FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Quantum Computing for Optimizing Routes in Smart Cities

André Filipe de Soveral Torres Lopes dos Santos

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Professor Rui Filipe Lima Maranhão de Abreu

Second Supervisor: Professor João Paulo de Sousa Ferreira Fernandes

July 30, 2021

Quantum Computing for Optimizing Routes in Smart Cities

André Filipe de Soveral Torres Lopes dos Santos

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Jorge Barbosa
External Examiner: Prof. Inês Dutra
Supervisor: Prof. Rui Filipe Lima Maranhão de Abreu
Second Supervisor: Prof. João Paulo de Sousa Ferreira Fernandes

July 30, 2021

Abstract

In recent years, Quantum Computing has been gaining increasing importance as more and more examples of Quantum Supremacy (demonstrating that a programmable quantum device can solve a problem that no classical computer can solve in any feasible amount of time) has been demonstrated by Google, Honeywell and other places where quantum computing research is underway.

The potential of Quantum Computing is explored to address computational issues within the smart cities domain. Focusing in its application on the routing of garbage trucks.

The garbage truck routing issue was studied and a simplified model of route planning was considered. A modified "Travelling Salesman Problem" (TSP) algorithm was applied, with added restrictions concerning the start and the end of the route.

A few graphs from a TSP database were used. With each graph, the problem was formulated as a QUBO, and using a solver (Qbsolv), it was solved both by a classical computer (Laptop) and a Quantum Computer (Dwave Advantage System). Due to the limitation on time with the Dwave Quantum Computer, there was a limit to how many experiments that were possible to run on it.

The samples show that as the complexity of the problem increases, so does its deviation from the "optimal result", but nonetheless the results obtained show that it is possible to obtain a "good enough" result that has practical real world applications.

Keywords: Quantum Computing, Routing Algorithms, Travelling Salesman Problem

Resumo

Nos últimos anos, a Computação Quântica tem ganho cada vez mais importância à medida que mais e mais exemplos de Supremacia Quântica (demonstrando que um dispositivo quântico programável pode resolver um problema que nenhum computador clássico pode resolver em qualquer período de tempo viável) têm sido apresentados pela Google, Honeywell e outros lugares onde existe investigação na área de computação quântica.

O potencial da Computação Quântica será explorado para abordar questões computacionais no domínio das cidades inteligentes. Focando em sua aplicação ao roteamento de camiões do lixo.

A questão do roteamento dos camiões de lixo foi estudada e um modelo simplificado de planeamento de rotas foi considerado. Um algoritmo modificado do "Problema do caixeiro viajante" (TSP - sigla Inglesa) foi aplicado, com restrições adicionais relativas ao início e ao final da rota.

Alguns grafos de uma base de dados TSP foram usados. A cada grafo, o problema foi formulado como um QUBO, e usando um solver (Qbsolv), foi resolvido tanto por um computador clássico (Portátil) como por um Computador Quântico ("Dwave Advantage System"). Devido à limitação de tempo com o computador quântico da Dwave, houve um limite para o número de experiências a realizar nesse computador..

As amostras mostram que à medida que a complexidade do problema aumenta, aumenta também o seu desvio do "resultado ótimo", mesmo assim os resultados obtidos mostram que é possível obter um resultado "suficientemente bom" com aplicações práticas no mundo real.

Palavras-chave: Computação Quântica, Algoritmos de Roteamento , Problema do caixeiro viajante

Acknowledgements

I would like to thank my parents for their unending patience and comprehension not only during the development of this work. Their love and support are crucial pillars to my success.

I would also like to thank Professor Rui Maranhão and Professor João Paulo Fernandes for all the help and support he has provided as my advisor on this work, not only in any questions or issues that I had during the development of my work, but also for providing me with all the means and helpful tips that aided me in completing this work successfully.

André Lopes dos Santos

"For a successful technology,
reality must take precedence over public relations,
for nature cannot be fooled."

Richard P. Feynman

Contents

List of Figures

List of Tables

Abbreviations

QC	Quantum Computing
NISQ	Noisy Intermediate-scale quantum
GPU	Graphical Processing Unit
QPU	Quantum Processing Unit
QDK	Quantum Development Kit
SDK	Software Development Kit
QEC	Quantum Error Correction
VRPTW	Vehicle Routing Problem with Time Windows
BKS	Best Known Solution
QUBO	Quadratic Unconstrained Binary Optimization
TSP	Travelling Salesman Problem

Chapter 1

Introduction

A brief introduction to the work presented in this paper is summarized in this chapter. In section 1.1 we explain the context in which this work is inserted. In 1.2 we explain the motivation behind the work while in 1.3 the main objectives of this work are outlined. Finally in section 1.4 the structure of the document is explained.

1.1 Context

The last two decades have seen a rise of computer components embedded into a variety of machines and devices, including televisions, automobiles and even refrigerators. All of these devices also have been getting even more interconnected with each other providing symbiotic advantages to its users.

This has given rise to the concept of *Smart City*. A city in which all devices are interconnected and share information with each other. That information is then used to optimize daily life by providing optimized routes for automobiles (using the data from other automobiles), turning off certain lights when they are not being used (i.e. when no one is around) among other things.

One thing that *Smart Cities* will surely need, considering the increasing number interconnected vehicles is optimized routing algorithms that take into account the many variables present in a given situation.

Although promising, the implementation of such things present many challenges, mainly in the computational field, since there are millions of variables and agents in a given environment, optimization becomes very computationally taxing.

1.2 Motivation

As new and better technologies emerge, many of these can be used to tackle the challenges presented by *Smart Cities*. One such technological development that has shown promise in facing

these challenges (and many others) is **Quantum Computing**.

Although a technology that it's still in its infancy (or "pre-transistor" phase) some proof-of-concept results by some companies (e.g. Google) and some research groups have shown that it may be possible to perform certain tasks on a Quantum Computer in a few minutes, that would take modern supercomputers thousands of years to perform. Although many of these claims are still disputed.

When it comes to *Smart Cities*, routing and optimization problems are computationally heavy when using classic routing algorithms, even on supercomputers. This may present an area in which quantum computers can innovate.

1.3 Objectives

The main goals of this work are as follows:

- Study the different Quantum computer technologies available.
- Study current uses of Quantum computing and quantum algorithms in various fields
- Apply knowledge to develop, validate and test a quantum algorithm to Garbage truck routing in Smart Cities
- Use that algorithm on quantum computers and compare results with ones that are run on classical computers

1.4 Document Structure

This paper is structured in several chapters. In Chapter 2, a short explanation of Quantum computing and its current state will be addressed. In Chapter 3, some of the issues regarding the routing of garbage trucks will be discussed. Next, in Chapter 4, the approach taken and the details of the implementations are explained. In Chapter 5 the experimental results are shown and discussed. Finally, in Chapter 6, the main conclusions and proposal for future work can be found.

Chapter 2

Quantum Computing

In this Chapter we present the topic of Quantum Computing, its state of art and how it relates to the challenges in *Smart Cities*.

2.1 Background

One of the earliest definitions of Quantum Computers was presented by Richard Feynman [17] in which he proposed the use of quantum computers in order to understand quantum phenomena.

2.1.1 What is Quantum Computing?

To understand quantum computing, several quantum mechanical properties need to be explained in order to better understand how these contribute to how a quantum computer works:

- **Quantum Superposition**: quantum mechanical principle which states that a given particle (an electron in an atom for example) can be seen as existing in a superposition of possible states until it is observed (Schrödinger's Cat: a cat is in a box with poison and until someone opens the box to check whether the cat is dead or alive, the cat is considered to be both alive and dead) [23].
- **Quantum Entanglement**: when two (or more) particles interact in a way that each of their individual quantum states cannot be described independently from one another.
- **Interference**: property in physics in which a given wave can interfere with another wave when they superpose. This phenomenon can also be found in photons (shown by the double slit experiment, that proved that photons are both waves and particles).

These are important quantum properties in which quantum computing is based on. As such, quantum computing can be defined as : "A computing paradigm that exploits quantum mechanical properties (superposition, entanglement,interference...) of matter in order to do calculations" [16]

Although the term "quantum computer" is frequently used, taking into account the recent developments in noisy intermediate-scale quantum (NISQ) technology, it is more correct to refer to "quantum accelerators" the same way we talk about Graphical processing Units (GPU) as a "graphics accelerator", since it is believed that the first societal revelant application of quantum computing will be a hybrid between a classical computer and a quantum accelerator [10]. In many ocasions both these terms are used interchangeably.

2.1.2 From bits to qubits

In classical computing, the most basic unit of information is the "binary digit" or "bit". This can take one of two values: 0 or 1.

In quantum computing however, the most basic unit of information is the "Quantum bit" or "Qubit". Unlike its classical counterpart, it is not discrete but continuous and it can "take" an infinite number of values [16].

Qubits are represented in the Hilbert vector Space and are represented using a Dirac Notation $|0\rangle$ and $|1\rangle$.

In equation 2.1 a mathematical representation of a generic qubit in superposition can be seen

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

In this case α and β represent complex numbers such that:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.2}$$

Just like Schrödinger's Cat is in "superposition" of states until the box is open, so is the qubit until it is measured. When a qubit is measured, it "collapses" to either 0 or 1. Mathematically speaking, when we measure a qubit, it has a probability $|\alpha|^2$ of being 0 and a probability $|\beta|^2$ of being 1.

To represent a qubit, a Bloch Sphere is frequently used as can be seen in Figure 2.1.

If we take into account equations 2.1 and 2.2, we can find angles γ, δ and θ such that:

$$\alpha = e^{i\gamma} cos\frac{\theta}{2} \tag{2.3}$$

$$\beta = e^{i\delta} sin\frac{\theta}{2} \tag{2.4}$$

Then equation 2.1 will become:

$$|\psi\rangle = e^{i\gamma} cos\frac{\theta}{2}|0\rangle + e^{i\delta} sin\frac{\theta}{2}|1\rangle \tag{2.5}$$

Considering that $\varphi = \delta - \gamma$:

$$|\psi\rangle = e^{i\gamma}(cos\frac{\theta}{2}|0\rangle + e^{i\varphi} sin\frac{\theta}{2}|1\rangle) \tag{2.6}$$

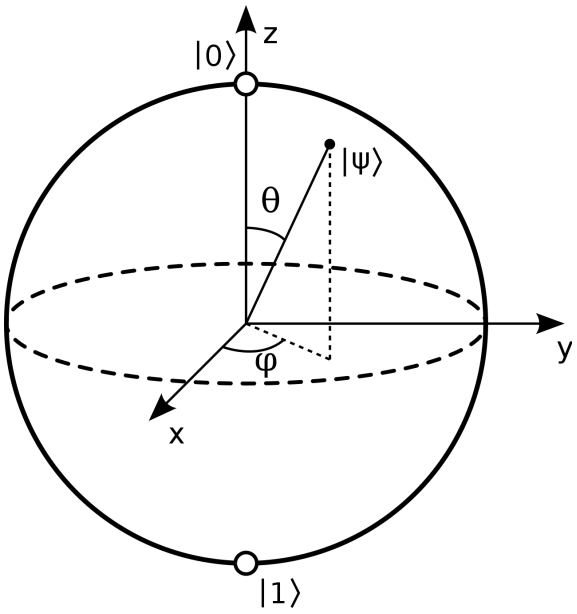


Figure 2.1: Bloch Sphere representation of a qubit. (Credit: By Smite-Meister - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=5829358)

Since the global phase (the "$e^{i\gamma}$ " factor) is physically irrelevant we can rewrite equation 2.6 as:

$$|\psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle \tag{2.7}$$

Another important aspect concerning qubits, is that currently there are three types of qubits [10]: *Real Qubits*, *Realistic Qubits* and *Perfect Qubits*.

Real Qubits are the type of qubits that currently exist in modern quantum computers. These types of qubits however still have problems such as decoherence (the environment interacting with the qubits and changing their quantum states) and large error rates ($10^{-}2$) [10], which need to be addressed until a quantum device becomes feasible for commercial use.

Realistic qubits are considered the "middle ground" between *real qubits* and *perfect qubits* since they take into consideration the physical architecture of a quantum device in which a qubit's interaction is limited to its nearest neighbor but it does assume that decoherence and error rates are rather smaller than in real qubits.

Perfect qubits: are the theoretical qubits that many quantum programmers use in order to develop quantum algorithms, without considering decoherence and error rates of real qubits.

Although there are ways to perform error correction on qubits using Quantum Error Correction (QEC) methods [10, 8], these can add an enormous overhead in the number of qubits required for correction (up to 9 qubits per qubit), so further developments in the development of better physical qubits is required for a viable commercial application of quantum devices.

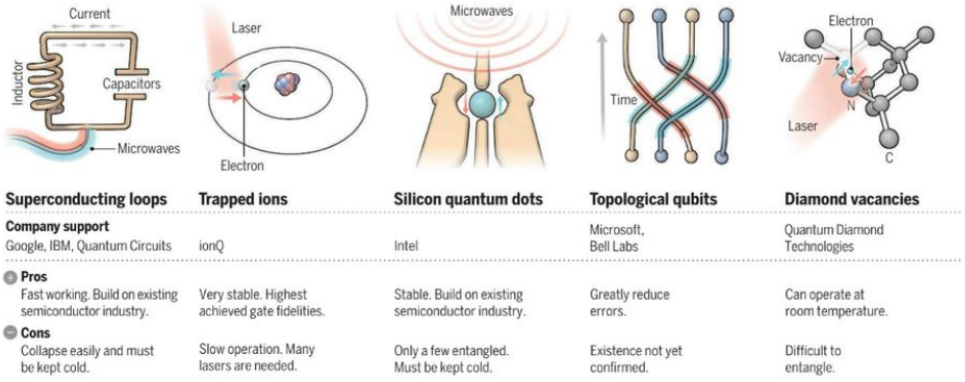


Figure 2.2: Technologies that make quantum computers possible (Credit: [16])

2.2 State of Art

In the last 20 years, the field of quantum computing has gained increasing attention and investment by research groups and several companies such as IBM and Google. Although physical quantum computers still have a lot of limitations on implementing real qubits it has not slowed down the development of new algorithms and methods using quantum computers.

2.2.1 Physical Implementations quantum computing

At the time of writing, there are many different technologies and methods to build a quantum computer with real qubits. Companies like IBM, Intel, Google and many other are investing in the implementation of such technologies although, as of yet, no technology has proven more adequate than another, each having its own advantages and disadvantages. Those can be found in Figure 2.2 [16].

It can be seen that many of the limitations are either the extreme conditions in which quantum computers must be or the difficulty to keep them stable and to use some of its properties (like entanglement). Hopefully in the next decade or so, the researchers that are working on developing these quantum computers can improve on these technologies and allow for a more widespread use of these.

2.2.2 Quantum Computer Architectures

There are two (main) candidates for the architecture of quantum computers: Quantum gate-based computers and quantum annealing-based computers[10]:

Quantum Gate-Based Computers are the more "traditional" Quantum computers, in which the solution interferes constructively while the wrong solutions interferes destructively (using the *interference* property of quantum mechanics), biasing the system towards the correct solution. Examples of these type of computers are the ones by IBM or Intel.

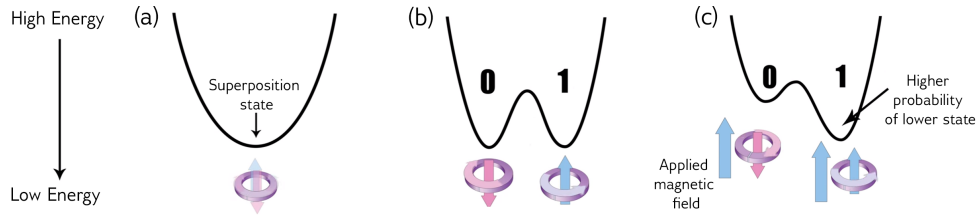


Figure 2.3: Energy diagram changes over time as the quantum annealing process runs and a bias is applied (Credit: [6])

Quantum Annealers, on the other hand, "initialize the system in a low-energy state and gradually introduce the parameters of a problem you wish to solve. The slow change makes it likely that the system ends in a low-energy state of the problem, which corresponds to an optimal solution." [6]

In this type of system, all the qubits that are used are initialized in a superposition state and then an external magnetic field (with programmable quantity) is applied to the qubits (i.e. a *bias* is applied), making it more probable that the desired value will be the the one that results in the lowest energy as can be seen in Figure 2.3 .

Although *bias* alone is not enough, in order to link qubits together in such a way that they can influence each other, *couplers* are used to link qubits. They can make qubits end with the same, or opposite, states. This takes advantage of the quantum mechanical phenomenon of *entaglement* in which two qubits can be thought of as a single object with four different states, each corresponding to a different combination of the two qubits: (0,0), (0,1), (1,1), and (1,0).The relative energy of each state depends on the biases of qubits and the coupling between them, and in the end, the desired state will be the one with the lowest energy, as seen in Figure 2.4 .

2.2.3 Quantum Programming

2.2.3.1 Quantum programming languages and frameworks

There already exists a great variety of quantum programming languages in order to program quantum accelerators. A few examples are as follows:

- **Q# - Microsoft** [7]: Microsoft's open source programming language for running quantum algorithms. It is a part of its Quantum Development Kit (QDK) that in addition to the language and its standard library, it also includes libraries for chemistry and Machine Learning
- **Qiskit - IBM** [4]: is an open source Software development kit (SDK) developed/supported by IBM to work with quantum computers at the level of pulses, circuits and application modules.
- **OpenQASM** [13]: is an intermediate representation of quantum instructions. Basically *Assembly Language* for quantum circuits, its name "QASM" stands for "Quantum assembler".

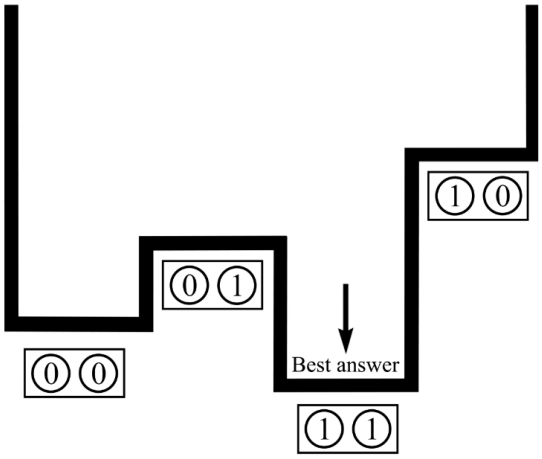


Figure 2.4: Energy diagram showing the desired state with the lowest energy, i.e. best answer (Credit: [6])

- **Quipper** [19]: A Functional and scalable quantum programming language that is suited to model of programming in which a classical computer controls a quantum device, making it possible to apply formal methods on quantum algorithms.

- **Cirq - Google** [1]: An open source framework for Python developed by Google that it is used to write, manipulate and optimize quantum circuits.

- **Ocean Software - Dwave** [2]: A suite of tools (SDK) that allow developers to solve hard problems using Dwave's adiabatic quantum computers (based on quantum annealing).

2.2.3.2 Quantum Algorithms

There are some quantum algorithms that may be considered "purely quantum" such as:

- **Grover's Algorithm**: A search algorithm that can find a and unsorted element in a list in $O(\sqrt{N})$ time as oposed to $O(N)$ time for classical algorithms [16].

- **Shor's Algorithm**: A quantum algorithm that can find a factor of a n-bit integer in time $O(n^2(\log n)(\log\log n))$. The best classical algorithm takes $O(e^{cn^{\frac{1}{3}}(\log n)^{\frac{2}{3}}})$ time [16].

The above algorithms are some of the most well known algorithms in quantum gate-based computing.

Most applications however, are hybrid solutions that employ both classical and quantum computing. As quantum computing evolves however, more specialized methods for testing and verifying these types of programs become necessary. There is already some developments in the field of quantum program testing and verification [20, 26] , and many more developments may emerge in

the coming years. Given the hybrid nature of many programs, many testing and verification methods that are used in classical computing [27, 14, 25] may be reused or adapted for these quantum programs.

2.2.4 Applications of Quantum computing

Since quantum computing is in its infancy, many of its applications have been more "proof-of-concept" than real world application. Most of this is due to the current state of hardware in quantum devices. But there have been development in several fields that use quantum computing as a tool:

Cryptography is one such field in which quantum computers may contribute and/or even render certain cryptographic protocols (such as RSA) obsolete [9]. Although much research in this field is still underway, any breakthrough may have repercussion in the field of personal privacy and even Cryptocurrency.

Machine Learning is another field in which quantum computing is used to speed up Machine Learning processes [11]. Successful application and "commercial deployment" of machine learning (and artificial intelligence in general) on quantum computers could have consequences that are difficult to predict.

These are two fields, out of many, that benefit from the application of quantum computing. Other fields include bioinformatics, materials science, finance, among others.

2.3 Summary

Quantum computing has proven itself to be a field with great potential, but still in its infancy and with much work ahead towards its "maturity", both in hardware and software. Although the issue of energy routing has not been explored in conjuction with quantum computing, some developments have been made in the field of vehicle routing [12] and the Tail Assignment Problem [24] make the prospects of its use in the routing of Garbage trucks more optimistic.

Chapter 3

Routing of Garbage Trucks

Routing Algorithms have become a cornerstone of our interconnected society. From finding the best routes in trading scenarios, to finding the best way to route information across different computer systems through the internet. In this chapter, the challenges that the routing of garbage trucks present and algorithms that tackle those same challenges are shown.

3.1 Background

Concerning waste collection, there are three major areas: residential waste collection, commercial waste collection and roll-on-roll-off. [21] The way that trucks are routed through the different pick-up points varies depending on the type of waste it is being collected.

In [21], a Vehicle routing problem with time windows (VRPTW) is introduced and applied to commercial waste collection, which takes into account collection stops with time-windows, drivers' lunch times, etc. In this chapter we will focus on some of the routing challenges present in this study that are relevant to the work.

3.2 Waste collection

3.2.1 Main areas of waste collection

The three main areas of waste collection have their own way of operating [21]:

In *residential waste collection* concerns mainly with servicing private homes. The number of those may vary between 150 to 1300 homes daily. The amount of times a given home is serviced however may vary according to weather, geography, etc.

In *commercial waste collection*, as the name implies, involves servicing mostly commercial costumers like small office building, restaurants and strip malls. The number of costumers may vary between 60-400 costumers and those same costumers may have specific time-windows for garbage collecting. The frequency of service depends on the type of costumer.

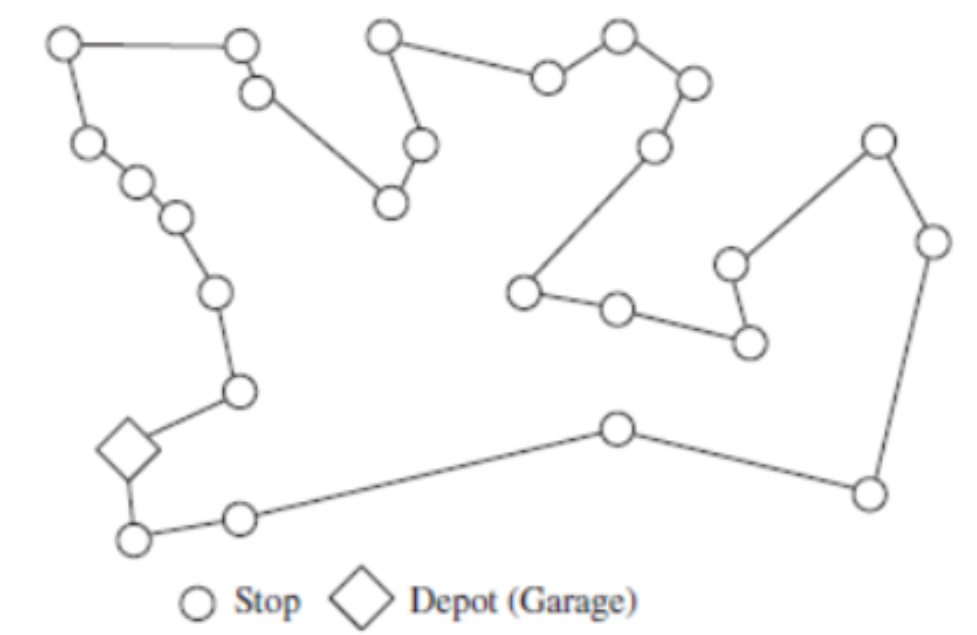


Figure 3.1: Routing of a garbage vehicle without considering the dump operations (from [21])

The *roll-on-roll-off waste collection* is very similar to the commercial waste collection but it differs on the size of the container, given that in this case the container is of a larger capacity. Many times, the truck will take an empty container to a costumer, leave it there and take the full container to the garbage disposal facility to empty it, and afterward take that same empty container to a costumer that uses a container of similar size.

The focus of this chapter concerns the challenges in *commercial waste collection*.

3.2.2 Approaches to truck routing

A first approach to this issue involves simply the routing of a garbage vehicle without considering the dump operations as it is shown in Figure 3.1.

This approach however is very simplistic but not very realistic.

The next iteration of this approach considers a single disposal facility as seen in Figure 3.2.

The final iteration (and the focus of the current approach) is one where several disposal facilities are considered, as seen in Figure 3.3.

3.2.2.1 Assumptions and simplifications

As seen in [21], there are a lot of factors that come into play concerning waste disposal. To determine the route between the the different garbage collection spots, it needs to be determined which spots a given truck will need to pass through. To address this problem, a first clustering phase must be performed, not only considering the truck capacity but also the proximity between points.This will also take into account the different disposal facilities as seen in Figure 3.3.

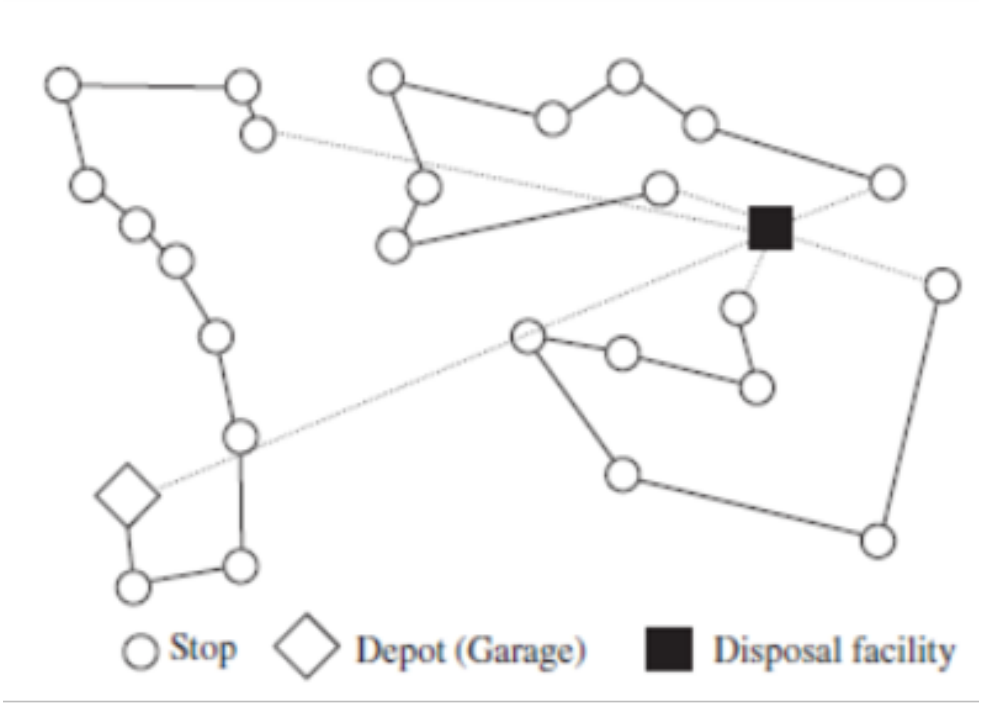


Figure 3.2: Routing of a garbage vehicle considering a single disposal facility (from [21])

Afterwards, small routes can be calculated between each depot or disposal facility, by calculating the distance between all the points in the route and then simplifying it to a complete graph (see Figure 4.2). Then a "travelling salesman problem" with restrictions can be applied.

3.3 Summary

The "Garbage truck" routing problem is a complex one that is part of daily life, be it in normal cities or *Smart Cities*. The problem of vehicle routing in itself is a computationally intensive task in some situations. Quantum computing is increasingly present in many aspects of our day-to-day lives, improving it in many ways, and its use in garbage truck routing (and other pratical applications) is of great interest.

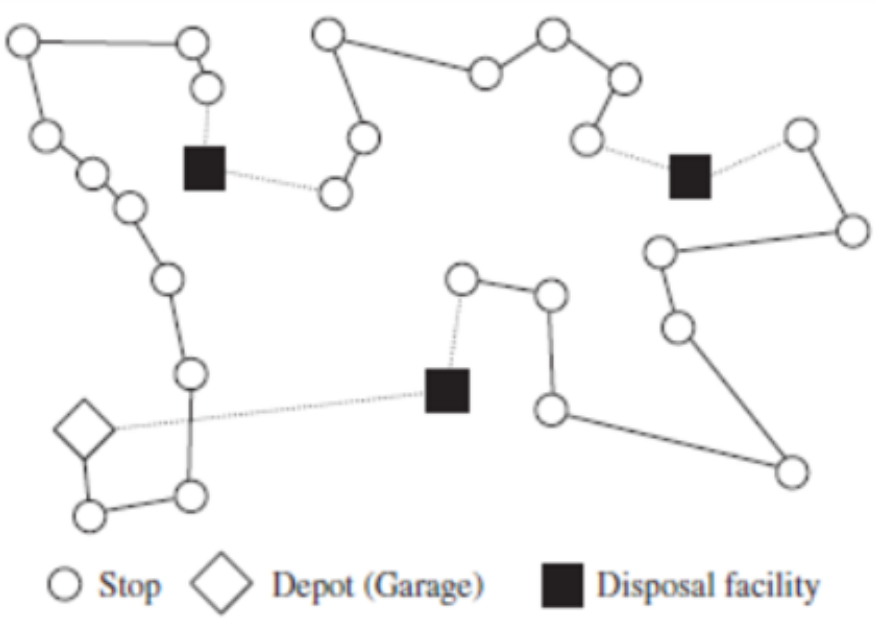


Figure 3.3: Routing of a garbage vehicle considering several disposal facilities (from [21])

Chapter 4

Methodology and Experimental work

4.1 Problem Description

As was described in the previous chapter, The routing of garbage trucks can be divided into several phases:

- The division of the different collection points into several clusters (according to the trucks capacity) with a disposal facility to serve as an endpoint.
- The routing of said truck between those different points and ending in a disposal facility, meaning that there is a specific starting position and ending position (which might not be the same)
- The disposal facility becomes a new starting location for another cluster and the previous step is repeated.

Although this process may involve more variables such as time-windows, this simplification is done in order to focus on the routing aspect. In Figure 4.1 the whole process can be seen and which part of the process the use of quantum computing will be focused. The second half of this process (after clustering) can be summarized in Figure 4.2, where a given graph (cluster) with its relevant points can, after determining the optimal routes between all relevant nodes by classical means, be simplified to a complete graph. This problem is similar to the one described by Feld et al. [15], with the main difference being the addition of certain restrictions regarding starting and ending nodes for the route calculations.

Although here the focus of the quantum computing efforts will be on the routing phase (via the modified travelling salesman), there is also the possibility of using the "Knapsack Problem" on the clustering phase [15] but it is not the focus of this work.

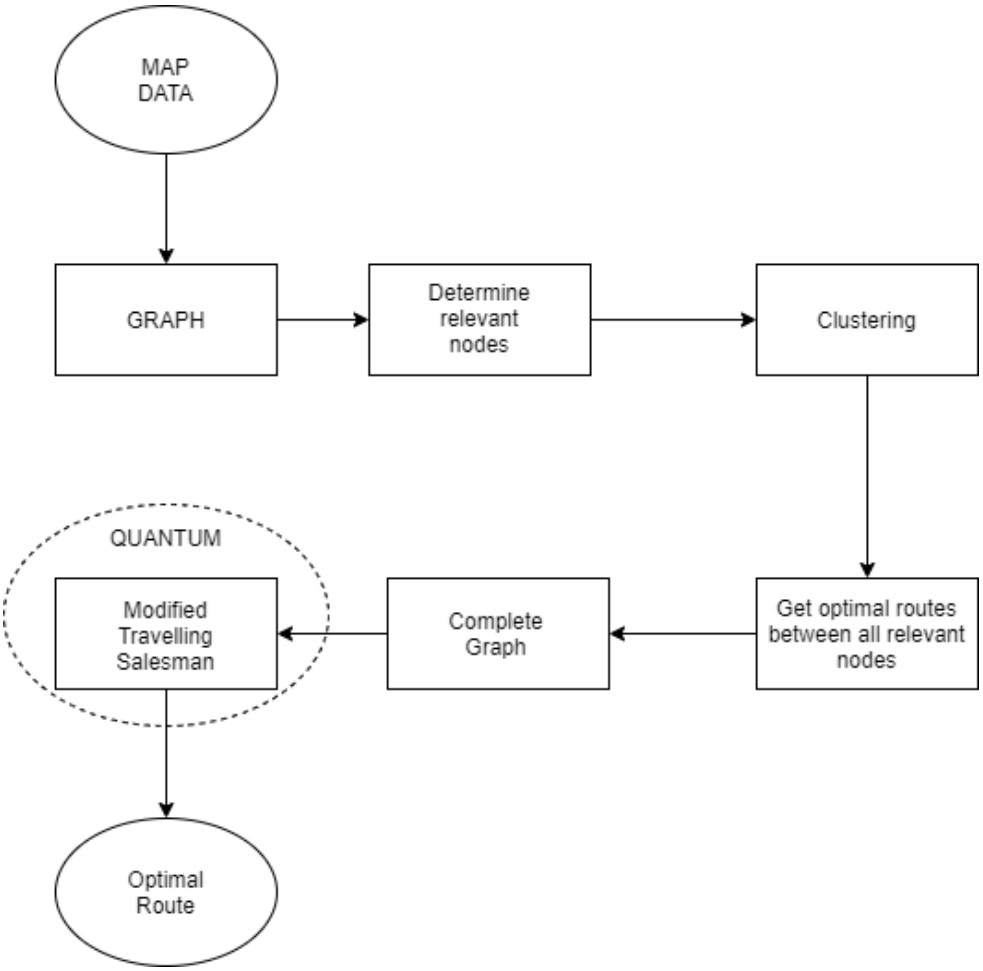


Figure 4.1: Simplified representation of the process to determine the routes of garbage trucks

4.2 Hardware

The approach taken is a hybrid one: while the main calculations for solving TSP are performed on the Quantum Computer, all the preprocessing and postprocessing required is performed on a classical computer. Each computer's features are presented as follows:

- Classical computer:
  - **CPU:** Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz
  - **RAM:** 32 Gb
  - **GPU:** NVIDIA GeForce GTX 670MX
- Quantum Processor - Dwave Advantage System:
  - **Type:** Quantum Annealer
  - **Number of Qubits:** 5000

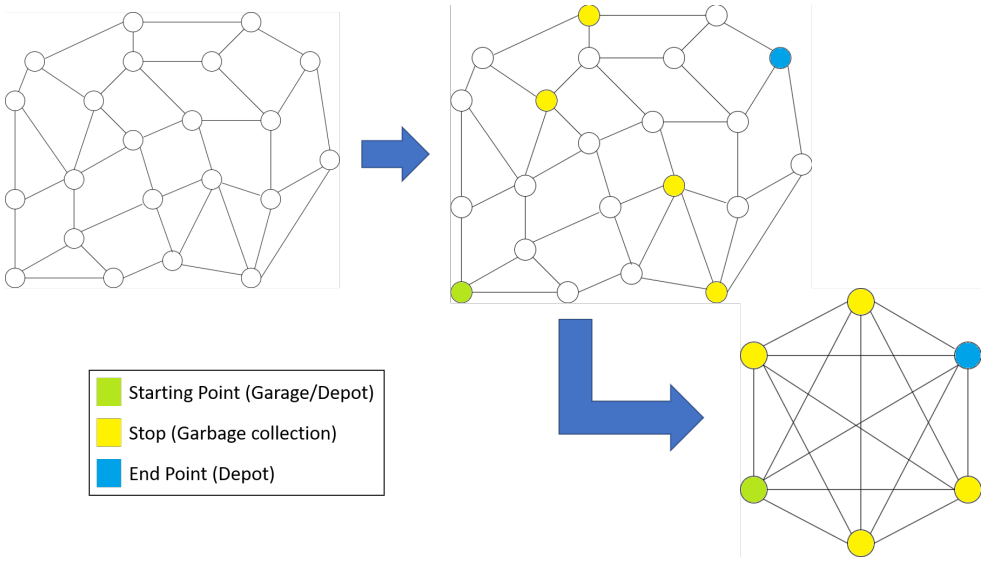


Figure 4.2: From normal graph with relevant nodes to a simplified complete graph

- **Qubit connectivity:** 15

Processing time is a factor that is studied and as such the specifications of the hardware used is important.

4.3 Approach

This can be approached as a "Travelling Salesman Problem" (TSP) but with the caveat that we have a fixed starting and ending point, which will be assumed that are not the same.

The "Travelling Salesman Problem" is described as: "A salesman has to visit a certain number of cities, visiting each city only once. Given the distances between each pair of cities, in what order does the salesman need to visit the cities, in order to minimize the distance travelled"

This is an NP-Hard problem.

4.3.1 Mathematical Representation

In order to use the DWave QPU, the given problem must be represented as a Binary Quadratic Model (BQM). Every optimization problem needs an objective and constraints, for the traditional "Travelling Salesman Problem" the following objectives and constraints are applied:

- **Objective:** Minimize the total distance for the route
- **Constraint:** Visit each node (or "city") exactly once

In the problem at hand, the truck must start its journey on the truck depot (henceforth considered to be node 0 on every graph) and must visit all other nodes on the graph ending in the disposal

facility (considered to be the node "n-1" on a graph with "n" nodes), as such the following constraints were added:

Additional Constraints:

- The first node in every route must be node 0
- the Last node in every route must be node "n-1" in a graph with "n" nodes.

Let a graph G be represented as $G = (V, E)$ (V - Vertices, E - Edges) and $N = |V|$ where:

- **x :** represents the different nodes (or vertices).
- **(u,v) :** represent the node number.
- **j :** represents the "order" of a given node in the route.
- W_{uv} **:** represents the weight of the edge between nodes "u" and "v".

Both the objective and the constraints can be expressed mathematically as:

- **Objective:**

$$min(\sum_{(uv)\in E} W_{uv} \sum_{j=1}^{N} x_{u,j} x_{v,j+1}) \tag{4.1}$$

- **Constraints:**

$$\sum_{u=1}^{N} x_{u,j} = 1 \tag{4.2}$$

$$\sum_{j=1}^{N} x_{u,j} = 1 \tag{4.3}$$

$$x_{1,1} = 1 \tag{4.4}$$

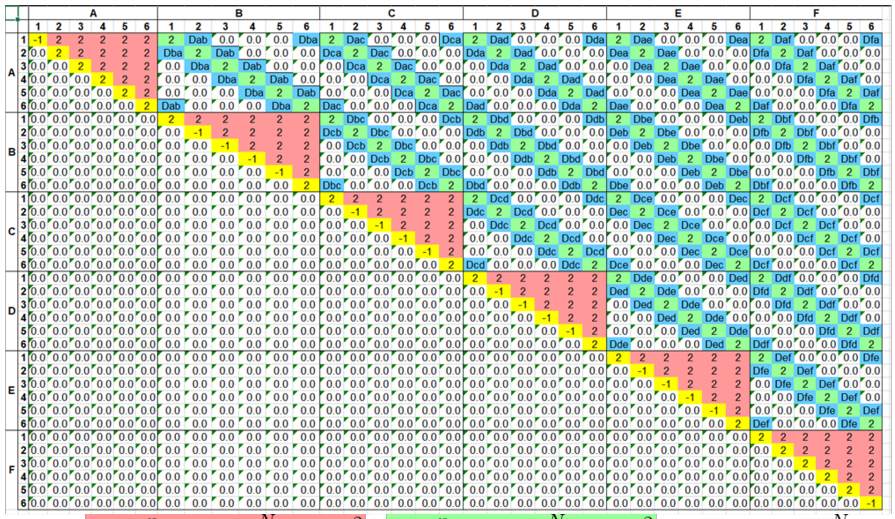
$$x_{N,N} = 1 \tag{4.5}$$

This is the traditional mathematical representation of the problem. However to work with the Dwave QPU, this problem and constraints must be expressed as a quadratic, unconstrained binary optimization (QUBO). In [22], many QUBO formulations for NP-complete and NP-hard problems are defined, including the "Travelling Salesman Problem". This is expressed, acording to [22] as:

$$H_A = A \sum_{v=1}^{n} (1 - \sum_{j=1}^{N} x_{v,j})^2 + A \sum_{j=1}^{n} (1 - \sum_{v=1}^{N} x_{v,j})^2 + A \sum_{(uv)\notin E} \sum_{j=1}^{N} x_{u,j} x_{v,j+1} \tag{4.6}$$

$$H_B = B \sum_{(uv)\in E} W_{uv} \sum_{j=1}^{N} x_{u,j} x_{v,j+1} \tag{4.7}$$

$$H = H_A + H_B \tag{4.8}$$



$$H_A = A\sum_{v=1}^{n}(1-\sum_{j=1}^{N} x_{v,j})^2 + A\sum_{j=1}^{n}(1-\sum_{v=1}^{N} x_{v,j})^2 + A\sum_{(uv)\notin E}\sum_{j=1}^{N} x_{u,j}x_{v,j+1}$$

$$H_B = B\sum_{(uv)\in E} W_{uv}\sum_{j=1}^{N} x_{u,j}x_{v,j+1}$$

$$H_C = A(x_{1,1}-1)^2 + A(x_{n,n}-1)^2$$

Figure 4.3: Example of a QUBO Matrix and how each equation influences it.

To this traditional formulation, a third factor must be added to account for the constraint concerning the first and last node of the route.

$$H_C = A(x_{1,1}-1)^2 + A(x_{n,n}-1)^2 \tag{4.9}$$

The values of "A" and "B" can be though of as "penalty" values that ensures that the constraints and limitations are not violated and the total energy is minimized. According to Lucas [22], $0 < Bmax(W_{uv}) < A$, and although several values for "A" and "B" were experimented (respecting the restriction set forth by Lucas [22]), it was found that the optimal value for the experiments ahead were the values used by Feld et al [15], in which $B = 1$ and $A = (number\_of\_nodes) * max(W_{uv})$.

These expressions are expressed in a QUBO Matrix as seen in Figure 4.3.

4.3.2 Preprocessing

In the preprocessing phase, the graph data is provided and, depending on whether method is being used to calculate (classical or quantum), different preprocessing methods are used. In the **classical** method, the graph is translated into a ".lp" file to use with the "SCIP" solver (to get the optimal route value for randomly generated graphs), but with the **quantum** method the graph is adapted into a QUBO matrix.

4.3.3 Classical Solvers

In order to evaluate the quality and precision of the results obtained by the QPU, the problems need to be solved in a classical manner to get the actual results and use them as a comparison to

the quantum results.

4.3.3.1 SCIP

SCIP [18] is a mixed integer (linear and nonlinear) programming solver, inserted in the "SCIP Optimization Suite 7.0". Although it has many functionalities, the main one that was used in this work involves:

- stating the problem in an ".lp" file with a given objective function (minimize/maximize) and the constraints
- running that file with the SCIP solver
- SCIP returns an ".sol" file with the solution

Below, in listing 4.1, an example of an ".lp" file and the corresponding ".sol" file (in listing 4.2) is shown for a random complete graph with 5 nodes. In the first one (".lp" file) the "problem statement" can be seen and in the second one (".sol" file), the respective solution can be found.

```
1 Minimize
2  obj:  [ 14 x0_1 * x1_2 ]/2 +  [ 14 x1_1 * x0_2 ]/2 +  [ 14 x0_2 * x1_3 ]/2 +  [ 14
       x1_2 * x0_3 ]/2 +  [ 14 x0_3 * x1_4 ]/2 +  [ 14 x1_3 * x0_4 ]/2 +  [ 14 x0_4
      * x1_5 ]/2 +  [ 14 x1_4 * x0_5 ]/2 +  [ 30 x0_1 * x2_2 ]/2 +  [ 30 x2_1 * x0_2
       ]/2 +  [ 30 x0_2 * x2_3 ]/2 +  [ 30 x2_2 * x0_3 ]/2 +  [ 30 x0_3 * x2_4 ]/2 +
         [ 30 x2_3 * x0_4 ]/2 +  [ 30 x0_4 * x2_5 ]/2 +  [ 30 x2_4 * x0_5 ]/2 +  [ 2
      x0_1 * x3_2 ]/2 +  [ 2 x3_1 * x0_2 ]/2 +  [ 2 x0_2 * x3_3 ]/2 +  [ 2 x3_2 *
      x0_3 ]/2 +  [ 2 x0_3 * x3_4 ]/2 +  [ 2 x3_3 * x0_4 ]/2 +  [ 2 x0_4 * x3_5 ]/2
      +  [ 2 x3_4 * x0_5 ]/2 +  [ 10 x0_1 * x4_2 ]/2 +  [ 10 x4_1 * x0_2 ]/2 +  [ 10
       x0_2 * x4_3 ]/2 +  [ 10 x4_2 * x0_3 ]/2 +  [ 10 x0_3 * x4_4 ]/2 +  [ 10 x4_3
      * x0_4 ]/2 +  [ 10 x0_4 * x4_5 ]/2 +  [ 10 x4_4 * x0_5 ]/2 +  [ 18 x1_1 * x2_2
       ]/2 +  [ 18 x2_1 * x1_2 ]/2 +  [ 18 x1_2 * x2_3 ]/2 +  [ 18 x2_2 * x1_3 ]/2 +
         [ 18 x1_3 * x2_4 ]/2 +  [ 18 x2_3 * x1_4 ]/2 +  [ 18 x1_4 * x2_5 ]/2 +  [ 18
       x2_4 * x1_5 ]/2 +  [ 6 x1_1 * x3_2 ]/2 +  [ 6 x3_1 * x1_2 ]/2 +  [ 6 x1_2 *
      x3_3 ]/2 +  [ 6 x3_2 * x1_3 ]/2 +  [ 6 x1_3 * x3_4 ]/2 +  [ 6 x3_3 * x1_4 ]/2
      +  [ 6 x1_4 * x3_5 ]/2 +  [ 6 x3_4 * x1_5 ]/2 +  [ 8 x1_1 * x4_2 ]/2 +  [ 8
      x4_1 * x1_2 ]/2 +  [ 8 x1_2 * x4_3 ]/2 +  [ 8 x4_2 * x1_3 ]/2 +  [ 8 x1_3 *
      x4_4 ]/2 +  [ 8 x4_3 * x1_4 ]/2 +  [ 8 x1_4 * x4_5 ]/2 +  [ 8 x4_4 * x1_5 ]/2
      +  [ 12 x2_1 * x3_2 ]/2 +  [ 12 x3_1 * x2_2 ]/2 +  [ 12 x2_2 * x3_3 ]/2 +  [
      12 x3_2 * x2_3 ]/2 +  [ 12 x2_3 * x3_4 ]/2 +  [ 12 x3_3 * x2_4 ]/2 +  [ 12
      x2_4 * x3_5 ]/2 +  [ 12 x3_4 * x2_5 ]/2 +  [ 26 x2_1 * x4_2 ]/2 +  [ 26 x4_1 *
       x2_2 ]/2 +  [ 26 x2_2 * x4_3 ]/2 +  [ 26 x4_2 * x2_3 ]/2 +  [ 26 x2_3 * x4_4
      ]/2 +  [ 26 x4_3 * x2_4 ]/2 +  [ 26 x2_4 * x4_5 ]/2 +  [ 26 x4_4 * x2_5 ]/2 +
       [ 24 x3_1 * x4_2 ]/2 +  [ 24 x4_1 * x3_2 ]/2 +  [ 24 x3_2 * x4_3 ]/2 +  [ 24
      x4_2 * x3_3 ]/2 +  [ 24 x3_3 * x4_4 ]/2 +  [ 24 x4_3 * x3_4 ]/2 +  [ 24 x3_4 *
       x4_5 ]/2 +  [ 24 x4_4 * x3_5 ]/2
3 Subject To
4  c1: x0_1 + x0_2 + x0_3 + x0_4 + x0_5 = 1
5  c2: x1_1 + x1_2 + x1_3 + x1_4 + x1_5 = 1
```

```
 c3: x2_1 + x2_2 + x2_3 + x2_4 + x2_5 = 1
 c4: x3_1 + x3_2 + x3_3 + x3_4 + x3_5 = 1
 c5: x4_1 + x4_2 + x4_3 + x4_4 + x4_5 = 1
 c6: x0_1 + x1_1 + x2_1 + x3_1 + x4_1 = 1
 c7: x0_2 + x1_2 + x2_2 + x3_2 + x4_2 = 1
 c8: x0_3 + x1_3 + x2_3 + x3_3 + x4_3 = 1
 c9: x0_4 + x1_4 + x2_4 + x3_4 + x4_4 = 1
 c10: x0_5 + x1_5 + x2_5 + x3_5 + x4_5 = 1
 c11: x0_1 = 1
 c12: x4_5 = 1
Binary
 x0_1
 x0_2
 x0_3
 x0_4
 x0_5
 x1_1
 x1_2
 x1_3
 x1_4
 x1_5
 x2_1
 x2_2
 x2_3
 x2_4
 x2_5
 x3_1
 x3_2
 x3_3
 x3_4
 x3_5
 x4_1
 x4_2
 x4_3
 x4_4
 x4_5
End
```

Listing 4.1: Sample lp file

```
solution status: optimal solution found
objective value:                                20
x0_1                                             1   (obj:0)
x1_4                                             1   (obj:0)
x2_3                                             1   (obj:0)
x3_2                                             1   (obj:0)
x4_5                                             1   (obj:0)
quadobjvar                                      20   (obj:1)
```

Listing 4.2: Sample sol file

The use of this tool implies its own preprocessing (creating the ".lp" file dynamically) and postprocessing (reading and extracting the solutions from the ".sol" file)

4.3.4 Quantum Solver

After expressing the problem as a QUBO and making the QUBO matrix, the logical qubits (the square of the number of nodes) are embedded to the physical qubits, i.e. 2 logical qubits may require 4 physical ones (2 physical qubits for each logical one). Although there are several types of embedding, in this work an automatic embeddder ("EmbeddingComposite") was used for simplicity.

The "chain-strenght" parameter was dynamically set as 5 times the parameter A, since for problems of different sizes, the "chain strength" needs to be different. Although there are some heuristics in setting this parameter, the best way to find the best value is by experimentation, and this way proved the best for this specific case.

4.3.5 Limitations

Although the previous two methods of calculation (SCIP Quantum Solver) were initially used, these proved to have some limitations of their own.

The initial idea was to generate random complete graphs and have SCIP provide the optimal solution and compare it with the solution given by the Quantum Solver. The problem however was one of scale-up. With a complete graph of 5 nodes (25 variables) , the QPU would return the optimal value around 60-80% of the time, any higher and that percentage would decrease. At a higher number of nodes (around 10) the embedding would get more complex, resulting in warnings from the QPU such as: "Chain length greater than 7", or even the impossibility of embedding the problem altogether.

At 15 nodes or more, SCIP also becomes a limitation since the calculations can take a very long time to calculate the optimal solution (sometimes a whole day depending on the problem).

In order to have optimal solutions for larger graphs, the database TSPLIB [5] was used. It has various examples of complete graphs with the optimal solutions for the "travelling salesman problem" for each of them.

As for the limitation regarding the QPU, a decomposing solver, named QBsolv was used.

4.3.6 QBsolv

QBsolv is a "decomposing solver that finds a minimum value of a large quadratic unconstrained binary optimization (QUBO) problem by splitting it into pieces." These pieces can then be solved by classical means using a tabu algorithm or using the Dwave QPU.

This results in performing a lot of small calculations instead of a larger one. This has several parameters as well that one can set. The ones used are as follows:

- **solver limit** (the maximum number of variables each piece has): 20
- **num repeats** (the number of times each calculation is repeated): 50 (default value)

The main issue with this method is that with large problems, it can require a large number of calculations to give a result (sometimes thousands), requiring a lot of QPU time to perform said calculations (dozens of seconds) and with Dwave's limit on 1 minute of QPU time per month per account, some of the results with larger graphs are indicative and not a thorough analysis (see next chapter). This same time limitation is the reason that the "num repeats" is 50, since increasing it could consume more of the QPU time available. Although this can have an influence in problems involving larger graphs, it is a reasonable trade-off considering all factors and the scope of this work.

4.3.7 Post Processing

In the postprocessing phase, after gathering the results of the QPU (either traditional solver or QBSolv), the different solutions are:

1. Checked for validity
2. Checked to see if they are the minimum among all the results

In the end, the minimum route for a given graph is obtained.

Chapter 5

Results and Discussions

In this chapter, the results obtained and their discussion is presented. Starting with some preliminary results involving randomly generated complete graphs and proceeding to results using graphs from the TSPLib database, that have known solutions.

5.1 Results using Randomly Generated Graphs

In Table 5.1 some preliminary results using randomly generated graphs can be observed. The SCIP solver was used to determine the ideal route and the minimum value for a given graph and that value is compared to the results obtained by the classic version of the QBsolv solver, as well as the quantum version of the same solver using the Dwave QPU.

Experiment	Number of nodes	SCIP Result	QbSolv classic	QbSolv Quantum
1	5	18	18	18
2	10	30	30	31
3	15	34	46	46

Table 5.1: Results using Randomly Generated Graphs and the minimum value for the route

As can be seen in the table, Qbsolv does not always return the optimal result and sometimes the classic and quantum results using the Qbsolv solver do not return the same result, this last point may be due to the inherent uncertainty of present quantum computers.

5.2 Results using Graphs from TSPLib

Given that for graphs with a larger number of nodes it becomes unfeasible to use SCIP to obtain the minimum route for a given graph (it takes too much time), the following results were obtained using graphs from TSPLib where the best known solution is readily available.

Due to the fact that Qbsolv uses a lot of QPU time, the results are divided into two sections: Classical Results and Quantum Results. In the classical results, Qbsolv (using the tabu algorithm)

		QBSolv classic Results				Difference	
Sample	**Nodes**	**Minimum**	**Average**	**max**	**BKS**	**min and BKS**	**Avg and BKS**
burma	14	3498	3768.28	4077	3323	5.27%	13.40%
ulysses	22	6069	7013.79	7568	7013	-13.46%	0.01%
bayg	29	2146	2468.35	2617	1610	33.29%	53.31%

Table 5.2: Variance of Qbsolv classic results compared to Best Know Solution (BKS)

was run 100 times for each of the studied samples in order to study how the results vary, since although it uses classical computing, the results are not deterministic for this solver. In the quantum results, due to the limitations of QPU time, one of the samples was repeated in order to see how the values varied and the other were performed only once as proof of concept.

The chosen samples were "burma14", "ulysses22" and "bayg29". The number in the respective names are representative of the number of nodes in the graph.

5.2.1 Classical Results

In the following graphs, the results of using Qbsolv with the tabu algorithm are shown and are compared with the "Best Known Solution" and the "Random route solution" (the average route if the "salesman" chose which node to visit randomly) that is expressed in equation 5.1.

$$random\_route = \sum_{(uv)\in E} W_{uv}/N \tag{5.1}$$

In Table 5.2 it can be seen how much a minimum value obtained and the average value differ from the Best known Solution (BKS).

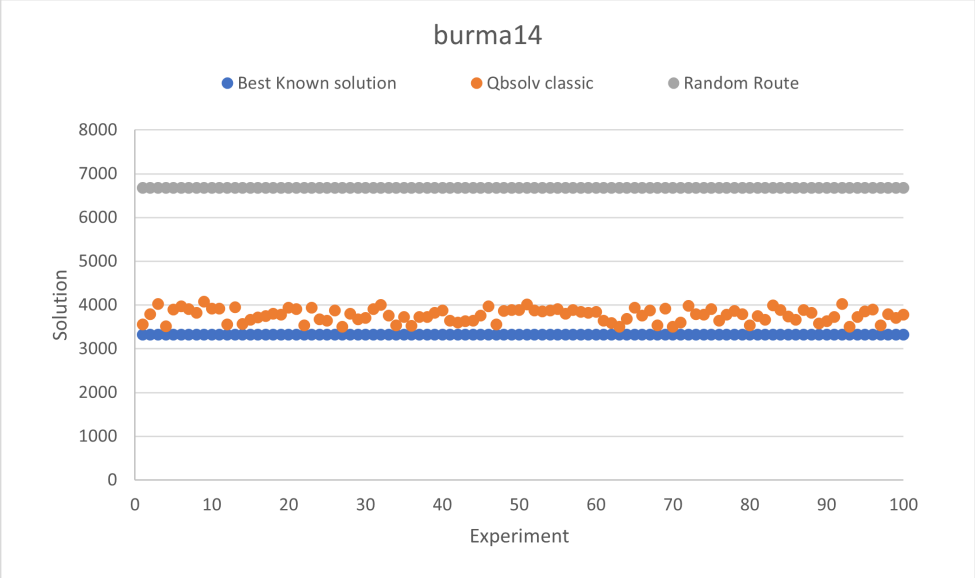


Figure 5.1: Results of Qbsolv (classic) for the "burma14" graph

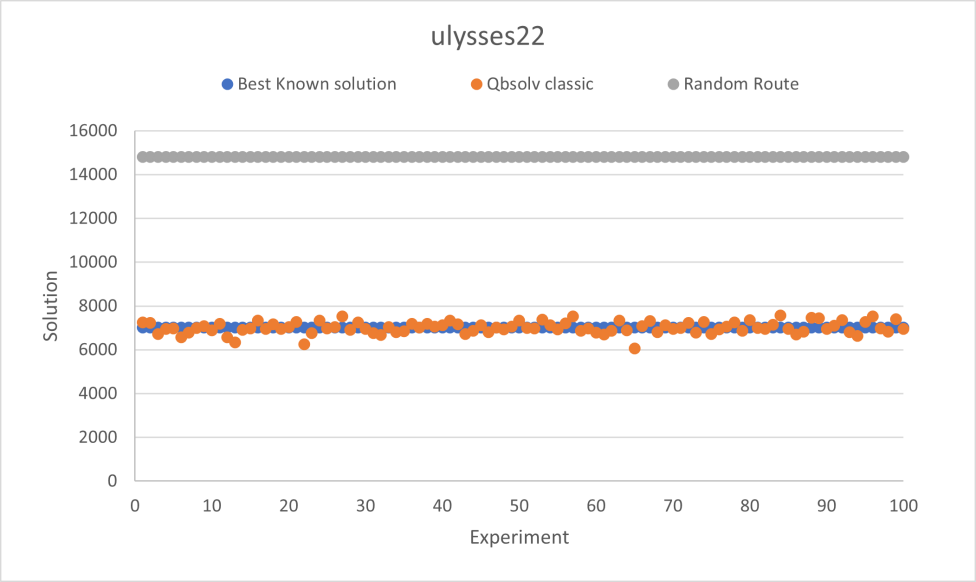


Figure 5.2: Results of Qbsolv (classic) for the "ulysses22" graph

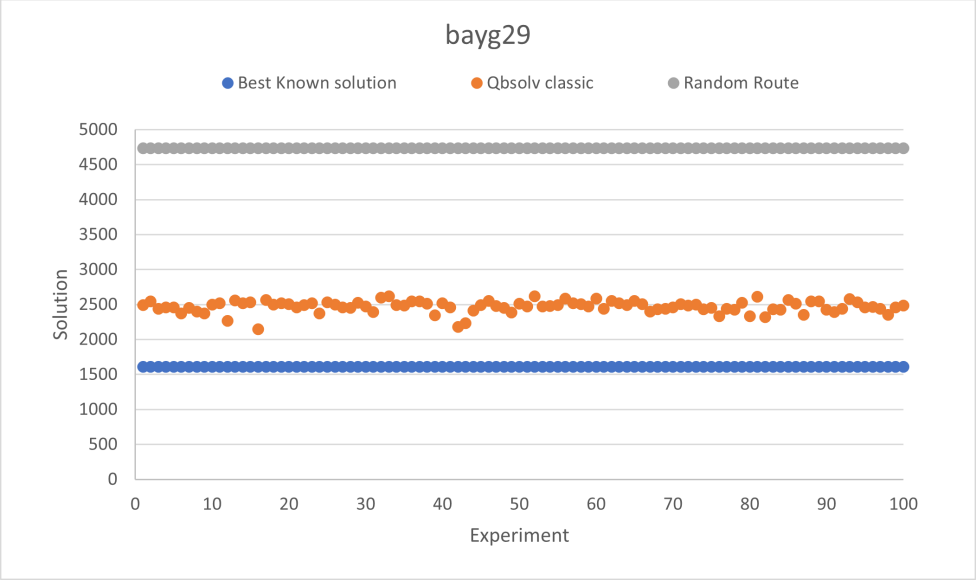


Figure 5.3: Results of Qbsolv (classic) for the "bayg29" graph

5.2.2 Quantum Results

Due to the limitation of QPU time, the only sample that was repeated was "burma14" since it is the smallest sample and its QPU time would be the smallest, and the results of how much the quantum results vary compared to the classical ones can be seen. In Figure 5.4 these results are shown.

In Figure 5.5, the QPU time and classic processing time can be seen as a function of the number of nodes. All the raw results can be seen in Table 5.3, including the random route solution and the Best Known Solution (BKS) for the different samples. The Quantum time (or "QPU time") shown

					Results		Time (s)	
Experiment	**Sample**	**Nodes**	**BKS**	**Random**	**Classic**	**Quantum**	**Classic**	**Quantum**
1	burma14	14	3323	6672.15	3637	3793	4.73	8.15
2	burma14	14	3323	6672.15	3828	3498	6.80	6.12
3	burma14	14	3323	6672.15	3616	3530	5.78	5.11
4	burma14	14	3323	6672.15	3742	3933	6.31	6.31
5	burma14	14	3323	6672.15	4016	3722	6.81	6.26
6	burma14	14	3323	6672.15	3680	3725	7.06	6.26
7	burma14	14	3323	6672.15	3508	3553	6.27	6.16
8	burma14	14	3323	6672.15	3576	3850	5.87	5.55
9	burma14	14	3323	6672.15	3962	3732	5.82	5.06
10	burma14	14	3323	6672.15	3880	3765	6.57	4.64
11	burma14	14	3323	6672.15	3782	3515	7.32	8.84
12	burma14	14	3323	6672.15	3870	3846	6.00	4.97
13	burma14	14	3323	6672.15	3698	3815	6.66	4.59
14	burma14	14	3323	6672.15	3749	3764	6.16	4.73
15	burma14	14	3323	6672.15	3849	3701	6.45	8.61
16	bayg29	29	1610	4736.64	2481	2531	9.55	17.19
17	ulysses22	22	7013	14809.62	7027	6745	7.30	14.43

Table 5.3: Results for the experiments using the QPU

in that table was measured directly on the Quantum Computer Web Interface made available by DWave ("DWave Leap") after each experiment.

5.3 Discussion

As can be seen in all the graphs (figures 5.1, 5.2 and 5.3), the results for the classic Qbsolv, although not ideal, it can be considered "good enough" since they are close to the best known solution and much better than the "random route" value.

The "ulysses22" sample however, both in the classical results (Figure 5.2) and also in the quantum result (Table 5.3) presents a solution that is better than the "Best Known Solution". This should be impossible since the postprocessing phase, ensures that all the results are valid and it is assumed that there is no better solution than the "Best Know solution". One other possibility is the calculation of the weight of the edges calculated wrong, but at the time of writing, no obvious error was found in that regard.

The other classical results, as expected, as a problem gets more and more complex (more nodes and consequently more variables) the more it deviates from the ideal solution as can be seen in Table 5.2 (excluding the aforementioned "ulysses 22" sample that is an outlier). Although not ideal, the results are still much better than using a "random route" approach, making these results not "ideal" but "good enough" for many instances.

As for the quantum results, we can see in Figure 5.4 that the results using Qbsolv don't differ much whether using a classic method or the QPU. But as can be seen in Figure 5.5 and in Table

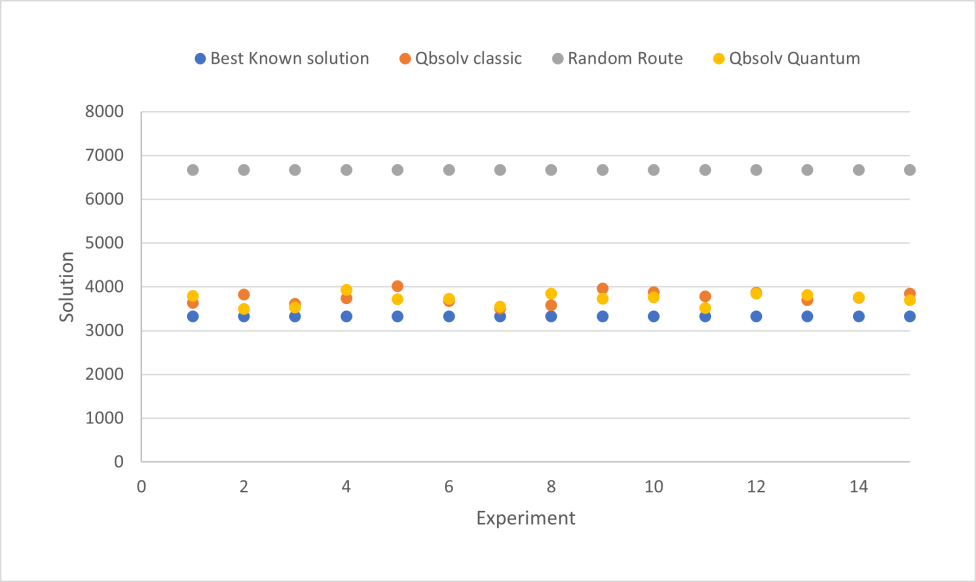


Figure 5.4: Results of Qbsolv (quantum) for the "burma14" graph

5.3, the QPU time has some variance, but for more complex problems, the classic CPU seems to solve Qbsolv faster than the QPU (looking at the samples with 22 and 29 nodes and an issue that has been discussed in forum posts [3]). But these are single results, and a more thorough study of more complex problems is needed in order to reach a definitive conclusion regarding the QPU performance, which was not possible in this case due to the limits that Dwave imposes on QPU time.

Another limitation is the hardware itself. Although there have been improvements to Dwave's QPU (from the 2000Q to the Advantage system) in terms of total number of qubits and their connectivity, it still proves difficult to embed into the physical qubits problems with a large number of variables, requiring the use of solvers such as Qbsolv (that divides the problem into smaller problems) that even when using the classical methods, still are non deterministic and may not return the ideal solution.

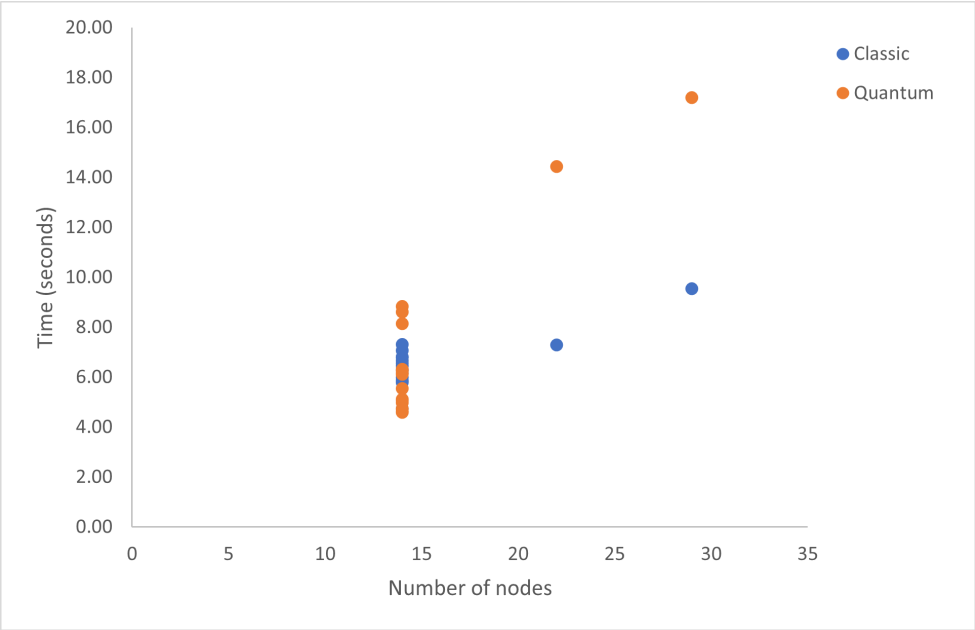


Figure 5.5: Classic time and QPU time in relation to the number of nodes

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Quantum computing as a field is still very much in its infancy (pre-transistor phase) and the many results in this work show that, in this specific work, it didn't manage to outperform a classical computer. However the QPU managed to provide "good enough" results, that in many situations is all that is required. In terms of processing time, the QPU did not manage to beat the CPU in finding a solution, but further developments and Software Engineering may change that in the Future.

Concerning the routing problem, the application of a modified "Travelling Saleman Problem" in order to find the best route for garbage trucks from the garage, to the Depot, passing through all the pick up points, proved to be not ideal but "good enough" and even small improvements to a given route, can bring great benefits long-term.

6.2 Future Work

Although a step in the right direction, there are still some areas where quantum computing could be applied to the garbage pick-up problem, specifically:

- The "clustering" phase could possibly also be solved with quantum computing through the "knapsack problem"
- The addition of variables concerning time-windows.
- Adaptation of the algorithms to the arc-routing problems.

These future steps could provide a more accurate and useful algorithm that takes advantage of all that quantum computing has to offer and even make its use viable in a commercial setting.

References

[1] Cirq | Google Quantum AI. https://quantumai.google/cirq. [Online; accessed 31-01-2021].

[2] D-Wave Ocean Software Documentation — Ocean Documentation 3.3.0 documentation. https://docs.ocean.dwavesys.com/en/stable/. [Online; accessed 12-02-2021].

[3] dwave-hybrid vs. qbsolv: really which works best? – D-Wave Systems. https://support.dwavesys.com/hc/en-us/community/posts/360052899913-dwave-hybrid-vs-qbsolv-really-which-works-best-. [Online; accessed 25-07-2021].

[4] Qiskit. https://qiskit.org/. [Online; accessed 31-01-2021].

[5] TSPLIB. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/. [Online; accessed 18-05-2021].

[6] Welcome to D-Wave — D-Wave System Documentation documentation. https://docs.dwavesys.com/docs/latest/c_gs_1.html. [Online; accessed 11-02-2021].

[7] What are the Q# programming language and QDK? - Microsoft Quantum | Microsoft Docs. https://docs.microsoft.com/en-us/quantum/overview/what-is-qsharp-and-qdk?view=qsharp-preview. [Online; accessed 05-11-2020].

[8] Juhar Abdella, Khaled Shuaib, and Saad Harous. Energy Routing Algorithms for the Energy Internet. In *9th International Conference on Intelligent Systems 2018: Theory, Research and Innovation in Applications, IS 2018 - Proceedings*, pages 80–86. Institute of Electrical and Electronics Engineers Inc., jul 2018.

[9] Daniel J. Bernstein. Introduction to post-quantum cryptography. In *Post-Quantum Cryptography*, pages 1–14. Springer Berlin Heidelberg, jan 2009.

[10] Koen Bertels, A. Sarkar, T. Hubregtsen, M. Serrao, A.A. Mouedenne, A. Yadav, A. Krol, I. Ashraf, and C. Garcia Almudever. Quantum Computer Architecture Toward Full-Stack Quantum Accelerators. *IEEE Transactions on Quantum Engineering*, 1:1–17, 2020.

[11] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning, sep 2017.

[12] Michał Borowski, Paweł Gora, Katarzyna Karnas, Mateusz Błajda, Krystian Król, Artur Matyjasek, Damian Burczyk, Miron Szewczyk, and Michał Kutwin. New hybrid quantum annealing algorithms for solving vehicle routing problem. In *Lecture Notes in Computer*

Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 12142 LNCS, pages 546–561. Springer, jun 2020.

[13] Andrew W. Cross, Lev S. Bishop, John A. Smolin, and Jay M. Gambetta. Open Quantum Assembly Language. jul 2017.

[14] Saikat Dutta, Owolabi Legunsen, Zixin Huang, and Sasa Misailovic. Testing probabilistic programming systems. In *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, volume 13, pages 574–586, New York, New York, USA, oct 2018. Association for Computing Machinery, Inc.

[15] Sebastian Feld, Christoph Roch, Thomas Gabor, Christian Seidel, Florian Neukart, Isabella Galter, Wolfgang Mauerer, and Claudia Linnhoff-Popien. A Hybrid Solution Method for the Capacitated Vehicle Routing Problem Using a Quantum Annealer. *Frontiers in ICT*, 6(JUN):13, jun 2019.

[16] Elias Fernandez-Combarro Alvarez. A practical introduction to quantum computing: from qubits to quantum machine learning and beyond. Dec 2020. https://cds.cern.ch/record/2748136 [Online; accessed 31-01-2021].

[17] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, jun 1982.

[18] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin, March 2020.

[19] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: A Scalable Quantum Programming Language. *ACM SIGPLAN Notices*, 48(6):333–342, apr 2013.

[20] Yipeng Huang and Margaret Martonosi. Statistical Assertions for Validating Patterns and Finding Bugs in Quantum Programs. *Proceedings - International Symposium on Computer Architecture*, pages 541–553, may 2019.

[21] Byung In Kim, Seongbae Kim, and Surya Sahoo. Waste collection vehicle routing problem with time windows. *Computers and Operations Research*, 33(12):3624–3642, dec 2006.

[22] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2:1–14, feb 2014.

[23] Dennis Morris. *Quantum Mechanics: An Introduction*. Mercury Learning and Information, Dulles, Virginia, dec 2017.

[24] Luís Noites Martins, Ana Paula Rocha, and António Castro. Applying Quantum Annealing to the Tail Assignment Problem. Technical report, aug 2020.

[25] Alexandre Perez, Rui Abreu, and Arie Van Deursen. A Theoretical and Empirical Analysis of Program Spectra Diagnosability. *IEEE Transactions on Software Engineering*, pages 1–1, jan 2019.

[26] Robert Rand. *Formally Verified Quantum Programming (PhD Thesis)* . Publicly Accessible Penn Dissertations University, 2018.

[27] W. Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. A survey on software fault localization. *IEEE Transactions on Software Engineering*, 42(8):707–740, aug 2016.