Hybrid Quantum Classical Algorithms for Machine Learning and Optimization and Applications in Transport and Scheduling Problems

By

Athanasios Karakos

School of Electrical and Computer Engineering
Technical Univeristy of Crete

Thesis Commitee
Professor Dimitris G. Angelakis (Supervisor)
Associate Professor Vasilis Samoladas
Professor Thrasyvoulos Spyropoulos

Chania September 2025

Abstract

This thesis investigates hybrid quantum-classical algorithms with a focus on Quadratic Unconstrained Binary Optimization (QUBO) formulations and their application to transport and scheduling problems. It begins with an overview of fundamental concepts in quantum mechanics, including qubits, quantum gates, and entanglement, to provide the necessary background. Building on this foundation, we analyze QUBO formulations and hybrid algorithms such as QAOA, VQA, ADAPT-QAOA, and qubit-efficient encoding schemes. We further explore the role of hybrid algorithms in generative AI, where transformer-based architectures are employed to generate parameterized quantum circuits. A substantial part of the work is devoted to a multimodal transport scheduling problem, modeled using integrated QUBO formulations. Simulations were carried out on both classical and quantum backends. The results show that quantum algorithms can deliver competitive solutions and highlight their strong potential for scalability as quantum hardware advances.

Table of Contents

Chapter 1

Introduction

1.1 Introduction

Over the past decades, classical computing has transformed nearly every aspect of science and technology. Yet, as computational demands continue to grow, many problems remain beyond the reach of even the most powerful supercomputers. These include tasks such as simulating quantum systems in chemistry and materials science, optimizing large-scale industrial processes, and training complex machine learning models. To address these challenges, a new paradigm **quantum computing** has emerged.

Quantum computing leverages the principles of *quantum mechanics*, the fundamental theory governing the behavior of matter and energy at microscopic scales. Unlike classical bits, which can be either 0 or 1, *quantum bits (qubits)* can exist in a superposition of states, enabling them to represent multiple possibilities at once. Moreover, phenomena such as *entanglement* and *quantum interference* provide powerful mechanisms for processing information in ways that classical systems cannot replicate.

The potential of quantum computing is highlighted by algorithms such as *Shor's algorithm*, which threatens the security of widely used cryptographic schemes by factoring large numbers efficiently, and *Grover's algorithm*, which offers quadratic speedups for unstructured search problems. Beyond these landmark results, recent research has focused on the development of quantum algorithms for optimization, simulation, and machine learning, areas where quantum devices may provide practical advantages.

Despite this promise, building large-scale fault-tolerant quantum computers remains a formidable challenge. Current hardware, often referred to as the *Noisy Intermediate-Scale Quantum (NISQ)* era, consists of devices with tens to hundreds of imperfect qubits. To harness these systems effectively, researchers have turned to *hybrid quantum-classical*

approaches, which combine quantum state preparation and measurement with classical optimization techniques.

Quantum computing thus represents both a scientific frontier and a technological opportunity. By exploiting the unique features of quantum mechanics, it has the potential to transform fields ranging from cryptography and optimization to materials science and artificial intelligence. This thesis situates itself within this broader effort, focusing on the development and application of quantum algorithms for solving optimization problems.

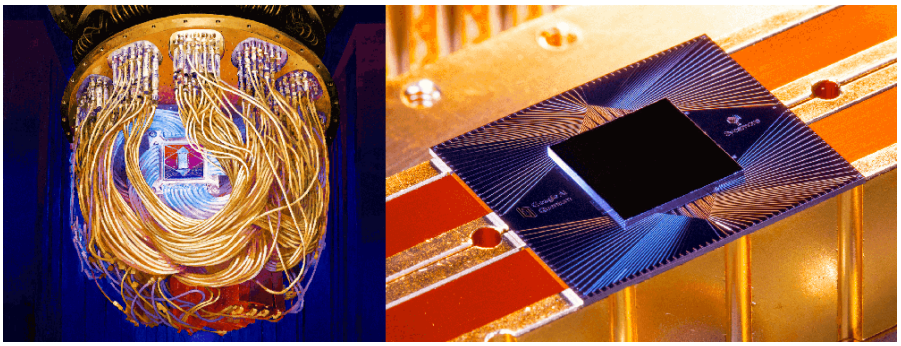


Figure 1.1: Transmon superconducting quantum processor created by Google's Artificial Intelligence division. It is called Sycamore and it has 53 qubits. In 2019, Sycamore completed a task in 200 seconds that Google claimed, in a Nature paper, would take a state-of-the-art supercomputer 10,000 years to finish. Thus, Google claimed to have achieved quantum supremacy. Source.

Chapter 2

Background in Quantum Mechanics

2.1 Quantum Spin and Quantum Bits

In quantum computing, the basic unit of information is the quantum bit, or *qubit*, which serves a role similar to that of a classical bit. Unlike classical bits, which are strictly either 0 or 1, qubits can exist in multiple states at once due to the principles of **superposition and entanglement**. Qubits can be realized physically in several ways, such as through superconducting circuits, trapped ions, or photons, with each approach offering distinct advantages and challenges.

To understand and describe the core principles of quantum mechanics, it is useful to consider a theoretical model that illustrates the behavior of a qubit. A common choice is quantum spin. While this usually refers to an intrinsic property of electrons, it can also be treated as a conceptual model for qubits. Quantum spin is often depicted as an arrow pointing in a specific direction, providing a visual way to grasp the abstract nature of qubit states.

2.2 Quantum States

Here we delve into the concept of quantum states, using the quantum spin system, particularly spins along the z-axis, as a primary example. The states of spin up and down are introduced and denoted by *ket* vectors $|u\rangle$ and $|d\rangle$ or $|0\rangle$ and $|1\rangle$ respectively. Paul Dirac introduced the symbol $|*\rangle$ in 1958 to represent quantum states, which are essentially vectors of complex numbers.

In classical computing, the state space is just a set of all possible states, like $\{0, 1\}$ for a bit. But in quantum computing, the state space is a vector space, which is a bit

more complex. A quantum state is shown as a vector in a special kind of complex vector space called the **Hilbert space**, represented by $\mathcal{H}$. Think of it as a bigger, more flexible version of the Euclidean space you're familiar with, able to stretch over finite or infinite dimensions.

In this quantum world, we use *kets* to represent the vectors in the Hilbert space. The simplest example involves a two-level quantum system where the states $|0\rangle$ and $|1\rangle$ are the basic building blocks. These two states form an orthonormal basis for the vector space and are defined like this:

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

The basis $|0\rangle$ and $|1\rangle$ is the most common among the various bases used in quantum computation. It is used because it naturally corresponds to classical bits 0 and 1. One major difference between a qubit and a classical bit is that a qubit can exist in multiple states simultaneously. This unique feature of quantum mechanics is called superposition. Mathematically, **superposition is shown as a linear combination of states**:

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{C}^2$$

Here, a and b are known as complex probability amplitudes. They are connected to probabilities, even if this connection isn't obvious at first. By themselves, a and b don't have direct experimental implications, but their magnitudes are important. When you measure a qubit, it collapses to the state $|0\rangle$ with a probability of $|a|^2 = a^*a$, and to the state $|1\rangle$ with a probability of $|b|^2 = b^*b$. This probabilistic nature is described by the Born rule, highlighting the random aspect of quantum mechanics. Before measurement, the qubit's state is represented by $|\psi\rangle$. Upon measurement, the state collapses to one outcome, with probabilities given by the squares of the magnitudes of a and b. This means the probabilities must add up to 1:

$$|a|^2 + |b|^2 = 1$$

More generally, any quantum state in $\mathbb{C}^n$ is written as:

$$|\psi\rangle = \sum_{i=1}^{n} c_i\,|u_i\rangle$$

where vectors like $|u_1\rangle, |u_2\rangle, ..., |u_n\rangle$ form an orthonormal basis in $\mathbb{C}^n$. The probabilities derived from these vectors create a valid probability distribution, ensuring they sum

to one:

$$\sum_{i=1}^{n} |c_i|^2 = \sum_{i=1}^{n} c_i^* c_i = 1$$

Geometrically, the state of a quantum system is a normalized vector with a magnitude of 1 in the multi-dimensional Hilbert space of all basis states. Throughout our study, it's clear that "ket" vectors are complex vectors. Each ket has a corresponding complex conjugate called a "bra" (coming from 'bracket'). If a ket is a vertical column vector, its bra is a horizontal row vector:

$$|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix}, \quad |\psi\rangle^\dagger = \langle\psi| = \begin{pmatrix} a^* & b^* \end{pmatrix} \in \mathbb{C}^2$$

The bracket, defined as the dot product between a bra and a ket, is written as $\langle\psi|\phi\rangle$.

2.3 Bloch Sphere Representation

Since quantum spin can be visualized as an arrow pointing in a direction within three-dimensional space, it is essential to define the three spatial directions: x, y, and z, to describe the spin's orientation. Additionally, we must consider an extra degree of freedom that specifies the specific direction of the spin along its defined axis. For instance, when oriented along the z-axis, this quantum system can be in two states: spin up ($\sigma_z = +1$) or spin down ($\sigma_z = -1$). To determine the spin, we need a measurement apparatus capable of interfacing with the quantum system to inform us about its spin along a specified direction. The measurement procedure involves aligning the apparatus with the direction of interest, such as the z-axis, and observing the outcomes: $+1$ if the spin points up or -1 if it points down.

Qubits and, in general, particles like photons can have both a wave-like behavior and a particle-like behavior; this is called wave-particle duality. Before measuring a qubit, its behavior is given by a wave function, but after the measurement, this behavior collapses and the particle is measured in a certain state. The measurement is done on the spin of the particle. To better represent the spin of a qubit/particle in a 3D space, the Bloch sphere is used. **The Bloch sphere is a geometric visualization of a quantum state as a point in a sphere. The north and south poles of the sphere represent $|1\rangle$ and $|0\rangle$, and the equator represents an equal superposition of the two**. Because of the states representing spins, states are also denoted as:

$$|0\rangle = |\uparrow\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = |\downarrow\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The qubit's state in the Bloch sphere is written in the form:

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

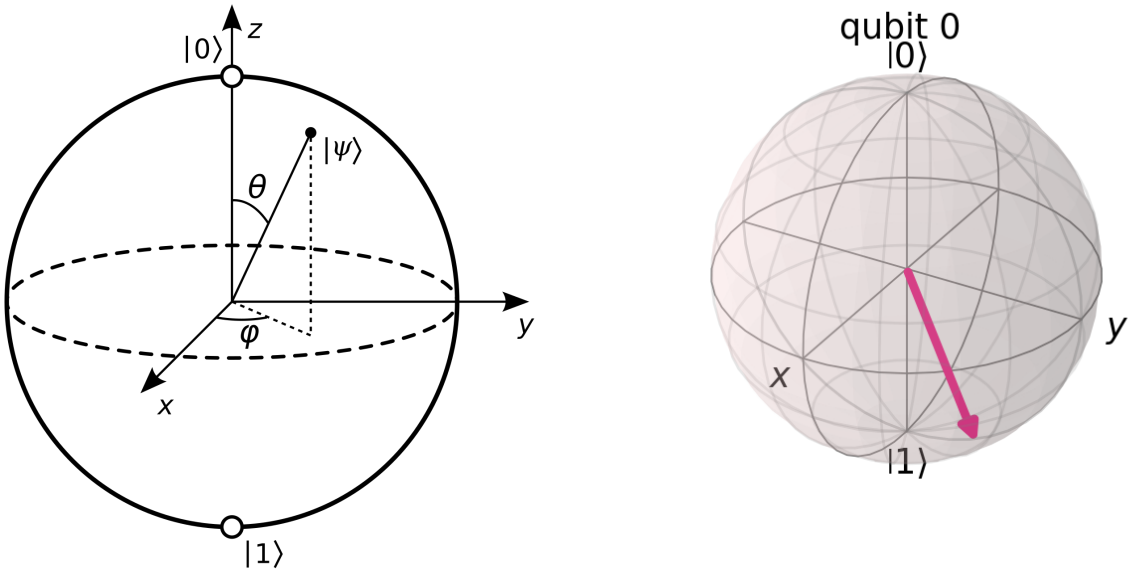
with norm

$$\| \, |\psi\rangle \, \|^2 = \cos^2(\frac{\theta}{2}) + \sin^2(\frac{\theta}{2}) = 1$$

Notice that the angles $\theta \in [0, \pi)$ and $\phi \in [0, 2\pi)$ that determine the qubit's state can be identified geometrically with the polar and azimuthal angles on a sphere as shown in fig: 2.1a.

The geometrical coordinates of Chania is the polar $\theta = 35^o, 31', 0.12''N$, and the azimuthal $\phi = 24^o, 1', 0.12''E$, angles. Then $\theta = 35^o$, $\phi = 24^o$. The qubit Chania, in the basis $\{|0\rangle, |1\rangle\}$:

$$|\texttt{Chania}\rangle = \cos(\frac{35}{2})\,|0\rangle + e^{i24}\sin(\frac{35}{2})\,|1\rangle$$



(a) Block Sphere visualization. (b) Visualization of qubit at state $|\texttt{Chania}\rangle$.

Figure 2.1: Bloch Sphere.

2.4 Fundamental Linear Algebra Concepts

Linear Operators

A linear operator M is a linear transformation $M : V \to W$ between two vector spaces V and W . If $|\psi\rangle$ is a state vector in $\mathbb{C}^2$, then,

$$M|\psi\rangle = M\left(\sum_i c_i |u_i\rangle\right) = \sum_i c_i M|u_i\rangle$$

where $c_i \in \mathbb{C}^2$

Hermitian Conjugate, Hermitian Operator

The Hermitian conjugate, or adjoint (†), is the complex conjugate of a transposed matrix:

$$M^\dagger = (M^*)^T$$

Hermitian operators, also called self-adjoint operators, are operators (or matrices) that are equal to their adjoint (or conjugate transpose):

$$M^\dagger = M$$

Eigenvalues and Eigenvectors

An eigenvector $|u_i\rangle$ of an operator $M \in \mathbb{C}^{n\times n}$ is a non-zero vector that satisfies:

$$M|u_i\rangle = \lambda_i |u_i\rangle$$

where λ_i is the eigenvalue. The number of eigenvalues corresponds to the dimensions of the operator. An operator with $n \times n$ dimensions has n eigenvectors and n corresponding eigenvalues. However, the number of distinct eigenvalues with their corresponding eigenvectors can be less than n. These are found by solving the characteristic equation:

$$det(M - \lambda I) = 0$$

where I is the identity matrix.

Spectral Decomposition

Any linear, square operator $A \in \mathbb{C}^{n\times n}$ can be expressed as:

$$A = V\Lambda V^\dagger$$

where Λ is a diagonal matrix containing the eigenvalues of A on its diagonal and V is a matrix whose columns are the eigenvectors of A. An equivalent expression is:

$$A = \sum_i \lambda_i \left|\lambda_i\right\rangle \left\langle\lambda_i\right|$$

where the eigenvectors $|\lambda_i\rangle$ form an orthonormal set and λ_i are their corresponding eigenvalues.

Commutator

The commutator between two operators A and B is defined as:

$$[A, B] = AB - BA$$

If $[A, B] = 0$, the two operators commute. A crucial property of commuting operators is:

$$[A, B] = 0 \Leftrightarrow e^{A+B} = e^A e^B$$

Projective Operator

A projective operator P, or projector, acts on the state space of the system $|\psi\rangle$ by projecting it onto a particular subspace. This operator is not unitary and affects the state $|\psi\rangle$ irreversibly. An operator is projective if:

$$P^2 = P \quad \text{(Idempotence)}, \quad P^\dagger = P \quad \text{(Hermitian)}$$

Projective operators are mutually orthogonal and form a complete set:

$$\sum_i P_i = I$$

Unitary Operator

A linear operator $U \in \mathbb{C}^{n\times n}$ is unitary if it satisfies:

$$U^\dagger U = UU^\dagger = I$$

A unitary operator can also be expressed as:

$$U = e^{iH}$$

where H is some Hermitian operator.

Tensor Product

The tensor product, or Kronecker product, is a mathematical operation between two tensors (including vectors and matrices). The tensor product between vectors $v \in \mathbb{C}^n$ and $u \in \mathbb{C}^m$ is defined as:

$$v \otimes u = \begin{pmatrix} v_1 u \\ v_2 u \\ \vdots \\ v_n u \end{pmatrix} \in \mathbb{C}^{n \times m}$$

The tensor product between matrices $A_{m \times n}$ and $B_{p \times q}$ is defined as:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix} \in \mathbb{C}^{mp \times nq}$$

where each $a_{ij}\ B$ denotes scalar multiplication of the matrix B by the matrix element a_{ij}.

2.5 Essential Quantum Principles

The wave function

Every physical system has a wave function. For a particle, this wave function is denoted as $\Psi(x, t)$ and depends on its position x at time t. For qubits, the state is represented by a complex vector $|\psi\rangle$ in the Hilbert space. This vector encapsulates all accessible information about the system.

Observables and Operators

Physical observables are described by linear operators, specifically Hermitian operators. Observables are measurable quantities like position, momentum, energy, and angular momentum. This principle stems from the fact that the expectation value of an observable's operator must be real, hence the operator must be Hermitian. For spin qubits, the quantum state of the spin is represented as a three-dimensional vector on the Bloch sphere, with basis $\sigma_x, \sigma_y, \sigma_z$. These operators, which represent observable spin components, are Hermitian. Measuring the spin in any direction will yield either 1 or -1, done by an

apparatus interacting with the system. Think of this apparatus as a black box that can be oriented along each axis to measure the respective spin component.

Measurement and Eigenvalues

When measuring an observable quantity, the result will always be one of the eigenvalues $|\lambda_i\rangle$ of the corresponding operator. The state where we measure λ_i with certainty is the corresponding eigenstate $|\lambda_i\rangle$. After the measurement, the system will be in the eigenstate corresponding to the measured value, often different from its pre-measurement state. This sudden, probabilistic change is called the "collapse of the wave function". For spin qubits, measuring a component of the spin yields only ± 1, implying that the result is always one of the eigenvalues of the spin operators, which are ± 1.

Probability of Measurement Outcomes

If a quantum system is in state $|\psi\rangle$ and we measure an observable M, we will get the eigenvalue λ_i with probability

$$Pr(\lambda_i) = |\langle\psi|\lambda_i\rangle|^2 = \langle\psi|\lambda_i\rangle\langle\lambda_i|\psi\rangle$$

Here, λ_i denotes an eigenvalue and $|\lambda_i\rangle$ the corresponding eigenvector of the Hermitian operator describing M. Because Hermitian operators have real eigenvalues, the possible measurement outcomes are well defined. The product $|\lambda_i\rangle\langle\lambda_i|$ defines a projective operator P_i. This formalism links quantum states and measurements, emphasizing that while all information about a quantum state is encoded in $|\psi\rangle$, it becomes accessible only through measurement—an act that inevitably alters the system. Hence, any computation must be completed before observation.

For spin qubits, the measurable quantities correspond to the three spin components $\sigma_z, \sigma_x, \sigma_y$, each represented by a Hermitian operator. Such operators can be decomposed into mutually orthogonal eigenvectors, each associated with a unique real eigenvalue. When we measure a spin component, we obtain one of these eigenvalues, providing indirect information about the system's pre-measurement state. Importantly, measurement can only yield eigenvector states, but since the eigenvectors of each operator form a complete orthonormal basis, any quantum state can be expressed as a linear combination of them. In particular, the computational basis $|0\rangle, |1\rangle$ introduced earlier corresponds to the eigenvectors of the σ_z operator.

2.6 Time Evolution of Quantum Systems

The time evolution of a closed quantum system is governed by the Schrödinger equation:

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

where $\hbar = \frac{h}{2\pi}$ is the reduced Planck's constant, and $\hat{H}$ is the Hamiltonian of the system. The Hamiltonian is a Hermitian operator describing the system's energy, with eigenvalues corresponding to quantized energy levels. Measuring the system's energy results in one of these eigenvalues.

The solution to the Schrödinger equation with a time-independent Hamiltonian describes how quantum states at different times t_1 and t_0 are connected:

$$|\psi(t_1)\rangle = e^{-iH(t_1-t_0)/\hbar}|\psi(t_0)\rangle$$

A noteworthy aspect of this equation is that the operator governing the evolution of an isolated quantum system is **unitary**. Consequently, the transformations applied to quantum states correspond to unitary operations. In the following section, we explore how this principle forms the foundation for constructing quantum circuits.

2.7 Quantum Gates

For an operator to qualify as a quantum gate, it must be unitary. As previously discussed, a quantum state can be represented as a vector in a multidimensional Hilbert space. Quantum gates act on these states by performing rotations that preserve their norm, ensuring that the associated probabilities remain consistent with the principles of quantum mechanics. Moreover, the time evolution of a closed quantum system must be reversible. In other words, if an operator U maps a state $|s_1\rangle$ to $|s_2\rangle$, its inverse $U^\dagger$ must map $|s_2\rangle$ back to $|s_1\rangle$:

$$U^\dagger U = I$$

which is only possible if the matrix U is unitary. Therefore, every unitary matrix can serve as a quantum gate. A quantum circuit is shown as horizontal wires, each representing a qubit's evolution from its initial to final state. Quantum gates are placed along the wires to indicate the operations applied to the qubits during computation.

$$|\text{initial}\rangle - \boxed{U} - |\text{final}\rangle$$

2.7.1 Single-Qubit Gates

- **Identity Operator**

  The identity operator is the identity matrix that leaves the quantum state $|\psi\rangle$ intact.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad I|\psi\rangle = |\psi\rangle$$

- **Pauli Operators**

  In the case of spin qubits, each quantum state can be expressed as a combination of three measurable components: σ_z, σ_x, and σ_y. These are the Pauli operators, named after Wolfgang Pauli, and are represented by Hermitian matrices. Their matrix forms and corresponding circuit symbols are:

$$X = \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

  – Pauli-X or σ_x flips the state between $|0\rangle$ and $|1\rangle$:

$$X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle$$

  – Pauli-Y or σ_y applies a bit-flip and a phase-flip to the qubit:

$$Y|0\rangle = i|1\rangle, \quad Y|1\rangle = -i|0\rangle$$

  – Pauli-Z or σ_z applies a phase-flip on $|1\rangle$ but leaves $|0\rangle$ intact:

$$Z|0\rangle = |0\rangle, \quad Z|1\rangle = -|1\rangle$$

- **Hadamard Gate**

  Hadamard gate performs a specific linear transformation on a qubit to create a uniform superposition of its basis states. The matrix representation and symbol of the Hadamard gate are:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$$

$$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$$

These states when measured in the computational basis, there is a 50% probability of observing the qubit in the $|0\rangle$ state and a 50% probability of observing it in the $|1\rangle$ state.

- **Rotation Gates**

  Rotation gates rotate qubits around x, y and z axis based on an angle θ. Every matrix R_x, R_y, R_z makes a rotation around the corresponding axis and their matrix representation is as follows:

$$R_X(\theta) = e^{\frac{-i\theta X}{2}} = \begin{bmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}$$

$$R_Y(\theta) = e^{\frac{-i\theta Y}{2}} = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ -\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}$$

$$R_Z(\theta) = e^{\frac{-i\theta Z}{2}} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$

2.7.2 Multi-Qubit Gates

Multi-qubit gates are applied to more than one qubit and can be created by combining single qubit gates, using the tensor product. These gates are classified into two categories: local and non-local operators.

- **Local Operators**

  Local operators are those that can be expressed as a tensor product of single-qubit gates. For instance, consider the following two-qubit operator:

$$(X \otimes Z)\,|10\rangle = X\,|1\rangle \otimes Z\,|0\rangle = |0\rangle \otimes |0\rangle = |00\rangle$$

  We see that the operator can be factorized and that single-qubit gates act on each qubit separately.

- **Non Local Operators**

  Non-local operators are unitary operators that cannot be factorized into a tensor product of single qubit gates. The action of the gate on one qubit is directly controlled by the state of another qubit. The most common class of non-local operators is control gates, which require two qubits: **a control qubit and a target qubit**. If the control qubit is in the $|1\rangle$ state, a unitary operation is

performed on the target qubit. If the control qubit is in the $|0\rangle$ state, no operation is executed on the target qubit. These gates are also known as Controlled-U gates. The unitary matrix representation of these gates is:

$$P_0 \otimes I + P_1 \otimes U$$

where $P_0 = |0\rangle \langle 0|$ and $P_1 = |1\rangle \langle 1|$ are the projector operators in the computational basis.

The most important type of controlled gate is the Controlled-NOT (CNOT) gate, where the unitary operation on the target qubit is the X gate:

$$C_{NOT} = P_0 \otimes I + P_1 \otimes X = \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix}_{4\times4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2.8 Entanglement

Entanglement is one of the most fundamental and fascinating features of quantum mechanics. It arises when the quantum states of two or more qubits become correlated in such a way that the state of each qubit cannot be described independently of the others. Remarkably, this correlation persists even when the qubits are separated by large distances.

In such cases, measuring one qubit immediately determines the state of its partner, regardless of the separation between them. This unique property is a key resource that enables quantum computers to tackle problems beyond the reach of classical machines.

A well-known example of entangled states is provided by the Bell states—a set of four two-qubit states that represent maximal entanglement. These states are defined as follows:

$$|b_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

$$|b_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$

$$|b_{10}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

$$|b_{11}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$

The Bell states constitute a complete basis for the two-qubit Hilbert space and play a central role in numerous quantum information protocols, including quantum teleportation and superdense coding. Entanglement, as exemplified by these states, enables quantum computers to represent and manipulate information in ways unattainable by classical systems. By harnessing the correlations between entangled qubits, quantum algorithms can address specific problems more efficiently, highlighting the computational advantages inherent to quantum mechanics.

2.9 Quantum Walks

A classical *random walk* is a process where a particle moves across the nodes of a graph by choosing a neighboring node at random at each step. Over time, the particle's position follows a probability distribution that tends to spread slowly and eventually converges to a stationary distribution determined by the graph.

A quantum walk is the quantum version of this process. Instead of probabilities, the particle evolves according to quantum amplitudes, allowing it to explore many paths simultaneously. Interference between paths leads to very different dynamics: the walker spreads much faster and does not simply converge like in the classical case.

In this section we describe quantum walks taking place on graphs with an evolution in discrete time steps [1, 2]. Discrete-time quantum walk further includes two specific approaches: coin quantum walk and scattering quantum walk. Coin quantum walk primarily operates on the nodes of a graph and requires a coin space with a dimension equal to the degree of the nodes, making it more suitable for regular graphs. On the other hand, scattering quantum walk encodes all edges of the graph into quantum ground states, eliminating the need for a coin operator.

Let $G = (V, E)$ be a graph, where $V = \{v_1, ..., v_N\}$ denotes the set of nodes and E represents the set of edges. Define $A \in \mathbb{R}^{N\times N}$ as the adjacency matrix of G and $d_i = \sum_{j=1}^{N} A_{i,j}$ the degree of node u_i.

The state obtained after one step of quantum walk evolution from $|i, j\rangle$ is:

$$|i, j\rangle \xrightarrow{U_{QW}} \left(\frac{2}{d_j} - 1\right)|j, i\rangle + \frac{2}{d_j} \sum_{k\neq i,(j,k)\in E} |j, k\rangle$$

after one step of quantum walk, the walker has moved from v_i to v_j (fig: 2.2) . In the next step, starting from v_j , there is a probability of $\left(\frac{2}{d_j} - 1\right)^2$ to return to v_i, and a probability of $\left(\frac{2}{d_j}\right)^2$ to reach another node v_k that is adjacent to v_j but different from v_i.

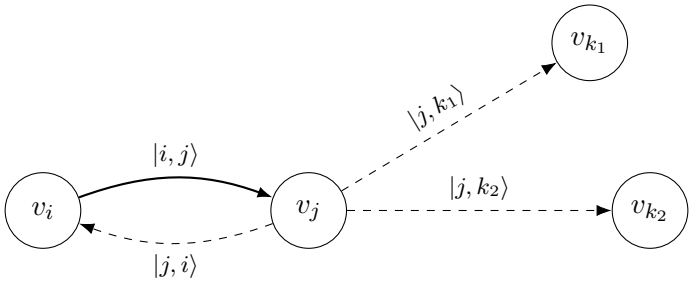


Figure 2.2: Example Scattering Quantum Walk.

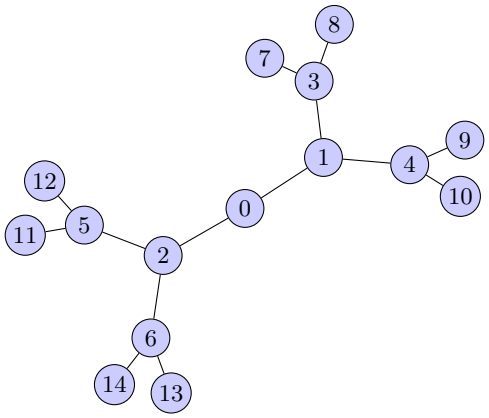
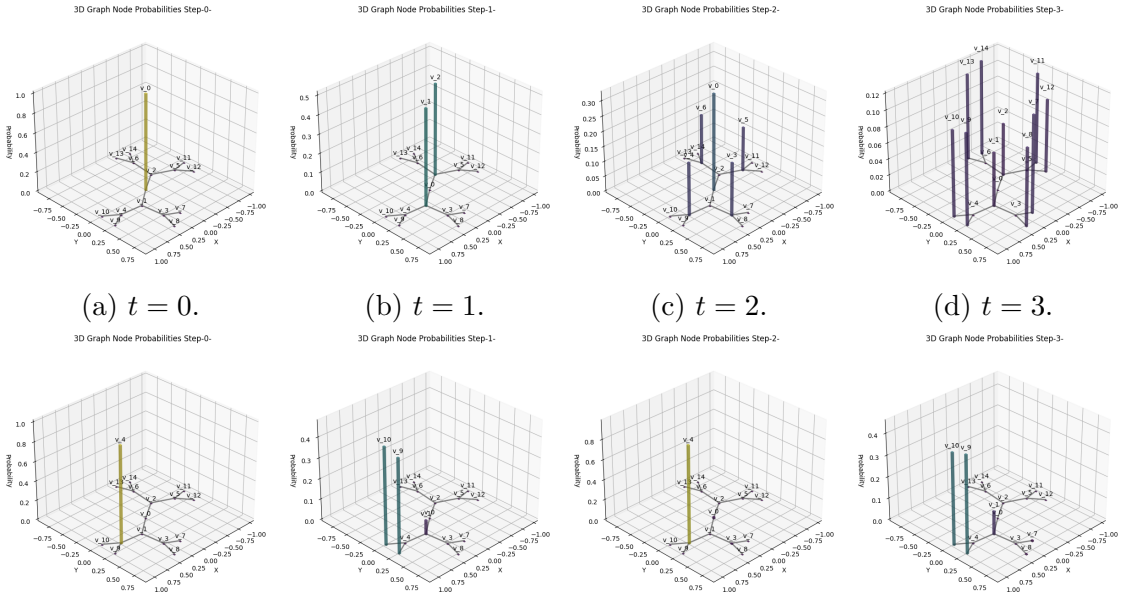


Figure 2.3: Graph with 15 nodes, Balanced tree.

Figure 2.4 illustrates the behavior of quantum walks on the balanced tree graph shown in Figure 2.3. The walk is initialized at different starting points: the central vertex (top row) and a non-central vertex (bottom row). At each time step, the probability distribution evolves across the nodes. When starting from the central vertex, the evolution is symmetric and spreads evenly through the tree structure. In contrast, when starting from a non-central vertex, the resulting distributions are asymmetric, reflecting the graph's topology and the walker's biased access to neighboring nodes. These differences highlight how the initial position strongly influences the dynamics of quantum walks on graphs.



(a) $t = 0$. (b) $t = 1$. (c) $t = 2$. (d) $t = 3$.

(e) $t = 0$. (f) $t = 1$. (g) $t = 2$. (h) $t = 3$.

Figure 2.4: A comparison of the probability distributions obtained from the first three steps of an unbiased quantum walk on a graph tree starting from (a)–(d) a central vertex or (e)–(h) a non-central vertex. The symmetric distribution obtained in the central case is relatively intuitive compared to the distribution obtained from the non-central case.

Chapter 3

Quadratic Unconstrained Binary Optimization Problems (QUBO)

Combinatorial optimization represents a central class of problems in optimization, where the task is to arrange, select, or combine elements from a finite set in order to achieve a defined objective. These problems appear in a wide range of domains, including logistics, finance, telecommunications, and computer science. Typical examples include finding the shortest path in a network, allocating limited resources efficiently, or designing optimal schedules. The defining feature of combinatorial optimization is the exponential growth of possible configurations as the problem size increases, which makes finding exact solutions computationally demanding.

A particularly important subclass is the **Quadratic Unconstrained Binary Optimization (QUBO)** formulation. In QUBO, the problem is expressed in terms of binary decision variables (each taking values 0 or 1) and a quadratic cost function that encodes the objective. Unlike constrained formulations, QUBO problems do not impose additional restrictions on variable interactions, making them flexible but also computationally hard. Despite their wide applicability in practice, efficiently solving QUBO problems remains an open challenge, as they are generally believed to belong to the $\mathrm{FP}^{\mathrm{NP}}$ category of computational complexity. This dual nature-broad relevance and inherent difficulty-explains why QUBO problems are a major focus in both classical and quantum optimization research.

3.1 QUBO formulation

QUBO variables collectively form a binary vector that minimizes a quadratic function. In strict mathematical language, this description can be expressed as follows: Find the optimal bitstring x^* such that:

$$x^* = \underset{x}{\operatorname{argmin}} \quad C(x) := x^T Q x$$

If n_c is the number of variables then $x = (x_1, ..., x_{n_c}) \in \{0, 1\}^{n_c}$ is the binary vector and Q is the $n_c \times n_c$ **"QUBO matrix"**. We can equivalently rewrite this expression:

$$x^* = \underset{x}{\operatorname{argmin}} \quad C(x) := \sum_{i,j} x_i Q_{ij} x_j$$

To verify that an optimal solution has been reached, one can use brute-force enumeration or exact solvers such as Branch-and-Bound or Branch-and-Cut. However, these approaches quickly become infeasible as the number of candidate solutions grows exponentially with problem size, which is often the case in industrial applications. This limitation has motivated the development of classical methods aimed at finding high-quality approximate solutions within reasonable time. Widely used optimization frameworks include Gurobi, CPLEX, and SCIP, while other strategies are based on heuristics and metaheuristics like Tabu search, Path Relinking, evolutionary algorithms, and Simulated Annealing.

3.2 The Ising Model

The Ising model is a mathematical framework in physics, originally introduced by Ernst Ising to study phase transitions in systems of interacting spins arranged on a lattice. In this model, each spin interacts only with its nearest neighbors and can take one of two discrete values, -1 or $+1$. The energy of the system is described by the Hamiltonian, defined as:

$$H_{Ising} = \sum_i \sum_j J_{ij} s_i s_j + \sum_i h_i s_i$$

where s_i represents the spin in the i-th position, h_i represents an external magnetic field affecting spin-i and J_{ij} is the interaction strength between neighboring spins:

- If $J < 0$, then the interaction is ferromagnetic, where the spins tend to align with each other

- If $J > 0$, the interaction is antiferromagnetic, and the spins tend to be opposite
- If $J = 0$, then there is no interaction between the magnetic dipoles

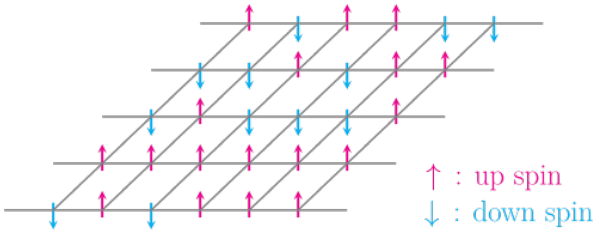


Figure 3.1: Two-dimensional Ising model shown as a lattice of interacting spins. Source.

The Ising model, originally developed in statistical physics to study ferromagnetism, has become a powerful tool for energy minimization in computational science. By representing systems with binary spin variables, the goal is to identify the ground state-the spin configuration that minimizes the Hamiltonian. This directly parallels optimization tasks, where the objective is to find the global minimum of a cost or energy function.

3.2.1 Mapping QUBO Problems onto the Ising Model

The Ising model is particularly relevant in this context due to its direct correspondence with QUBO. By mapping a QUBO problem to an Ising formulation, or vice versa, optimization tasks can be approached through the physical principles underlying spin systems.

The first step involves transforming $x_i \in \{0, 1\}$ to $s_i \in \{-1, +1\}$. This transformation is given by:

$$x_i = \frac{1 + s_i}{2}$$

Then we replace x_i to the cost expression:

$$C(x) = \sum_{i,j} x_i Q_{ij} x_j \quad \xrightarrow{x_i = \frac{1+s_i}{2}}$$

$$C(s) = \sum_{i,j} \frac{1 + s_i}{2} Q_{ij} \frac{1 + s_j}{2}$$

$$= \frac{1}{4} \sum_{i,j} Q_{ij}(s_i + s_j + s_s s_j + 1)$$

$$= \frac{1}{4}\left(\sum_{i,j} Q_{ij}s_is_j + 2\sum_i\sum_j Q_{ij}s_i + \sum_{i,j} Q_{ij} + \sum_i Q_{ii}\right)$$

where, in the last line, we have used $(s_i)^2 = 1$. With $J_{ij} = \frac{Q_{ij}}{4}$, $h_i = \frac{1}{2}\sum_i Q_{ij}$ and an offset of:

$$\texttt{offset} = \frac{1}{4}\left(\sum_{ij} Q_{ij} + \sum_i Q_{ii}\right)$$

which can be safely discarded as it does not change the optimal solution to the problem.

With the mapping to the Ising model established, the next step is to express it in its quantum form. In this framework, the Hamiltonian is promoted to an operator, and the minimization task becomes evaluating the expectation value $\left\langle \hat{H}_{Ising} \right\rangle$. The spin variables $s_i \in \{-1, 1\}$ are then represented by the eigenvalues of an operator acting on the computational basis states $|0\rangle$ and $|1\rangle$. The natural choice for this operator is the Pauli $\hat{\sigma}^z$, leading to the formulation of the Quantum Ising Hamiltonian as:

$$\hat{H}_{Ising} = \sum_i\sum_j J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z + \sum_i h_i\hat{\sigma}_i^z$$

3.3 Adiabatic Theorem and Quantum Annealing

Adiabatic Theorem

The Adiabatic Quantum Theorem states that if a Hamiltonian $H(t)$ evolves slowly enough and maintains a spectral gap, a system initialized in the ground state $|\psi(0)\rangle$ of $H(0)$ will remain in the instantaneous ground state $|\psi(t)\rangle$ throughout the evolution from $H(0) = H_i$ to $H(T) = H_f$.

Quantum Annealing

Quantum annealing encodes an optimization problem into an Ising Hamiltonian H_{Ising} whose ground state represents the solution. The process begins with an easily prepared Hamiltonian H_0, and the system evolves under a time-dependent Hamiltonian:

$$H(t) = (1 - s(t))H_0 + s(t)H_{Ising}$$

where $s(0) = 0$ and $s(T) = 1$. By the Adiabatic Theorem, if the evolution is slow enough, the system remains in the ground state, transitioning from that of H_0 to H_{Ising}. Typically, $H_0 = \sum_{i=1}^n \hat{\sigma}_i^x$ with uniform superposition ground state $\psi_0 = \frac{1}{\sqrt{2^N}}\sum_{x\in\{0,1\}^N} |x\rangle$.

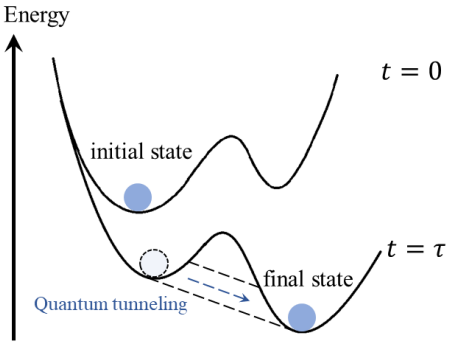


Figure 3.2: Quantum adiabatic evolution in quantum annealing. Source.

A distinctive feature of quantum annealing is quantum tunneling (fig: 3.2), which enables the system to traverse energy barriers and avoid local minima—something classical methods often struggle with. This allows exploration of a broader solution space and improves the likelihood of reaching the global minimum.

The efficiency of quantum annealing is governed by the minimum energy gap $\Delta_{\min}$ between the ground and first excited states during evolution. The annealing time must satisfy:

$$T = \mathcal{O}\left(\frac{1}{\Delta_{min}^2}\right)$$

meaning smaller gaps require slower evolution to maintain adiabaticity, while larger gaps allow faster solutions. Hence, the performance of quantum annealing depends strongly on the problem's energy landscape.

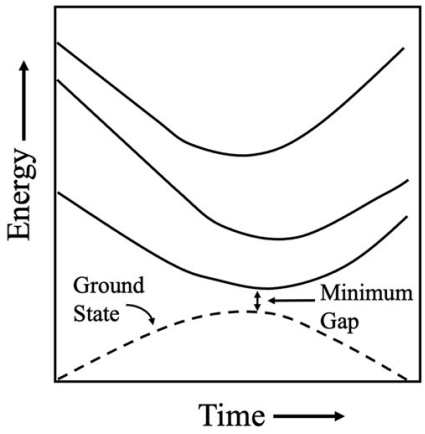


Figure 3.3: **A schematic illustrating quantum annealing as eigenenergies versus time**. The ground state is at the bottom of the energy axis, and the excited states appear above. During the anneal the first excited state approaches the ground state and diverges. The minimum distance between the ground state and the first excited state in the anneal defines the minimum gap. Source.

3.4 Hybrid Quantum-Classical Algorithms

As quantum computing progressed, alternative methods for solving Quadratic Unconstrained Binary Optimization (QUBO) problems began to emerge, particularly those leveraging flexible, gate-based quantum hardware. This shift was motivated in part by the hardware specialization required for Quantum Annealing. In this context, hybrid quantum-classical algorithms gained prominence. These methods integrate quantum and classical computation in a way where the classical component is essential to the algorithm's operation, rather than serving merely as a supplementary tool.

A key family of such methods is the Variational Quantum Algorithms (VQAs), which are inspired by the Rayleigh–Ritz variational principle. VQAs aim to approximate the optimal solution x^* by preparing a quantum state $|\psi^*\rangle$ that, when measured, yields x^* with high probability. This is achieved through a parameterized unitary operator $\hat{U}(\theta)$, known as the ***ansatz***, where the parameters θ are optimized to approximate the target state.

$$|\psi(\theta)\rangle = \hat{U}(\theta)\,|\psi_0\rangle$$

where $|\psi_0\rangle$ is usually taken to be $|0\rangle^{\otimes n_c}$. The algorithm maps the QUBO problem into an *Ising Hamiltonian* and then operates in multiple iterations with the contribution of a classical computer. In each iteration the parameters are adjusted using a classical optimizer, such that the expectation value:

$$\langle\psi(\theta)|\,H_{Ising}\,|\psi(\theta)\rangle$$

gets minimized. If the mapping of the QUBO problem to the Ising model is carried out correctly, the quantum state $|\psi(\theta)\rangle$ that minimizes the expectation value $\langle\psi(\theta)|\,H_{\text{Ising}}\,|\psi(\theta)\rangle$ corresponds to the target state $|\psi^*\rangle$. The structure of the ansatz $U(\theta)$ plays a decisive role in distinguishing between different classes of VQAs and directly influences the quality of the solutions obtained. The selection of an appropriate $U(\theta)$ depends on several considerations, including the feasibility of implementation on available quantum hardware, the ansatz's expressibility-its ability to generate a sufficiently rich set of states—and its reachability, namely its capacity to approximate or reproduce the desired target state within the optimization landscape.

The quantum state generated by the device can be expressed as a superposition of all possible basis states, each corresponding to a classical solution (bitstring):

$$|\psi\rangle = \sum_i a_i\,|x_i\rangle$$

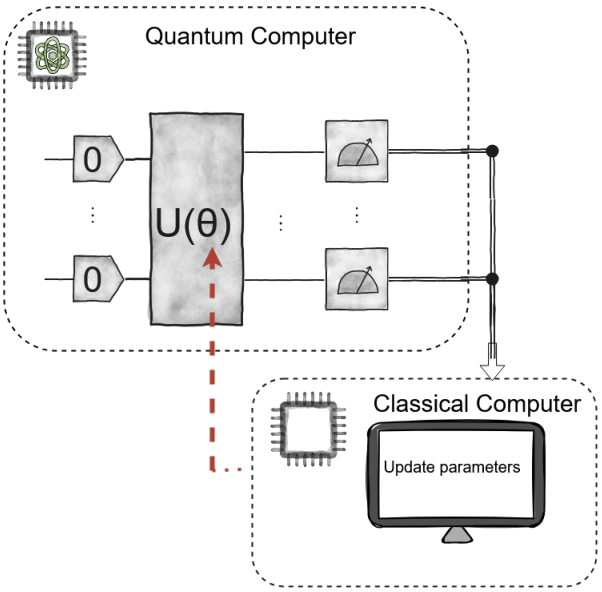


Figure 3.4: **Schematic diagram of a Variational Quantum Algorithm (VQA)**. The inputs to a VQA are: a cost function $C(\theta)$, with θ a set of parameters that encodes the solution to the problem, an ansatz whose parameters are trained to minimize the cost. At each iteration of the loop one uses a quantum computer to efficiently estimate the cost. This information is fed into a classical computer that leverages the power of optimizers to navigate the cost landscape $C(\theta)$ and solve the optimization problem.

where $|x_i\rangle$ is the bitstring and $|a_i|^2$ is the corresponding probability.

To ensure the correct solution is obtained from the superposition, the probability of the optimal bitstring $|a_i^*|^2$ must be close to 1. The angle parameters in VQAs are tuned using classical optimizers, which may be either gradient-based or gradient-free. Several optimizers have been specifically developed for VQAs; however, the intrinsic NP-hardness of the underlying problems remains, meaning that optimizing these parameters is itself still an NP-hard task for classical methods.

3.5 Quantum Approximate Optimization Algorithm (QAOA)

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum–classical approach for solving QUBO problems [3]. Inspired by quantum annealing, it employs Trotterization to approximate the time evolution of a quantum system in discrete steps. Like other variational quantum algorithms (VQAs), QAOA seeks the optimal solution

by preparing a parameterized quantum state. The variational parameters are given by $\gamma = (\gamma_1, \gamma_2, \dots)$ and $\beta = (\beta_1, \beta_2, \dots)$, which define the ansatz $\hat{U}(\beta, \gamma)$ derived from the annealing Hamiltonian through Trotterization.

The Trotterization theorem enables the decomposition of the exponential of non-commuting Hamiltonians, providing the foundation for how QAOA emulates the dynamics of quantum annealing.

Suppose the annealing Hamiltonian:

$$H(t) = A(t)H_0 + B(t)H_{Ising}$$

The time evolution of the quantum state is:

$$|\psi(t)\rangle = e^{-it(A(t)H_0 + B(t)H_{Ising})} |\psi_0\rangle$$

By applying the Trotterization theorem the exponential term can be decomposed and the quantum state is expressed as follows:

$$|\psi(t)\rangle \simeq (e^{itA(t)H_0/n} e^{itB(t)H_{Ising}/n})^n |\psi_0\rangle$$

If we replace the time variables $tA(t)$ and $tB(t)$ with the variational parameters γ and β then we arrive at the QAOA ansatz, written here for higher orders of expansion up to P (circuit layers):

$$|\psi(\beta, \gamma)\rangle = \prod_{p=1}^{P} U_M(\beta_p) U_C(\gamma_p) |\psi_0\rangle$$

where

$$U_M(\beta_p) = e^{-i\beta_p H_0}$$

$$U_C(\gamma_p) = e^{-i\gamma_p H_{ising}}$$

The QAOA ansatz translates the application of the problem and mixing Hamiltonians into quantum gates. Specifically, the operators $e^{-i\gamma_k H_C}$ and $e^{-i\beta_k H_M}$ are implemented using standard quantum gates. For the problem Hamiltonian H_C:

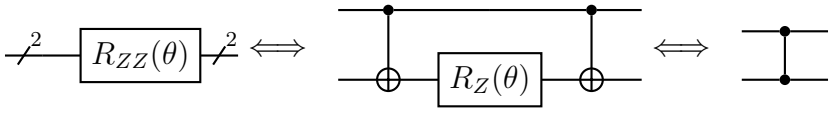
$$\hat{H}_{Ising} = \sum_i \sum_j J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z + \sum_i h_i \hat{\sigma}_i^z$$

The unitary operator $U_C = e^{-i\gamma_k H_C}$ can be implemented using a combination of R_{ZZ} gates and single-qubit R_Z rotations if H_C includes interactions between pairs of qubits and individual qubit terms.

The R_{ZZ} gate represents an entangling gate that acts on two qubits. This gate applies a phase depending on the state of both qubits and is defined as:

$$R_{ZZ}(\theta) = \exp(-i\frac{\theta}{2}\hat{\sigma}^z \otimes \hat{\sigma}^z) \Rightarrow$$

$$R_{ZZ}(2\gamma_k J_{ij}) = \exp(-i\gamma_k J_{ij}\, \hat{\sigma}^z \otimes \hat{\sigma}^z)$$



The R_Z gate represents a single-qubit rotation around the Z-axis and is defined as:

$$R_Z(\theta) = \exp(-i\frac{\theta}{2}\hat{\sigma}^z)$$

$$R_Z(2\gamma_p h_i) = \exp(-i\gamma_p h_i \hat{\sigma}^z)$$

where J_{ij} are the interaction coefficients and h_i are the individual qubit coefficients.

For the mixing Hamiltonian H_M:

$$H_M = \sum_{j=1}^{n} \hat{\sigma}_j^x$$

The unitary operator $e^{-i\beta_k H_M}$ translates to applying single-qubit X-rotations:

$$R_X(\theta) = \exp(-i\frac{\theta}{2}\hat{\sigma}_j^x)$$

$$R_X(2\beta_k) = \exp(-i\beta_k \hat{\sigma}_j^x)$$

By mapping these Hamiltonians into sequences of quantum gates, QAOA harnesses quantum circuits to efficiently navigate the solution space. The variational parameters γ and β are iteratively adjusted to optimize the measurement outcomes, steering the system toward the optimal solution.

3.6 Hardware-Efficient Ansatz

A key limitation of QAOA is that its circuits are tailored to specific problem Hamiltonians, which may not align well with the native gate set of quantum hardware. This mismatch often results in deeper circuits with higher gate counts, making them more vulnerable to noise and decoherence. Moreover, achieving exact solutions typically requires a large number of layers P, further increasing circuit depth.

For near-term quantum devices, where coherence times and gate fidelities are limited, a more practical alternative is the **hardware-efficient ansatz (HEA)**. This parameterized quantum circuit is designed to match the native gate set and connectivity of the hardware [4]. **It generally consists of alternating layers of parameterized single-qubit rotations and entangling gates**, with the entangling gates chosen to reflect the two-qubit operations natively supported by the device (e.g., CNOT or CZ).

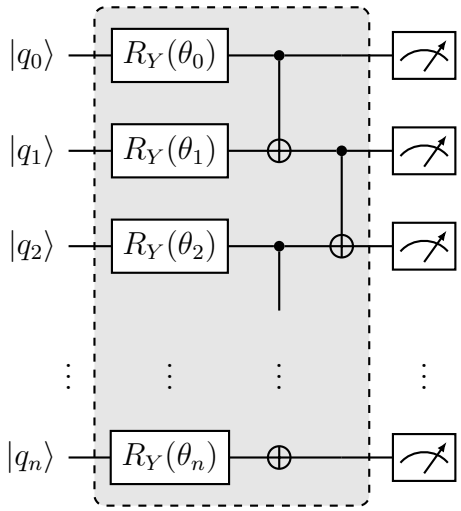


Figure 3.5: Single Layer (R_Y, C_X) Harware Efficient Ansatz.

This structure can be repeated with additional layers to increase the expressibility of the ansatz, allowing it to represent more complex quantum states. However as the circuit depth grows, the unitary transformations become highly random and approach a uniform distribution. The gradient of the cost function becomes exponentially small as the depth increases, so it is difficult to optimize its parameters. This phenomenon is called Barren plateaus.

3.7 The Max-Cut Problem

The maximum cut (*Max-Cut*) problem is a combinatorial optimization problem that involves dividing the vertices of a graph into two disjoint sets such that the number of edges between the two sets is maximized. More formally, given an undirected graph $G = (V, E)$ where V is the set of vertices and E is the set of edges, the Max-Cut problem asks to partition the vertices into two disjoint subsets, S and T, such that the number of edges with one endpoint in S and the other in T is maximized. We can apply Max-Cut to solve various problems including: clustering, network design, phase transitions, etc.

We'll start by creating a problem graph (simple 4-node undirected graph):

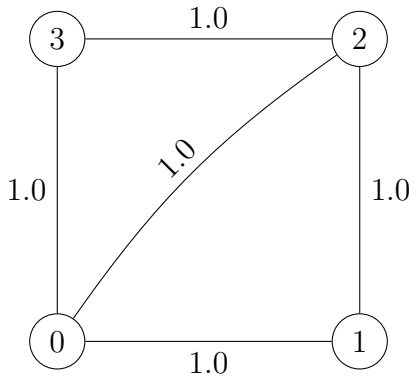


Figure 3.6: Problem Graph for Max-Cut.

This problem can be expressed as a binary optimization problem.

- For each node $0 \leq i \leq n$, where n is the number of nodes of the graph (in this case $n = 4$), we will consider the binary variable x_i. This variable will have the value 1 if node i is one of the groups that we'll label 1 and 0 if it's in the other group, that we'll label as 0.
- We will also denote as w_{ij} (element (i, j) of the adjacency matrix w) the weight of the edge that goes from node i to node j. Because the graph is undirected, $w_{ij} = w_{ji}$.

Then we can formulate our problem as maximizing the following cost function:

$$C(x) = \sum_{i,j=0}^{n} w_{ij} x_i (1 - x_j)$$

$$= \sum_{i,j=0}^{n} w_{ij} x_i - \sum_{i,j=0}^{n} w_{ij} x_i x_j =$$

$$= \sum_{i,j=0}^{n} w_{ij} x_i - \sum_{i=0}^{n} \sum_{j=0}^{i} 2 w_{ij} x_i x_j$$

To solve this problem with a quantum computer/simulator, we are going to express the cost function as the expected value of an observable. However, the observables natively consist of Pauli operators, that have eigenvalues 1 and -1 instead of 0 and 1. That's why we are going to make the following change of variable:

$$z_i = 1 - 2x_i \rightarrow x_i = \frac{1 - z_i}{2}$$

This implies that:

$$x_i = 0 \rightarrow z_i = 1$$

$$x_i = 1 \rightarrow z_i = -1$$

So the new cost function we want to maximize is:

$$C(z) = \sum_{i,j=0}^{n} w_{ij}(\frac{1 - z_i}{2})(1 - \frac{1 - z_i}{2})$$

$$offset - \sum_{i=0}^{n}\sum_{j=0}^{i} \frac{w_{ij}}{2} z_i z_j$$

Instead of maximizing the Cost function, we are going to minimize the following:

$$-C(z) = \sum_{i=0}^{n}\sum_{j=0}^{i} \frac{w_{ij}}{2} z_i z_j + offset$$

Now that the cost function is ready, we can make the analogy with the Pauli-Z operator:

$$z_i = Z_i = \overbrace{I}^{n-1} \otimes \cdots \otimes \overbrace{Z}^{i} \otimes \cdots \otimes \overbrace{I}^{0}$$

It is equivalent to applying the Z gate on the i-th qubit

$$Z(a\,|0\rangle + b\,|1\rangle) = a\,|0\rangle - b\,|1\rangle$$

Finally the Hamiltonian that is extracted from the cost function is:

$$\hat{H} = \sum_{<i,j>} w_{ij} \frac{Z_i Z_j}{2}$$

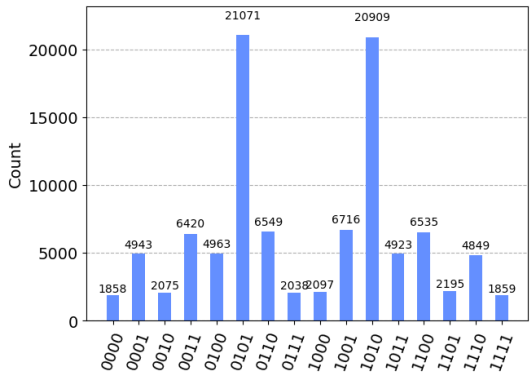
In our example:

$$\hat{H} = IIZZ + IZZI + ZIIZ + ZZII + IZIZ$$

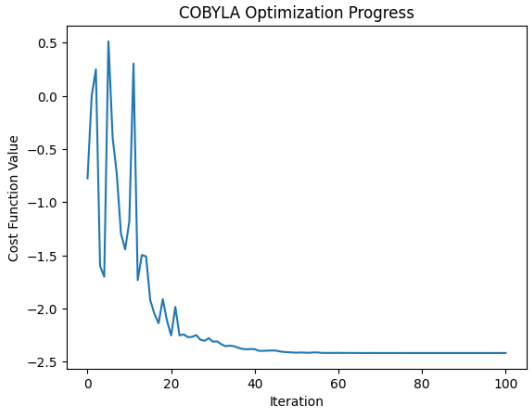
3.7.1 Solving the Max-Cut problem using QAOA and Hardware Efficient VQA

- The initial parameters $(\overrightarrow{\gamma}, \overrightarrow{\beta})$ are randomly selected from the distribution $\mathcal{U}(0, 2\pi)$.
- The classical optimizer is a gradient free optimizer, *COBYLA*

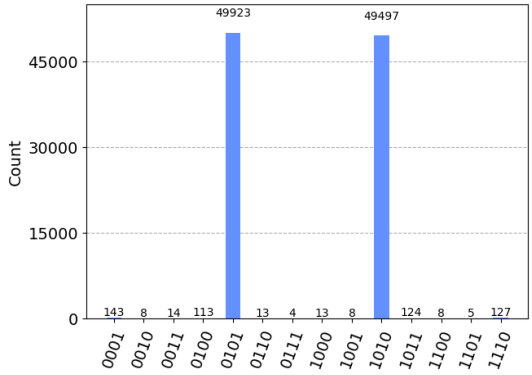
In this work, we implemented both the Quantum Approximate Optimization Algorithm (QAOA) (fig: 3.7) and the Hardware-Efficient Ansatz (HEA) (fig: 3.8) to showcase the application of quantum techniques in solving computational problems.



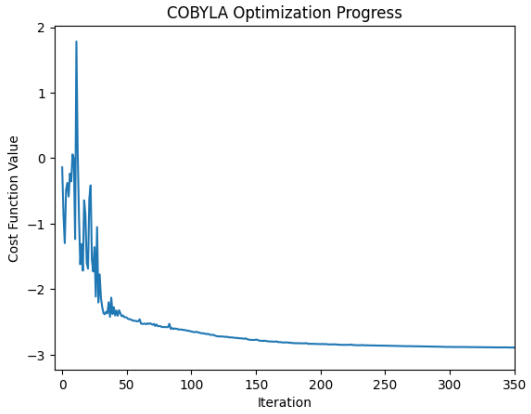
(a) 2-layers QAOA.



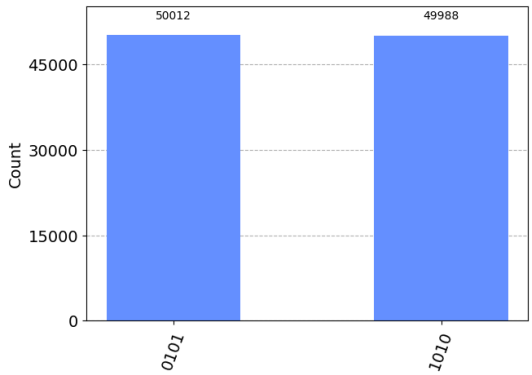
(b) 2-layers QAOA.



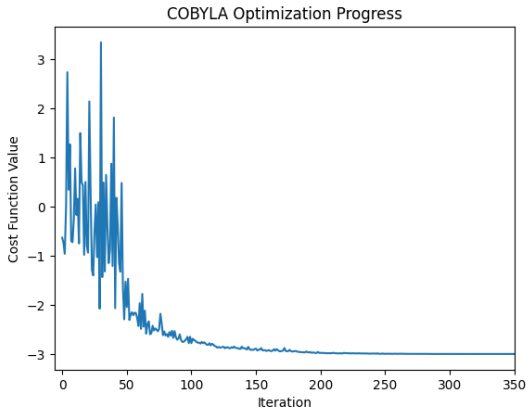
(c) 5-layers QAOA.



(d) 5-layers QAOA.



(e) 10-layers QAOA.



(f) 10-layers QAOA.

Figure 3.7: QAOA results, layers = $\{2, 5, 10\}$. Histogram (left), Cost function (right).

We observe in fig: 3.7 that as the number of layers increases, the solution becomes more accurate, meaning it becomes more likely to measure the two optimal bitstrings "0101" and "1010" and less likely to measure a suboptimal solution. This accuracy comes at the expense of more optimizer iterations, since there are more optimization variables to tune. For comparison, we also run the same problem using a single-layer Hardware Efficient Ansatz in fig: 3.8.

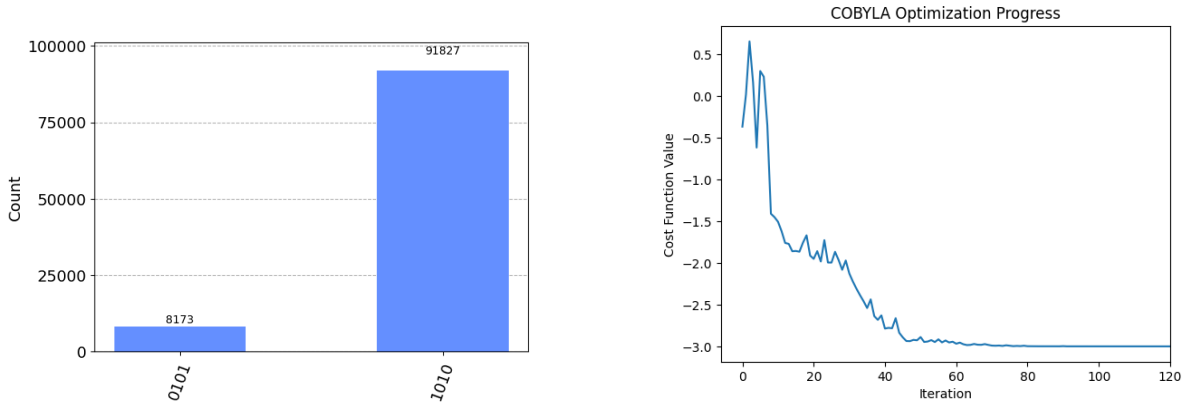


Figure 3.8: HEA 2-layers (VQA) results. Histogram (left), Cost function (right).

We can derive the optimal cut from the optimal solution given by the histogram and divide the nodes into two subgroups. One group contains green nodes and the other blue nodes. It is easily observed that the 2 groups are $\{0, 2\}$ & $\{1, 3\}$.

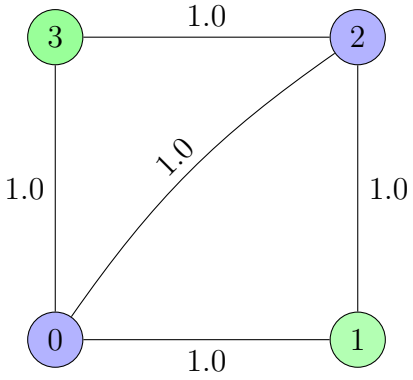


Figure 3.9: Max-Cut solution of the problem graph.

Standard VQA typically targets one of these solutions, whereas QAOA tends to consistently form a superposition of all optimal bitstrings. This behavior of QAOA might be related to its ansatz, which is inspired by adiabatic computing. In

adiabatic computing, the process starts from a uniform superposition state, naturally favoring superposition solutions.

3.8 Adaptive Derivative Assembled Problem Tailored-Quantum Approximate Optimization Algorithm (ADAPT-QAOA)

In QAOA the performance improves with the number of alternating layers in the ansatz, but the latter is limited by coherence times in existing and near-term quantum processors. Moreover, more layers implies more variational parameters, which introduces challenges for the classical optimizer. There are various studies that suggest an alternating way to build ansatzes for solving combinatorial problems [5], [6]. In the case of QAOA an adaptive approach called *ADAPT-QAOA* [7] describes an algorithm, based on QAOA, that results to faster convergence and reduction in entangled gates.

3.8.1 Framework

In QAOA the variational ansatz consists of p layers, each containing the cost Hamiltonian H_c and a mixer, H_M :

$$|\psi(\overrightarrow{\gamma}, \overrightarrow{\beta})\rangle = (\prod_{k=1}^{p}[e^{-iH_M\beta_k}e^{-iH_C\gamma_k}])\,|\psi_{ref}\rangle$$

Where $|\psi_{ref}\rangle = |+\rangle^{\otimes n}$, n is the number of qubits. The initial state is prepared by applying Hadamard gate to all the n qubits.

The variational parameters γ_k & β_k are chosen such that the $\langle\psi_p(\gamma,\beta)|\,H_C\,|\psi_p(\gamma,\beta)\rangle$ is minimized, then the resulting energy and state provide an approximate solution to the optimization problem. The accuracy of the result and the efficiency with which it can be obtained depend sensitively on H_M . **In the standard QAOA ansatz**, the mixer is chosen to be a single-qubit X rotation applied to all qubits. **In ADAPT-QAOA the ansatz** consists of the standard cost Hamiltonian H_C and the mixing Hamiltonian is replaced by one operator extracted from an *operator pool*.

3.8.2 Operator Pool

The mixer operator is selected from a collection of operators. In order to define this collection, suppose **a set of qubits Q**. The mixer operator of the standard QAOA is a

single operator, thus the pool for the standard QAOA is $P_{QAOA} = \sum_{i \in Q} X_i$. If we want to enrich our collection, we need to include two additional operator pools. one consisting entirely of single-qubit mixers, and one with both single-qubit and multi-qubit entangling gates:

- $P_{QAOA} = \sum_{i \in Q} X_i$
- $P_{Single} = \cup_{i \in Q} \{X_i, Y_i\} \cup \{\sum_{i \in Q} Y_i\} \cup P_{QAOA}$
- $P_{Multi} = \cup_{i \times j \in Q \times Q} \{B_i C_j | B_i, C_j \in \{X, Y, Z\}\} \cup P_{single}$

Each operator pool contains $O(1)$, $O(n)$ and $O(n^2)$ elements respectively.

3.8.3 Optimization Process

The quantum ansatz is grown one layer at a time using a gradient-based selection criterion. At each iteration k, operators A_j from the predefined mixer pool are evaluated via the energy gradient:

$$-i \left\langle \psi^{(k-1)} \right| e^{i\gamma_0 H_C} [H_C, A_j] e^{-i\gamma_0 H_C} \left| \psi^{(k-1)} \right\rangle$$

where H_C is the cost Hamiltonian, γ_0 is the initial variational parameter of the problem Hamiltonian. The operator A_k with the largest gradient norm is selected and appended to the circuit as $e^{-\beta_k A_k} e^{-\gamma_k H_c}$, producing the updated variational state:

$$\left| \psi^k \right\rangle = e^{-\beta_k A_k} e^{-\gamma_k H_C} \left| \psi^{(k-1)} \right\rangle$$

After each layer is added, all variational parameters $\{\beta_m, \gamma_m\}$ are re-optimized to minimize the cost function $\langle \psi(k) | H_c | \psi(k) \rangle$. The iterative process continues until the gradient norm falls below a predefined threshold.
The steps are the following:

1. Initialize circuit, apply Hadamard gates
2. Initialize the operator pool **A**
3. Compute the gradient for each operator $-i \left\langle \psi^{(k-1)} \right| e^{i\gamma_0 H_C} [H_C, A_j] e^{-i\gamma_0 H_C} \left| \psi^{(k-1)} \right\rangle$
4. Append the operator with the largest gradient as $e^{-\beta_k A_k} e^{-\gamma_k H_c}$
5. The updated state is $\left| \psi^k \right\rangle = e^{-\beta_k A_k} e^{-\gamma_k H_C} \left| \psi^{(k-1)} \right\rangle$
6. Optimize all the parameters $\{\beta_m, \gamma_m\}_{m=1}^k$
7. Return to step 3, until max gradient $<$ Threshold.

Example Output

```
--- ADAPT Step 1 ---
Gradient norm: 0.477012
Adding operator SparsePauliOp(['XIXI'], coeffs=[1.+0.j])
    with gradient 0.160171
Step 1 minimized energy: -0.9999986078285611
--- ADAPT Step 2 ---
Gradient norm: 0.085417
Converged!
```

Table 3.1: Console output after executing ADAPT-QAOA algorithm with the Hamiltonian from equation 3.1.

3.8.4 Max-Cut with ADAPT-QAOA

Suppose we have the graph from the previous section. The solution leads to the two groups {0,2} & {1,3}. The Hamiltonian is equal to:

$$\widehat{H} = ZZII + IZZI + ZIIZ + IZIZ + IIZZ \tag{3.1}$$

- We set the threshold to stop the optimization equal to $thres = 0.1$
- The initial γ value is $\gamma_0 = 0.01$
- The operator pool contains 64 elements

Executing the algorithm, the output is in table: 3.1 and the partitioned graph is shown below:

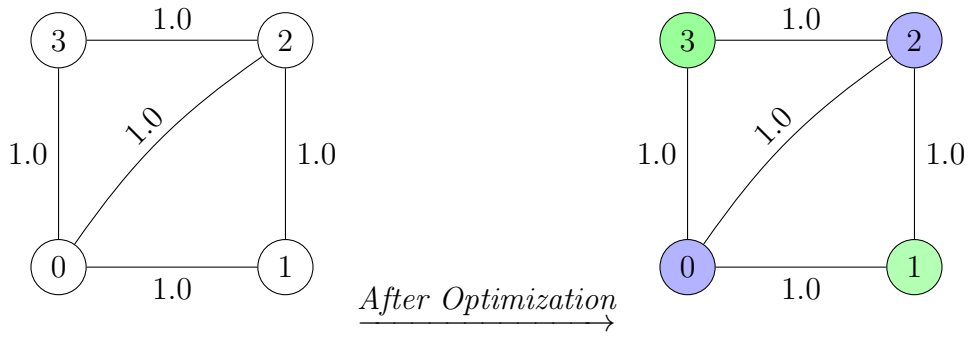
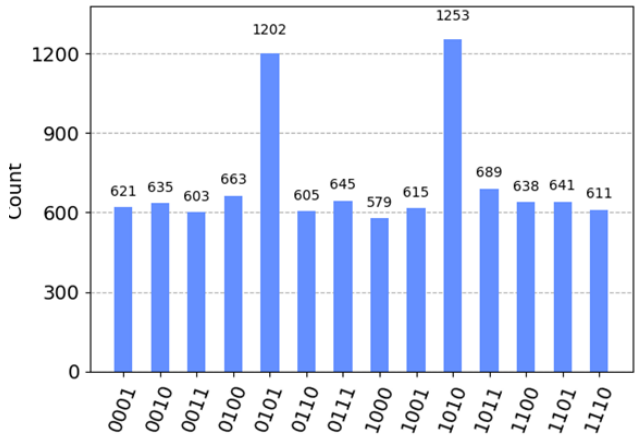


Figure 3.10: Reminder Max-Cut problem and solution.

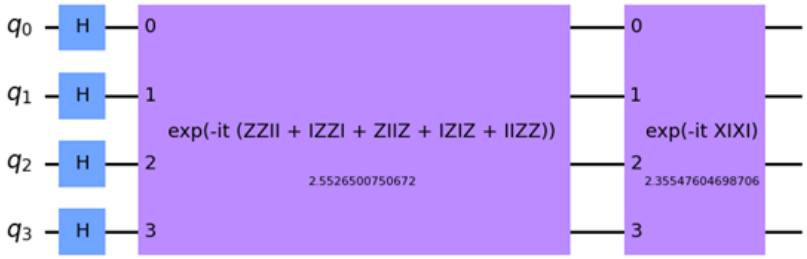
The results are shown in fig: 3.11. As we can see, we were lucky and the algorithm converged after 1 iteration. After the first step the algorithm found the best operator

$(XIXI)$ and in the next iteration the max gradient was lower than the predefined $threshold = 0.1$ thus the optimization stopped. The histogram shows the probabilities distribution and the optimal states.

The top bitstings 0101 & 1010 indicate that the nodes should be divided to these as shown below: **(green and blue group)**



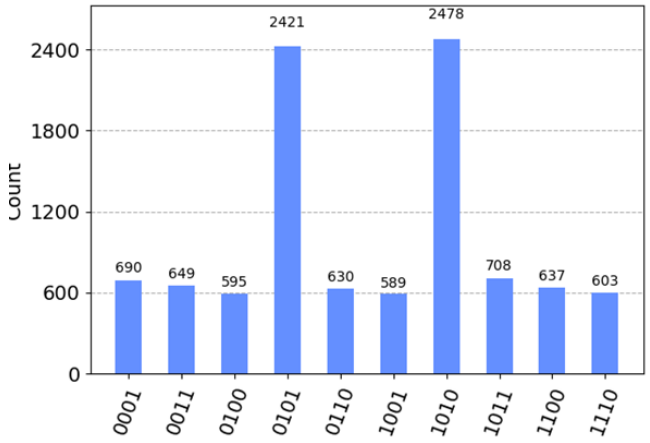
(a) Probability distribution after measuring the ansatz below.



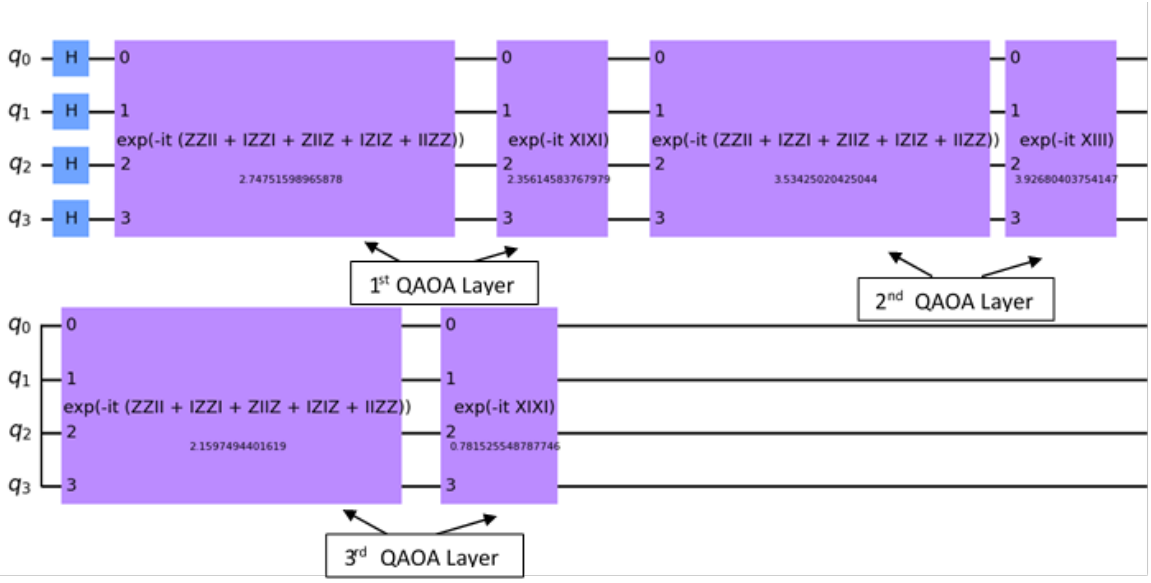
(b) Iteratively grown ansatz. 1-layer with mixing hamiltonian chosen from the P_{multi} operator pool.

Figure 3.11

If we are not lucky and the process requires more optimization runs to converge then as we can see in fig: 3.12 the optimal circuit would require more layers but the optimal bitstrings are separated more clearly from any other solution.



(a) Probability distribution after measuring the ansatz below.



(b) Iteratively grown ansatz. 3-layer with mixing hamiltonians chosen from the P_{multi} operator pool.

Figure 3.12

3.8.5 Computational Challenges of ADAPT-QAOA

While ADAPT-QAOA shows promise as a systematic way to construct efficient variational ansätze, its computational demands make it challenging for both classical simulation and current quantum processors. On a classical laptop, the repeated optimization cycles and exponential scaling of statevector simulations restrict feasible experiments to relatively small qubit counts. On real NISQ devices, the growing circuit depth, noise sensitivity, and need for multiple evaluations per optimization step make large-scale implementations

impractical at present. Therefore, ADAPT-QAOA remains largely a research tool, requiring high-performance computing resources for classical studies and future fault-tolerant quantum hardware for practical deployment.

3.9 Qubit Compression Strategies for Binary Optimization

This section summarizes the material from prior work on qubit compression and reduction [8–10].

Each possible solution of a QUBO problem $x_i \in \{0,1\}^{n_c}$ is represented by a quantum basis state $|x_i\rangle$, resulting in a quantum space of 2^{n_c} basis states. The parameterized quantum state produced by the VQA circuit can be described as:

$$|\psi(\theta)\rangle = \sum_{i=1}^{2^{n_c}} a_i(\theta)\,|x_i\rangle$$

Here, $a_i(\theta)$ denotes the amplitude of $|x_i\rangle$. By linking each classical solution x to a basis state $|x\rangle$, the quantum state $|\psi(\theta)\rangle$ encodes all possible solutions in a linear superposition. This inherent quantum parallelism enables simultaneous exploration of many classical solutions and serves as a key motivation for applying quantum algorithms to classical problems.

Instead of using our qubits to map the entire set of classical variables, we could use them to map a smaller subset of classical variables and somehow identify which subset we used. In the novel encoding scheme developed in [8], it is demonstrated a significant reduction in the number of qubits needed for QUBO problems.

The realization of this idea begins by dividing n_c binary variables into R disjoint subgroups of equal size n_a, so that $R = \frac{n_c}{n_a}$. Each subgroup is encoded using n_a ancilla qubits, while $n_r = \lceil \log_2(R) \rceil$ register qubits specify subgroup addresses via binary encoding, yielding a total of $n_q = n_a + n_r$ qubits. This encoding reduces qubit requirements compared to full encoding while still capturing correlations within subgroups. The resulting quantum state is

$$|\psi(\theta)\rangle = \sum_{r}^{R} \beta(\theta) \left(\sum_{i=1}^{2^{n_a}} a_r^i(\theta)\,|x_a^i\rangle \right) \otimes |r\rangle$$

where $|x_a\rangle$ and $|r\rangle$ are the basis states spanned by the n_a ancilla qubits and the n_r register qubits, respectively.

There are two special cases that arise within this encoding scheme. One of them is the full or complete encoding is when the number of ancillas is equal to the number of classical variables $n_a = n_c$, which is the same encoding used in quantum QUBO solvers such as Quantum Annealing and QAOA, where the problem is mapped to an Ising Hamiltonian. Another special case to consider is when $n_a = 1$, where each subgroup contains only one binary variable, resulting in $R = n_c$ subgroups. In this scenario, the total number of qubits required is $n_q = 1 + \lceil \log_2(n_c) \rceil$. This configuration achieves the highest possible reduction in the number of qubits required by this encoding scheme, which is exponential compared to the original full encoding. This approach, called *minimal encoding*, is limited to representing distributions of statistically independent variables, since only n_c coefficients encode a distribution over 2^{n_c} solutions. Still, it illustrates the essence of the general strategy.

By adding ancilla qubits, one can capture n_a-body correlations between variables, where the value of one variable depends on another. In the full encoding, the state can represent the complete n_c-body correlation structure of the problem.

3.9.1 Adaptation to QUBO Problems

The cost function for solving QUBO problems is:

$$C = \sum_{i=1}^{2^{n_c}} Pr(x_i) x_i^T Q x_i$$

In this approach, the cost function is straightforward since each x_i corresponds to the entire bitstring $|x_i\rangle$, directly representing a complete classical solution. By contrast, a qubit-efficient scheme requires a different process: here, the generated bitstrings encode only partial information rather than the full solution. To reconstruct the complete solution, one must collect an ancilla measurement from each register and concatenate them. Thus, a state like $|x_a^i\rangle |r\rangle$ represents only a fragment of the classical solution. The cost function for qubit-efficient formulations is then defined accordingly:

$$\begin{aligned} C(\theta) &= \textstyle\sum_x Pr_\theta(x) x^T Q x \\ &= \textstyle\sum_x Pr_\theta(x) \sum_{i,j=1}^{n_c} x_i Q_{ij} x_j \\ &= \textstyle\sum_{i,j=1}^{n_c} Q_{ij} \sum_{\{x | x_i=1, x_j=1\}} Pr_\theta(x) \\ &= \textstyle\sum_{i,j=1}^{n_c} Q_{ij} P_{i,j}^{1,1}(\theta)(1-\delta_{ij}) + \sum_{i=1}^{n_c} Q_i P_i^1(\theta) \end{aligned}$$

Here, x denotes the full classical bitstring obtained by concatenation. For a given pair x_i and x_j, setting both equal to 1 restricts the system to the subspace where these variables take fixed values, leaving 2^{n_c-2} possible configurations. The term $P_{i,j}^{1,1}$ gives the probability of both x_i and x_j being 1, while P_i^1 refers to the probability of x_i alone being 1. This formulation of the cost function is general and can be adapted to any encoding scheme, though the exact computation and translation of probabilities depend on the chosen encoding.

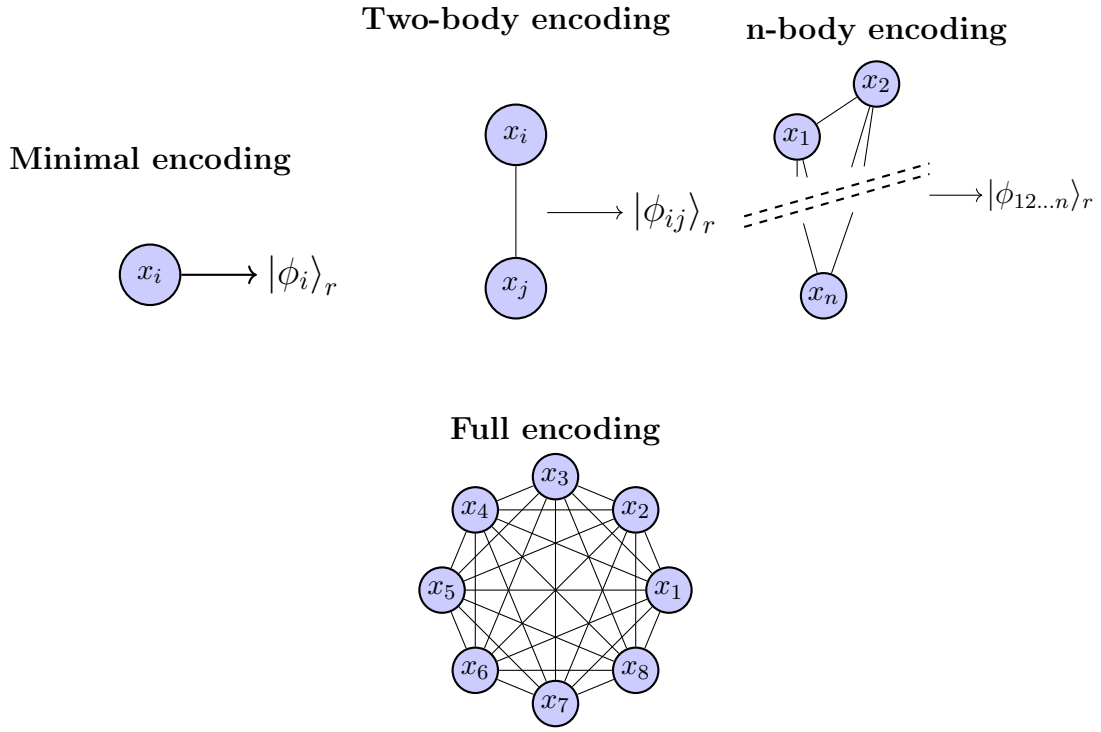


Figure 3.13: **Schematic representation of the encoding schemes**. In the minimal encoding, each of the 2^{n_r} basis states $|\psi_i\rangle$ is used to represent a single classical variable x_i (vertex). In the n-body (two-body) encoding scheme, groups of n (two) classical variables are formed and each basis state represents a unique encoded group. In the full encoding (fully connected graph), each basis state represents an entire graph.

3.9.2 The Minimal Encoding

The minimal encoding represents the extreme case where the ancilla qubit is $n_a = 1$, so each subgroup contains only one variable. This encoding scheme sacrifices the ability

to capture correlations but allows for problem sizes to be scaled exponentially with the number of qubits. There are $R = n_c$ subgroups, the register qubits are $n_r = \log_2 R = \log_2 n_c$, consequently it requires $n_q = 1 + \log_2 n_c$ qubits to represent n_c classical variables. The quantum state can be expressed as:

$$|\psi(\theta)\rangle_{n_a=1} = \sum_{1}^{n_c} \beta_i(\theta)[a_i(\theta)\,|0\rangle_a + b_i(\theta)\,|1\rangle_a] \otimes |\phi_i\rangle_r$$

where $|\phi_i\rangle_r$ and $\{|0\rangle_a, |1\rangle_a\}$ are computational basis states of the register (ancilla) qubits. The probability of the ith classical variable to have the value 1 or 0 is given by $Pr(x_i = 1) = |b_i|^2$ and $Pr(x_i = 0) = 1 - |b_i|^2 = |a_i|^2$ respectively. The coefficients $\beta_i(\theta)$ capture the likelihood of measuring each register state $|\phi_i\rangle_r$.

In the minimal encoding scheme, we can not capture correlations between different classical binary variables since they are all part of a different ancilla group. Therefore we have to consider them independent.

$$P^{1,1}_{i,j}(\theta) = Pr(x_i = 1)Pr(x_j = 1) = |b_i|^2|b_j|^2 \;\text{ if } i \neq j$$

$$P^{1}_{i}(\theta) = Pr(x_i = 1) = |b_i|^2 \;\text{ if } i = j$$

Thus, the cost is expressed as:

$$C(\theta) = \sum_{i,j=1}^{n_c} Q_{ij} P^{1,1}_{i,j}(\theta)(1 - \delta_{ij}) + \sum_{i=1}^{n_c} Q_{ii} P^{1}_{i}(\theta) \Leftrightarrow$$

$$C(\theta) = \sum_{i,j=1}^{n_c} Q_{ij}|b_i|^2|b_j|^2(1 - \delta_{ij}) + \sum_{i=1}^{n_c} Q_{ii}|b_i|^2(\theta)$$

Testing the Minimal Encoding

To demonstrate the minimal encoding scheme, we will apply it to the Subset Sum problem. We are given a vector **a** and a constant S. The objective is to find a subset of the elements in a such that the sum of the subset equals S.

$$a = \begin{bmatrix} a1 \\ a2 \\ a3 \\ a4 \end{bmatrix} \begin{matrix} \searrow \\ \nearrow \end{matrix} S$$

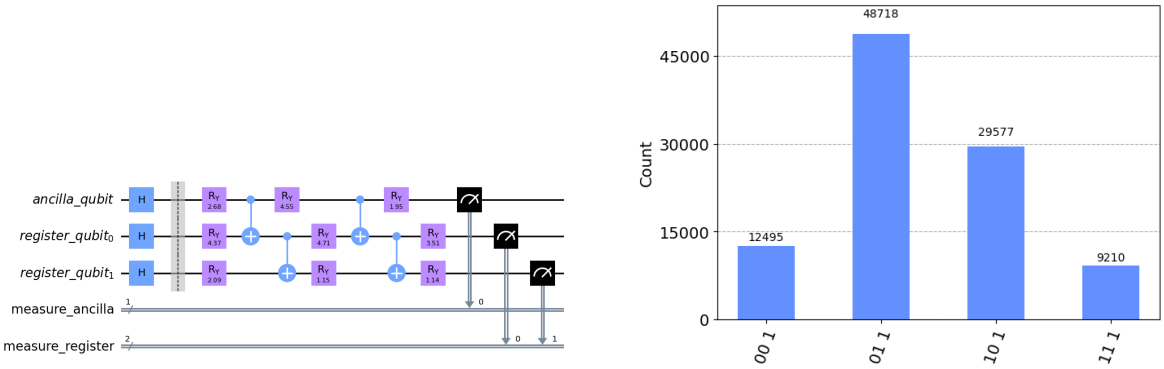
The goal is to find the optimal parameters that minimize the cost function:

$$\underset{x}{\text{minimize}} \quad C(x) = (a^T \cdot x - S)^2$$

where x is a binary vector with the same length as a.

Every encoding type will be represented with a Hardware Efficient Ansatz and the QAOA will be consisted of 10 layers. Suppose a simple subset sum problem with $a = [1, 1, 1, 1]$ and $S = 4$. We have $n_c = 4$, thus $n_q = 1 + \log_2(n_c) = 1 + 2 = 3$ qubits. The obvious solution is to use all the elements from the array 'a' so the x vector must be equal to $x = [1, 1, 1, 1]$.

The optimal solution should be one that has the ancilla qubit measured at state 1 for each possible register state (fig: 3.14). That means taking into account all variables from vector a to sum up to S



(a) Final Circuit for Minimal Encoding 3 qubits.

(b) Measurement results.

Figure 3.14: Results from the subset sum quantum experiment.

By executing 5 experiments, each starting the optimization from a different initial point, we observe the differences between the QAOA, HEA and Compressed HEA regarding the convergence of the cost function. The dashed lines are the mean value of the 5 runs, the shaded regions show the standard deviation from the mean (fig: 3.15):

We observe in fig: 3.15 that all three implementations converge to a cost function value of 0, indicating that they are finding the optimal solution. The significant outcome here is that the minimal encoding appears to work correctly, thereby opening the door to testing it in larger scale experiments alongside other encoding schemes.

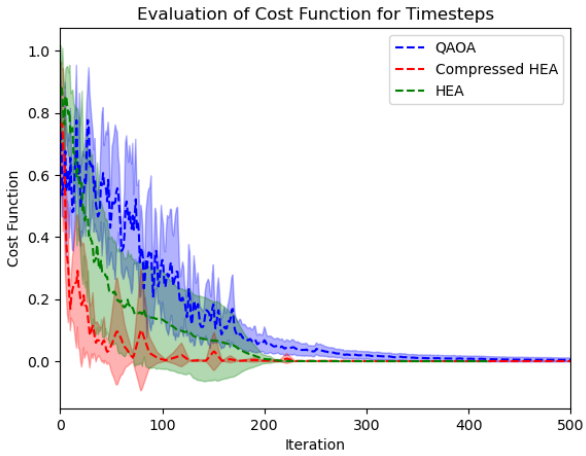


Figure 3.15: Convergence of the cost function for QAOA, HEA, and Compressed HEA.

3.9.3 Two-body Encoding

Multiple ancilla qubits can be included within a group. While this requires more qubits compared to minimal encoding, it enables the capture of correlations between variables. For clarity, we illustrate the case of using two ancilla qubits per register subgroup.

In the general case where a variable can be contained in more than one pair:

$$|\psi(\theta)\rangle_{n_a=2} = \sum_{(i,j)\in P}^{R} \beta_{ij}(\theta)[a_{ij}(\theta)\,|00\rangle_a + b_{ij}(\theta)\,|01\rangle_a + c_{ij}(\theta)\,|10\rangle_a + d_{ij}(\theta)\,|11\rangle_a] \otimes |\phi_{ij}\rangle_r$$

where P is the set of pairs, $\{|\phi_{ij}\rangle_r\}$ and $\{|00\rangle_a, |01\rangle_a, |10\rangle_a, |11\rangle_a\}$ are the computational basis states of the register and ancilla qubits, respectively. Here, each register $|\phi_{ij}\rangle_r$ corresponds to a pair of classical variables x_i as shown in Fig: 3.13.

In the simplified case where all pairs are disjoint, there are $R = n_c/2$ subgroups, and it requires $n_q = 2 + \log_2(n_c/2)$ qubits to represent n_c binary variables. The quantum state is expressed as follows:

$$|\psi(\theta)\rangle_{n_a=2} = \sum_{r=1}^{R} \beta_r(\theta)[a_r(\theta)\,|00\rangle_a + b_r(\theta)\,|01\rangle_a + c_r(\theta)\,|10\rangle_a + d_r(\theta)\,|11\rangle_a] \otimes |r\rangle_r$$

From this state, we can derive that the probability of the i-th classical variable, if it belongs to the ancilla group of the r-th register, taking the value 1 is given by:

$$Pr(x_i = 1) = \begin{cases} |c_r(\theta)|^2 + |d_r(\theta)|^2 & : \text{ if } i \text{ is the first bit of register } r \\ |b_r(\theta)|^2 + |d_r(\theta)|^2 & : \text{ if } i \text{ is the second bit of register } r \end{cases}$$

To calculate the probabilities $P_{i,j}^{1,1}$ and P_i^1, we need to distinguish the cases where the i-th and j-th variables are in the same ancilla group or in different subgroups:

$$P_{i,j}^{1,1}(\theta) = \begin{cases} Pr(x_i = 1)Pr(x_j = 1) & : \text{ if } i,j \text{ are in different ancilla groups} \\ |d_r(\theta)|^2 & : \text{ if } i,j \text{ are in the same ancilla group} \end{cases}$$

$$P_i^1(\theta) = Pr(x_i = 1)$$

3.9.4 Testing the Efficient Encoding Scheme

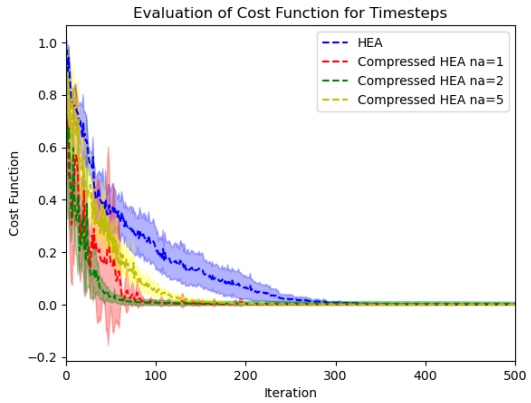
We continue with a series of larger-scale experiments to examine how the length of the ancilla affects the convergence rate and accuracy of the solution We perform 4 experiments. Each experiment has a different number of classical variables $n_c = \{10, 20, 50, 100\}$ with a number of ancilla qubits $n_a = \{1, 2, 5\}$, and for each case, we plot the results for HEA and compressed HEA. Every circuit had 3 layers (ry, cx). The subset sum problem remains the same each time:

$$a = [\overbrace{1, ..., 1}^{n_c}]$$

$$S = n_c$$

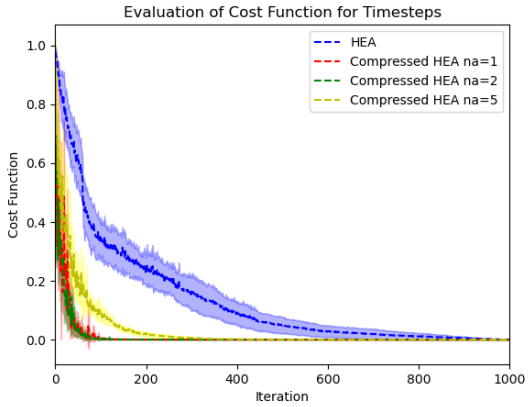
The experiments with ancilla length [50, 100] (fig: 3.16c, 3.16d) will undergo some changes. The HEA for the full encoding will be replaced with a Compressed HEA with $n_a = 10$. Even with 20 qubits, the full encoding was already computationally expensive.

In Figure 3.16, we observe that as problem size grows, encoding schemes with fewer ancilla qubits (and thus fewer total qubits) converge faster and more accurately than those with larger ancilla counts. **This highlights the strength of efficient encodings in tackling instances that would be infeasible for conventional VQAs**. Moreover, the superior performance of smaller ancilla sizes ($n_a = 1, 2$) over larger ones ($n_a = 5, 10$) is not immediately intuitive, since higher ancilla counts should, in principle, capture more correlations and yield better accuracy. We suspect this counterintuitive result arises from the exponential growth of the Hilbert space with n_a, combined with the increased measurement demands. **With more ancillas, significantly more shots are required for reliable statistics**. For example, in minimal encoding, failing to measure a register still leaves a 50% chance of assigning the correct value, while in the $n_a = 10$ case, the chance drops to $1/2^{10}$. Although $n_a = 10$ appears worse here, we do not expect performance to always degrade with larger ancilla sizes. Instead, an optimal trade-off likely exists, and results may improve as n_a approaches the full encoding limit.



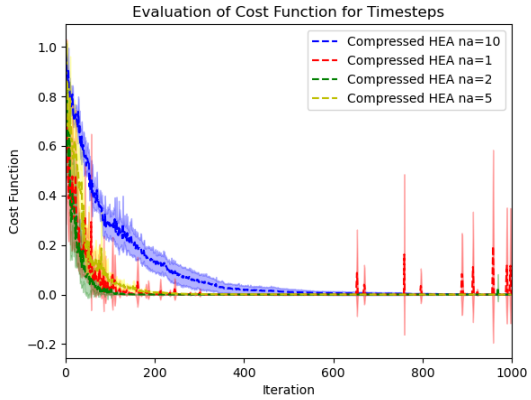
(a) **Subset Sum problem $n_c = 10$.**

- 10 qubits for full encoding
- 6 qubits using $n_a = 5$
- 5 qubits using $n_a = \{1, 2\}$



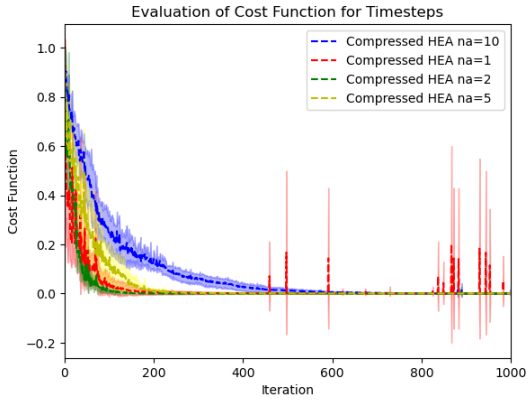
(b) **Subset Sum problem $n_c = 20$.**

- 20 qubits for full encoding
- 7 qubits using $n_a = 5$
- 6 qubits using $n_a = \{1, 2\}$



(c) **Subset Sum problem $n_c = 50$.**

- 13 qubits using $n_a = 10$
- 9 qubits using $n_a = 5$
- 7 qubits using $n_a = \{1, 2\}$



(d) **Subset Sum problem $n_c = 100$.**

- 14 qubits using with $n_a = 10$
- 10 qubits using $n_a = 5$
- 8 qubits using $n_a = \{1, 2\}$

Figure 3.16: Experimental results for the Subset Sum problem with varying number of classical variables (n_c).

Chapter 4

Quantum Assisted Generative AI

4.1 Introduction to Language models & Transformers

The emergence and rapid advancement of Large Language Models (LLMs) such as ChatGPT has had a significant global impact, revolutionizing our interactions with artificial intelligence and expanding our understanding of its capabilities.
GPT's architecture is a multi-layer decoder-only Transformer (fig: 4.1). The primary part of the architecture is a stack of transformer blocks, each of which is composed of two main components: **a (masked) multi-head self-attention** mechanism followed by a position-wise fully connected **feed-forward network**. **Layer Normalization and Residual Connections** are placed around these two main components. The transformer blocks are stacked on top of each other, with each layer processing the output of the previous one. The next sections describe the purpose of language models and moreover, the ways to be combined with variational quantum algorithms.

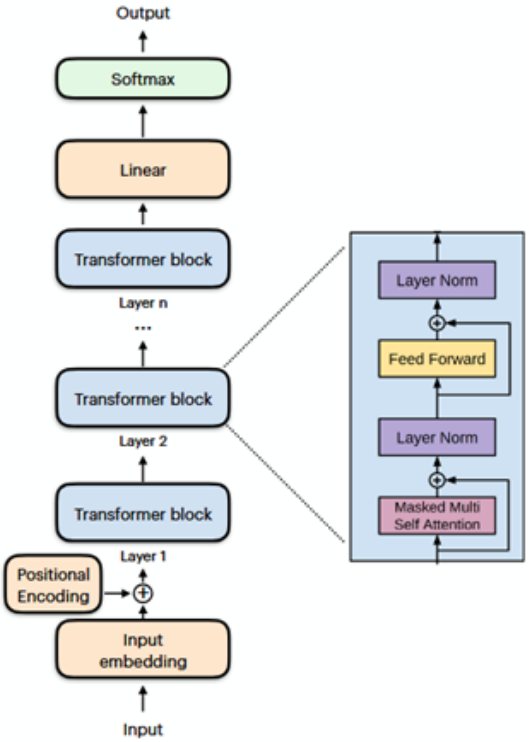


Figure 4.1: GPT's Architecture, adapted from [11].

4.1.1 Next word prediction process of GPT

In the inference stage, a language model (here, GPT) takes in a sequence of one-hot encoded tokens, and generates predictions for the next word in a sequence (fig: 4.2). The sequence of one-hot encoded tokens is first transformed through word embedding. These embeddings, after the positional encodings are added, are then input into the transformer blocks. Then, a final linear layer is applied to map the outputs from the transformer blocks back into the vocabulary space, generating a sequence of transformed vectors. The last transformed vector is passed through a softmax activation function, yielding a probability distribution across the vocabulary, indicating the likelihood of each word as the next sequence component.

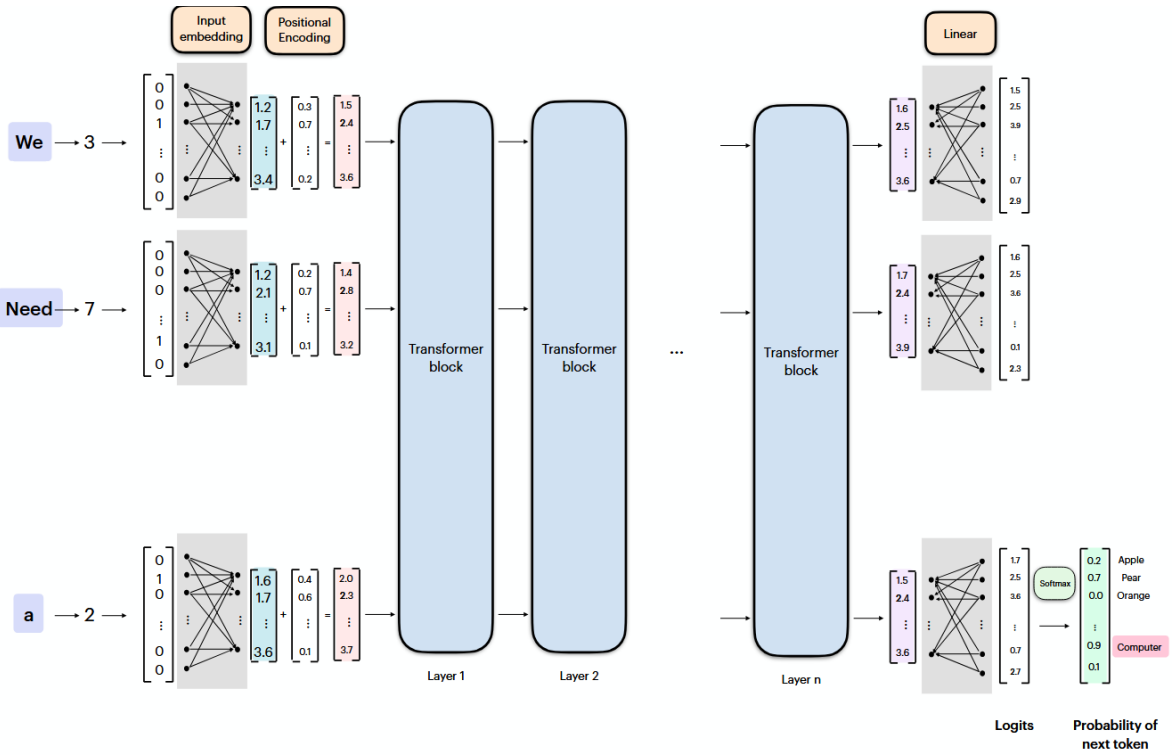


Figure 4.2: Next word prediction process of GPT. The initial sequence 'we', 'need', 'a'. The next sequence component is 'computer'.

4.2 QAOA-GPT: Generation of Adaptive Quantum Approximate Optimization Algorithm Circuits

This section summarizes the material published in

- Ilya Tyagin, Marwa H. Farag, Kyle Sherbert, Karunya Shirali, Yuri Alexeev, and Ilya Safro. QAOA-GPT: Efficient generation of adaptive and regular quantum approximate optimization algorithm circuits, 2025. [12]

This work shows that generative AI could be a promising avenue to generate compact quantum circuits in a scalable way.
QAOA-GPT, is a generative framework that leverages Generative Pretrained Transformers (GPT) to directly synthesize quantum circuits for solving quadratic unconstrained binary optimization problems. QAOA-GPT, generates high quality quantum circuits for new problem instances unseen in the training as well as successfully parametrizes QAOA

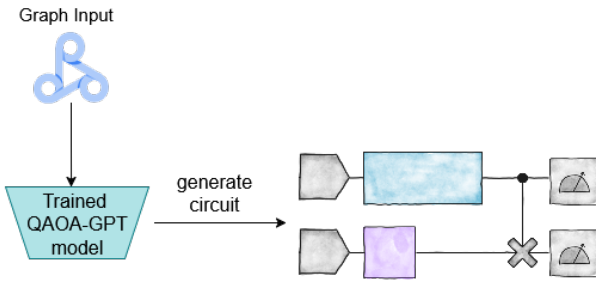


Figure 4.3: Circuit generation using a trained model. The input is a graph the output is a complete circuit with already computed parameters.

layers. Using QAOA-GPT to generate quantum circuits fig: 4.3 will significantly decrease both the computational overhead of classical QAOA and adaptive approaches that often use gradient evaluation to generate the circuit and the classical optimization of the circuit parameters.

In the scope of this thesis we will demonstrate the QAOA-GPT on the Maxcut problem on graphs. To diversify the training circuits and ensure their quality, we generated a synthetic dataset using the ADAPT-QAOA, a method explained in Chapter 3. The dataset was small because ADAPT-QAOA takes a lot of time to complete if we want to achieve optimal results and the main purpose is to showcase the QAOA-GPT architecture.

4.2.1 Graph Level embedding (FEATHER)

Graph embedding refers to the process of representing graph nodes as vectors which encode key information of the graph such as semantic and structural details, allowing machine learning algorithms and models to operate on them. In other words they are basically low-dimensional, compact graph representations that store relational and structural data in a vector space. Graph embeddings, as opposed to conventional graph representations, condense complicated graph structures into dense vectors while maintaining crucial graph features.

FEATHER [13] is a non-parametric graph embedding method based on characteristic functions that represent local distributions of node features. For each node u, the method computes the r-scale random walk weighted characteristic function:

$$\phi_u(\theta, r) = \sum_{w \in V} \hat{A}^r_{u,w} \cdot e^{i\theta x_w}$$

where $\hat{A} = D^{-1}A$ is the normalized adjacency matrix and $\hat{A}^r_{u,w}$ is the probability of reach-

ing node w from u in r steps via a random walk. This function is evaluated at a set of d frequencies $\theta \in \Theta$ to form a complex-valued embedding that captures the higher-order structure and attribute distribution around each node. Graph-level representations are constructed by pooling node-level embeddings (e.g., using the mean).

$$\overrightarrow{x} = x_1, .., x_{|V|} \quad \text{, set of features, each node has as ingle value as a feature}$$

$$x_w = log(deg(w) + 1) \quad \text{, the feature of each node}$$

$$\overrightarrow{\theta} = \theta_1, ..., \theta_k \quad \text{, set of k frequencies}$$

We will compute the graph level embedding for all the u nodes Alg: 1. The node level embedding is extracted by calculating the mean value for every value of θ_i. Thus the result is a vector with dimensionality $\mathbb{R}^d$, where $d = 2rk$.

Algorithm 1 Graph embedding pseudocode.

Data:
$\hat{A}$: `Normalized adjacency matrix`
$\hat{X}$:$[x_1, x_2, .., x_{|V|}]_{1\times|V|}$ `vector of node features`
$\overrightarrow{\theta}$: $[\theta_1, \theta_2, .., \theta_k]_{1\times k}$ `vector of k eval points`
r : `Scale of empirical graph characteristic function`

```
H = x^T θ
H = [cos(H)|sin(H)]
Z = [ ]
for i in range(r):
    H = Â · H
    Z = [Z|H]
return Z
```

4.2.2 Model Architecture and Training QAOA-GPT

The final training set is used to train QAOA-GPT, a decoder-only Transformer model based on the nanoGPT [14] implementation of GPT-2. The model was trained from scratch and was not initialized from any pre-trained earlier model due to using custom circuit tokenization schema (i.e., we do not use fine tuning or prompt engineering).
The token embeddings are sequences of graph-circuit tokens

$$graph\_token =< bos > (i, j), w, \ldots, < end\ of\ graph >$$

$$circuit\_token =< new\ layer > o_k, \gamma_k, \beta_k \ldots$$

$$token = [\ graph\_token \mid circuit\_token\ ]$$

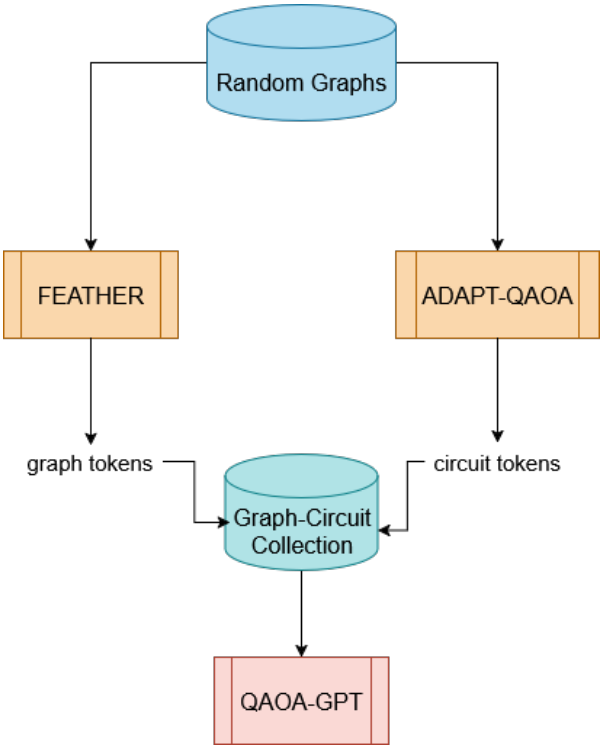


Figure 4.4: QAOA-GPT training architecture.

Where (i, j) is the edge of the graph connecting node i and j and w the weight of the edge. Given a graph we construct the Hamiltonian and we solve the Max-Cut instance using ADAPT-QAOA. γ_k and β_k are the parameters of each QAOA layer and o_k is the index of the mixing Hamiltonian from the operator pool.
Given a tokenized sequence $x_{1:T}$, the model uses:

- Token embeddings $E_{tok} \in \mathbb{R}^{V \times d}$
- Positional embeddings $E_{pos} \in \mathbb{R}^{T \times d}$
- Graph embeddings $e_G \in \mathbb{R}^d$ from FEATHER

The Transformer input is computed as:

$$X = E_{tok}(x_{1:T}) + E_{pos} + e_G \otimes 1_{1 \times T}$$

The dataset was created by generating graphs from the Erdős–Rényi random graph model G(n, p) where p was set to p = 0.5

- The dataset consists of
  - 20 ER graphs with n = 4 nodes
  - 30 ER graphs with n = 5 nodes
  - 27 ER graphs with n = 6 nodes
- Total 77 graphs. Each graph has random weights on edges, sampled from the uniform distribution $\mathcal{U}(0, 1)$
- Train/Test split: 90/10
- Vocabulary size: 563, block size: 103

Training was conducted on a **GPU: NVIDIA GeForce GTX 1650**, took approximately 20 minutes to complete 20,000 iterations. Due to the dataset's small size and the model's capacity, there is a risk of overfitting (the model learns the training data too well, including its noise and outliers, leading to poor generalization on new, unseen data).

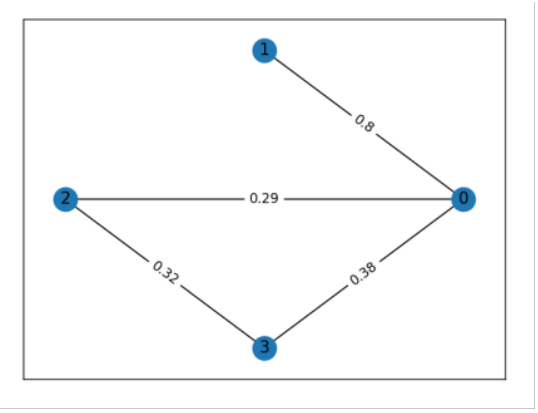
To showcase the trained model, we give as input a new graph, generated from a different random graph model, Barabasi-Albert model BA(n, m), where n = Number of nodes and m = Number of edges to attach from a new node to existing nodes.

The random generated graph inserted in the trained model is shown In Fig: 4.5a along with the optimal bitstrings found with ADAPT-QAOA 4.5b. The optimal bitstrings are the solution to the Max-Cut of the given graph:

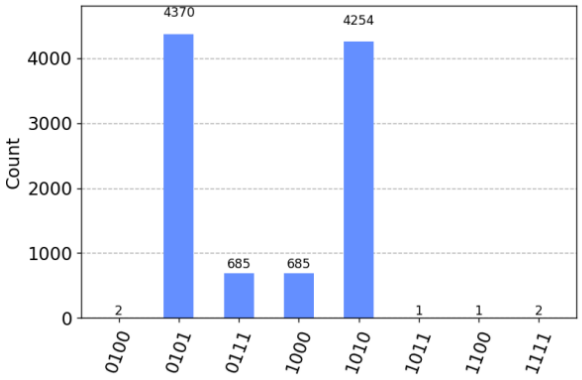
$$Opt1 = [0101], \quad Opt2 = [1010]$$

$$Solution = [Node\_3, \mathbf{Node\_2}, Node\_1, \mathbf{Node\_0}]$$

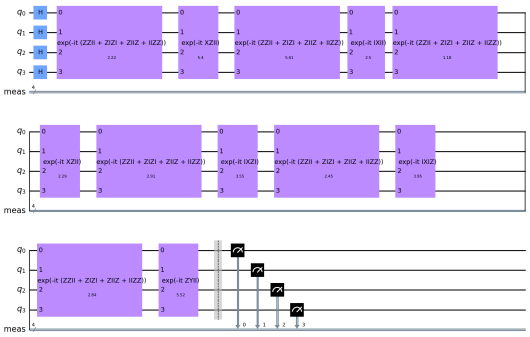
The trained QAOA-GPT generates a complete Quantum Circuit that consists of 7-QAOA-layers (Fig: 4.5c). Each layer has a different mixing Hamiltonian. The circuit generation happens in an instant without the need of optimizing any parameters.



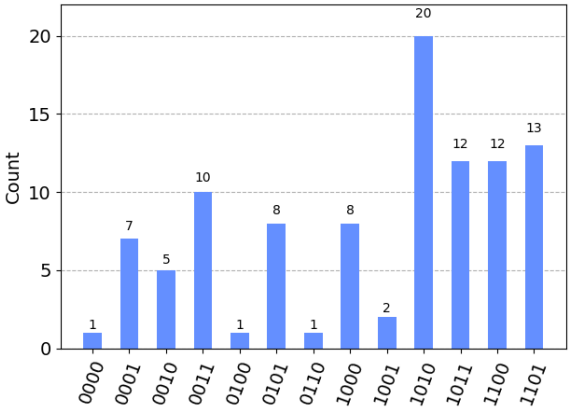
(a) Random generated *BA* graph, input to the trained gpt model.



(b) Optimal Max-Cut of Fig: 4.5a, found using ADAPT-QAOA.



(c) Generated Circuit from trained QAOA-GPT to provide the Max-Cut solution.



(d) Probability distribution of solutions, using 100 shots, after measuring the generated circuit. The bar with the highest probability corresponds to the optimal solution.

Figure 4.5: Testing the trained model.

The generated circuit in Fig: 4.5c, after measuring all the qubits, we get the probability distribution of the quantum state Fig: 4.5d. The solution with the highest probability is 1010 which gives the correct cut of the graph. **The model was trained on a small dataset for demonstrating purposes. As we can see, it could provide the correct solution to a given example.**

Chapter 5

Leveraging Quantum Algorithms for Transport Industry Problems

This chapter is the core contribution of this thesis and focuses on solving transportation problems using hybrid quantum-classical algorithms. Specifically, we will explore **Quantum Computing Applications on Scheduling Transshipment Operations**.

Transshipment problems form a subgroup of transportation problems, where transshipment is allowed. Transshipment is the shipment of goods or containers to an intermediate destination, and then from there to yet another destination. One possible reason is to change the means of transport during the journey (for example from ship transport to road transport), known as transloading shown in Fig. 5.1. Another reason is to combine small shipments into a large shipment (consolidation), dividing the large shipment at the other end (deconsolidation).

These Problems aim to optimize transportation time and cost over a specific period and provide an optimal schedule for logistic processes to execute. In the scope of this thesis, we aim to solve a transshipment problem that falls in the category of Multimodal Freight Optimization, a similar problem is analyzed in this study [15]. During the year, we built an optimization engine that is adjusted to variations of this type of problems and is able to use either classical or quantum solvers to accommodate the user.

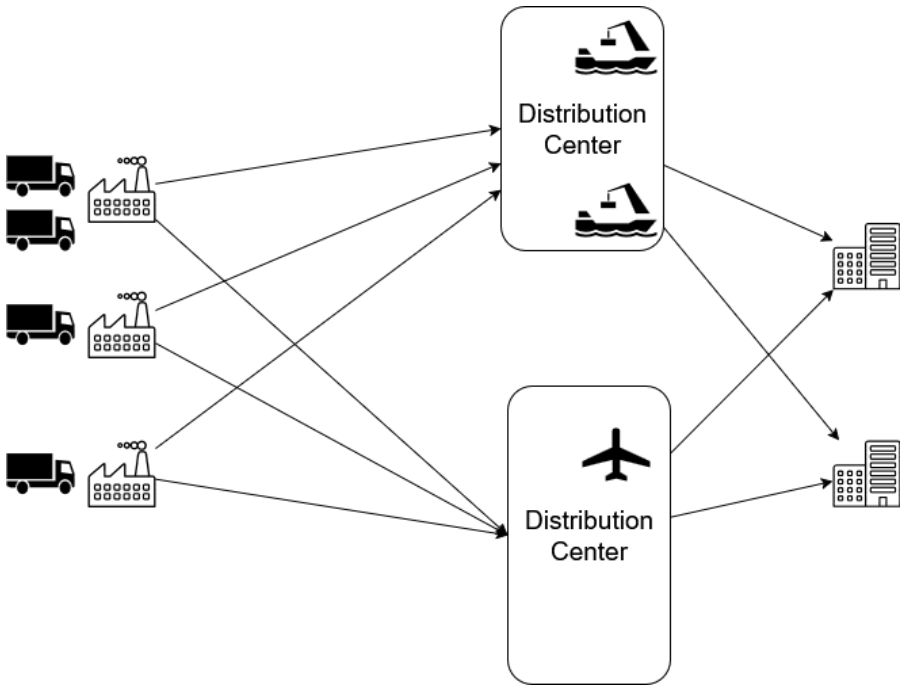


Figure 5.1: Visualization of transportation network.

5.1 Problem Description and Modeling

The main purpose of the problem is delivering goods to a desired destination across a supply chain network as shown in Fig.5.2. **The goods are located in *Logistic Centers*** (starting nodes), each Logistic Center has a number of available **Trucks** to transport the goods. The trucks are loaded to the maximum capacity and start moving on the road to reach the Departure Ports. In the ***Departure Port*** (intermediate nodes) there are available **Barges** that transport cargo to the ***Arrival Ports*** (destination nodes).

Once a truck arrives at a Departure Port two things can happen: Either it unloads its cargo and returns to the initial Logistic Center, or it goes aboard the available Barge and the unloading process begins once the barge arrives at an Arrival Port. If the Truck embarks on the Barge, it follows the same travel path with the assigned Barge. When the Barge returns to the initial Departure Port, the embarked truck returns to the originating Logistic Center.

This process continues until there aren't any remaining goods at the Logistic Centers.

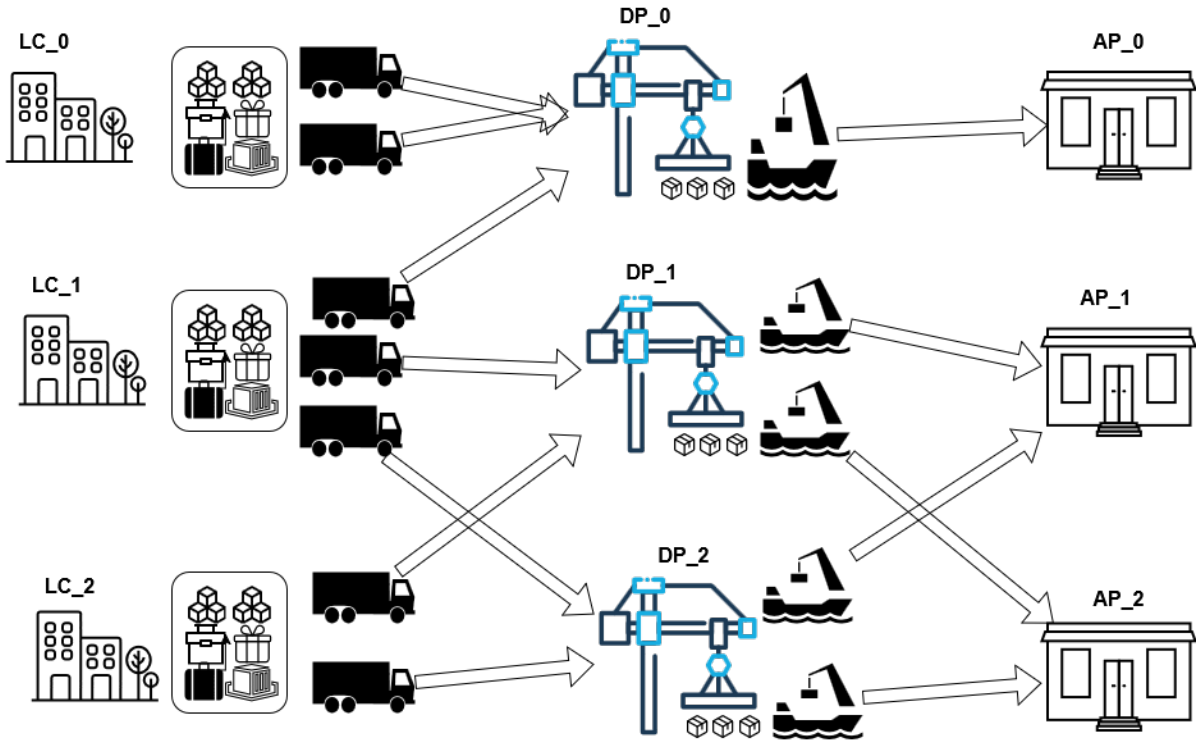


Figure 5.2: Visualization of the general problem. In the first (leftmost) layer, we have the 3 logistic centers with their reserves of goods. Goods are assigned and loaded into trucks. The trucks are sent to a departure port where they have the option to either unload or embark on the ships. In the second layer, the goods at the departure ports that have been unloaded from the trucks are then loaded onto the barges. The barges are then sent to the arrival ports.

The key components of the mathematical part include:

- **Decision Variables**:
  - Variables that decide whether a truck should be assigned to a land travel route: Let $\mathbf{x_{i,j,l}} \in \{\mathbf{0}, \mathbf{1}\}$ be the binary variable associated with the assignment of truck **l** from logistic center **i** to departure port **j**. Setting $x_{i,j,l} = 1$ indicates that vehicle l is assigned to take the trip from logistics center i to departure port j.
  - Variables that decide whether a barge should be assigned to a sea travel route: Let $\mathbf{y_{j,k,r}} \in \{\mathbf{0}, \mathbf{1}\}$ be the binary variable associated with the assignment of barge **r** from departure port **j** to arrival port **k**. Setting $y_{j,k,r} = 1$ indicates that barge r is assigned to take the trip from departure port j to arrival port k.

- **Environment parameters**:
  - $\mathbf{t_{i,j,l}}$ is a time parameter associated with the overall time required for truck **l** to travel the route from logistic center **i** to departure port **j**. This parameter is based on a heuristic distance between the two endpoints and the travel speed of truck **l**. (It also includes the loading/unloading time of truck l, which varies depending on the truck type. Unloading might even be unnecessary if the truck is meant to embark on the barge, and this factor is also taken into consideration in this parameter.)
  - $\mathbf{c_l}$ is a capacity parameter associated with the capacity truck **l**. We assume that the truck are loaded to their maximum capacity, so the $\mathbf{c_l}$ is equal to volume of good transported by the truck $\mathbf{c_l}$.
  - $\mathbf{t_{j,k,r}}$ is a time parameter associated with the overall time required for barge **r** to travel the route from departure port **j** to arrival port **k**. This parameter is based on a heuristic distance between the two endpoints and the travel speed of the barge. (It also includes the loading time of the barge.)
  - $\mathbf{c_r}$ corresponding capacity parameter for the barges
  - $\mathbf{a_l}$ This is a parameter that serves as a label indicating the destination arrival port for the cargo of truck **l**. While the final trip to this destination will be carried out by a barge, the transportation by land vehicles also plays a significant role in the proper delivery of goods.
- **Objective function for Land transportation**:

  The objective function will be a sum of total execution time and volume of transported goods. It should be able to balance the conflicting objectives of minimizing total transportation time while maximizing the volume of transported goods (conflicting because the more goods we transport, the more time we might need). Such a cost function is:

$$\min_{x} \quad w_t \sum_{i,j,l} t_{i,j,l} x_{i,j,l} - w_c \sum_{i,j,l} c_l x_{i,j,l} \tag{5.1}$$

  where $w_t, w_c > 0$.
- **Objective function for Sea transportation**:

The objective function will be a sum of total execution time of the available barges

$$\min_{y} \quad w_{y_t} \sum_{j,k,r} t_{j,k,r} y_{j,k,r} \tag{5.2}$$

where $w_{y_t} > 0$.

- **Constraints**:
  - **No truck will be assigned to more than one trip** per iteration(one trip or no trip):

$$\sum_{i,j} x_{i,j,l} \leq 1 \quad \forall l \tag{5.3}$$

  - **No barge will be assigned to more than one trip**:

$$\sum_{j,k} y_{j,k,r} \leq 1 \quad \forall r \tag{5.4}$$

  - **Matching the volume of goods arriving at departure ports via land vehicles with the volume of goods departing via barges**:

$$\sum_{i,l} c_l x_{i,j,l} = \sum_{k,r} c_r y_{j,k,r} \quad \forall j \tag{5.5}$$

    This ensures that no additional trucks are employed if there are insufficient barges available to carry their cargo. Similarly, no additional barges will be utilized unless they are scheduled to operate at full capacity. (Here, we assume it is preferable to wait slightly longer for a barge to be fully loaded rather than have half-empty barges in operation.) **This approach also prevents goods from being stuck at the departure port for an extended period.**

5.1.1 Constraint Adjustment

Delivering the goods to their designated destination (improving the capacity matching constraint):
Equation 5.5 indicates that the capacities of trucks entering the departure port j must equal the capacities of barges exiting the same departure port.

However, by simply matching capacities, we generally encounter the problem of filling a barge with goods that are supposed to go to different directions. We need to incorporate a parameter into the equation to ensure that the barge is filled with commodities of the same type and that the barge is sent to the correct arrival port.

The reformed constraint will look like this:

$$\sum_{i,l} a_l c_l x_{i,j,l} = \sum_r c_r \sum_k (C^k y_{j,k,r}) \quad \forall j \tag{5.6}$$

How to handle the a_l labels: Each good has a designated arrival port, and therefore, each good should be assigned a label indicating its destination. A fundamental rule for barges is that they must carry goods with the same label, as a barge has only one destination arrival port, and this destination must be correct for all of its cargo. If a truck is to be loaded onto a barge or arrives at a departure port where all barges are destined for the same arrival port, then this truck should carry goods with the same label, as all of its cargo will ultimately reach the same arrival port. However, if there are multiple barges at the departure port, each with a different arrival port destination, then the truck may carry cargo with different labels, provided that the cargo is appropriately divided among the corresponding barges.

Let's consider an example for futher illustration of the idea: We have 10 trucks, each with a capacity of 10 units, entering the departure port. Half of the trucks carry cargo destined for arrival port 0, and the other half carry cargo destined for arrival port 1. Thus, we set $a_l = C^0$ for 5 trucks and $a_l = C^1$ for the other 5 trucks. Suppose we have two barges, each with a capacity of 50 units, and there are 3 arrival ports to choose from (AP_0, AP_1, AP_2). Let $C = 10$, the equation 5.6 becomes:

$$5{\cdot}10{\cdot}10^0{+}5{\cdot}10{\cdot}10^1 = 50(10^0 y_{j,0,1}{+}10^1 y_{j,1,1}{+}10^2 y_{j,2,1}){+}50(10^0 y_{j,0,2}{+}10^1 y_{j,1,2}{+}10^2 y_{j,2,2}) \Leftrightarrow$$

$$550 = 50(10^0 y_{j,0,1} + 10^1 y_{j,1,1} + 10^2 y_{j,2,1}) + 50(10^0 y_{j,0,2} + 10^1 y_{j,1,2} + 10^2 y_{j,2,2}) \Leftrightarrow$$

In order to match the order of magnitude on the left-hand side of the equation, the y variables should take the values:

$$y_{j,0,1} = 1, \quad y_{j,1,1} = 0, \quad y_{j,2,1} = 0, \quad \textit{and} \quad y_{j,0,2} = 0, \quad y_{j,1,2} = 1, \quad y_{j,2,2} = 0$$

or

$$y_{j,0,1} = 0, \quad y_{j,1,1} = 1, \quad y_{j,2,1} = 0, \quad \textit{and} \quad y_{j,0,2} = 1, \quad y_{j,1,2} = 0, \quad y_{j,2,2} = 0$$

This successfully indicates that one barge will go to arrival port 0, the other will go to arrival port 1 and no barge assigned to arrival port 2 exactly as anticipated.

5.1.2 An integrated land and sea transport model

Combining the two formulations from the sea and land part, we arrive to an integrated model, including the previously discussed constraints. This reads as follows:

$$\text{minimize} \quad w_{xt} \sum_{i,j,l} t_{i,j,l} x_{i,j,l} - w_{xc} \sum_{i,j,l} c_l x_{i,j,l} + w_{yt} \sum_{j,k,r} t_{j,k,r} y_{j,k,r}$$

$$\text{subject to} \quad \sum_{i,j} x_{i,j,l} \leq 1 \quad \forall l,$$

$$\sum_{j,k} y_{j,k,r} \leq 1 \quad \forall r,$$

$$\sum_{i,l} c_l a_l x_{i,j,l} = \sum_{r} c_r \sum_{k} (C^k y_{j,k,r}) \quad \forall j$$

The remainder of this chapter is dedicated to exploring how quantum algorithms, specifically those designed for optimization such as QAOA, can be employed to address this model. We translate the problem into a form suitable for quantum computation, typically through QUBO (Quadratic Unconstrained Binary Optimization) formulation, which facilitates implementation on current quantum hardware and local simulators. Through this investigation, we aim to assess both the computational complexity and the solvability of various problem instances. By evaluating the performance of quantum algorithms on different scale and configurations, we explore the feasibility of extending quantum approaches to real-world scheduling challenges and large-scale environments. This serves as a critical step toward understanding the limitations and capabilities of near-term quantum devices in solving practically relevant optimization problems.

After benchmarking the quality of the quantum algorithm solutions, the next step involves integrating the quantum solver within the optimization engine, which repeatedly executes the optimization process across a defined time horizon to produce a schedule. The final model's output should be in the form of a schedule:

Table 5.1: Output information for a single land vehicle.

Veh ID: Truck_0					
Trip ID	**Start**	**End**	**Depart. Time (mins)**	**Arrival Time (mins)**	**Assigned Sea Trip ID**
L101	LC1	DP1	0	25	NA
L102	DP1	LC1	35	60	NA
...	...	...	...	...	...

5.2 Exploring Quantum Algorithms for Transport Optimization

In this section we will solve instances of the problem explained in the previous section, to compare the solutions between quantum and classical algorithms and to evaluate the results in local noisy and noiseless simulators and in cloud quantum processors too.

We will formulate the mathematical model into a QUBO problem. As QUBO problems are NP-hard in general, they serve as important benchmarks for testing classical heuristics as well as quantum optimization algorithms.

The problem is slightly simplified to reduce the number of variables required without affecting the mathematical model.

- All trucks have the same attributes (capacity, speed)
- All barges have the same attributes (capacity, speed)
- All trucks are already loaded and their final destinations are predefined
- All land routes have the same distance
- All sea routes have the same distance

The binary variables are created by iterating over the attributes of the Land and Sea Vehicles (LCs, DPs, APs). The dataset is encoded to binary string $x = [\{0,1\},\{0,1\},...]$, each one of them represents potential routes for land and sea vehicles therefore a trip schedule. The goal is to find the optimal bitstring x^* that minimizes the function $\sum_{i,j} x_i Q x_j$.

Suppose we have a bitstring $x = [b_0, b_1, ..., b_n]$ with n binary variables. The first m elements refer to land variables and the rest to sea variables. A single bit $b_i = 1$ verifies that a vehicle V travels from location A to location B

$$\overrightarrow{x} = [\underbrace{b_0, b_1, ..., b_{m-1}}_{\#Land\_Variables} | \underbrace{b_m, b_1, ..., b_n}_{\#Sea\_Variables}]$$

$$\#Land\_Variables = Trucks \times DPs$$

$$\#Sea\_Variables = Barges \times APs$$

The classical binary variables are translated to qubits and the quantum state is parameterized $|x(\overrightarrow{\theta})\rangle$ and the quantum circuit works alongside a classical optimizer to

obtain the values of $\overrightarrow{\theta^*}$ that lead to the optimal solution. E.g:

$$|b_0\rangle = \{|0\rangle, |1\rangle\} = |Truck_0\_LC_0\_DP_0\_AP_0\rangle$$

$$|b_1\rangle = \{|0\rangle, |1\rangle\} = |Truck_0\_LC_0\_DP_1\_AP_0\rangle$$

The QUBO matrix Q is a $n \times n$ matrix where n the number of variables. The diagonal elements describe the cost function and the off-diagonal elements the problem constraints. The constraints are transformed into squared penalty terms and are added into the cost function. E.g:

Suppose the constraint $\sum_i x_i \leq 1$

$$Q^P = P\left(\sum_i x_i - 1\right)^2, \quad \text{where P is a large positive number}$$

$$Q^P = P\left(\left(\sum_i x_i\right)^2 - 2\sum_i x_i + 1\right)$$

$$Q^P = P\left(\sum_i x_i^2 + 2\sum_{i<j} x_i x_j - 2\sum_i x_i + 1\right)$$

Since x_i is a binary variable, $\sum_i x_i^2 = \sum_i x_i$

$$Q^P = P\left(\sum_i x_i + 2\sum_{i<j} x_i x_j - 2\sum_i x_i + 1\right)$$

Thus, the correct final form is:

$$Q^P = P\left(2\sum_{i<j} x_i x_j - \sum_i x_i + \text{offset}\right)$$

- **At-most-one trip per truck**

Initial formulation: $\sum_{i,j} x_{i,j,l} \leq 1 \quad \forall l$

$$Q^{P_1} = P_1\left(\sum_{i,j} x_{i,j,l} - 1\right)^2$$

$$Q^{P_1} = P_1\left(2\sum_{i,j<i'j'} x_{i,j,l}x_{i',j',l} - \sum_{i,j} x_{i,j,l}\right)$$

- **At-most-one trip per barge**

$$\text{Initial formulation:} \quad \sum_{j,k} y_{j,k,r} \leq 1 \quad \forall r$$

$$Q^{P_2} = P_2 \left(\sum_{j,k} y_{j,k,r} - 1 \right)^2$$

$$Q^{P_2} = P_2 \left(2 \sum_{j,k<j'k'} y_{j,k,r} y_{j',k',r} - \sum_{j,k} y_{j,k,r} \right)$$

- **Match Truck capacities that arrive to a DP with Barge capacities**

$$\text{Initial formulation:} \sum_{i,l} c_l a_l x_{i,j,l} = \sum_r c_r \sum_k (C^k y_{j,k,r}) \quad \forall j$$

$$Q^{P_3} = P_3 \left(\sum_{i,l} c_l a_l x_{i,j,l} - \sum_r c_r \sum_k (C^k y_{j,k,r}) \right)^2$$

Final form of the expanded penalty term:

$$Q^{P_3} = P_3 \Bigg[\left(\sum_{i,l} (c_l a_l)^2 x_{i,j,l} + 2 \sum_{(i,l)<(p,q)} (c_l a_l c_q a_q) x_{i,j,l} x_{p,j,q} \right)$$
$$- 2 \left(\sum_{i,l,r,k} (c_l a_l c_r C^k) x_{i,j,l} y_{j,k,r} \right)$$
$$+ \left(\sum_{r,k} (c_r C^k)^2 y_{j,k,r} + 2 \sum_{(r,k)<(s,p)} (c_r C^k c_s C^p) y_{j,k,r} y_{j,p,s} \right) \Bigg]$$

- **Enforce Trucks to reach the correct destination**

$$Q^{P_4} = P_4 \sum_{j,l} \left(x_{i,j,l} - \sum_{r=0} x_{i,j,l} y_{j,k,r} \right) \quad \forall k$$

where k, are the destination points of the trucks. This constrained was added to the QUBO model to ensure the correct distribution of goods.

The final cost function that is mapped to the Q matrix is the sum of these penalty terms (and any other unstated cost terms):

$$Q = w_{xt} \sum_{i,j,l} t_{i,j,l} x_{i,j,l} - w_{xc} \sum_{i,j,l} c_l x_{i,j,l} + w_{yt} \sum_{j,k,r} t_{j,k,r} y_{j,k,r} + Q^{P_1} + Q^{P_2} + Q^{P_3} + Q^{P_4}$$

5.2.1 Quantum Experiments in Simulators

In this chapter, we carefully selected 3 test cases. Those three cases are implemented in local machines simulating both noiseless and noisy quantum circuits. Table. 5.2 defines the three logistics-style test cases and their VQA/QAOA/Compressed VQA workloads.

Table 5.2: Problem scenarios and workload sizes.

Case	Scenario	Alg.	Vars	Qubits	Tasks	Shots*
1.	1 LC, 4 Trucks 2 DPs, 2 Barges, 2 APs	QAOA	12	12	150	5 000
		VQA	12	12	350	5 000
		Compr. VQA	12	5	200	1 000
2.	1 LC, 5 Trucks 3 DPs, 3 Barges, 2 APs	QAOA	21	21	350	20 000
		VQA	21	21	550	20 000
		Compr. VQA	21	9	250	10 000
3.	3 LCs, 100 Trucks, 3 DPs, 40 Barges, 3 APs	Compr. VQA	420	15	300	50 000

* Shots per optimizer iteration.

Validation Metrics

To validate the quality of every solution produced in this chapter we used the following metrics:

- Cost Function Evolution: The minimum and maximum values of the cost function were found using classical optimization software and were used to normalize the cost function values from 0 to 1.
- Fraction of solutions: shows how often good solutions appear compared to random or all solutions (also normalized).
- Route decision: A graph for the final answer is produced, so we can validate the solution based on the right assignments and produce a right assignment ratio.

Noise Models

We run noisy emulations to validate if QPUs (quantum processing unit) would have a big impact in the quality of results. We created two noise models that simulate noise of really good and moderate quantum hardware to get upper and lower bound values in terms of noise. We will refer to the low noise model as noise model 1, and to the high noise model as noise model 2, and the specific attributes of each noise model are highlighted in the table: 5.3.

Noise Model 1	**Noise Model 2**	**Description**
$T_1 = 200\mu s$	$T_1 = 50\mu s$	The relaxation time, representing the characteristic time for a qubit to decay from the excited state $\|1\rangle$ to the ground state $\|0\rangle$.
$T_2 = 100\mu s$	$T_2 = 20\mu s$	The dephasing time, describing the timescale over which a qubit loses phase coherence due to environmental noise.
$Prob_{1qubit} = 0.001$	$Prob_{1qubit} = 0.003$	The probability of a single-qubit gate error occurring during execution.
$Prob_{2qubit} = 0.003$	$Prob_{2qubit} = 0.01$	The probability of a two-qubit gate error occurring, typically higher due to the complexity of multi-qubit operations.
$P_{1\to0} = 0.01$	$P_{1\to0} = 0.01$	The probability of a relaxation error $\|1\rangle \to \|0\rangle$ occurring during idle or gate operations.
$P_{0\to1} = 0.01$	$P_{0\to1} = 0.01$	The probability of a thermal excitation error $\|0\rangle \to \|1\rangle$ occurring due to interaction with the environment.

Table 5.3: Noise parameters used in the simulations.

Case Scenario 1 with 12 variables, (small scale)

Suppose we have **one Logistic Center** with **4 Trucks** loaded to the maximum capacity. The first 2 trucks carry goods with destination to the first Arrival Port and the other 2 trucks carry goods with destination to the second Arrival Port. There are **2 Departure Ports**, each with a Barge ready to move the goods to their corresponding Arrival Ports.

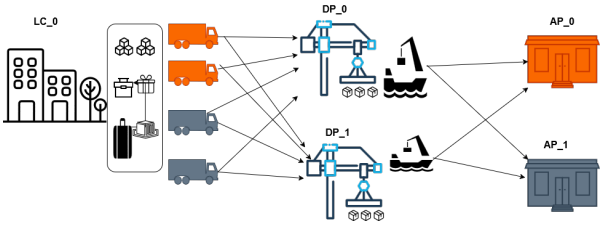
There are 8 binary variables for the Land Vehicles and 4 for the Sea Vehicles. Total 12 variables E.g:

$$|b_0\rangle = \{|0\rangle, |1\rangle\} = |Truck_0\_LC_0\_DP_0\rangle$$

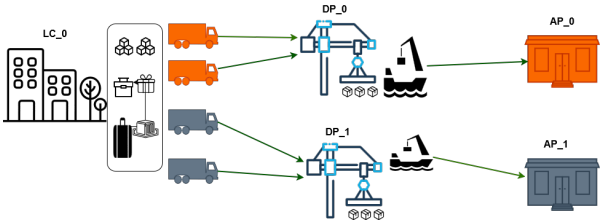
$$|b_1\rangle = \{|0\rangle, |1\rangle\} = |Truck_0\_LC_0\_DP_1\rangle$$

The optimal bitstring can be easily created with annealers, Google OR Tools or even by simple numerics and corresponds to:

$$x^* = [\,1\,0\,1\,0\,0\,1\,0\,1\,1\,0\,0\,1\,]$$



(a) Possible routes: 8 Land routes + 4 Sea routes.



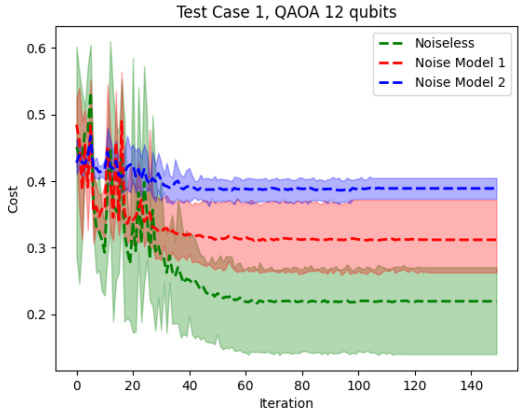
(b) Optimal routes.

Figure 5.3: Orange trucks need to arrive to orange APs and Grey trucks to grey APs.

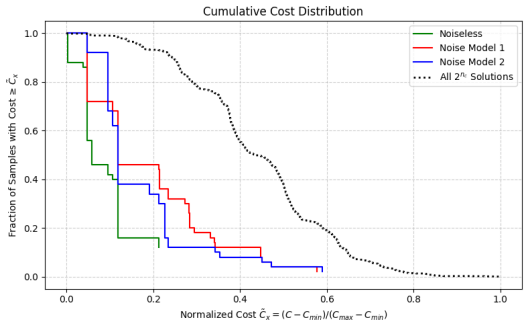
After executing the Test Case, we get the depicted results in figure: 5.4 and we came to the following conclusions:

- QAOA doesn't perform well under noisy conditions. The results of noise simulations deviate significantly from the noiseless case. It can reach a final cost with more than 20% gap from the optimal. Moreover, the cost evolution exhibits greater variance across runs, indicating larger deviations from the mean compared to other algorithms.

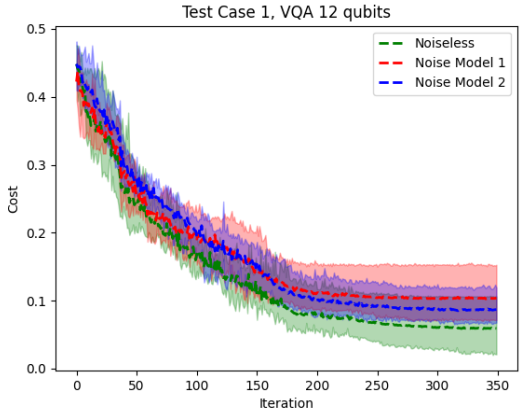
- VQA and Compresed VQA seem to work better for this kind of problem. Noise simulations don't deviate much from the noiseless one. A good indicator to compare the quality of the results each algorithm produce is to count the number of trucks that reach the correct final destination. Compressed VQA produces a result where 3/4 trucks reach the correct destination (sometimes 4/4, but not always) while only 5 qubits are used. Standard VQA produces an optimal result where 4/4 trucks arrive at the correct location. On the other hand with QAOA only 2/4 trucks arrive at the correct location.



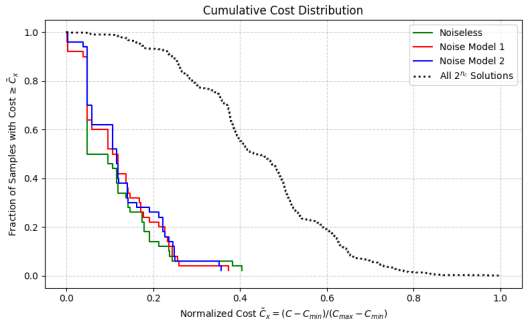
(a) Cost evolution (QAOA 5-layers).



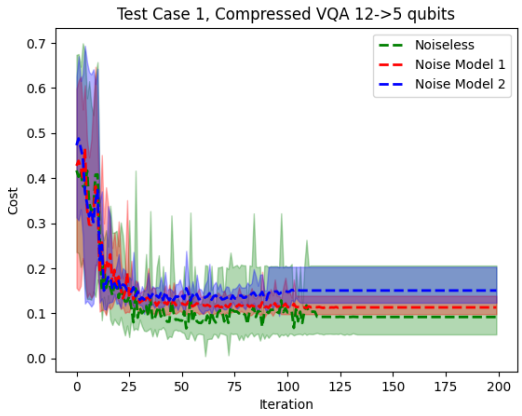
(b) Fraction of solutions (QAOA). Sampled top 10 solutions of each optimization run.



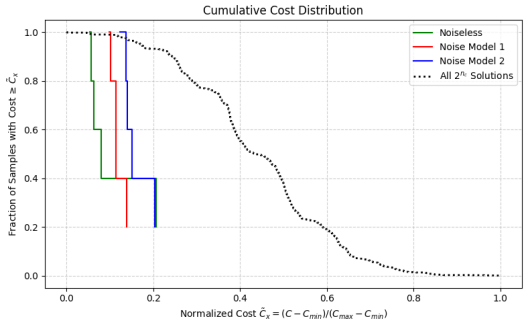
(c) Cost evolution (VQA 2-layers).



(d) Fraction of solutions (VQA). Sampled top 10 solutions of each optimization run.



(e) Cost evolution (Compressed VQA 2-layers).



(f) Fraction of solutions (Compressed VQA). Sampled top 10 solutions of each optimization run.

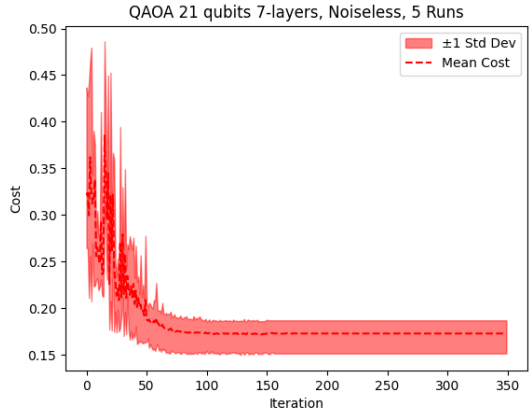
Figure 5.4: Experimental results from Test Case 1 comparing QAOA, VQA, and Compressed VQA. Cost evolution across 5 optimization runs of noiseless and noisy simulation. Dashed (- - -) lines are the mean costs.

Case Scenario 2 with 21 variables, (medium scale)

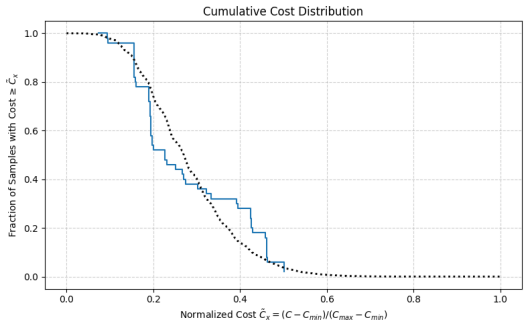
In this scenario we have the same setup as before but with more trucks and barges(5 trucks, 3 barges). The number of variables is 21. QAOA and VQA require 21 qubits while Compressed VQA only 9.

Judging from the results from figure: 5.5 we can confirm that:

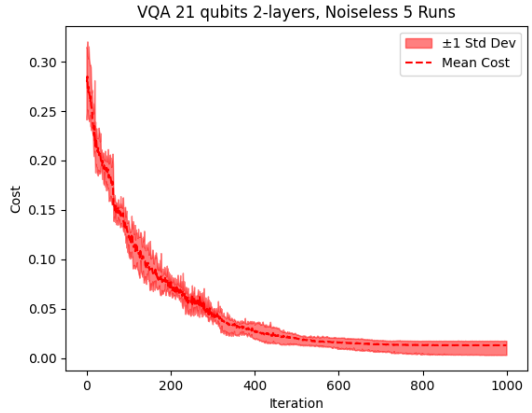
- QAOA produces low quality results. We can see from the fraction of solutions (fig: 5.5b) that the solution's curve is similar to the random one.
- While VQA produces the optimal solution in the noiseless simulation, it took more than 8 hours to execute. QAOA also took many hours to complete. We didn't perform noise simulations for these two algorithms because the estimated delay would be days.
- Compressed VQA performed well with a 5% gap from the optimal cost in the noiseless case and ∼10% in the noise cases. It also ensures that most trucks arrive at the correct location (4/5 trucks) but the solution involves multiple truck assignments too, which is a constraint violation. If any constraint is violated we cannot produce a schedule that contains multiple assignments for the same truck or transporting goods to a port where there isn't any barge available.



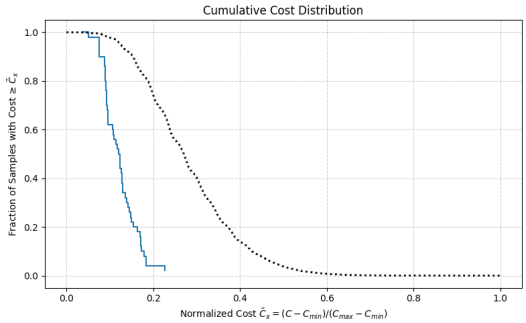
(a) Cost evolution (QAOA 7-layers).



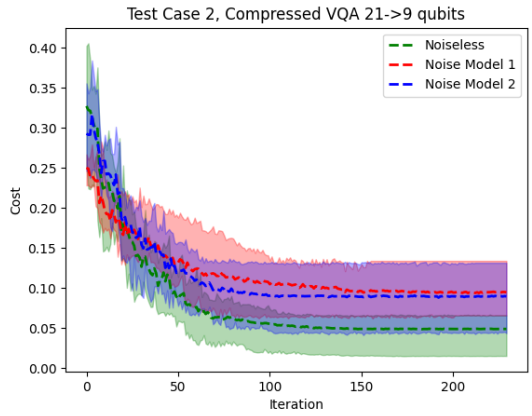
(b) Fraction of solutions (QAOA). Sampled top 10 solutions of each optimization run.



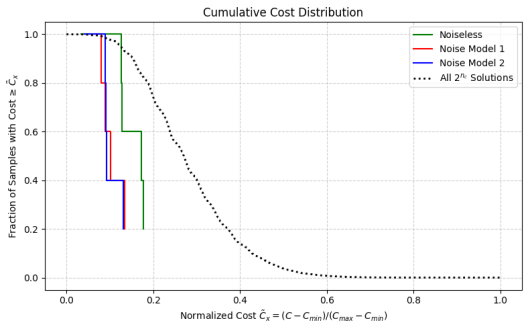
(c) Cost evolution (VQA 2-layers).



(d) Fraction of solutions (VQA). Sampled top 10 solutions of each optimization run.



(e) Cost evolution (Compressed VQA 2-layers).



(f) Fraction of solutions. Sampled top solutions of each optimization run.

Figure 5.5: Experimental results from Test Case 1 comparing QAOA, VQA, and Compressed VQA. Cost evolution across 5 optimization runs of noiseless and noisy simulation. Dashed (- - -) lines are the mean costs.

Case Scenario 3 with 420 variables, (large scale)

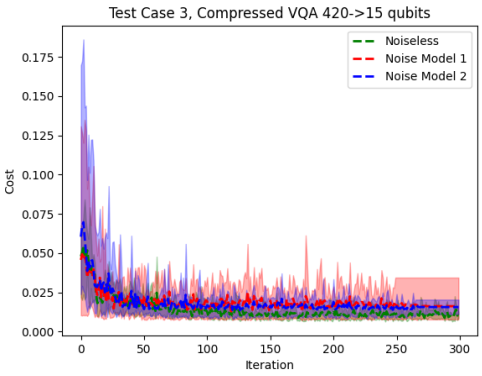
In this scenario, we have the same setup with the previous cases but with 100 trucks and 40 barges. The number of variables is 420 and we solved this case with only 15 qubits using the Compressed VQA method because, as we saw in the previous scenario, the 21 variables are already pushing the limits of local simulators for QAOA and VQA. We used a classical MILP solver to find the lower bound of the cost function in order to normalize the values in figure: 5.6. We couldn't find the optimal value from the QUBO matrix directly because it took too long to evaluate all 2^{420} possible solutions.

As we can see, the quantum approach reaches a final cost very close to the optimal we found using classical MILP solver, with 2.5% difference. The solution contains more than 100 assignments, so we created a graph (fig: 5.7) that visualizes the assignments in the following order:

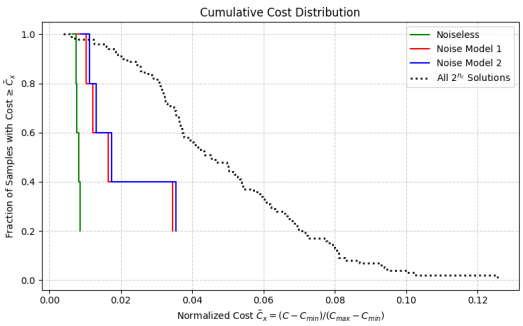
$$\boxed{LC} \rightarrow \boxed{Truck} \rightarrow \boxed{DP} \rightarrow \boxed{Barge} \rightarrow \boxed{AP}$$

The graph is too dense because of the large number of assignments, but the information is decoded and presented:

- Trucks utilized 77 out of 100
- Barges utilized 37 out of 40
- Duplicate Assignments for Trucks & Barges: 42 trucks & 17 Barges
- Trucks that reached the correct destination: 55.8%



(a) Cost evolution across 5 optimization runs of noiseless simulations. Dashed (- - -) lines are the mean costs.



(b) Fraction of solutions. Sampled top solutions of each optimization run.

Figure 5.6: Test Case 3, Compressed VQA 420 → 15 qubits, 2-layers, 5 optimization runs.

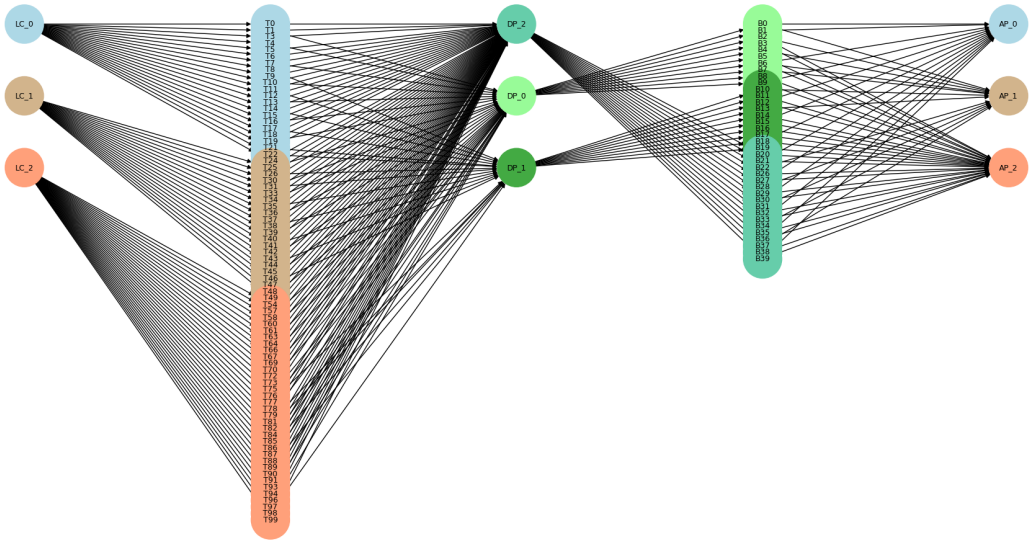


Figure 5.7: Graph representation of the solution found with compressed vqa. Leftmost nodes are the LCs connected to the trucks. Middle nodes are the DPs connected from the left with the trucks and from the right with the Barges. Rightmost nodes are the APs connected with the Barges. Each connection is a route assignment indicated by the solution.

5.2.2 Post-Processing of Quantum Samples

When solving the truck–barge assignment as a QUBO with QAOA (or other VQAs), the measured bitstrings correspond to candidate solutions. However, due to the probabilistic nature of the quantum sampler and the use of penalty terms to encode constraints, many sampled bitstrings may violate feasibility constraints. For example:

- a truck may be assigned to multiple destinations simultaneously
- a barge may be used for more than one trip
- the flow balance between truck deliveries to departure ports and barge shipments to arrival ports may not be satisfied

To extract valid logistics schedules from these raw bitstrings, we introduce a post-processing repair step. The repair process projects each sampled bitstring onto the feasible solution space by enforcing the hard constraints of the original problem. We designed a Mixed-Integer Linear Programming (MILP) repair model that projects each sampled bitstring onto the feasible solution space.

Let $z \in \{0,1\}^n$ be the repaired binary vector corresponding to trucks and barges, and let $b \in \{0,1\}^n$ be the original bitstring obtained from the quantum sampler. To enforce closeness to the sampled solution, we introduce binary flip indicators f_i such that:

$$f_i = |z_i - b_i|, \quad i \in \{1, ..., n\}$$

The objective of the MILP repair is to **minimize the number of bit flips** while also incorporating a small weight on the original objective function to break ties between equally close solutions:

$$\min \lambda \sum_i f_i + \epsilon \cdot Obj(z)$$

where:

- λ is the penalty on bit flips
- ϵ is a small coefficient applied to the original cost function (to prefer lower-cost solutions among those with the same number of flips).

In practice, the model uses the sampled bitstring as a warm start, which allows the solver to converge quickly. The model isn't a full-scale MILP because **it only repairs the given bitstring rather than optimizing from scratch**. The output is a guaranteed feasible truck–barge assignment that is minimally altered from the original quantum solution.

5.3 Composing an optimization engine for the full Transport Scheduling Problem

A Full Time Horizon Optimization Model optimizes the entire transportation process at once. The formulation of such a model would be very similar to the previous one, with the addition of a *time* dimension in the decision variables. **Adding the *time* dimension to the problem increases its size, making it difficult to solve using any solver**, let alone the underdeveloped quantum solvers. Additionally, since the optimization is done over the whole horizon at once, the algorithm isn't dynamic. If something unexpected arises (blocked routes, vehicle malfunction), we would need to recalculate the entire schedule.

Instead of the full horizon approach we propose a **Greedy method** to solve the scheduling issue. In this approach, we discretize time into timesteps, where in each timestep there is an assignment of land and sea vehicles to trips. We call this a greedy approach because, in each timestep, an optimization problem is solved that optimizes the trip assignment only for the current timestep, without considering past or future assignments.

5.3.1 Integrating the classical solver

For the classical solver we used an existing optimization library (OR-TOOLS [16]) to provide the necessary functions for solving Transport optimization instances similar to the quantum test cases. The optimizer recognizes the mathematical formulation of the integrated land and sea model:

$$\text{minimize} \quad w_{xt} \sum_{i,j,l} t_{i,j,l} x_{i,j,l} - w_{xc} \sum_{i,j,l} c_l x_{i,j,l} + w_{yt} \sum_{j,k,r} t_{j,k,r} y_{j,k,r}$$

$$\text{subject to} \quad \sum_{i,j} x_{i,j,l} \leq 1 \quad \forall l,$$

$$\sum_{j,k} y_{j,k,r} \leq 1 \quad \forall r,$$

$$\sum_{i,l} c_l a_l x_{i,j,l} = \sum_{r} c_r \sum_{k} (C^k y_{j,k,r}) \quad \forall j$$

Although we need to modify the last constraint before deploying this model to the classical solver. There is a potential problem in the constraint responsible for matching

imported truck capacities to exporting barge capacities:

$$\sum_{i,l} c_l a_l x_{i,j,l} = \sum_r c_r \sum_k (C^k y_{j,k,r}) \quad \forall j$$

This constraint might be violated not due to suboptimal truck-barge assignments, but because the (classical) optimizer tries to enforce this and we expect to have slight mismatches in truck-barge capacities.

Solving Capacity Mismatches Using Slack Variables

To solve this issue, we introduce a few additional positive non-decision variables whose sole purpose is to fill this gap. We call them slack variables, denoted as $s_{j,k}$. The upper bound of the slack variables should be lower than the smallest available truck capacity. When the slack variables are "activated", they should not replace any truck. To handle the different orders of magnitude, we will use the slack variables in a slightly modified form, such as:

$$\sum_{i,l} c_l a_l x_{i,j,l} + \sum_k C^k s_{k,j} = \sum_r c_r \sum_k (C^k y_{j,k,r}) \quad \forall j$$

With this update to the constraint, we allow the cumulative truck capacities to be slightly less than the cumulative barge capacities, but not the other way around in case of a mismatch. **This assumes that no unfitted goods are left at the ports**.

5.3.2 Integrating the quantum solver

Since we benchmarked well-known quantum algorithms across problems of varying sizes, we propose an algorithm selection strategy based on our findings. QAOA will be excluded, as its performance was consistently poor. Standard VQA achieved optimal results but only for small qubit counts. Therefore, we will primarily rely on Compressed VQA, adjusting the number of shots and ancilla qubits according to the problem size.

Variable range	Algorithm	Shots	Tasks
$\#Variables \leq 12$	VQA	5000	400
$12 < \#Variables < 21$	Compressed VQA, $n_a = 2$	10000	200
$21 \leq \#Variables < 100$	Compressed VQA, $n_a = 7$	20000	250
$\#Variables \geq 100$	Compressed VQA, $n_a = 9$	50000	250

Table 5.4: Algorithm selection strategy based on the number of decision variables.

5.3.3 Validating on a Complex Use Case

We will test both solvers on two different versions of scheduling simulators:

- **Fixed timestep rescheduling**: New wave of trucks are deployed are regular, pre-defined intervals (e.g., every 30, 60, 90 minutes)
- **Variable timestep rescheduling**: Rescheduling is triggered when a predefined percentage of vehicles become available again (e.g., 60%, 70%, 80%)

The use case we will demonstrate is more complex than the quantum cases and contains real world parameters.

- LCs: 3
- DPs: 3
- APs: 3
- Trucks: 13
  - Types: 3
- Barges: 6
- Goods: 42
  - Types: 4

- Some goods need to be dropped off at the Departure Ports. The trucks that aren't dropping off the goods need to embark to the barges.
- When multiple barges are connected (n-connected), they are treated as one combined barge unit for both loading and unloading operations.

Representation of the Joining and Disjoining of Barges

Barge Joining Logic. When two or more barges are connected, they are considered as a single transport unit during loading and unloading. The total capacity of the unit increases with the number of connected barges, but so does the time required for cargo handling. In other words, larger combined barges reduce the number of loading/unloading events but incur longer processing times per event.

We do not include barge merging in the optimization because it is not a decision that requires optimization; it is strictly determined based on the goods that need to be transferred. If we are compelled to merge barges due to specific requirements - such as a truck that needs two barges - we do so; otherwise, we keep them separate. Generally, a reasonable approach is to deploy all the cargo that requires barge merging at the beginning, and once those goods are transported, we split the barges and continue with single barges thereafter. This way, we only have to handle the merging and unmerging procedures once, saving time in the overall process. If the full list of goods is not known in advance, we can take the reverse approach by starting with single barges and merging them later, once all the goods requiring barge merging have been identified.

Results Overview

The whole architecture of the optimization engine is shown in figure: 5.8. The input data are processed and translated to the initial *Transport Environment*. Then the Scheduler (*Fixed Timestep or Variable Timestep Scheduler*) calls a *Solver* (classical or quantum) to perform optimization using the current environment variables and return the selected truck and barge assignments. Then the parameters are updated (truck and barge availability, goods assigned to trucks and barges) and the Solver is executed again using the updated new environment variables.

The output is a schedule, with details about each trip. We have Land trips and Sea trips. Each trip has an id and contains information about the assigned vehicle and goods loaded, as well as the starting location, the destination, and the travel times (depart, arrival). **The output is provided as an Excel spreadsheet. A convenient way to**

visualize the trips along the time axis is through a Gantt chart (Figure: 5.9). A Gantt chart is a type of bar chart that illustrates a schedule by showing activities (in this case, trips) as horizontal bars positioned over a time axis, making it easy to track their duration and overlaps. The labels of each bar are the trip ids. The Land Trips are the green bars with labels '$L : tripID$'. The Sea trips are the blue bars with labels '$S : tripID$'. The $y - axis$ corresponds to the vehicles '$LV : vehicleID$' for land vehicles and '$SV : vehicleID$' for sea vehicles. The Gantt charts in Figure 5.9 show the trips that were assigned to each vehicle using different solvers and schedulers. The Gantt chart only shows the trips across the time. **The vertical gap between a set of land trips and a set of sea trips is the unloading from trucks and loading to barges** or the barge joining/unjoining, or some other process that requires time.

To compare the two solvers on both schedulers, we visualize how goods are transported over time in Figure: 5.10. The classical solver outperforms the quantum solver on both schedulers, as expected, with the Variable Timestep with an 80% availability using the classical solver being the clear winner. As we saw in the previous section, the quantum solver could reach a cost with a 5%, 10% difference at best from the optimal. This difference may seem insignificant, but if there is a gap from the optimal in every timestep, the result of the total schedule would differ as well. The transportation of all the goods finishes at time 880 using the Variable Timestep Scheduler with the Classical Solver, at time 900 using the Fixed Timestep Scheduler with the Classical Solver, at time 1850 using the Variable Timestep Scheduler with the Quantum Solver and at time 2100 using the Fixed Timestep Scheduler with the Quantum Solver. The Fixed Timestep Scheduler using quantum solver seems to take the worst decisions. The Variable Timestep scheduler with the Quantum Solver took suboptimal decisions when the remaining goods were less than 10. Until then the transportation was conducted very similarly to the classical solver. The goods that require barge joining delayed the rest of the transportation that's why it took more time. This is a parameter that is not included on the optimization model so it is hard to be avoided by the solver.

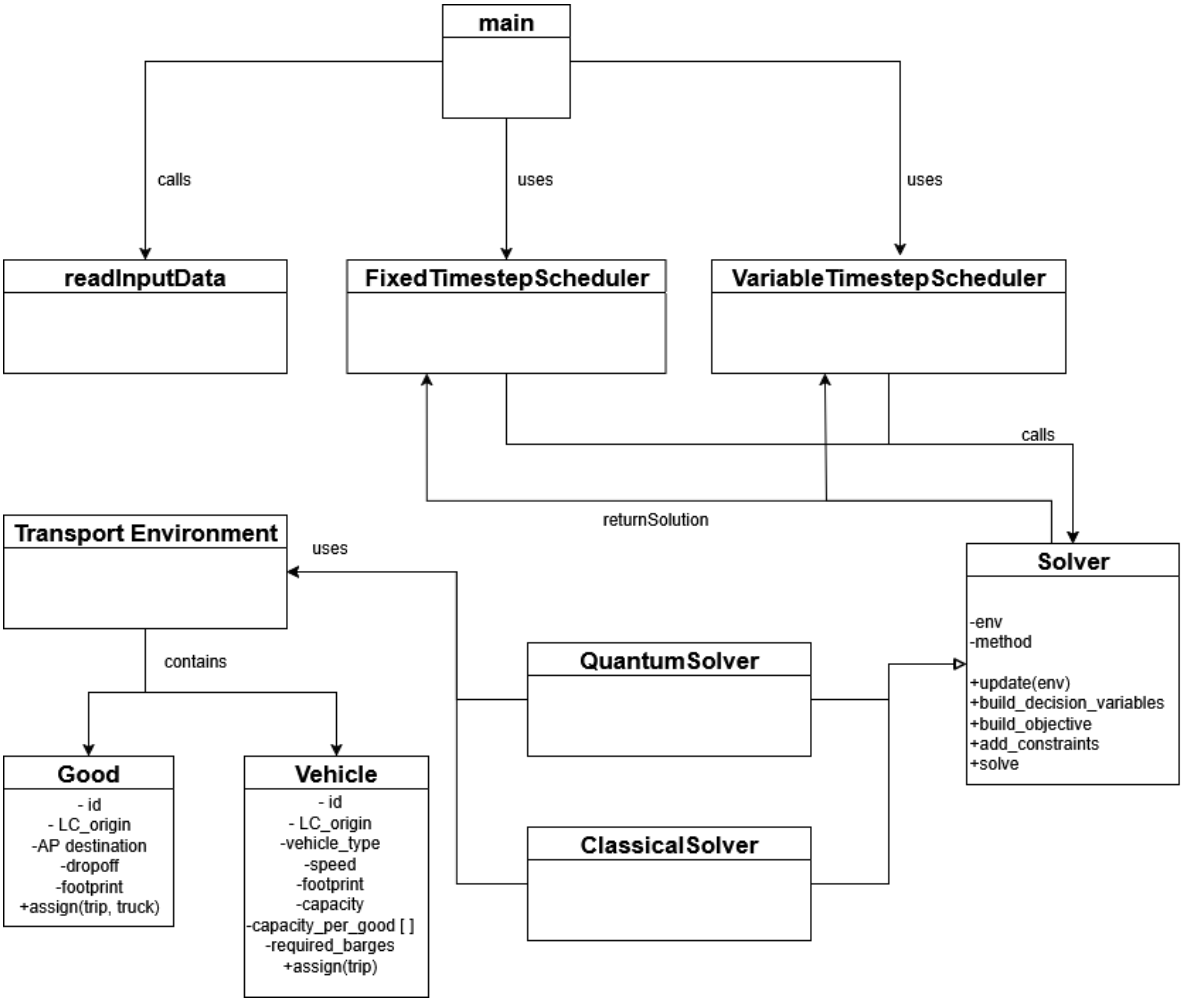
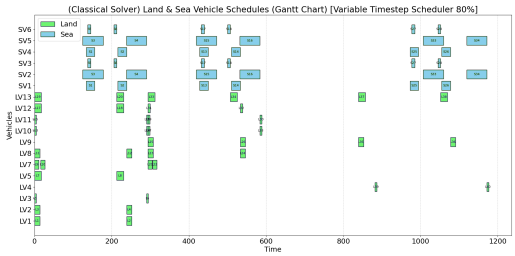
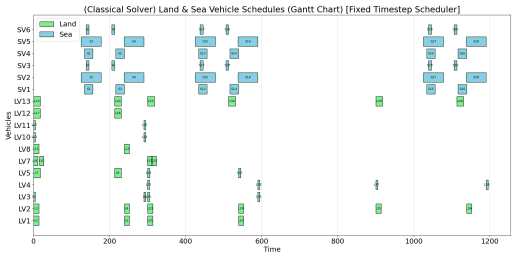


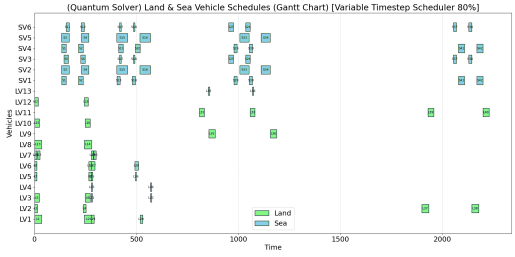
Figure 5.8: UML-Based High-Level View of the Optimization Engine.



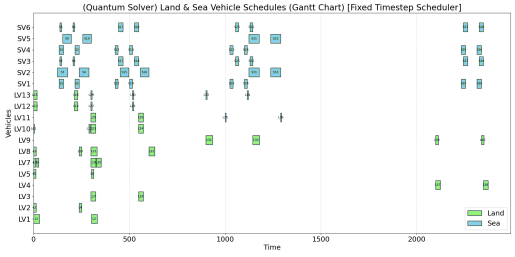
(a) Gantt chart, using classical solver, variable timestep 80% availability in each optimization run.



(b) Gantt chart, using classical solver, fixed timestep, timestep value: 100, perform optimization in every step.



(c) Gantt chart, using quantum solver, variable timestep 80% availability in each optimization run.



(d) Gantt chart, using quantum solver, fixed timestep, timestep value: 100, perform optimization in every step.

Figure 5.9: Quantum & Classical Solver, Land & **Sea** Scheduling.

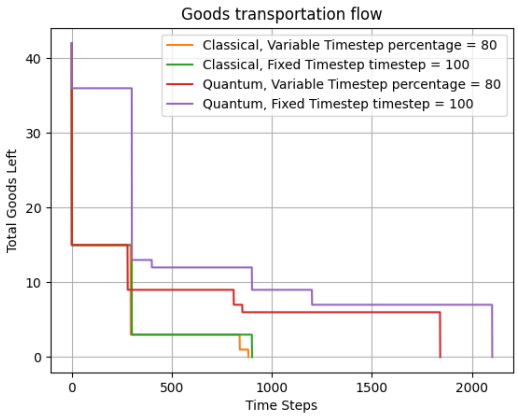


Figure 5.10: Transported goods across time.

Chapter 6

Conclusion and Future Work

In this thesis we explored the capabilities of hybrid quantum-classical algorithms, focusing on Transshipment Optimization and Scheduling. Although classical optimization techniques currently outperform our results, it is encouraging to note that quantum computers are, in principle, capable of addressing such problems and hold the potential to surpass classical methods in the near future—once devices acquire more fault-tolerant qubits. The fundamentally different way in which quantum systems process information, coupled with the possibility of significant speedups in specific applications, makes research in this field a highly promising and exciting endeavor.

While this thesis demonstrates promising applications of quantum algorithms in optimization and transport scheduling, several directions remain open for future exploration: **Scaling to larger problem instances:** Current experiments were limited by the number of available qubits and noise on NISQ devices. Future research could explore error-mitigation techniques, advanced circuit compilation strategies, or distributed quantum-classical workflows to scale these methods to industry-sized problems. **Improved qubit compression strategies:** The qubit-efficient encodings discussed here represent a step towards reducing resource requirements. Investigating ways to import MILP mathematics to the cost function of the quantum part may enable more accurate solutions with fewer qubits. **Application to other industries:** Although this thesis focused on transport optimization, similar formulations arise in logistics, finance, energy grid management, network routing, community detection and transport resilience that could yield valuable insights. **Advancing quantum generative models:** The proposed QAOA-GPT highlights the potential of combining large language models with quantum circuit design. Future work may extend this line of research by using better graph embeddings generated from quantum walks and train a model that can potentially solve

larger scale industry problems efficiently.

These directions show that, although quantum computing is still in its early stages, practical techniques are already being applied to real-world problems, accelerating progress and opening opportunities to refine methods, broaden applications, and close the gap between theory and practice.

Bibliography

[1] Boxuan Ai.
Discrete-time quantum walks: A quantum advantage for graph representation, 2024.

[2] Scott D. Berry, Paul Bourke, and Jingbo B. Wang.
qwviz: Visualisation of quantum walks on graphs.
Computer Physics Communications, 182(10):2295–2302, 2011.

[3] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann.
A quantum approximate optimization algorithm, 2014.

[4] Lorenzo Leone, Salvatore F.E. Oliviero, Lukasz Cincio, and M. Cerezo.
On the practical usefulness of the hardware efficient ansatz.
Quantum, 8:1395, July 2024.

[5] Ho Lun Tang, V.O. Shkolnikov, George S. Barron, Harper R. Grimsley, Nicholas J. Mayhall, Edwin Barnes, and Sophia E. Economou.
Qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor.
PRX Quantum, 2(2), April 2021.

[6] Harper R. Grimsley, Sophia E. Economou, Edwin Barnes, and Nicholas J. Mayhall.
An adaptive variational algorithm for exact molecular simulations on a quantum computer.
Nature Communications, 10(1), July 2019.

[7] Linghua Zhu, Ho Lun Tang, George S. Barron, F. A. Calderon-Vargas, Nicholas J. Mayhall, Edwin Barnes, and Sophia E. Economou.
An adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer, 2022.

[8] Benjamin Tan, Marc-Antoine Lemonde, Supanut Thanasilp, Jirawat Tangpanitanon, and Dimitris G. Angelakis.
Qubit-efficient encoding schemes for binary optimisation problems.
Quantum, 5:454, May 2021.

[9] Elias X. Huber, Benjamin Y. L. Tan, Paul R. Griffin, and Dimitris G. Angelakis.
Exponential qubit reduction in optimization for financial transaction settlement.
EPJ Quantum Technology, 11(1), August 2024.

[10] Ioannis D. Leonidas, Alexander Dukakis, Benjamin Tan, and Dimitris G. Angelakis.
Qubit efficient quantum algorithms for the vehicle routing problem on nisq processors, 2023.

[11] Yidong Liao and Chris Ferrie.
Gpt on a quantum computer, 2024.

[12] Ilya Tyagin, Marwa H. Farag, Kyle Sherbert, Karunya Shirali, Yuri Alexeev, and Ilya Safro.
Qaoa-gpt: Efficient generation of adaptive and regular quantum approximate optimization algorithm circuits, 2025.

[13] Benedek Rozemberczki and Rik Sarkar.
Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models, 2020.

[14] Andrej Karpathy.
NanoGPT.
`https://github.com/karpathy/nanoGPT`, 2022.

[15] Claudia Archetti, Lorenzo Peirano, and M. Grazia Speranza.
Optimization in multimodal freight transportation problems: A survey.
European Journal of Operational Research, 299(1):1–20, 2022.

[16] Laurent Perron and Vincent Furnon.
OR-Tools.
`https://developers.google.com/optimization/`.