial solution, see Table 4.7.

The results showed better performance regarding the solving time with QA than CBC, Figure 4.8. Unlike the steps required to prepare both problems, CBC proved more efficient. Whether QA can be used in the industry to solve this particular problem faster still needs further investigation, especially concerning the embedding procedure. Therefore, the CBC solver was better applied in real-world applications than QA, as it presented an overall shorter total solving time, Figure 4.7. Furthermore, the applicability of QA to optimize a company's daily processes needs a more significant improvement margin; if this gap is not high enough, the industry will not rent QC time through the cloud (from one of the commercial QC servers). Especially for companies whose main branch is other than research or technology. The incentive for using QC must be higher than its financial costs. New developments are expected, and this technology's acceptance has risen recently; it will likely become more affordable for further tests and applications soon.

5.2 Limitations and Future Works

Regarding the VRP problem setting, six main points are worth pointing out:

1) In Section 3 we studied a 5-index formulation in which the variable $x^k_{(i,t,j,t')}$ represented whether vehicle k travels from supplier i at time t to supplier j arriving at time t', such that $t' = t + \lceil \frac{s_i}{\Delta} \rceil + \lceil \frac{\tau_{ij}}{\Delta} \rceil$. One reason this formulation was chosen was the straightforward introduction of the loading time s_i into the traveling time. Notice that t already contains all the information required for knowing the arrival

time, and including the variable t' in the time-expanded matrix solely increases the problem's complexity. Therefore, a one-index reduction, i.e., a 4-index formulation $x^k_{(i,t,j)}$ transferring the loading time s_i into the problem constraints is recommended.

2) The time-expanded index formulation was selected due to the employment of binary decision variables. Nevertheless, including only binary decision variables might lower the possibility of testing new algorithms that can decrease the solving time.

3) In the formulated setting, only one depot was considered. As explained in Section 2.1, a multi-depot approach would help to reduce further the number of transports needed in Siemens' operational processes even more.

4) The guideline in this work corresponds to a static perspective of the routing problem, i.e., all the information needed for modeling the problem never changes during the planning horizon. Therefore, a dynamic view would be advantageous for a better practical application responding in real-time to customer requests.

5) Throughout the reviewed literature, the tested VRP models employed geodesic distances. Introducing real emulated routes is closer to the original problem. However, there are still opportunities for enhancement. The presented work did not consider changes in travel time, such as traffic or closed routes; this could be achieved by creating a time-dependent distance matrix $\tau_{ij}(t)$ with currently implemented packages that forecast vehicle traffic.

6) It would be interesting to test the methodology with real-world data.

It is suggested that a dynamic multi-point approach is tailored according to the needs of Siemens. Furthermore, a 4-index time-expanded network course is more likely to perform satisfactorily with QA solvers. On the other hand, the embedding time in QA represented around 50% of the total time needed for solving the sample. Moreover, due to this limitation, instances S_i with a higher number of nodes ($n > 12$) could not be tested. Hence, optimizing the embedding time is key for speeding up QA's computation time. If this challenge is overcome, QA will be closer to performing more promising than CC.

Notations

Indices	
i, j	Nodes
t, t'	Time points
k	Vehicles
Sets	
N	Nodes in the space network, except the depot
N_0	Nodes in the space network, including the depot
$\mathcal{N}_\mathcal{T}$	Nodes in the time-expanded network, except the depot
$\mathcal{N}_{0_\mathcal{T}}$	Nodes in the time-expanded network, including the depot
A	Arcs in the space network
$\mathcal{A}_\mathcal{T}$	Arcs in the time-space network
$\mathcal{T}$	Discretized time-expanded set
Functions	
z	Adjusted CVRPTW minimization function
z_1	Distance function
z_2	Number of vehicles function
$\mathcal{H}$	General Hamiltonian function
$\mathcal{H}_i$	Hamiltonian representation of constraint i

Table 5.1: Notation used in Chapter 3

Parameters	
d_{ij}	Distance from i to j, $d_{ij} = 0$ if $i = 0$ or $i = j$
τ_{ij}	Travel time from i to j, $t_{ij} = 0$ if $i = 0$ or $i = j$
q_i	Cargo to be picked at supplier i
v_i	Volume to be picked at supplier i
cl_i	Number of collis corresponding to v_i
c	Maximum vehicle weight capacity
m	Maximum vehicle volume
s_i	Service time at supplier i
H	Simulation horizon
t_w	Time-window length
e_i	Earliest arrival window at supplier i
l_i	Latest arrival window at supplier i
e_0	Earliest arrival window at the depot
l_0	Latest arrival window at the depot
Δ	Discretization constant
Q_{ij}	QUBO matrix coefficient
ρ	Penalty Term.
n_q	Number of weight slack variables
n_m	Number of volume slack variables
Decision Variables	
$x^k_{itjt'}$	Binary variable. Value 1 if vehicle k goes from i at time t and arrives to j at time t', 0 otherwise
y_k	Binary variable. Value 1 if vehicle k is used
s_{ik}	Positive real variable. Starting service time at supplier i with vehicle k
λ_{lk}	Slack binary variables for weight constraints
μ_{lk}	Slack binary variables for volume constraints

Table 5.1: Notation used in Chapter 3

Acronyms

AQC Adiabatic Quantum Computing.

AWS Amazon Web Service.

CBC Coin-or branch and cut.

CC Classical Computing.

CVRP Capacitated Vehicle Routing Problem.

CVRPTW Capacitated Vehicle Routing Problem with Time Windows.

DA Digital Annealing.

EDA Exploratory Data Analysis.

HGV Heavy Goods Vehicle.

MDVRP Multi-Depot Vehicle Routing Problem.

MDVRPPD Multi-Depot Vehicle Routing Problem with Pick-up and Delivery.

MDVRPPDTW Multi-Depot Vehicle Routing Problem with Pick-up, Delivery and Time Windows.

mTSP multi-Traveling Salesman Problem.

QA Quantum Annealing.

QC Quantum Computing.

QUBO Quadratic Unconstrained Binary Optimization.

RSA Rivest-Shamir-Adleman.

SA Simulated Annealing.

SND Service Network Design.

TSP Traveling Salesman Problem.

VRP Vehicle Routing Problem.

VRPB Vehicle Routing Problem with Backhauls.

VRPPD Vehicle Routing Problem with Pick-up and Delivery Problem.

List of Tables

List of Figures

Bibliography

[1] E. E. Agency. "Carbon dioxide emissions from Europe's heavy-duty vehicles". In: (Apr. 2018).

[2] Z. Mann. "The Top Eight Misconceptions about NP-Hardness". In: *Computer* 50 (May 2017), pp. 72–79.

[3] S. Nanda Kumar and R. Panneerselvam. "A Survey on the Vehicle Routing Problem and Its Variants". In: *Intelligent Information Management* 04 (Jan. 2012).

[4] P. F. Catherine McGeoch and K. Boothby. *White paper: Programming according to the fences and gates model for developing assured, secure software systems*. Tech. rep. 14-1063A-A. D-Wave Technical Report Series, June 2022. URL: https://www.dwavesys.com/media/eixhdtpa/14-1063a-a_the_d-wave_advantage2_prototype-4.pdf.

[5] Brilliant.org. *Getting Started with D-Wave Solvers*. 2022. URL: https://brilliant.org/wiki/quantum-tunneling/ (visited on 09/26/2022).

[6] S. Feld, C. Roch, T. Gabor, et al. "A Hybrid Solution Method for the Capacitated Vehicle Routing Problem Using a Quantum Annealer". In: *Frontiers in ICT* 6 (2019). ISSN: 2297-198X. URL: https://www.frontiersin.org/articles/10.3389/fict.2019.00013.

[7] D.-W. S. Inc. *QPU-Specific Physical Properties: Advantage_system6.1*. Tech. rep. 09-1272A-A. D-Wave, May 2022. URL: https://docs.dwavesys.com/docs/latest/doc_physical_properties.html.

[8] J. R. Kelly Boothby Andrew D. King. *Zephyr Topology of D-Wave Quantum Processors*. Tech. rep. 14-1056A-A. D-Wave, Sept. 2021. URL: https://www.dwavesys.com/media/2uznec4s/14-1056a-a_zephyr_topology_of_d-wave_quantum_processors.pdf?_gl=1*1wmrvg3*_ga*NDUxMDk3NzcuMTY2NjA5MDk2Nw..*_ga_DXNKH9HE3W*MTY2NjUzNTE4Ni45LjEuMTY2NjUzNTU2NC41My4wLjA..

[9] S. Okada, M. Ohzeki, M. Terabe, et al. "Improving solutions by embedding larger subproblems in a D-Wave quantum annealer". In: *Scientific reports* 9.1 (2019), pp. 1–10.

[10] B. Wang, X. Yang, and D. Zhang. "Research on Quantum Annealing Integer Factorization Based on Different Columns". In: *Frontiers in Physics* 10 (2022). ISSN: 2296-424X. URL: https://www.frontiersin.org/articles/10.3389/fphy.2022.914578.

[11] G. Dantzig, R. Fulkerson, and S. Johnson. "Solution of a Large-Scale Traveling-Salesman Problem". In: *Journal of the Operations Research Society of America* 2.4 (1954), pp. 393–410. ISSN: 00963984. (Visited on 10/05/2022).

[12] C. Wu, Y. Liang, H. Lee, et al. "Solving constrained traveling salesman problems by genetic algorithms". In: *Progress in Natural Science* 14 (July 2004).

[13] G. Clarke and J. W. Wright. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". In: *Operations Research* 12.4 (1964), pp. 568–581. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/167703 (visited on 10/05/2022).

[14] B. E. Gillett and L. R. Miller. "A Heuristic Algorithm for the Vehicle-Dispatch Problem". In: *Operations Research* 22.2 (1974), pp. 340–349. URL: https://EconPapers.repec.org/RePEc:inm:oropre:v:22:y:1974:i:2:p:340-349.

[15] A. Husban. "An Exact Solution Method for the MTSP". In: *The Journal of the Operational Research Society* 40.5 (1989), pp. 461–469. ISSN: 01605682, 14769360. URL: http://www.jstor.org/stable/2583618 (visited on 10/05/2022).

[16] B. Eksioglu, A. Vural, and A. Reisman. "The vehicle routing problem: A taxonomic review". In: *Computers Industrial Engineering* 57 (Nov. 2009), pp. 1472–1483.

[17] B. L. Golden, T. L. Magnanti, and H. Q. Nguyen. "Implementing vehicle routing algorithms". In: *Networks* 7.2 (1977), pp. 113–148. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230070203. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230070203.

[18] G. Laporte and Y. Nobert. "A Branch and Bound Algorithm for the Capacitated Vehicle Routing Problem". In: *OR Spectr.* 5.2 (June 1983), pp. 77–85. ISSN: 0171-6468. URL: https://doi.org/10.1007/BF01720015.

[19] G. Laporte and F. Semet. "5. Classical Heuristics for the Capacitated VRP". In: Jan. 2002, pp. 109–128. ISBN: 978-0-89871-498-2.

[20] J.-F. Cordeau, G. Laporte, M. Savelsbergh, et al. "Vehicle Routing". In: vol. 14. Jan. 2007, pp. 195–224.

[21] F. Gheysens, B. Golden, and A. Assad. "A comparison of techniques for solving the fleet size and mix vehicle routing problem". In: *Operations-Research-Spektrum* 6.4 (1984), pp. 207–216.

[22] R. Baldacci, P. Toth, and D. Vigo. "Recent advances in vehicle routing exact algorithms". In: *4OR* 5 (Nov. 2007), pp. 269–298.

[23] B. L. Golden and W. Stewart Jr. "Held at Nat'l. Bur. of Stds., Gaithersburg, MD, April 14-15, 1977.(Issued February 1978)". In: *NBS Special Publication* 503 (1978), p. 252.

[24] M. Noorizadegan and B. Chen. "Vehicle Routing with Probabilistic Capacity Constraints". In: *European Journal of Operational Research* 270 (Apr. 2018), pp. 544–555.

[25] M. M. Solomon. "Vehicle Rouiting and Scheduling with Time Window Constraints: Models and Algorithms (Heuristics)". English. PhD thesis. 1984, p. 201. ISBN: 9798662134843. URL: https://www.proquest.com/dissertations-theses/vehicle-routing-scheduling-with-time-window/docview/303304246/se-2.

[26] M. Desrochers, J. Desrosiers, and M. Solomon. "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows". In: *Oper. Res.* 40.2 (Apr. 1992), pp. 342–354. ISSN: 0030-364X.

[27] M. M. Solomon and J. Desrosiers. "Time Window Constrained Routing and Scheduling Problems". In: *Transportation Science* 22.1 (1988), pp. 1–13. ISSN: 00411655, 15265447. URL: http://www.jstor.org/stable/25768291 (visited on 10/06/2022).

[28] G. Desaulniers, J. Desrosiers, A. Erdmann, et al. "VRP with Pickup and Delivery". In: Jan. 2002, pp. 225–242. ISBN: 978-0-89871-498-2.

[29] G. Berbeglia, J.-F. Cordeau, and G. Laporte. "Dynamic pickup and delivery problems". In: *European Journal of Operational Research* 202.1 (2010), pp. 8–15. ISSN: 0377-2217. URL: https://www.sciencedirect.com/science/article/pii/S0377221709002999.

[30] G. Berbeglia, J.-F. Cordeau, and G. Laporte. "Dynamic pickup and delivery problems". In: *European Journal of Operational Research* 202.1 (2010), pp. 8–15. ISSN: 0377-2217. URL: https://www.sciencedirect.com/science/article/pii/S0377221709002999.

[31] H. Min, J. Current, and D. Schilling. "The Multiple Depot Vehicle Routing Problem with Backhauling". English. In: *Journal of Business Logistics* 13.1 (1992), p. 259. URL: https://www.proquest.com/scholarly-journals/multiple-depot-vehicle-routing-problem-with/docview/212603295/se-2.

[32] D. Bertsimas and J. Tsitsiklis. "Simulated Annealing". In: *Statistical Science* 8.1 (1993), pp. 10–15. URL: https://doi.org/10.1214/ss/1177011077.

[33] A. S. Alfa, S. S. Heragu, and M. Chen. "A 3-OPT based simulated annealing algorithm for vehicle routing problems". In: *Computers Industrial Engineering* 21.1 (1991), pp. 635–639. ISSN: 0360-8352. URL: https://www.sciencedirect.com/science/article/pii/0360835291901653.

[34] I. Osman. "Meta-strategy simulated annealing and Tabu search algorithms for the vehicle routine problem". In: *Annals of Operations Research* 41 (Dec. 1993), pp. 421–451.

[35] A. Lim and W. Zhu. "A Fast and Effective Insertion Algorithm for Multi-depot Vehicle Routing Problem with Fixed Distribution of Vehicles and a New Simulated Annealing Approach". In: *Advances in Applied Artificial Intelligence*. Ed. by M. Ali and R. Dapoigny. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–291.

[36] M. Filipec, D. Skrlec, and S. Krajcar. "An efficient implementation of genetic algorithms for constrained vehicle routing problem". In: *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*. Vol. 3. IEEE. 1998, pp. 2231–2236.

[37] B. Bullnheimer, R. F. Hartl, and C. Strauss. "Applying the ant system to the vehicle routing problem". In: *Meta-heuristics*. Springer, 1999, pp. 285–296.

[38] H. Ghaziri. "Supervision in the self-organizing feature map: Application to the vehicle routing problem". In: *Meta-heuristics*. Springer, 1996, pp. 651–660.

[39] B. Fleischmann. "The vehicle routing problem with multiple use of vehicles". In: *Fachbereich Wirtschaftswissenschaften, Universität Hamburg* (1990).

[40] I. D. Giosa, I. L. Tansini, and I. O. Viera. "New assignment algorithms for the multi-depot vehicle routing problem". In: *Journal of the Operational Research Society* 53.9 (2002), pp. 977–984.

[41] S. Salhi and M. Sari. "A multi-level composite heuristic for the multi-depot vehicle fleet mix problem". In: *European Journal of Operational Research* 103.1 (1997), pp. 95–112. ISSN: 0377-2217. URL: https://www.sciencedirect.com/science/article/pii/S0377221796002536.

[42] D. Pisinger and S. Ropke. "A general heuristic for vehicle routing problems". In: *Computers and Operations Research* 34.8 (2007), pp. 2403–2435. ISSN: 0305-0548. URL: https://www.sciencedirect.com/science/article/pii/S0305054805003023.

[43] G. Laporte, Y. Nobert, and S. Taillefer. "Solving a family of multi-depot vehicle routing and location-routing problems". In: *Transportation science* 22.3 (1988), pp. 161–172.

[44] C. Barnhart and G. Laporte. *Handbooks in operations research and management science: Transportation*. Elsevier, 2006.

[45] B. Crevier, J.-F. Cordeau, and G. Laporte. "The multi-depot vehicle routing problem with inter-depot routes". In: *European journal of operational research* 176.2 (2007), pp. 756–773.

[46] C. Contardo and R. Martinelli. "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints". In: *Discrete Optimization* 12 (2014), pp. 129–146. ISSN: 1572-5286. URL: https://www.sciencedirect.com/science/article/pii/S1572528614000097.

[47] R. Dondo and J. Cerdá. "A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows". In: *European Journal of Operational Research* 176.3 (2007), pp. 1478–1507. ISSN: 0377-2217. URL: https://www.sciencedirect.com/science/article/pii/S0377221705008672.

[48] J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, et al. "A literature review on the vehicle routing problem with multiple depots". In: *Computers & Industrial Engineering* 79 (2015), pp. 115–129.

[49] L. Bianchi. *Notes on Dynamic Vehicle Routing - The State of the Art. Technical Report - IDSIA*. 2000.

[50] G. Yu and Y. Yang. "Dynamic routing with real-time traffic information". In: *Operational Research* 19 (Dec. 2019), pp. 1–26.

[51] Y. Wang, Y. Zhang, and J. Ma. "Dynamic Real-Time High-Capacity Ride Sharing Model with Subsequent Information". In: *IET Intelligent Transport Systems* 14 (July 2020).

[52] F. Lucas, R. Billot, M. Sevaux, et al. "Some insights about the use of machine learning for solving VRP". In: *31st European Conference on Operational Research (EURO 31)*. Athens, Greece, July 2021. URL: https://hal.archives-ouvertes.fr/hal-03285115.

[53] J. Li, L. Xin, Z. Cao, et al. "Heterogeneous Attentions for Solving Pickup and Delivery Problem via Deep Reinforcement Learning". In: *IEEE Transactions on Intelligent Transportation Systems* 23.3 (2022), pp. 2306–2315.

[54] J. D. Hidary and J. D. Hidary. *Quantum computing: an applied approach*. Vol. 1. Springer, 2021.

[55] P. Benioff. "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines". In: *Journal of Statistical Physics* 22 (May 1980), pp. 563–591.

[56] P. W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Review* 41.2 (1999), pp. 303–332. eprint: https://doi.org/10.1137/S0036144598347011. URL: https://doi.org/10.1137/S0036144598347011.

[57] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. URL: https://doi.org/10.1145/359340.359342.

[58] R. Orús, S. Mugel, and E. Lizaso. "Quantum computing for finance: Overview and prospects". In: *Reviews in Physics* 4 (2019), p. 100028. ISSN: 2405-4283. URL: https://www.sciencedirect.com/science/article/pii/S2405428318300571.

[59] J. M. Garcia. "How Quantum Computing Could Remake Chemistry". In: *Scientific American* (Mar. 2021). URL: https://www.scientificamerican.com/article/how-quantum-computing-could-remake-chemistry/.

[60] S. Lloyd. "Capacity of the noisy quantum channel". English. In: *Physical Review A - Atomic, Molecular, and Optical Physics* 55.3 (1997). Cited By :513, pp. 1613–1622. URL: www.scopus.com.

[61] S. Muralidharan, J. Kim, N. Lütkenhaus, et al. "Ultrafast and Fault-Tolerant Quantum Communication across Long Distances". In: *Phys. Rev. Lett.* 112 (25 June 2014), p. 250501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.112.250501.

[62] L. Gyongyosi, S. Imre, and H. V. Nguyen. "A Survey on Quantum Channel Capacities". In: *IEEE Communications Surveys and Tutorials* 20.2 (2018), pp. 1149–1205.

[63] J. Biamonte, P. Wittek, N. Pancotti, et al. "Quantum machine learning". In: *Nature* 549.7671 (2017), pp. 195–202.

[64] S. B. Ramezani, A. Sommers, H. K. Manchukonda, et al. "Machine Learning Algorithms in Quantum Computing: A Survey". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8.

[65] A. W. Harrow and A. Montanaro. "Quantum computational supremacy". In: *Nature* 549.7671 (2017), pp. 203–209.

[66] I. Ryan Mandelbaum. *Five years ago today, we put the first quantum computer on the cloud. Here's how we did it.* 2021. URL: https://research.ibm.com/blog/quantum-five-years (visited on 10/04/2022).

[67] C. C. McGeoch. "Adiabatic quantum computation and quantum annealing: Theory and practice". In: *Synthesis Lectures on Quantum Computing* 5.2 (2014), pp. 1–93.

[68] T. Kadowaki and H. Nishimori. "Quantum annealing in the transverse Ising model". In: *Phys. Rev. E* 58 (5 Nov. 1998), pp. 5355–5363. URL: https://link.aps.org/doi/10.1103/PhysRevE.58.5355.

[69] M. Aramon, G. Rosenberg, E. Valiante, et al. "Physics-Inspired Optimization for Quadratic Unconstrained Problems Using a Digital Annealer". In: *Frontiers in Physics* 7 (2019). ISSN: 2296-424X. URL: https://www.frontiersin.org/articles/10.3389/fphy.2019.00048.

[70] M. Ayodele, R. Allmendinger, M. López-Ibáñez, et al. "Multi-Objective QUBO Solver: Bi-Objective Quadratic Assignment Problem". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. Boston, Massachusetts: Association for Computing Machinery, 2022, pp. 467–475. ISBN: 9781450392372. URL: https://doi.org/10.1145/3512290.3528698.

[71] D-Wave. *Getting Started with D-Wave Solvers*. 2022. URL: https://docs.dwavesys.com/docs/latest/c_gs_2.html (visited on 10/02/2022).

[72] D. Zubov, F. Volponi, and M. Khosravy. "D-wave quantum computing Ising model: A case study for the forecasting of heat waves". In: *2015 International Conference on Control, Automation and Information Sciences (ICCAIS)*. 2015, pp. 149–152.

[73] S. Yarkoni, E. Raponi, T. Bäck, et al. "Quantum annealing for industry applications: Introduction and review". In: *Reports on Progress in Physics* (2022).

[74] H. Irie, G. Wongpaisarnsin, M. Terabe, et al. "Quantum annealing of vehicle routing problem with time, state and capacity". In: *International Workshop on Quantum Technology and Optimization Problems*. Springer. 2019, pp. 145–156.

[75] R. H. Warren. "Solving the traveling salesman problem on a quantum annealer". In: *SN Applied Sciences* 2.1 (2020), pp. 1–5.

[76] S. Jain. "Solving the Traveling Salesman Problem on the D-Wave Quantum Computer". In: *Frontiers in Physics* 9 (2021). ISSN: 2296-424X. URL: https://www.frontiersin.org/articles/10.3389/fphy.2021.760783.

[77] A. Crispin and A. Syrichas. "Quantum Annealing Algorithm for Vehicle Scheduling". In: *2013 IEEE International Conference on Systems, Man, and Cybernetics*. 2013, pp. 3523–3528.

[78] S. Bao, M. Tawada, S. Tanaka, et al. "An approach to the vehicle routing problem with balanced pick-up using Ising machines". In: *2021 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. IEEE. 2021, pp. 1–4.

[79] R. Harikrishnakumar, S. Nannapaneni, N. H. Nguyen, et al. "A Quantum Annealing Approach for Dynamic Multi-Depot Capacitated Vehicle Routing Problem". In: *ArXiv* abs/2005.12478 (2020).

[80] M. Borowski, P. Gora, K. Karnas, et al. "New hybrid quantum annealing algorithms for solving vehicle routing problem". In: *International Conference on Computational Science*. Springer. 2020, pp. 546–561.

[81] R. K. Nath, H. Thapliyal, and T. S. Humble. "A review of machine learning classification using quantum annealing for real-world applications". In: *SN Computer Science* 2.5 (2021), pp. 1–11.

[82] J. Brence, D. Mihailović, V. Kabanov, et al. "Boosting the Performance of Quantum Annealers using Machine Learning". In: *arXiv preprint arXiv:2203.02360* (2022).

[83] Y.-Q. Chen, Y. Chen, C.-K. Lee, et al. "Optimizing quantum annealing schedules with Monte Carlo tree search enhanced with neural networks". In: *Nature Machine Intelligence* 4.3 (2022), pp. 269–278.

[84] R. Ayanzadeh, M. Halem, and T. Finin. "Reinforcement quantum annealing: A hybrid quantum learning automata". In: *Scientific reports* 10.1 (2020), pp. 1–11.

[85] A. P. Lorenzo Pautasso and H. S. .-.-. M. D. blue. *Five years ago today, we put the first quantum computer on the cloud. Here's how we did it.* Feb. 2021. URL: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/quantum-computing-use-cases-are-getting-real-what-you-need-to-know (visited on 10/11/2022).

[86] N. Mohseni, P. McMahon, and T. Byrnes. "Ising machines as hardware solvers of combinatorial optimization problems". In: *Nature Reviews Physics* (Apr. 2022).

[87] M. Zaman, K. Tanahashi, and S. Tanaka. "PyQUBO: Python Library for Mapping Combinatorial Optimization Problems to QUBO Form". In: *IEEE Transactions on Computers* 71.4 (2022), pp. 838–850.

[88] E. K. Grant and T. S. Humble. *Adiabatic Quantum Computing and Quantum Annealing*. July 2020. URL: https://oxfordre.com/physics/view/10.1093/acrefore/9780190871994.001.0001/acrefore-9780190871994-e-32.

[89] S. Irnich, P. Toth, and D. Vigo. "Chapter 1: The Family of Vehicle Routing Problems". In: Nov. 2014, pp. 1–33. ISBN: 978-1-61197-358-7.

[90] B. Kallehauge, J. Larsen, O. B. Madsen, et al. "Vehicle Routing Problem with Time Windows". In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. M. Solomon. Boston, MA: Springer US, 2005, pp. 67–98. ISBN: 978-0-387-25486-9. URL: https://doi.org/10.1007/0-387-25486-2_3.

[91] Y. Yuan, D. Cattaruzza, M. Ogier, et al. "A note on the lifted Miller Tucker Zemlin subtour elimination constraints for routing problems with time windows". In: *Operations Research Letters* 48.2 (2020), pp. 167–169. ISSN: 0167-6377. URL: https://www.sciencedirect.com/science/article/pii/S016763772030016X.

[92] N. Boland, M. Hewitt, L. Marshall, et al. "The Continuous-Time Service Network Design Problem". In: *Operations Research* 65 (July 2017).

[93] N. Boland, M. Hewitt, D. M. Vu, et al. "Solving the Traveling Salesman Problem with Time Windows Through Dynamically Generated Time-Expanded Networks". In: *Integration of AI and OR Techniques in Constraint Programming*. Ed. by D. Salvagnin and M. Lombardi. Cham: Springer International Publishing, 2017, pp. 254–262.

[94] V. Duc Minh, M. Hewitt, N. Boland, et al. "Solving Time Dependent Traveling Salesman Problem with Time Windows". In: June 2018.

[95] S. Albiński, T. G. Crainic, and S. Minner. "The Day-before Truck Platooning Planning Problem and the Value of Autonomous Driving". In: 2020.

[96] N. Boland, M. Hewitt, L. Marshall, et al. "The price of discretizing time: a study in service network design". In: *EURO Journal on Transportation and Logistics* 8.2 (2019). Special Issue: Advances in vehicle routing and logistics optimization: exact methods, pp. 195–216. ISSN: 2192-4376. URL: https://www.sciencedirect.com/science/article/pii/S2192437620300327.

[97] V. Korablev, I. Makeev, E. Kharitonov, et al. "Approaches to Solve the Vehicle Routing Problem in the Valuables Delivery Domain". In: *Procedia Computer Science* 88 (2016). 7th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2016, held July 16 to July 19, 2016 in New York City, NY, USA, pp. 487–492. ISSN: 1877-0509. URL: https://www.sciencedirect.com/science/article/pii/S1877050916317252.

[98] A. Lucas. "Ising formulations of many NP problems". In: *Frontiers in Physics* 2 (2014). ISSN: 2296-424X. URL: https://www.frontiersin.org/articles/10.3389/fphy.2014.00005.

[99] D-Wave. *Getting Started with D-Wave Solvers*. 2022. URL: https://docs.dwavesys.com/docs/latest/c_gs_2.html (visited on 09/26/2022).

[100] J. Hidary. *Quantum Computing: An Applied Approach*. Jan. 2019, p. 146. ISBN: 978-3-030-23921-3.

[101] A. Verma and M. Lewis. "Variable Reduction For Quadratic Unconstrained Binary Optimization". In: (May 2021).

[102] Google. *Python Client for Google Maps Services*. 2022. URL: https://google-maps-services-python.readthedocs.io/_/downloads/en/latest/pdf/ (visited on 08/27/2022).

[103] Google. *Open Route Service - Quickstart*. 2022. URL: https://openrouteservice-py.readthedocs.io/en/latest/ (visited on 10/02/2022).

[104] Google. *Open Route Service - Quickstart*. 2022. URL: https://developers.google.com/optimization/introduction/overview (visited on 10/05/2022).

[105] C.-O. Foundation. *CBC*. 2022. URL: `https://github.com/coin-or/Cbc` (visited on 10/05/2022).

[106] V. Choi. "Minor-embedding in adiabatic quantum computation: I. The parameter setting problem". In: *Quantum Information Processing* 7.5 (2008), pp. 193–209.

[107] AWS-Braket. *AWS*. 2022. URL: `https://aws.amazon.com/braket/?nc1=h_ls` (visited on 10/14/2022).