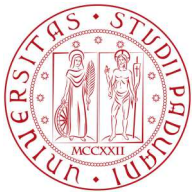


University of Padova

Department of Mathematics

Master Thesis in Data Science

Quantum Integer Programming for the Capacitated Vehicle Routing Problem

Supervisor

Professor Francesco Rinaldi

University of Padova

Master Candidate

Anthony Palmieri

Academic Year

2022-2023

Dedicated to my cherished family and friends, whose unwavering presence has been my source of strength through the highs and lows of these years. A heartfelt tribute to my mom, whose boundless support and sacrifices have paved the way for me to pen these words today.

Abstract

Quantum computing, with its inherent speedups, holds the promise of revolutionizing optimization problems by providing faster and more efficient solutions. In this study, we explore the application of quantum optimization techniques to tackle the challenging Vehicle Routing Problem (VRP) and its popular variant, the Capacitated VRP (CVRP).
The VRP is a combinatorial optimization problem centered on determining the most efficient routes for a fleet of vehicles to deliver goods to a set of customers, while minimizing either the total travel distance or overall cost. Given its classification as a well-known NP-hard problem, finding optimal solutions for larger instances can be computationally demanding. By leveraging the power of quantum computing, we aim to explore novel approaches and algorithms that can potentially offer significant improvements in solving VRP instances.
To formulate the VRP, we employ the classical two-index approach, where binary variables (x_{ij}) indicate whether an arc (i, j) is traversed by a vehicle. To ensure meaningful solutions, we incorporate subtour elimination constraints. Although these constraints improve the final solutions, they quickly become problematic when transitioning to QUBO formulation, as they introduce many integer slack variables, resulting in an excessive number of variables for gate-based quantum models. To address this issue, we also explore a heuristic two-phase approach.
According to this heuristic, nodes are first clustered into disjoint groups, with each cluster containing the depot. Routing is then performed individually within each cluster.
During the clustering phase, we investigate various methodologies. Initially, we approach the problem as a Multi Knapsack Problem (MKP) with a quadratic objective function. However, this formulation presents challenges for gate-based quantum algorithms, which struggle to navigate the complex energy landscape of the MKP.
Subsequently, we address the clustering phase by employing a customized version of the Louvain Modularity Maximization Algorithm, enhanced with classical greedy subroutines aimed at aligning the clusters with the forthcoming routing phase.
Additionally, we explore an alternative approach where we model the clustering phase as a Modularity Maximization problem, which is tackled using quantum algorithms. To ensure coherence with the subsequent routing phase, we introduce two supplementary classical greedy subroutines to refine the clusters identified through Modularity Maximization.
The routing phase is instead modelled as a Travelling Salesman Problem (TSP). Here, we sequentially add subtour elimination constraints as needed (*DFJ strategy*) rather than adding all the constraints in advance (*MTZ strategy*). This adaptive approach signifi-

cantly reduces the number of qubits, making the models suitable for both gate-based and annealing-based quantum devices.
For quantum optimization, we investigate various quantum algorithms, including Quantum Adiabatic Computation, Quantum Approximate Optimization Algorithm (QAOA), and Variational Quantum Eigensolver (VQE) with hardware-efficient ansatz. To gauge the performance of these quantum algorithms, we employ Gurobi as a classical benchmarking method. Furthermore, owing to the current hardware limitations of NISQ devices, our focus is primarily on instances of relatively small size.
By analyzing these quantum approaches and comparing them with classical methods, we aim to gain insights into the potential quantum speedups for solving complex VRP instances. With ongoing advancements in quantum hardware development, hopefully, quantum optimization will become a robust strategy for real-world transportation and logistics applications.

Contents

Listing of figures

1 Introduction

In the ever-evolving landscape of computing, quantum computing has emerged as a revolutionary paradigm, holding the promise to reshape the frontiers of traditional computational methodologies. Harnessing the principles of quantum mechanics, quantum computing leverages quantum bits (qubits) to represent and process information in fundamentally different ways from classical computers. Two distinct paradigms have emerged: Quantum Annealers and Gate-Based Models. These models represent distinct approaches to harnessing the power of quantum mechanics for computation, each with its own strengths and limitations.

Quantum Annealers, such as those offered by D-Wave Systems, are designed to solve optimization problems by leveraging quantum fluctuations to search for low-energy states in a problem landscape. These machines are particularly well-suited for combinatorial optimization problems.

In contrast, Gate-Based Quantum Computers, like those developed by IBM, Google, and other research institutions, operate through a sequence of quantum logic gates to perform quantum computations. These gate-based models offer greater flexibility in algorithm design and can execute a wider range of quantum algorithms. Quantum gates can be combined to create complex quantum circuits capable of implementing various algorithms, including those aimed at solving optimization problems.

Indeed, one particularly intriguing application of quantum computing lies in the realm of Operations Research and Mathematical Optimization. Many real-world challenges, from supply chain management to financial portfolio optimization, hinge on solving intricate optimization problems. Quantum computing's inherent ability to explore multiple possibilities simultaneously, driven by concepts like superposition and entanglement, has fueled excitement about its potential to revolutionize optimization algorithms.

A quintessential problem that encapsulates the complexity of real-world optimization challenges is the Vehicle Routing Problem (VRP). In its essence, the VRP seeks to deter-

mine the most efficient routes for a fleet of vehicles to deliver goods to a set of locations, minimizing costs while adhering to some constraints like travel time and vehicle capacities. The VRP finds wide applications in logistics, transportation, and distribution networks, where optimizing routes can lead to substantial savings in resources, time, and environmental impact.
The VRP, however, is not just an industrial puzzle; it stands as a formidable combinatorial optimization challenge.
While the VRP has numerous variants, we chose the Capacitated VRP as our focus due to its significance in transportation and logistics, and its tractable size for investigation on current quantum hardware. As of the present state of quantum technology, constraints such as qubit decoherence and limited connectivity pose challenges in dealing with larger and more complex VRP variants. Our choice of the CVRP enables a meaningful exploration of quantum-assisted optimization while considering the constraints of current quantum hardware capabilities.
Classical approaches to solving the VRP span a spectrum of techniques. These include heuristic methods such as Nearest Neighbor, Savings Algorithm, and Clarke-Wright savings algorithm. Metaheuristics like Genetic Algorithms, Simulated Annealing, and Tabu Search have also been extensively employed. Exact algorithms such as Branch and Bound, and Branch and Cut are employed for smaller instances where optimality is crucial. However, as problems'size grow, the limits of classical computing become evident, prompting the exploration of alternative computational paradigms.
This Master's thesis ventures into relatively unexplored territories, investigating quantum-assisted approaches for the Vehicle Routing Problem. Our goal is to investigate the potential of quantum computing techniques to rival classical methods in solving VRP instances. While various quantum optimization algorithms exist, our core focus centers on Adiabatic Quantum Optimization (AQO), Quantum Approximate Optimization Algorithm (QAOA), and Variational Quantum Eigensolver (VQE).
Through experimental analysis, we seek to uncover insights into the feasibility and advantages of quantum-assisted optimization in tackling the CVRP, contributing to the growing body of knowledge at the intersection of quantum computing and combinatorial optimization.
This thesis is structured as follows: Chapter 2 provides a review of key linear algebra concepts relevant to quantum computing, with a specific focus on Hilbert Spaces and tensor products.
In Chapter 3, we delve into an introductory exploration of quantum mechanics and quantum computing. This chapter covers foundational principles such as superposition, entanglement, the Bloch sphere, and quantum circuits. Within this chapter, we present the essential arsenal of gates that forms the bedrock of quantum computing. This ensemble encompasses Pauli gates, Hadamard gates, Toffoli gates, and CNOT gates, each playing a pivotal role in quantum computations.

Chapter 4 marks our transition to the realm of combinatorial optimization. Here, we revisit the basics of combinatorial optimization problems and introduce the mathematical frameworks underpinning quantum optimization, namely QUBO and Ising formulations. This chapter elucidates the transformation of integer problems into QUBO and outlines the bidirectional conversion between QUBO and Ising models. We also address the challenges related to qubit embedding.
Transitioning into Chapter 5, our focus shifts towards critical quantum optimization algorithms. These encompass Adiabatic Quantum Optimization, Variational Quantum Eigensolver (VQE), and the Quantum Approximate Optimization Algorithm (QAOA). Within this chapter, we delve into the foundational principles that underpin these algorithms, exploring their theoretical underpinnings, variations, and optimization techniques documented in the literature.
Chapter 6 shifts our attention to the Vehicle Routing Problem (VRP), particularly the Capacitated VRP (CVRP). We discuss the unique challenges associated with these combinatorial optimization problems, present commonly used formulations, and introduce the specific formulation employed in our study. Additionally, we provide an overview of both exact and heuristic methods commonly utilized for solving such problems. This chapter introduces a two-step heuristic approach that combines clustering and routing strategies, a pivotal component of our numerical experiments.
In Chapter 7, we present a comprehensive account of our experiments, methodologies, and the results derived from quantum-assisted optimization techniques.
Finally, the concluding chapter summarizes our findings, discusses their implications, and reflects on the future potential of quantum-assisted optimization within the domain of the VRP. Through this journey, our aim is to contribute to the evolving landscape of quantum optimization and its applicability to real-world combinatorial challenges.

2
Linear Algebra Recap

2.1 Hilbert Spaces and Linear Operators

Our exploration into the realm of quantum computing begins with an examination of the fundamental mathematical underpinnings of quantum mechanics. At the heart of this discussion lies the concept of Hilbert space, a mathematical construct of paramount importance and extraordinary power and versatility. Let us start with a definition

Definition 2.1.1
*A **Hilbert space** $\mathbb{H}$ is a*

1. *complete complex vector space, that is,*

$$\Psi, \Phi \in \mathbb{H} \text{ and } a, b \in \mathbb{C} \implies a\Psi + b\Phi \in \mathbb{H}, \tag{2.1}$$

2. *with a (positive-definite) **scalar product***

$$\begin{aligned} \langle \cdot | \cdot \rangle : \mathbb{H} \times \mathbb{H} &\to \mathbb{C} \\ (\Psi, \Phi) &\mapsto \langle \Psi | \Phi \rangle \end{aligned} \tag{2.2}$$

such that for all $\Phi, \Psi, \Phi_1, \Phi_2 \in \mathbb{H}$ and $a, b \in \mathbb{C}$

$$\langle \Psi | \Phi \rangle = \overline{\langle \Phi | \Psi \rangle} \tag{2.3}$$
$$\langle \Psi | \Psi \rangle \geq 0 \tag{2.4}$$
$$\langle \Psi | \Psi \rangle = 0 \Leftrightarrow \Psi = 0 \tag{2.5}$$
$$\langle \Psi | a\Phi_1 + b\Phi_2 \rangle = a\langle \Psi | \Phi_1 \rangle + b\langle \Psi | \Phi_2 \rangle \tag{2.6}$$

and this scalar product induces a ***norm*** *in which* $\mathbb{H}$ *is* ***complete***

$$\begin{aligned} \|\cdot\| : \mathbb{H} &\to \mathbb{R} \\ \Psi &\mapsto \sqrt{\langle \Psi | \Psi \rangle}. \end{aligned} \tag{2.7}$$

A subset $\mathbb{H}_{sub} \subset \mathbb{H}$ *which is a vector space and inherits the scalar product and the norm from* $\mathbb{H}$ *is called a sub Hilbert space or simply a* ***subspace*** *of* $\mathbb{H}$.

Completeness of $\mathbb{H}$ in the norm $\|\cdot\|$ means, that every in $\mathbb{H}$ Cauchy-convergent sequence $(\Phi_n)_{n\in\mathbb{N}} \subset \mathbb{H}$ has a limit $\lim_{n\to\infty} \Phi_n = \Phi \in \mathbb{H}$ that also lies in $\mathbb{H}$.
As we will see, the vectors we deal with in quantum computing are normalized vectors which lie in a tensor product of many Hilbert spaces. In our study, such vectors will be written as linear combination of orthonormal vectors known as the **Computational Basis**. It is then worth giving also the following definitions.

Definition 2.1.2
A vector $\Psi \in \mathbb{H}$ *is called* ***normed*** *or a* ***unit vector*** *if* $\|\Psi\| = 1$. *Two vectors* $\Psi, \Phi \in \mathbb{H}$ *are called* ***orthogonal*** *to each other if* $\langle \Psi | \Phi \rangle = 0$.

Definition 2.1.3
Let $\mathbb{H}$ *be a Hilbert space and* I *an index set. A set* $\{\Phi_j \mid j \in I\} \subset \mathbb{H}$ *of vectors is called* ***linearly independent*** *if for every finite subset* $\{\Phi_1, \Phi_2, \ldots, \Phi_n\}$ *and* $a_k \in \mathbb{C}$ *with* $k = 1, \ldots, n$

$$a_1\Phi_1 + a_2\Phi_2 + \cdots + a_n\Phi_n = 0$$

holds if and only if $a_1 = a_2 = \cdots = a_n = 0$.
A Hilbert space $\mathbb{H}$ *is called* ***finite-dimensional*** *if* $\mathbb{H}$ *contains at most* $\dim \mathbb{H} < \infty$ *linearly independent vectors; otherwise* $\mathbb{H}$ *is called infinite-dimensional.*
A set of vectors $\{\Phi_j \mid j \in I\} \subset \mathbb{H}$ *is said to* ***span*** $\mathbb{H}$ *if for every vector* $\Phi \in \mathbb{H}$ *there are* $a_j \in \mathbb{C}$ *with* $j \in I$ *such that*

$$\Phi = \sum_{j\in I} a_j \Phi_j.$$

A linearly independent set $\{\Phi_j \mid j \in I\}$ *of vectors that spans* $\mathbb{H}$ *is called a* ***basis*** *of* $\mathbb{H}$ *and the vectors* Φ_j *of such a set are called basis vectors. A basis* $\{e_j \mid j \in I\} \subset \mathbb{H}$ *whose*

vectors satisfy

$$\langle e_j | e_k \rangle = \delta_{jk} := \begin{cases} 0 & \text{if} \quad j \neq k \\ 1 & \text{if} \quad j = k \end{cases} \tag{2.8}$$

is called an ***orthonormal basis (ONB)****. The Hilbert space* $\mathbb{H}$ *is called* ***separable*** *if it has a countable basis.*

In general quantum mechanics makes use of infinite-dimensional Hilbert spaces. However, in order to understand the aspects of quantum mechanics relevant for quantum computing it is sufficient to consider only finite-dimensional Hilbert spaces, which are always complete.
The dual space $\mathbb{H}^*$ of a separable Hilbert space $\mathbb{H}$ is also a vector space with the same dimension as $\mathbb{H}$. This identification of $\mathbb{H}$ with $\mathbb{H}^*$ motivates the "bra" and "ket" notation derived from the word "bracket" and introduced by Dirac. Bra-vectors are elements of $\mathbb{H}^*$ and are written as $\langle \Phi |$. Ket-vectors are elements of $\mathbb{H}$ and are written as $|\Psi\rangle$. Because of the above-mentioned bijection between the Hilbert space $\mathbb{H}$ and its dual space $\mathbb{H}^*$, each vector $|\Phi\rangle \in \mathbb{H}$ corresponds to a vector in $\mathbb{H}^*$, which is then denoted as $\langle \Phi |$. In the following, we embrace this distinctive Dirac notation, prominent in quantum mechanics and quantum computing.

Definition 2.1.4
A linear map $A : \mathbb{H} \to \mathbb{H}$ *is called an* ***operator*** *on the Hilbert space* $\mathbb{H}$*. The set of all operators on* $\mathbb{H}$ *is denoted by* $\mathcal{L}(\mathbb{H})$*. The operator* $A^\dagger : \mathbb{H} \to \mathbb{H}$ *that satisfies*

$$\langle A^\dagger \Psi | \Phi \rangle = \langle \Psi | A \Phi \rangle \qquad \forall |\Psi\rangle, |\Phi\rangle \in \mathbb{H} \tag{2.9}$$

is called the ***adjoint*** *operator to* A*. If* $A^\dagger = A$ *then* A *is called* ***self-adjoint****.*

In the finite-dimensional case self-adjoint is the same as **hermitian**.

Definition 2.1.5
An operator $A \in \mathcal{L}(\mathbb{H})$ *is called* ***normal*** *if* $AA^\dagger = A^\dagger A$*.*

As we are about to see, Quantum Computing entirely revolves around a particular type of linear operators, namely unitary operators.

Definition 2.1.6
*An operator U on $\mathbb{H}$ is called **unitary** if*

$$\langle U\Psi|U\Phi\rangle = \langle\Psi|\Phi\rangle \qquad \forall|\Psi\rangle,|\Phi\rangle \in \mathbb{H}. \tag{2.10}$$

The set of all unitary operators on $\mathbb{H}$ is denoted by $\mathcal{U}(\mathbb{H})$.

This definition basically tells us that unitary operators preserve inner products in the Hilbert space and thus also norm of vectors. This means that applying unitary transformations to a norm one vector will always result in a norm one vector. Operators which preserve the inner product are also called **isometries**.
Let us also recall the definition of eigenstates of an operator.

Definition 2.1.7
*Let A be an operator on a Hilbert space $\mathbb{H}$. A scalar value $\lambda \in \mathbb{C}$ is called **eigenvalue** of A if there exists a vector $|\Psi\rangle \in \mathbb{H} \setminus \{0\}$ such that*

$$A|\Psi\rangle = \lambda|\Psi\rangle, \tag{2.11}$$

*$|\Psi\rangle$ is called **eigenvector** of A with respect to the eigenvalue λ.*
*The linear subspace that is spanned by all eigenvectors for a given eigenvalue λ of an operator A is called **eigenspace** of λ and denoted by $Eig(A,\lambda)$. The set*

$$\sigma(A) := \{\lambda \in \mathbb{C} \mid \lambda \text{ eigenvalue of } A\} \tag{2.12}$$

*is called the **spectrum** of the operator A.*

Eigenvalues of self-adjoint operators are always real and the eigenvalues of unitary operators always have absolute value 1.
Let us also recall one of the most important theorems in applied linear algebra, the **(Complex) Spectral Theorem**.

Theorem 2.1.1
Let $\mathbb{H}$ be a finite-dimensional inner product space over $\mathbb{C}$ and $A \in \mathcal{L}(\mathbb{H})$. Then A is normal if and only if there exists an orthonormal basis for $\mathbb{H}$ consisting of eigenvectors for A.

2.2 Tensor Product of Hilbert Spaces

We conclude this chapter recalling the definition and properties of the tensor product. In the realm of quantum mechanics, the intricate interplay between multiple particles is

at the heart of understanding the behavior of complex systems. Grasping how these particles interact is pivotal, and this is where the tensor product of Hilbert spaces emerges as a foundational mathematical tool.
When we focus on individual particles, Hilbert spaces provide the mathematical framework to describe their quantum states and behaviors. However, as we venture into the realm of composite systems comprising multiple particles, the intricacy of their interactions demands a more comprehensive approach. The tensor product of Hilbert spaces steps in to meet this demand by extending the formalism to encompass these interconnected entities.
By employing the tensor product, we expand our mathematical landscape to represent not just individual particle states, but also the joint states of the entire composite system. This formalism elegantly captures how particles influence one another and form collective behaviors, enabling us to explore the complex dynamics that emerge in such systems. This becomes particularly pertinent in scenarios involving entanglement, where particles instantaneously share correlations regardless of spatial separation.
In this context, we present a less formal elucidation of the tensor product involving two finite-dimensional Hilbert spaces, which aligns with our specific objectives. While a broader definition exists, our primary focus is on defining the computational rules for tensor products, rather than delving deeply into the mathematical framework.
Let $\mathbb{H}^A$ and $\mathbb{H}^B$ be Hilbert spaces, $|\Phi\rangle \in \mathbb{H}^A$ and $|\Psi\rangle \in \mathbb{H}^B$ vectors in these, and define

$$\begin{aligned} |\Phi\rangle \otimes |\Psi\rangle : &\mathbb{H}^A \times \mathbb{H}^B \to \mathbb{C} \\ &(\xi,\ \eta) \mapsto \langle \xi|\Phi\rangle_{\mathbb{H}^A} \langle \eta|\Psi\rangle_{\mathbb{H}^B} \end{aligned} \tag{2.13}$$

where the subscripts indicate in which Hilbert space the scalar product is to be calculated. In the following we omit such subscripts to ease the notation.
This map is *anti-linear* and *continuous* in ξ and η. Define the set of all such maps and denote it by

$$\mathbb{H}^A \otimes \mathbb{H}^B := \{\Psi : \mathbb{H}^A \times \mathbb{H}^B \to \mathbb{C} \mid \text{anti-linear and continuous}\}. \tag{2.14}$$

This is a vector space over $\mathbb{C}$ since for $\Psi_1, \Psi_2 \in \mathbb{H}^A \otimes \mathbb{H}^B$ and $a, b \in \mathbb{C}$ the map defined by

$$(a\Psi_1 + b\Psi_2)(\xi, \eta) := a\Psi_1(\xi, \eta) + b\Psi_2(\xi, \eta) \tag{2.15}$$

is also in $\mathbb{H}^A \otimes \mathbb{H}^B$. The null-map is the null-vector, and $-\Psi$ is the additive-inverse vector for Ψ. According to (2.13) then $|\Phi\rangle \otimes |\Psi\rangle$ is a vector in the vector space of the anti-linear and continuous maps $\mathbb{H}^A \otimes \mathbb{H}^B$ from $\mathbb{H}^A \times \mathbb{H}^B$ to $\mathbb{C}$ as defined in (2.14). In order to simplify the notation, from now on we shall also write $|\Phi \otimes \Psi\rangle := |\Phi\rangle \otimes |\Psi\rangle$.
We now want to define a scalar product on such space.

For vectors $|\Phi_k\rangle \otimes |\Psi_k\rangle \in \mathbb{H}^A \otimes \mathbb{H}^B$ with $k \in \{1,2\}$ and $|\Phi_k\rangle \in \mathbb{H}^A, |\Psi_k\rangle \in \mathbb{H}^B$ we define

$$\langle \Phi_1 \otimes \Psi_1 | \Phi_2 \otimes \Psi_2 \rangle := \langle \Phi_1 | \Phi_2 \rangle_{\mathbb{H}^A} \langle \Psi_1 | \Psi_2 \rangle_{\mathbb{H}^B}. \tag{2.16}$$

With (2.16) we have a scalar product for vectors of the form $|\Phi\rangle \otimes |\Psi\rangle \in \mathbb{H}^A \otimes \mathbb{H}^B$. In order to define a scalar product for all $\Psi \in \mathbb{H}^A \otimes \mathbb{H}^B$ we consider Orthonormal Basis (ONBs) in the subspaces. Let $\{|e_a\rangle\}_a \subset \mathbb{H}^A$ be an ONB in $\mathbb{H}^A$ and $\{|f_b\rangle\}_b \subset \mathbb{H}^B$ be an ONB in $\mathbb{H}^B$. The set $\{|e_a\rangle \otimes |f_b\rangle\}_{a,b} \subset \mathbb{H}^A \otimes \mathbb{H}^B$ is then orthonormal since

$$\langle e_{a_1} \otimes f_{b_1} | e_{a_2} \otimes f_{b_2} \rangle = \langle e_{a_1} | e_{a_2} \rangle_{\mathbb{H}^A} \langle f_{b_1} | f_{b_2} \rangle_{\mathbb{H}^B} = \delta_{a_1 a_2} \delta_{b_1 b_2}. \tag{2.17}$$

Considering an arbitrary vector $\Psi \in \mathbb{H}^A \otimes \mathbb{H}^B$, one has for this anti-linear map

$$\begin{aligned} \Psi(\xi, \eta) &= \Psi\left(\sum_a |e_a\rangle\langle e_a|\xi\rangle, \sum_b |f_b\rangle\langle f_b|\eta\rangle \right) \\ &= \sum_{a,b} \underbrace{\Psi(|e_a\rangle, |f_b\rangle)}_{=:\Psi_{ab} \in \mathbb{C}} \langle \xi | e_a \rangle \langle \eta | f_b \rangle \\ &= \sum_{a,b} \Psi_{ab} \cdot (|e_a \otimes f_b\rangle)(\xi, \eta). \end{aligned}$$

This proves that every vector $\Psi \in \mathbb{H}^A \otimes \mathbb{H}^B$ can be written as a linear combination of the form

$$\Psi = \sum_{a,b} \Psi_{ab} |e_a \otimes f_b\rangle. \tag{2.18}$$

The scalar product of Ψ in (2.18) and a vector $\Phi = \sum_{a,b} \Phi_{ab} |e_a \otimes f_b\rangle$ is then defined as

$$\begin{aligned} \langle \Psi | \Phi \rangle &= \sum_{a_1,b_1} \sum_{a_2,b_2} \overline{\Psi_{a_1 b_1}} \Phi_{a_2 b_2} \langle e_{a_1} \otimes f_{b_1} | e_{a_2} \otimes f_{b_2} \rangle \\ &= \sum_{a,b} \overline{\Psi_{ab}} \Phi_{ab}. \end{aligned} \tag{2.19}$$

The scalar product thus defined on all of $\mathbb{H}^A \otimes \mathbb{H}^B$ is positive-definite and independent of the choice of the ONBs. The bra-vector belonging to $|\Psi\rangle$ in (2.18) is then

$$\langle \Psi | = \sum_{a,b} \overline{\Psi_{ab}} \langle e_a \otimes f_b | \tag{2.20}$$

and acts as in (2.19) on a $|\Phi\rangle \in \mathbb{H}^A \otimes \mathbb{H}^B$. The norm of $|\Psi\rangle \in \mathbb{H}^A \otimes \mathbb{H}^B$ is calculated as

$$\||\Psi\||^2 = \langle\Psi|\Psi\rangle = \sum_{a,b} |\Psi_{ab}|^2 \tag{2.21}$$

and for any $|\Phi\rangle \in \mathbb{H}^A, |\Psi\rangle \in \mathbb{H}^B$ we have

$$\||\Phi \otimes \Psi\rangle\| = \sqrt{\langle\Phi \otimes \Psi|\Phi \otimes |\Psi\rangle} = \sqrt{\langle\Phi|\Phi\rangle\langle\Psi|\Psi\rangle} = \|\Phi\| \cdot \|\Psi\|. \tag{2.22}$$

Hence, $\mathbb{H}^A \otimes \mathbb{H}^B$ is a complex vector space with scalar product (2.19), which induces a norm (2.21). For finite-dimensional subspaces then $\mathbb{H}^A \otimes \mathbb{H}^B$ is complete in this norm and thus a Hilbert space. For our purposes it suffices to view $\mathbb{H}^A \otimes \mathbb{H}^B$ as the set of linear combinations of the form (2.18), with $\sum_{a,b} |\Psi_{ab}|^2 < \infty$ and the calculation rules (2.19) and (2.21).

Definition 2.2.1
*The Hilbert space $\mathbb{H}^A \otimes \mathbb{H}^B$ with the scalar product (2.19) is called the **tensor product** of the Hilbert spaces $\mathbb{H}^A$ and $\mathbb{H}^B$.*

Proposition 2.2.1
Let $\{|e_a\rangle\}_a \subset \mathbb{H}^A$ be an ONB in $\mathbb{H}^A$ and $\{|f_b\rangle\}_b \subset \mathbb{H}^B$ be an ONB in $\mathbb{H}^B$. The set $\{|e_a \otimes f_b\rangle\}_{a,b} = \{|e_a\rangle \otimes |f_b\rangle\}_{a,b}$ forms an ONB in $\mathbb{H}^A \otimes \mathbb{H}^B$, and for finite-dimensional $\mathbb{H}^A, \mathbb{H}^B$ one has

$$\dim\left(\mathbb{H}^A \otimes \mathbb{H}^B\right) = \dim \mathbb{H}^A \cdot \dim \mathbb{H}^B. \tag{2.23}$$

Proposition 2.2.2 (Properties of Tensor Product)
Let $|\Phi\rangle \in \mathbb{H}^A, |\Psi\rangle \in \mathbb{H}^B, a, b \in \mathbb{C}$, then the following identities hold

$$\begin{aligned}
(a|\Phi\rangle) \otimes |\Psi\rangle &= |\Phi\rangle \otimes (a|\Psi\rangle) = a(|\Phi\rangle \otimes |\Psi\rangle) \\
a(|\Phi\rangle \otimes |\Psi\rangle) + b(|\Phi\rangle \otimes |\Psi\rangle) &= (a+b)(|\Phi\rangle \otimes |\Psi\rangle) \\
(|\Phi_1\rangle + |\Phi_2\rangle) \otimes |\Psi\rangle) &= |\Phi_1\rangle \otimes |\Psi\rangle + |\Phi_2\rangle \otimes |\Psi\rangle \\
|\Phi\rangle \otimes (|\Psi_1\rangle + |\Psi_2\rangle) &= |\Phi\rangle \otimes |\Psi_1\rangle + |\Phi\rangle \otimes |\Psi_2\rangle.
\end{aligned}$$

The approach we have outlined extends to larger products of Hilbert Spaces as long as they are finite in number. However, we will avoid diving deeper into the details of this theoretical derivation. The finer points of this theory aren't our main focus; what truly matters is the method for calculating tensor products between vectors. Nevertheless,

the comprehensive theory in this chapter assures that our computations in subsequent sections align with fundamental theoretical principles. The primary reference for this chapter is [1], which provides an excellent mathematically rigorous perspective on quantum computing.
We conclude this chapter with a numerical example to show how to compute tensor products in practice.

2.3 Numerical Example

In this example, we consider two Hilbert spaces, $\mathbb{H}^A = \mathbb{H}^B = \mathbb{C}^2$, both of dimension 2. The orthonormal bases (ONBs) for both spaces are represented by the standard basis in $\mathbb{C}^2$:

$$\{|e_1\rangle, |e_2\rangle\} = \{|f_1\rangle, |f_2\rangle\} = \left\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right\}.$$

In quantum computing, we often use a simplified notation for these basis vectors:

$$|0\rangle := \begin{pmatrix}1\\0\end{pmatrix}, \qquad |1\rangle := \begin{pmatrix}0\\1\end{pmatrix}. \tag{2.24}$$

Now, for any $j \in \{1, 2\}$, consider complex coefficients a_j and b_j, and define quantum states as:

$$|\phi_1\rangle = a_1|0\rangle + b_1|1\rangle = \begin{pmatrix}a_1\\b_1\end{pmatrix}$$
$$|\phi_2\rangle = a_2|0\rangle + b_2|1\rangle = \begin{pmatrix}a_2\\b_2\end{pmatrix}.$$

To compute their tensor product, multiply each component of $|\phi_1\rangle$ with the entire vector $|\phi_2\rangle$ and stack these vectors vertically:

$$|\phi_1\rangle \otimes |\phi_2\rangle = \begin{pmatrix}a_1 a_2\\a_1 b_2\\b_1 a_2\\b_1 b_2\end{pmatrix}.$$

This computation extends to vectors of different dimensions, such as:

$$\begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \otimes \begin{pmatrix} a_2 \\ b_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} a_1 a_2 \\ a_1 b_2 \\ a_1 c_3 \\ b_1 a_2 \\ b_1 b_2 \\ b_1 c_3 \end{pmatrix}.$$

Similarly, we can compute tensor products of matrices by applying the same rule. For example, if we have two matrices:

$$\begin{aligned} \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \otimes \begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \end{pmatrix} &= \begin{pmatrix} a_1 \begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \end{pmatrix} & a_2 \begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \end{pmatrix} \\ a_3 \begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \end{pmatrix} & a_4 \begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \end{pmatrix} \end{pmatrix} \\ &= \begin{pmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 & a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_1 b_4 & a_1 b_5 & a_1 b_6 & a_2 b_4 & a_2 b_5 & a_2 b_6 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 & a_4 b_1 & a_4 b_2 & a_4 b_3 \\ a_3 b_4 & a_3 b_5 & a_3 b_6 & a_4 b_4 & a_4 b_5 & a_4 b_6 \end{pmatrix}. \end{aligned}$$

This framework can be extended to tensor products of any number of Hilbert spaces, as long as there is a finite number of them. So, for $\mathbb{H}^A \otimes \mathbb{H}^B \sim \mathbb{C}^4$, if we consider tensor products of the ONBs, we obtain:

$$\{|e_a\rangle \otimes |f_b\rangle\}_{a,b\in\{1,2\}} = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\} = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right\}$$

where the rightmost set denotes the standard basis in $\mathbb{C}^4$. Using this basis, we can also derive the tensor product of $|\phi_1\rangle$ and $|\phi_2\rangle$ as follows:

$$\begin{aligned} |\phi_1\rangle \otimes |\phi_2\rangle &= (a_1|0\rangle + b_1|1\rangle) \otimes (a_2|0\rangle + b_2|1\rangle) \\ &= a_1 a_2 |0\rangle \otimes |0\rangle + a_1 b_2 |0\rangle \otimes |1\rangle + b_1 a_2 |1\rangle \otimes |0\rangle + b_1 b_2 |1\rangle \otimes |1\rangle \\ &= a_1 a_2 |00\rangle + a_1 b_2 |01\rangle + b_1 a_2 |10\rangle + b_1 b_2 |11\rangle \\ &= \begin{pmatrix} a_1 a_2 \\ a_1 b_2 \\ b_1 a_2 \\ b_1 b_2 \end{pmatrix}. \end{aligned}$$

Now, let us introduce an additional Hilbert space $\mathbb{H}^C \sim \mathbb{C}^2$ with its ONB

$$\{|g_c\rangle\}_{c=1,2} = \{|0\rangle, |1\rangle\} = \left\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right\}.$$

For the composite space $\mathbb{H}^A \otimes \mathbb{H}^B \otimes \mathbb{H}^C \sim \mathbb{C}^8$, the ONB becomes:

$$\begin{aligned}\{|e_a\rangle \otimes |f_b\rangle \otimes |g_c\rangle\} &= \{|000\rangle, |001\rangle, |010\rangle, |011\rangle|100\rangle, |101\rangle, |110\rangle, |111\rangle\} \\ &= \left\{\begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix}\right\}\end{aligned}$$

where the last set now denotes the standard basis in $\mathbb{C}^8$.
If we now introduce another quantum state $|\phi_3\rangle$ with coefficients a_3 and b_3:

$$|\phi_3\rangle = a_3|0\rangle + b_3|1\rangle = \begin{pmatrix}a_3\\b_3\end{pmatrix}, \tag{2.25}$$

then, the tensor product of $|\phi_1\rangle$, $|\phi_2\rangle$, and $|\phi_3\rangle$ yields:

$$|\phi_1\rangle \otimes |\phi_2\rangle \otimes |\phi_3\rangle = \cdots = \begin{pmatrix}a_1a_2a_3\\a_1a_2b_3\\a_1b_2a_3\\a_1b_2b_3\\b_1a_2a_3\\b_1a_2b_3\\b_1b_2a_3\\b_1b_2b_3\end{pmatrix}.$$

3 Quantum Computing

3.1 Intro to Quantum Computation

Today's computers rely on traditional physics. They are constrained by operating locally and by the fact that classical bits can only be in one state at a time. However, the landscape of modern quantum physics paints a distinct picture, where quantum systems can concurrently exist in multiple states, yielding interference phenomena during their evolution. Notably, spatially separated quantum systems can become *entangled*, enabling operations to exert *non-local* influences as well.
Quantum computation explores the computational potential driven by quantum principles, merging the foundations of quantum mechanics with computer science. Quantum algorithms, outpacing classical counterparts, are a key focus. The inception of quantum computation traces back to the early 1980s, marked by analog quantum computer ideas and the definition of the universal quantum Turing machine. Initial years witnessed sporadic advancements until Peter Shor's groundbreaking discovery in 1994 of efficient quantum algorithms for factorization and discrete logarithm problems. This shift, influenced by Simon's work, captured immense interest by potentially undermining classical cryptography.
Studying quantum computation is crucial because although current chip miniaturization has made classical computers extremely powerful, at microscopic levels, quantum effects emerge. If correctly harnessed rather then ignored, these quantum effects can speed up computations significantly and even allow us to perform tasks impossible for classical computers.
In the late 1990s, initial quantum computing progress saw the creation of a 2-qubit quantum computer, followed by a 5-qubit version in 2001. Despite gradual advancements in various technologies, challenges persist due to qubit decoherence and noise. Quantum error-correcting codes and fault-tolerant computing have been proposed as

solutions, though practical implementation remains tough. Despite this, while quantum computing is still an emerging field, some limited applications have already shown promise. Even if large-scale quantum computers don't materialize, the idea of quantum-mechanical computers holds potential in diverse areas beyond just faster computing paradigms.

3.2 Postulates of Quantum Mechanics

In this chapter, we offer a concise and informal introduction to Quantum Computing. We begin by outlining the four fundamental principles that form the basis of quantum mechanics – these are referred to as *postulates*. These postulates serve as the essential axioms upon which our understanding of the quantum world is constructed. They are derived from a combination of observations and theoretical insights. This enumeration of the postulates aids in establishing the overarching framework that Quantum Computing is founded upon. It is important to note that there exist various versions of these postulates.
After presenting the postulates, our discussion shifts to a more practical exploration, providing a numerical portrayal of a quantum system.

Postulate 1
*The **state space** of any closed physical system is a complex vector space. At any given point in time, the system is completely described by a **state vector**, which is a **unit vector** in its state space.*

Postulate 2
*The continuous-time evolution of a closed quantum system is described by the **Schrödinger equation**:*

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

*where H is a fixed Hermitian operator known as the **Hamiltonian**.*

By solving this differential equation one gets:

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle \quad \text{where} \quad U(t) = e^{-iHt}$$

and $|\psi(0)\rangle$ is the state at $t = 0$. One can check that $U(t)$ is unitary. It is important to note that the exponentiation we're dealing with here is matrix exponentiation, which doesn't adhere to the conventional rules of traditional exponentiation.

Postulate 3
*A **measurement** with input dimension n, output dimension m, and outcome set S is a collection of $|S|$ matrices of size $m \times n$,*

$$\{M_k : k \in S\} \subset M_{m,n}(\mathbb{C})$$

*known as **measurement operators**, that satisfy the completeness relation*

$$\sum_{k \in S} M_k^\dagger M_k = I_n, \tag{3.1}$$

where I_n is the $n \times n$ Identity matrix.
If the system is in state $|\psi\rangle \in \mathbb{C}^n$ before the measurement, the probability of outcome $k \in S$ and the corresponding post-measurement state $|\psi_k\rangle \in \mathbb{C}^m$ is

$$p(k) = \langle\psi|M_k^\dagger M_k|\psi\rangle = \||M_k|\psi\rangle\|^2, \qquad |\psi_k\rangle = \frac{M_k|\psi\rangle}{\sqrt{\langle\psi|M_k^\dagger M_k|\psi\rangle}}.$$

By the completeness relation, probabilities of all outcomes add up to 1: $\sum_{k \in S} p(k) = 1$.
An **orthogonal measurement** is a measurement whose measurement operators are projectors onto an orthonormal basis of the Hilbert Space. In our case, such basis will always be Standard Basis of the vector space (i.e. the Computational Basis of the system).

Postulate 4
The state space of a composite physical system is the tensor product of the state spaces of the individual component physical systems. If one component is in state $|\psi_1\rangle$ and a second component is in state $|\psi_2\rangle$, the state of the combined system is $|\psi_1\rangle \otimes |\psi_2\rangle$.

This postulate serves as the justification for our extensive exploration of Hilbert spaces and tensor products in the preceding chapter.

3.3 Quantum Systems in practice

Consider some physical system that can be in N different, mutually exclusive classical states. We call these states $|0\rangle, |1\rangle, \ldots, |N-1\rangle$ and assume they form an orthonormal basis of the underlying Hilbert space. Roughly, by a "classical" state we mean a state in which the system can be found if we observe it. A pure quantum state (usually just

called state) $|\psi\rangle$ is a **superposition** of classical states, written

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \cdots + \alpha_{N-1}|N-1\rangle.$$

Here α_i is a complex number that is called the amplitude of $|i\rangle$ in $|\psi\rangle$. Note that, as a consequence of the third postulate (3) with $M_k = |k\rangle\langle k|$ for each $k = 0,\ldots,N-1$, the squared amplitudes sum to 1 and thus define a probability distribution over the possible states that the quantum state can assume. Namely

$$|\alpha_0|^2 + |\alpha_1|^2 + \cdots + |\alpha_{N-1}|^2 = 1.$$

Intuitively, a system in quantum state $|\psi\rangle$ is "in all classical states at the same time", each state having a certain amplitude (which we can think of as a probability value). It is in state $|0\rangle$ with amplitude α_0, in state $|1\rangle$ with amplitude α_1, and so on. Mathematically, the states $|0\rangle,|1\rangle,\ldots,|N-1\rangle$ form an orthonormal basis of an N-dimensional Hilbert space (i.e., an N-dimensional vector space equipped with an inner product). A quantum state $|\psi\rangle$ is a vector in this space, usually written as an N-dimensional column vector of its amplitudes:

$$|\psi\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-2} \\ \alpha_{N-1} \end{pmatrix}. \tag{3.2}$$

Such a vector is sometimes called a "ket". Its conjugate transpose is the following row vector, sometimes called a "bra":

$$\langle\psi| = \begin{pmatrix} \overline{\alpha_0} & \overline{\alpha_1} & \cdots & \overline{\alpha_{N-2}} & \overline{\alpha_{N-1}} \end{pmatrix}. \tag{3.3}$$

The reason for this terminology (often called *Dirac notation* after Paul Dirac) is that an inner product $\langle\phi|\psi\rangle$ between two states corresponds to the dot product between a bra and a ket vector ("bracket"): $\langle\phi|\psi\rangle$. We can combine different Hilbert spaces using tensor product: if $|0\rangle,|1\rangle,\ldots,|N-1\rangle$ are an orthonormal basis of space $\mathbb{H}^A$ and $|0\rangle,|1\rangle,\ldots,|M-1\rangle$ are an orthonormal basis of space $\mathbb{H}^B$, then the tensor product space $\mathbb{H} = \mathbb{H}^A \otimes \mathbb{H}^B$ is an NM-dimensional space spanned by the set of states

$$\{|i\rangle \otimes |j\rangle \mid i \in \{0,\ldots,N-1\}, j \in \{0,\ldots,M-1\}\}.$$

An arbitrary state in $\mathbb{H}$ is of the form

$$|\psi\rangle = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \alpha_{ij} |i\rangle \otimes |j\rangle. \tag{3.4}$$

Such a state is called **bipartite**. Similarly we can have tripartite states that "live" in a Hilbert space that is the tensor product of three smaller Hilbert spaces, etc. The decomposition presented in (3.4) is termed **Schmidt Decomposition**.
Starting from the Schmidt Decomposition we can now delve deeper into the characterization of quantum states by introducing the concept of **separability**. In the context of quantum mechanics and specifically composite systems, separability plays a crucial roles in understanding the correlations and interactions between quantum systems.

Definition 3.3.1
*Consider two arbitrary quantum systems A and B, with respective Hilbert spaces $\mathbb{H}^A$ and $\mathbb{H}^B$. The Hilbert space of the composite system is the tensor product $\mathbb{H} = \mathbb{H}^A \otimes \mathbb{H}^B$. If the first system is in state $|\psi\rangle_A$ and the second in state $|\phi\rangle_B$, the state of the composite system is $|\psi\rangle_A \otimes |\phi\rangle_B$. States of the composite system that can be represented in this form are called **separable states**, or **product states**.*
Not all states are separable states (and thus product states).
Fix a basis $\{|i\rangle_A\}_i$ for $\mathbb{H}^A$ and a basis $\{|j\rangle_B\}_j$ for $\mathbb{H}^B$. The most general state in $\mathbb{H}^A \otimes \mathbb{H}^B$ is of the form

$$|\psi\rangle_{AB} = \sum_{i,j} c_{ij} |i\rangle_A \otimes |j\rangle_B.$$

This state is separable if there exist coefficients $\{c_i^A\}_i, \{c_j^B\}_j$ so that $c_{ij} = c_i^A c_j^B$, yielding

$$|\psi\rangle_A = \sum_i c_i^A |i\rangle_A \qquad\qquad |\phi\rangle_B = \sum_j c_j^B |j\rangle_B.$$

*It is inseparable if for any vectors $\{c_i^A\}_i, \{c_j^B\}_j$ at least for one pair of coordinates c_i^A, c_j^B we have $c_{ij} \neq c_i^A c_j^B$. If a state is inseparable, it is called an **entangled state**.*

Entanglement is yet another fundamental concept intrinsic to quantum mechanics. This phenomenon arises as quantum states of multiple particles become intricately interwoven, transcending classical limits. In entanglement, the attributes of one particle instantaneously influence those of another, even when separated by substantial distances, challenging classical causal understanding. When particles are entangled, measuring the state of one particle promptly dictates the state of another, irrespective of physical

separation. This interconnection forms the bedrock of quantum information theory, underpinning technologies like quantum cryptography and quantum computing. In our specific context of quantum optimization, we will gain a more practical understanding of entanglement in the subsequent chapters, particularly during our exploration of Variational Quantum Algorithms and the Variational Quantum Eigensolver. To simplify, by strategically entangling qubits, the optimization procedure gains the ability to consider multiple qubits simultaneously. This enhancement boosts the optimization process, enabling it to efficiently explore potential solutions across multiple qubits concurrently and navigate solution landscapes more effectively.
Having established a foundational understanding of superpositions and entanglement, let us now explore the concrete actions that can be performed on quantum states: measurement and unitary evolution.

3.3.1 Measurement

Although many different kinds of measuremnt are possible (as described by the third postulate), in this thesis the only ones we are going to need are measurements in the computational basis, which are thus the only ones we are going to discuss.
Going back to the third postulate, these measurement are obtained by considering the operators $\{M_k\}_k = \{|k\rangle\langle k|\}_k$, i.e. the orthogonal projectors onto the basis states $\{|k\rangle\}_k$ for $k = 0, \dots, N-1$. Applying such measurement M_k causes the quantum state $|\psi\rangle$ to collapse onto the basis state $|k\rangle$ with probability

$$\begin{aligned}\langle\psi|M_k^\dagger M_k|\psi\rangle &= \sum_{i,j=0}^{N-1} \overline{\alpha_i}\alpha_j\langle i|M_k^\dagger M_k|j\rangle \\ &= \sum_{i,j=0}^{N-1} \overline{\alpha_i}\alpha_j\langle i|k\rangle\langle k|j\rangle \\ &= \sum_{i,j=0}^{N-1} \overline{\alpha_i}\alpha_j\delta_{ik}\delta_{kj} = \overline{\alpha_k}\alpha_k = |\alpha_k|^2.\end{aligned}$$

The corresponding post-measurement state is then given by

$$|\psi_k\rangle = \frac{\alpha_k}{\sqrt{|\alpha_k|^2}}|k\rangle.$$

Thus, the act of observing a quantum state establishes a probability distribution across classical states, determined by the squared of the amplitudes. Namely, the process of measuring a quantum state within the computational basis results in the collapse of the superposition onto the observed state, with the probability dictated by the square of the

corresponding amplitude.
This requirement leads to the condition $\sum_{i=0}^{N-1} |\alpha_i|^2 = 1$, implying that the amplitude vector possesses a (Euclidean) norm of 1. Upon measuring $|\psi\rangle$ and obtaining the outcome j, the original state $|\psi\rangle$ effectively vanishes, leaving behind solely the state $|j\rangle$. In simpler terms, the act of observing $|\psi\rangle$ causes the quantum superposition to "collapse" onto the classical state $|j\rangle$ that was observed. Consequently, any "information" encapsulated within the amplitudes α_i is irretrievably lost post-measurement.

3.3.2 Unitary evolution

In addition to measurements, quantum states can also undergo transformation through operators. Quantum mechanics imposes a specific constraint on the permissible operators – they must be unitary transformations. When a quantum state is transformed by a linear operator, it evolves into a new superposition. Since measuring this new superposition should also give a probability distribution over the basis states, it follows that the squared amplitudes of this superposition must again sum to 1. This requirement ensures that the only transformations allowed are those that preserve the norm of vectors, namely unitary transformations. A notable feature of unitary transformations is their inherent reversibility. Any unitary transformation admits an inverse, allowing the system to be reverted to its original state. This characteristic stands in stark contrast to measurements, which are inherently irreversible processes.

3.4 Qubits

Having acquired a fundamental understanding of quantum mechanics, we are now prepared to explore the domain of quantum computing, where the central focus revolves around **qubits**.
A classical bit serves as the most fundamental unit of information, conveying a choice between binary alternatives typically symbolized as 0 and 1, Yes and No, or True and False. In quantum mechanics, to represent a physical system that can be in two different, mutually exclusive classical states, we use vectors in a two-dimensional Hilbert space. This two-dimensional Hilbert space can be effectively aligned with the complex plane $\mathbb{C}^2$. The most obvious and simple choice for the orthonormal basis is then:

$$\{|0\rangle, |1\rangle\} := \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\},$$

which is also referred to as the **Computational Basis** for the qubit.
As we saw in the previous section, the quantum state of the system can be described by

a superposition involving the states $|0\rangle$ and $|1\rangle$

$$|\psi\rangle = a|0\rangle + b|1\rangle, \quad \text{with} \quad |a|^2 + |b|^2 = 1, \quad a, b \in \mathbb{C}. \tag{3.5}$$

Superpositions are distinctively quantum in nature and lack a counterpart within the realm of classical bits. Classical computing does not encompass these state combinations. Consequently, the storage capacity of a two-dimensional quantum system far surpasses that of a classical bit, offering a significantly expanded scope for information representation.

We can always examine a bit to determine whether it is in the state 0 or 1, but as we saw in the previous section this is not possible in quantum mechanics. We cannot examine a qubit to determine its quantum state, that is, the values of α and β. Instead, quantum mechanics tells us that if we measure a qubit we get either the result 0, with probability $|\alpha|^2$, or the result 1, with probability $|\beta|^2$, where $|\alpha|^2 + |\beta|^2 = 1$. For example, a qubit can be in the state

$$|+\rangle := \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{3.6}$$

which, when measured, gives the result 0 fifty percent ($|\frac{1}{\sqrt{2}}|^2$) of the time, and the result 1 fifty percent of the time. We will return often to this specific quantum state $|+\rangle$.

Since $|a|^2 + |b|^2 = 1$, we may rewrite equation (3.5) as

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right) \tag{3.7}$$

where θ, ϕ, γ are real numbers. The outer factor of $e^{i\gamma}$ has no observable effects, and for that reason we can effectively write

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle. \tag{3.8}$$

The numbers θ and ϕ define a point on the unit three-dimensional sphere. This sphere is often called the **Bloch sphere**; it provides a useful means of visualizing the state of a single qubit, and we will come back to this visualization in future chapters. However, notice that this intuition is limited to single qubit systems, as there is no simple generalization of the Bloch sphere for systems composed of multiple qubits.

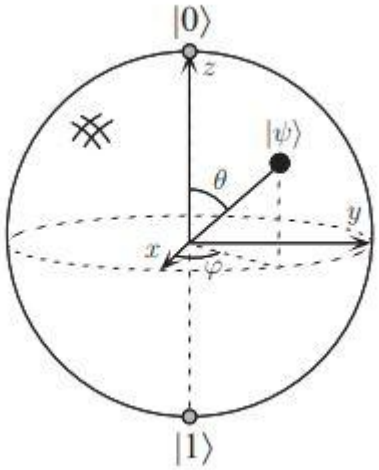


Figure 3.1: Bloch sphere representation of a qubit.

Following the fourth postulate we can easily extend the framework to systems composed of multiple qubits: it suffices to consider the tensor product of the quantum states describing the individual qubits.
Suppose for example we have two qubits. If these were two classical bits, then there would be four possible states, $00, 01, 10$, and 11. Correspondingly, a two qubit system has four computational basis states denoted $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. A pair of qubits can also exist in superpositions of these four states, so the quantum state of two qubits involves associating a complex coefficient – an amplitude – with each computational basis state, such that the state vector describing the two qubits is

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle.$$

Similar to the case for a single qubit, the measurement would bring the supeposition to collapse onto one of the four basis states $00, 01, 10, 11$ with probability $|\alpha_{00}|^2$, $|\alpha_{01}|^2$, $|\alpha_{10}|^2$, $|\alpha_{11}|^2$ respectively. The normalization condition again imposes that

$$\sum_{i,j\in\{0,1\}} |\alpha_{ij}|^2 = 1.$$

For a two qubit system, we could also measure just a subset of the qubits. For instance measuring the first qubit alone gives 0 with probability $|\alpha_{00}|^2 + |\alpha_{01}|^2$, leaving the post-measurement state

$$|\psi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}.$$

Notice that such measurement corresponds to consider the projector $M = |0\rangle\langle 0| \otimes I$ in the third postulate. The first term $|0\rangle\langle 0|$ acts on the first qubit as projection on the state $|0\rangle$; the second term I acts on the second qubit as the Identity, leaving it unchanged. Writing it out explicitly

$$M = |0\rangle\langle 0| \otimes I = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

This framework immediately generalizes to systems with any finite number of qubits. More generally, we may consider a system of n qubits. The computational basis states of this system are of the form

$$|x_1 x_2 \cdots x_n\rangle = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle, \quad \text{with} \quad |x_i\rangle \in \{|0\rangle, |1\rangle\} = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\} \ \forall i.$$

Therefore, a quantum state for such a system is determined by 2^n amplitudes. For $n = 500$, this count exceeds the estimated atoms in the Universe. Storing these complex numbers is unfeasible on any classical computer. The extent of Hilbert space is indeed immense. Yet, in reality, Nature adeptly processes such voluminous data. This potential computational power is what we aim to utilize, but how can we regard quantum mechanics as a computational framework? We explore this in the next section.

3.4.1 2-qubits Entangled States

We wrap up this section by examining an instance of a 2-qubits entangled state. This will provide us with a tangible illustration of the concepts discussed earlier. An example of highly recognized and significant entangled states for two qubits is the collection known as the **Bell states**

$$|\Phi^{\pm}\rangle = \frac{1}{\sqrt{2}}|00\rangle \pm \frac{1}{\sqrt{2}}|11\rangle$$
$$|\Psi^{\pm}\rangle = \frac{1}{\sqrt{2}}|01\rangle \pm \frac{1}{\sqrt{2}}|10\rangle.$$

It can easily be shown that such states are not separable, thus they are entangled states. In the upcoming section on quantum circuits, we are going to examine particular unitary transformations that facilitate the entanglement of multiple qubits within a quantum system.

3.5 Quantum Circuits

Changes occurring to a quantum state can be described using the language of quantum computation. Just as a classical computer is constructed with electrical circuits comprising wires and logic gates, a quantum computer is assembled using a quantum circuit composed of wires and fundamental quantum gates. These gates facilitate the transport and manipulation of quantum information. In this section, we outline a range of basic quantum gates and provide illustrative examples of circuits where they are utilized.

3.5.1 Single qubit gates

Quantum gates on a single qubit correspond to 2×2 matrices, mirroring the qubit's two-dimensional vector representation. However, are there limitations on the matrices allowable as quantum gates? Indeed, there are. The normalization requirement mandates that squared amplitudes sum to 1 for any quantum state, a condition which must be preserved after gate application. This leads to a crucial constraint: gate transformations must conserve Euclidean vector norms. Remarkably, the matrix describing a single-qubit gate, denoted as U, must adhere to unitarity: $U^\dagger U = I$, where $U^\dagger$ is the adjoint of U (achieved via transpose and complex conjugation) and I represents the 2×2 identity matrix. Strikingly, unitarity stands as the sole constraint on quantum gate; any unitary matrix serves as a valid quantum gate.
We now list some elementary gates providing for each of them the matrix representation, a description of the underlying transformation and also the graphical circuit representation.

- The most basic example of gate is the **Identity gate**, which basically acts as the Identity operator, preserving the state of the qubit.

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \text{—}\boxed{I}\text{—}$$

- The **Bitflip gate** X (a.k.a. **NOT-gate**) negates the bit in the computational basis, i.e., it swaps $|0\rangle$ and $|1\rangle$.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \text{—}\boxed{X}\text{—}$$

- The **Phaseflip gate** Z puts a minus in front of $|1\rangle$, while leaving $|0\rangle$ unchanged.

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad \text{—}\boxed{Z}\text{—}$$

- The **Pauli-Y gate** Y is a single-qubit rotation through π radians around the Y-axis of the Bloch Sphere.

$$Y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix} \qquad \text{—}\boxed{Y}\text{—}$$

  Notice that $Y = iXZ = iZX$, so it can be thought of as both a bit flip and a phase flip.

- The **Phase gate** R_ϕ rotates the phase of the $|1\rangle$ state by an angle ϕ, while leaving $|0\rangle$ unchanged.

$$R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \qquad \text{—}\boxed{R_\phi}\text{—}$$

  Note that Z is a special case of this: $Z = R_\pi$. The $R_{\pi/4}$ gate is often just called the T gate.

- Possibly the most important 1-qubit gate is the **Hadamard** gate

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \text{—}\boxed{H}\text{—}$$

  Applying the Hadamard gate H to the initial state $|0\rangle$ and then measuring yields an equal chance of observing either $|0\rangle$ or $|1\rangle$. Similarly, applying H to $|1\rangle$ and measuring results in an equal chance of obtaining $|0\rangle$ or $|1\rangle$. This property significantly aids quantum optimization: applying Hadamard to the $|0\rangle$ state effectively places it into a **uniform superposition**. We will delve deeper into the implications of this phenomenon as we explore Variational Quantum Algorithms in the upcoming chapters.

The gates X, Z, Y are recognized as **Pauli gates**, playing a foundational role in quantum optimization, as we will soon discover. Occasionally, we'll denote these matrices as $\sigma^X, \sigma^Z, \sigma^Y$ respectively.

3.5.2 Multiple qubit gates

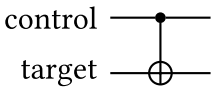
The prototypical multi-qubit quantum logic gates are the controlled gates. Controlled gates are essential components of quantum computing that allow for conditional operations based on the states of two qubits: the control qubit and the target qubit.
In a controlled gate, the control qubit serves as the decision-maker. Its state determines

whether the gate operation is applied to the target qubit or not. When the control qubit is in a particular state (usually $|1\rangle$), the gate acts on the target qubit. However, when the control qubit is in a different state (typically $|0\rangle$), the gate has no effect on the target qubit.
These gates introduce a layer of programmable logic into quantum circuits, enabling qubits to communicate and cooperate based on specific conditions. Controlled gates play a foundational role in quantum algorithms by allowing for selective entanglement, quantum error correction, and a range of other operations crucial for harnessing quantum power.
In a 2-qubit system, the most notable type of controlled gate is the **Controlled-Not gate** CNOT, which swaps the state of the second qubit if the first qubit is in state $|1\rangle$, and does nothing otherwise. Writing it out explicitly

$$\begin{aligned}\text{CNOT} &= |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\end{aligned}$$

and we have

$$\begin{aligned}&\text{CNOT}(|0\rangle|0\rangle) = |0\rangle|0\rangle \\ &\text{CNOT}(|0\rangle|1\rangle) = |0\rangle|1\rangle \\ &\text{CNOT}(|1\rangle|0\rangle) = |1\rangle|1\rangle \\ &\text{CNOT}(|1\rangle|1\rangle) = |1\rangle|0\rangle.\end{aligned}$$

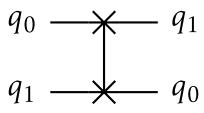


However, controlled gates possess broader applicability. Instead of the Pauli X gate, any unitary transformation can be employed. In fact, as we will see, for the Variational Quantum Eigensolver we use parameterized unitaries in lieu of the X gate.
Another 2-qubit gate that proves highly valuable in various applications is the **Swap gate**. This gate exchanges the states of the two qubits engaged in the operation.
When we consider the orthonormal computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, the action of this gate can be described as follows:

$$
\begin{aligned}
\text{SWAP}(|0\rangle|0\rangle) &= |0\rangle|0\rangle \\
\text{SWAP}(|0\rangle|1\rangle) &= |1\rangle|0\rangle \\
\text{SWAP}(|1\rangle|0\rangle) &= |0\rangle|1\rangle \\
\text{SWAP}(|1\rangle|1\rangle) &= |1\rangle|1\rangle,
\end{aligned}
$$

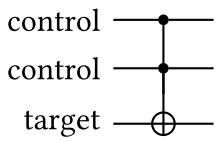
which brings us to the matrix representation

$$
\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.
$$



Naturally, these concepts extend to multi-qubit systems as well.
Consider for example a scenario with 3 qubits, where the goal is to exchange the state of the third qubit only when the first two qubits are both in the state $|1\rangle$. The corresponding matrix representation is as follows:

$$
\begin{aligned}
\text{CCNOT} = {} & |0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I + \\
& + |0\rangle\langle 0| \otimes |1\rangle\langle 1| \otimes I + \\
& + |1\rangle\langle 1| \otimes |0\rangle\langle 0| \otimes I + \\
& + |1\rangle\langle 1| \otimes |1\rangle\langle 1| \otimes X.
\end{aligned}
$$



Performing all the necessary calculations, we obtain

$$
\text{CCNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.
$$

Actually, this gate holds substantial significance within quantum computing, and it is known as the **Quantum Toffoli gate**. It serves as the quantum counterpart to the classical Toffoli gate, bridging the two realms. In classical logic circuits, the Toffoli gate, or CCNOT gate, acts as a universal reversible logic gate. This implies that any classical reversible circuit can be constructed using Toffoli gates. The quantum Toffoli gate, in

turn, enables the simulation of reversible classical logic gates on quantum computers. This ensures that quantum computers possess the capability to execute any computation achievable by classical (deterministic) computers.

3.5.3 Universality of various sets of elementary gates

Which set of elementary gates should we allow? There are several reasonable choices.

- The set of all 1-qubit operations together with the 2-qubit CNOT gate is universal, meaning that any other unitary transformation can be built from these gates.

Allowing all 1-qubit gates is not very realistic from an implementational point of view, as there are continuously many of them, and we cannot expect experimentalists to implement gates to infinite precision. However, the model is usually restricted, only allowing a small finite set of 1-qubit gates from which all other 1-qubit gates can be efficiently approximated.

- The set consisting of CNOT, Hadamard, and the phase-gate $T = R_{\pi/4}$ is universal in the sense of approximation, meaning that any other unitary can be arbitrarily well approximated using circuits of only these gates. The *Solovay-Kitaev Theorem* says that this approximation is quite efficient.

It is often convenient to restrict to real numbers and use an even smaller set of gates:

- The set of Hadamard and Toffoli (CCNOT) is universal for all unitaries with real entries in the sense of approximation, meaning that any unitary with only real entries can be arbitrarily well approximated using circuits of only these gates.

The primary references for this chapter are [2], [3].

4

Quantum Integer Programming: Problem Formulation

In this chapter, we finally delve into Quantum Optimization, a cornerstone of practical quantum computing applications. We initiate our exploration by recalling basis concepts of combinatorial optimization, its significance in real-world scenarios, and the associated challenges. Subsequently, we shift our focus to Quantum Unconstrained Binary Optimization (QUBO) and Ising formulations, pivotal mathematical constructs within the realm of quantum optimization.
These formulations provide the groundwork for Adiabatic Quantum Computation and Variational Quantum Algorithms, both of which hold paramount importance in the realm of quantum optimization. Adiabatic Quantum Computation capitalizes on quantum annealing to approach optimization problems, while Variational Quantum Algorithms utilize parameterized quantum circuits to enhance optimization processes.
The QUBO and Ising formulations discussed in this chapter will lead us towards the deployment of quantum algorithms to address the complex Vehicle Routing Problem, as elaborated in the next chapters of this thesis.

4.1 Combinatorial Optimization

In this section, we delve into an exploration of Combinatorial Optimization, a fundamental area within the broader field of mathematical optimization. Combinatorial optimization revolves around the selection of the best solution from a finite set of possibilities, typically in situations where exhaustive search methods are impractical due to the combinatorial explosion of possibilities. Let us start by recalling the definition of *Integer Programs*.

Definition 4.1.1
*A **Pure Integer Linear Program** is a problem of the form*

$$\begin{aligned} \min \quad & c^T x \\ \textit{subject to} \quad & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

where the data, usually rational, are the row vector $c = \begin{pmatrix} c_1 & \dots & c_n \end{pmatrix}$, the $m \times n$ matrix $A = (a_{ij})$, and the column vector $b = \begin{pmatrix} b_1 & \dots & b_m \end{pmatrix}^T$. The column vector $x = \begin{pmatrix} x_1 & \dots & x_n \end{pmatrix}^T$ contains the variables to be optimized.
*If some of the variables x_i's are let to be continuous variables, then we have a **Mixed Integer Linear program**.*

The condition $x \in \mathbb{Z}^n$ is commonly denoted as the **Integrality Constraint**. If the variables x_i are confined to assuming values solely from a finite set of integers, then we are within the realm of **Combinatorial Optimization**.

Definition 4.1.2
A Combinatorial Optimization Problem is a problem of the form

$$\begin{aligned} \min \quad & c^T x \\ \textit{subject to} \quad & Ax \leq b \\ & l \leq x \leq u \\ & x, l, u \in \mathbb{Z}^n \end{aligned}$$

where the data, usually rational, are the row vector $c = \begin{pmatrix} c_1 & \dots & c_n \end{pmatrix}$, the $m \times n$ matrix $A = (a_{ij})$, the column vector $b = \begin{pmatrix} b_1 & \dots & b_m \end{pmatrix}^T$ and the column vectors of lower and upper bounds $l = \begin{pmatrix} l_1 & \dots & l_n \end{pmatrix}^T$, $u = \begin{pmatrix} u_1 & \dots & u_n \end{pmatrix}^T$. The column vector $x = \begin{pmatrix} x_1 & \dots & x_n \end{pmatrix}^T$ contains the variables to be optimized.

While a typical Integer Program involves an objective function which is linear in x, our focus extends to encompass objective functions that exhibit quadratic dependence on x as well.

4.1.1 CO Problems in Industry

Combinatorial optimization problems hold significant relevance across a multitude of industries, playing a pivotal role in addressing real-world challenges and enhancing operational efficiency. These problems arise in various domains, ranging from logistics and transportation to manufacturing, telecommunications, and beyond. For instance, in the realm of logistics, the Vehicle Routing Problem (VRP) seeks to determine the most efficient routes for a fleet of vehicles to deliver goods to a set of locations while minimizing costs and adhering to various constraints. Similarly, in manufacturing, production scheduling involves optimizing the allocation of resources and scheduling tasks to ensure streamlined operations and minimal downtime. Telecommunications networks necessitate optimizing the routing of data packets to reduce congestion and latency, while portfolio optimization in finance aims to allocate resources across a range of investment options to maximize returns while managing risk. In the field of energy, power distribution networks require optimal allocation of resources to meet demand and minimize losses.
The complexity of these problems, coupled with their pervasive presence in industry, underscores the practical significance of combinatorial optimization as a tool for informed decision-making and resource allocation, driving cost savings, improved resource utilization, and overall operational efficiency.

4.1.2 CO Problems Solution Strategies Overview

Despite their paramount significance, unfortunately, combinatorial optimization problems often belong to the NP (nondeterministic polynomial time) complexity class. This classification signifies that while verifying the correctness of a proposed solution can be done efficiently, discovering the optimal solution within a reasonable timeframe remains a challenge. These problems frequently encompass a very large solution space and can escalate in complexity as the problem's dimensions expand. As a result, achieving optimal solutions for combinatorial optimization problems is deemed computationally demanding, often necessitating specialized algorithms or approximative strategies. Prominent among exact methods are techniques like **Branch and Bound**, **Branch and Cut**, **Branch and Cut and Price**, **Dynamic Programming**, and specialized graph algorithms. However, these methods are best suited for smaller problems where exact solutions are crucial. As problems grow larger, the exponential increase in solution possibilities quickly makes even the most efficient exact methods computationally impractical. This prompts exploration of approximate approaches that strike a balance between computational efficiency and solution quality.
In the realm of approximation strategies, we find practical tools like **greedy algorithms**, **local search methods**, **metaheuristics** and **heuristics**. While these approaches don't

guarantee optimal solutions, they effectively navigate the complex landscape of solution options. Their choices might not lead to the absolute best solution, but they consistently produce solutions of respectable quality. While optimality is not guaranteed, these methods offer a clear computational advantage when tackling complex and large-scale combinatorial problems.
Heuristics are typically tailored to specific problem instances, utilizing domain-specific knowledge and insights to guide their search for good solutions. On the other hand, metaheuristics represent a more general class of algorithms that provide a framework for solving a wide range of combinatorial problems, often incorporating generic search and optimization strategies. Some of the most famous and widely used metaheuristics include:

- **Simulated Annealing (SA):** Borrowing concepts from the annealing process in metallurgy, SA probabilistically accepts worse solutions initially and gradually reduces the likelihood of accepting them as the search progresses. The most commonly used acceptance criterion is the *Metropolis Acceptance Criterion*, though more sophisticated variations exist.

- **Tabu Search:** Tabu search is a type of *Local Search* algorithm that incorporates a short-term memory mechanism. This memory serves the purpose of preventing the algorithm from revisiting previously explored solutions and helps it navigate away from local optima.

- **Iterated Local Search (ILS):** ILS combines local search techniques with perturbation strategies, repeatedly and iteratively exploring neighborhoods of solutions.

- **Variable Neighborhood Search (VNS):** VNS systematically explores different neighborhoods of solutions, changing the neighborhood structure during the search process.

- **Genetic Algorithms (GA):** Inspired by biological evolution, GAs employ mechanisms such as *selection, crossover, and mutation* to iteratively explore the solution space. When the mutation phase incorporates *Local Search* techniques, it transforms into what is known as a **Memetic Algorithm**.

- **Ant Colony Optimization (ACO):** ACO is inspired by the foraging behavior of ants and involves simulating the behavior of ant colonies to find optimal paths.

- **Particle Swarm Optimization (PSO):** PSO models the collective behavior of individuals (particles) in a swarm to optimize a problem by iteratively adjusting their positions.

Local Search algorithms are often an integral part of many metaheuristic methods, as they provide the *exploitation* phase necessary for improving solutions. Striking a balance between the *exploitation* and *exploration* phases is essential in achieving effective optimization. Together, these approaches offer powerful tools for solving complex combinatorial optimization problems, even if they cannot guarantee global optimality. An excellent resource about metaheuristics is [4].

4.1.3 Binary Encoding of CO Problems

Before proceeding, it is worth noting that every combinatorial optimization problem can be transformed into an optimization problem where all variables are binary. This can be accomplished by representing each variable $x_i \in [l_i, u_i] \cap \mathbb{Z}$ as a combination of binary variables. The two primary encoding methods for achieving this transformation are as follows:

- **Unary encoding**: We express x_i as the sum of binary variables, each multiplied by 1. This is formulated as:

$$x_i = l_i + \sum_{j=1}^{u_i - l_i} x_{ij}, \tag{4.1}$$

  where $x_{ij} \in \{0, 1\}$ for each i, j.

- **Logarithmic encoding**: We express the integer variable x_i using a collection of binary variables. Each of these binary variables is associated with a specific power of 2 as in the binary representation of x_i. The precise number of binary variables needed is computed as $V = \lceil \log_2(u_i - l_i + 1) \rceil$. The expression for x_i becomes:

$$x_i = l_i + \sum_{j=0}^{V-2} 2^j x_{ij} + \left(u_i - l_i - \sum_{j=0}^{V-2} 2^j \right) x_{i,V-1} \tag{4.2}$$

  with $x_{ij} \in \{0, 1\}$ for every i, j.

Despite demanding a more careful management of variables, logarithmic encoding shines by drastically reducing the number of variables needed for the binary transformation of a problem. This reduction is particularly crucial in our context, given the limitations of current quantum hardware. With the computational capacity of quantum computers being limited, logarithmic encoding emerges as the sole practical choice. Other encoding methods would lead to an overwhelming number of variables, exceeding the capabilities of present-day quantum computing setups. Hence, within the realm of quantum optimization, logarithmic encoding will be our go-to strategy.

4.2 QUBO and Ising Formulations

In the domain of quantum optimization, the Quadratic Unconstrained Binary Optimization (QUBO) framework emerges as a pivotal link between combinatorial optimization and the realm of quantum computation. QUBO provides a rigorous approach to restructure combinatorial optimization problems into a mathematical framework that harmonizes with the inherent properties of quantum systems. In QUBO, the optimization problem is reformulated to include binary variables that exclusively assume values of 0 or 1. However, QUBO extends beyond mere binary variable representation; it systematically incorporates constraints into the optimization process by assimilating them into the cost function. This integration is akin to a well-known technique in Operations Research termed **penalty methods**. In essence, constraints are transmuted into additive terms within the objective function, accompanied by penalty factors that serve to scale the importance of adhering to these constraints.
In our context, QUBO's strength becomes evident especially in its direct association with Ising models stemming from quantum physics and statistical mechanics. This mapping of QUBO instances to Ising models establishes a connection between optimization and quantum mechanics, allowing for the use of quantum computing to address complex optimization problems.

4.2.1 QUBO Formulation

The typical structure of a QUBO problem takes the form:

$$\min_{x\in\{0,1\}^n} x^T Q x. \tag{4.3}$$

Here, the real matrix Q is of size $n \times n$, generally symmetric or upper triangular. But how does this form emerge? Let us start with a Combinatorial Optimization problem in the shape of:

$$\begin{aligned} \min \quad & f(x) && (4.4)\\ \text{subject to} \quad & Ax \leq b && (4.5)\\ & l \leq x \leq u && (4.6)\\ & x, l, u \in \mathbb{Z}^n. && (4.7) \end{aligned}$$

Here, the objective function f remains linear or, at most, quadratic in x. The transition to the QUBO form involves a few steps:

1. Introduction of **integer** slack variables to convert inequality constraints (4.5) into equalities. This resembles the process in linear programming that shifts a problem

into standard form. The inequality $Ax \leq b$ transforms into $A'x' = b$, where A' and x' encompass the set of slack variables as well.

2. Each integer variable, including slack variables, is encoded as a combination of binary variables (possibly using logarithmic encoding). This maps $f(x) \rightarrow f''(x'')$ and $A'x' = b \rightarrow A''x'' = b$, where now x'' consists of only binary variables $x''_i \in \{0,1\}\ \forall i$.

3. The constraint $A''x'' = b$ is discarded and absorbed into the cost function, resulting in a modified objective $\hat{f}(x'') = f''(x'') + \lambda \|A''x'' - b\|^2$, with λ acting as the penalty factor, penalizing solutions that do not comply with the constraints. This produces a final **unconstrained** optimization problem.

4. By appropriately grouping and combining the terms in the modified objective function, we can deduce the QUBO matrix Q. Each entry Q_{ij} represents the coefficient of the interaction between binary variables x''_i and x''_j in the QUBO problem.

Regarding the third step, of course more general penalization techniques can be employed, where each squared constraint is assigned a distinct penalty based on its importance. However, determining the penalty factors is an open research question, and in practice, it commonly involves relying on trial and error with various guesses.
While it may not be a primary focus for us, it is noteworthy that the QUBO formulation can also be extended to encompass more general Integer Programs. These programs take the form of:

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & h(x) = 0 \\ & g(x) \leq 0 \\ & x \in \mathbb{Z}^n, \end{aligned} \tag{4.8}$$

where both the objective and constraint functions are unrestricted, allowing for general functions of x that go beyond linear or quadratic polynomials. This is possible because of the following theorem, which we report without proof.

Theorem 4.2.1
Any pseudo-Boolean function defined as $f : \{0,1\}^n \rightarrow \mathbb{R}$ can be written uniquely as a

sum of multi-linear functions.

$$f(x) = a_0 + \sum_i a_i x_i + \sum_{ij} a_{ij} x_i x_j + \sum_{ijk} a_{ijk} x_i x_j x_k + \dots$$

We then define new variables $x_{ij} = x_i x_j$ for all terms of order greater than 2 in the above function, continuously until we are left with just quadratic terms. To account for $x_{ij} = x_i x_j$ we need however to introduce new constraints. We define

$$H(x_i, x_j, x_{ij}) = 3x_{ij} + x_i x_j - 2x_{ji} x_i - 2x_{ij} x_j$$

and we have that $x_{ij} = x_i x_j$ iff $H(x) = 0$. Using this procedure, any problem resembling (4.8) can be transformed into QUBO form by:

1. Converting inequalities into equalities by introducing integer slack variables.

2. Expressing variables in binary form (e.g., using logarithmic encoding, unary encoding, or bounded encoding).

3. Quadratizing the objective and constraints, as outlined in the previous theorem.

4. Removing the constraints and adding them as positive penalty terms directly in the objective function.

While we won't directly apply this result, it underscores QUBO's versatility and power. The greatest challenge remains finding the right penalty factors for scaling constraints. Currently, this is an open question for most problems, and in practice, we often rely on trial and error with random guesses.
For a deeper investigation into QUBO, I recommend consulting [5].

4.2.2 Ising Formulation

Ising models are foundational tools in the field of physics, essential for comprehending phase transitions and critical phenomena in various materials. These models enable the study of spin interactions, shedding light on the behavior of magnetic materials under varying conditions like temperature and external magnetic fields.
The relevance of Ising models extends beyond physics. Their formulation aligns seamlessly with Quantum Unconstrained Binary Optimization (QUBO) problems, allowing their application in mathematical optimization scenarios.
Of particular interest to our discussion is the suitability of Ising models for simulation

on quantum computers. This aspect underscores their significance in bridging theoretical quantum mechanics with practical quantum computing capabilities.
A classical Ising model can be written as a quadratic function of a set of N spins $s_i = \pm 1$:

$$\mathcal{E}(s_1, \ldots, s_N) = -\sum_{i<j} J_{ij} s_i s_j - \sum_{i=1}^{N} h_i s_i. \tag{4.9}$$

In Equation (4.9), the coefficients J_{ij} are denoted as **couplers**, representing the strength of interaction between spins s_i and s_j. On the other hand, the coefficients h_i are referred to as **biases**, reflecting the impact of spin s_i on the system's total energy $\mathcal{E}$. From such Ising Energy $\mathcal{E}$ we can then get an **Ising Hamiltonian** H by replacing each spin variable s_i with the related *Pauli-operator*

$$\sigma_i^z = \underbrace{I \otimes I \otimes \cdots \otimes I}_{i-1 \text{ terms}} \otimes \underbrace{Z}_{i\text{-th term}} \otimes \underbrace{I \otimes I \otimes \cdots \otimes I}_{N-i \text{ terms}}, \tag{4.10}$$

where I is the 2×2 Identity matrix and Z is the 2×2 Pauli-Z operator.
In the following, to ease the notation we will also write $\sigma_i^z = I^{\otimes i-1} \otimes Z \otimes I^{\otimes N-i}$.
Since σ_i^z is tensor product of N 2×2 matrices, it follows that each σ_i^z is a $2^N \times 2^N$ square matrix. The final **Problem Hamiltonian** is then

$$H_p = \mathcal{E}(\sigma_1^z, \ldots, \sigma_N^z) \tag{4.11}$$

which is thus a square matrix of order 2^N as well. Observe that due to the diagonal nature of each σ_i^z in the computational basis (attributed to the diagonal property of I, Z in this basis), it can be deduced that H_p also assumes a diagonal form in the computational basis. Bear in mind this insight, as it will enhance our comprehension of the functioning of quantum optimization algorithms in the upcoming chapter.
You can find numerous Ising formulations of widely recognized combinatorial optimization problems in the reference [6].

4.2.3 From QUBO to Ising

Consider a QUBO problem

$$\min_{x\in\{0,1\}^n} \quad x^t Q x = \sum_{ij} Q_{ij} x_i x_j$$

where Q can be assumed to be a symmetric or upper triangular matrix.
With a change of variable $x_i = \frac{1-s_i}{2}$, $s_i \in \{-1, 1\}$ we get the *Ising Energy* as in (4.9)

$$\mathcal{E}(s_1, \cdots, s_n) = -\sum_{ij} J_{ij} s_i s_j - \sum_{1}^{n} h_i s_i.$$

Substituting each s_i with the related Pauli-Z operator σ_i^z as in (4.10) we get the *Problem Ising Hamiltonian* H as in (4.11). Now arises the question: if we aim to minimize the Ising Energy function $\mathcal{E}$ (equivalent to minimizing the QUBO energy), why do we consider the Ising Hamiltonian H which is a square matrix of order 2^n? The rationale behind this choice stems from the subsequent observation.
Given a quantum state $|z\rangle = |z_1\rangle \otimes \cdots \otimes |z_n\rangle$ with $|z_i\rangle \in \{|0\rangle, |1\rangle\}$ for every i, we find

$$\begin{aligned}
H|z\rangle &= (-1)\left(\sum_i h_i \sigma_i^z + \sum_{ij} J_{ij}\sigma_i^z\sigma_j^z\right)|z_1\rangle \otimes \cdots \otimes |z_n\rangle \\
&= -\sum_i h_i |z_1\rangle \otimes \cdots \otimes \underbrace{(-1)^{z_i}|z_i\rangle}_{Z|z_i\rangle} \otimes \cdots \otimes |z_n\rangle + \\
&\quad -\sum_{ij} J_{ij}|z_1\rangle \otimes \cdots \otimes \underbrace{(-1)^{z_i}|z_i\rangle}_{Z|z_i\rangle} \otimes \cdots \otimes \underbrace{(-1)^{z_j}|z_j\rangle}_{Z|z_j\rangle} \otimes \cdots \otimes |z_n\rangle \\
&= -\left(\sum_i h_i (-1)^{z_i} + \sum_{ij} J_{ij}(-1)^{z_i}(-1)^{z_j}\right)|z_1\rangle \otimes \cdots \otimes |z_n\rangle
\end{aligned}$$

Thus

$$H|z\rangle = -\left(\sum_i h_i (-1)^{z_i} + \sum_{ij} J_{ij}(-1)^{z_i}(-1)^{z_j}\right)|z\rangle$$

where $z_i \in \{0, 1\}$ for $i = 1, \cdots, n$.
Defining $s := (s_1, s_2, \ldots, s_n)$ with $s_i := (-1)^{z_i} \in \{-1, +1\}$ we find

$$H|z\rangle = \mathcal{E}(s)|z\rangle,$$

where $\mathcal{E}$ is the Ising Energy defined in (4.9).
In the following, to ease the notation we use $s = s(z) := (-1)^z := \begin{bmatrix}(-1)^{z_1} & \dots & (-1)^{z_n}\end{bmatrix}$.
Thus, there is a correspondence between the energy function $\mathcal{E}$ and the eigen-structure of H. In particular

- $\min \mathcal{E}$ = smallest eigenvalue of H;
- $\arg\min \mathcal{E}$ = ground-states of H (i.e. the eigenvectors associated to the lowest eigenvalue).

To conclude this section we propose a numerical example.

4.2.4 Numerical Example

Consider the problem

$$\begin{aligned} &\min f(x) = x_1 x_2 \\ &\text{s.t.} \quad x_1, x_2 \in \{0, 1\}, \end{aligned}$$

we have

- $\min f = 0$;
- $\arg\min f = (0,0), (0,1), (1,0)$.

With the change of variables $x_i = \frac{1-s_i}{2}$, $s_i \in \{-1, +1\}$ we get

$$\min \mathcal{E} = \frac{1}{4}(1 - s_1 - s_2 + s_1 s_2) \sim \min \mathcal{E} = -s_1 - s_2 + s_1 s_2.$$

Clearly:

- $\min \mathcal{E} = -1$;
- $\arg\min \mathcal{E} = (1,1), (1,-1), (-1,1)$.

Substituting s_i with $\sigma_i^z = I^{\otimes i-1} \otimes Z \otimes I^{\otimes 2-i}$ we get

$$H = -(Z \otimes I) - (I \otimes Z) + (Z \otimes Z) = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

and we have

- $\lambda_{min}(H) = -1$;
- ground-states$(H) = \{e_1 = |00\rangle, e_2 = |01\rangle, e_3 = |10\rangle\}$,

which are exactly the solutions we were looking for.
Recall that

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

4.2.5 Minor Embedding

Mapping the Ising model onto the physical architecture of a quantum device is a critical step in making quantum computers useful for optimization tasks. This process essentially entails finding a way to match the virtual qubits representing spins in the Ising model to the physical qubits available on the quantum hardware. This mapping, known as **Minor Embedding**, boils down to a graph homomorphism problem, which is recognized as a computationally demanding challenge within the realm of NP-hard problems.

Definition 4.2.1
Let $U = (V(U), E(U))$ be a fixed hardware graph. Given $G = (V(G), E(G))$, the minor-embedding of G is defined by the map $\phi : G \to U$ such that:

- *each vertex v_i in $V(G)$ is mapped to a connected subtree T_i of U;*
- *there exists a map $\tau : V(G) \times V(G) \to V(U)$ such that for each arc $(i, j) \in E(G)$, there are corresponding nodes $i_{\tau(i,j)} \in V(T_i)$ and $j_{\tau(j,i)} \in V(T_j)$ such that $(i_{\tau(i,j)}, j_{\tau(j,i)}) \in E(U)$.*

*Given G, if ϕ exists, we say that G is **embeddable** in U.*
*In graph theory, G is called a **minor** of U. We denote the minor-embedding $\phi(G)$ of G by G_{emb}.*

The function τ plays a crucial role in maintaining the relationships among the original nodes within the set $V(G)$. In essence, the presence of τ guarantees that any connection between two nodes in the graph G is preserved as a connection between the corresponding subtrees in U.
Effectively solving this puzzle requires determining how to assign virtual qubits to physical qubits while respecting the connectivity constraints of the quantum device and preserving the inherent structure of the problem. A successful mapping or embedding is indispensable for the efficient execution of Ising-based optimization algorithms on real quantum hardware. The goal of minor embedding is to find an efficient mapping that preserves the logical qubit's operations and interactions while utilizing the available physical qubits. How does this embedding process influence the Ising Model? Suppose G is the variables dependency graph, as depicted by the QUBO matrix Q, and consider the original Ising Energy

$$\mathcal{E} = -\sum_{ij \in E(G)} J_{ij} s_i s_j - \sum_{i \in V(G)} h_i s_i. \tag{4.12}$$

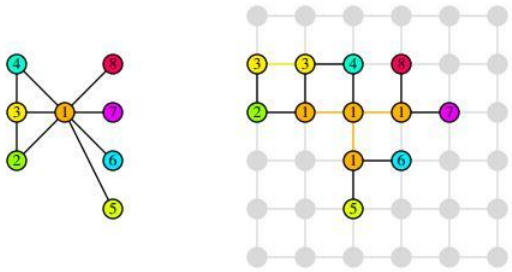


Figure 4.1: G_{emb}(right) is a minor-embedding of G(left) in the square lattice U. Each vertex (called a logical qubit) of G is mapped to a (connected) subtree of (same color/label) vertices (called physical qubits) of U. G is called a (graph) minor of U.

Suppose G_{emb} is a minor-embedding of G, and let $\mathcal{E}_{emb}$ be the embedded energy function associated with G_{emb}:

$$\mathcal{E}_{emb} = -\sum_{ij \in E(G_{emb})} J'_{ij} s_i s_j - \sum_{i \in V(G_{emb})} h'_i s_i. \tag{4.13}$$

Our goal is to find out the requirement for new parameters h's, J's, such that one can solve the original Ising problem on G by solving the embedded Ising problem on its embedding G_{emb}, or equivalently, such that there is an one-one correspondence between the minimum of $\mathcal{E}$ and the minimum of $\mathcal{E}_{emb}$.
Recall that we have the following relationships:

$$V(G_{emb}) = \bigcup_{i \in V(G)} V(T_i) \tag{4.14}$$

$$E(G_{emb}) = \left(\bigcup_{i \in V(G)} E(T_i) \right) \cup \left(\bigcup_{(i,j) \in E(G)} (i_{\tau(i,j)}, j_{\tau(j,i)}) \right). \tag{4.15}$$

In equation (4.15), we can see that the total set of edges after the embedding consists of two parts: the set of edges within each node's subtree and the set of edges connecting these subtrees. We distinguish between these two sets of edges. The latter set, representing the edges in the original graph, is denoted as:

$$OE(G_{emb}) = \bigcup_{(i,j) \in E(G)} (i_{\tau(i,j)}, j_{\tau(j,i)}).$$

For convenience, for each $i_k \in V(T_i)$ we also define

$$Onbr(i_k) := nbr(i_k) \cap OE(G_{emb}),$$

where $nbr(i_k)$ is the set of edges in $E(U)$ originating from node i_k. Essentially, $Onbr(i_k)$ represents the set of edges connected to node $i \in V(G)$ in the embedded graph. Notably

$$(i,j) \in E(G) \iff \exists i_k \in V(T_i): \quad j_{\tau(j,i)} \in Onbr(i_k).$$

The above intuition suggests that we use the same coupler strength for each original edge, namely

$$J_{i_{\tau(i,j)} j_{\tau(j,i)}} = J_{ij} \qquad \forall (i_{\tau(i,j)}, j_{\tau(j,i)}) \in OE(G_{emb}),$$

and we employ ferromagnetic coupler strength, denoted as $F_i^e (< 0)$, for each edge $e \in E(T_i)$. Furthermore, we redistribute the bias h_i of a logical qubit i to its physical qubits within T_i. This redistribution is achieved by selecting h_{i_k} for each physical qubit $i_k \in V(T_i)$ such that:

$$\sum_{i_k \in V(T_i)} h_{i_k} = h_i.$$

This strategy ensures a consistent treatment of coupler strengths for original edges and ferromagnetic interactions within subtrees while properly distributing the bias values across the physical qubits within each subtree. Therefore, we have

$$\mathcal{E}_{emb} = -\sum_{(i,j)\in E(G)} J_{ij} s_{i_{\tau(i,j)}} s_{j_{\tau(j,i)}} - \sum_{i \in V(G)} \left(\sum_{i_k \in V(T_i)} h'_{i_k} s_{i_k} + \sum_{(i_p, i_q) \in E(T_i)} F_i^{pq} s_{i_p} s_{i_q} \right). \tag{4.16}$$

To ensure the algorithm's accuracy and secure the best possible solution, our goal is to equate the embedded energy with the original energy. The central concept is adjusting the qubit biases (h'_{i_k}) and coupler constants (F_i^{pq}) within the embedded energy to mirror the energy landscape of the original function. This alignment is pivotal as it ensures that the optimal solution of the embedded Hamiltonian corresponds to the optimal solution of the original problem.

Determining the optimal qubit biases and coupler constants involves an optimization process, which can be complex due to the noise and imperfections associated with the physical qubits and their interactions. Various strategies are available for addressing this optimization challenge. In our scenario, we won't be directly solving this problem; instead, we will rely on the built-in optimization strategies provided by the libraries we employ, which generally yield satisfactory outcomes. Nonetheless, this succinct subsec-

tion underscores another significant challenge encountered in quantum optimization. More details can be found in the reference [7].

5

Quantum Integer Programming: Quantum Optimization Algorithms

In this chapter, we finally begin an in-depth exploration of quantum optimization. This field resides at the intersection of quantum physics, computational mathematics, and algorithmic methodologies, holding the potential to transform our approach to intricate optimization problems. Central to our investigation is the **Ising Hamiltonian**, a foundational concept that serves as a link between classical and quantum optimization approaches. Our discourse follows two distinct avenues: **Adiabatic Quantum Computation** and **Variational Quantum Algorithms**.
The former capitalizes on the inherent quantum properties of physical systems. The latter merges quantum resources with classical techniques, giving rise to hybrid strategies for optimization.
We commence our discussion with *Adiabatic Quantum Computation*, demonstrating also its applicability in the *Graver Augmentation strategy*. This connection bridges quantum computing with Algebraic Geometry optimization methods. Subsequently, we delve into *Variational Quantum Algorithms*. After introducing the overarching framework, we delve deeper into the nuances of the **Variational Quantum Eigensolver (VQE)** and **Quantum Approximate Optimization Algorithm (QAOA)**.
It is important to recognize that the quantum optimization algorithms discussed here represent just a subset of the broader landscape. Numerous other algorithms exist, including **Quantum Random Walk**-based optimization and **Quantum Evolutionary Algorithms**, among others. While we won't delve into these approaches in this discussion, it is worth noting that Quantum Random Walk-based optimization shares similarities with QAOA in terms of its overall structure. On the other hand, Quantum Evolutionary Algorithms can be seen as quantum counterparts to memetic algorithms, which implies they rely on populations of solutions. However, due to our constrained

resources, we didn't explore these approaches. Our limitations stemmed from the fact that population-based algorithms would have necessitated an excessive number of calls to the quantum hardware.

5.1 Adiabatic Quantum Computation

Adiabatic Quantum Computation (AQC) is quite different from the circuit model of quantum computing. In the circuit model, computation spans the entire Hilbert space through a series of quantum logic gates. AQC, however, follows a distinct approach. It begins with an initial Hamiltonian, often referred to as the **driver**, which possesses a ground state that is easy to prepare, and then it gradually transforms to a final Hamiltonian. At the end of this transformation, the ground state of the final Hamiltonian encodes the solution to the optimization problem.
The **Adiabatic Theorem** assures that if the Hamiltonian evolves *slowly* enough, the system will closely follow the instantaneous ground state. This method exhibits intricate links to condensed matter physics, computational complexity theory, and heuristic algorithms. Adiabatic quantum algorithms designed for optimization tasks often employ **stoquastic** Hamiltonians, characterized by their sole possession of non-positive off-diagonal elements within the computational basis.
However, the introduction of **non-stoquastic** Hamiltonians into adiabatic quantum computation (AQC) leads to an intriguing development. AQC begins to resemble the circuit model of quantum computation. This resemblance implies that non-stoquastic AQC and other universal quantum computation models can simulate each other, and this emulation only requires a polynomial increase in computational resources.
As a result, the term AQC today often includes this broader, non-stoquastic context, extending its scope beyond optimization to cover a wider range of computational scenarios.
In order to establish the foundations of AQC, it is essential to introduce the notion of a ***k*-local** Hamiltonian. For the sake of generality, suppose we are operating with p-state particles. Similar to how a qubit can exist in a superposition of two states concurrently, a particle with p distinct states can similarly inhabit all of these states simultaneously. This essentially means that the particle lives within a Hilbert Space of dimensions equal to p (for instance, a qubit is a 2-state particle).
In this setting, a k-local Hamiltonian H is essentially an hermitian matrix taking the form

$$H = \sum_{i=1}^{r} H_i,$$

with each H_i influencing, in a non-trivial manner, at most k particles. In simpler terms, H_i acts as the Identity Operator on all particles except for a maximum of k particles.

Definition 5.1.1
*A **k-local Adiabatic Quantum Computation** is specified by two k-local Hamiltonians, H_0 and H_1, acting on a number n of p-state particles, $p \geq 2$. The ground state of H_0 is unique and is a product state. The output is a state that is ϵ-close in l_2-norm to the ground state of H_1. We define the **schedule** of the AQC as a time-dependent function $s(t) : [0, t_f] \rightarrow [0, 1]$, with t_f being the smallest time such that the final state of an adiabatic evolution generated by*

$$H(s)\Big|_{s(t)=s(t_f)} = (1-s)H_0 + sH_1\Big|_{s(t)=s(t_f)}$$

is ϵ-close in l_2-norm to the ground state of H_1.

Several noteworthy points should be highlighted:

1. There is no stringent requirement for the ground state of H_1 to be unique. For instance, when H_1 corresponds to the **Problem Hamiltonian** associated to an optimization problem, the existence of multiple ground states does not present an issue, as any of these states can effectively represent a solution to the optimization problem.

2. On occasion, it proves advantageous to explore adiabatic quantum computation within an excited state, namely a state associated with energies that are neither the lowest nor the highest within the Hamiltonian. In other words, an excited state refers to an eigenvector of the Hamiltonian that corresponds to an energy eigenvalue which is neither the lowest (ground state) nor the highest within the system's spectrum of energy eigenvalues.

3. Introducing more general *paths* between H_0 and H_1 can offer valuable insights. This could involve the incorporation of an intermediate Hamiltonian that becomes null at both $s = 0$ and $s = 1$, expanding the range of possible trajectories for the adiabatic evolution.

The computational time t_f in an adiabatic algorithm is bounded by a scaling behavior no worse than $1/\Delta^3$, where Δ represents the gap between the energy of the ground state (lowest eigenvalue) and the energy of the first excited state (second lowest eigenvalue) within the Hamiltonian $H(s)$ that controls the adiabatic evolution. With the provision that the Hamiltonian experiences sufficiently smooth variations, this scaling can be further enhanced to approximately $O(1/\Delta^2)$. Although these conditions offer practical

guidelines, it is important to acknowledge that they necessitate constraining the minimum gap between eigenvalues of a complex many-body Hamiltonian—an inherently challenging task.
Nonetheless, there exist instances where an analysis of the energy gap is feasible. Such examples serve to illustrate the enhanced computational efficiency offered by Adiabatic Quantum Computation (AQC) in comparison to classical computation. This is evident even in the context of adiabatic versions of fundamental algorithms inherent to the circuit model.
However, it is more common to encounter scenarios where either the energy gap analysis does not unveil any computational acceleration compared to classical methods, or where a definitive conclusion about the acceleration remains elusive. In fact, there is limited knowledge regarding adiabatic quantum speedups, especially in the original context of addressing classical combinatorial optimization problems. This particular aspect remains a highly active area of research.
We begin by reviewing the adiabatic theorem, which forms the backbone of AQC. The adiabatic approximation theorem states, roughly, that for a system initially prepared in the ground state $|\epsilon_0(0)\rangle$ (or any other eigenstate $|\epsilon_j(0)\rangle$) of a time-dependent Hamiltonian $H(t)$, the time evolution governed by the Schrödinger equation

$$i\hbar\frac{\partial|\psi(t)\rangle}{\partial t} = H(t)|\psi(t)\rangle \tag{5.1}$$

(we set $\hbar \equiv 1$ from now on) will approximately keep the actual state $|\psi(t)\rangle$ of the system in the corresponding instantaneous ground state $|\epsilon_0(t)\rangle$ (or any other eigenstate $|\epsilon_j(t)\rangle$) of $H(t)$, provided that $H(t)$ varies *sufficiently slowly*. Quantifying the exact nature of this slow variation is the subject of the **Adiabatic Theorem (AT)**, which exists in many variants. Here we just report an approximate version of the Adiabatic Theorem, more rigorours variants can be found in [8].
Let $|\epsilon_j(t)\rangle$ ($j \in \{0,1,2,\dots\}$) denote the instantaneous eigenstate (i.e. eigenvector) of $H(t)$ with energy (i.e. eigenvalue) $\epsilon_j(t)$ such that $\epsilon_j(t) \leq \epsilon_{j+1}(t)\ \forall j,t$, i.e., $H(t)|\epsilon_j(t)\rangle = \epsilon_j(t)|\epsilon_j(t)\rangle$ and $j = 0$ denotes the (possibly degenerate) ground state. Assume that the initial state is prepared in one of the eigenstates $|\epsilon_j(0)\rangle$. Suppose we have $H(s) = A(s)H_0 + B(s)H_1$, where $A(s)$ and $B(s)$ are monotonically decreasing and increasing, respectively. The Schrödinger equation then becomes

$$\frac{1}{t_f}\frac{\partial|\psi_{t_f}(s)\rangle}{\partial s} = -iH(s)|\psi_{t_f}(s)\rangle. \tag{5.2}$$

An adiabatic condition subject to this formulation is given by

$$\frac{1}{t_f}\max_{s\in[0,1]}\frac{|\langle\epsilon_i(s)|\partial_s H(s)|\epsilon_j(s)\rangle|}{|\epsilon_i(s)-\epsilon_j(s)|^2} << 1 \quad \forall j \neq i.$$

This condition gives rise to the widely used criterion that the total adiabatic evolution time should be large on the timescale set by the minimum of the square of the inverse spectral gap $\Delta_{ij}(s) = \epsilon_i(s) - \epsilon_j(s)$. In most cases one is interested in the ground state, so that $\Delta_{ij}(s)$ is replaced by

$$\Delta \equiv \min_{s\in[0,1]} \Delta(s) = \min_{s\in[0,1]} \epsilon_1(s) - \epsilon_0(s). \tag{5.3}$$

Despite being widely used, these arguments are approximate, in the sense that they do not result in strict inequalities and do not result in bounds on the closeness between the actual time-evolved state and the desired eigenstate. More rigorous arguments can be found in the reference [8].

To summarize, Adiabatic Quantum Computation solves optimization problems by gradually changing the system's Hamiltonian. It transitions from an initial Hamiltonian H_0 to a final Hamiltonian H_1 encoding the problem we wish to solve. The algorithm starts in the ground state of H_0 and slowly varies the Hamiltonian over time. The adiabatic theorem guarantees that if the Hamiltonian changes slowly enough, the system stays in its ground state throughout the process. At the end of the evolution we get the ground state of H_1, optimal solution to the optimization problem.

Now we come across a question: how do we go about carefully picking the starting Hamiltonian, H_0, and the ending Hamiltonian, H_1?

Choosing H_1 is quite straightforward: we simply set it as H_p, where H_p is the Problem Hamiltonian defined in (4.11). Indeed, as we discussed earlier, there's a direct link between the ground state of this Hamiltonian and the optimal solution to the original optimization problem (4.2.3).

On the flip side, when it comes to selecting H_0, we have more freedom. We can opt for a general Hamiltonian, **as long as it doesn't commute** with H_p. This need for non-commuting Hamiltonians is crucial in Adiabatic Quantum Computation (AQC). If H_0 and H_1 were to commute, it would imply that they can be simultaneously diagonalized and share a common set of eigenvectors, which would also diagonalize the interpolating Hamiltonian $H(s)$. Now if $H(s)$ is diagonal for every step s, the system, starting from the ground state of H_0, would remain in that state throughout the evolution, without transitioning to the ground state of H_1. Thus, having non-commuting Hamiltonians becomes crucial for a meaningful adiabatic evolution.

However, there's another significant constraint when it comes to picking H_0: for the algorithm to work effectively, Adiabatic Quantum Computation (AQC) requires begin-

ning with the ground state of H_0. Yet, in general, finding this ground state is a challenging task, comparable to solving the original complex optimization problem. This situation leads to the common practice of choosing H_0 as

$$H_0 = \sum_{i=1}^{n} \sigma_i^x = \sum_{i=1}^{n} \underbrace{I \otimes I \otimes \cdots \otimes I}_{i-1 \text{ terms}} \otimes \underbrace{X}_{i\text{-th term}} \otimes \underbrace{I \otimes I \otimes \cdots \otimes I}_{n-i \text{ terms}} = \sum_{i=1}^{n} I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i}, \tag{5.4}$$

where X denotes the Pauli-X operator (i.e. the bit-flip operator). This Hamiltonian, known as the *Transverse Field Hamiltonian*, possesses a well-known ground state, which is remarkably convenient due to its nature as a superposition of all possible basis states within the Hilbert space:

$$|\epsilon_0(0)\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle.$$

Note also that

$$|+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \underbrace{H \otimes H \otimes \cdots \otimes H}_{n \text{ terms}} |\underbrace{00\cdots0}_{n \text{ terms}}\rangle. \tag{5.5}$$

. To recap, selecting H_0 as defined in (5.4) proves advantageous for several reasons:

- It possesses a known ground state, namely $|+\rangle^{\otimes n}$.
- This ground state happens to be a uniform superposition of all possible basis states within the Hilbert space. This property is favorable for optimization since it permits the algorithm to simultaneously explore each potential solution.
- Preparing this ground state is efficient, requiring only Hadamard transformations, which are known for their practical implementation efficiency.

Let us also spend some words on the annealing schedule $s(t)$. Ensuring the delicate balance within the annealing schedule holds utmost significance; excessive rapidity in progression could potentially trigger shifts towards higher-energy states. On the contrary, an excessively gradual evolution could extend computation times beyond practical limits. Crafting a suitable annealing schedule lacks a one-size-fits-all rule. Linear schedules are often preferred due to their simplicity of implementation and their ability to yield reasonably effective outcomes. However, schedules tailored individually to match each problem's characteristics and the specific quantum hardware are also a viable option.

For instance, in the study conducted by the authors in [9], they investigate the evolution of the interpolating Hamiltonian defined as follows:

$$H(t) = H_p + \mathbf{\Gamma}(t)H_0 = -\sum_{ij} J_{ij}\sigma_i^z\sigma_j^z - \sum_i h_i\sigma_i^z - \mathbf{\Gamma}(t)\sum_i \sigma_i^x.$$

Remarkably, they establish intriguing parallels between the *transverse field term* $\mathbf{\Gamma}(t)$ and the *temperature* $T(t)$ in classical Simulated Annealing. They discovered that altering $\mathbf{\Gamma}(t)$ in Quantum Annealing (QA) and $T(t)$ in Simulated Annealing (SA) from infinity to zero with analogous functional forms,

$$\mathbf{\Gamma}(t) = T(t) = c/t,\ c/\sqrt{t},\ c/\ln(t+1) \quad \text{with } t : 0 \to \infty$$

yielded promising outcomes. This adjustment enabled Quantum Annealing (AQC) to converge more swiftly than classical SA for the specified problem instances.
Concluding this section, it is essential to understand what it means to "evolve" a quantum system under a given Hamiltonian $H(t)$. Recall the Schrödinger equation

$$i\hbar\frac{\partial|\psi(t)\rangle}{\partial t} = H(t)|\psi(t)\rangle,$$

solving this equation leads to a unitary time operator $U(t, t_0)$:

$$U(t, t_0) = \tau e^{\frac{-i}{\hbar}\int_{t_0}^{t} H(t')dt'}.$$

This operator allows us to express $|\psi(t)\rangle$ as $|\psi(t)\rangle = U(t, t_0)|\psi(t_0)\rangle$. Calculating the time operator often involves numerical approximation methods, such as the **Trotter-Suzuki Decomposition**. This technique breaks down the time evolution operator into a product of simpler operators by dividing the interval $[t_0, t]$ into smaller disjoint intervals of size δ. The size δ is chosen so that $H(t')$ remains nearly constant within each interval $[t_0 + k\delta,\ t_0 + (k+1)\delta]$. This enables us to factor $H(t')$ out of the integral, leading to the expression:

$$U(t, t_0) = \tau e^{\frac{-i}{\hbar}\sum_k H(t_0+k\delta)\delta}.$$

Applying the **Trotter-Lie formula** yields the approximation:

$$U(t, t_0) \sim \tau \prod_k e^{\frac{-i}{\hbar}H(t_0+k\delta)\delta}.$$

Recalling that $H(t) = (1 - s(t))\,H_0 + s(t)H_p$ and utilizing the Trotter approximation once more, we arrive at the form:

$$U(t, t_0) \sim \tau \prod_k e^{-i\beta_k H_0} e^{-i\gamma_k H_p}, \tag{5.6}$$

for properly defined parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. This equation will become relevant again when we discuss the Quantum Approximate Optimization Algorithm.
For a more comprehensive understanding of AQC, please refer to the following references: [8, 10].

5.1.1 Quantum Graver Augmentation

We now consider a different approach to Integer Programming by means of Algebraic Geometry theory. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a real-valued function. We want to solve the general, possibly non-linear, integer optimization problem:

$$\begin{aligned} &\min f(x) \\ &\text{s.t.} \quad Ax = b \qquad A \in \mathbb{Z}^{m\times n}, b \in \mathbb{Z}^m \\ &\qquad l \le x \le u \qquad\qquad x, l, u \in \mathbb{Z}^n. \end{aligned}$$

One approach to solving such problem is to use an augmentation procedure: start from an initial feasible solution (which itself can be hard to find) and take an improvement step (augmentation) until one reaches the optimal solution. Augmentation procedures such as these need *test sets* or *optimality certificates*.

Definition 5.1.2
*A set $S \in \mathbb{Z}^n$ is called a **test set** or **optimality certificate** if for every nonoptimal but feasible solution x_0 there exists $t \in S$ and $\lambda \in \mathbb{Z}_+$ such that $x_0 + \lambda t$ is feasible and $f(x_0 + \lambda t) \le f(x_0)$. The vector t (or λt) is called the improving or **augmenting direction**.*

Some observations:

- If the optimality certificate is given, any initial feasible solution x_0 can be augmented to the optimal solution in polynomial time.
- If S is finite, one can enumerate over all $t \in S$ and check if it is augmenting (improving).
- If S is not practically finite, or if all elements $t \in S$ are not available in advance, it is still practically enough to find a subset of S that is feasible and augmenting.

The question now is, how do we find test sets for a given Integer Program? Algebraic Geometry offers us two ways:

- **Reduced Gröbner Basis** of the *Toric Ideal* of A:

$$< x^u - x^v \mid u - v \in ker(A) \cap \mathbb{Z}^n_+ >;$$

- **Graver Basis** of the integer kernel of A:

$$\mathcal{L}(A) = \{x \in \mathbb{Z}^n \mid Ax = 0,\, x \neq 0\} = (ker(A) \setminus \{0\}) \cap \mathbb{Z}^n.$$

Before proceeding, it is important to observe that in both scenarios, the process of computing test sets remains completely independent of the objective function f. Instead, it solely revolves around the constraint matrix A, which defines the vector space linked to the feasible region. Indeed, augmentation procedures treat the objective as an oracle. This allows us to extend such methods also to complex non-linear integer programs. Moreover, in augmentation procedures it does not matter from which feasible solution one begins, nor does the sequence of improving steps taken: the final stop is an optimal solution (given that the starting point is a feasible solution).
In the context of our discussion, our attention is directed exclusively towards **Graver test sets**, as they can be effectively computed using Adiabatic Quantum Computation. Consequently, we will not delve into the topic of Gröbner basis. It is noteworthy that Gröbner and Graver bases share a close relationship; for in-depth insights, refer to [11, 12].
Before introducing the definition of the Graver basis, it is essential the following definition.

Definition 5.1.3
*Let $x, y \in \mathbb{R}^n$. We say that x is **conformal** to y, written $x \sqsubseteq y$, when $x_i y_i \geq 0$ (x and y lie on the same orthant) and $|x_i| \leq |y_i|$ for $i = 1, \ldots, n$.*

Definition 5.1.4
The Graver basis of an integer matrix A is defined to be the finite set of $\sqsubseteq$-minimal elements (indecomposable elements) in the lattice $\mathcal{L}(A) = \{x \in \mathbb{Z}^n \mid Ax = 0, x \neq 0\}$. We denote by $\mathcal{G}(A) \subset \mathbb{Z}^n$ the Graver basis of A.

Once found, the Graver basis can be used to augment any feasible solution towards the optimum. **GAMA (Graver Augmented Multi-seed Algorithm)** is an augmentation algorithm which uses the Test set provided by the Graver basis to optimize a given Integer Program. *Multi-seed* refers to the use of multiple feasible solutions as starting points,

each one of them augmented in parallel using the Graver basis. At the end, we select as best solution the one achieving lowest objective function value. More details can be found in the original paper [13].

However, the question arises: how can we effectively compute this basis? Various al-

Algorithm 5.1 Graver Augmented Multi-seed Algorithm (GAMA)

Input: Matrix A, vector b, cost function $f(x)$, bound $[l, u]$.
Output: Global solution(s) x^*.
Initialize: terminated solutions set, $termSols = \{\emptyset\}$.
Using any Graver extraction algorithm input: A, extract $\mathcal{G}(A)$.
Find multiple feasible solutions, satisfying $l \leq x \leq u$.
for any feasible solutions $x = x_0$:
  while $g \in \mathcal{G}(A)$:
    if $l \leq (x+g) \leq u$ and $f(x+g) < f(x)$:
      $x = x + g$
    end if
  end while
  $termSols \leftarrow (x, f(x))$
end for
return $x^* = \{x \mid f(x) = \min_f(termSols)\}$

gorithms exist, with the **Pottier algorithm** being one of the most prominent. Despite several acceleration techniques, the computation of such a basis often remains computationally challenging on classical computers. This is where quantum computing steps in. The idea is to leverage quantum devices to calculate the Graver basis, followed by a classical iteration over its elements to progressively refine a feasible solution towards optimality.

To achieve this objective, we aim to leverage the framework of Adiabatic Quantum Computation (AQC). More specifically, our goal is to reframe the task of identifying the integer kernel of matrix A into a Quadratic Unconstrained Binary Optimization (QUBO) problem. Subsequently, we transform this QUBO into an Ising model and address it using the Adiabatic Quantum Computation approach. The overall procedure unfolds as follows. Consider the problem statement

$$\begin{aligned} &\min\ x^T A^T A x \\ &\text{s.t.}\ \ l \leq x \leq u \\ &\qquad l, x, u \in \mathbb{Z}^n, A \in \mathbb{Z}^{m \times n}; \end{aligned}$$

First of all we need to binarize each x_i. Suppose for instance $x_i = l_i + e_i^T X_i$ where $X_i^T = \begin{pmatrix} X_{i1} & \dots & X_{ik_i} \end{pmatrix}$, $e_i^T = \begin{pmatrix} 2^0 & 2^1 & \dots & 2^{k_i} \end{pmatrix}$, with $X_{ij} \in \{0,1\}\ \forall i,j$. At this point

redefine

$$x = L + EX = \underbrace{\begin{pmatrix} l_1 \\ l_2 \\ l_3 \\ \vdots \\ l_n \end{pmatrix}}_{L:=} + \underbrace{\begin{pmatrix} e_1^T & 0 & 0 & \cdots & 0 \\ 0 & e_2^T & 0 & \cdots & 0 \\ 0 & 0 & e_3^T & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & e_n^T \end{pmatrix}}_{E:=} \underbrace{\begin{pmatrix} \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1k_1} \end{pmatrix}^T \\ \begin{pmatrix} X_{21} & X_{22} & \dots & X_{2k_2} \end{pmatrix}^T \\ \begin{pmatrix} X_{31} & X_{32} & \dots & X_{3k_3} \end{pmatrix}^T \\ \vdots \\ \begin{pmatrix} X_{n1} & X_{n2} & \dots & X_{nk_n} \end{pmatrix}^T \end{pmatrix}}_{X:=} \tag{5.7}$$

where L is an offset introduced to account for the bounds on x.
At this point we have the QUBO problem:

$$\begin{aligned} &\min\ (L+EX)^T A^T A(L+EX) \\ &\text{s.t.}\quad X_i \in \{0,1\}\ \ \forall i. \end{aligned}$$

Now convert to Ising using the usual change of variables $X_i = \frac{1-Z_i}{2}$, $Z_i \in \{-1,1\}$ for each i. The final step involves resolving the Ising formulation using Adiabatic Quantum Computation. Typically, the set of solutions obtained is not inherently $\sqsubseteq$-minimal. To derive the Graver basis, a classical filtering process is applied to render it $\sqsubseteq$-minimal. The resultant set represents the sought-after (partial) Graver basis.
The process outlined above provides a simplified version of the original algorithm as introduced in [14]. In the original publication, the authors incorporate a total of four classical post-processing routines:

1. a post-processing strategy to transform quantum annealer's near-optimal solutions into optimal solutions;
2. a filtering on the set of optimal solutions so to make it $\sqsubseteq$-minimal and extract Graver elements;
3. a further post-processing to turn the unused elements from the Kernel into additional Graver elements;
4. an adaptive binarization that shifts the middle point and the length of the encoding based on suboptimal solutions obtained in the previous iteration.

The initial three routines focus on maximizing the extraction of Graver elements while minimizing the number of calls to the annealer. On the other hand, the adaptive binarization technique is geared towards optimizing the qubit count in the Ising formulation. For more comprehensive insights, reference the source material [14]. What

captivates our attention here is the underlying concept: transforming the problem into a Quadratic Unconstrained Binary Optimization (QUBO) and employing Adiabatic Quantum Optimization.
However, our journey doesn't conclude here. Let us remember that augmentation strategies also necessitate starting feasible solutions as a foundation for augmentation. Yet, in the realm of complex optimization problems, obtaining even a single feasible solution can be as challenging as solving the entire problem itself. Could Adiabatic Quantum Computation be harnessed to yield feasible solutions as well? The answer is affirmative, requiring only a brief adaptation of the aforementioned procedure.
To this end just notice that finding x such that $Ax = b$ is equivalent to minimize the squared norm $||Ax - b||^2 = (Ax - b)^T(Ax - b)$ which, after binary encoding, leads us to

$$\begin{aligned} &\min\ X^T Q_b X \\ &\text{s.t.}\quad Q_b = E^T A^T A E + 2\text{diag}(L^T A^T A - b^T A)E \\ &\qquad X_i \in \{0,1\} \quad \forall i. \end{aligned}$$

Solving this QUBO using a AQC will give us many feasible solutions. Once such points are found, we can augment each point in parallel using our (partial) Graver Basis. These augmentations will lead us to the optimum point with high likelihood.

5.2 Variational Quantum Algorithms

Let us turn our attention to a distinct category of quantum optimization methods known as **Variational Quantum Algorithms (VQAs)** [15]. These algorithms have risen to prominence as the primary approach for achieving quantum advantages on Noisy Intermediate Scale Quantum (NISQ) devices. In contrast to the pure quantum approach of Adiabatic Quantum Computation (AQC), VQAs are hybrid quantum-classical algorithms designed for execution on gate-based quantum models. They employ parameterized quantum circuits, often referred to as **ansatzes**, to operate on quantum computers using gate-based techniques. Subsequently, the optimization of these parameters is delegated to a classical optimizer. Among the most notable VQAs are the **Variational Quantum Eigensolver (VQE)**[16] and the **Quantum Approximate Optimization Algorithm (QAOA)**[17]. While VQE and QAOA exhibit distinct characteristics, they share the overarching framework of VQAs. Here's a step-by-step overview:

1. Start by defining a quantum circuit, denoted as $U(\boldsymbol{\theta})$, which relies on adjustable parameters $\boldsymbol{\theta}$. This circuit's purpose is to prepare an initial quantum state, represented as $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_0\rangle$, with $|\psi_0\rangle$ being a fixed starting quantum state.

2. Select a specific cost function, $C(\boldsymbol{\theta})$, designed to quantify the performance of the quantum circuit. In our context, the chosen cost function takes the form

$C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H_p(\sigma_1^z, \cdots, \sigma_n^z)|\psi(\boldsymbol{\theta})\rangle$.

3. Opt for a *classical* optimization algorithm, which will be responsible for minimizing the cost function by adjusting the parameters $\boldsymbol{\theta}$.

4. Initialize the parameters $\boldsymbol{\theta}$ with an initial estimate and establish a convergence criterion to guide the optimization process.

5. Execute the Variational Quantum Algorithm (VQA) by repetitively applying the quantum circuit $U(\boldsymbol{\theta})$ to generate the state $|\psi(\boldsymbol{\theta})\rangle$. Afterward, perform measurements on the state using a chosen basis and leverage the measurement outcomes to compute the cost function $C(\boldsymbol{\theta})$. Following this step, update the parameters $\boldsymbol{\theta}$ via the classical optimization method. Continue iterating through this process until the predefined convergence criterion is satisfied.

Now, let us delve into each step with more precision.
In Variational Quantum Algorithms (VQAs), the initial step involves constructing a quantum circuit, a pivotal tool for manipulating the quantum state, and aligning it with the optimal solutions of the given optimization problem. This is achieved by defining a parameterized unitary operator, represented as $U(\boldsymbol{\theta})$, as mentioned in (1). Throughout the optimization process, these $\boldsymbol{\theta}$ parameters undergo gradual adjustments with the goal of steering the quantum state closer to optimality. Just as in Adiabatic Quantum Computation (AQC), in VQAs we need an initial quantum state too, denoted as $|\psi_0\rangle$. Commonly, this starting state is chosen as either $|0\rangle^{\otimes n} = |00\ldots0\rangle$ (the all-zero state) or $|+\rangle^{\otimes n}$, though alternate selections are viable as well. In each iteration of VQA, the quantum state $|\psi(\boldsymbol{\theta})\rangle$ is derived as a result of applying the unitary operator $U(\boldsymbol{\theta})$ to the initial state $|\psi_0\rangle$. The fundamental idea revolves around systematically tuning the parameters $\boldsymbol{\theta}$ in such a manner that $|\psi(\boldsymbol{\theta})\rangle$ progressively converges towards the optimal solution.
The second step (2) of Variational Quantum Algorithms (VQAs) centers on the selection of the cost function. As discussed in the previous chapter, the optimal solution to our combinatorial optimization problem corresponds to the lowest energy eigenvector of the Problem Hamiltonian H_p. Importantly, H_p is both real and diagonal in the computational basis, rendering it hermitian. A well-known result in Linear Algebra tells us that, for a Hermitian matrix H, the following holds true:

$$\underset{x\neq 0}{\operatorname{argmin}} \frac{x^T H x}{x^T x} = \underset{x:\, \|x\|=1}{\operatorname{argmin}} x^T H x = u_1 \qquad \min_{x\neq 0} \frac{x^T H x}{x^T x} = \min_{x:\, \|x\|=1} x^T H x = \lambda_1, \tag{5.8}$$

where u_1 represents the eigenvector associated with the lowest eigenvalue λ_1 of H. The quadratic form featured in this equation is referred to as the **Ritz Quotient**. This elucidates why we opt for the cost function $C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H_p(\sigma_1^z, \cdots, \sigma_n^z)|\psi(\boldsymbol{\theta})\rangle$.

Let us delve a bit deeper into this cost function, often referred to as the *Expected Energy of H_p in the state $|\psi(\boldsymbol{\theta}\rangle$*. The terminology arises from the fact that the cost function $C(\boldsymbol{\theta})$ in each iteration represents an actual Mean Value. This is due to the quantum state $|\psi(\boldsymbol{\theta})\rangle$ being a superposition of states:

$$|\psi(\boldsymbol{\theta})\rangle = \sum_{x\in\{0,1\}^n} \alpha_x(\boldsymbol{\theta})|x\rangle, \qquad \sum_{x\in\{0,1\}^n} |\alpha_x(\boldsymbol{\theta})|^2 = 1.$$

Considering that H_p is diagonal in the *orthonormal* computational basis, we can simplify the expression to get

$$\begin{aligned} C(\boldsymbol{\theta}) &= \sum_{y,x\in\{0,\,1\}^n} \overline{\alpha_y(\boldsymbol{\theta})}\alpha_x(\boldsymbol{\theta})\langle y|\underbrace{H_p|x\rangle}_{=\mathcal{E}((-1)^x)|x\rangle} \\ &= \sum_{y,x\in\{0,\,1\}^n} \overline{\alpha_y(\boldsymbol{\theta})}\alpha_x(\boldsymbol{\theta})\mathcal{E}((-1)^x)\langle y|x\rangle \\ &= \sum_{x\in\{0,\,1\}^n} \overline{\alpha_x(\boldsymbol{\theta})}\alpha_x(\boldsymbol{\theta})\mathcal{E}((-1)^x)\langle x|x\rangle \\ &= \sum_{x\in\{0,1\}^n} |\alpha_x(\boldsymbol{\theta})|^2\mathcal{E}((-1)^x). \end{aligned}$$

Hence, $C(\boldsymbol{\theta})$ effectively represents the Mean Value of a random variable distributed across the set $\{\mathcal{E}((-1)^x) \mid x \in \{0,\,1\}^n\}$ with probabilities determined by $\{|\alpha_x|^2 \mid x \in \{0,\,1\}^n\}$. $\mathcal{E}(\cdot)$ is the Ising Energy as defined in (4.2.3). It is crucial to bear this in mind as it underscores why simulating VQAs on classical computers is highly inefficient.
The third step (3) in this procedure involves selecting a classical optimizer to fine-tune $C(\boldsymbol{\theta})$. In principle, any classical optimizer is a valid choice. However, it is important to recognize that different optimizers come with their own advantages and limitations, so the selection should be considered thoughtfully.
For instance, *Gradient-Based* algorithms are a viable option, with partial derivatives estimated using finite differences. Yet, this method requires two calls to the quantum computer for each parameter θ_m, as partial derivative estimation relies on the formula:

$$\frac{\partial C(\boldsymbol{\theta})}{\partial\theta_m} \sim \frac{C(\theta_1,\ldots,\theta_{m-1},\theta_m+\epsilon,\theta_{m+1},\ldots,\theta_M) - C(\theta_1,\ldots,\theta_{m-1},\theta_m-\epsilon,\theta_{m+1},\ldots,\theta_M)}{2\epsilon}$$

with $\epsilon \sim 0$. While this approach can potentially lead to quicker convergence, it also demands more quantum resources.
Another option is the *Nelder-Mead simplicial approximation method*, which requires even more calls to the quantum devices. In this approach, approximately $M+1$ evalua-

tions are necessary at each iteration. The choice of the classical optimization algorithm holds significant importance within Variational Quantum Algorithms (VQAs). While theoretically any algorithm is suitable, practical considerations warrant a more judicious selection.

Several commonly used algorithms include *Nelder-Mead* [18], *Gradient Methods* such as *L-BFGS* [19] (employing gradient estimation via finite differences), *Bayesian Optimization* [20], *Powell's method* [21], *COBYLA* [22], *BOBYQA* [23], *Multi-Level Single Linkage* [24], and *APOSMM* [25]. A comparative analysis of some of these methods can be found in the reference [26].

Step (4) involves the initialization of parameters $\boldsymbol{\theta}$. We have flexibility here, ranging from random initial guesses to more sophisticated strategies. In cases where expert knowledge is available or for relatively straightforward problems, leveraging this expertise can lead to better initialization. Additionally, Machine Learning techniques can be employed. ML approaches typically revolve around running the algorithm on simpler instances of the same combinatorial optimization problem. Once optimal parameters for these simpler cases are identified, they can be used to initialize parameters for more complex instances. We will delve deeper into this in the upcoming subsection on QAOA.

It is crucial to note that VQAs, much like other heuristics, are highly sensitive to parameter initialization. This sensitivity arises from their typically complex, non-convex energy landscapes. Just like the choice of the classical optimizer, the initialization step carries significant importance as well.

Finally, step (5) establishes an iterative process, guiding our framework toward the desired optimum. When optimal parameters $\boldsymbol{\theta}^*$ are discovered, the final quantum state can be computed as $|\psi\rangle = U(\boldsymbol{\theta}^*)|\psi_0\rangle$. Measuring this state in the computational basis yields the optimal solution for our combinatorial optimization problem.

To conclude this section, let us delve a little bit deeper into the classical simulation of VQAs.

As we have discussed, the cost function $C(\boldsymbol{\theta})$ inherently represents an **Expected Value**. This probabilistic nature arises from the fundamentals of quantum mechanics and quantum computing. However, classical computers operate deterministically, lacking the capacity to reason in terms of probabilities. Consequently, they cannot directly compute the Expected Value $C(\boldsymbol{\theta})$. In classical simulations, we resort to estimating $C(\boldsymbol{\theta})$, typically through **Monte Carlo approximation** techniques. Unfortunately, this introduces a significant computational inefficiency.

To illustrate, consider a particular iteration of VQAs, where we have the quantum state $|\psi(\boldsymbol{\theta})\rangle = \sum_{x\in\{0,1\}^n} \alpha_x(\boldsymbol{\theta})|x\rangle$. During classical simulation, what we possess is an observed value of $|\psi(\boldsymbol{\theta})\rangle$ — essentially a measurement of $|\psi(\boldsymbol{\theta})\rangle$ in the computational basis. In

essence, we obtain some state $|x\rangle$ with a probability given by $|\alpha_x(\boldsymbol{\theta})|^2$. To estimate

$$C(\boldsymbol{\theta}) = \sum_{x \in \{0,1\}^n} |\alpha_x(\boldsymbol{\theta})|^2 \mathcal{E}((-1)^x),$$

we must repeatedly compute and observe $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_0\rangle$ numerous times with the same parameter set $\boldsymbol{\theta}$. Each observation yields a state $|x\rangle$. Suppose that after k iterations, we have observed $x_1, \ldots, x_k$. We can then approximate $C(\boldsymbol{\theta})$ using the empirical mean:

$$C(\boldsymbol{\theta}) \sim \frac{1}{k} \sum_{i=1}^{k} \mathcal{E}((-1)^{x_i}).$$

The challenge here lies in the substantial number of samples required to accurately estimate the mean value. A typical approach involves continuous sampling until the unbiased sample variance of the estimator falls below a specified threshold ϵ.
Another challenge in the classical simulation of VQAs arises due to memory constraints. For instance, consider a combinatorial optimization (CO) problem with n variables, where the corresponding Ising Hamiltonian H_p becomes a square matrix of dimension 2^n, and the quantum state becomes a vector of the same dimension 2^n. Quantum computers exploit properties like superposition and entanglement to perform operations on multiple qubits simultaneously, working effectively with high-dimensional quantum states. Classical computers, on the other hand, typically require the entire vector representation of the quantum state, which in CO problems grows exponentially with the number of variables. This exponential growth in memory requirements can quickly exceed the available resources of classical computers. As a result, classical simulations of VQAs face considerable memory constraints.
To address this challenge, classical simulations may implement parallelization techniques, such as distributing the computation across multiple processing cores or leveraging specialized hardware like GPUs. However, it is important to note that classical parallelization methods do not replicate the same inherent parallelism found in quantum computation.
Quantum computers naturally exploit parallelism through quantum gates that operate on multiple qubits simultaneously due to the principles of superposition and entanglement. In contrast, classical computers require manual parallelization efforts, which can make simulating quantum algorithms computationally demanding and resource-intensive.

5.2.1 Variational Quantum Eigensolver

VQE plays a fundamental role in the realm of Variational Quantum Algorithms (VQAs), offering extensive versatility in designing and employing parameterized Ansatzes.

Initially aimed at quantum chemistry, VQE's origins trace back to addressing molecular interactions within quantum systems. Quantum chemistry applications often incorporate the **Coupled Cluster Ansatz** to capture intricate electron correlations.
However, since its first appearance VQE has quickly extended its influence into various domains, including combinatorial optimization. Its effectiveness hinges on skillfully selecting the Ansatz. When addressing combinatorial optimization, an **Hardware Efficient Ansatz** is commonly favored. This designation implies the Ansatz can be efficiently executed on specific quantum gate architecture we are working with. Typically, Hardware Efficient Ansatzes encompass rotations around the Bloch Sphere's axis, followed by entanglement operations to link qubits. The introduction of entanglement establishes intricate interdependencies among qubits, causing changes in one qubit to impact those entangled with it. Diverse entanglement strategies, such as linear and circular entanglement, allow for distinct qubit connections. Linear entanglement joins neighboring qubits, while circular entanglement creates a loop encompassing all qubits, including the first and last. More intricate patterns involve multi-qubit connections, presenting a broad spectrum of possibilities. This flexibility underscores VQE's power, granting us substantial freedom in customizing both entanglement patterns and the overall Ansatz.
In our context, as we delve into subsequent chapters, our approach embraces a specific Ansatz structure which involves parameterized rotations around the Y-axis of the Bloch sphere, followed by linear entanglement achieved through *parameterized controlled gates*. Following this, Y-rotations are individually applied to each qubit once again. This tailored configuration underscores VQE's adaptability and potential, demonstrating also its capability to address complex quantum challenges across diverse fields.

5.2.2 Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA) represents a specialized Variational Quantum Algorithm (VQA) tailored for solving combinatorial optimization problems. QAOA, following the VQA paradigm, approaches optimization problems by mapping them onto Hamiltonians and then employing a sequence of unitary operations, represented as a quantum circuit, to prepare a quantum state whose measurement yields the optimal solution.
In QAOA, a distinctive feature involves the alternation between two distinct Hamiltonians, referred to as the *Mixing Hamiltonian* (or *driver*) and the *Problem Hamiltonian*. This approach parallels the methodology of Adiabatic Quantum Computation, where an initial Hamiltonian H_0 and a final Hamiltonian $H_1 = H_p$ are used. The Mixing Hamiltonian H_0 serves the purpose of creating a superposition of all feasible solutions to the optimization problem, while the Problem Hamiltonian H_p is employed to dynamically adjust the state towards the optimal solution.
Thus, the QAOA quantum circuit ansatz typically comprises alternating layers of two

types of unitary operations, each parameterized by specific angles denoted as γ and β, respectively. Below is an illustrative example of a QAOA ansatz with p layers:

$$|\psi(\boldsymbol{\beta},\boldsymbol{\gamma})\rangle = U_{\beta_p} U_{\gamma_p} \cdots U_{\beta_1} U_{\gamma_1} |+\rangle^{\otimes n} \tag{5.9}$$

where

$$U_{\beta_j} = e^{-i\beta_j \sum_1^n \sigma_j^x}, \qquad U_{\gamma_j} = e^{-i\gamma_j H_p}, \qquad |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle.$$

The vectors γ and β represent p-dimensional parameter vectors that require optimization to achieve the best possible approximation of the optimal solution.
Notably, the study in [27] has demonstrated that the selection of such an ansatz is optimal when the angles γ and β vary within predefined finite ranges.
Does the equation in (5.9) ring a bell? Indeed, it closely resembles the equation we derived in (5.6). In essence, QAOA is a discretization of Adiabatic Quantum Computation (AQC) using Trotter approximation formulas. This connection is highly significant as it implies that, as p approaches infinity, QAOA becomes equivalent to AQC, ultimately converging to the optimal solution.
Furthermore, in the context of problems like *Maxcut, Max2Sat*, and *Modularity Maximization*, an interesting observation emerges: as p increases, the optimal β parameter tends to decrease while the optimal γ parameter tends to increase. This progression closely aligns with the annealing process in Adiabatic Quantum Computation (AQC), where the system transitions gradually from H_0 to H_p.

Geometric Interpretation of QAOA

Now, let us delve into a geometric understanding of the QAOA ansatz. For simplicity, we will use the notation:

$$\sigma_i^x = X_i = I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i},$$

where X denotes the Pauli-X matrix.
First, let us consider the *Mixing Unitary* H_M (i.e., H_0) in QAOA:

$$e^{-i\beta H_M} = e^{-i\beta \sum_i X_i} = e^{-i\beta \sum_i I^{\otimes i-1}\otimes X\otimes I^{\otimes n-i}} \overset{(1)}{=} \prod_i e^{-i\beta (I^{\otimes i-1}\otimes X\otimes I^{\otimes n-i})} \overset{(2)}{=} \prod_i I^{\otimes i-1} \otimes e^{-i\beta X} \otimes I^{\otimes n-i}.$$

Before explaining the reasons behind (1) and (2), let us introduce the concept of matrix exponentiation.
For a square matrix A, matrix exponentiation is computed using the Taylor series ex-

pansion of the exponential function:

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = I + A + \frac{1}{2}A^2 + \cdots$$

It is worth noting that if A is diagonalizable (meaning A can be represented as UDU^{-1} with U being invertible and D diagonal), then we have:

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = \sum_{k=0}^{\infty} \frac{(UDU^{-1})^k}{k!} = \sum_{k=0}^{\infty} \frac{UD^kU^{-1}}{k!} = U\left(\sum_{k=0}^{\infty} \frac{D^k}{k!}\right)U^{-1} = Ue^DU^{-1},$$

where the exponentiation of the diagonal matrix D involves simply exponentiating each of its individual entries.
It is important to highlight that matrix exponentiation doesn't follow the classical rules of exponentiation. Additionally, for two matrices A and B, the following conditions are equivalent:

- $e^{A+B} = e^Ae^B$;
- A, B commute;
- A, B are simultaneously diagonalizable;
- there exists U change of basis such that

  $$A = UD_AU^{-1}, \quad B = UD_BU^{-1}$$

  with D_A, D_B diagonal matrices.

Now, let us explore the reasoning behind (1). The Pauli-X matrix is unitarily diagonalizable, meaning we can represent X as $X = UDU^{-1}$ for some unitary U and diagonal D. Therefore, we can change the basis in each Hilbert space in the tensor product to transform:

$$I^{\otimes i-1} \otimes X \otimes I^{\otimes n-i} \to I^{\otimes i-1} \otimes D \otimes I^{\otimes n-i} \quad \forall i.$$

Since all the matrices in the sum are simultaneously diagonalizable, we have:

$$e^{-i\beta\sum_i X_i} = e^{-i\beta\sum_i I^{\otimes i-1}\otimes X\otimes I^{\otimes n-i}} = \prod_i e^{-i\beta(I^{\otimes i-1}\otimes X\otimes I^{\otimes n-i})}.$$

This explains (1). But what about (2)? Notice that

$$\begin{aligned}
e^{\alpha I\otimes A\otimes I} &= \sum_k \frac{(\alpha I\otimes A\otimes I)^k}{k!} \\
&= I\otimes I_A\otimes I + \alpha(I\otimes A\otimes I) + \frac{1}{2}\alpha^2(I\otimes A\otimes I)^2 + \cdots \\
&= I\otimes I_A\otimes I + \alpha(I\otimes A\otimes I) + \frac{1}{2}\alpha^2(I\otimes A\otimes I)(I\otimes A\otimes I) + \cdots \\
&\overset{(2.1)}{=} I\otimes I_A\otimes I + \alpha(I\otimes A\otimes I) + \frac{1}{2}\alpha^2(II\otimes AA\otimes II) + \cdots \\
&= I\otimes I_A\otimes I + \alpha(I\otimes A\otimes I) + \frac{1}{2}\alpha^2(I\otimes A^2\otimes I) + \cdots \\
&\overset{(2.2)}{=} I\otimes I_A\otimes I + (I\otimes \alpha A\otimes I) + (I\otimes \frac{1}{2}\alpha^2A^2\otimes I) + \cdots \\
&= I\otimes \sum_k \frac{(\alpha A)^k}{k!}\otimes I \\
&= I\otimes e^{\alpha A}\otimes I
\end{aligned}$$

where we used the following two properties of the tensor product:

2.1 $(A\otimes B)(C\otimes D) = AC\otimes BD$;

2.2 $\alpha(A\otimes B) = \alpha A\otimes B = A\otimes \alpha B$ for every $\alpha\in\mathbb{C}$.

Generalizing the previous calculations we finally prove

$$e^{-i\beta(I^{\otimes i-1}\otimes X\otimes I^{\otimes n-i})} = I^{\otimes i-1}\otimes e^{-i\beta X}\otimes I^{\otimes n-i} \quad \forall i.$$

To conclude, we proved that:

$$\begin{aligned}
e^{-i\beta\sum_i X_i} = e^{-i\beta\sum_i I^{\otimes i-1}\otimes X\otimes I^{\otimes n-i}} &= \prod_i e^{-i\beta(I^{\otimes i-1}\otimes X\otimes I^{\otimes n-i})} \\
&= \prod_i I^{\otimes i-1}\otimes e^{-i\beta X}\otimes I^{\otimes n-i}.
\end{aligned}$$

At this point we have:

$$
\begin{aligned}
&I^{\otimes i-1} \otimes e^{-i\beta X} \otimes I^{\otimes n-i} |\psi_1 \cdots \psi_n\rangle \\
&= (I^{\otimes i-1} \otimes e^{-i\beta X} \otimes I^{\otimes n-i})(|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle) \\
&= I|\psi_1\rangle \otimes \cdots \otimes I|\psi_{i-1}\rangle \otimes e^{-i\beta X}|\psi_i\rangle \otimes I|\psi_{i+1}\rangle \otimes \cdots \otimes I|\psi_n\rangle \\
&= |\psi_1\rangle \otimes \cdots \otimes |\psi_{i-1}\rangle \otimes e^{-i\beta X}|\psi_i\rangle \otimes |\psi_{i+1}\rangle \otimes \cdots \otimes |\psi_n\rangle.
\end{aligned}
$$

We find that the transformation leaves all the qubits unchanged but for the i-th one, which is transformed via $|\psi_i\rangle \mapsto e^{-i\beta X}|\psi_i\rangle$.
Now the question is, what is such transformation actually doing? We have

$$
X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}}_{U} \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}^{-1}}_{U^{-1}}
$$

where in this case $U^{-1} = U^T = U$. At this point

$$
e^{-i\beta X} = U \cdot \begin{bmatrix} e^{-i\beta} & 0 \\ 0 & e^{i\beta} \end{bmatrix} \cdot U^{-1}.
$$

We then have:

$$
\begin{aligned}
e^{-i\beta X}|0\rangle &= e^{-i\beta X} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \cdots = \cos\beta|0\rangle - i\sin\beta|1\rangle \\
e^{-i\beta X}|1\rangle &= e^{-i\beta X} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \cdots = -i\sin\beta|0\rangle + \cos\beta|1\rangle.
\end{aligned}
$$

At this point the geometric interpretation follows easily. When we apply the Mixing Unitary on the state $|\psi\rangle$, it essentially applies a qubit-wise rotation to the state, rotating each qubit individually within its own Bloch Sphere. This rotation stems from the factorization:

$$
e^{-i\beta \sum_i X_i} = \prod_i I^{\otimes i-1} \otimes e^{-i\beta X} \otimes I^{\otimes n-i}.
$$

Each term in the product leaves all qubits in $|\psi\rangle$ unchanged except for the i-th qubit, on which a rotation is applied. Combining all these terms collectively rotates the entire state. These rotations enable the algorithm to explore the configuration space, seeking optimal solutions. The goal is to find the correct angles that rotate the initial state toward the optimal solution for the optimization problem.
Similarly to H_M, H_p can also be seen as a rotation operator, with the rotation being

guided by problem-specific information encoded in the problem Hamiltonian. Since H_p is diagonal in the computational basis, we have:

$$e^{-i\gamma H_p} \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle = \sum_{x\in\{0,1\}^n} e^{-i\gamma\lambda_x} \alpha_x |x\rangle$$

where λ_x is the eigenvalue of H_p corresponding to the basis state $|x\rangle$, namely $H_p|x\rangle = \lambda_x|x\rangle$.

QAOA mixing operator

Now, let us explore some variants of the Quantum Approximate Optimization Algorithm (QAOA), specifically, what happens when we consider a different mixing Hamiltonian.
The choice of the *Transverse Field Mixing Hamiltonian* (5.4) in QAOA has been quite popular and convenient. However, it is essential to understand that it is not the only option available. In certain situations, especially for constrained optimization problems, it can be beneficial to design custom mixers tailored to the specific problem. These custom mixers have two primary objectives:

- preserving the feasibility of solutions;
- enabling exploration across the entire feasible region.

By carefully designing a mixer, it becomes possible to incorporate problem-specific knowledge or take advantage of the inherent structure of the optimization problem. This can lead to improved performance and potentially better solutions.
With a fixed mixing operator, all the problem's dependencies must be captured in the cost Hamiltonian H_p. As we discussed in the chapter related to QUBO problems, the general strategy is to include hard constraints as penalty terms in the cost function and then convert this cost function into a cost Hamiltonian. However, this approach means that the algorithm must search a much larger solution space than if the evolution were confined to feasible configurations only, which can make the search less efficient.
To address this issue, one can consider replacing the fixed driver (i.e., the mixing Hamiltonian H_M) with an alternative driver that restricts the evolution to the feasible subspace only. Intuitively, mixing operators that limit the search to feasible solutions should result in more efficient algorithms.
In a paper by Venturelli et al. [28], two fundamental design criteria for a good mixing operator are laid out:

- Preserve the feasible subspace.
- Provide transitions between all pairs of states corresponding to feasible points.

The paper offers several examples of mixers designed for specific combinatorial optimization problems, such as *Graph Coloring* and *Job Shop Scheduling*, among others. The primary concept behind crafting such mixers is to identify a **swap rule** that allows for easy transitions between feasible solutions. It is worth noting that devising such mixers is a challenging task, both theoretically and practically. Custom mixers tend to be complex and may not be efficiently implementable on all quantum devices.
Before we proceed further, let us examine a practical example.

Problem: *Given a graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, find the largest subset $V_0 \subset V$ of mutually non-adjacent vertices. Subsets that meet this condition are referred to as independent.*

- The configuration space is the set of n-bit strings representing subsets $V_0 \subset V$ of vertices, where $i \in V_0$ if and only if $x_i = 1$.
- The feasible region F is represented by the subset of all n-bit strings corresponding to independent sets of G.
- Given an independent set V_0, we can add a vertex $w \notin V_0$ to V_0 while maintaining feasibility only if none of its neighboring vertices $nbhd(w)$ are already in V_0.
- On the other hand, we can always remove any vertex $w \in V_0$ without affecting feasibility.
- Hence, a **bit-flip** operation at a vertex, **controlled** by its neighbors (adjacent vertices), suffices both to remove and add vertices while maintaining the independence property.

These considerations suggest the custom mixer described below. For each vertex, define the partial mixer as a multiply-controlled X operator

$$H_v = 2^{-D_v} X_v \otimes \bigotimes_{w \in nbhd(v)} (I + Z)_w, \tag{5.10}$$

where D_v is the degree of v in G.
Notice that for each of the vertices non explicitly appearing in H_v we have an underlying and not written tensor product with an Identity matrix.
The total mixer is then given by $H = \sum_{v \in V(G)} H_v$ with corresponding mixing unitary

$$U = e^{-i\beta H} = e^{-i\beta \sum_v H_v}.$$

Why define H_v as given in Equation (5.10)? Let us break it down. Consider a fixed neighbor w in the neighborhood of v and assume that w belongs to V_0, which means

$x_w = 1$. In this case, the action of $(I + Z)$ on $|x_w\rangle$ results in $(I + Z)|1\rangle = |1\rangle - |1\rangle = 0$. As a result, when we take the tensor product, the overall action of H_v on the state $|x\rangle$ collapses to zero, specifically, $H_v|x\rangle = 0$. This means that the Hamiltonian H_v, which controls the value of x_v, does not contribute to the total Hamiltonian H.
Now, let us consider the opposite scenario where, for every neighbor w in the neighborhood of v, w is not in V_0, meaning $x_w = 0$ for all $w \in nbhd(v)$. In this case, the action of $(I + Z)$ on $|x_w\rangle$ results in $(I + Z)|0\rangle = |0\rangle + |0\rangle = 2|0\rangle$. Since there are D_v neighbors, the constant factor 2^{-D_v} simplifies, and the Hamiltonian H_v effectively acts as the identity on all $w \in nbhd(v)$. Now, each neighbor term remains in state $|0\rangle$, thus not the zero vector, and the action of X_v survives in the final tensor product, effectively changing the state of variable x_v.
The paper also discusses some partitioning strategies to efficiently implement such mixer. More details in the reference [28].

QAOA Parameter Optimization

Much like in Variational Quantum Eigensolver (VQE) and VQA approaches in general, optimizing the parameters in the Quantum Approximate Optimization Algorithm (QAOA) poses significant challenges due to the non-convex nature of the objective function and the presence of low-quality local optima. We devote a subsection to this topic to explore some intriguing ideas that have emerged in the literature.
As expected from heuristic algorithms, parameter optimization is highly sensitive to the starting point, i.e., the initial values of β and γ. While in some special cases, optimal angles can be found analytically [29], this is generally not the norm. Nonetheless, prior knowledge about the problem can guide the choice of angle intervals. For example, if we anticipate that a particular qubit should be in a specific state (e.g., $|0\rangle$ or $|1\rangle$) in the optimal solution, we can restrict the angles to a specific interval. This restriction of angles helps expedite the optimization process. However, it is essential to strike a balance between exploration and exploitation when deciding on angle restrictions.
What's particularly interesting here is the empirical observation that QAOA's optimal parameters tend to cluster in the same region for problems belonging to the same class. This clustering phenomenon holds across instances of different sizes and complexities. It has been noted in problems such as Maxcut [30], Max2Sat [31], and Modularity Maximization [32] on graphs. This clustering behavior of optimal parameters suggests the potential application of a machine learning approach.
The idea behind this approach is to train a model on smaller problem instances and leverage the learned parameters as a starting point for optimization on larger and more complex instances. Suppose we have a training set of T instances, each with a known optimal solution denoted as $|g_t\rangle$. The goal is to find the best set of QAOA parameters that can lead the parameterized quantum state, once measured in the computational basis, to reproduce the optimal solution for these training instances. These learned pa-

rameters can then be used to construct the final circuit for the problem, or alternatively, they can serve as a warm start for further optimization on test instances.
The training instances can be generated randomly by fixing the problem's size and randomly varying its parameters, or they could be selected from benchmark instances. It is crucial to keep in mind that current Noisy Intermediate-Scale Quantum (NISQ) devices have a very limited number of qubits, often incapable of handling even moderately sized instances. In most cases, the training instances will be quite small. Nevertheless, despite this limitation, the literature suggests that the learned parameters tend to generalize effectively, even when applied to larger instances.
Having discussed the selection of the training set, the next critical aspect in a machine learning model is the choice of the loss function. In our case, how do we determine the loss function? One possibility, as suggested by [31], is to consider the *Mean Squared Overlap Loss*

$$L(\boldsymbol{\beta}, \boldsymbol{\gamma}) = -\frac{1}{T} \sum_{t=1}^{T} |\langle \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) | g_t \rangle|^2.$$

The t-th term in the sum represents the probability of the QAOA final state collapsing onto the optimal solution for the t-th training instance upon measurement. Indeed, assume

$$|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle = \sum_{x \in \{0,1\}^n} \alpha_x(\boldsymbol{\beta}, \boldsymbol{\gamma}) |x\rangle,$$

we have

$$|\langle \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) | g_t \rangle|^2 = \left| \sum_{x \in \{0,1\}^n} \overline{\alpha_x(\boldsymbol{\beta}, \boldsymbol{\gamma})} \langle x | g_t \rangle \right|^2 = |\alpha_{g_t}(\boldsymbol{\beta}, \boldsymbol{\gamma})|^2$$

so that

$$L(\boldsymbol{\beta}, \boldsymbol{\gamma}) = -\frac{1}{T} \sum_{t=1}^{T} |\alpha_{g_t}(\boldsymbol{\beta}, \boldsymbol{\gamma})|^2.$$

Minimizing this loss function classically aims to determine values for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ that, on average, bring the quantum state $|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle$ close to the optimal solution for each of the training instances. In particular, we seek values for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ to maximize the probability that, upon measurement, the quantum state collapses onto the optimal solution for each training instance.

Of course, there are various alternative approaches to obtaining initial angle values. For example, in [33], a different loss function is proposed:

$$L(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{1}{T} \sum_{t=1}^{T} |\langle \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) | H_t | \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) \rangle|.$$

Here, H_t represents the Problem Hamiltonian associated with the t-th training instance. These Hamiltonians are linked to combinatorial optimization problems similar to the one we aim to solve but typically smaller in scale.
In a more recent study [34], a different approach is taken, leveraging Deep Reinforcement Learning [35] and Kernel Density Estimation techniques [36]. By utilizing learned knowledge, this approach has demonstrated the capability to yield efficient solutions with significantly fewer quantum circuit evaluations.

QAOA Gate Depth

As previously mentioned, QAOA is essentially a discretization of Adiabatic Quantum Computation (AQC) achieved through Trotterization. In theory, as the gate depth, denoted as p, tends towards infinity, QAOA converges to the true optimum due to the adiabatic theorem of quantum mechanics. However, in practice, achieving this limit ($p \to \infty$) is not feasible. The limitations of current quantum hardware architectures often restrict us from going beyond $p > 2$ due to resource constraints. This is because increasing the number of layers in the quantum circuit requires more executions on the quantum hardware, which significantly impacts both resource usage and execution time. Empirically, it has been observed that a large number of random initial guesses, on the order of $2^{O(p)}$, are needed to converge to a good solution.
So, how can we extend the QAOA ansatz to efficiently handle larger values of p? Researchers, as discussed in [30], have noted that as the depth p increases, optimal parameter values tend, once again, to cluster in the same regions. Leveraging this clustering behavior, it becomes possible to generate an initial set of parameters for a $(p+1)$-depth QAOA circuit based on the optimal parameters obtained for a p-depth circuit. This warm start strategy can lead to better solutions compared to random initialization.
In their paper, the authors propose two heuristics, known as the FOURIER and INTERP heuristics, for constructing $(p+1)$-depth parameters using the previously obtained optimal parameters. This approach significantly reduces the number of initial guesses to approximately $O(poly(p))$, making it more practical for higher values of p. Let us take a closer look at the INTERP strategy.
For a given problem instance, the optimization process starts at $p = 1$. After reaching a local optimum with parameters $(\boldsymbol{\gamma}^L_{(p)}, \boldsymbol{\beta}^L_{(p)})$, the depth is incremented to $p+1$. To initialize parameters for the $(p+1)$-depth circuit, the values $(\boldsymbol{\gamma}^0_{(p+1)}, \boldsymbol{\beta}^0_{(p+1)})$ are generated

using the following procedure:

$$\left[\gamma^0_{(p+1)}\right]_i = \frac{i-1}{p}\left[\gamma^L_{(p)}\right]_{i-1} + \frac{p-i+1}{p}\left[\gamma^L_{(p)}\right]_i \tag{5.11}$$

for $i = 1, 2, \ldots, p+1$. Here, by $\left[\gamma\right]_i$ we denote the i-th element of the parameter vector γ, and $\left[\gamma^L_{(p)}\right]_0 = \left[\gamma^L_{(p)}\right]_{p+1} \equiv 0$. The process for obtaining $\beta^0_{(p+1)}$ follows a similar approach.
After determining $(\gamma^0_{(p+1)}, \beta^0_{(p+1)})$ as described earlier, the optimization procedure is applied to reach a local optimum $(\gamma^L_{(p+1)}, \beta^L_{(p+1)})$ for the $(p+1)$-level QAOA. Subsequently, the depth is increased by one, and this process is repeated iteratively until the desired depth level is achieved.
In summary, the INTERP heuristic generates $(p+1)$ initial parameter sets by linearly interpolating the previously determined optimal parameters for QAOA. This approach efficiently initializes parameters for higher-depth circuits.
In contrast, the FOURIER heuristic takes a different approach. It combines the optimal parameters obtained for the previous p-depth circuit using a Fourier sum-like method. For more comprehensive details on the FOURIER heuristic, please refer to the provided reference [30].

5.2.3 Adiabatically Assisted VQE

In the preceding sections, we delved into the intricacies of both Adiabatic Quantum Computation (AQC) and Variational Quantum Algorithms (VQAs). Let us now provide a concise summary of their respective advantages and disadvantages:

- **Adiabatic Quantum Computation (AQC)**
  - **PRO**: Theoretically, given a large enough annealing time, it unfailingly converges to the solution.
  - **PRO**: It can be fine-tuned to select a more efficient adiabatic path.
  - **CON**: Its speed may degrade exponentially depending on the gap between the lowest and second-lowest eigenvalues of the Hamiltonian.

- **Variational Quantum Algorithm (VQA)**
  - **PRO**: Utilizes an arbitrary parameterized quantum circuit to minimize the problem Hamiltonian.
  - **PRO**: Explores the parameter space through classical optimization algorithms.

- **CON**: Vulnerable to the choice of initial parameters, often converging to local optima.
- **CON**: Requires a large number of measurements.

Now, can we leverage the strengths of both these approaches to create a more robust algorithm? Indeed, that's precisely what **AAVQE (Adiabatically Assisted Variational Quantum Eigensolver)** accomplishes. AAVQE operates by iteratively applying VQE to a series of Hamiltonians $\hat{H}(s) = (1-s)H_0 + sH_p$, which evolve from the initial Hamiltonian H_0 ($s = 0$) to the Problem Hamiltonian H_p ($s = 1$) by discrete time steps Δs.
The algorithm can be summarized as follows:

1. Prepare the ground state of a simple Hamiltonian $\hat{H}(s_0)$ with a quantum circuit using VQE. Its final characterization is given by the parameters $\boldsymbol{\theta}_{s_0}^{(f)}$.
2. Add a step Δs, that is $s_{i+1} = s_i + \Delta s$.
3. Run a VQE on $\hat{H}(s_{i+1})$ using as initial parameters $\boldsymbol{\theta}_{s_{i+1}}^{(0)} = \boldsymbol{\theta}_{s_i}^{(f)}$, the final parameters from the previous step. Compute the new set of optimal parameters $\boldsymbol{\theta}_{s_{i+1}}^{(f)}$.
4. If $s = 1$ stop, else go to 2).

For more comprehensive details on AAVQE, please refer to the provided reference in [37].

5.3 Quantum Assisted Eigensolver

QAE, as introduced in [38], presents a novel hybrid classical-quantum algorithm designed for estimating the ground state of Hamiltonians. Notably, it distinguishes itself from Variational Quantum Algorithms (VQAs) by eschewing the use of a classical-quantum feedback loop during optimization. In QAE, quantum hardware is employed just once to calculate the matrices necessary for the subsequent classical optimization step.
Consider a given Hamiltonian $H = \sum_i \beta_i U_i$, $U_i \in SU(2^n)$, the QAE algorithm revolves around three primary steps:

1. **Selection of an Ansatz**: A quantum state Ansatz is chosen, composed as a linear combination of quantum states obtained from a set $\{|\psi_j\rangle\}_{j=1}^{m}$. These states are defined such that $\langle\psi_j|\psi_k\rangle \neq 1$ for $j \neq k$, and each $|\psi_j\rangle$ can be represented

as $|\psi_j\rangle = V_j|0\rangle^{\otimes n}$, where V_j is an efficiently implementable unitary operator in $SU(2^n)$. The Ansatz takes the form:

$$|\psi(\boldsymbol{\alpha})\rangle = \sum_{j=1}^{m} \alpha_j |\psi_j\rangle.$$

2. **Calculation of Overlap Matrices**: Quantum computation is utilized to calculate two critical matrices, D and E, for the subsequent classical optimization. These matrices are defined as follows: D is determined by the elements D_{jk} and is computed using the formula:

$$D_{jk} = \sum_i \beta_i \langle\psi_j|U_i|\psi_k\rangle \quad \text{for all } j,k = 1,\ldots,m.$$

E is constructed with elements E_{jk} and is computed as:

$$E_{jk} = \langle\psi_j|\psi_k\rangle \quad \text{for all } j,k = 1,\ldots,m.$$

3. **Execution of Quadratically Constrained Quadratic Program (QCQP)**: A classical computer is employed to solve a Quadratically Constrained Quadratic Problem (QCQP). The goal is to minimize the expression $\overline{\alpha}^T D\alpha$ under the constraint $\overline{\alpha}^T E\alpha = 1$, where α belongs to $\mathbb{C}^m$. This classical optimization step yields the desired ground state information.

It is important to note that QAE requires only a single use of the quantum computer to calculate the matrices D and E. Subsequently, the algorithm proceeds with classical computation, making it computationally efficient.
Despite its apparent simplicity, QAE remains a relatively unexplored approach. The algorithm's success hinges significantly on the selection of the initial states $\{|\psi_j\rangle\}_j$, which represents a critical aspect for achieving favorable results. In scenarios where expert knowledge is available, leveraging such insights to provide suitable initial states could potentially elevate the competitiveness of QAE when compared to the other methods discussed in previous sections.
To wrap up this chapter, let us delve into the rationale behind the matrices D and E.
D is formulated by assessing the *Expected Energy of the Hamiltonian H within the state* $|\psi(\alpha)\rangle$:

$$\langle H(\alpha)\rangle = \sum_{j,k,i} \beta_i \overline{\alpha_j} \alpha_k \langle\psi_j|U_i|\psi_k\rangle = \sum_{j,k} \overline{\alpha_j} \left(\sum_i \beta_i \langle\psi_j|U_i|\psi_k\rangle \right) \alpha_k = \sum_{j,k} \overline{\alpha_j} D_{jk} \alpha_k = \overline{\alpha}^T D\alpha.$$

Thus, D is represented as a set of elements D_{jk} determined by the following expression:

$$D_{jk} = \sum_i \beta_i \langle \psi_j | U_i | \psi_k \rangle \quad \forall j,k = 1,\ldots,m.$$

On the other hand, E is derived by ensuring that $|\psi(\alpha)\rangle$ adheres to the condition of being a valid quantum state, specifically, having a unit norm:

$$\sum_{j,k=1}^{m} \overline{\alpha_j} \langle \psi_j | \psi_k \rangle \alpha_k \stackrel{!}{=} 1.$$

This condition can be expressed concisely as:

$$\overline{\alpha}^T E \alpha = 1.$$

In this case, E is defined as a collection of elements E_{jk}:

$$E_{jk} = \langle \psi_j | \psi_k \rangle \quad \forall j,k = 1,\ldots,m.$$

Consequently, the ultimate goal leads to the formulation of the Quadratically Constrained Quadratic Problem (QCQP), which is summarized as follows:

$$\begin{aligned} &\min \ \overline{\alpha}^T D \alpha \\ &\text{s.t.} \ \ \overline{\alpha}^T E \alpha = 1 \\ &\qquad \alpha \in \mathbb{C}^m. \end{aligned}$$

This problem can be efficiently addressed using classical computing techniques.
Once more, Quantum Assisted Eigensolver (QAE) remains relatively uncharted territory, primarily because of the intricate task of selecting the initial set of states $\{|\psi_j\rangle\}_{j=1}^m$. When a reliable strategy for this selection is not available, the effectiveness of QAE doesn't match up to the other algorithms we have examined in this chapter.

6

The Capacitated Vehicle Routing Problem

6.1 Intro to Vehicle Routing Problems

The **Vehicle Routing Problem (VRP)** family encompasses a class of logistical challenges involving the efficient delivery of goods or services. Given a collection of specific transportation requests and a fleet of vehicles, the primary goal is to optimize the allocation of tasks to vehicles, sequence the deliveries or services, and minimize overall costs. This entails determining which vehicle serves which requests, in what order, while ensuring that each vehicle's route is operationally viable.
In VRP scenarios, the transportation requests are typically concentrated at specific locations within a road network. The fundamental components of VRP encompass the transportation requests and their logistical requirements, the available fleet of vehicles, associated costs, potential revenues (if applicable), and the feasibility of the generated routes.
At its core, VRP focuses on efficiently servicing a set of customers within a designated timeframe, using a fleet of vehicles stationed at one or more depots. These vehicles are operated by assigned crews or drivers, and they navigate through a defined road network. The ultimate goal is to determine a set of routes, with each route executed by a single vehicle that originates and concludes its journey at its respective depots. These routes must satisfy all customer demands, adhere to operational constraints, and achieve the lowest possible total transportation cost.
The road network utilized for transporting goods is typically represented as a graph. In this graph, the edges symbolize distinct road segments, and the nodes correspond to intersections, as well as the depots and customers locations. Whether these edges are directed or undirected depends on whether they permit travel in just one direction, as seen in one-way streets, common in urban or highway networks, or in both directions. Each edge within this network is linked to two key attributes: a cost, typically reflecting

its length, and a travel time. The travel time might vary based on the type of vehicle or the specific time period during which the edge is traversed.
Characteristics commonly associated with customers include:

- The specific vertex within the road network where the customer is situated.
- The quantity of goods they require, which may consist of various types, that need to be either delivered to or collected from the customer.
- Time intervals within the day, known as *time windows*, during which the customer can receive service. This often accounts for factors such as the customer's operating hours or traffic-related restrictions.
- The durations needed for delivering or collecting goods at the customer's location. These durations might vary based on the type of vehicle employed.
- The subset of available vehicles that can be used for servicing the customer. This is often due to factors like accessibility limitations or particular loading and unloading needs.

In certain situations, it may not be feasible to fully meet the demands of every customer. In such cases, options include reducing the quantities to be delivered or collected, or deciding not to serve certain customers at all. To handle these scenarios, different priorities or penalties associated with partial or complete service omission can be assigned to the customers.
The routes designed for serving customers have both starting and ending points, which can be one or more depots situated at different locations on the road graph. Each depot is characterized by the number and types of vehicles it possesses and its overall capacity for handling goods.
In certain practical scenarios, customers are initially divided among the depots, and the vehicles are required to return to their home depot after each route. In such instances, the broader VRP can be divided into several independent sub-problems, each associated with a distinct depot.
The transportation of goods is carried out using a fleet of vehicles, and typical attributes of these vehicles encompass:

- The home depot of the vehicle and the potential to conclude a service at a depot different from its home base.
- The vehicle's capacity, which could be expressed in terms of maximum weight, volume, or the number of pallets it can transport.

- The possibility of subdividing the vehicle into compartments, each characterized by its capacity and the types of goods it can accommodate.
- Equipment available for loading and unloading operations.
- A subset of arcs within the road graph that can be traversed by the vehicle.
- Costs associated with utilizing the vehicle, which may be per distance unit, per time unit, per route, and so on.

Drivers operating these vehicles are required to adhere to various constraints outlined in union contracts and company regulations. These constraints include factors such as working hours during the day, the number and duration of breaks during service, maximum allowable driving periods, and overtime. In the subsequent discussions, the constraints applicable to drivers are encompassed within those associated with the corresponding vehicles.
The routes must adhere to various operational constraints, which are contingent upon factors such as the nature of the goods being transported, the desired service quality, and the characteristics of both customers and vehicles. Some typical operational constraints include:

- Ensuring that the current load of the associated vehicle along each route does not exceed the vehicle's capacity.
- Customers served on a route can either require the delivery or the collection of goods, or both options might exist.
- Customers can only be served within their specified time windows and during the working periods of the drivers affiliated with the vehicles visiting them.

Additionally, precedence constraints can be enforced regarding the order in which customers are visited within a route. This is particularly relevant in scenarios such as *pickup and delivery* problems, where vehicles must handle both the collection and the delivery of goods, and the items collected from pickup customers must be transported to the corresponding delivery customers by the same vehicle.
The process of evaluating the overall cost of the routes and verifying that they adhere to the operational constraints necessitates access to information regarding the travel cost and travel time between every pair of customers, as well as between the depots and the customers. To accomplish this, the original road graph, which often possesses a sparse structure, is typically transformed into a complete graph. This complete graph is comprised of vertices corresponding to the road graph's own vertices, including customers and depots.

For each pair of vertices denoted as i and j within this complete graph, an arc labeled as (i, j) is established. Each of these arcs is linked to two distinct parameters: a travel cost denoted as c_{ij} and a travel time represented as t_{ij}. This graph can be either directed or undirected, depending on the specific context.
Vehicle routing problems can encompass various, and at times conflicting, objectives. Some common objectives include:

- Minimizing the overall transportation cost, which is contingent on the total distance traveled (or the total travel time) and the fixed costs associated with the vehicles in use (including their associated drivers).
- Minimizing the count of vehicles (or drivers) required to serve all customers.
- Achieving a balance in the routes concerning both travel time and vehicle load.
- Reducing penalties related to the partial service of customers.

Moreover, these objectives can be combined in a weighted manner, allowing for the consideration of multiple criteria simultaneously.
A specific instance of the VRP, occurring when only one vehicle is accessible at the depot, and no additional operational constraints are imposed, reduces to the well-known **Traveling Salesman Problem (TSP)**.
The considerable interest of the global research community in the various VRP variants stems not only from their recognized complexity as combinatorial optimization problems but also from their practical significance in industry. As a result, researchers from academic and industrial backgrounds alike are actively engaged in this field. The numerous real-world applications have clearly demonstrated that employing computerized solution techniques for VRP, both in planning and operational contexts, results in significant reductions in overall transportation costs. Compared to non-optimized procedures, substantial cost savings and more efficient use of the vehicle fleet can be achieved. Furthermore, these planning software tools facilitate automation, standardization, and integration into an organization's broader planning processes, leading to quicker and more cost-effective planning compared to manual methods.
It is also noteworthy that the mathematical models and the exact and metaheuristic algorithms developed for the VRP serve as a crucial reference point not only for research in this specific area but also for the broader field of combinatorial optimization. Indeed, the VRP and its various incarnations are frequently employed as benchmarks for testing new models and algorithmic techniques initially aimed at tackling other challenging combinatorial optimization problems. A valuable reference for a deeper exploration of the Vehicle Routing Problem (VRP) is provided in [39].

6.2 Capacitated VRP

The **Capacitated Vehicle Routing Problem (CVRP)** stands out as the most fundamental and extensively explored variant within the realm of Vehicle Routing Problems (VRP). In the CVRP, all customers exclusively represent delivery locations, each with known and fixed demand values, which cannot be subdivided. The fleet of vehicles is composed of identical units stationed at a solitary central depot. The sole constraints imposed are those related to the vehicle's capacity. The primary goal is to minimize the overall cost, which is typically defined as a composite function of both the number of routes used and their respective lengths or travel times, while ensuring the service of all customers.

As discussed earlier, we can characterize the CVRP as a graph theoretical problem. To this end, consider a graph $G = (V, A)$, comprising a set of vertices $V = 0, \ldots, n$ and an arc set A. Within this structure, vertices indexed from 1 to n correspond to the various customers, while vertex 0 is associated with the depot. Occasionally, the depot is represented by vertex $n + 1$.

Each arc (i, j) in the graph is associated with a nonnegative cost c_{ij}, representing the expense incurred for traveling from vertex i to vertex j. Generally, the use of loop arcs, (i, i), is disallowed in this context. When the graph is directed, the cost matrix A becomes asymmetric, leading to what is known as the **Asymmetric Capacitated Vehicle Routing Problem (ACVRP)**. Conversely, in cases where $c_{ij} = c_{ji}$ holds for all $(i, j) \in A$, the problem assumes the form of the **Symmetric Capacitated Vehicle Routing Problem (SCVRP)**. In symmetric instances, the arc set A is often replaced by a set of undirected edges, denoted as E. In this discussion, our focus will primarily be on the more general ACVRP. However, the mathematical formulations provided can be readily adapted to accommodate the symmetric scenario as well.

To ensure the problem's feasibility, several key attributes are associated with graph G. Firstly, it must be **strongly connected** to avoid dead-end routes that render the problem unsolvable. Moreover, it is typically assumed that graph G is **complete**, meaning that every pair of vertices is directly connected by an arc or edge.

For any given vertex i, the *forward star*, represented as $\delta^+(i)$, is defined as the collection of vertices j for which the arc (i, j) exists, signifying those destinations directly accessible from i. Similarly, the *backward star* of vertex i, denoted as $\delta^-(i)$, encompasses the vertices j connected by arcs (j, i), indicating the origins from which i can be directly reached. When considering a vertex subset $S \subseteq V$, $\delta(S)$ and $E(S)$ represent the set of edges $e \in E$ with one or both endpoints within S, respectively. In cases where only a single vertex, say $i \in V$, is under consideration, $\delta(i)$ is used instead of $\delta(\{i\})$.

In several practical cases, the cost matrix satisfies the triangle inequality

$$c_{ik} + c_{kj} \geq c_{ij} \qquad \forall i, j, k \in V.$$

In other words, it is not convenient to deviate from the direct link between two vertices i and j. The presence of the triangle inequality is sometimes required by the algorithms for CVRP, and this may be obtained in a simple way by adding a suitably large positive quantity M to the cost of each arc. However, this method, while ensuring the triangle inequality, can significantly distort the metric, potentially resulting in poor lower and upper bounds compared to those based on the original costs.
It is important to note that when the cost of each arc in the graph equals the cost of the shortest path between its endpoints, the corresponding cost matrix inherently satisfies the triangle inequality.
In some instances, the vertices are mapped to points in a plane, each with specific coordinates. The cost c_{ij} for each arc $(i, j) \in A$ is then defined as the Euclidean distance between the respective points representing vertices i and j. In this setup, the cost matrix is symmetric and inherently satisfies the triangle inequality, leading to a problem variant known as the **Euclidean Symmetric Capacitated Vehicle Routing Problem (Euclidean SCVRP)**.
Each customer, indexed as i ($i = 1, \ldots, n$), has a known nonnegative demand, denoted as d_i, representing the quantity to be delivered. The depot is assigned a fictitious demand of $d_0 = 0$. Given a subset of vertices $S \subseteq V$, the notation $d(S) = \sum_{i \in S} d_i$ is used to denote the total demand associated with that particular set S.
At the central depot, there is a fleet of K identical vehicles, each equipped with a capacity of C. To ensure the problem's feasibility, it is assumed that for every customer $i = 1, \ldots, n$, their demand d_i is strictly less than the vehicle capacity C. Moreover, it is assumed that each vehicle can execute at most one route. The constraint $K \geq K_{\min}$ must be enforced, where $K_{\min}$ represents the minimum number of vehicles required to service all customers. Determining $K_{\min}$ can be achieved by solving the **Bin Packing Problem (BPP)** associated with the CVRP. BPP entails finding the minimum number of bins (vehicles), each with a capacity of C, needed to accommodate all n items (customers), each with a nonnegative weight (demand) d_i, $i = 1, \ldots, n$.
Although BPP is NP-hard in the strong sense, instances with hundreds of items can be optimally solved very effectively. Below a general instance of BPP where we wish to

pack n items in bins of capacity at most C.

$$\min \sum_{i=1}^{n} y_i \tag{6.1}$$

$$\text{s.t.} \sum_{i} x_{ij} = 1 \qquad \forall j = 1, \ldots, n \tag{6.2}$$

$$\sum_{j} d_j x_{ij} \leq C \qquad \forall i = 1, \ldots, n \tag{6.3}$$

$$\sum_{j} d_j x_{ij} \leq C y_i \qquad \forall i = 1, \ldots, n \tag{6.4}$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 1, \ldots, n \tag{6.5}$$

$$y_i \in \{0, 1\} \qquad i = 1, \ldots, n. \tag{6.6}$$

In the previous formulation of BPP, the binary variables x_{ij} account for the assignment of item j to the i-th bin; the binary variables y_i on the other hand account for the usage of the i-th bin.
Given a set $S \subseteq V \setminus \{0\}$, we denote by $r(S)$ the minimum number of vehicles needed to serve all customers in S, i.e., the optimal solution value of the BPP associated with itemset S. Note that $r(V \setminus \{0\}) = K_{\min}$. Often, $r(S)$ is replaced by the trivial BPP lower bound

$$r(S) = \lceil d(S)/C \rceil. \tag{6.7}$$

The Capacitated Vehicle Routing Problem (CVRP) can be defined as the task of identifying a set of precisely K simple circuits, where each circuit corresponds to a vehicle route. These circuits collectively aim to minimize the total cost, which is defined as the sum of the costs associated with the arcs traversed by these circuits. This optimization problem comes with certain conditions:

- **Depot Visitation:** Each circuit must start and end its route to the depot vertex.
- **Customer Coverage:** Every customer vertex should be visited by exactly one circuit.
- **Capacity Constraint:** The total demand of the vertices visited within a circuit cannot surpass the vehicle's capacity C.

The realm of CVRP research has explored various adaptations and extensions of the fundamental problem. Here are some notable variants:

- **Unused Vehicles:** In cases where the number of available vehicles K exceeds the minimum necessary ($K > K_{\min}$), it may be permissible to have some vehicles remain unused. This can be achieved by introducing fixed costs related to vehicle deployment. Consequently, the objective may expand to minimize not only the total cost but also the number of circuits (i.e., the vehicles utilized).

- **Diverse Capacities:** It may happen that the vehicles at hand differ in their capacities, i.e., they possess varying capacities C_k for $k = 1, \ldots, K$.

- **Single Customer Routes:** Some problem formulations disallow routes that consist of only one customer.

CVRP is recognized as an NP-hard problem and serves as a generalization of the well-known Traveling Salesman Problem (TSP). The TSP aims to determine the lowest-cost simple circuit that visits all vertices in the graph G (known as a **Hamiltonian circuit**) and typically arises when $C > d(V)$ and $K = 1$ (i.e., a single vehicle with ample capacity). Therefore, any relaxation techniques or algorithms developed for the TSP can be applied to the CVRP as well.

6.2.1 Problem Formulation

In this subsection, we introduce two primary formulations employed for modeling the Capacitated Vehicle Routing Problem (CVRP). These formulations can be broadly categorized into two types: **Vehicle Flow Formulations** and **Set-Partitioning Formulations**.

Vehicle Flow Formulations utilize integer variables associated with each arc or edge in the graph. These variables count the number of times a vehicle traverses a specific arc or edge. They are frequently used for the fundamental versions of VRP. These models are well-suited when the solution cost can be expressed as the sum of arc-related costs, and when the primary constraints pertain to direct transitions between customers within a route. Such constraints can be effectively captured by defining the arc set and arc costs appropriately. However, vehicle flow models have limitations. They are less suitable for addressing practical scenarios where solution costs depend on factors such as the entire sequence of vertices or the type of vehicle assigned to a route. Additionally, the linear programming relaxation of vehicle flow models can be notably weak when faced with stringent additional operational constraints.

Set-Partitioning Formulations, on the other hand, involve an exponential number of binary variables, each associated with a distinct feasible circuit. The VRP is reformulated as a **Set-Partitioning Problem (SPP)**. In this context, the goal is to determine a collection of circuits with the minimum cost, ensuring that each customer is served exactly once and potentially accommodating additional constraints. A key advantage of this

model type is its flexibility in handling diverse route costs, including those dependent on the entire sequence of arcs and the vehicle type. Furthermore, the side constraints, if any, can be directly checked when building the routes, allowing for a more concise set of inequalities. This results in a formulation with a linear programming relaxation that is typically much stronger compared to that of the previous models. Nevertheless, it is essential to note that these models typically entail managing a very large number of variables.

Vehicle Flow Model Formulation

The model is a **two-index vehicle flow formulation** that uses $O(n^2)$ binary variables x to indicate if a vehicle traverses an arc in the optimal solution. In other words, variable x_{ij} takes value 1 if arc $(i,j) \in A$ belongs to the optimal solution and takes value 0 otherwise.

$$\min \quad \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{6.8}$$

$$\text{s.t.} \quad \sum_{i\in\delta^-(j)} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\} \tag{6.9}$$

$$\sum_{j\in\delta^+(i)} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\} \tag{6.10}$$

$$\sum_{j\in\delta^-(0)} x_{j0} = K \tag{6.11}$$

$$\sum_{j\in\delta^+(0)} x_{0j} = K \tag{6.12}$$

$$\sum_{i\notin S} \sum_{j\in S} x_{ij} \geq r(S) \qquad \forall S \subseteq V \setminus \{0\},\ S \neq \emptyset \tag{6.13}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{6.14}$$

The indegree and outdegree constraints (6.9) and (6.10) impose that exactly one arc enters and leaves each vertex associated with a customer, respectively. Analogously, constraints (6.11) and (6.12) impose the degree requirements for the depot vertex. The so-called **Capacity-Cut Constraints (CCCs)** of (6.13) impose both the connectivity of the solution and the vehicle capacity requirements. In fact, they stipulate that each cut $(V \setminus S, S)$ defined by a customer set S is crossed by a number of arcs not smaller than $r(S)$ (minimum number of vehicles needed to serve set S). The CCCs remain valid also if $r(S)$ is replaced by the trivial BPP lower bound given in (6.7).
Observe that when $|S| = 1$ or $S = V \setminus \{0\}$ the CCCs (6.13) are weakened forms of the corresponding degree constraints (6.9)–(6.10). Note also that, because of the degree

constraints (6.9)–(6.10), we have

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} = \sum_{i \in S} \sum_{j \notin S} x_{ij} \qquad \forall S \subseteq V \setminus \{0\},\ S \neq \emptyset.$$

CCCs constraints as expressed in (6.13) have a cardinality growing exponentially with n. This means that it is practically impossible to solve directly the linear programming relaxation of the problem.
To address this challenge, one approach is to initially consider only a limited subset of these constraints and incorporate the remaining constraints selectively as required. This approach aligns with the essence of the recursive **Dantzig-Fulkerson-Johnson (DFJ)** formulation for the Traveling Salesman Problem (TSP).
Alternatively, a set of constraints equivalent to those in (6.13) but with only a polynomial cardinality can be derived. This can be accomplished by adopting the **subtour elimination constraints** of **Miller-Tucker-Zemlin (MTZ)** formulation for the TSP

$$u_i - u_j + C x_{ij} \leq C - d_j \qquad \forall i \neq j \in V \setminus \{0\}, \text{ with } d_i + d_j \leq C \tag{6.15}$$

$$d_i \leq u_i \leq C \qquad \forall i \in V \setminus \{0\} \tag{6.16}$$

$$u_i \geq 0 \qquad \forall i \in V \setminus \{0\} \tag{6.17}$$

where u_i is an additional continuous variable representing the load of the vehicle after visiting customer i. It is easy to see that these new constraints impose both the capacity and the connectivity requirements of ACVRP. Indeed, when $x_{ij} = 0$, constraint (6.15) is not binding since $u_i < C$ and $u_j > d_j$ whereas when $x_{ij} = 1$, they impose that $u_j > u_i + d_j$.
Notice however that the linear programming relaxation of formulation (6.8)–(6.12), (6.14), (6.15)–(6.17) generally is much weaker than that of formulation (6.8)–(6.14).
Additionally, it is important to note that, as discussed in previous sections, when dealing with Combinatorial Optimization (CO) problems on quantum devices, we need to formulate them as Quadratic Unconstrained Binary Optimization (QUBO) or Ising problems, which rely entirely on binary variables. Consequently, continuous variables like u_i must be discretized in some manner.
Two-index vehicle flow models have found extensive application in modeling the fundamental versions of Symmetric Capacitated Vehicle Routing Problem (SCVRP) and Asymmetric Capacitated Vehicle Routing Problem (ACVRP), along with certain variants like the Vehicle Routing Problem with Backhauls (VRPB). However, their applicability tends to be limited when dealing with more intricate VRP versions. This is primarily because they are only suitable when the solution cost can be expressed as the sum of costs attributed to traversed arcs. Moreover, they do not readily provide information about which vehicle is assigned to traverse a particular arc within the solution. Hence, these models are ill-suited for scenarios where the cost or feasibility of a circuit depends on the overall sequence of vertices or the type of vehicle allocated to the route.

A possible way to partially overcome some of the drawbacks associated with the two-index models is to explicitly indicate the vehicle that traverses an arc, so that more complex constraints may be imposed on the routes. In this way one obtains the so-called **three-index vehicle flow formulation** of SCVRP and ACVRP, which uses $O(n^2K)$ binary variables x: variable x_{ijk} counts the number of times arc $(i,j) \in A$ is traversed by vehicle k ($k = 1,\dots,K$) in the optimal solution. In addition, there are $O(nK)$ binary variables y: variable y_{ik} ($i \in V$; $k = 1,\dots,K$) takes value 1 if customer i is served by vehicle k in the optimal solution and takes value 0 otherwise. The three-index model for ACVRP is given in the following.

$$\min \quad \sum_{(i,j)\in A} \sum_{k=1}^{K} c_{ij} x_{ijk} \tag{6.18}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} y_{ik} = 1 \qquad \forall i \in V \setminus \{0\} \tag{6.19}$$

$$\sum_{k=1}^{K} y_{0k} = K \tag{6.20}$$

$$\sum_{j\in\delta^+(i)} x_{ijk} = \sum_{\delta^-(i)} x_{jik} = y_{ik} \qquad \forall i \in V,\ k = 1,\dots,K \tag{6.21}$$

$$\sum_{i\in V} d_i y_{ik} \le C \qquad \forall k = 1,\dots,K \tag{6.22}$$

$$\sum_{i\in S} \sum_{j\notin S} x_{ijk} \ge y_{hk} \qquad S \subseteq V \setminus \{0\},\ h \in S,\ k = 1,\dots,K \tag{6.23}$$

$$y_{ik} \in \{0,1\} \qquad \forall i \in V,\ k = 1,\dots,K \tag{6.24}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in A,\ k = 1,\dots,K. \tag{6.25}$$

Constraints (6.19)–(6.21) impose that each customer is visited exactly once, that K vehicles leave the depot, and that the same vehicle enters and leaves a given customer (at most once), respectively.
Constraints (6.22) are the capacity restriction for each vehicle k, whereas constraints (6.23) impose the connectivity of the route performed by each vehicle k. These latter constraints may be replaced by **Subtour Elimination Constraints (SECs)**

$$\sum_{i\in S} \sum_{j\in S} x_{ijk} \le |S| - 1 \qquad S \subseteq V \setminus \{0\},\ |S| \ge 2,\ k = 1,\dots,K, \tag{6.26}$$

which impose that for each vehicle k at least 1 arc leaves each vertex set S visited by vehicle k and not containing the depot. Alternatively, the three-index version of the

generalized Miller-Tucker-Zemlin subtour elimination constraints (6.15) can be used.

$$u_{ik} - u_{jk} + Cx_{ijk} \leq C - d_j \qquad \forall i, j \in V \setminus \{0\}, i \neq j, (i,j) \in A \text{ such that } d_i + d_j \leq C, k = 1, \ldots, K, \tag{6.27}$$

$$d_i \leq u_{ik} \leq C \qquad \forall i \in V \setminus \{0\}, k = 1, \ldots, K. \tag{6.28}$$

Note that these constraints replace also the capacity requirements (6.22).
Once more, it is essential to emphasize that in order to derive QUBO/Ising formulations, it is necessary to discretize and binarize the continuous variables, like u for instance.
Three-index vehicle flow models have found extensive application in modeling more intricate versions of the Vehicle Routing Problem (VRP), such as the *Vehicle Routing Problem with Time Windows (VRPTW)*. This popularity arises from their greater adaptability in accommodating additional problem constraints and features. However, it is important to acknowledge that these models come with the drawback of significantly increasing the number of variables involved. On the flip side, they serve as a generalization of the two-index models, which can be obtained by simply defining $x_{ij} = \sum_{k=1}^{K} x_{ijk}$.
To model the case in which some vehicles may be left unused can be obtained by replacing constraints (6.11) and (6.12) in the two-index model with

$$\sum_{i \in V} x_{i0} \leq K \tag{6.29}$$

$$\sum_{i \in V} x_{i0} = \sum_{j \in V} x_{0j}, \tag{6.30}$$

whereas in the three-index model, constraint (6.20) may be replaced with

$$\sum_{k=1}^{K} y_{0k} \leq K. \tag{6.31}$$

Three-index vehicle flow models may easily take into account the case of a nonhomogeneous fleet, where each vehicle may have a different capacity C_k, $k = 1, \ldots, K$. This is obtained by replacing C with C_k in the capacity constraints (6.22).
Finally, in some cases, routes serving a single customer are not allowed. In the models for the ACVRP, this can be imposed by adding the following additional constraints:

$$x_{0j} + x_{j0} \leq 1 \qquad \forall j \in V \setminus \{0\}. \tag{6.32}$$

Set Partitioning Formulation

The set-partitioning (SP) formulation of the VRP was originally proposed by Balinski and Quandt and uses a possibly exponential number of binary variables, each associated with a different feasible circuit of G. More specifically, let $H = \{H_1, \ldots, H_q\}$ denote the collection of all the circuits of G, each corresponding to a feasible route, with $q = |H|$. Each circuit H_j has an associated cost c_j. In addition, let a_{ij} be a binary coefficient that takes value 1 if vertex i is visited (or covered, in the set partitioning jargon) by route H_j and takes value 0 otherwise.
The binary variable x_j, $j = 1, \ldots, q$, is equal to 1 if and only if circuit H_j is selected in the optimal solution. The model is

$$\min \sum_{j=1}^{q} c_j x_j \tag{6.33}$$

$$\text{s.t.} \sum_{j} a_{ij} x_j = 1 \qquad \forall i \in V \setminus \{0\} \tag{6.34}$$

$$\sum_{j=1}^{q} x_j = K \tag{6.35}$$

$$x_j \in \{0, 1\} \qquad \forall j = 1, \ldots, q. \tag{6.36}$$

Constraints (6.34) impose that each customer i is covered by exactly one of the selected circuits, and (6.35) requires that K circuits are selected. This is a very general model that may easily take into account several constraints as, for example, time windows, since route feasibility is implicitly considered in the definition of set H. Moreover, the linear programming relaxation of this formulation typically is very tight.
One of the main drawbacks here is represented by the huge number of variables, which, in non-tightly-constrained instances with tens of customers, may easily run into the billions. The explicit generation of all the feasible circuits is thus normally impractical, and one has to resort to a **column generation** approach to solve the linear programming relaxation.

6.2.2 Solution Strategies

Exact algorithms for solving the Capacitated Vehicle Routing Problem (CVRP) primarily rely on *Branch and Bound* techniques. Branch and Bound methods have been extensively applied over the past few decades to tackle the CVRP and its various versions. Notably, for complex variants like the Asymmetric CVRP (ACVRP) and the Distance-Constrained CVRP (DCVRP), these algorithms remain at the forefront of exact solution methods.

As highlighted in the introduction, the CVRP is an extension of the well-known Traveling Salesman Problem (TSP). The TSP seeks to determine a Hamiltonian circuit with the minimum cost, visiting a given set of points exactly once. Consequently, many exact approaches for the CVRP have evolved from the substantial and successful body of work dedicated to solving the TSP.
Historically, until the late 1980s, the most effective exact methods for the CVRP primarily relied on Branch and Bound algorithms employing fundamental combinatorial relaxations.
In more recent times, the best results have been achieved using *Branch and Cut* algorithms. These sophisticated approaches merge Branch and Bound procedures with cutting-plane techniques. To understand these methods better, let us start with a brief overview of Branch and Bound.
The *linear relaxation* of an integer linear program (IP) is a linear program derived from IP by removing the constraint that all variables must be integers. Consequently, the optimal value z_{LP} of this relaxation (in the minimization case) serves as a lower bound for the optimal value z_{IP} of the integer linear program. When the constraint set of an integer linear program is small enough to allow its linear relaxation to be solved using an LP (Linear Programming) solver, a classical approach to solving it is Branch and Bound, coupled with linear programming bounds.
Here's how it works: Initially, the linear relaxation is solved. If the optimal solution x^* is **integral** (all variables are integers), the problem is solved. Otherwise, a fractional variable x_i is chosen, and two new linear programs are constructed.
In the first linear program, an upper bound on x_i is set to $\lfloor x_i^* \rfloor$, while in the second, a lower bound is established at $\lceil x_i^* \rceil$. These two programs create the **branching**. The process then continues with classical branch-and-bound, where the bounds are determined by the optimal solution values of the linear programs associated with the nodes in the search tree.
However, when the integer linear program has a large number of constraints or when the linear relaxation is enhanced by incorporating families of valid inequalities (typically of exponential size), the constraint system cannot be directly handed off to an LP solver. In such cases, a cutting-plane technique becomes necessary to address the linear program.
The cutting-plane algorithm is designed to solve integer programs (IP) with a large number of constraints by iteratively refining the linear relaxation (LP(∞)) of the IP. Here's how it works:

1. Begin with LP(0), a linear program that contains a manageable subset of constraints from LP(∞). Solve LP(0) to obtain an optimal solution $\overline{x}^0$.

2. If $\overline{x}^0$ is feasible for the integer program (IP), it is an optimal solution. Otherwise, assume there exists a black-box algorithm, called the **separation algorithm**,

which can either provide a violated constraint from LP(∞) or confirm that all constraints are satisfied.

3. If the separation algorithm returns violated constraints, add them to LP(0) to obtain LP(1). Note that for every $h \geq 0$, if $z_{LP(h)}$ is the optimal value of LP(h), we have $z_{LP(h)} \leq z_{LP(h+1)} \leq z_{LP(\infty)} \leq z_{IP}$.

4. Repeat the process until you either find an optimal solution to IP or reach the linear relaxation LP(∞). In practice, the separation algorithm might not always be exact, but it still provides a lower bound on z_{IP}.

5. If you haven't obtained an optimal solution to IP, perform branching. This involves breaking the problem into two new problems, typically by adding upper and lower bounds to a variable with a fractional value. Each of these new problems is solved recursively using the same method. The optimal solution to the original problem will be the best of these two solutions.

The cutting-plane procedure concludes either with an optimal solution to IP or with an optimal solution x^* to the linear relaxation that is not an optimal solution to IP.
The core of the Branch and Cut technique lies in the fusion of branching with cutting-plane methods. This approach harnesses the strengths of both branching and cutting-plane methodologies. The bounds produced at each node of the enumeration tree are generally improved because new inequalities are added to the subproblem's formulation. Additionally, the separation algorithm takes advantage of the branching process, perturbing the fractional solution in a way that can't be eliminated by an inequality from LP(∞).
This interdisciplinary approach, where different techniques complement each other, is a hallmark of Branch and Cut. Branch and Cut has proven to be a successful approach for solving many combinatorial optimization problems. However, its performance can be suboptimal under certain conditions:

1. When a robust algorithm for the cutting plane phase is lacking.

2. When the number of iterations in the cutting plane phase becomes excessively high.

3. When the linear program becomes unsolvable due to its size.

4. When the branching tree grows too large, making it unlikely to terminate within a reasonable timeframe.

For issues (1) and (2), there are partial solutions. Regarding (1), Branch and Cut doesn't necessarily require an exact solution for the relaxed linear program (LP) before

proceeding to the branching phase. Hence, an approximate but effective separation procedure, like a good heuristic, can be sufficient. As for (3), the problem can often be circumvented by periodically cleaning up the linear program, removing inactive constraints.
The central challenge, however, lies in addressing problem (4). Most difficulties in Branch and Cut stem from this scenario. One approach to mitigate this issue is to enhance LP(∞) by either adjusting existing inequalities or introducing new ones. This tactic is effective because there's a direct correlation between problem (4) and the gap between the integer program's solution value and its linear relaxation. To alleviate this, we need to strengthen the linear relaxation, typically by adding linear inequalities that hold true for all solutions. These inequalities, although unnecessary in the integer formulation, push the optimal solution value of the linear relaxation closer to that of the integer program. However, discovering these inequalities is a challenging task and falls under the domain of the **polyhedral study** of the problem.
Besides exact methods, there are situations where obtaining high-quality approximate solutions suffices. This is where heuristics and metaheuristics come into play.
There are two primary categories of heuristics commonly employed for solving the Vehicle Routing Problem (VRP): classical heuristics and metaheuristics.

- **Classical Heuristics:** These heuristics are the foundation of many VRP solution methods. They fall into the category of **construction** and **improvement** procedures. Classical heuristics tend to explore the solution space to a limited extent and are known for providing good-quality solutions within reasonable computation times. One of their key advantages is that they can be easily adapted to handle various constraints and complexities encountered in real-world VRP scenarios. As a result, they remain widely used in commercial VRP software packages.

- **Metaheuristics:** In contrast to classical heuristics, metaheuristics focus on extensive exploration of promising areas within the solution space. These methods employ advanced techniques such as sophisticated neighborhood search strategies, memory structures, and the recombination of solutions. Consequently, metaheuristics tend to generate solutions of much higher quality than classical heuristics. However, this improvement in solution quality often comes at the cost of increased computation time. Additionally, metaheuristics typically depend on context-specific parameters that need careful tuning, making them less versatile for different problem instances. Importantly, metaheuristics depart from classical approaches by accepting deteriorating or even infeasible intermediate solutions during the search process.

While a deep dive into metaheuristics is beyond the scope of this discussion, comprehensive resources on the topic can be found in the literature, such as the reference [4]. In the following, we will provide a brief overview of classical heuristics for solving the CVRP. These can be broadly categorized into three main groups:

- **Constructive Heuristics:** Constructive heuristics work gradually to construct a feasible solution while paying attention to minimizing the overall solution cost. However, these methods do not include a dedicated phase for improving an initial solution. Instead, they focus on creating a viable route plan.
- **Two-Phase Heuristics:** Two-phase heuristics divide the VRP into its natural components and address them separately, often with the potential for feedback loops between the two stages. There are two distinct classes of two-phase heuristics:
  - **Cluster-First, Route-Second Methods:** In this approach, the vertices are initially grouped into feasible clusters, and then a dedicated vehicle route is constructed for each of these clusters. This method organizes the problem by identifying clusters of customers that can be served efficiently together.
  - **Route-First, Cluster-Second Methods:** In contrast, this method starts by constructing a tour that visits all vertices, encompassing all customers and the depot. Subsequently, this tour is segmented into feasible vehicle routes, determining how the customers are allocated to specific routes.
- **Improvement Methods:** Improvement methods aim to enhance an existing feasible solution by executing a series of modifications, such as edge or vertex exchanges, either within individual vehicle routes or between them. These modifications are designed to reduce the total solution cost. Importantly, many constructive algorithms also integrate improvement steps at various stages, blurring the distinction between constructive and improvement methods.

The choice of which heuristic approach to employ often depends on the specific problem instance and its characteristics. Each of these categories has its strengths and weaknesses, and selecting the most appropriate method may involve considering factors like problem size, complexity, and computational resources.

6.3 Two-Phase Heuristic

For the remainder of this chapter, we will focus our discussion on the two-phase heuristic **Cluster-First, Route-Second**. This approach is particularly relevant in the context

of current NISQ (Noisy Intermediate-Scale Quantum) devices. These quantum computing systems face significant constraints, including limited qubit connectivity and the presence of noise, which restrict their ability to handle problems with a large number of variables. Therefore, to effectively address even relatively small instances of optimization problems on NISQ devices, it becomes crucial to minimize the number of variables in the problem formulation.
The two-phase heuristic, *Cluster-First, Route-Second*, emerges as a valuable strategy in this context. It allows us to decompose the original optimization problem into two separate optimization problems, each of which involves a significantly reduced number of variables. This reduction in the number of variables aligns well with the capabilities and limitations of current quantum hardware, making this heuristic an appealing choice for quantum-based optimization tasks.

6.3.1 Clustering Phase

The clustering phase can be executed using various clustering algorithms. A typically good clustering can be accomplished by addressing a Multi-Knapsack Problem (MKP). In this context, the vehicles are likened to knapsacks, and the task is to partition customers among these knapsacks without surpassing the maximum weight limit imposed by each knapsack. Importantly, all knapsacks have identical weight limits, corresponding to the maximum capacity of the vehicles.
To attain the customers clustering, the following combinatorial optimization (CO) problem can be solved to optimality. In this problem, we utilize the variable z_{ki} to denote the assignment of customer i to cluster (or vehicle) k. It is worth noting that in this context, a cluster represents the group of customers served by a single vehicle, so the total number of clusters matches the number of available vehicles.

$$\min \quad \sum_{k=1}^{K} \sum_{(i,j)\in A} c_{ij} z_{ki} z_{kj} \tag{6.37}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} z_{k0} = K \tag{6.38}$$

$$\sum_{k=1}^{K} z_{ki} = 1 \qquad \forall i \in V \setminus \{0\} \tag{6.39}$$

$$\sum_{\substack{i\in V \\ i\neq 0}} d_i z_{ki} \leq C \quad \forall k = 1,\dots,K \tag{6.40}$$

$$z_{ki} \in \{0,1\} \qquad \forall k = 1,\dots,K;\ i \in V. \tag{6.41}$$

It is important to note that in this scenario, we are dealing with a quadratic objective function. Specifically, for a given knapsack (vehicle) denoted as k, the summation $\sum_{(i,j)\in A} c_{ij} z_{ki} z_{kj}$ represents an upper bound on the total travel cost that vehicle k would incur to serve all its assigned customers.
The constraint (6.38) ensures that every knapsack contains the node 0, which corresponds to the depot. This implies that all vehicles will include the depot node within their routes, as they should.
On the other hand, constraints (6.39) guarantee that each node (customer) is assigned to exactly one knapsack, meaning that every customer is served by precisely one vehicle.
Lastly, constraint (6.40) enforces that the weight limit of each knapsack (vehicle) is not exceeded. In other words, the total demand of customers assigned to each vehicle must not surpass the vehicle's capacity.
By solving this Multi-Knapsack Problem (MKP), we can achieve a typically good clustering of customers. Then, within each cluster, a subsequent routing phase is performed.
Once again, note that any clustering algorithm can be utilized in this phase. However, it is crucial to ensure that:

- Each cluster satisfies the capacity constraint.
- The number of clusters matches the number of vehicles.
- Each cluster is connected to the depot.
- Vehicles can traverse all the nodes within their respective clusters.

Typically, the fourth point is not heavily emphasized since it is assumed that the underlying graph G is complete.

6.3.2 Routing Phase

The routing phase is formulated using the well-known Traveling Salesman Problem (TSP). In this phase, the goal is to determine the best route for each vehicle, considering only the customers within its assigned cluster. The fundamental TSP can be represented

as follows:

$$\min \quad \sum_{\substack{(i,j)\in A \\ i,j\in Cluster}} c_{ij}x_{ij} \tag{6.42}$$

$$\text{s.t.} \quad \sum_{\substack{j\in N^+(i) \\ j\in Cluster}} x_{ij} = 1 \qquad \forall i \in Cluster \tag{6.43}$$

$$\sum_{\substack{i\in N^-(j) \\ i\in Cluster}} x_{ij} = 1 \qquad \forall j \in Cluster \tag{6.44}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in Cluster. \tag{6.45}$$

However, this formulation does not consider the issue of **subtours**, which are cycles that do not encompass all the nodes. To generate meaningful solutions, it is crucial to prevent the formation of subtours. There are two prominent methods for enforcing subtour elimination in the Traveling Salesman Problem, known as **MTZ** and **DFJ**, named after their respective creators.

MTZ Subtour Elimination

This formulation, put forward by Miller, Tucker, and Zemlin, introduces a method for tackling subtours by introducing continuous decision variables that represent the times at which each customer is visited. Specifically, for each customer i in the set V excluding the depot (i.e. $V \setminus \{0\}$), a variable t_i is introduced to indicate the time of visit. To ensure the elimination of subtours, the following constraints are added:

$$t_j > t_i \quad \text{if } x_{ij} = 1,$$

which can be formulated as LP constraint as follows

$$t_j \geq t_i - B(1 - x_{ij}), \tag{6.46}$$

for a large constant $B >> 0$.
It is worth noting that when $x_{ij} = 0$, meaning that arc $(i,j) \in A$ is not visited during the tour, constraint (6.46) is trivially satisfied due to the very large value of B. Conversely, when arc $(i,j) \in A$ is visited during the tour, constraint (6.46) enforces the condition that node i must be visited before node j.
MTZ constraints bear a striking resemblance to the polynomial subtour elimination constraints introduced in the two and three-index formulations of CVRP. However, there are certain challenges associated with these types of constraints. First of all, they necessitate the inclusion of n additional time variables, along with approximately $O(n^2)$

extra constraints. Furthermore, for the conversion to QUBO, each of these time variables must be discretized. Additionally, each of the new constraints must be transformed into an equality, necessitating the introduction of supplementary slack variables, which also need to be discretized. In the grand scheme of things, this approach becomes unmanageable for current NISQ devices, as they are not equipped to handle such a high number of variables.

DFJ Subtour Elimination

This formulation was proposed by Dantzig, Fulkerson, Jhonson. To eliminate subtours, for every set S of customers, add a constraint saying that the tour leaves S at least once.

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \qquad \forall S \subset Cluster. \tag{6.47}$$

Once again, it is worth noting that these types of constraints are not new; they were already present in the original formulations of the Capacitated Vehicle Routing Problem (CVRP). Before adopting the polynomial constraints as in the Miller-Tucker-Zemlin (MTZ) strategy, the CVRP was initially formulated with these constraints. However, the Dantzig-Fulkerson-Johnson (DFJ) strategy introduces even more variables than MTZ. DFJ includes an exponential number of constraints, and each of these constraints necessitates the addition of a corresponding slack variable. These slack variables must also be discretized before the problem can be converted into the QUBO (Quadratic Unconstrained Binary Optimization) form.
So why is the Dantzig-Fulkerson-Johnson (DFJ) strategy preferred for NISQ devices in our case? The answer is quite straightforward: we do not need to consider all subsets $S \subset Cluster$ from the beginning. Instead, we can employ a recursive approach by which we add constraints only when necessary, specifically, constraints are added for a subset S only if subtour S appears in the previous optimal solution. The general strategy can be outlined as follows:

1. Start by solving the Traveling Salesman Problem (TSP) with the Linear Programming (LP) formulation without Subtour Constraints.
2. If no subtours are present in the current solution, proceed to step 6.
3. Add subtour constraints only for the subtours that exist in the current solution.
4. Solve the TSP problem again with the newly added constraint.
5. Return to step 2.
6. Return the final TSP solution.

This strategy proves to be highly efficient for our case because NISQ devices can only handle relatively small problem instances. Intuitively, we expect only a small number of subtours to appear in these instances. This stands in contrast to the Miller-Tucker-Zemlin (MTZ) strategy, which necessitates the addition of all $O(n^2)$ constraints in advance, regardless of whether subtours will actually be present.

7

Quantum CVRP

Now that we have established the fundamentals of Quantum Integer Programming and familiarized ourselves with Vehicle Routing Problems (VRP), we can delve into the practical application of quantum algorithms to address the VRP. This final chapter represents the culmination of our theoretical discussions as we put these concepts into action, applying the quantum algorithms we have explored to specific instances of the Capacitated Vehicle Routing Problem (CVRP).
It is worth noting, as we have mentioned multiple times, that current Noisy Intermediate-Scale Quantum (NISQ) devices are limited in terms of qubit count. Consequently, we're constrained to working with very small instances. In this regard, we don't anticipate groundbreaking results. VRP is notoriously challenging even for classical algorithms, which have been in development for much longer than quantum algorithms. Our goal is to, at the very least, replicate the performance of classical approaches on these small instances we're working with.
The structure of this chapter unfolds as follows: In the next section, we provide a concise overview of the most noteworthy outcomes that the literature has generated thus far. Subsequently, we delve into the specifics of our experimentation, discussing the quantum architectures employed, the quantum algorithms subjected to testing, the classical algorithms used for simulation, and the methodology adopted for the random generation of test instances. To benchmark our final results against state-of-the-art classical approaches, we employed Gurobi, which efficiently computes optimal solutions for the small instances we investigated. The code developed for this master's thesis can be accessed on GitHub [40].

7.1 Literature Overview

The first work we mention is [41], a **Systematic Literature Review** of Quantum Computing for Routing Problems. It spans 53 different papers, from 2004 to 2021, in the intersection of Quantum Computing and Routing Problems. TSP engages most of the researchers (32 out of 53 papers), while the VRP accounts for 13 out of 53 papers. The rest of the papers deal with other routing problems. This work served as useful starting point for further research. In the following we discuss the most relevant among the papers we found.

QVRP [42]

In this research, the author addresses the *Vehicle Routing Problem with Time Windows (VRPTW)* using a route-based formulation (i.e. a Set-Partitioning Formulation). They adopt a greedy approach to construct the set of possible routes. One notable contribution of this work is the introduction of a new encoding scheme aimed at reducing the number of qubits required, from n_c to $n_q = 1 + \log_2(n_c)$, where n_c represents the number of variables. Instead of employing one qubit per variable, they utilize the following encoding:

$$|\psi(\theta)\rangle = \sum_{k=1}^{n_c} \beta_k(\theta)[a_k(\theta)|0\rangle_a + b_k(\theta)|1\rangle_a] \otimes |\phi_k\rangle_r.$$

In this equation, the qubits in the *register* $|\phi_k\rangle_r$ serve to identify the variable x_k, while the *ancilla* qubit $[a_k(\theta)|0\rangle_a + b_k(\theta)|1\rangle_a]$ stores its value.
The authors propose a Variational Quantum Eigensolver (VQE) approach with a Hardware Efficient Ansatz:

$$U(\theta) = \prod_{l=1}^{L} \left(\bigotimes_{j=0}^{n_q} R_Y(\theta_{lj})_j \cdot \prod_{j=0}^{n_q-1} CNOT(j, j+1) \right) \cdot \bigotimes_{j=0}^{n_q} H_j.$$

Here, $CNOT(j, j+1) = |0\rangle\langle 0|_j \otimes I_{j+1} + |1\rangle\langle 1|_j \otimes X_{j+1}$ represents the Controlled-NOT gate applied between qubits j and $j+1$, where the subscripts indicate the qubits these transformations operate on. The initial state is set to the all-zero qubit state, denoted as $|\psi_0\rangle = \bigotimes_{j=0}^{n_q} |0\rangle$.
An analysis of the ansatz reveals several key characteristics: Hadamard gates introduce superposition among qubits, CNOT gates entangle subsequent qubits (*linear entanglement*), and Y-axis rotations introduce variational parameters that are to be optimized. This same structure is repeated for a number L of layers. Specifically, remember that when we apply Hadamard gates to the zero state, it effectively transforms it into a **uniform superposition** of all the other states.

The authors discovered that their reduced encoding method can yield results comparable to those achieved with full encoding for small-scale problem instances. However, when dealing with larger problems, which are currently intractable using the full encoding approach, their strategy fails to find optimal solutions.
Keep in mind this ansatz, as it will closely resemble the Hardware Efficient Ansatz we have selected for our numerical experiments.

QVRP [43]

The authors of this study address the Capacitated Vehicle Routing Problem (CVRP) with multiple vehicles using a node-based formulation. They introduce variable $x_{v,t}^{k} \in \{0,1\}$ to denote whether vehicle k visits customer v at time step t. The time steps are limited to a maximum of $T = n$ to account for the worst-case scenario, where one vehicle visits all customers. However, this leads to a total of $K \cdot V \cdot T$ variables, which can be prohibitively large.
To address this issue, the authors employ a heuristic two-phase approach, dividing the CVRP into a Clustering Phase (CP) and a subsequent Traveling Salesman Problem (TSP) phase. In the Clustering Phase, the goal is to group customers in such a way that the demands within each group do not exceed the truck's capacity, and the groups are geographically close together. This clustering phase is modeled as a Multi-Knapsack problem. Then, the TSP is solved independently for each cluster. This approach significantly reduces the number of qubits required for implementation on Noisy Intermediate-Scale Quantum (NISQ) devices.
In their research, the authors focus specifically on solving the TSP problem, excluding the Multi-Knapsack portion. They consider problem instances of sizes 4, 5, and 6 (nodes).
They apply both Quantum Approximate Optimization Algorithm (QAOA) and Variational Quantum Eigensolver (VQE) to tackle the TSP, with various algorithmic variants:

- QAOA;
- Recursive QAOA;
- Warm-start QAOA;
- Constraints-Preserving Custom Mixer QAOA, incorporating a TSP custom mixer as given in [28].

For VQE, they use the Hardware Efficient Ansatz, which can be represented as:

$$U(\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \prod_{j} CNOT_{j,j+1}(\gamma_{j+1}) \cdot \bigotimes_{j} R_Z(\beta_j)_j \cdot \bigotimes_{j} R_X(\alpha_j)_j,$$

where

$$CNOT_{j,j+1}(\gamma_{j+1}) = |0\rangle\langle 0|_j \otimes I_{j+1} + |1\rangle\langle 1|_j \otimes R_X(\gamma_{j+1})_{j+1}.$$

The subscripts indicate the qubits upon which these transformations are applied. The rotations around the X and Z axes introduce variational parameters for optimization, while parameterized CNOT gates linearly entangle subsequent qubits.
The study includes an extensive exploration of hyperparameters, encompassing the choice of the classical optimizer and the penalty factors in the Quadratic Unconstrained Binary Optimization (QUBO) formulation.
The authors' findings reveal that VQE generally outperforms QAOA, even when different classical optimizers are considered. *Powell* [21] and *NFT* [44] optimizers are particularly effective on the considered problem instances. Let us briefly revisit the working principles of these methods:

- The Powell method, also known as Powell's conjugate direction method, is an iterative and gradient-free optimization approach. The algorithm commences with a specified initial point and a set of predetermined search directions. The optimization process involves minimizing the objective function through a bidirectional line search along each search direction, sequentially. The computation of the next iterate entails moving from the current point along a linear combination of search directions. The coefficients in this linear combination are the optimal step-sizes determined according to the previous bidirectional line searches. At the end of each iteration, the set of search directions is updated based on this new linear combination and the iterative process starts again.

- NFT, or Nakanishi-Fujii-Todo algorithm, represents a sequential minimal optimization method designed specifically for Variational Quantum Algorithms (VQAs). This technique breaks down the optimization problem into solvable subproblems by focusing on a subset of parameters within the parameterized quantum circuit. Choosing a single parameter simplifies the cost function into a periodic sine curve with a period of 2π, allowing for exact minimization of the chosen parameter. Through iterative application of this approach, the parameterized quantum circuits can be optimized to minimize the overall cost function.

Interestingly, the study underscores a trend observed in other research and during our own testing phase as well: the critical challenge in the two-phase heuristic lies in solving the Multi Knapsack Problem. While TSP on small instances can be efficiently solved to optimality by current quantum algorithms, the Multi Knapsack Problem presents a more complex energy landscape that quantum algorithms struggle to navigate successfully. Notably, the authors found that the energy landscape under VQE appeared to be smoother than that under QAOA for their tests. This observation aligns with the results we obtained in our own numerical experiments.

QVRP [45]

In this study, the authors employ simulated Variational Quantum Eigensolver (VQE) to address Vehicle Routing Problem (VRP) instances with 3 and 4 cities. They introduce binary variables $x_{ij} \in \{0,1\}$ for $(i,j) \in E(G)$ to represent whether arc (i,j) is included in a route.
The primary goal of this investigation is to assess the resilience of VQE in the face of various noisy quantum channels. The outcomes reveal that the performance of VQE is significantly influenced by the particular noise model utilized. Generally, noise has a negative impact, although the degree of this impact varies depending on the source of noise. In numerous cases, noise has minimal consequences during the initial stages of the quantum circuits. However, as additional layers are introduced, performance rapidly declines, in contrast to noise-free scenarios, primarily due to the more pronounced and adverse influence of noise.
Their empirical results indicate that the *COBYLA* [22] optimizer outperforms other available optimizers in Qiskit when applied to VQE circuits.
COBYLA (Constrained Optimization by Linear Approximation) is an iterative, derivative-free algorithm designed for nonlinearly constrained optimization. In each iteration, linear approximations to the objective and constraint functions are formed through interpolation at the vertices of a simplex. A trust region bound constrains each change to the variables. Consequently, a new vector of variables is calculated, potentially replacing one of the current vertices. This replacement aims to enhance the shape of the simplex. The trust region radius is never increased and is reduced when well-conditioned simplex approximations fail to yield considerable improvements. This reduction continues until the trust region radius reaches a specified value controlling the final accuracy.

QVRP [46]

In this study, the authors employ the Quantum Approximate Optimization Algorithm (QAOA) to address instances of the *Heterogeneous Vehicle Routing Problem (HVRP)* with varying sizes, such as $(n,K) = (3,1),(3,2),(4,1)$. These instances involve different types of vehicles, each with distinct capacities. The problem is divided into two components: a routing problem (TSP) and a capacity problem (Multi Knapsack Problem). Variables, denoted as $y^v_{i\alpha}$, are used to indicate whether vehicle v visits node i at position α in its route, while z^v_k is employed to represent whether vehicle v has capacity k out of its total available capacity Q_v. To mitigate the number of variables in the Multi Knapsack Problem, a logarithmic encoding is applied.
Unlike classical two-phase heuristics, where the Multi-Knapsack Problem (MKP) and TSP are solved sequentially, the authors take a different approach. They model the two problems separately but then combine them to formulate a final joint Problem Hamiltonian. This Hamiltonian is constructed as the sum of the routing Hamiltonian (derived

from the TSP) and the Multi Knapsack Hamiltonian (stemming from the MKP). The total number of qubits utilized is given by $\#q = N_0^2 \cdot V + \sum_{v=1}^{V} \lfloor \log_2 Q_v \rfloor + 1$.
The authors investigate the algorithm's performance with regard to both the classical optimizer and the depth p of the quantum circuit. They find that *Basin-Hopping with BFGS Local Search* [47] and *Differential-Evolution* [48] optimizers yield satisfactory results, albeit with longer optimization times. Nonetheless, the final outcomes are not competitive with classical approaches. In general, solving routing problems with additional capacity constraints poses a significant challenge for Variational Quantum Algorithms (VQAs). Once again, the primary obstacle appears to be the intricate nature of capacity constraints. The energy landscape of the Multi Knapsack Problem (MKP) exhibits numerous scattered local minima. Before proceeding, let us recall that:

- Basin-hopping is a global optimization technique that operates through iterative steps involving random perturbation of coordinates, local optimization, and acceptance or rejection of new coordinates based on a minimized function value. The local optimization step in this case employs a BFGS Local Search [47], where the descent direction is determined by preconditioning the gradient with curvature information. This is achieved by progressively enhancing an approximation to the Hessian matrix of the objective function, derived solely from gradient evaluations (or approximate gradient evaluations).
- Differential Evolution (DE) is a type of Genetic Algorithm characterized by a unique *mutation* process. In DE, the *mutation* of an agent involves adding the weighted difference between two other population vectors. The parameters of the mutated vector are then mixed with the parameters of a predefined vector known as the target vector, resulting in the creation of a trial vector (*crossover*). If the trial vector produces a lower cost function value than the target vector, the trial vector replaces the target vector in the subsequent generation (*selection*).

QVRP [49]

In this study, the Quantum Approximate Optimization Algorithm (QAOA) is applied to address instances of the Vehicle Routing Problem (VRP) with varying sizes, specifically $(n, K) = (4, 2), (5, 2), (5, 3)$. Notably, these instances do not incorporate any capacity constraints. The binary variables, denoted as x_{ij}, are utilized to indicate whether an arc (i, j) is included in any of the routes. The algorithm is tested with various depths p, primarily using the *COBYLA* optimizer.
For instances where $(n, K) = (4, 2)$ and $(5, 3)$, the optimal solution is found with a high probability. However, for the instance with size $(5, 2)$, the optimal solution is not successfully identified.

QVRP [50]

In this research, a Quantum Unconstrained Binary Optimization (QUBO) formulation is introduced to address the *Time-Scheduled multiple Capacitated Vehicle Routing Problem (TS-mCVRP)*. The variables, denoted as $x^{(i)}_{\tau a,|,c_1,\ldots,c_M}$, are used to indicate whether vehicle i visits node a at time τ, with capacities $c_1,\ldots,c_M$. Time discretization is applied to make τ a finite integer variable. The multiple capacities can also be viewed as potential *states* in which the vehicle operates. The QUBO formulation is tested using the **D-WAVE 2000Q** quantum computer, which has 2048 qubits.
To facilitate this quantum computation, minor embedding is performed utilizing a built-in library, and chains are broken according to the minimum energy strategy. The instance size chosen for experimentation is $(6, 3, 2h, 15m)$, which corresponds to 6 customers, 3 vehicles, a 2-hour time frame, with time-scheduling units set at 15 minutes.

QVRP [51]

The authors of this study tackle the Capacitated Vehicle Routing Problem (CVRP) using a 2-Phase-Heuristic approach, consisting of clustering and routing phases. They break down the CVRP into two separate Quantum Unconstrained Binary Optimization (QUBO) problems and solve them sequentially.
A novel approach is proposed, where clustering is performed using a classical algorithm, and routing is tackled via Quantum Annealing. Notably, clustering is not solved by the Multi Knapsack problem (MKP). The authors employ the *QBSolv* tool provided by D-WAVE, which is designed to map large QUBO problems onto physical quantum hardware. QBSolv divides the QUBO into smaller components, which are solved independently in an iterative manner as long as improvements are observed.
This division of the QUBO matrix into components is carried out using a classical Tabu Search heuristic (4.1.2) in each iteration. In addition to splitting and embedding the QUBO into subQUBOs, *QBSolv* also manages the unembedding and merging of solutions from subproblems.
The core idea revolves around solving the clustering phase without using MKP. The clustering phase is further subdivided into two steps:

1. **Cluster Generation Algorithm**

    - Choose the core stop of a cluster, i.e. the first customer in a cluster, based on maximum demand or largest distance to the depot.
    - Compute the geometric center of the cluster.
    - From the set of unclustered customers, select the customer with the smallest distance to the cluster center and add it to the cluster.

- Recompute cluster center and repeat previous steps until the demand of a customer to be added would exceed the vehicle's capacity.
- If this is the case, a new core stop is selected and the still unclustered customers are assigned to the new cluster.
- This procedure stops when each customer has been assigned to a cluster.

2. **Cluster Improvement**

   - Assign a customer v_i belonging to cluster C_k to another cluster C_j, if that would reduce the distance to the cluster center.
   - **Beware:** assigning a customer to a new cluster must not violate the capacity constraint.
   - If the reassignment is possible and valid, clusters' centers are recalculated and the improvement process begins again.
   - The improvement step terminates if it is not possible to assign a customer to another cluster or when a certain stop criterion is reached (e.g., number of iterations).

The proposed approach is evaluated on various CVRP benchmark datasets. While it doesn't achieve best known solutions for such instances, it does produce good approximations. However, it falls short of being competitive with certain classical heuristics.

7.2 Methodology

In our extensive testing, we assessed several algorithms: Quantum Annealing, Quantum Graver Augmentation, the Variational Quantum Eigensolver (VQE), and Quantum Approximate Optimization Algorithm (QAOA). For VQAs, we explored different circuit depths, denoted as p.
To enhance the Quantum Graver Augmentation method, we developed additional procedures aimed at transforming nearly optimal solutions into truly optimal ones. These procedures basically linearly combine integer solutions x such that $\|Ax\|_1$ is approximately zero, so to get new integer solutions y such that $\|Ay\| = 0$, with A being the constraint matrix. By employing these classical routines, we significantly increase the number of test set elements obtained in a single interaction with the quantum annealer. Additionally, for both VQE and QAOA, we introduced supplementary routines to facilitate the warm start of the parameter optimization process. Further details about these

routines will be provided shortly. For VQE and QAOA we rigorously evaluated various optimizers, including *ADAM, AQGD, CG, COBYLA, L-BFGS-B, Gradient Descent, Nelder-Mead, NFT, P-BFGS, Powell, SLSQP, SPSA, QNSPSA, TNC* and *UMDA*, all already available in the **Qiskit** library.
In the upcoming sections, we will see the best-performing algorithms to be: *Powell (Powell's method)* [21], *COBYLA* [22], and *UMDA*. We have already provided an intuitive description for both Powell and COBYLA; let us now do the same for UMDA.
UMDA (Univariate Marginal Distribution Algorithm) is a specific type of *Estimation of Distribution Algorithm (EDA)*. EDAs are stochastic search algorithms belonging to the family of *Evolutionary Algorithms*. Unlike classic genetic algorithms that maintain a population of solutions, EDAs maintain a probabilistic model from which the population is sampled. This probabilistic model is updated in each iteration using the best individuals from previous generations. UMDA is a univariate EDA, where new individuals are sampled from univariate normal distributions.
Before proceeding, let us also revisit how the *SPSA* [52] and *TNC* [53] algorithms work, as these two also demonstrated notable performance, although not the best in our study.

- SPSA (Simultaneous Perturbation Stochastic Approximation) is akin to gradient-based methods like gradient descent. What sets SPSA apart is its stochastic approximation to the gradient, employing a finite-difference approach. In this method, both the right and left derivatives are approximated through random perturbations, introducing an element of randomness that enhances the algorithm's robustness to noise.

- TNC refers to Qiskit's implementation of the *Truncated Newton* method. Truncated-Newton methods are a family of techniques designed for tackling large nonlinear optimization problems. In each iteration, the current solution estimate is updated by approximately solving the Newton equations through an iterative algorithm. This gives rise to a doubly iterative method: an outer iteration for the nonlinear optimization problem and an inner iteration for the Newton equations. The inner iteration is usually halted or *truncated* before reaching the solution to the Newton equations.

7.2.1 First Approach: Full 2-Index Formulation

Our experimental journey begins by delving into the full two-index formulation of the Capacitated Vehicle Routing Problem (CVRP). To effectively address capacity and subtour elimination constraints, we employ the MTZ (Miller-Tucker-Zemlin) strategy, a notable choice due to its inclusion of a manageable polynomial number of inequalities,

in contrast to the exponentially increasing count found in the original formulation. To establish classical benchmark results, we initially tackle this formulation utilizing the Gurobi solver. Following this classical benchmarking, we transform the problem into its Quadratic Unconstrained Binary Optimization (QUBO) equivalent and proceed to solve it leveraging both Quantum Annealing and Quantum Graver Augmentation techniques. Because of the substantial number of variables involved in this formulation, gate-based quantum algorithms become computationally impractical. Consequently, we also explore an alternative strategy already mentioned earlier: the two-phase heuristic approach *Cluster-First, Route-Second.*

7.2.2 Two Phase Heuristic: Clustering via MKP

To tackle the clustering phase, we embraced three distinct strategies.
Firstly, in line with established practices in the literature, we approached the clustering phase by employing a Multi Knapsack approach (MKP). As before we utilize the Gurobi solver to derive classical benchmark results. Subsequently, we convert the problem to QUBO and explor the application of the several quantum optimization algorithms introduced before.
For VQE, our chosen approach hinges on an Hardware Efficient Ansatz. This ansatz comprises parameterized Y-rotation gates and parameterized controlled-Y-rotation gates. The Y-rotation gates introduce parameters, allowing the classical optimization algorithm control over individual qubits. Conversely, the parameterized controlled gates facilitate the linear entanglement of consecutive qubits, rendering them topologically dependent on one another. This design enables the variation of one qubit to influence the remaining ones. We apply this identical structure for a specified number p of layers. Below we report the circuit for 3 qubits and 3 layers.

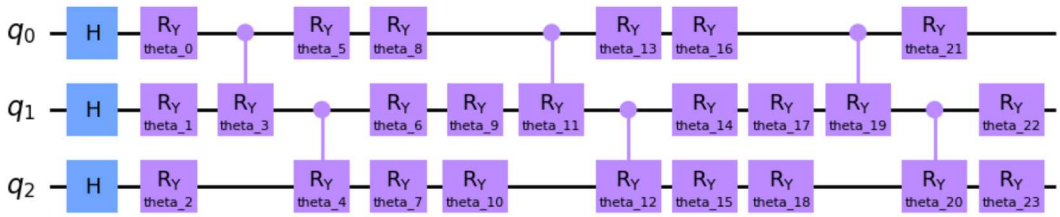


Figure 7.1: VQE Hardware Efficient Ansatz for MKP

As we discussed earlier, empirical findings emphasized the substantial advantages of warm start routines for Variational Quantum Algorithms (VQAs), particularly as the circuit depth p increases. To address this, we devised a warm start strategy. Specif-

ically, for a given circuit depth $(p+1)$, the initial set of parameters is determined by considering the optimal parameters obtained from the preceding circuit depth p and by initializing the remaining ones uniformly at random within the interval $[-2\pi, 2\pi]$.
Even for the Quantum Approximate Optimization Algorithm (QAOA), we adopted diverse strategies by testing a range of depths p. Once again, we leveraged a warm start approach to initialize the parameters for depth $(p+1)$. In this case, the initialization is carried out using the *INTERP* strategy, as outlined in (5.11). This approach involves obtaining the starting parameters for depth $(p+1)$ by linearly interpolating the optimal parameters obtained for depth p.

7.2.3 Two Phase Heuristic: Clustering via Louvain Algorithm

The alternative clustering strategy we employ revolves around the utilization of **Louvain Communities**. In contrast to the traditional approach of finding clusters using the Multi Knapsack Problem (MKP), we sought to address the clustering problem by leveraging the **Louvain Modularity Maximization Algorithm**. This innovative approach stems from the recognition, as documented in the existing literature, that solving the MKP poses considerable challenges for Variational Quantum Algorithms (VQAs). This difficulty arises from the highly unstable energy landscape riddled with numerous local minima.
To facilitate this novel approach, we developed a customized version of the Louvain Modularity Maximization Algorithm. This adaptation was designed to accommodate not only capacity constraints but also limitations on the number of clusters (equating the number of vehicles in our context) and depot-reachability. Our tailored Louvain algorithm is reported in (7.1). With the exception of the two additional concluding subroutines, the sole difference from the classical Louvain algorithm lies in the capacity check performed before merging communities.
The subroutine for reducing the number of communities is reported in (7.2). After reducing the number of communities, an additional crucial step is ensuring that each cluster is reachable from the depot. To address this, we have introduced another greedy routine, which is reported in (7.3).
Please note that the pseudocodes presented here are simplified versions of the original algorithms, which can be accessed in their entirety here [40]. The full algorithms are somewhat more complex and have been optimized further. Nevertheless, these pseudocodes are sufficient to outline the fundamental ideas behind these algorithms.
We employed the Louvain communities both as clusters for the routing phase and as a basis for warm-starting the Quantum Approximate Optimization Algorithm (QAOA) in the Multi Knapsack Problem. In the latter approach, we constructed an MKP solution c^* from the Louvain communities and converted this solution into QAOA initial angles

Algorithm 7.1 Customized Louvain Modularity Maximization

Input: CVRP graph G, number of vehicles K, vehicles capacity C, customers' demands $\{d_i\}_{i=1,\dots,n}$.
Output: Clustering of nodes so that each cluster's total demand does not exceed C and each cluster is reachable from the depot node.
Initialize: Each node i has its own community $\{i\}$.
Compute modularity.
while modularity increases:
    Permute the nodes.
    for i in permuted nodes:
        for $j \in \{v \in V(G) \mid A_{iv} > 0\} \cup \{i\}$:
            Merge i's community with j's community.
            Record modularity for the new partitioning.
            Check capacity constraints on the final community (i.e. the sum of the capacities of its nodes must not exceed C).
            Undo the merge.
        end for
        Merge i's community with j's community so to produce the largest possible increase in modularity while also respecting the capacity constraint.
        Update modularity.
    end for
    Collapse the communities into nodes and start again on the *Condensed Graph*.
end while
Apply greedy subroutine to reduce the number of communities down to K (if needed).
Apply greedy subroutine to connect each community to the depot (if needed).
return final set of communities.

Algorithm 7.2 Reduce Number of Communities to K

Input: CVRP graph G, communities $\{Comm \mid Comm \in Communities\}$, vehicles capacity C, number of vehicles K.
Output: New set of $Communities$ with $|Communities| \leq K$ and $\sum_{node \in Comm} d_{node} \leq C$ for each $Comm \in Communities$.
while number of communities is grater than K:
    Select the least connected community LCC (community with lowest total weighted degree).
    Select the least connected node lcn (node with lowest weighted degree) in the least connected community LCC.
    for each community $Comm \in Communities \setminus \{LCC\}$:
        if $Comm$ presents at least one $node$ such that $A_{lcn,node} > 0$:
            Move lcn to $Comm$.
            Record new modularity.
            Check capacity constraints on $Comm$ is still satisfied.
            Undo the move.
        end if
    end for
    Move lcn to $Comm$ so to produce a minimal decrease in overall modularity while also ensuring that communities' capacities do not exceed C.
end while
return final set of communities.

Algorithm 7.3 Connect Each Community to Depot

Input: CVRP graph G, communities $\{Comm \mid Comm \in Communities\}$, vehicles capacity C.
Output: New set of $Communities$ such that each $Comm \in Communities$ is connected to the depot node.
Count the number of connection to the depot for each $Comm \in Communities$.
if every community $Comm$ has at least one node connected to the depot:
    return $Communities$
end if
while there exists a $Comm$ with no connection to the depot:
    Randomly pick a community i not connected to the depot.
    Select the least conncected node $lcn(i) \in i$ (node with lowest weighted degree).
    for each community $j \in Communities \setminus \{i\}$ which have at least two nodes connected to the depot:
        From j select $dlcn(j)$, the depot-connected node within j with lowest connectivity within j (depot-connected node with lowest weighted degree).
        if $lcn(i)$ has at least one neighbor in j and $dlcn(j)$ has at least one neighbor in i:
            Move $lcn(i) \to j$ and $dlcn(j) \to i$
            Record the new modularity.
            Check the capacity constraints on communities i and j
            Undo the move.
        end if
    end for
    Move $lcn(i) \to j$ and $dlcn(j) \to i$ so to produce the minimal decrease in overall modularity while also ensuring capacity constraints on the final set of communities.
    For each community count the number of connection to the depot
end while
return final set of communities

using the following formula:

$$\theta_i = 2\arcsin\left(\sqrt{c_i^*}\right), \tag{7.1}$$

we then used such angles to build the initial state

$$|\psi_0\rangle = \bigotimes_{i=0}^{n-1} R_Y(\theta_i)|0\rangle^{\otimes n}, \tag{7.2}$$

with R_Y being parameterized Y-rotation gates.
When warm-starting QAOA we must ensure that the *mixing operator* has the initial state $|\psi_0\rangle$ as its ground state. We therefore chose the Hamiltonian

$$H_{M,i}^{(ws)} = \begin{pmatrix} 2c_i^* - 1 & -2\sqrt{c_i^*(1-c_i^*)} \\ -2\sqrt{c_i^*(1-c_i^*)} & 1 - 2c_i^* \end{pmatrix} \tag{7.3}$$

as *mixing operator* for qubit i.
For more detailed information regarding warm-start QAOA, please refer to the original paper [54].

7.2.4 Two Phase Heuristic: Clustering via Modularity Maximization

Our latest clustering approach once again revolves around Modularity Maximization. In this case, we model Modularity Maximization on a directed graph, framing it as a combinatorial optimization problem that we tackle using our quantum optimization algorithms.
To quantify Modularity in an undirected graph $G = (V, A)$, we have the expression:

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta_{c_i, c_j}. \tag{7.4}$$

Here, m represents the total number of edges in the graph; A_{ij} is the weight of arc $(i, j) \in A$; k_i denotes the degree of node i in the graph, and δ_{c_i, c_j} is an indicator function. This function equals 1 if the community of node i, denoted as c_i, is the same as the community of node j, i.e., c_j, and it equals 0 otherwise. This formulation of modularity

is extendable to directed graphs by considering:

$$Q = \frac{1}{m} \sum_{i,j} \left(A_{ij} - \frac{k_i^{\text{in}} k_j^{\text{out}}}{m} \right) \delta_{c_i, c_j}. \tag{7.5}$$

In this case, k_i^{in} and k_j^{out} represent the in-degree of node i and the out-degree of node j in the graph, respectively.
Modularity within a graph quantifies how different the actual links in the network are compared to what we would expect from randomly rewiring the edges. The term $(k_i k_j)/(2m)$ can be seen as the probability of randomly connecting nodes i and j in the graph. When a set of nodes exhibits high modularity, it implies that the edges within the group are more likely than random, indicating a potential community structure. In problems involving Modularity Maximization, our goal is to partition the nodes into communities so to maximize the graph's overall modularity.
The next question now is, how to reformulate equation (7.5) as a QUBO (Quadratic Unconstrained Binary Optimization) problem suitable for our quantum algorithms? Luckily, this just requires a straightforward adaptation:

$$\max \quad \frac{1}{m} \sum_{v} \sum_{i,j} \left(A_{ij} - \frac{k_i^{in} k_j^{out}}{m} \right) z_{v,i} z_{v,j} \tag{7.6}$$

$$\text{s.t.} \quad \sum_{v} z_{v,i} = 1 \quad \forall i = 0, \ldots, n-1 \tag{7.7}$$

$$z_{v,i} \in \{0,1\} \quad \forall v = 1, \ldots, k;\ i = 0, \ldots, n-1. \tag{7.8}$$

Here, the binary variables $z_{v,i}$ indicate whether node i is assigned to cluster v or not.
It is worth noting that the constraint on the number of clusters is intrinsically satisfied since $v = 1, \ldots, k$. However, this alone doesn't guarantee the feasibility of the clusters. We still need to ensure that capacity constraints are met and that each cluster is directly connected to the depot node. To assess the first condition, we implemented a classical greedy routine similar to what's presented in algorithm (7.3). This new routine can be found in (7.4). On the other hand, to ensure depot connectivity we once again rely on (7.3). As mentioned previously, note that the pseudocode provided in (7.4) is a simplified and less optimized representation of the original algorithm, which can be found in [40].

7.2.5 Two Phase Heuristic: Routing via TSP

With the clusters established, we now proceed to the routing phase. In this phase, we model the problem using the classical Traveling Salesman Problem (TSP), as described in the previous chapter. As previously discussed, to account for subtours elimination

Algorithm 7.4 Check Capacity Constraints on Communities

Input: CVRP graph G, communities $\{Comm \mid Comm \in Communities\}$, vehicles capacity C, customers' demands $\{d_i\}_{i=1,\dots,n}$.
Output: New set of $Communities$ such that each $Comm \in Communities$ satisfies the capacity constraint.
Compute the capacity $C(Comm)$ for each $Comm \in Communities$.
if every community $Comm$ satisfies capacity constraints:
    return $Communities$
end if
while there exists a $Comm$ which does not satisfy capacity constraint:
    Randomly pick a community i not satisfying capacity constraint.
    Select the least conncected node $lcn(i) \in i$ (node with lowest weighted degree).
    for each community $j \in Communities \setminus \{i\}$ with $C(j) < C$:
        if $lcn(i)$ has at least one neighbor in j and $C(j) + d(lcn(i)) \leq C$:
            Move $lcn(i) \to j$
            Record the new modularity
            Check the capacity constraint on community j
            Undo the move.
        end if
    end for
    Move $lcn(i) \to j$ so to produce the minimal decrease in overall modularity while also ensuring capacity constraints on the final set of communities.
    Compute the capacity $C(Comm)$ for each $Comm \in Communities$
end while
return final set of communities

while minimizing the number of variables, we employ a recursive Dantzig-Fulkerson-Johnson (DFJ) strategy.
Once again, we conducted tests using Quantum Annealing, Quantum Graver Augmentation, Variational Quantum Eigensolver (VQE), and Quantum Approximate Optimization Algorithm (QAOA). Classical solutions for comparison are obtained using Gurobi.
For VQE, due to the problem's simplified structure, we chose to employ a simpler Hardware Efficient Ansatz, outlined below (3 qubits, 3 layers):

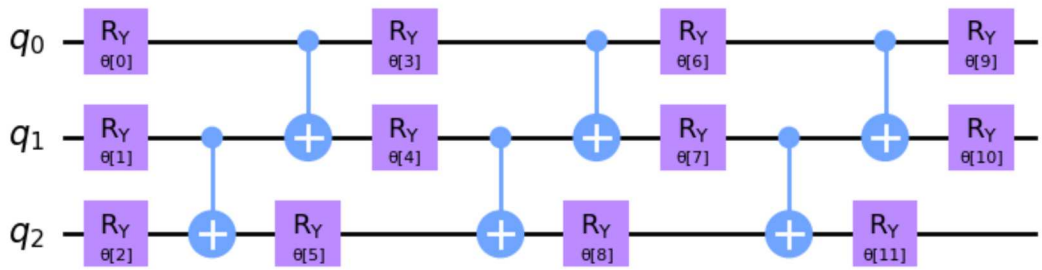


Figure 7.2: VQE Hardware Efficient Ansatz for TSP

In this case, notice that for qubit entanglement we are employing simple *CNOT* gates, which do not introduce any additional parameters.

7.3 Experimental Design

In this section, we put our theoretical discussions into practical application. Here, we provide an overview of our numerical experiments and the design decisions we made. Certain design choices were influenced by resource limitations, and we will delve into the reasoning behind these decisions shortly. This section exclusively focuses on the methodology and process of conducting the experiment, reserving the presentation and analysis of results for the subsequent final section.

Experimental Setup

To evaluate quantum annealing-based algorithms, we harnessed the capabilities of the *D-Wave quantum annealer*, specifically the **Advantage System 6.3**. This system is constructed upon the **Pegasus Hardware Graph**, featuring 5614 physical qubits arranged in a grid with dimensions of $15 \times 15 \times 12$. It possesses a degree-15 connectivity, meaning that each qubit is intricately linked to 15 other qubits. As we are aware, minor embedding plays a pivotal role in achieving commendable performance via Quantum

Annealing (QA). For our experiments, we utilized the pre-built minor embedding strategy provided by D-Wave. This choice aligns with findings in the existing literature, as discovering a proficient embedding can be as intricate as solving the original combinatorial optimization problem. Fortunately, the built-in embedding strategy typically yields favorable results.
On the other hand, to assess simulated Variational Quantum Algorithms (VQAs), we employed the **Statevector Simulator** available in the *Qiskit* library. This simulator is characterized by its ideal, noise-free nature. We opted for this noise-free simulator to simplify the scenario and gauge the performance of VQAs under ideal conditions. As we will observe, even in these ideal conditions, VQAs encounter challenges on certain instances. In this context, running VQAs on real quantum architectures may introduce additional issues and potentially result in poorer performances. Moreover, the study outlined in [45] reveals that VQA performance can exhibit significant variation depending on the type of noise injected into the model, with certain types of noise further deteriorating performance. This supported our decision to assess VQAs in an ideal, noise-free environment, where one would intuitively expect the algorithms to perform optimally. It is crucial to reiterate that we did not have access to a real gate-based quantum model. Therefore, the algorithms were executed in a simulated environment, which can yield performance results that may vary considerably from those on actual quantum hardware.
The code used for our experiments, provided in [40], was entirely developed using the *Google Colab* platform. We utilized the free version of Colab, which provides a maximum of 12GB RAM. This resource limitation posed challenges for some VQA instances, where the extensive memory requirements led to runtime failures. This happened especially when unequal customers'demands were tested, which is the reason we then decided to keep all demands the same.

Replicability and Validity

In order to guarantee the replicability and validity of our experiments, we generate an initial seed at the start of our Python notebook. All the seeds are recorded within the final dataframe that we utilize for our analyses. This ensures that the results we present can be reliably replicated. It is important to note, however, that quantum computers themselves are inherently stochastic devices, meaning that re-executing the algorithms may yield different outcomes, even when employing the same fixed seed.

Selection of Test Instances

Our test dataset comprises 25 randomly generated instances. To create a CVRP graph, we utilize a random matrix, denoted as E, in which each entry e_{ij} is drawn from a uniform distribution within the range of $[1, 10]$. These entries signify the costs incurred by

the vehicles while traversing the arcs from node i to node j. It is important to note that graphs generated in this manner are always complete, meaning that each node is connected to every other node. This completeness was a necessary simplification, essential for our two-phase heuristic approach to be effective.
Indeed, solving the CVRP problems using the full formulation turned out to be impractical both for Quantum Annealears and gate-based quantum models. Quantum Annealing and Quantum GAMA faced significant challenges in providing feasible solutions consistently across different runs. These issues likely stem from factors such as embedding complexities, chain-breaking strategies, and constraints associated with the limited annealing schedule. On the other hand, the full formulation is just too big for current gate-based models, which cannot handle such high number of variables. These observations forced us to rely on the two-phase heuristic approach.
The first phase involves clustering of nodes. We employed three distinct methods for clustering: the Multiple Knapsack Problem (MKP), Modularity Maximization (MM), and the Louvain algorithm. However, it is crucial to understand that none of these methods can guarantee the validity of clusters for the subsequent Traveling Salesman Problem (TSP) phase. Valid clusters must meet specific criteria, including having at least two nodes connected to the depot and a path connecting these two nodes while also visiting all the other nodes within the cluster. These criteria are challenging to satisfy, and refining clusters using classical routines would be excessively time-consuming due to the combinatorial nature of this refinement phase. To circumvent this issue, we restricted our analysis to complete graphs only. Indeed, in a complete graph, any subgraph induced by the clusters remains complete, resolving connectivity problems. This choice aligns with common practices found in the existing literature.
Our test instances are characterized by relatively small dimensions, mostly featuring 5 nodes and 2 vehicles ($n = 5, K = 2$). We also conducted tests using classical TSP instances by setting $K = 1$. However, as we explored larger instances, challenges emerged. Instances of size $(n = 10, K = 1)$ proved too large for simulation in Qiskit. Additionally, AQC and Quantum GAMA struggled in generating feasible solutions, likely due to challenges associated with minor embedding, as the increased number of variables led to more extensive and intricate qubit chains.
Similarly, instances of size $(n = 10, K = 2)$ had their own challenges as well. VQA faced issues in the routing phase as one cluster exceeded the Qiskit simulator's qubit limit. In contrast, AQC and Quantum GAMA provided effective solutions and approximations to Gurobi's optimal results. The main issue though lies in the high resource demand of these instances. Indeed, running an instance of size $(10, 2)$ on the D-Wave quantum annealer consumes resources equivalent to handling about 6 instances of size $(5, 2)$.
For this reasons, considering our resource constraints, we made the choice to focus predominantly on instances of size $(5, 1)$ and $(5, 2)$.
It is worth noting that even though these instances may appear small, they pose chal-

lenges due to the presence of many inequality constraints in their respective formulations. For each inequality, we must introduce integer slack variables, which are subsequently binary encoded. Simulated Variational Quantum Algorithms (VQAs) encounter difficulties in managing larger instances due to these constraints.
Customer demands are assumed to be uniform and unitary (i.e., = 1). Testing instances with uneven demands led to memory-related issues with simulated VQAs since a greater number of slack variables needed to be introduced, increasing the overall problem size. We also tested different vehicle capacities. Initially, we set $C = n$, meaning that each vehicle could potentially visit all customers. We then explored scenarios with $C = \lceil \frac{n}{2} \rceil + 1$, requiring the utilization of both vehicles. The minimum number of vehicles needed to serve all customers is determined according to the trivial lower bound in (6.7).

Algorithms and Tools

In this experiment, we evaluated various algorithms, including Quantum Annealing, Quantum GAMA, Variational Quantum Eigensolver (VQE), and Quantum Approximate Optimization Algorithm (QAOA). For the annealing-based algorithms, we opted for the maximum allowable annealing time as reduced schedules yielded inferior results. In the case of simulated Variational Quantum Algorithms (VQAs), such as VQE and QAOA, we conducted extensive tests with different classical optimizers and warm start strategies, as discussed earlier.
For benchmarking and comparison, we used Gurobi as our classical reference, assessing both the objective function value and execution time. While CVRP instances are generally challenging to solve, in our case the relatively small sizes allowed Gurobi to find optimal solutions consistently.
A crucial aspect of VQAs is the selection of the underlying classical optimizer. Our investigation involved testing various optimizers available within the Qiskit framework. We explored different maximum iteration counts and circuit depths for both VQE and QAOA. Our findings highlighted several key observations:

- Gradient-based methods, particularly in the context of VQE, exhibited slower convergence rates. This is partly attributed to the need for estimating partial derivatives via finite differences, necessitating double access to the quantum hardware. In the case of VQE, this approach was particularly time-consuming, given the substantial number of parameters in the Hardware Efficient Ansatz, approximately $3np$, where n denotes the number of qubits and p represents the circuit depth. On the other hand, gradient methods for QAOA proved relatively faster, due to the considerably smaller number of parameters, which is only $2p$. However, they did not excel in terms of solution quality.

- COBYLA emerged as a high-performing optimizer, demonstrating efficiency in terms of both speed and solution quality. It stood out as one of the fastest algo-

rithms tested and consistently achieved optimal solutions in most runs. Notably, COBYLA adapted well to increased circuit depth, enhancing solution quality. This made it a favorable choice for optimizing both VQE and QAOA.

- The Powell method emerged as the best-performing optimizer among the options considered. It exhibited remarkable performance for both VQE and QAOA, even with higher circuit depths. Interestingly enough, increased circuit depth did not lead to significant improvements in solution quality, rather just an increase in overall execution time.

- SPSA delivered quite good results, particularly for VQE, although its performance was less impressive for QAOA. In contrast, TNC performed well with QAOA but exhibited suboptimal performance with VQE.

- The UMDA algorithm proved to be among the top-performing options for both VQE and QAOA. While it was slower than some alternatives, it consistently delivered solution quality on par with Powell and COBYLA.

Following extensive testing and evaluation, we decided to employ the Powell optimizer for the clustering phase, with 300 iterations and circuit depths of 2, both for VQE and QAOA. For the routing phase instead, we chose COBYLA, utilizing only 100 iterations and a circuit depth of 1, for both VQE and QAOA. This decision was influenced by the fact that Powell proved to be too slow for the TSP phase, occasionally requiring over 3 hours to complete. The smaller number of iterations and circuit depth did not significantly compromise solution quality, with most results remaining in close proximity to the optimal solutions produced by Gurobi.

Experimental Procedure

As previously discussed, the pivotal connection between combinatorial optimization and quantum computing hinges on the Quadratic Unconstrained Binary Optimization (QUBO)/Ising formulations. This implies that to execute our algorithms, it is imperative to construct the QUBO representation from the original CVRP graph.
A significant challenge in this process revolves around the selection of the penalty term, which governs the scaling of constraints when transitioning from the original constrained optimization problem to its unconstrained counterpart. Given our goal of achieving total feasibility, where each constraint must be rigorously satisfied, we adopted a uniform penalty term, denoted as P, for each squared constraint. It is worth emphasizing that the selection of P remains a non-trivial matter and continues to be an active area of research. After conducting experiments with various penalty values, we ultimately settled on the choice $P = \sum_{ij} e_{ij} + 1$, a selection commonly found in existing literature. Remember that in this context, matrix $E = \{e_{ij}\}_{ij}$ is randomly generated, and

its entries determine the cost associated with traversing the CVRP graph.
Recall also that the QUBO formulation exclusively relies on binary variables. To optimize efficiency and minimize the number of qubits required, we implemented logarithmic encoding for the representation of integer variables as combinations of binary variables. Additional details regarding this encoding method are expounded in the dedicated chapter of this thesis.
To conclude, notice that in algorithms based on modularity maximization, modularity is computed with respect to the matrix $\hat{E} = \{\exp(-e_{ij})\}_{ij}$. Indeed, our goal is to partition nodes so that the total travel distance/cost in each cluster is minimized.

Performance Metrics

To assess and juxtapose the performance of various algorithms, we employ both objective function value and execution time as key evaluation metrics. It is important to note that interpreting the execution times of our quantum algorithms can be somewhat deceptive. This is primarily due to the following factors:

- For Quantum Annealing, our recorded execution time encompasses the entire duration from when we submit the problem to the DWave quantum annealer. This duration includes not only the time spent solving the problem on the quantum computer but also the time it spends in the annealer's queue before processing. The actual time required to solve the problem is instead reflected by the *Quantum Processing Unit* access time.
- In the case of Variational Quantum Algorithms (VQAs), the execution time is notably misleading. These algorithms are simulated classically, and on certain instances, they took up to half an hour to complete. The execution time would have been significantly shorter if we had access to real quantum gate-based models.

Data Collection

Our data collection procedure meticulously records the outcomes of each experiment. This encompasses various data points such as the ultimate objective function values, execution times, the number of variables, embedding details, parameters specific to Variational Quantum Algorithms (VQAs), the quantity of extracted Graver elements, and more. All of this information is methodically stored within a dataframe, laying the groundwork for a comprehensive and in-depth analysis, which we are going to present in the next section.

7.4 Results and Analysis

In this final section, we present and discuss the results obtained from our numerical experiments. To facilitate this discussion and gain deeper insights into our findings, we heavily rely on boxplots. These graphical representations offer a valuable tool for comprehending the distribution of data across various algorithms and different phases of our Two-Phase Heuristic approach. We predominantly employ two distinct types of boxplots to conduct our analysis. The first set of boxplots serves to compare the objective function values achieved by the different algorithms we tested. The second set allows us to assess the execution times of these algorithms.
It is worth noting that the objective function values used for constructing the boxplots are relative to the Gurobi optimal objective values f^*_{Gurobi}. Specifically, for a given algorithm $Algo$, we compute the relative objective function value as

$$\frac{f^*_{Algo} - f^*_{Gurobi}}{f^*_{Gurobi}},$$

with f^*_{Algo} being $Algo$'s optimal objective function value. When this quotient closely approaches zero, it signifies that the algorithm provides a strong approximation to the Gurobi optimum, which we consider as the benchmark result.
We apply a similar strategy to evaluate the execution times. The relative execution time for a given algorithm $Algo$ is computed as

$$\frac{t_{Algo}}{t_{Gurobi}},$$

where t_{Algo}, t_{Gurobi} represent the execution time of algorithm $Algo$ and Gurobi, respectively.
As a result, to mirror the performance of Gurobi, our subsequent boxplots aim for objective function values to be in proximity to 0 and for execution times to be close to 1.
Furthermore, please take note that within the following boxplots, as is customary, the thick black lines signify the median of the distribution. Indeed, the median often serves as a more robust statistical measure than the mean.
To conclude, also recall that our study, owing to the significant resource constraints and specific design choices, is primarily an exploration of quantum methodologies for tackling the Capacitated Vehicle Routing Problem (CVRP). The primary goal is not to directly compete with classical approaches but to explore the potential of quantum techniques in this domain.
We start by analyzing TSP instances of size $(n = 5, K = 1)$, which make up one fifth of the entire dataset.

7.4.1 TSP Instances

We begin our discussion with the following boxplot.

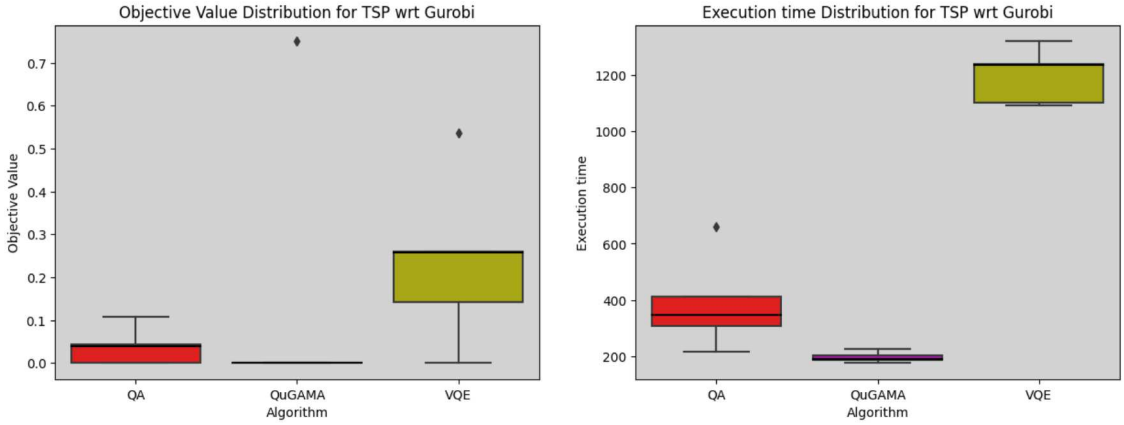


Figure 7.3: Boxplots for TSP instances with dimensions $(5, 1)$. On the left, a boxplot comparing the distribution of optimal objective function values. On the right, a boxplot comparing the execution times of the various algorithms.

The first notable observation is the absence of QAOA in the results. This omission is primarily due to the challenging nature of TSP instances for QAOA, which struggled to provide solutions corresponding to feasible routes.
Examining the left-side plot, it becomes evident that Quantum GAMA emerges as the top-performing algorithm. It consistently achieves Gurobi's optimal objective function value on all but one instance. Quantum Annealing closely follows, offering optimal solutions that are only slightly less optimal than those found by Gurobi. In contrast, VQE stands out as the least effective algorithm, consistently displaying significant gaps in comparison to Gurobi's optimal objectives.
A similar pattern emerges in the execution time boxplot. Quantum GAMA continues to be the top-performing algorithm, followed by Quantum Annealing, and then VQE. However, it is important to note that even for Quantum GAMA, the execution times are approximately 200 times larger than those of Gurobi. Additionally, as previously discussed, these execution times can be somewhat misleading, especially for VQE, which is classically simulated via Qiskit.
In conclusion, we provide a plot displaying the achieved optimal objective function values on the test instances.

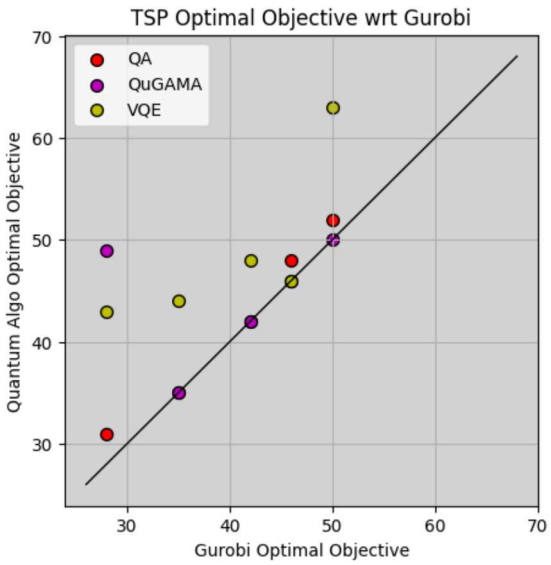


Figure 7.4: Optimal objective function values for TSP instances with dimensions $(5, 1)$. The x-axis represents Gurobi's optimal objective values, while the y-axis depicts the optimal objectives obtained by various quantum algorithms.

7.4.2 CVRP Instances

We now shift our focus to the assessment of actual CVRP instances with dimensions $(n = 5, K = 2)$. This section follows a systematic approach: we initiate the discussion by evaluating the performance of our quantum algorithms in cases where clustering is executed via the Multi-Knapsack Problem method. Subsequently, we delve into an analysis of the outcomes when Modularity Maximization is employed for clustering. Following that, we wrap up our evaluation by scrutinizing the results obtained when Louvain Community Detection is employed for the clustering phase.
Throughout this discussion, we conduct a comparative analysis of the Two-Phase Heuristic approach against the complete CVRP Two-Index formulation. This comparison serves to assess the suitability of the Two-Phase heuristic approach within our quantum framework. As in preceding sections, our analysis is reinforced through the utilization of boxplots and scatterplots.

Clustering via Multi-Knapsack Problem

We initiate our examination with the following set of boxplots associated with the MKP clustering phase. The objective function values and execution times are relative to the MKP solved using Gurobi as a reference.

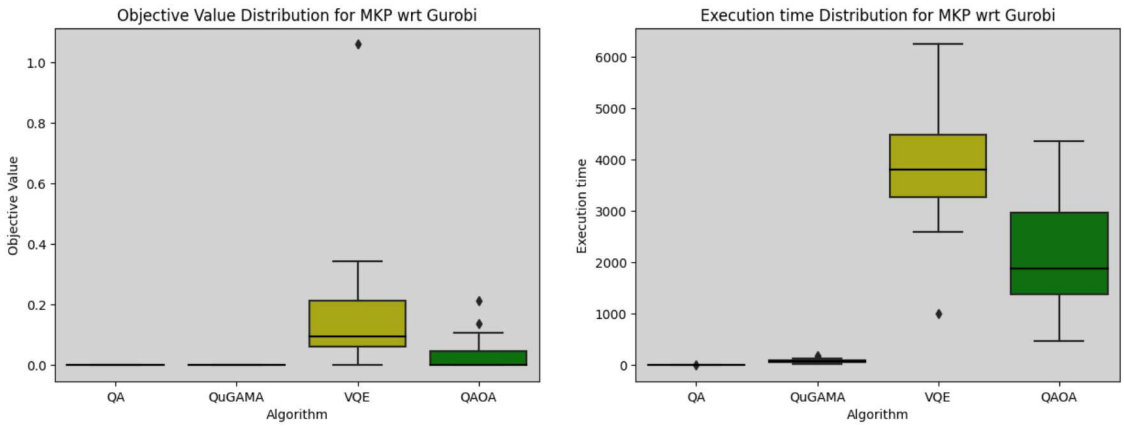


Figure 7.5: Boxplots for MKP clustering of CVRP instances of size $(5, 2)$. On the left, a boxplot comparing the distribution of optimal objective function values. On the right, a boxplot comparing the execution times of the various algorithms.

Similar to previous case, both QA and Quantum GAMA deliver notably strong performances, matching Gurobi's results in terms of objective function values and execution times. Conversely, VQAs exhibit comparatively less favorable results, with QAOA now outperforming VQE, achieving generally better outcomes in both objective function values and execution times. Indeed, on MKP QAOA performs relatively well. It aligns with

Gurobi's optimal objectives for half of the test instances and offers strong approximations for the remaining cases.
Following the MKP clustering phase, our next step is to evaluate the routing phase (TSP) for the identified clusters.

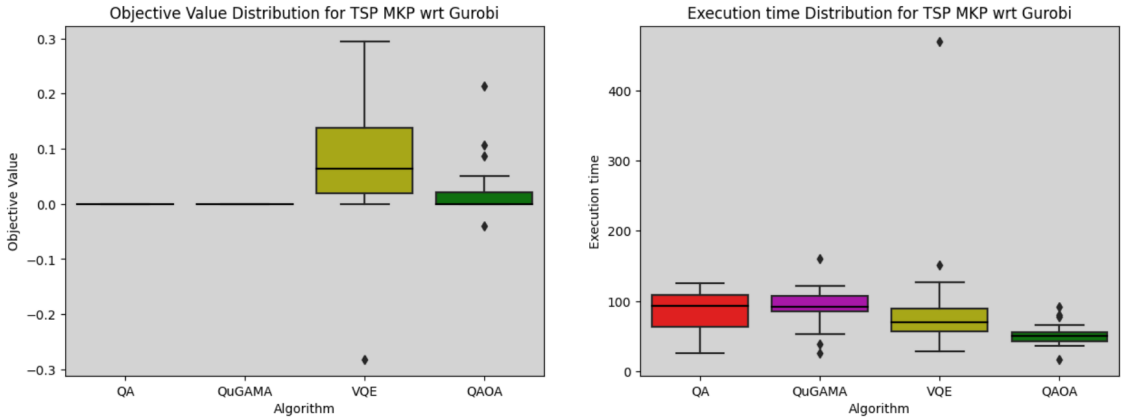


Figure 7.6: Boxplots for TSP after MKP clustering. On the left, a boxplot comparing the distribution of optimal objective function values. On the right, a boxplot comparing the execution times of the various algorithms. On the left plot, we observe the presence of two outliers where VQE and QAOA seem to achieve even better optimal solutions than Gurobi. This could be attributed to the fact that the optimal clusters found by Gurobi during the MKP phase might not be the most optimal for the subsequent TSP phase. In these cases, the suboptimal clusters identified by VQE and QAOA during the MKP phase appear to be more suitable for the ensuing TSP phase.

Once more, Quantum Annealing and Quantum GAMA consistently rank as the top-performing algorithms, matching Gurobi's optimal solutions across all test instances. Notably, VQAs also demonstrate significant performance, with QAOA once again outperforming VQE. Surprisingly, despite being classically simulated, VQAs exhibit execution times that are generally comparable to or even better than QA and QuGAMA, with QAOA being the fastest among them.
As evident from the plot on the left, certain test instances showcase the capability of our quantum algorithms to outperform Gurobi, achieving lower objective function values. This phenomenon is likely attributed to the heuristic nature of our two-phase approach. The optimal MKP clusters identified by Gurobi may not be the most suitable choice for the subsequent routing phase. Interestingly, suboptimal clusters obtained by VQE and QAOA appear to deliver superior performance in the ensuing TSP phase. However, it is crucial to note that in such plot, the reference values are the optimal objective values found by Gurobi when applied via the two-phase heuristic approach. If we were to compare our algorithms with the true optimum as found by Gurobi when solving the full CVRP formulation, then we expect no test instance to exhibit such behavior, and indeed, that is exactly what happens.

Let us now direct our attention to the final set of boxplots. In these plots, objective function values and execution times are measured relative to the optimal objective and execution time achieved by Gurobi when solving the full two-index formulation of CVRP, without utilizing the two-phase heuristic approach. The plotted objective function values represent the sum of TSP optimal objectives on the MKP clusters, while the execution times encompass both clustering and routing times. Notably, these plots also showcase Gurobi's performance when applied through the two-phase heuristic, allowing us to evaluate the validity of this heuristic approach against the complete two-index formulation.

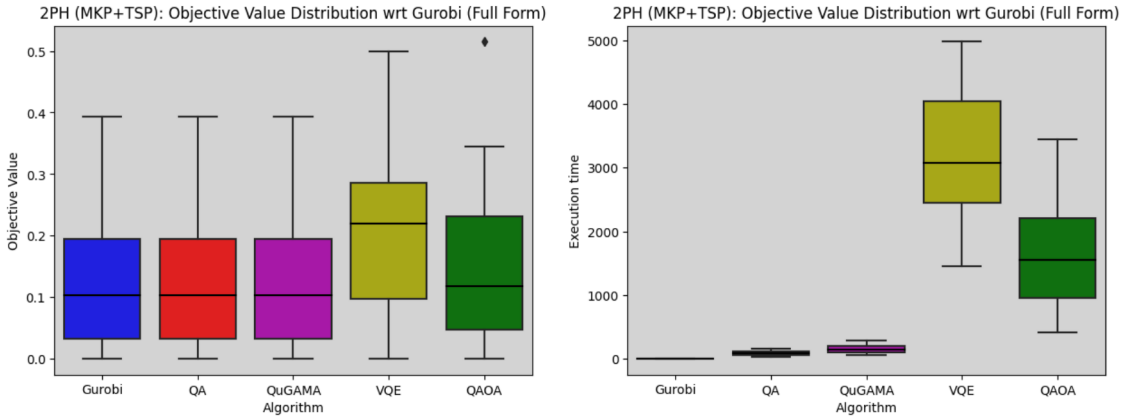


Figure 7.7: Boxplots for the Two Phase Heuristic approach: MKP + TSP. On the left, a boxplot comparing the distribution of optimal objective function values. On the right, a boxplot comparing the execution times of the various algorithms.

Upon inspecting Gurobi's boxplot, our initial observation is that the two-phase heuristic approach appears to fall short of retrieving the optimal solution for the complete CVRP formulation. This outcome aligns with our expectations, considering the heuristic nature of this approach. Nonetheless, the two-phase heuristic still manages to provide commendable approximations to the true optimum.
QA and Quantum GAMA continue to stand out as the best-performing algorithms, matching Gurobi's performance within the context of this heuristic approach, as previously seen in the earlier set of boxplots. Once again, VQAs exhibit relatively less favorable results.
We conclude this subsection by examining the following set of scatter plots, which illustrate the comparison between Gurobi's optimal objectives and those achieved by our quantum algorithms. In the final rightmost plot, Gurobi's objectives are the optimal objectives of the complete two-index CVRP formulation.

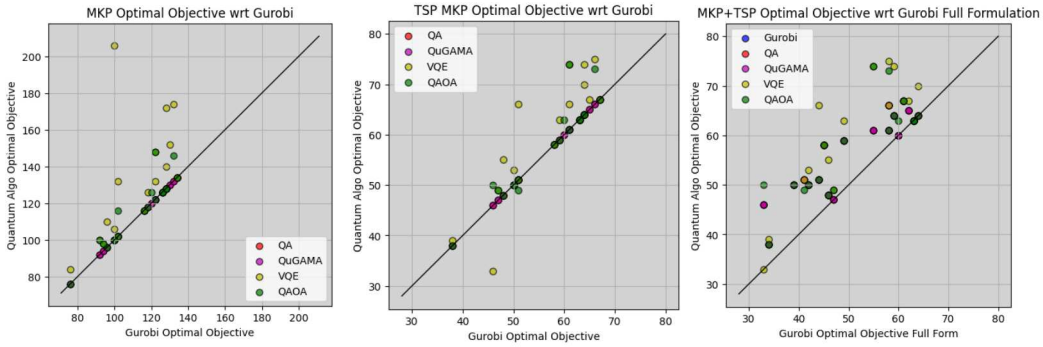


Figure 7.8: *Left:* Optimal objective function values for MKP clustering phase. *Center:* Optimal objective function values for TSP after MKP clustering. *Right:* Optimal objective function values for Two Phase Heuristic approach: MKP + TSP. In each plot, the x-axis represents Gurobi's optimal objective values, while the y-axis depicts the optimal objectives obtained by various quantum algorithms.

Clustering via Modularity Maximization

Once more, we commence by examining the boxplot associated with the Modularity Maximization clustering phase. Objective function values and execution times are, as before, measured relative to MM solved using Gurobi as a reference.

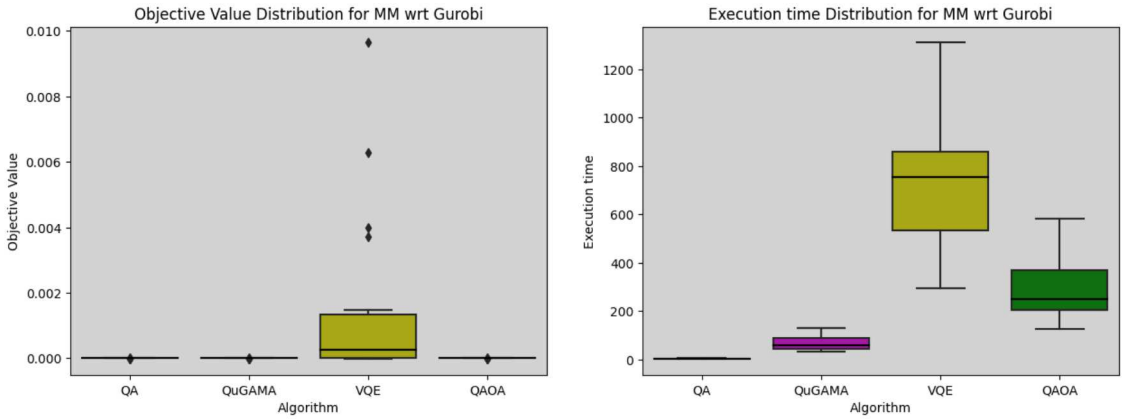


Figure 7.9: Boxplots for MM clustering of CVRP instances of size $(5, 2)$. On the left, a boxplot comparing the distribution of optimal objective function values. On the right, a boxplot comparing the execution times of the various algorithms.

The Modularity Maximization approach, as previously introduced in (7.2.4), is notably simpler in comparison to MKP, particularly due to the absence of capacity constraints inequalities. As evident, even VQAs exhibit commendable performance in this scenario,

with QAOA achieving results on par with Gurobi, Quantum Annealing, and Quantum GAMA. The latter two, once again, emerge as the top-performing algorithms, excelling in both objective function value and execution time. In contrast, VQE delivers slightly inferior results, although it still offers very satisfactory approximations to Gurobi's optimum.
Following the MM clustering phase, our next step is to evaluate the routing phase (TSP) for the identified clusters.

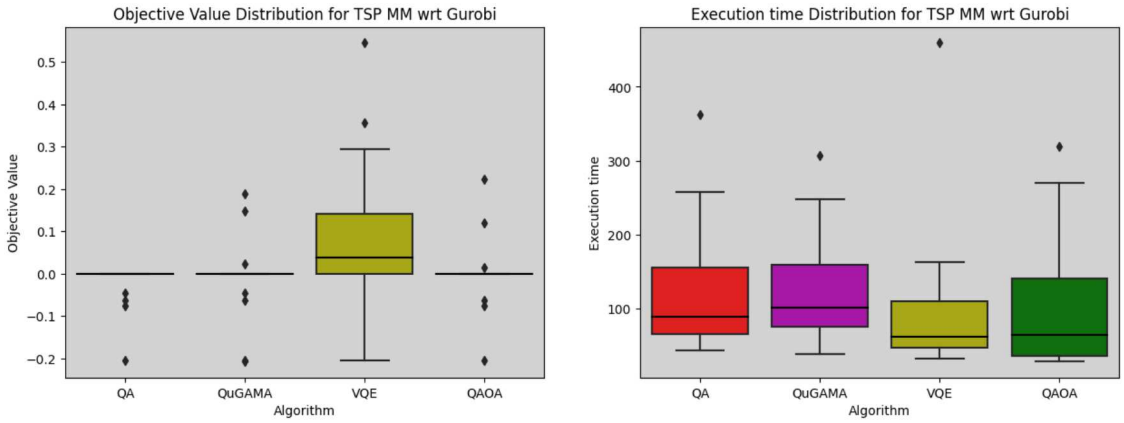


Figure 7.10: Boxplots for TSP after MM clustering. On the left, a boxplot comparing the distribution of optimal objective function values. On the right, a boxplot comparing the execution times of the various algorithms. On the left plot, we observe the presence of many outliers where our quantum algorithms seem to achieve even better optimal solutions than Gurobi. This could be attributed to the fact that the optimal clusters found by Gurobi during the MM phase might not be the most optimal for the subsequent TSP phase. In these cases, the suboptimal clusters identified by our quantum algorithms during the MM phase appear to be more suitable for the ensuing TSP phase.

The TSP on MM clusters appears to pose more significant challenges than before. Although Quantum Annealing, Quantum GAMA, and QAOA continue to perform on par with Gurobi for most instances, we observe a number of outliers, signifying instances where the algorithms encountered heightened difficulty. Once more, VQE emerges as the worst performing algorithm. The struggles faced by these algorithms during the routing phase become even more evident when examining the corresponding execution times. In contrast to the previous MKP case, the execution times are generally longer, even for quantum annealing-based algorithms. It is noteworthy that, in this context, VQE appears to be the fastest algorithm on average, but it is essential to recall that it also exhibits the poorest performance.
Similar to our previous observations, there are several outliers where our quantum algorithms appear to attain better optimal solutions than Gurobi. This occurrence can likely be attributed to the heuristic nature of the two-phase approach, as in some instances, suboptimal clusters may prove to be more suitable for the subsequent routing phase.

Again, it is crucial to note that in the above plot, the reference values are the optimal objective values found by Gurobi when applied via the two-phase heuristic approach. If we were to compare our algorithms with the true optimum as found by Gurobi when solving the full CVRP formulation, then we expect no test instance to exhibit such behavior.
Next, we turn our attention to a final set of boxplots. These boxplots depict objective function values and execution times relative to Gurobi's optimal objective and execution time when solving the complete two-index formulation of CVRP, without employing the two-phase heuristic approach. The plotted objective function values represent the sum of TSP optimal objectives for the MM clusters, while the execution times are the combined clustering and routing execution times. These plots also include Gurobi's performances when applied through the two-phase heuristic, aiding us in assessing the validity of this heuristic approach in comparison to the complete two-index formulation.

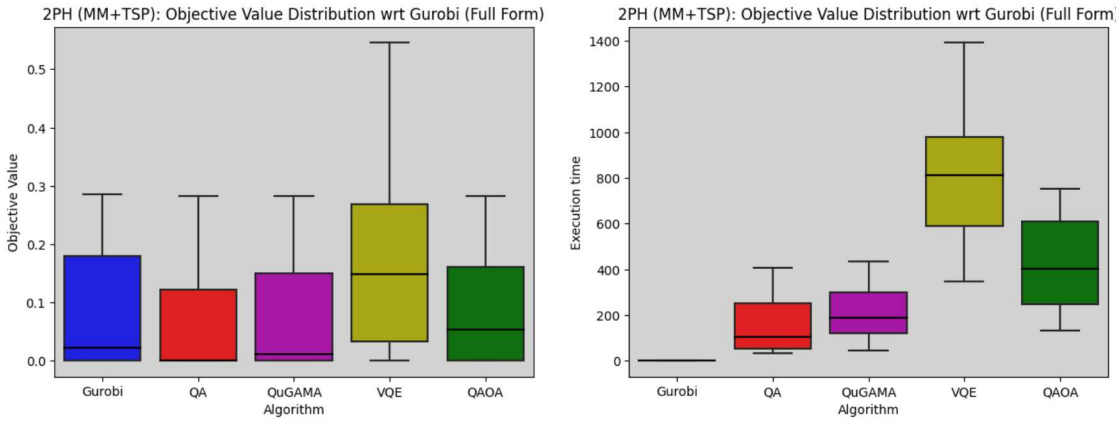


Figure 7.11: Boxplots for the Two Phase Heuristic approach: MM + TSP. On the left, a boxplot comparing the distribution of optimal objective function values. On the right, a boxplot comparing the execution times of the various algorithms.

These boxplots offer intriguing insights. They indicate that when clusters are formed based on Modularity Maximization, Quantum Annealing and Quantum GAMA manage to surpass Gurobi within our two-phase heuristic approach. Additionally, it is worth noting that, in this scenario, several instances achieve the same optimal solution as the full two-index formulation when using the two-phase heuristic. Specifically, Quantum Annealing successfully matches Gurobi's optimal solutions from the two-index formulation in half of the test instances.
Overall, all algorithms, with the exception of VQE, perform reasonably well, delivering strong approximations to the true optimum. Once more, VQAs tend to be the slowest algorithms; however, it is crucial to keep in mind that these are executed through classical simulation.
To conclude this subsection, we turn our attention to the following scatter plots that

compare Gurobi's optimal objectives with the optimal objectives achieved by our quantum algorithms. Once again, in the final rightmost plot, Gurobi's objectives are the optimal objectives of the complete two-index CVRP formulation.

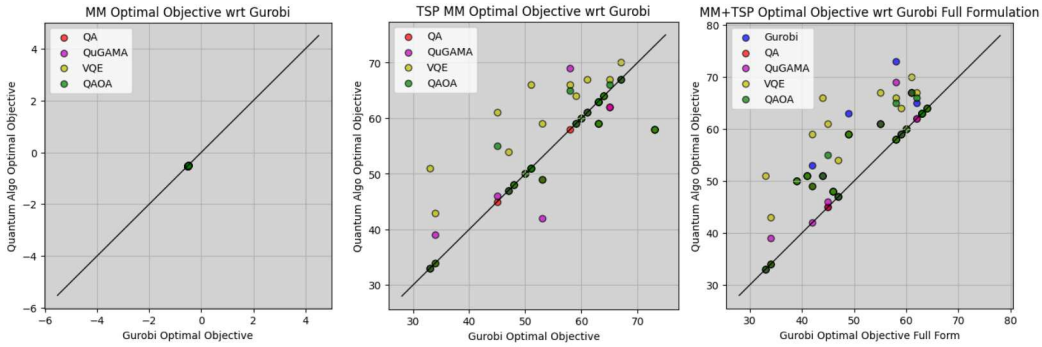


Figure 7.12: *Left:* Optimal objective function values for MM clustering phase. *Center:* Optimal objective function values for TSP after MM clustering. *Right:* Optimal objective function values for Two Phase Heuristic approach: MM + TSP. In each plot, the x-axis represents Gurobi's optimal objective values, while the y-axis depicts the optimal objectives obtained by various quantum algorithms.

Clustering via Louvain Algorithm

Louvain clustering is established using a classical greedy algorithm, rendering boxplots unnecessary for the clustering phase. We can thus delve directly into the routing phase on Louvain clusters.

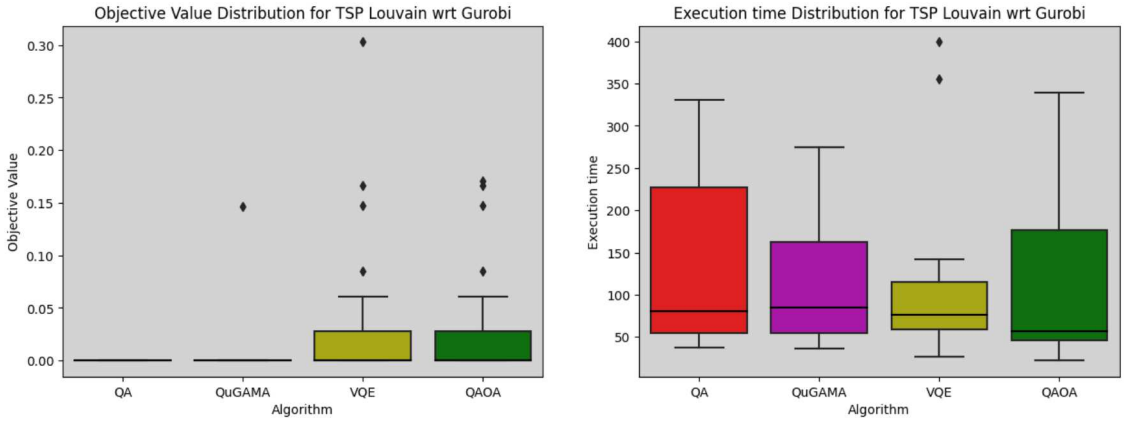


Figure 7.13: Boxplots for TSP after Louvain clustering. On the left, a boxplot comparing the distribution of optimal objective function values. On the right, a boxplot comparing the execution times of the various algorithms.

The Louvain clusters have proven to be the most effective choice in our evaluation. All algorithms demonstrate strong performance on these clusters. Quantum Annealing and Quantum GAMA are able to match Gurobi's optimal solutions, and VQAs also achieve remarkable approximations in these cases. Yet, the execution times for TSP now seem slightly longer when compared to the earlier clustering methods.
Now, let us examine a final pair of boxplots. Similar to previous instances, these boxplots represent objective function values and execution times relative to the optimal objective and execution time achieved by Gurobi when solving the complete two-index formulation of CVRP, without employing the two-phase heuristic approach. The plotted objective function values represent the sum of TSP optimal objectives on the Louvain clusters, and execution times are the sum of clustering and routing execution times. It is worth noting that these plots also include Gurobi's performances when applied via the two-phase heuristic, providing insight into the validity of this approach compared to the complete two-index formulation.

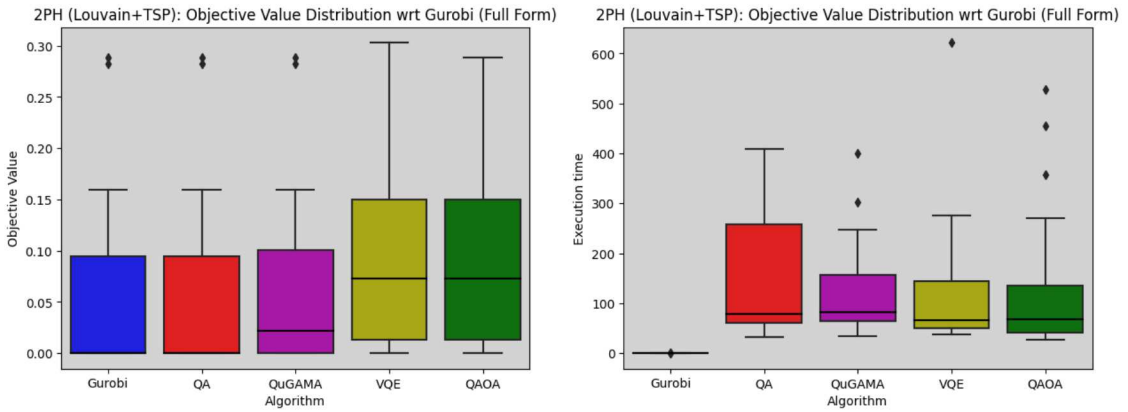


Figure 7.14: Boxplots for the Two Phase Heuristic approach: Louvain + TSP. On the left, a boxplot comparing the distribution of optimal objective function values. On the right, a boxplot comparing the execution times of the various algorithms.

We can see that, as previously mentioned, Louvain clustering appears to be the most promising option for our purposes. Despite the heuristic nature of our approach, these clusters enable Gurobi and Quantum Annealing to achieve the true optimal solution in half of the test instances. On the remaining instances, the two-phase heuristic consistently delivers impressive approximations to the true optimum. Once again, Quantum Annealing and Quantum GAMA stand out as the top-performing algorithms, while VQAs show relatively less favorable performance. Nonetheless, they still manage to provide reasonable approximations to the true optima.
To conclude this subsection, let us examine the scatter plots contrasting Gurobi's optimal objective with the optimal objectives achieved by our quantum algorithms. As

before, in the final rightmost plot, Gurobi's objectives represent the optimal objectives of the complete two-index CVRP formulation.

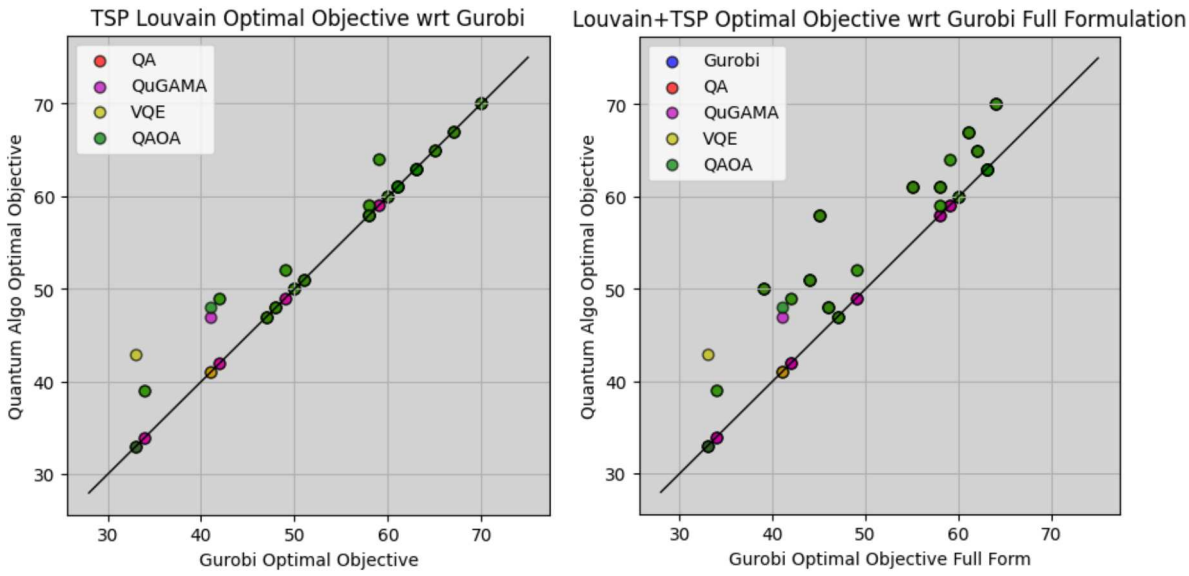


Figure 7.15: *Left:* Optimal objective function values for TSP after Louvain clustering. *Right:* Optimal objective function values for Two Phase Heuristic approach: Louvain + TSP. In each plot, the x-axis represents Gurobi's optimal objective values, while the y-axis depicts the optimal objectives obtained by various quantum algorithms.

Final Observations

In this closing subsubsection, we conclude our study by comparing the performance of our quantum algorithms within the context of the two-phase heuristic while utilizing various clustering algorithms. To begin, we present the following plots, showing the optimal objective function values.

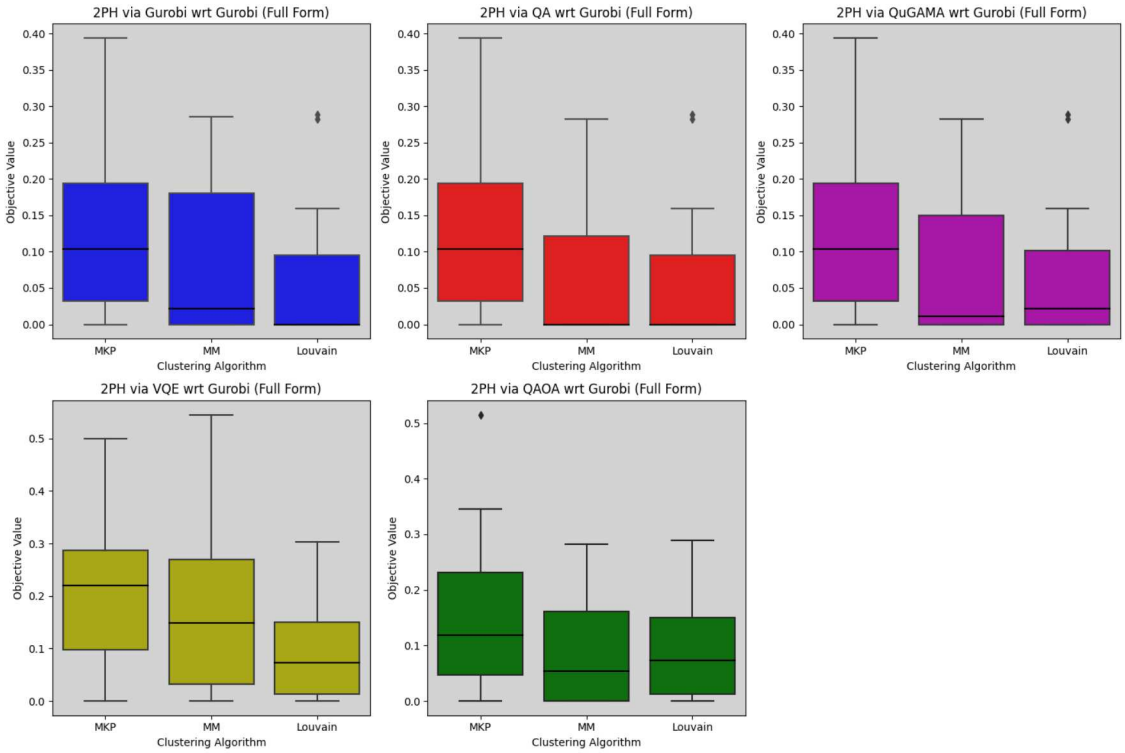


Figure 7.16: Boxplots to illustrate the performance of several algorithms when implemented via the two-phase heuristic approach. Performances are evaluated considering various clustering algorithms: MKP, MM, and Louvain. The plots, presented from left to right and top to bottom, correspond to Gurobi, Quantum Annealing, Quantum GAMA, VQE, and QAOA.

As evident from the Gurobi boxplots, the two-phase heuristic approach consistently provides substantial approximations to the optimal solutions derived from the full two-index formulations. Unexpectedly, both MM and Louvain clustering appear to outperform MKP clustering by a significant margin. In particular, Louvain clustering facilitates the two-phase heuristic in achieving the true optimal solution in half of the test instances. This discovery holds considerable significance, especially as the current literature favors MKP clustering, often without considering Louvain clustering as an alternative. However, our results indicate that Louvain clustering considerably outperforms MKP clustering, enhancing the effectiveness of the subsequent routing phase. It is important to note that what we refer to as "Louvain" is essentially a customized Louvain

algorithm, as described in (7.1).
The same trend is evident in the remaining plots, where clusters obtained through MM and Louvain consistently outperform those derived from MKP. This observation carries substantial significance, primarily because Louvain clusters can be computed efficiently even for graphs with a high number of nodes. In contrast, solving MKP on a graph with numerous nodes would inevitably lead to lengthy execution times. This indicates that our Louvain approach not only delivers superior results but is also expected to scale more effectively when dealing with larger problem instances.
As expected, Quantum Annealing and Quantum GAMA consistently exhibit superior performance compared to VQAs. Of the two, Quantum Annealing generally outperforms Quantum GAMA, even achieving the true optimum on half of the test instances when clustering with MM or Louvain. Within the realm of VQAs, QAOA tends to outperform VQE, providing more accurate approximations to the true optimum.
The consistent performance of Quantum GAMA, almost on par with Gurobi and Quantum Annealing, holds significant importance in our study. Quantum GAMA is a relatively new algorithm, and as far as our knowledge extends, there is no comprehensive study focused on addressing the Capacitated Vehicle Routing Problem (CVRP) using this algorithm. What makes Quantum GAMA particularly noteworthy is its status as a test-set-based algorithm. In the process of computing the test set, the algorithm does not involve the objective function explicitly; instead, the objective function acts as an oracle in the final classical augmenting subroutine. This characteristic endows Quantum GAMA with a robust capability to handle highly complex combinatorial optimization problems with intricate, nonlinear objective functions. The algorithm's remarkable performance, nearly matching that of Gurobi, underscores its substantial potential in this domain.
Lastly, let us take a brief look at the variations in execution times. In the upcoming plot (7.17), we depict the overall execution time of the Two-Phase heuristic approach for each algorithm. This cumulative execution time is computed by adding the clustering time and the routing time for each algorithm and clustering strategy. As observed, the Louvain strategy tends to decrease the overall execution time of the two-phase heuristic approach. This improvement is primarily attributed to the clustering phase itself. Clustering via Louvain, a greedy algorithm, proves significantly faster than achieving clustering through optimally solving a combinatorial optimization problem.
The only algorithm that appears to deviate slightly from this trend is Quantum Annealing, where MKP clustering seems to be the fastest on average. However, even for Quantum Annealing, Louvain clustering achieves shorter execution times on half of the test instances. Additionally, from the same plot, we infer that our quantum algorithms do not rival Gurobi in terms of execution times. Gurobi emerges as the fastest solver, surpassing our quantum algorithms by several orders of magnitude. Finally, it is worth noting that VQA exhibits a considerably slower pace compared to pure quantum

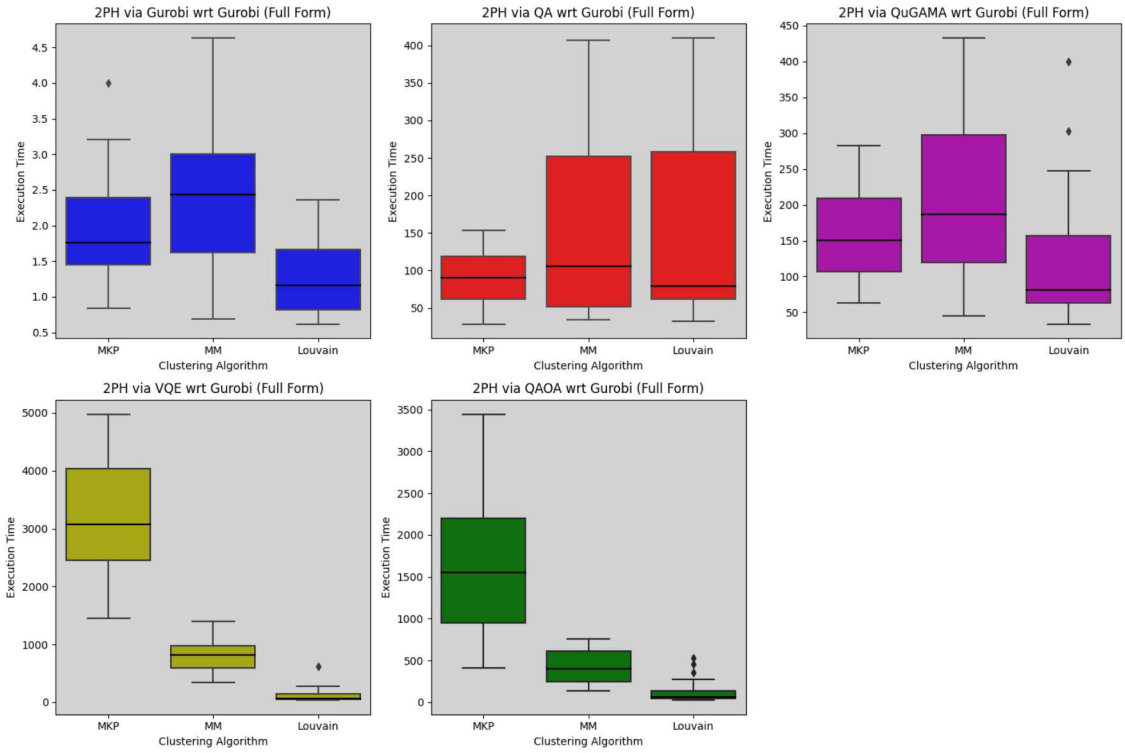


Figure 7.17: Boxplots to illustrate the execution times of several algorithms when implemented via the two-phase heuristic approach. Execution times are computed as the sum of both clustering times and routing times. We compare various clustering algorithms: MKP, MM, and Louvain. The plots, presented from left to right and top to bottom, correspond to Gurobi, Quantum Annealing, Quantum GAMA, VQE, and QAOA.

algorithms. It is essential to remember, though, that these algorithms are simulated classically, rendering their execution times less significant in our study. Additionally, for pure quantum algorithms, the total execution times include not only the time spent on quantum hardware usage but also the classical routines used to refine the output from the quantum hardware.
The notable speed advantage of Louvain clustering over MKP or MM clustering aligns with our earlier observation: the Louvain approach is anticipated to exhibit even better scalability for larger problems, where achieving optimality in MKP and MM would be significantly slower.
To conclude, we reiterate that our study is primarily an exploration of quantum alternatives for the Capacitated Vehicle Routing Problem (CVRP), rather than a direct competition against classical methods. Despite this, our investigation has revealed several valuable insights, highlighting the promising performance of quantum algorithms. In some cases, they even perform on par with Gurobi, a leading commercial solver for combinatorial optimization problems. One of the most significant findings is the effectiveness of MM and Louvain clusters, which enhance the two-phase heuristic approach, enabling it to achieve better approximations while also enhancing its scalability to larger problem instances.

8 Conclusion

Quantum computing, hinging on the unique principles of quantum mechanics, holds the potential to transform optimization problems by providing faster and more efficient solutions. In this research, we employed quantum optimization methods to address the intricate Capacitated Vehicle Routing Problem (CVRP), a widely recognized and challenging combinatorial optimization problem in the logistics and transportation industries.

The Capacitated Vehicle Routing Problem (CVRP) represents the most basic variant within the family of Vehicle Routing Problems (VRPs). CVRP's core goal is to optimize the routing of a fleet of vehicles for delivering goods to a designated group of customers, while adhering to constraints like maximum vehicle capacity. The primary aim is usually to minimize the total travel distance or overall transportation cost. CVRP essentially expands upon the well-known Traveling Salesman Problem (TSP) by introducing multiple vehicles and the challenge of optimally allocating nodes among these vehicles.

In order to apply quantum techniques to tackle the CVRP (as well as any other combinatorial problem), it is essential to convert these optimization problems so to make them compatible with the quantum hardware. This is accomplished by utilizing Quadratic Unconstrained Binary Optimization (QUBO) and Ising formulations. QUBO enables the conversion of a generic combinatorial optimization problem into an unconstrained binary problem. In essence, QUBO formalisms resemble what is known as Penalty methods in Operations Research. The fundamental idea is to transform the original constrained problem into an unconstrained one by removing the constraints and instead including them as additive positive penalty terms directly into the cost function. In the domain of quantum mechanics, these unconstrained binary problems are commonly referred to as Ising models. QUBO/Ising acts as the bridge connecting combinatorial optimization with quantum algorithms.

In this thesis, we delved into a range of quantum algorithms, encompassing both pure quantum approaches such as Quantum Annealing and Quantum GAMA (a relatively recent and unexplored algorithm), as well as hybrid classic-quantum algorithms like Variational Quantum Eigensolver (VQE) and Quantum Approximate Optimization Algorithm (QAOA). Pure quantum algorithms were evaluated using the *DWave Advantage System 6.3*, an actual quantum computer. On the other hand, VQE and QAOA were simulated classically using the *statevector simulator* from the Qiskit library. This simulator provides an ideal, noise-free simulation environment.
Our goal was to evaluate the effectiveness of these algorithms in solving the CVRP and to conduct a comparative analysis against the classical solver Gurobi, which served as a reference point for assessing classical performance.
To accurately represent the CVRP, we initially utilize the widely recognized two-index vehicle flow formulation, employing variables x_{ij} to signify whether a vehicle has traversed the arc (i, j). However, this formulation was found to be impractical, primarily because of the stringent constraints imposed by current Noisy Intermediate-Scale Quantum (NISQ) devices, which encounter difficulties when dealing with a larger number of variables.
As a result, our approach shifted towards a two-phase heuristic strategy. In this strategy, we address the CVRP sequentially: initially, we utilize clustering techniques to group nodes into separate clusters, with each cluster assigned to one vehicle. Then, we proceed to address the classic Traveling Salesman Problem (TSP) within each of these individual clusters. This two-phase heuristic method effectively reduces the total number of variables required to model the problem, thereby mitigating the computational demands that made the complete two-index vehicle flow formulation impractical.
Within the framework of this two-phase heuristic approach, the commonly employed clustering algorithm in the existing literature is the Multi Knapsack Problem (MKP). Typically, for determining optimal clusters, we solve an MKP with a quadratic objective function. In this thesis, in addition to investigating this MKP-based clustering strategy, we also examined two other clustering algorithms based on modularity maximization. The first strategy entails clusters determination through the explicit resolution of the Modularity Maximization optimization problem. The second strategy revolves around the utilization of the Louvain algorithm, a classical greedy modularity maximization algorithm.
To evaluate the various algorithms and clustering strategies, we employed 25 randomly generated test instances. These instances were intentionally kept relatively small, with a maximum of 5 nodes and 2 vehicles for service. Unfortunately, this was the most reasonable choice, taking into account the current constraints imposed by hardware limitations and resource costs.
Significantly, our research highlighted Quantum Annealing and Quantum GAMA as the standout performers, consistently surpassing their hybrid counterparts. These algo-

rithms consistently showcased remarkable capabilities, often on par with Gurobi. Their robust performance underscores their potential in tackling intricate optimization problems.
On the other hand, Variational Quantum Algorithms (VQAs) exhibited comparatively less favorable performance when contrasted with Quantum Annealing and Quantum GAMA. Nevertheless, VQAs still managed to provide reasonably accurate approximations to the true optimal solutions. This observation sheds light on the differing performance between purely quantum algorithms and hybrid approaches like VQAs.
It is also crucial to recall that the VQE and QAOA algorithms were executed using the statevector simulator, which provides an idealized, noise-free simulation environment. This aspect holds particular significance because real-world quantum devices are subject to various sources of noise. Therefore, the expected performance of VQAs on actual quantum hardware might further decline due to the presence of these quantum noisy channels. Consequently, the overall trends we observed for VQAs, while promising on their own, could encounter additional challenges during the transition from the ideal simulator to practical quantum computing platforms.
Our investigation has also showed the significant role of clustering strategies within the context of the Capacitated Vehicle Routing Problem (CVRP). Notably, the Louvain clustering approach emerged as a standout performer, surpassing the traditional Multi-Knapsack Problem (MKP) approach. In addition to delivering improved final solutions, Louvain clusters are also expected to enhance the scalability of the approach for larger instances, as solving Louvain on a large graph is considerably faster than achieving optimality in an MKP.
In addition to Louvain, even the clusters derived from the direct solution of Modularity Maximization combinatorial problem appear to be superior to those obtained through MKP. This finding underscores the potential advantages of incorporating Modularity Maximization algorithms into the two-phase heuristic approach for CVRP. These algorithms have shown promise in generating enhanced cluster configurations, which, in turn, can elevate the quality of routing solutions.
These findings question the prevailing literature, which frequently favors MKP clustering while overlooking the exploration of alternative clustering methods. Our results indicate that MKP may not necessarily be the most optimal choice for the two-phase heuristic strategy.
While we appreciate these remarkable discoveries, it is crucial to recognize the constraints of our research. Our primary focus has been on instances of relatively modest size, owing to limitations in resources and the existing state of quantum hardware. Looking ahead, it becomes imperative to revisit these algorithms as more potent quantum hardware emerges, with the aim of evaluating their performance on larger instances, which may pose distinct and greater challenges.
Potential future enhancements also encompass the exploration of alternative quantum

optimization algorithms, such as Quantum Evolutionary Algorithms and Quantum Walk-based approaches, among others. These algorithms were omitted from the thesis due to their extensive computational resource requirements. Quantum Evolutionary Algorithms, as the name implies, are the quantum counterparts of memetic algorithms, implying that they operate on populations of solutions.
Furthermore, it would be intriguing to investigate the application of machine learning techniques to fine-tune the parameters of Variational Quantum Algorithms (VQAs). Some of these machine learning approaches are already outlined in the relevant section of the thesis, although computational tests were unfeasible due to resource limitations.
Lastly, and perhaps most importantly, the clustering phase, a pivotal component of our two-phase heuristic approach, offers ample opportunity for further improvement through the examination of various classical algorithms. This presents an avenue for enhancing the efficiency and effectiveness of the entire CVRP-solving process.
In summary, within the dynamic and ever-evolving landscape of quantum computing, our research stands as an initial step, paving the way for further exploration and inquiry. It provides a glimpse of the considerable potential in using quantum computing for solving complex optimization problems.

References

[1] W. Scherer, *Mathematics of Quantum Computing: An Introduction*, 01 2019.

[2] R. de Wolf, "Quantum computing: Lecture notes," 2023.

[3] M. Nielsen and I. Chuang, "Quantum computation and quantum information, 10th anniversary edition," 01 2010.

[4] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics*, 3rd ed., ser. International Series in Operations Research & Management Science. Springer, 2019.

[5] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using qubo models," 2019.

[6] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, 2014. [Online]. Available: https://doi.org/10.3389%2Ffphy.2014.00005

[7] V. Choi, "Minor-embedding in adiabatic quantum computation: I. the parameter setting problem," 2008.

[8] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, jan 2018. [Online]. Available: https://doi.org/10.1103%2Frevmodphys.90.015002

[9] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Phys. Rev. E*, vol. 58, pp. 5355–5363, Nov 1998. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.58.5355

[10] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum computation by adiabatic evolution," 2000.

[11] B. Sturmfels, "What is a gröbner basis?" vol. 52, no. 10, 2005.

[12] D. E. Bernal, S. Tayur, and D. Venturelli, "Quantum integer programming (quip) 47-779: Lecture notes," 2021.

[13] H. Alghassi, R. Dridi, and S. Tayur, "Gama: A novel algorithm for non-convex integer programs," 2019.

[14] ——, "Graver bases via quantum annealing with application to non-linear integer programs," 2019.

[15] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, aug 2021.

[16] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. 1, jul 2014.

[17] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014.

[18] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 01 1965.

[19] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization." *Math. Program.*, vol. 45, no. 1-3, pp. 503–528, 1989. [Online]. Available: http://dblp.uni-trier.de/db/journals/mp/mp45.html#LiuN89

[20] P. I. Frazier, "A tutorial on bayesian optimization," 2018.

[21] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, no. 2, pp. 155–162, 01 1964. [Online]. Available: https://doi.org/10.1093/comjnl/7.2.155

[22] ——, *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, 1994, pp. 51–67. [Online]. Available: https://app.dimensions.ai/details/publication/pub.1046127469

[23] M. Powell, "The bobyqa algorithm for bound constrained optimization without derivatives," *Technical Report, Department of Applied Mathematics and Theoretical Physics*, 01 2009.

[24] A. H. G. Rinnooy Kan and G. T. Timmer, "The Multi Level Single Linkage Method For Unconstrained And Constrained Global Optimization," Erasmus University Rotterdam, Econometric Institute Archives 272327, 1985. [Online]. Available: https://ideas.repec.org/p/ags/eureia/272327.html

[25] J. Larson and S. Wild, "Asynchronously parallel optimization solver for finding multiple minima," *Mathematical Programming Computation*, vol. 10, pp. 1–30, 02 2018.

[26] A. Pellow-Jarman, I. Sinayskiy, A. Pillay, and F. Petruccione, "A comparison of various classical optimizers for a variational quantum linear solver,"

Quantum Information Processing, vol. 20, no. 6, jun 2021. [Online]. Available: https://doi.org/10.1007%2Fs11128-021-03140-x

[27] Z.-C. Yang, A. Rahmani, A. Shabani, H. Neven, and C. Chamon, "Optimizing variational quantum algorithms using pontryagin's minimum principle," *Physical Review X*, vol. 7, no. 2, may 2017. [Online]. Available: https://doi.org/10.1103%2Fphysrevx.7.021027

[28] S. Hadfield, Z. Wang, B. O'Gorman, E. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, feb 2019. [Online]. Available: https://doi.org/10.3390%2Fa12020034

[29] O. D. Parekh, C. Ryan-Anderson, and S. Gharibian, "Quantum optimization and approximation algorithms." 1 2019. [Online]. Available: https://www.osti.gov/biblio/1492737

[30] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Physical Review X*, vol. 10, no. 2, jun 2020. [Online]. Available: https://doi.org/10.1103%2Fphysrevx.10.021067

[31] D. Wecker, M. B. Hastings, and M. Troyer, "Training a quantum optimizer," *Physical Review A*, vol. 94, no. 2, aug 2016. [Online]. Available: https://doi.org/10.1103%2Fphysreva.94.022309

[32] R. Shaydulin, I. Safro, and J. Larson, "Multistart methods for quantum approximate optimization," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, sep 2019. [Online]. Available: https://doi.org/10.1109%2Fhpec.2019.8916288

[33] G. E. Crooks, "Performance of the quantum approximate optimization algorithm on the maximum cut problem," 2018.

[34] S. Khairy, R. Shaydulin, L. Cincio, Y. Alexeev, and P. Balaprakash, "Learning to optimize variational quantum circuits to solve combinatorial problems," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, pp. 2367–2375, apr 2020. [Online]. Available: https://doi.org/10.1609%2Faaai.v34i03.5616

[35] A. Ng, "Cs229 lecture notes," 2022.

[36] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," 2017.

[37] A. Garcia-Saez and J. I. Latorre, "Addressing hard classical problems with adiabatically assisted variational quantum eigensolvers," 2018.

[38] K. Bharti, "Quantum assisted eigensolver," 2020.

[39] P. Toth and D. Vigo, *Vehicle Routing. Problems, Methods, and Applications*, 2nd ed., 2014.

[40] APalmier99, "Quantum_integer_programming," https://github.com/APalmier99/Quantum_Integer_Programming, 2023.

[41] E. Osaba, E. Villar-Rodriguez, and I. Oregi, "A systematic literature review of quantum computing for routing problems," *IEEE Access*, vol. 10, pp. 55 805–55 817, 2022.

[42] I. D. Leonidas, A. Dukakis, B. Tan, and D. G. Angelakis, "Qubit efficient quantum algorithms for the vehicle routing problem on quantum computers of the nisq era," 2023.

[43] L. Palackal, B. Poggel, M. Wulff, H. Ehm, J. M. Lorenz, and C. B. Mendl, "Quantum-assisted solution paths for the capacitated vehicle routing problem," 2023.

[44] K. M. Nakanishi, K. Fujii, and S. Todo, "Sequential minimal optimization for quantum-classical hybrid algorithms," *Physical Review Research*, vol. 2, no. 4, Oct. 2020. [Online]. Available: http://dx.doi.org/10.1103/PhysRevResearch.2.043158

[45] N. Mohanty, B. K. Behera, and C. Ferrie, "Analysis of the vehicle routing problem solved via hybrid quantum algorithms in presence of noisy channels," 2023.

[46] D. Fitzek, T. Ghandriz, L. Laine, M. Granath, and A. F. Kockum, "Applying quantum approximate optimization to the heterogeneous vehicle routing problem," 2021.

[47] C. G. BROYDEN, "The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations," *IMA Journal of Applied Mathematics*, vol. 6, no. 1, pp. 76–90, 03 1970. [Online]. Available: https://doi.org/10.1093/imamat/6.1.76

[48] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 01 1997.

[49] U. Azad, B. K. Behera, E. A. Ahmed, P. K. Panigrahi, and A. Farouk, "Solving vehicle routing problem using quantum approximate optimization algorithm," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2022. [Online]. Available: https://doi.org/10.1109%2Ftits.2022.3172241

[50] H. Irie, G. Wongpaisarnsin, M. Terabe, A. Miki, and S. Taguchi, "Quantum annealing of vehicle routing problem with time, state and capacity," 2019.

[51] S. Feld, C. Roch, T. Gabor, C. Seidel, F. Neukart, I. Galter, W. Mauerer, and C. Linnhoff-Popien, "A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer," *Frontiers in ICT*, vol. 6, jun 2019. [Online]. Available: https://doi.org/10.3389%2Ffict.2019.00013

[52] J. Spall, "An overview of the simultaneous perturbation method for efficient optimization," 02 2001.

[53] R. S. Dembo and T. Steihaug, "Truncated-newtono algorithms for large-scale unconstrained optimization," *Mathematical Programming*, vol. 26, pp. 190–212, 1983. [Online]. Available: https://api.semanticscholar.org/CorpusID:40537623

[54] D. J. Egger, J. Mareč ek, and S. Woerner, "Warm-starting quantum optimization," *Quantum*, vol. 5, p. 479, jun 2021. [Online]. Available: https://doi.org/10.22331%2Fq-2021-06-17-479

[55] N. K. Suresh and P. Ramasamy, "A survey on the vehicle routing problem and its variants," *Intelligent Information Management*, 2012.

[56] A. M. Childs, "Lecture notes on quantum algorithms," 2022.

[57] R. M. Parrish, E. G. Hohenstein, P. L. McMahon, and T. J. Martínez, "Quantum computation of electronic transitions using a variational quantum eigensolver," *Physical Review Letters*, vol. 122, no. 23, jun 2019. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.122.230401

[58] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme, "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Science and Technology*, vol. 3, no. 3, p. 030503, jun 2018. [Online]. Available: https://doi.org/10.1088%2F2058-9565%2Faab822

[59] W. Vinci and D. A. Lidar, "Non-stoquastic hamiltonians in quantum annealing via geometric phases," *npj Quantum Information*, vol. 3, no. 1, sep 2017. [Online]. Available: https://doi.org/10.1038%2Fs41534-017-0037-z

[60] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. Love, and A. Aspuru-Guzik, "Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz," 2018.

[61] R. Shaydulin, H. Ushijima-Mwesigwa, I. Safro, S. Mniszewski, and Y. Alexeev, "Network community detection on small quantum computers," *Advanced Quantum Technologies*, vol. 2, no. 9, p. 1900029, jun 2019. [Online]. Available: https://doi.org/10.1002%2Fqute.201900029

Acknowledgments

I extend heartfelt gratitude to Professor Francesco Rinaldi, my supervisor, whose invaluable guidance has played a crucial role in shaping the trajectory of my research and ensuring the successful completion of this thesis. My deepest appreciation also goes to my family for their unwavering support, both financially and emotionally, throughout these years. Lastly, I want to express my gratitude to my friends; your enduring patience and companionship have been invaluable in fostering my personal growth.