DIPLOMA THESIS

Hybrid quantum-classical algorithms and applications in scheduling problems

Author:

Theodoros Kalamarakis

Thesis Committee:

Prof. Dimitris G. Angelakis (Supervisor)

Prof. Dionysios Christopoulos

Associate Prof. Vasileios Samoladas

School of Electrical & Computer Engineering

Technical University of Crete

July 25, 2024

Abstract

This thesis explores hybrid quantum-classical algorithms, focusing on their applications in scheduling problems. It begins by outlining the foundational principles of quantum mechanics, including the role of qubits, quantum gates, and entanglement. The study then examines Quadratic Unconstrained Binary Optimization (QUBO) and its significance in combinatorial optimization, highlighting the use of classical and quantum methods such as Grover's algorithm and quantum annealing. Next, the thesis introduces qubit-efficient encoding schemes for quantum optimization, designed to enhance the representation and processing of binary optimization problems in near-term quantum computers. These schemes are analysed through theoretical exploration and practical testing on quantum simulators and cloud-based quantum hardware, focusing on the Max-Cut and Subset Sum problems. A significant portion is dedicated to the smart scheduling of EV charging, demonstrating how qubit-efficient quantum optimization can efficiently manage large-scale EV charging schedules. The study compares the performance of quantum algorithms against classical approaches, presenting detailed results and insights from noisy simulations and real-world executions on cloud quantum processing units (QPUs). The findings suggest that hybrid quantum-classical algorithms hold substantial promise in solving complex scheduling problems, offering more efficient and scalable solutions. This work contributes to the growing body of knowledge in quantum computing applications, providing practical frameworks and methodologies for future research and industry applications.

Contents

Chapter 1

Introduction

Path to Quantum Supremacy

The quest for computational supremacy has been a relentless pursuit, stretching from the mechanical calculators of the 17th century, pioneered by figures like **Blaise Pascal** and **Gottfried Wilhelm Leibniz**, through the invention of the electronic computer in the 20th century, to the dawn of the quantum computing era in the late 20th and early 21st centuries. Quantum computing, a concept that first gained traction through the theoretical work of physicists like **Richard Feynman** and **David Deutsch**, emerges as a groundbreaking paradigm with the potential to revolutionize various domains through its inherent capabilities far surpassing those of classical computing. Unlike classical computers, which process information in binary bits (0s and 1s), quantum computers utilize quantum bits, or qubits, that can exist in multiple states simultaneously, thanks to the principles of superposition and entanglement. This fundamental difference allows quantum computers to perform complex calculations more efficiently than their classical counterparts, showcasing potential advantages in several critical areas.

Quantum Supremacy and Beyond

Quantum supremacy, a term first used by **John Preskill** in 2012, describes a quantum computer's ability to solve a problem that is practically impossible for classical computers, serving as a significant milestone in the field's evolution. However, moving beyond

quantum supremacy involves leveraging these capabilities to address real-world problems, highlighting areas where quantum computing holds a distinct advantage.

Impacts of Quantum Computing

Quantum computing presents significant challenges and opportunities across various fields. In cryptography, quantum algorithms, such as Shor's algorithm introduced by Peter Shor in 1994, can decrypt widely used encryption methods, necessitating a reevaluation of security protocols. Conversely, quantum cryptography offers theoretically unbreakable encryption based on quantum mechanics principles. Quantum computing's ability to simulate molecular and quantum systems also provides unparalleled opportunities in drug discovery and materials science. While classical computers struggle to model complex chemical reactions, quantum computers can manage vast datasets and execute calculations much faster, expediting new drug and material development. Additionally, the confluence of quantum computing and artificial intelligence (AI) holds potential to enhance machine learning algorithms. Quantum computers' capacity to process and analyze large datasets simultaneously could significantly improve AI's efficiency and power, leading to advancements in pattern recognition, natural language processing, and predictive analytics.

Optimization Problems

Besides the previously mentioned filed, quantum computing shows remarkable advantages in solving optimization problems in various industries, including logistics, finance, routing, energy, and scheduling. Quantum algorithms are very good at finding the best solution from many possibilities, a task that becomes extremely difficult for classical computers as problems grow larger. In logistics, quantum computing can make supply chain management more efficient and cost-effective. In finance, it can improve investment strategies by enhancing portfolio optimization and risk assessment. For transportation, quantum computing can find the shortest and most efficient routes, saving time and resources. In the energy sector, quantum algorithms can make power grid management and energy distribution more sustainable and reliable. Additionally, quantum computing can handle complex scheduling problems, such as workforce allocation, project management,

and manufacturing processes, ensuring optimal use of resources and time. These abilities make quantum computing a powerful tool for solving difficult optimization problems that classical computers struggle with.

1.1 Thesis Outline

1. **Background in Quantum Mechanics**
   This chapter covers the fundamental principles of quantum mechanics necessary for understanding quantum computing. Topics include quantum spin, quantum states, and essential linear algebra concepts, as well as the principles governing quantum systems and their time evolution.

2. **Quadratic Unconstrained Binary Optimization Problem**
   This chapter explains the Quadratic Unconstrained Binary Optimization (QUBO) problem, a key focus of the thesis. It discusses the formulation of QUBO problems, their relevance in combinatorial optimization, and various classical and quantum methods for solving them.

3. **Qubit-Efficient Encoding Schemes for Binary Optimization** Problems
   This chapter explores various encoding schemes designed to efficiently represent binary optimization problems using qubits. It delves into the cost functions associated with these schemes, their limitations, and the performance of different encoding methods in practical scenarios such as the Subset Sum problem.

4. **Applications of Encoding Schemes for Industry Optimization**
   This chapter applies the discussed encoding schemes to real-world industry optimization problems. It specifically focuses on the new problem of smart scheduling of electric vehicle (EV) charging, demonstrating how qubit-efficient quantum optimization can be implemented and comparing it with classical algorithms.

Chapter 2

Backround In Quantum Mechanics

2.1 Quantum Spin and Quantum Bits

In the realm of quantum computing, a quantum bit, or *qubit*, for short, is the fundamental unit of quantum information, analogous to a bit in classical computing. However, unlike classical bits, which exist solely in states of 0 or 1, qubits can exist in multiple states simultaneously, thanks to the principles of superposition and entanglement. There are several physical implementations of qubits, including superconducting qubits, trapped ion qubits, and photonic qubits, each with its own set of advantages and challenges. To conceptually grasp and articulate the fundamental rules of quantum mechanics, it is beneficial to employ a theoretical model that exemplifies the nature of a qubit. This model is often the quantum spin. Technically, we discuss the spin of an electron, which is an intrinsic property, but in theoretical terms, it can be considered independently as a model for understanding qubits. Quantum spin is often visualized as an arrow that points in a particular direction.

2.1.1 Quantum Spin Experiment

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. If we perform the measurement multiple times without disturbing the post-measurement state of the quantum system, we will consistently receive the same result in a deterministic fashion. However, if we manage to reset the quantum system to its initial state between each execution of the experiment, the results of each measurement will be uncertain. This outcome introduces us to two major characteristics of quantum mechanics that distinguish it from classical physics: the probabilistic nature of quantum systems and the fact that measurement impacts and alters the state of the system. These aspects lead to unique and abstract features of quantum mechanics, such as the Heisenberg Uncertainty Principle. This principle, which is influenced by the disruptive nature of measurement, states that we cannot simultaneously determine the position and momentum of a particle with precise accuracy. Furthermore, this principle can be generalized to any two non-commutative properties of the particle.

2.2 Introduction to Quantum States

Undoubtedly, the new and non-intuitive concepts revealed by these experiments stretched beyond the bounds of classical understanding and necessitated the creation of a novel language. This led directly to the development of quantum mechanics. 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:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \in \mathbb{C}^2$$

Here, α and β are known as complex probability amplitudes. They are connected to probabilities, even if this connection isn't obvious at first. By themselves, α and β 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 $|\alpha|^2 = \alpha^*\alpha$, and to the state $|1\rangle$ with a probability of $|\beta|^2 = \beta^*\beta$. 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 α and β. This means the probabilities must add up to 1:

$$|\alpha|^2 + |\beta|^2 = 1.$$

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

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

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:

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

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

2.3 Fundamental Linear Algebra Concepts

2.3.1 Definitions

Definition 2.3.1. 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 $\mathbb{C}^n$, then,

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

where $c_i \in \mathbb{C}$.

Definition 2.3.2. Hermitian Conjugate The Hermitian conjugate, or adjoint (†), is the complex conjugate of a transposed matrix:

$$M^\dagger = [M^*]^T.$$

Definition 2.3.3. Hermitian Operator Hermitian operators, also called self-adjoint operators, are operators (or matrices) that are equal to their adjoint (or conjugate transpose):

$$M^\dagger = M.$$

Definition 2.3.4. Eigenvalues and Eigenvectors An eigenvector $|v_i\rangle$ of an operator $M \in \mathbb{C}^{n\times n}$ is a non-zero vector that satisfies:

$$M|v_i\rangle = \lambda_i|v_i\rangle,$$

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.

Definition 2.3.5. 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:

$$A = \sum_i \lambda_i|\lambda_i\rangle\langle\lambda_i|,$$

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

Definition 2.3.6. Normal Operator A linear operator $M \in \mathbb{C}^{n\times n}$ is normal if it commutes with its adjoint:

$$MM^\dagger = M^\dagger M.$$

Definition 2.3.7. 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:

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

Definition 2.3.8. 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:

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

Projective operators are mutually orthogonal and form a complete set:

$$\sum_i P_i = I.$$

Definition 2.3.9. 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.

Definition 2.3.10. 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}.

Tensor products are crucial in quantum computing and will be used extensively in subsequent chapters. Below are the important properties of tensor products.

2.4 Principles of Quantum Mechanics

To grasp the concept of qubits, it's crucial to understand the rules governing the quantum realm. Unlike classical physics, we lack the natural intuition to comprehend and visualize quantum phenomena. Our brains are wired for the macroscopic world, governed by Newtonian physics, so the idea of being in two states simultaneously doesn't make sense. However, we can rely on abstract mathematics to bridge this gap. Principles in quantum mechanics are fundamental assumptions that form the foundation of the theory, helping us build a framework to make experimentally testable predictions.

Principle 1: 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.

Principle 2: 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 $\hat{\sigma}_x, \hat{\sigma}_y$, and $\hat{\sigma}_z$. 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.

Principle 3: Measurement and Eigenvalues

When measuring an observable quantity, the result will always be one of the eigenvalues λ_i 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.

Principle 4: 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

$$\text{pr}(\lambda_i) = |\langle\psi|\lambda_i\rangle|^2 = \langle\psi|\lambda_i\rangle\langle\lambda_i|\psi\rangle,$$

where λ_i is an eigenvalue and $|\lambda_i\rangle$ is an eigenvector of the Hermitian operator describing M. Since the operator is Hermitian, the possible measurements are distinct. We define the product $|\lambda_i\rangle\langle\lambda_i|$ as a Projective Operator P_i. This principle ties together quantum states and measurements, highlighting that all information about a quantum state is contained within $|\psi\rangle$. However, we can only access this information by interacting with and altering the system. Thus, any computation should happen before observation.

For spin qubits, the observable information includes the three spin components $\sigma_z, \sigma_x, \sigma_y$, described by Hermitian operators. Hermitian operators can be decomposed into mutually orthogonal eigenvectors, each associated with a unique real number, called an eigenvalue. When we measure a spin component, we obtain one of the eigenvalues, allowing us to infer the system's state before observation. Measurement can only definitively identify eigenvector states, but any state can be observed because the eigenvectors of each operator form an orthonormal basis. Each state can be written as a linear combination of the eigenvectors we intend to measure. The computational basis $\{|0\rangle, |1\rangle\}$ we defined earlier is simply the basis formed by the eigenvectors of the σ_z operator.

2.5 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:

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

An interesting and important point about this equation is that the operator responsible for the evolution of an isolated quantum system is unitary. This implies that the action of operators on quantum states is, in fact, unitary transformations. In the next section, we will see how this principle is utilized to realize quantum circuits.

2.6 Quantum Gates

For an operator to be considered a quantum gate, it must be a unitary operator. Previously, we described a quantum state as a vector within a multi-dimensional Hilbert space. Quantum gates function by rotating these qubits while preserving their norm (magnitude). This preservation is crucial for maintaining the probabilities in quantum mechanics. Additionally, the evolution of a closed quantum system over time needs to be reversible. This means that if an operator U transforms state $|s_1\rangle$ into state $|s_2\rangle$, its inverse, $U^\dagger$, should revert $|s_2\rangle$ 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.

2.6.1 Single Qubit Gates

In the context of spin qubits, each quantum state can be represented as a combination of three observable components: σ_z, σ_x, and σ_y. These components correspond to Hermitian operators known as the Pauli operators, named after physicist Wolfgang Pauli. The matrix representations and circuit symbols for these operators are:

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

Since these operators are Hermitian, they can be encoded as the Hamiltonian of a system. The time evolution of these operators generates rotations around the three coordinate axes x, y, and z of the Bloch Sphere. The matrix representation of these rotation operators is derived by solving the time-dependent Schrödinger Equation:

$$R_x(\theta) = e^{-i\theta/2X} = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \quad R_y(\theta) = e^{-i\theta/2Y} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$R_z(\theta) = e^{-i\theta/2Z} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$$

2.6.2 Hadamard Gate

Another fundamental quantum gate is the Hadamard gate, which 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:

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

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$$

These states are interesting because, 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. Superposition, which allows a qubit to be in both states simultaneously, enables quantum computers to perform computations that classical computers cannot. However, superposition alone is not enough to provide a computational advantage. The phenomenon of entanglement, explained later, is also crucial.

2.6.3 Multi-Qubit Gates

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

Definition 2.6.1. Local Operators Local operators are unitary operators that can be factorized using a tensor product. An operator is local if it acts on only one part of the composite system. The matrix representation of these gates is:

$$X \otimes (ZH) \quad \text{for the left circuit}$$

$$(XH) \otimes (ZHY) \quad \text{for the right circuit}$$

Changing the endianness alters the positions of the gates in the matrix representation as a tensor product. The operation's matrix representation can be broken into a tensor product of smaller operators, indicating it is a local operator. The tensor product can extend to more extensive systems. By correctly placing the operators within the tensor product, we can design any desired local gate. In quantum circuit diagrams, the lines (often perceived as wires) represent the progression of time from left to right. However, local gates do not fully utilize the computational power that quantum mechanics offers. Non-local operators exploit the unique quantum phenomena arising from interactions between particles, such as entanglement, which allows computations that would otherwise be impossible.

Definition 2.6.2. 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:

$$UP_0 \otimes I + P_1 \otimes U$$

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:

$$CNOT = P_0 \otimes I + P_1 \otimes X = \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The CNOT gate is crucial because it has a universal behavior. A universal set of quantum gates is a set of gates that can express any other unitary operation as a finite sequence of gates from that set. CNOT and arbitrary single qubit rotation gates form one such universal set of quantum gates.

2.7 Entanglement

Entanglement is one of the most intriguing and essential phenomena in quantum mechanics. It occurs when the quantum states of two or more qubits become intertwined such that the state of each qubit cannot be described independently of the state of the others. This interconnectedness persists even when the qubits are separated by large distances.

When qubits are entangled, measuring the state of one qubit instantaneously affects the state of the other, regardless of the distance between them. This phenomenon enables quantum computers to perform tasks that are infeasible for classical computers.

One of the simplest and most famous examples of entangled states are the Bell states.

These are specific quantum states of two qubits that represent the maximal entanglement between them. The four Bell states are given by:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

These Bell states form a basis for the space of two-qubit states and are fundamental in various quantum information protocols, such as quantum teleportation and superdense coding.

Entanglement allows quantum computers to process and store information in ways that classical computers cannot. By exploiting the correlations between entangled qubits, quantum algorithms can solve certain problems more efficiently, showcasing the potential computational advantage of quantum systems.

2.8 Grover's Algorithm

Grover's algorithm [1] is one of the most well-known quantum algorithms, offering a clear quantum advantage. It can find an element in an unsorted database much faster than classical computers. In classical computing, searching through an unstructured database involves checking each element one by one until the desired element is found. If the database contains N elements, this process takes on average $\frac{N}{2}$ steps, resulting in a runtime of $O(N)$. In contrast, Grover's algorithm can perform this search in $O(\sqrt{N})$, providing a quadratic speed-up. For example, given a list of 1 million elements, a quantum computer using Grover's algorithm could find the desired element in about 1000 steps.

2.8.1 The Algorithm step by step

Imagine we have an unstructured list of $N = 2^n$ elements and we want to find the index of a specific element, marked as ω. We define a function f such that $f(x) = 1$ if $x = \omega$ (the desired element) and $f(x) = 0$ otherwise:

$$f : \{0, 1, \ldots, N-1\} \to \{0, 1\}$$

$$f(x) = \begin{cases} 1 & \text{if } x = \omega \\ 0 & \text{otherwise} \end{cases}$$

All elements in the list can be encoded into quantum states like $|index\rangle \otimes |data\rangle$. To encode all possible indices, we need $O(\log_2 N) = O(n)$ qubits, plus some ancilla qubits for the data. This is similar to classical Random Access Memory (RAM).

We access f through an oracle, a unitary operator U_ω that acts on the data register:

$$U_\omega|x\rangle = \begin{cases} -|x\rangle & \text{if } x = \omega \\ |x\rangle & \text{otherwise} \end{cases}$$

This can be generalized as $U_\omega|x\rangle = (-1)^{f(x)}|x\rangle$, and the unitary matrix of this operator is:

$$U_\omega = \begin{pmatrix} (-1)^{f(0)} & 0 & \cdots & 0 \\ 0 & (-1)^{f(1)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (-1)^{f(N-1)} \end{pmatrix}$$

Initially, the system is prepared in a uniform superposition of all indices:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \otimes |data_i\rangle$$

For simplicity, we can represent this state as:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} |n\rangle$$

This state can be visualized as a normalized sum of two orthogonal vectors: one representing a uniform superposition of all elements except ω, and one containing only ω:

$$|s\rangle = \sin\theta|\omega\rangle + \cos\theta|s'\rangle, \quad |s'\rangle = \frac{1}{\sqrt{N-1}} \sum_{n \neq \omega} |n\rangle$$

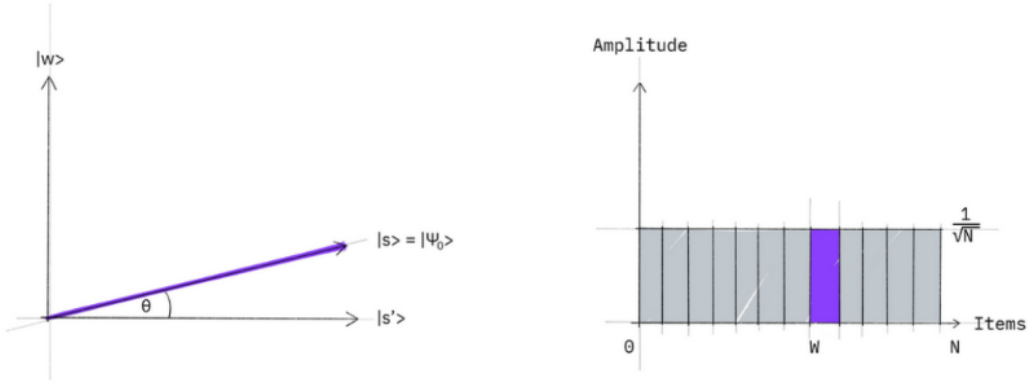


Figure 2.1: Grover's Algorithm initial state

The angle θ is calculated by projecting $|s\rangle$ onto $|s'\rangle$:

$$\langle s'|s\rangle = \frac{1}{\sqrt{N-1}} \cdot \frac{1}{\sqrt{N}} \sum_{n \neq \omega} \langle n|n\rangle = \sqrt{\frac{N-1}{N}}$$

Thus, the angle θ is:

$$\cos\theta = \sqrt{\frac{N-1}{N}}, \quad \sin^2\theta + \cos^2\theta = 1, \quad \sin^2\theta = \frac{1}{N}, \quad \sin\theta = \frac{1}{\sqrt{N}}, \quad \theta = \arcsin\left(\frac{1}{\sqrt{N}}\right)$$

Next, we apply the unitary transformation U_ω:

$$U_\omega|s\rangle = (I - 2|\omega\rangle\langle\omega|)(\sin\theta|\omega\rangle + \cos\theta|s'\rangle) = -\sin\theta|\omega\rangle + \cos\theta|s'\rangle$$

The final step is to reflect the state $U_\omega|s\rangle$ around the mean using Grover's Diffusion operator U_s:

$$U_s = 2|s\rangle\langle s| - I$$

Applying U_s gives:

$$U_s U_\omega|s\rangle = (2|s\rangle\langle s| - I)(-\sin\theta|\omega\rangle + \cos\theta|s'\rangle)$$

Using $\langle s|\omega\rangle = \sin\theta$ and $\langle s|s'\rangle = \cos\theta$:

$$U_s U_\omega|s\rangle = \sin(3\theta)|\omega\rangle + \cos(3\theta)|s'\rangle$$

With each iteration, the angle increases by 2θ. To reach $|\omega\rangle$, repeat $U_s U_\omega$ such that

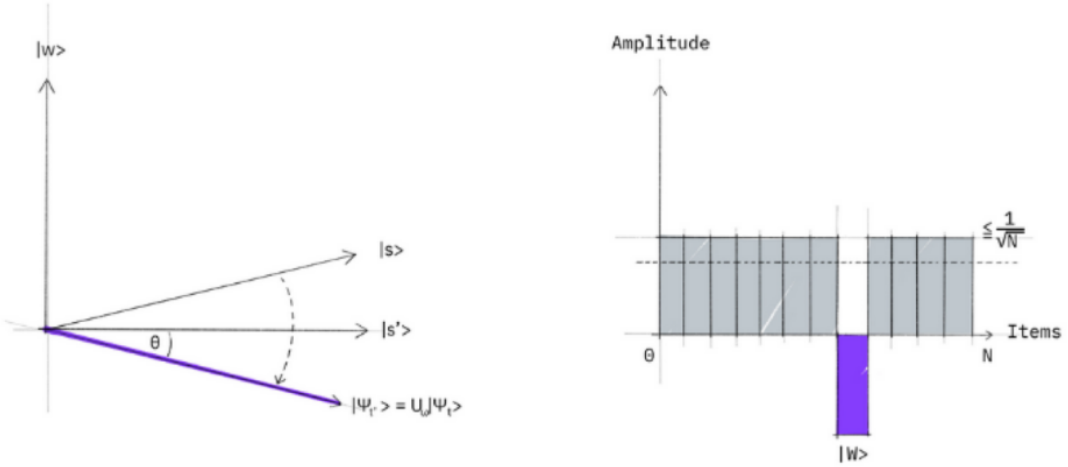


Figure 2.2: Grover's Algorithm: state after applying U_ω

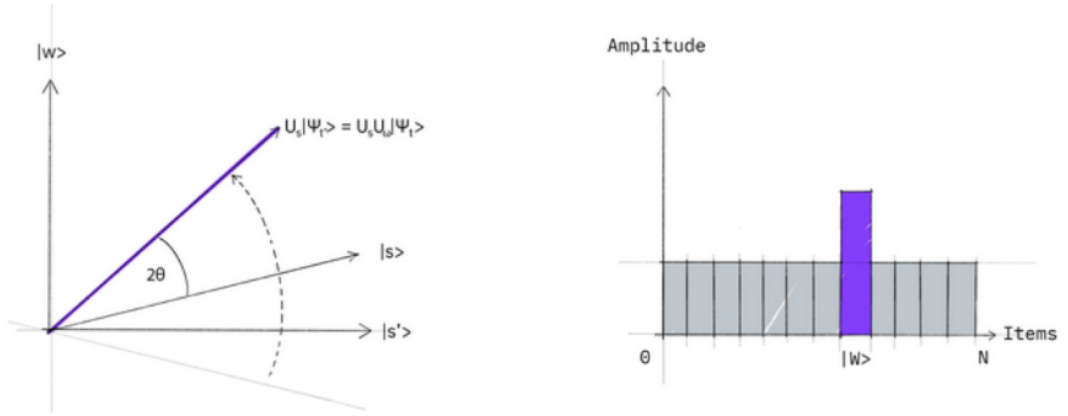


Figure 2.3: Grover's Algorithm: state after applying U_s

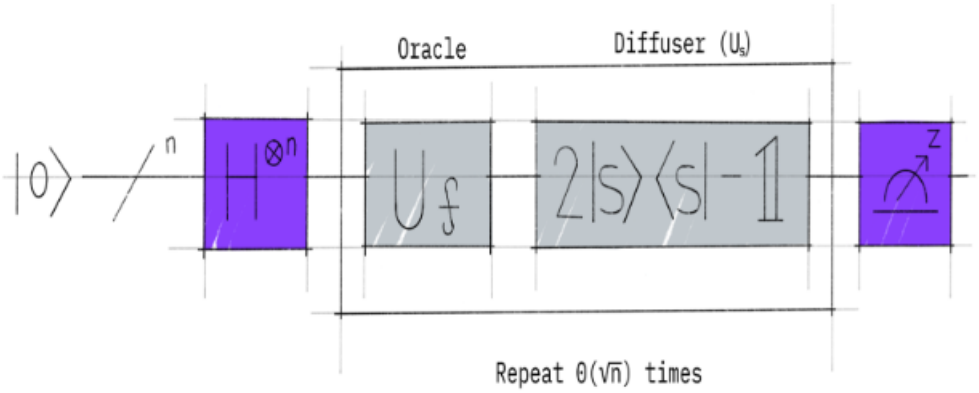


Figure 2.4: Grover's Algorithm: circuit

$\sin((2k+1)\theta) = 1$, where k is the number of iterations. This process is called amplitude amplification.

Given $\theta = \arcsin\left(\frac{1}{\sqrt{N}}\right)$, the number of steps required is:

$$k = \frac{\pi}{4}\frac{1}{\arcsin\left(\frac{1}{\sqrt{N}}\right)} + \frac{1}{2}$$

For large N:

$$\lim_{N\to\infty} \sin(\theta) = \lim_{N\to\infty} \sin\left(\frac{1}{\sqrt{N}}\right) \approx \frac{1}{\sqrt{N}} \Rightarrow \arcsin(x) \approx x$$

Thus, the number of steps approximates to:

$$k = \frac{\pi}{4}\sqrt{N} + \frac{1}{2} \approx O(\sqrt{N})$$

Chapter 3

Quadratic Unconstrained Binary Optimization Problem

Combinatorial optimization problems constitutes a fundamental category within the realm of optimization, involving the arrangement, selection, or combination of elements to achieve specific objectives. These problems arise across diverse domains such as logistics, finance, telecommunications, and computer science, among others. At their core, combinatorial problems challenge us to find the best arrangement or selection from a finite set of possibilities, often under constraints or with specific goals in mind. Examples include finding the shortest path in a network, allocating resources efficiently, scheduling tasks optimally, and many others. The inherent complexity of combinatorial problems, coupled with their wide-ranging applications, makes them both intellectually stimulating and practically significant. Efficiently solving these problems can lead to improved decision-making, resource allocation, and overall optimization in various fields. We are specifically concerned with combinatorial optimization problems in the form of **Quadratic Unconstrained Binary Optimization (QUBO)**, focuses on problems characterized by binary variables (where each variable can only be 0 or 1) and a quadratic cost function, without any other constraints on how these variables interact. Despite their significant practical importance, efficiently solving these types of problems remains a challenge, likely because they belong to the $\mathbf{FP^{NP}}$ category of computational problems.

Definition 3.0.1. NP (Nondeterministic Polynomial-Time) is the set of decision problems solvable in polynomial time by a nondeterministic Turing machine, or equivalently, **NP** is the set of decision problems verifiable in polynomial time by a deterministic Turing machine.

Definition 3.0.2. FP (Function Polynomial-Time): This is the class of function problems that can be solved by a deterministic Turing machine in polynomial time. Unlike the class P, which deals with decision problems that have yes/no answers, FP deals with problems that require a function to be computed or a specific output to be produced.

Definition 3.0.3. Oracle Access to NP : An oracle for a complexity class, like NP, is a hypothetical device that answers decision problems in that class instantaneously. A machine that has access to such an oracle can make queries to the oracle and use the responses in its computation.

Definition 3.0.4. $\mathbf{FP^{NP}}$: This class consists of those function problems that can be solved by a deterministic polynomial-time Turing machine that is equipped with an oracle for answering NP problems. The Turing machine can make a polynomial number of calls to the NP oracle during its computation.

In more practical terms, FP^{NP} captures the complexity class of computing functions that may involve solving several NP-hard problems as subtasks, with each subtask potentially being 'outsourced' to an oracle. NP-hard problems are those that cannot be solved in polynomial time by any deterministic Turing machine, which is a theoretical model representing what can be computed by classical computers under specific conditions. In simpler terms, there are no known algorithms that allow classical computers to solve NP-hard problems in a time that scales polynomially with the size of the input. This is where researchers are innovating, and quantum computing naturally enters the conversation as a potential technology that could perform exponentially faster than existing classical computing methods. However, it's important to note that while quantum computing may offer a significant speed advantage for certain types of problems, it does not necessarily mean it can completely overcome the NP-hardness of these problems. We focus on QUBO problems because they are particularly compatible with quantum anneal-

ing and other quantum computing methodologies, which we will elaborate on later. But first, let's clearly define what a QUBO problem is.

3.1 Defining of the QUBO problem

QUBO problems are a type of combinatorial problem where the variables can only take binary values of 0 or 1. These 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 $\mathbf{x}^*$ such that

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \quad C(\mathbf{x}) := \mathbf{x}^\top \mathbf{Q} \mathbf{x} \tag{3.1}$$

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

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \quad C(\mathbf{x}) := \sum_{i,j} x_i Q_{ij} x_j \tag{3.2}$$

To confirm an optimal solution has been identified, employing a brute force approach or exact solvers like Branch and Bound (or Branch and Cut) is a main strategy. Yet, the feasibility of such methods declines rapidly as the potential solutions increase exponentially with the problem size, a common scenario in industrial settings. This challenge has catalyzed the advancement of classical approaches designed to discover approximate solutions. Notable among these are comprehensive optimization platforms such as Gurobi, CPLEX, and SCIP. Additional classical strategies include relaxation techniques like the Goemans-Williamson algorithm, along with heuristics and metaheuristics such as Tabu search, Path relinking, evolutionary algorithms, and Simulated Annealing. A thorough examination of methods for addressing QUBO problems is available in the literature.

3.1.1 Quantum Computation for QUBO

Alternative strategies using quantum devices have also been proposed to solve QUBO problems. These include adaptations of Grover's search algorithm and amplitude amplification, where Grover's algorithm is tailored to efficiently search through the QUBO

solution space by using an oracle to identify optimal solutions. This oracle amplifies the probability of the optimal solution in the quantum state space, potentially reducing the solution time to $O(\sqrt{N})$, where N represents the size of the solution space.

However, given the limitations of existing NISQ devices, implementing Grover's Algorithm is impractical. Consequently, it is necessary to explore alternative methods that are capable of addressing the problem effectively.

3.2 The Ising Model

The Ising model is a mathematical model in statistical mechanics that has played a pivotal role in understanding phase transitions and critical phenomena. Named after Ernst Ising, who first analyzed the model in his doctoral thesis [2][3], the Ising model describes a system of interacting spins on a lattice.

3.2.1 Model Description

In its simplest form, the Ising model consists of discrete variables, called spins, which can take values of either +1 or -1. These spins are arranged on a lattice, and each spin interacts only with its nearest neighbors. The Hamiltonian of the Ising model, which describes the energy of a particular configuration of spins, is given by:

$$H_{Ising} = \sum_i \sum_j J_{ij} s_i s_j + \sum_i h_i s_i, \tag{3.3}$$

where s_i represents the spin at site i, J is the interaction strength between neighboring spins, and h is an external magnetic field affecting all spins. The J variables encode the type of the interaction. 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. Each magnet can also be manipulated via an external magnetic field, which we call bias and we denote as the h variables.

The Ising model, traditionally used in statistical physics to describe ferromagnetism, has found a profound application in the field of energy minimization problems in com-

putational science. This model's utility in energy minimization stems from its simple yet powerful representation of complex systems through binary variables, analogous to magnetic spins. The objective in energy minimization is to find the spin configuration that results in the lowest possible energy state of the system, known as the ground state. This problem is analogous to finding the global minimum of the energy landscape defined by the Hamiltonian. The task involves flipping the spins strategically to achieve configurations that continually lower the system's energy, which is analogous to solving optimization problems where one seeks the minimum cost or energy solution.

3.2.2 Computational Applications

In computational applications, the Ising model translates real-world problems into a framework of binary variables and interactions. For example, in optimization problems such as the traveling salesman problem or network design, variables and constraints can be mapped onto spins and their interactions in the Ising model. Each configuration of the spins represents a potential solution, and the model's dynamics, driven towards energy minimization, explore the solution space effectively.

It should now be evident why the Ising model is of interest in the context of QUBO problems. There is a clear similarity between the Ising Hamiltonian and the QUBO cost function. Therefore, by mapping the QUBO function to an Ising Hamiltonian, we can leverage natural laws to solve the optimization problem for us.

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

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

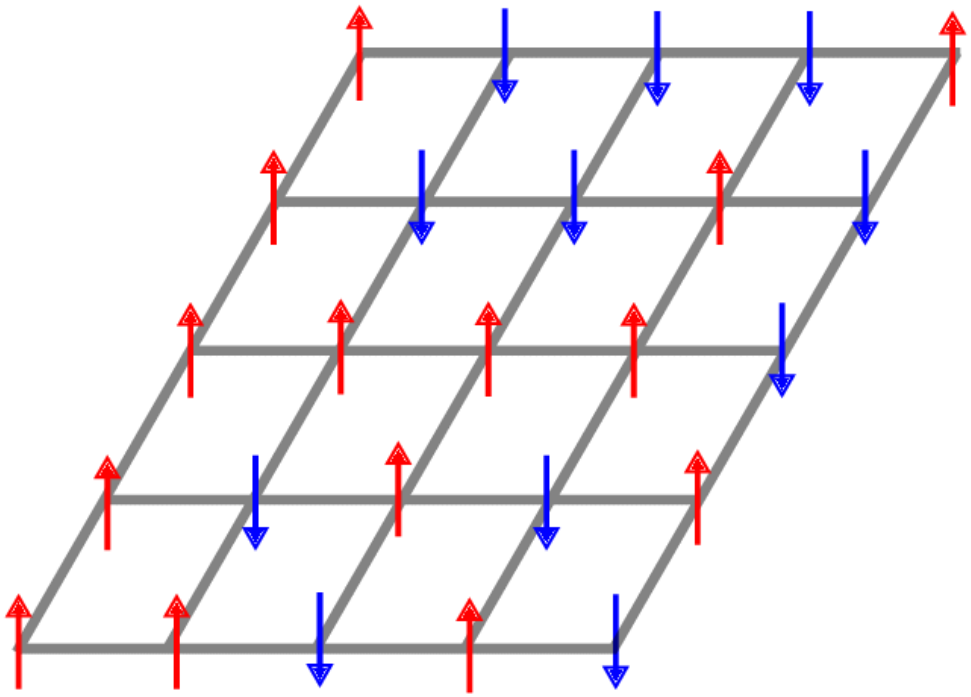


Figure 3.1: Simulation of the Ising model showing a phase transition from ordered to disordered state.

We simply need to substitute this into the expression:

$$
\begin{aligned}
C(\mathbf{x}) &= \sum_{i,j} x_i Q_{ij} x_j \xrightarrow{x_i = \frac{1+s_i}{2}} \\
C(\mathbf{s}) &= \sum_{i,j} \frac{1+s_i}{2} Q_{ij} \frac{1+s_j}{2} \\
&= \frac{1}{4} \sum_{i,j} Q_{ij}(s_i + s_j + s_i s_j + 1) \\
&= \frac{1}{4} \left(\sum_{i,j} Q_{ij} s_i s_j + 2 \sum_{i,j} \sum_{j} Q_{ij} s_i + \sum_{i,j} Q_{ij} + \sum_{i} Q_{ii} \right)
\end{aligned}
$$

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

$$
\frac{1}{4} \left(\sum_{ij} Q_{ij} + \sum_{i} Q_{ii} \right)
$$

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

Now that the mapping to the Ising model is complete, we need to convert the classical Ising model to its quantum counterpart. In this case, the Hamiltonian becomes an

operator and the minimization objective would be $\langle \hat{H}_{Ising} \rangle$. Therefore, the spin variables $s_i \in \{-1, 1\}$ should represent the eigenvalues of an operator acting on $|0\rangle$ and $|1\rangle$. This operator should be none other than the Pauli $\hat{\sigma}^z$. Thus, the Quantum Ising Hamiltonian would be:

$$\hat{H}_{Ising} = \sum_i \sum_j J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z + \sum_i h_i \hat{\sigma}_i^z, \tag{3.4}$$

3.3 Adiabatic Quantum Theorem and Quantum Annealing

3.3.1 Adiabatic Quantum Theorem

The Adiabatic Quantum Theorem is a fundamental concept in quantum mechanics which states that a quantum system remains in its instantaneous eigenstate if a given perturbation acts on it slowly enough and if there is a gap between the eigenvalue in question and the rest of the Hamiltonian's spectrum. Mathematically, consider a time-dependent Hamiltonian $H(t)$ which varies smoothly over time from $H(0) = H_i$ to $H(T) = H_f$. According to the adiabatic theorem, if the system is initially in the ground state $|\psi(0)\rangle$ of $H(0)$, and the evolution is slow enough, then the system will remain in the instantaneous ground state $|\psi(t)\rangle$ of $H(t)$ throughout the evolution.

3.3.2 Quantum Annealing

Quantum Annealing is a metaheuristic for solving combinatorial optimization problems by exploiting quantum mechanical effects [4],[5],[6]. It leverages the principles of the Adiabatic Quantum Theorem to find the global minimum of a given objective function.

Definition 3.3.1. A **heuristic** is a technique designed for problem solving more quickly when classic methods are too slow for finding an exact or approximate solution, or when classic methods fail to find any exact solution in a search space. This is achieved by trading optimality, completeness, accuracy, or precision for speed. In a way, it can be considered a shortcut.

Definition 3.3.2. A **metaheuristic** is a higher level procedure or heuristic designed to find, generate, or select a lower level procedure or heuristic (partial search algorithm) that may provide a sufficiently good solution for an optimization problem.

In Quantum Annealing, we encode the problem to be solved into an Ising model or a similar Hamiltonian H_{Ising} whose ground state represents the optimal solution. The annealing process starts with a simple initial Hamiltonian H_0 whose ground state is easy to prepare. The system is then evolved according to a time-dependent Hamiltonian of the form:

$$H(t) = (1 - s(t))H_0 + s(t)H_{Ising}, \tag{3.5}$$

where $s(t)$ is a monotonic function that varies from $s(0) = 0$ to $s(T) = 1$ over the annealing time T. By the Adiabatic Quantum Theorem, if the change is slow enough, the system will remain in the ground state of $H(t)$, transitioning from the ground state of H_0 to the ground state of H_{Ising}, thus solving the optimization problem. Here we set H_0 to be $H_0 = \sum_{i=1}^{N} \hat{\sigma}_i^x$ with ground state $\psi_0 = \frac{1}{\sqrt{2^N}} \sum_{x \in \{0,1\}^N} |x\rangle$ where N is the number of qubits.

A key feature utilized in quantum annealing is quantum tunneling. Quantum tunneling allows particles to traverse energy barriers that would be insurmountable according to classical mechanics. This process enables the system to explore a wider solution space more efficiently. As the system evolves, it can tunnel through high energy states that might trap classical algorithms, thus avoiding local minima and steadily moving towards the global minimum of the optimization landscape. This is particularly beneficial in complex problems where traditional methods would get stuck in suboptimal solutions due to the rugged nature of the energy landscape. By harnessing quantum tunneling, quantum annealing enhances the ability to find the best possible solution in a more efficient manner than classical approaches.

The time complexity of quantum annealing is fundamentally influenced by the minimum energy gap of the system's Hamiltonian. The minimum energy gap is the smallest difference between the ground state energy and the first excited state energy as the system evolves from the initial Hamiltonian to the final Hamiltonian representing the problem to be solved.

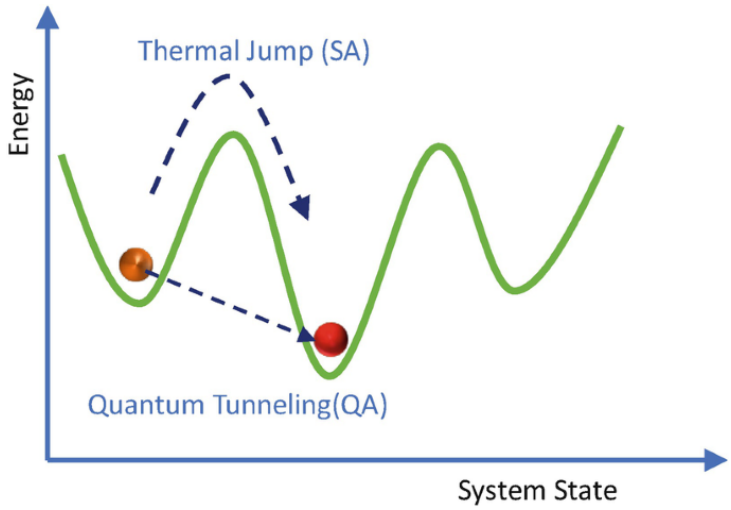


Figure 3.2: Representation of quantum tunneling.

In quantum annealing, the time required for a successful computation is inversely proportional to the square of the minimum energy gap. This means that if the minimum energy gap is very small, the system needs more time to adiabatically transition from the initial state to the ground state without jumping to higher energy states. Specifically, to ensure a high probability of finding the ground state, the annealing time T must satisfy $T = O(\frac{1}{\Delta^2_{min}})$ is the minimum energy gap. If the energy gap is large, the system can

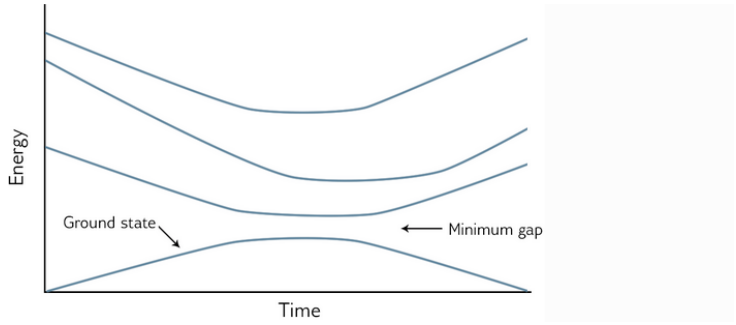


Figure 3.3: Representation of minimum gap between the ground state and the next lower energy state

evolve more quickly while maintaining a high probability of staying in the ground state. Conversely, if the gap is small, the system must evolve more slowly to avoid non-adiabatic transitions. Therefore, the efficiency of quantum annealing is heavily dependent on the characteristics of the energy landscape, and problems with larger minimum energy gaps can be solved more rapidly than those with smaller gaps. This relationship underscores

the importance of understanding and possibly manipulating the energy gap to optimize the performance of quantum annealing algorithms

3.4 Hybrid Quantum-Classical Algorithms

As quantum computation began to advance, alternative approaches to solving Quadratic Unconstrained Binary Optimization (QUBO) problems using more flexible, gate-based quantum hardware started to emerge. This shift was partly driven by the specialized hardware required for implementing Quantum Annealing. Among these new approaches, hybrid quantum-classical algorithms have become widely used, wherein a classical and a quantum computer work together to accomplish a task [7],[8]. The term "hybrid quantum-classical algorithms" refers to methods where the output from the classical computer is integral to the algorithm's operation, rather than instances where a quantum computer is merely supplemented or interfaced with a classical computer.

Variational Quantum Algorithms (VQA), inspired by the Rayleigh-Ritz variational principle, aims to find the optimal solution by preparing a quantum state that encodes the solution to the problem. Namely, if the goal of is to find the optimal solution, denoted as $\mathbf{x}^*$, VQA attempts to prepare a quantum state $|\psi^*\rangle$ such that, when measured, it yields the optimal solution $\mathbf{x}^*$ with high probability. This quantum state can be approximated using a set of parameters $\boldsymbol{\theta}$. This approximation is happening through a unitary operator $\hat{U}(\boldsymbol{\theta})$, which we call **ansatz**, such that.

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

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 [9]. In each iteration it the parameters are adjusted using a classical optimizer, such that the expectation value

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

gets minimized. If the mapping of the QUBO problem to the Ising model has been done properly, then the quantum state $|\psi(\boldsymbol{\theta})\rangle$ that cooresponds to the minimum energy

$\langle\psi(\boldsymbol{\theta})|\, H_{Ising}\, |\psi(\boldsymbol{\theta})\rangle$, would represent the desired $|\psi^*\rangle$ state. The specific form of $U(\boldsymbol{\theta})$ is a critical factor that differentiates various VQAs and can impact the quality of the solutions obtained. Choosing the optimal $U(\boldsymbol{\theta})$ depends on multiple factors, including the ease of implementation on current hardware, the ansatz's ability to generate a wide distribution of possible states (i.e., expressibility) [10], its capacity to approximate or reproduce the desired target state (i.e., reachability) [11].

The estimate of the ground state energy from a quantum device is $\langle H_{\text{Ising}}\rangle$. Classically, the expected value of a variable Y is given as $E[Y] = \sum_i \Pr(Y_i)Y_i$, where $\Pr(Y_i)$ is the probability of sampling a value Y_i from a probability distribution over all possible values of Y. Using the equivalence of C and H_{Ising}, and considering a quantum state as a probability distribution from which solutions $\vec{x}$ can be sampled, we obtain:

$$C = \sum_{i=1}^{2^{n_c}} \Pr(\mathbf{x_i})\mathbf{x_i}^\top \mathbf{Q}\mathbf{x_i} \tag{3.6}$$

Now, consider the classical cost function analogous to $\langle H_{\text{Ising}}\rangle$, with $\Pr(\mathbf{x_i})$ representing the probability of sampling a bitstring $\mathbf{x_i}$ from a probability distribution over all possible bitstrings. Equation 3.6 can be viewed as the general form of the classical QUBO cost function utilized in quantum approaches. The goal of the classical computer is to generate a probability distribution that maximizes the likelihood of sampling the optimal solution $\mathbf{x}^*$, thereby minimizing Eq 3.6

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):

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

For a cost function of the form given in Equation 3.6, the probability of sampling a bitstring $\mathbf{x}_i$ can be determined according to the Born rule: $\Pr(\mathbf{x}_i) = |a_i|^2$.

Another essential aspect of VQAs is the classical optimization strategy used to find the optimal parameters, $\boldsymbol{\theta}^*$. Classical optimizers typically employ either a gradient-free or gradient-based approach, and several optimization strategies have been specifically designed for optimizing VQAs [12],[13]. Solving the QUBO problem thus becomes a

task in non-convex optimization, where the objective is to find the optimal variational parameters so that the quantum circuit can consistently produce the solution to the QUBO problem. This approach does not eliminate the NP-hard complexity of the problem, as it has been shown that determining the optimal parameters for a variational algorithm is NP-hard.

3.5 Quantum Approximate Optimization Algorithm (QAOA)

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical algorithm designed to solve QUBO problems [14]. It mimics the quantum annealing process and leverages the Trotterization theorem to implement the time evolution of a quantum system in discrete steps. As the generic VQAs , the QAOA aims to find the optimal solution by preparing a quantum state that encodes the solution to the problem. In this case the set of parameters to approximate the optimal quantum state is $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, ...)$ and $\boldsymbol{\beta} = (\beta_1, \beta_2, ...)$ and the corresponding ansatz, $\hat{U}(\boldsymbol{\beta}, \boldsymbol{\gamma})$, is ispired from the quantum annealing time- evolution Hamiltonian along the trotterizetion theorem.

Theorem 3.5.1. Trotterization Theorem*:To approximate the continuous time evolution used in quantum annealing, QAOA employs the Trotter-Suzuki decomposition. This theorem allows us to break down the exponential of the sum of non-commuting Hamiltonians into a product of exponentials of the individual Hamiltonians.*

Now if we consider the annealing Hamiltonian

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

Then the time evolution of the quantum state would be.

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

If we apply the trotterization here we get:

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

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 :

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

Where

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

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

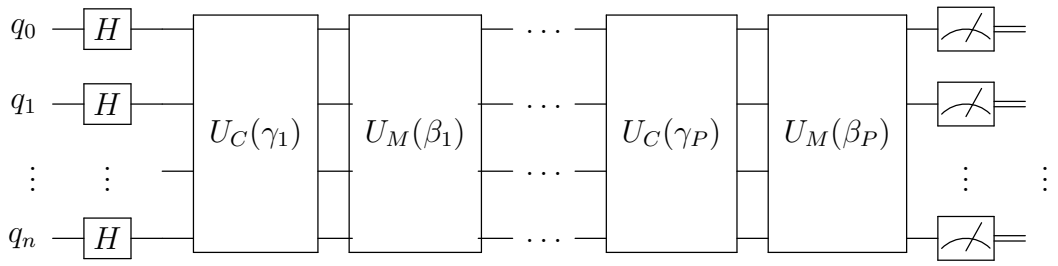


Figure 3.4: Multi Layer QAOA ansatz with alternating $U_M(\beta)$ and $U_C(\gamma)$ operators

3.5.1 QAOA Ansatz Operators as Quantum Gates

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 $e^{-i\beta_k H_B}$ 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 $Ue^{-i\gamma_k H_C}$ 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 and is defined as:

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

Thus in our case we replace the parameter θ with This gate applies a phase depending on the state of both qubits.

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

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

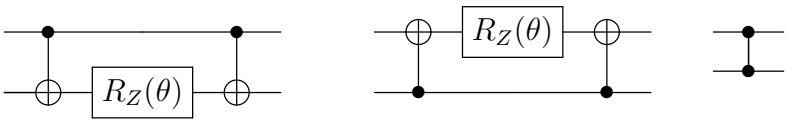


Figure 3.5: Equivalent quantum circuit decompositions of the $R_{ZZ}(\theta)$ gate

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

In our case we replace the parameter θ with

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

where J_{ij} are the interaction coefficients and h_i are the individual qubit coefficients. For the mixing Hamiltonian H_B:

$$H_B = \sum_{j=1}^{n} \hat{\sigma}_j^z$$

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

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

In our case we replace the parameter θ with

$$R_X(2\beta_p) = \exp\left(-i\beta_p\hat{\sigma}^x\right)$$

where $R_X(\theta)$ is the rotation gate around the X-axis by angle θ.

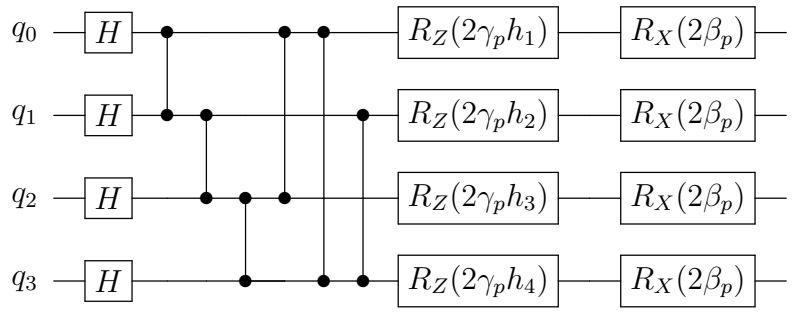


Figure 3.6: Single layer QAOA ansatz

By decomposing these Hamiltonians into quantum gates, QAOA leverages the power of quantum circuits to explore the solution space efficiently. The variational parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are tuned to optimize the final measurement outcomes, driving the system towards the optimal solution.

3.6 Hardware-Efficient Ansatz

The problem with QAOA is that its circuits are tailored to specific problem Hamiltonians and may not align well with the native gate set of the hardware. This can lead to higher gate counts and deeper circuits, which are more susceptible to noise and decoherence. The need for deeper circuits arises also from the fact that the number of layers P needs to be significantly large for the the solution to be exact

For near-term quantum computers, where gate fidelities and coherence times are limited, a more well-suited ansatz has been developped, the *hardware-efficient ansatz* is a type of parameterized quantum circuit designed to be compatible with the native gate set and connectivity of the quantum hardware [15]. This ansatz typically consists of alternating layers of single-qubit rotations and entangling gates. The single-qubit rotations are parameterized by angles, which are the variables to be optimized. The entangling gates are chosen based on the native two-qubit gates available on the hardware (e.g., CNOT, CZ).

In this circuit:

- $R_Y(\theta)$ are single-qubit rotations about the Y-axes.

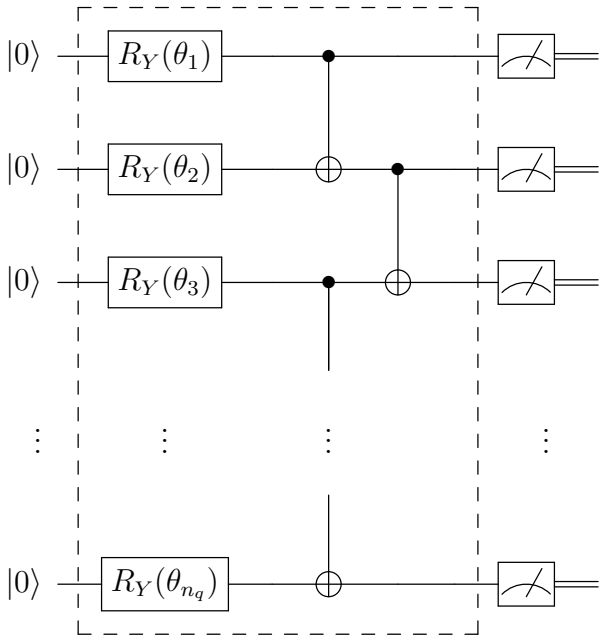


Figure 3.7: The R_Y-C_X hardware-efficient ansatz is designed for n_q qubits. The configuration enclosed by the dashed box constitutes one layer of the ansatz. This layer can be iteratively applied numerous times, contingent upon the preferred number of layers.

- The CNOT gates create entanglement between the qubits.
- The parameters θ_i are variational parameters to be optimized.

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 parameter landscape becomes increasingly flat due to the concentration of measure phenomenon. This results in gradients that are close to zero, impeding the optimization process and making it challenging to find the optimal parameters. This phenomenon is called Barren plateus [16]..

3.7 Applications of VQA

3.7.1 The Max-Cut Problem

The MAXCUT problem involves dividing the nodes of a given graph into two distinct groups in such a way that the number of edges between the two groups is maximized.

This problem has significant applications in fields such as network design, VLSI design, and statistical physics.

Problem Formulation

To solve the MAXCUT problem, we formulate an optimization problem. The objective is to maximize the cost function, which represents the number of edges between the two groups of nodes. The cost function can be mathematically expressed as:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{maximize}} \quad C(\mathbf{x}) := \sum_{\langle i,j \rangle} (x_i(1-x_j) + x_j(1-x_i)) = \sum_{\langle i,j \rangle} (x_i + x_j - 2x_i x_j) \\ &\text{subject to} \quad x_i \in \{0,1\} \end{aligned}$$

Explanation of Terms

- $\langle i,j \rangle$: Denotes the existence of an edge between nodes i and j.
- x_i: Indicates the group to which node i belongs, where $x_i \in \{0,1\}$. Here, "0" and "1" represent the two different groups.

Cost Function Details

The cost function $C(\mathbf{x})$ is designed to measure the total number of edges that cross between the two groups. Specifically:

- $x_i(1-x_j) + x_j(1-x_i)$ contributes to the cost function if nodes i and j belong to different groups.
- This term can be rewritten as $x_i + x_j - 2x_i x_j$, making the optimization formulation more tractable.

This clearly represents a QUBO problem, which we aim to solve using the previously discussed circuits.

3.7.2 Solving Max-Cut using QAOA and Hardware Efficient Ansatz

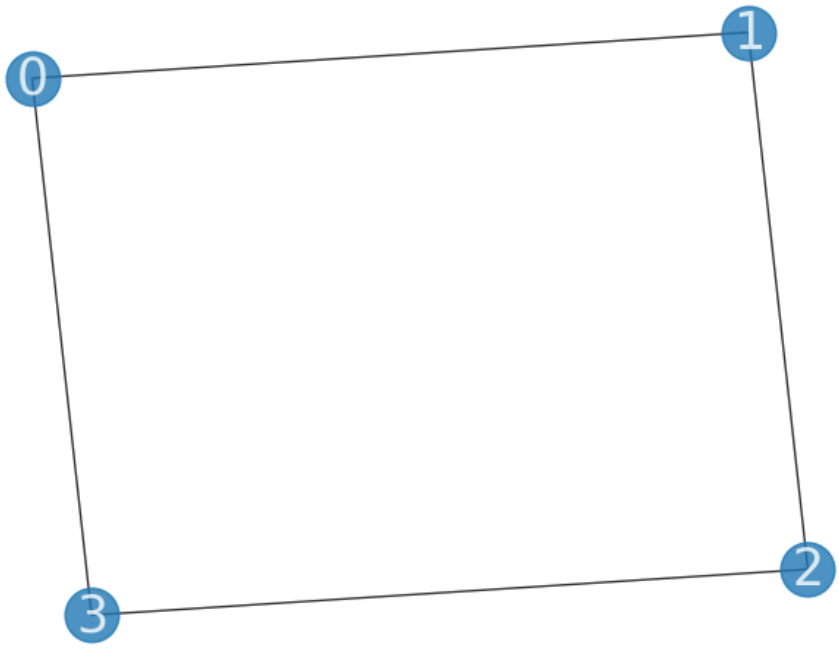


Figure 3.8: Graph for Max-Cut problem

In this work, we implement the Quantum Approximate Optimization Algorithm (QAOA) to solve the Max-Cut problem, as depicted in Fig. 3.8, using various numbers of layers P. By experimenting with different layers, we aim to analyze the algorithm's performance and efficiency, providing insights into its practical applications. The custom code developed for this purpose showcases the practical application of theoretical principles and demonstrates the capability to solve complex graph problems using quantum techniques

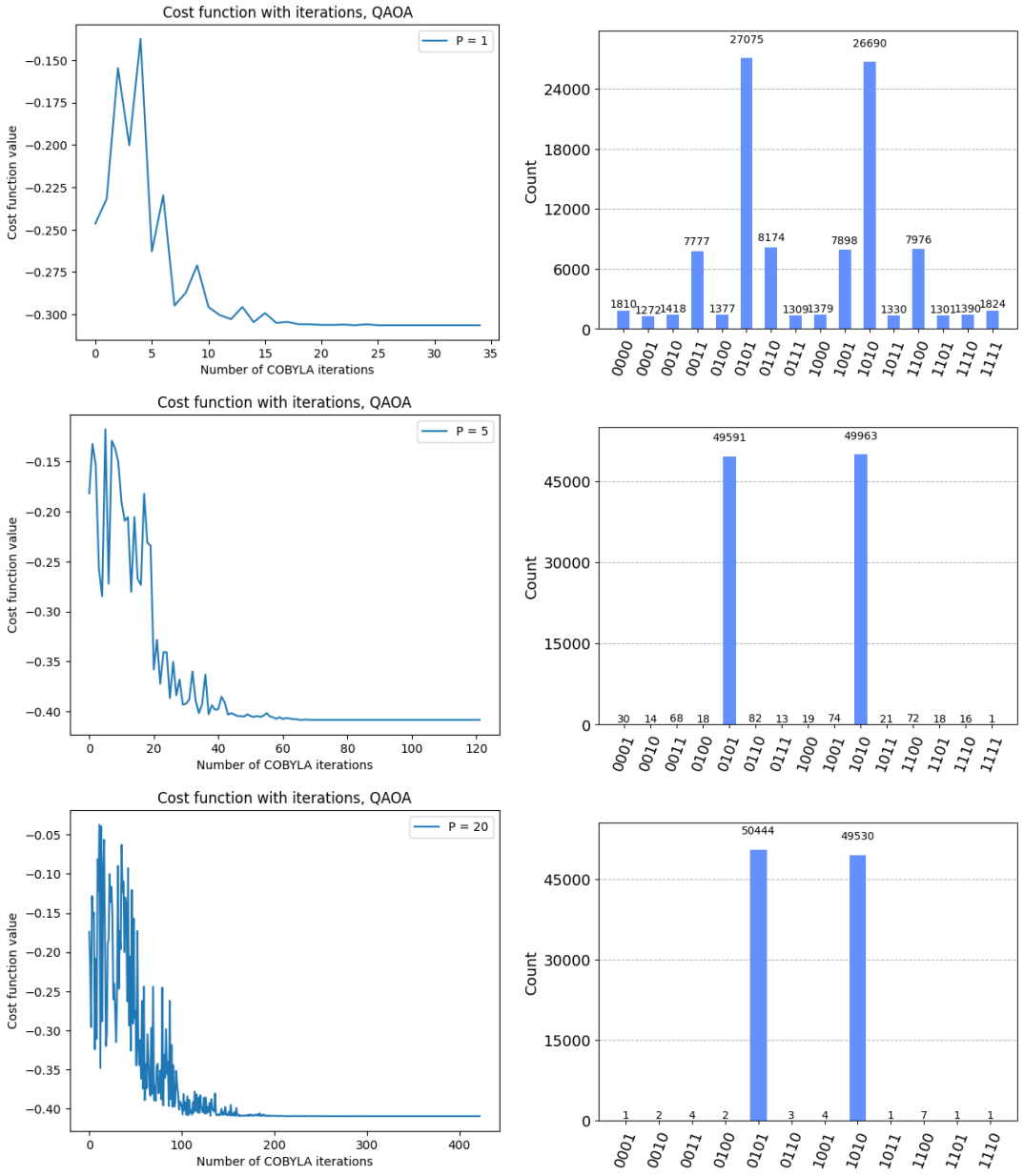


Figure 3.9: QAOA results for $P = 1, 5, 20$: Histogram of results (left) and Optimization plot (right).

We observe in 3.9 that as P increases, the solution becomes more accurate, meaning it becomes more likely to measure the two optimal bitstrings "0101" and "1010" and less

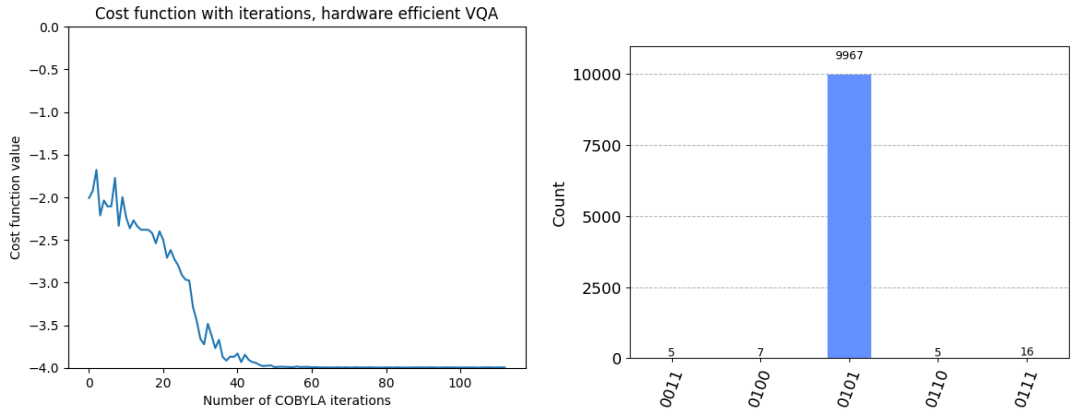


Figure 3.10: Hardware Efficient Ansatz results for one layer: Histogram of results (left) and Optimization plot (right).

likely to measure a suboptimal solution. This accuracy comes at the expense of more optimizer iterations, which increase as P increases, since there are more optimization variables to tune. For comparison, we also run the same problem using a single-layer Hardware Efficient Ansatz in 3.10.

3.8 Solving the SUBSET SUM Problem Using VQA

We are given a vector $\mathbf{a}$ and a constant S. The objective is to find a subset of the elements in $\mathbf{a}$ such that the sum of the subset equals S. To achieve this, we formulate the problem as a minimization of the cost function $C(\mathbf{x})$, where:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & C(\mathbf{x}) := \left(\mathbf{a}^T\mathbf{x} - S\right)^2 \\ \text{subject to} \quad & x_i \in \{0, 1\} \end{aligned}$$

Here, $\mathbf{x}$ is a binary vector with the same length as $\mathbf{a}$, where each element x_i can either be 0 or 1. The cost function $C(\mathbf{x})$ measures the squared difference between the weighted sum of the subset indicated by $\mathbf{x}$ and the target sum S. The goal is to minimize this cost function, effectively finding the subset of $\mathbf{a}$ that sums to S.

In our specific case, consider the vector $\mathbf{a}^T = [2, 1, 2, 1]$ and the target sum $S = 3$. The optimization problem then becomes:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & C(\mathbf{x}) := \left([2,1,2,1]\cdot\mathbf{x}-3\right)^2 \\ \text{subject to} \quad & x_i \in \{0,1\} \end{aligned}$$

This problem can be cast as a QUBO problem, which is solvable using VQAs such as the QAOA and HEA.

The implementation of these algorithms was carried out using custom-developed code. This involves defining the cost function, setting up the quantum circuit, and running the optimization to find the optimal solution $\mathbf{x}$ that minimizes $C(\mathbf{x})$.

The results for both QAOA and the Hardware Efficient Ansatz are detailed below, showcasing the effectiveness of these approaches in solving the subset sum problem:

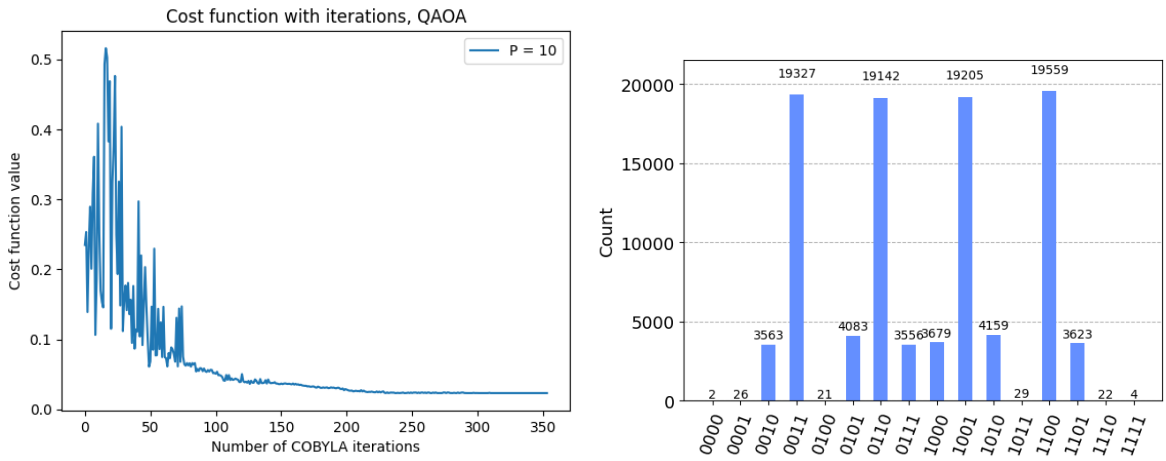


Figure 3.11: QAOA results given $\mathbf{a}^T = [2,1,2,1]$ and $S = 3$: Optimization plot (left) and Histogram of results (right).

3.9 General Observations

There are four different optimal bitstrings that solve the problem: [0011], [0110], [1001], [1100]. This means there are $2^4 - 1 = 15$ quantum states that represent equivalent solutions to the problem, encompassing every possible superposition combination of the optimal bitstrings.

VQA typically targets one of these 15 solutions, whereas QAOA tends to consistently form a superposition of all optimal bitstrings. This behavior of QAOA might be related to

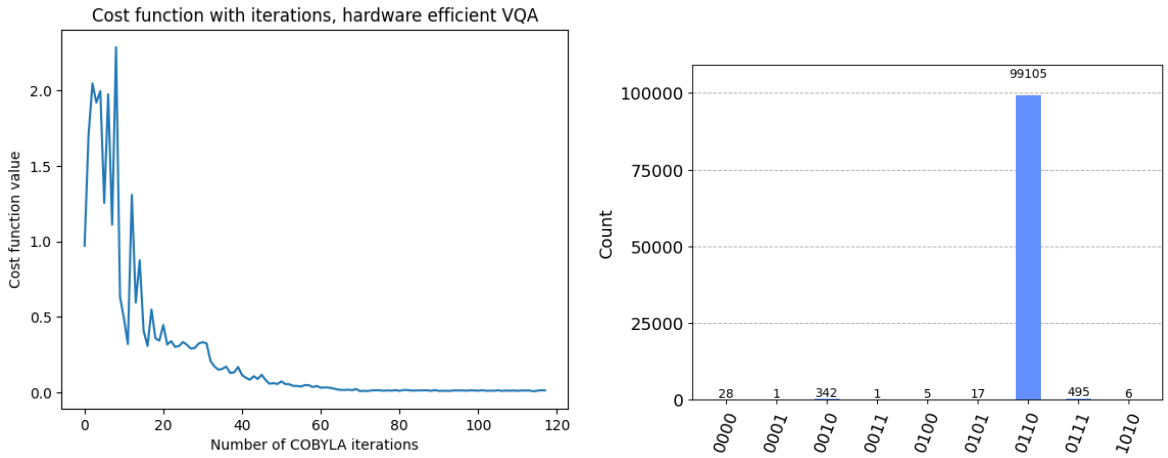


Figure 3.12: HEA results given $\mathbf{a}^T = [2, 1, 2, 1]$ and $S = 3$: Optimization plot (left) and Histogram of results (right).

its ansatz, which is inspired by adiabatic computing. In adiabatic computing, the process starts from a uniform superposition state, naturally favoring superposition solutions.

Chapter 4

Qubit-efficient Encoding Schemes for Binary Optimization Problem

This chapter summarizes the material published in

- Tan, B., Lemonde, M.A., Thanasilp, S., Tangpanitanon, J. and Angelakis,D.G., 2021. Qubit-efficient encoding schemes for binary optimisation problems. Quantum, 5, p.454.

Solving QUBO problems is one of the proposed applications of quantum computers. However, due to the current limitations of quantum hardware, the sizes of problems relevant to real-world QUBO use-cases are beyond the capabilities of today's quantum devices.

The idea behind the implementation of all the previously mentioned VQAs is based on the concept of energy minimization and leveraging the unique quantum ability to capture and evaluate multiple solutions simultaneously. This relationship to energy minimization is most accurate when each qubit represents a single classical variable, thus establishing a one-to-one mapping of classical variables to qubits, i.e., $n_c = n_q$. In the work by Tan et al. [17], this encoding is referred to as *complete encoding* or *full encoding*. It examines whether this approach can be replaced with a proposed, more qubit-efficient encoding.

This full encoding requires each possible solution of the QUBO problem $\mathbf{x_i} \in \{0,1\}^{n_c}$ to be 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:

$$|\psi(\boldsymbol{\theta})\rangle = \sum_{i=1}^{2^{n_c}} a_i(\boldsymbol{\theta}) \, |x_i\rangle \tag{4.1}$$

Here, $a_i(\boldsymbol{\theta})$ represents the amplitude of $|x_i\rangle$. By associating a classical solution $\mathbf{x}$ with a basis state $|x\rangle$, the state $|\psi(\theta)\rangle$ is capable of encoding all possible classical solutions in a linear superposition. This unique and fundamental property of quantum algorithms allows multiple classical solutions to be tested simultaneously, and this intrinsic parallelism is a strong motivator for developing quantum algorithms for classical problems.

All $a_i(\boldsymbol{\theta})$ in Eq. (4.1) can, in principle, be independent (subject to the normalization condition). Consequently, this quantum state can capture all possible correlations between the classical variables and exhibits expressive power beyond classical computation. The goal from here is to efficiently navigate the exponentially large Hilbert space and reach the basis state(s) that represent the exact or approximate solution(s) to the QUBO problem.

4.1 Qubit efficient encoding schemes

In this section, we will explore the novel encoding scheme developed in the cited work, which demonstrates a significant reduction in the number of qubits needed for binary optimization problems. The core idea arises from the question: what if, 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? This approach would require fewer qubits for a smaller set of variables, and we could leverage quantum superposition to simultaneously evaluate all the subsets that compose the full set of classical variables.

The realization of this idea begins by dividing our total number of binary variables, denoted as n_c, into R subgroups. Generally, these subgroups can consist of any number of binary variables, and a single binary variable can be part of multiple subgroups. However, for the scope of this work, considering disjoint subgroups with the same number of binary variables, denoted as n_a, should be sufficient to convey the essence of the encoding scheme. Figure 4.1 provides an overview of various methods to encode different subgroups of qubits using different numbers of ancilla qubits n_a, as well as how the addresses for these

subgroups are mapped to the basis states of register qubits.

The number of subgroups, R, depends on how the binary variables are grouped. For instance, if we divide the binary variables such that each subgroup contains an equal number of binary variables, and each binary variable appears in only one subgroup, the number of subgroups is given by $R = \frac{n_c}{n_a}$.

The encoding scheme operates by using ancilla qubits to represent the values of binary variables, exactly as in the full encoding but within the smaller subgroups. Since each subgroup consists of n_a variables, n_a ancilla qubits are required. Having defined the ancilla part of the qubit, we need to identify which subgroup these ancilla qubits correspond to. This is where register qubits come into play, representing the addresses of the R subgroups with each of their basis states. This is where the qubit reduction occurs and is the most vital point of this encoding scheme. To represent addresses using qubits, we do not need a one-to-one mapping; we simply need the binary encoding of each address. Therefore, for R subgroups, the register qubits require $n_r = \lceil \log_2(R) \rceil$ qubits, where $\lceil \cdot \rceil$ denotes the ceiling function. Thus, the total number of qubits needed is $n_q = n_a + n_r$. The quantum state representing our classical solutions is expressed as:

$$|\psi(\boldsymbol{\theta})\rangle = \sum_{r}^{R} \beta(\boldsymbol{\theta}) \left(\sum_{i=1}^{2^{n_a}} a_r^i(\boldsymbol{\theta}) \, |x_a^i\rangle \right) \otimes |r\rangle \tag{4.2}$$

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.

The full encoding, 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, can still be viewed as a limiting case of this scheme where $n_a = n_c$, meaning all binary variables are grouped into a single subgroup. In this scenario, $R = 1$, and we can disregard the register qubits. The number of qubits required is $n_q = n_a$, resulting in the conventional mapping where each binary variable is represented by a single qubit in the quantum device. The quantum state representing the full encoding is given in Eq. 4.1. For a QUBO problem, the optimal quantum state can be written as:

$$|\psi_c(\boldsymbol{\theta}^*)\rangle = \sum_{i}^{n_{\text{degen}}} a_{x_i}(\boldsymbol{\theta}^*) |x_i\rangle$$

, where the sum over i is taken over the number of degenerate solutions to the QUBO problem.

Another limiting 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 is referred to as the *minimal encoding*. The limitation of this compact mapping is its capacity to encode only the distribution functions of statistically independent classical variables. This is expected, as the quantum state uses only n_c coefficients to encode a probability distribution over 2^{n_c} solutions. Despite these limitations, we will examine this limiting case closely as it captures the core elements of the general encoding strategy.

In contrast, by increasing the number of ancilla qubits, one can capture n_a-body classical correlations between the binary variables. These classical correlations determine the value of one binary variable i based on the value of another binary variable j. Extending the number of ancillas to the full encoding allows the quantum state to capture the complete n_c-body classical correlation within the problem. A more detailed discussion about the limitations of the efficient encoding will take place later in this chapter.

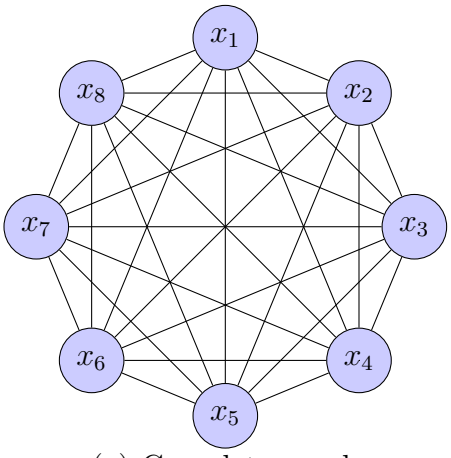
4.2 Cost Function

As discussed in the previous chapter, the general cost function of the QUBO problem is derived from Equation 4.3, which is restated here for clarity:

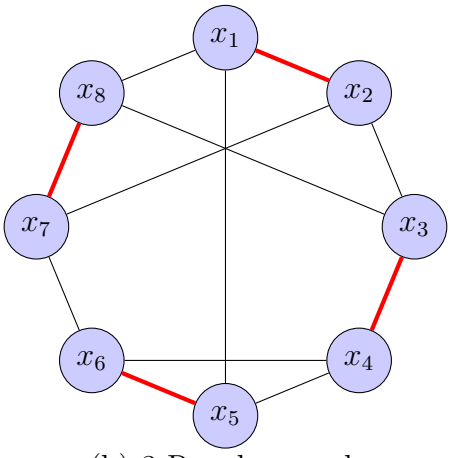
$$C = \sum_{i=1}^{2^{n_c}} \Pr(\mathbf{x_i}) \mathbf{x_i}^\top \mathbf{Q} \mathbf{x_i} \tag{4.3}$$

The calculation of this cost function is straightforward: we have structured the problem such that a classical solution $\mathbf{x_i}$ corresponds to a basis state $|x_i\rangle$ as described in Equation 4.1. Upon executing the quantum circuit (ansatz), we determine all the bitstrings $\mathbf{x_i}$ by iterating through all the basis states $|x_i\rangle$ of the quantum state in Equation 4.1, and we calculate their corresponding probabilities using the formula

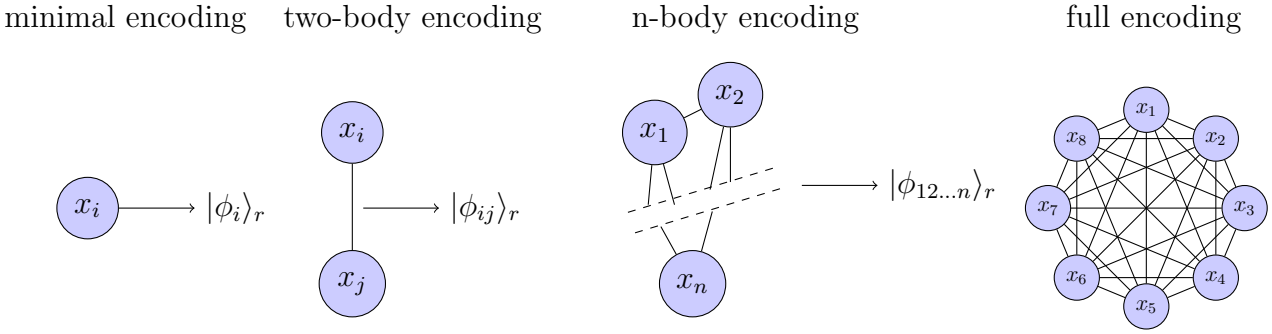
$$\Pr(\mathbf{x_i}) = |a_i(\boldsymbol{\theta})|^2$$



(a) Complete graph



(b) 3-Regular graph



(c) encoding schemes

Figure 4.1: Schematic representation of the encoding schemes. (a) Complete graph representing a dense QUBO matrix A. (b) A 3-Regular graph. The set of red edges is an example of perfect matching where each vertex is connected to only a single edge. (c) Encoding schemes where n_a-body correlation are encoded with n_a increasing from top to bottom. In the minimal encoding, each of the 2^{n_r} basis states $|\phi_i\rangle$ 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 complete encoding, each basis state represents an entire graph

However, for the efficient encoding scheme presented here, the basis states of the quantum state in Equation 4.2 (i.e., $|x_a^i\rangle\,|r\rangle$) do not directly correspond to the full bitstring $\mathbf{x_i}$. Consequently, the expression in Equation 4.3 is not suitable for the encoded scheme, and it must be reformulated in a more general manner that bypasses the need for explicit bitstrings.

$$C_{\boldsymbol{\theta}} = \sum_{\mathbf{x}} Pr_{\boldsymbol{\theta}}(\mathbf{x})\mathbf{x}^{\top}\mathbf{Q}\mathbf{x} \tag{4.4}$$

$$= \sum_{\mathbf{x}} Pr_{\boldsymbol{\theta}}(\mathbf{x}) \sum_{i,j=1}^{n_c} x_i Q_{ij} x_j \tag{4.5}$$

$$= \sum_{i,j=1}^{n_c} Q_{ij} \sum_{\{\mathbf{x}|x_i=x_j=1\}} Pr_{\boldsymbol{\theta}}(\mathbf{x}) \tag{4.6}$$

$$= \sum_{i,j=1}^{n_c} Q_{ij} P_{i,j}^{1,1}(\boldsymbol{\theta})(1-\delta_{ij}) + \sum_{i=1}^{n_c} Q_{ii} P_i^1(\boldsymbol{\theta}) \tag{4.7}$$

where $\mathbf{x}$ represents the ensemble of all 2^{n_c} possible solutions, while $\{\mathbf{x} \mid x_i = x_j = 1\}$ denotes the subset of 2^{n_c-2} solutions where the i-th and j-th variables in $\mathbf{x}$ are both $x_i = x_j = 1$. To transition from Eq. (4.5) to Eq. (4.6), we utilized the fact that only variables with values equal to 1 contribute to the cost function. In Eq. (4.7), $P_{i,j}^{1,1}$ represents the probability that both x_i and x_j are 1, and P_i^1 represents the probability that x_i is 1. δ_{ij} is the Kronecker delta. This cost function is generic and applicable to the various encoding schemes. The primary differences between the encoding schemes lie in the method of calculating the probabilities $P_{i,j}^{1,1}$ and P_i^1, as different encoding schemes result in different quantum states used to encode the probability distribution.

4.3 The Minimal Encoding

In this section, we delve deeper into the minimal encoding scheme. This approach represents the extreme case where $n_a = 1$, so each subgroup contains only one binary variable. Consequently, there are $R = n_c$ subgroups, and it requires $n_q = 1 + \log_2(n_c)$ qubits to represent n_c binary variables. For this minimal encoding, the quantum state in Eq. 4.2 is expressed as follows:

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

where $\{|\phi_i\rangle_r\}$ and $\{|0\rangle_a, |1\rangle_a\}$ are the computational basis states of the register and ancilla qubits, respectively. The idea is to create a one-to-one correspondence between each of the n_c classical variables x_i in $\mathbf{x}$ and a unique basis state $|\phi_i\rangle_r$, as shown in Fig. 4.1c.

The probability distribution captured by the minimal encoding quantum state is given by:

$$\Pr(\mathbf{x}) = \prod_{i=1}^{n_c} \Pr(x_i) \tag{4.8}$$

This describes a probability distribution over n_c independent variables, underscoring the minimal encoding scheme's limitation in capturing classical correlations between the binary variables for a fixed $\boldsymbol{\theta}$. The probability that the i-th classical variable takes the value 1 or 0 is given by $\Pr(x_i = 1) = |b_i(\boldsymbol{\theta})|^2$ and $\Pr(x_i = 0) = 1 - |b_i(\boldsymbol{\theta})|^2 = |a_i(\boldsymbol{\theta})|^2$, respectively. The coefficients $\beta_i(\boldsymbol{\theta})$ represent the likelihood of measuring each register state $|\phi_i\rangle$, and therefore the corresponding state of the ancilla qubit, with this probability given by $|\beta_i(\boldsymbol{\theta})|^2$.

For example, encoding the probability distribution over all solutions $\mathbf{x}$ of dimensions $n_c = 4$ requires $n_r = 2$. One can then define the mapping as $|\phi_1\rangle_r \equiv |00\rangle_r$, $|\phi_2\rangle_r \equiv |01\rangle_r$, $|\phi_3\rangle_r \equiv |10\rangle_r$, and $|\phi_4\rangle_r \equiv |11\rangle_r$. Using this mapping, the quantum state representing the unit probability of sampling $\mathbf{x} = (0, 0, 1, 1)$ would be:

$$|\psi\rangle_{n_a=1} = \frac{1}{2}\left(|0\rangle_a|00\rangle_r + |0\rangle_a|01\rangle_r + |1\rangle_a|10\rangle_r + |1\rangle_a|11\rangle_r\right)$$

It is important to note that at least four measurements are required to retrieve the bitstring $\mathbf{x}$ from the quantum state $|\psi\rangle_{n_a=1}$, to ensure that every register is measured. In contrast, with full encoding, only one measurement would have been sufficient.

4.3.1 Cost Function of minimal encoding

In the minimal encoding scheme we can not capture correlation between different classical bianry variables since they are all part of a differnent ancilla group. Therefore we have to consider them independent. The joint probability of two diffenent variables to be 1 is

$$P^{1,1}_{i,j}(\boldsymbol{\theta}) = Pr(x_i = 1)Pr(x_j = 1) \quad \text{if } i \neq j \tag{4.9}$$

$$P^{1}_{i}(\boldsymbol{\theta}) = Pr(x_i = 1) \quad \text{if } i = j \tag{4.10}$$

In terms of the quantum state amplitudes, this reads:

$$P^{1,1}_{i,j}(\boldsymbol{\theta}) = |b_i(\boldsymbol{\theta})|^2 |b_j(\boldsymbol{\theta})|^2 \quad \text{if } i \neq j \tag{4.11}$$

$$P^{1}_{i}(\boldsymbol{\theta}) = |b_i(\boldsymbol{\theta})|^2 \quad \text{if } i = j \tag{4.12}$$

Substituting these results into Eq.(4.7) and re-expressing them in terms of projectors, one gets

$$C_{\boldsymbol{\theta}} = \sum_{i,j=1}^{n_c} Q_{ij} P^{1,1}_{i,j}(\boldsymbol{\theta})(1-\delta_{ij}) + \sum_{i=1}^{n_c} Q_{ii} P^{1}_{i}(\boldsymbol{\theta}) \tag{4.13}$$

$$= \sum_{i,j=1}^{n_c} Q_{ij} |b_i(\boldsymbol{\theta})|^2 |b_j(\boldsymbol{\theta})|^2 (1-\delta_{ij}) + \sum_{i=1}^{n_c} Q_{ii} |b_i(\boldsymbol{\theta})|^2 \tag{4.14}$$

4.4 Two-body Correlation

In this section, we will discuss the two-body correlation encoding, which represents the case where $n_a = 2$, so each subgroup contains two binary variables. In the general case where a variable can be contained in more than one pair,

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

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 in $\mathbf{x}$ as shown in Fig. 4.1 c.

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 in Eq. 4.2 is expressed as follows:

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

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:

$$\Pr(x_i = 1) = \begin{cases} |c_r(\boldsymbol{\theta})|^2 + |d_r(\boldsymbol{\theta})|^2 & \text{if } i \text{ is the first bit of register } r \\ |b_r(\boldsymbol{\theta})|^2 + |d_r(\boldsymbol{\theta})|^2 & \text{if } i \text{ is the second bit of register } r \end{cases} \tag{4.15}$$

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

$$P^{1,1}_{i,j}(\boldsymbol{\theta}) = \begin{cases} \Pr(x_i = 1)\Pr(x_j = 1) & \text{if } i,j \text{ are in different ancilla groups} \\ |d_r(\boldsymbol{\theta})|^2 & \text{if } i,j \text{ are in the same ancilla group} \end{cases} \tag{4.16}$$

$$P^1_i(\boldsymbol{\theta}) = \Pr(x_i = 1) \tag{4.17}$$

4.5 Limitation of the encoding scheme

We have discussed the presented encoding scheme's inability to capture correlations between variables that do not fall within the same ancilla group, but what does that exactly mean? The original full encoding ansatz can test all possible solutions and assign an amplitude coefficient that reflects the quality of each solution. From these coefficients, a probability distribution emerges, describing the likelihood of measuring each solution. This probability distribution represents the joint probability of every classical variable.

However, the quantum circuit in the presented encoding scheme cannot capture the joint probability of classical variables that are not in the same ancilla subgroup. Consequently, it loses the ability to simultaneously evaluate different solutions if they include dependencies between disjoint ancilla subgroups. Let's understand this with an example. Suppose we have a QUBO problem that has optimal solutions $\mathbf{x}^*_\mathbf{a} = (1,1,1,1)$ and $\mathbf{x}^*_\mathbf{b} = (0,0,0,0)$. The full encoding would be able to prepare the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0000\rangle + \frac{1}{\sqrt{2}}|1111\rangle$$

This suggests that the bitstrings $(0,0,0,0)$ and $(1,1,1,1)$ are more likely to be sampled. However, if we attempt to solve this with, let's say, a two-body correlation encoding, it would be impossible to prepare such a state that highlights both solutions because we would not be able to capture the correlation that the second subgroup should be $(0,0)$

if the first subgroup is $(0, 0)$ or that the second subgroup should be $(1, 1)$ if the first subgroup is $(1, 1)$.

Trying to capture both solutions would result in a state like

$$|\psi\rangle = \frac{1}{2}|00\rangle_a|0\rangle_r + \frac{1}{2}|11\rangle_a|0\rangle_r + \frac{1}{2}|00\rangle_a|1\rangle_r + \frac{1}{2}|11\rangle_a|1\rangle_r$$

In this state, we sample the correct solutions $(0, 0, 0, 0)$ and $(1, 1, 1, 1)$ with probability $\frac{1}{4}$ each, but we also sample the incorrect solutions $(1, 1, 0, 0)$ and $(0, 0, 1, 1)$ with the same probability. To avoid sampling incorrect solutions, we approximate the correlations between different ancillas by considering them independent, i.e., $\Pr(x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1) = \Pr(x_1 = 1, x_2 = 1)\Pr(x_3 = 1, x_4 = 1)$. However, in doing so, we lose the ability to capture both solutions simultaneously. The circuit in this case would prepare a state like

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle_a|0\rangle_r + \frac{1}{\sqrt{2}}|00\rangle_a|1\rangle_r$$

or

$$|\psi\rangle = \frac{1}{\sqrt{2}}|11\rangle_a|0\rangle_r + \frac{1}{\sqrt{2}}|11\rangle_a|1\rangle_r$$

Another limitation of the encoding is that to properly decode the encoded bitstring, we need information regarding the ancilla states for every register. In practice, it is observed that some registers may not be visible in the resulting distribution. This could be due to insufficient circuit measurements or the inability of the produced quantum state to include some registers in the linear superposition. In such cases, we assume that all ancilla combinations corresponding to the unmeasured register have equal probabilities, which may lead to inaccurate results.

4.6 Testing the Efficient Encoding Scheme

In this section, we will conduct proof-of-concept experiments to validate the functionality of the encoding scheme and examine its behavior for different lengths of ancilla subgroups. As mentioned, we will exclusively examine schemes with ancillas of the same size that are disjoint from each other. The problem chosen for these experiments is the Subset Sum

problem, as we are interested in seeing how the scheme performs with a dense QUBO matrix (rather than the sparse Max-Cut matrix). The Subset Sum problem also allows for easy modification of the vector $\mathbf{a}$ and sum S without the need to create a new graph, as required in the Max-Cut case.

We will begin with a simple proof-of-concept experiment that compares the minimal encoding to the full encoding and the QAOA. The circuit implementing the encoding scheme is a Hardware Efficient Ansatz, so we will refer to it as "HEA" from now on. For the encoding scheme, we will refer to it as "Compressed HEA". In the following experiment, we will implement a simple two-layer Hardware Efficient Ansatz for the full encoding and the minimal encoding, as well as a 10-layer QAOA, for a small Subset Sum example where $\mathbf{a} = (\mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{1})$ and $S = 4$. This corresponds to 4 classical variables, thus 4 qubits for HEA and QAOA, and 3 qubits for Compressed HEA:

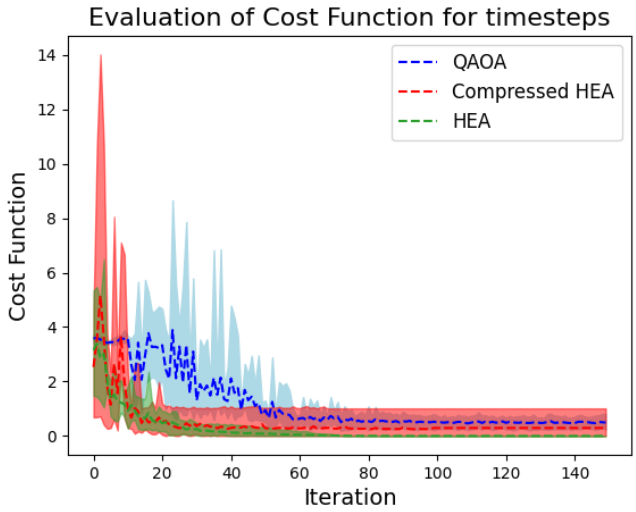


Figure 4.2: Performance comparison of Hardware Efficient Ansatz with full encoding, minimal encoding, and QAOA in a Subset Sum problem with 4 classical variables. The Hardware Efficient Ansatz uses two layers, whereas the QAOA uses ten. In all cases, the circuit is measured 10,000 times and 5 different experiments were conducted, each starting the optimization from a different initial point. The dashed line indicates the mean of all experiments.

We observe in 4.2 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.

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. For these experiments, we developed a versatile encoding algorithm capable of generating all possible encoding schemes and facilitating easy switching between schemes based on user input. This is achieved by parameterizing the ancilla length as a simple function argument. The flexibility of this algorithm allows for efficient experimentation with various encoding schemes, making it a valuable tool for our research.
Using this code, we perform six distinct experiments. In the first two experiments, we compare the performance of full encoding with three different encoding schemes where the ancilla length n_a is set to 1, 2, and 5. These comparisons are conducted for two Subset Sum problems, each involving a different number of classical variables: 10 and 20. The goal is to assess how varying the ancilla length influences the performance of the quantum algorithm in terms of both convergence rate and solution accuracy.

The experimental setup involves monitoring the evolution of the cost function across optimizer iterations, which is plotted on the left side of each figure. These plots help us understand how quickly the algorithm converges towards an optimal solution under different encoding schemes. Additionally, we analyze the distribution of solutions after convergence, which is shown on the right side of each figure. This analysis provides insights into the accuracy and reliability of the solutions produced by each encoding scheme.

.

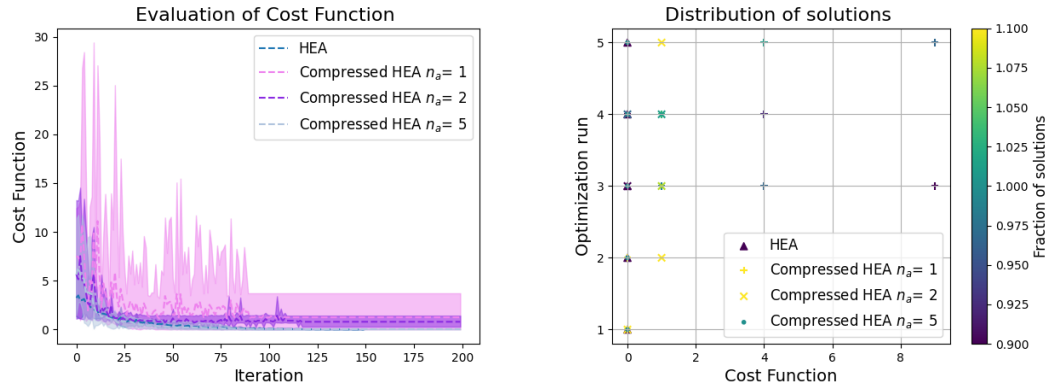


Figure 4.3: Comparison of different encoding schemes for the Subset Sum problem with $\mathbf{n_c = 10}$. This corresponds to **10 qubits for full encoding, 6 qubits for the scheme with $\mathbf{n_a = 5}$, 5 qubits for the scheme with $\mathbf{n_a = 2}$**, and **5 qubits for the scheme with $\mathbf{n_a = 1}$**.

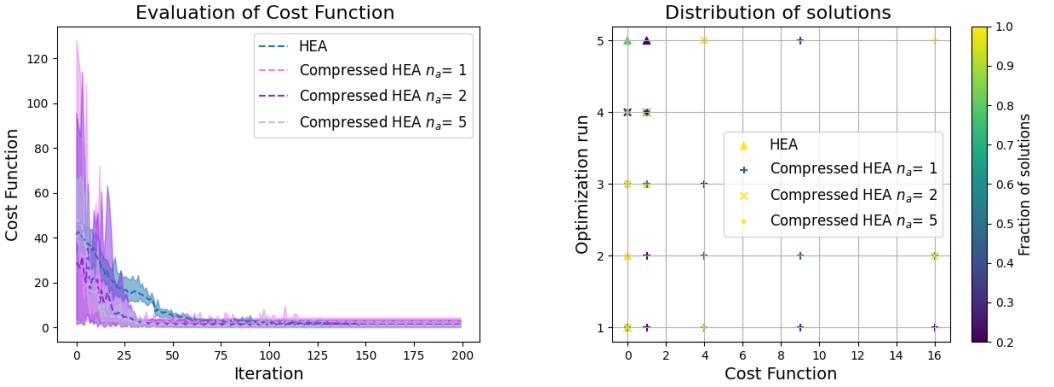


Figure 4.4: Comparison of different encoding schemes for the Subset Sum problem with $\mathbf{n_c = 20}$. This corresponds to **20 qubits for full encoding, 7 qubits for the scheme with $\mathbf{n_a = 5}$, 6 qubits for the scheme with $\mathbf{n_a = 2}$**, and **6 qubits for the scheme with $\mathbf{n_a = 1}$**.

Building on our initial experiments, we proceed to address four different Subset Sum problems of increasing sizes, specifically $n_c = 50, 100, 500, 1000$ classical variables. In these larger-scale problems, we replace the full encoding with a more efficient scheme where the ancilla length n_a is set to 10. This decision is based on our observations that such problem sizes surpass the practical capabilities of standard full encoding implemen-

tations.

The transition to an encoding scheme with $n_a = 10$ is crucial for managing the computational complexity and ensuring the feasibility of the experiments. By reducing the ancilla length, we aim to maintain the accuracy and efficiency of the quantum optimization process while handling significantly larger problem instances. This approach allows us to push the boundaries of current quantum hardware capabilities and explore the potential of qubit-efficient encoding in solving more complex and larger-scale problems.

The findings from these experiments are expected to provide valuable insights into the scalability of qubit-efficient encoding schemes. By demonstrating the effectiveness of the encoding scheme for large-scale Subset Sum problems, we highlight the potential for quantum algorithms to tackle real-world optimization challenges that are currently beyond the reach of classical and standard quantum approaches. This work lays the foundation for future research in quantum optimization, paving the way for more advanced and scalable quantum solutions.

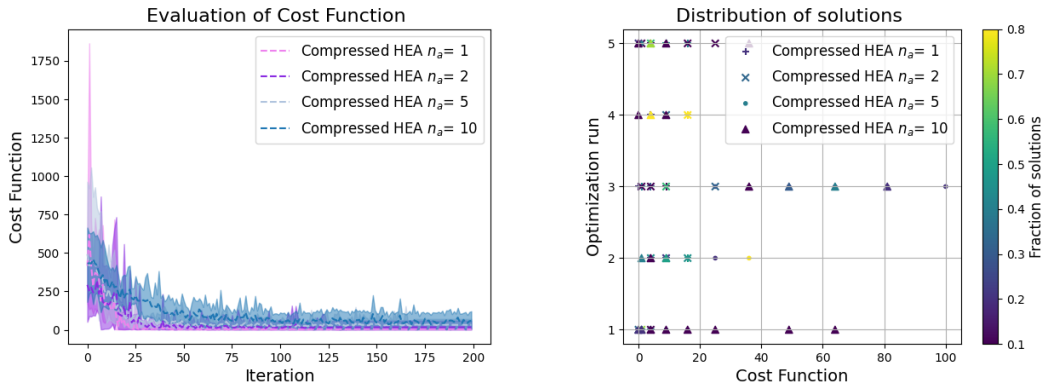


Figure 4.5: Comparison of different encoding schemes for the Subset Sum problem with $\mathbf{n_c = 50}$. This corresponds to **13 qubits for for the scheme with $\mathbf{n_a = 10}$, 9 qubits for the scheme with $\mathbf{n_a = 5}$, 7 qubits for the scheme with $\mathbf{n_a = 2}$**, and **7 qubits for the scheme with $\mathbf{n_a = 1}$**.

.

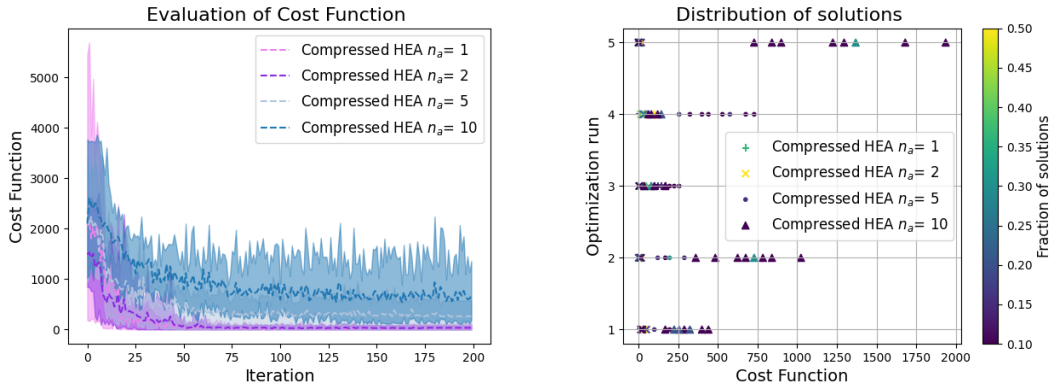


Figure 4.6: Comparison of different encoding schemes for the Subset Sum problem with $\mathbf{n_c = 100}$. This corresponds to **14 qubits for for the scheme with $\mathbf{n_a = 10}$, 10 qubits for the scheme with $\mathbf{n_a = 5}$, 8 qubits for the scheme with $\mathbf{n_a = 2}$**, and **8 qubits for the scheme with $\mathbf{n_a = 1}$**.

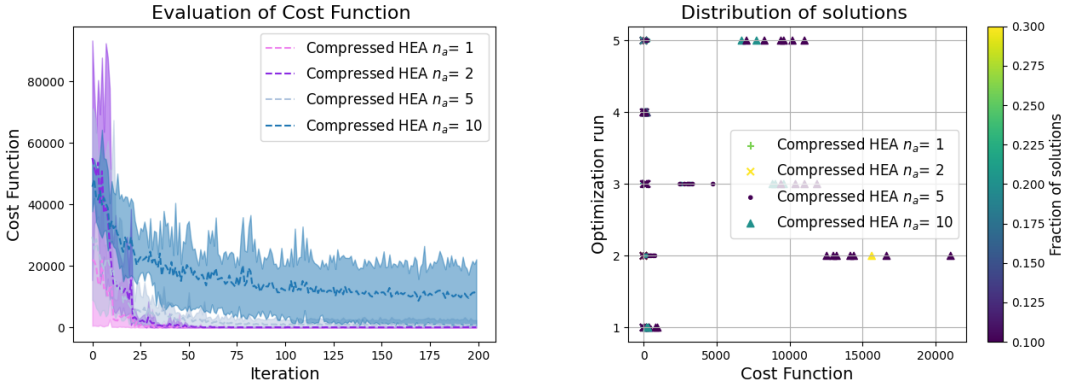


Figure 4.7: Comparison of different encoding schemes for the Subset Sum problem with $\mathbf{n_c = 500}$. This corresponds to **16 qubits for for the scheme with $\mathbf{n_a = 10}$, 12 qubits for the scheme with $\mathbf{n_a = 5}$, 10 qubits for the scheme with $\mathbf{n_a = 2}$**, and **10 qubits for the scheme with $\mathbf{n_a = 1}$**.

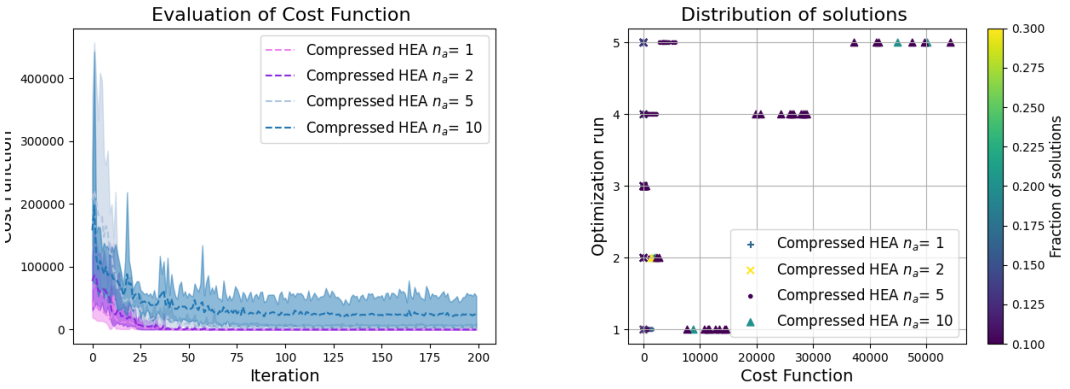


Figure 4.8: Comparison of different encoding schemes for the Subset Sum problem with $\mathbf{n_c = 1000}$. This corresponds to **17 qubits for for the scheme with $\mathbf{n_a = 10}$, 12 qubits for the scheme with $\mathbf{n_a = 5}$, 11 qubits for the scheme with $\mathbf{n_a = 2}$**, and **11 qubits for the scheme with $\mathbf{n_a = 1}$**.

In the figures above, we observe that as the problem size increases, the encoding schemes with smaller ancilla lengths (and therefore a smaller number of qubits) converge faster and more accurately compared to those with larger ancilla lengths. This demonstrates that the efficient encoding scheme enables us to tackle problem sizes that would be infeasible with traditional VQAs. However, the observation that smaller ancilla sizes ($n_a = 1, 2$) outperform larger ancilla sizes ($n_a = 5, 10$) is not straightforwardly intuitive and is complex to explain.

It also contradicts the perception that higher ancilla sizes should produce more accurate solutions due to their ability to capture more correlations. We suspect this result may be attributed to the increase in Hilbert space with larger ancilla sizes, along with the possibility of insufficiently measured registers, which have a more significant impact on these schemes. For instance, in the minimal encoding, if a register is not measured, there is still a good chance (50% probability) of randomly assigning the correct value, whereas in the $n_a = 10$ scheme, this probability is $\frac{1}{2^{10}}$.

Although $n_a = 10$ seems to be outperformed by smaller ancilla size encodings, we do not believe that increasing n_a would always result in lower performance. There is likely a balance to be reached, and results may improve as the number of ancilla qubits approaches that of the full encoding. For example, in the case of $n_c = 1000$, a scheme with $n_a = 200$ may yield better results than minimal encoding, but we do not have the

resources to test this in the scope of this thesis.

Chapter 5

Applications of Encoding Schemes for Industry Optimization

5.1 Smart Scheduling of EV Charging Problem

The Smart Scheduling of EV Charging Problem is part of a broader category of scheduling issues, a well-studied and universally relevant set of industrial challenges. These problems aim to optimize the allocation of resources over a specific period, focusing on achieving maximum efficiency. The Smart EV Charging problem specifically focuses on optimally allocating energy to EVs currently connected within a city's charging infrastructure. It involves satisfying each EV's distinct energy and charging requirements without surpassing the network's power capacity. Moreover, the ideal solution should minimize total energy expenditure while ensuring every vehicle's needs are met, demonstrating an efficient and sustainable approach to managing urban EV charging networks. Several classical algorithms exist for addressing the problem, including first-come-first-serve, least laxity first, earliest deadline first, and round robin. Among these, Model Predictive Control (MPC) is often regarded as the state of the art approach [18].A more detailed description of the problem is as follows:

Consider a set of electric vehicles (EVs) and a set of charging ports. Each EV, denoted by i, is characterized by its own energy demand e_i, its required charging duration τ_i^{end}, and its arrival time at the charging station. The charging network provides a constant voltage

at each charging port; however, the current supplied to each EV can only take discrete values from the set {8, 16, 32, 48, 64} amperes. The charging network also imposes a limit on the total energy supply it can provide at a single instance. Therefore, the total energy provided to the EVs should not surpass this limit. Let $r_i(t)$ represent the charging current received by EV i at time t.

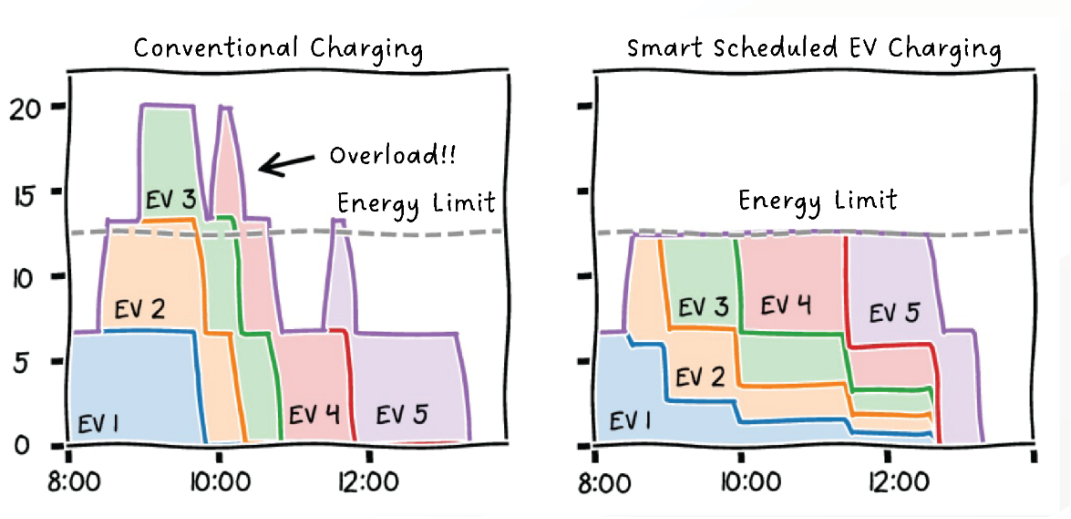


Figure 5.1: A picture that highlights the goal of the algorithm, which is to avoid surpassing the power limit of the network. This is achieved by distributing the charging energy for all the EVs throughout the horizon, so they receive lower energy for a longer duration, yet still manage to satisfy their desired charging time.

The objective is to determine the appropriate charging current $r_i(t)$ for each EV i at every time instance t, ensuring that the aforementioned conditions are satisfied

Adaptive Scheduling Algorithm

The preceding problem is approached with an algorithm based on (MPC), namely the Adaptive Scheduling Algorithm (ASA), which was initially deployed at the Caltech campus [19]. The algorithm functions as follows: It considers the set of active EVs (those currently being charged), along with their remaining energy requirements and the remaining duration of their charging sessions. Given this information, the algorithm constructs a charging schedule over a time horizon T. This schedule specifies the current that each active EV should receive at every time step, beginning from the present moment and

concluding at the end of the horizon. What distinguishes this scheduling algorithm from others, such as first-come-first-serve, earliest deadline first, etc., is its ability to consider future time steps and make decisions for current charging currents based on predictions of future conditions. To achieve this, we discretize time into timesteps t_k and introduce an optimization problem in which we define the optimization variables $\mathbf{r}$ as follows:

$$\mathbf{r} = \begin{bmatrix} r_1(t_1) & \dots & r_1(t_n) \\ \vdots & \ddots & \vdots \\ r_N(t_1) & \dots & r_N(t_n) \end{bmatrix}$$

Here, N represents the number of active EVs, and t_n is the last timestep in the horizon. The matrix is what will eventually be the charging schedule. The output of this matrix should satisfy the conflicting objectives of meeting the energy demands of every EV while also keeping the total network's energy consumption as low as possible.

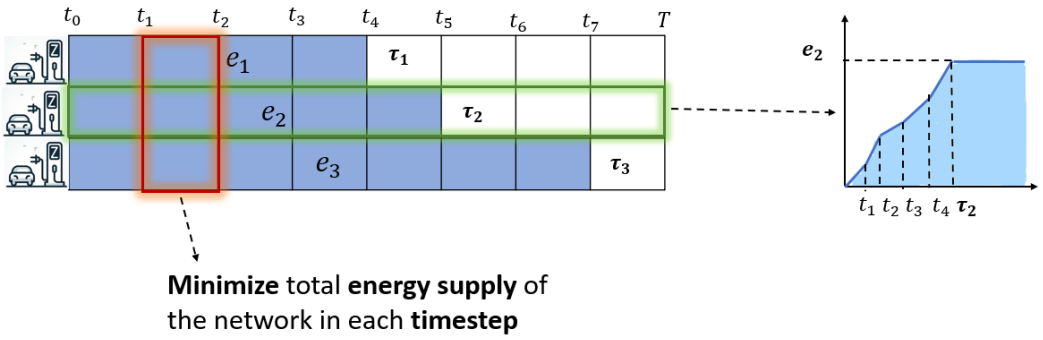


Figure 5.2: Figure illustrating the dual role of the charging schedule: to both meet the energy demand of each EV while minimizing the network's total energy consumption.

The optimization problem will look like this:

$$\begin{aligned} \underset{\mathbf{r}}{\text{minimize}} \quad & U(\mathbf{r}) \\ \text{subject to} \quad & r_i(t_k) \in \{0, 8, 16, 32, 48, 64\} \quad \forall i, k \end{aligned}$$

The cost function $U(\mathbf{r})$ varies depending on the use cases, which will be discussed later in this report. This optimization process is what will be formulated as QUBO problem. The optimization process will be repeated, and the charging schedule recalculated at every timestep, as EVs arrive and depart and the number of active EVs fluctuates. In every

iteration, the remaining energy demand and the remaining charging duration are adjusted as follows:

$$e_i(t_k) = e_i(t_{k-1}) - V r_i(t_k)\Delta t$$

$$d_i(t_k) = d_i(t_{k-1}) - 1$$

where Δt is the difference between timesteps.

5.2 Qubit efficient quantum optimization for smart EV charging

5.2.1 Cost function

In this section we will convert the original scheduling problem into QUBO and then employ one of the previous methods to solve it. Initially we need to have to define a cost function that describes the problem. Essentially, the cost function should be a combination of multiple cost functions, each serving a different aspect of the problem.

The first cost function penalizes any unmet energy and time demands. It will be referred to as "NC," which stands for "non-completion," and is defined as follows:

$$U^{NC}(\mathbf{r}) = \sum_{i=1}^{N} \left(\sum_{t_k=t_0}^{\tau_i^{end}} (V \cdot r_i(t_k) \cdot \Delta t) - e_i \right)^2 \tag{5.1}$$

The cost function in (7) ensures that EV i will receive the full amount of energy it requested e_i before the deadline τ_i^{end}. To reformulate this in a more convenient form, we eliminate τ_i^{end} by introducing indicators δ_{ik}, where $\delta_{ik} = 1$ if the timestep t_k is prior to the deadline τ_i^{end} ($t_k < \tau_i^{end}$), and $\delta_{ik} = 0$ otherwise. Given this, we rewrite U^{NC} as:

$$U^{NC}(\mathbf{r}) = \sum_{i=1}^{N} \left(\sum_{t_k=t_0}^{T} (V \cdot \delta_{ik} \cdot r_i(t_k) \cdot \Delta t) - e_i \right)^2 \tag{5.2}$$

Where T is the optimization horizon.

The second cost function accounts for the minimization of energy at each timestep. It will be referred to as "LV," which stands for "Load-variation":

$$U^{LV}(\mathbf{r}) = \sum_{t_k=t_0}^{T} \left(\sum_{i=1}^{N} r_i(t_k) \right)^2 \tag{5.3}$$

The third cost function aims to prioritize high-load charging in the initial timesteps, thereby clearing the field for potentially arriving EVs. It will be referred to as "QC," which stands for "Quick-charging":

$$U^{QC}(\mathbf{r}) = -\sum_{t_k=t_0}^{T} \frac{T-t_k}{T-t_0} \sum_{i=1}^{N} r_i(t_k) \tag{5.4}$$

Finally, we include a penalty term to prevent the network from charging beyond its power capacity:

$$\sum_{i=1}^{N} V \cdot r_i(t_k) \leq C \quad \forall t_k \tag{5.5}$$

Where C is the power capacity of the network. As mentioned, the problem's cost function is a weighted sum of the previous cost functions, including the penalty term:

$$\begin{aligned} \underset{\mathbf{r}}{\text{minimize}} \quad & U(\mathbf{r}) := w^{NC}U^{NC}(\mathbf{r}) + w^{LV}U^{LV}(\mathbf{r}) + w^{QC}U^{QC}(\mathbf{r}) \\ \text{subject to} \quad & \sum_{i=1}^{N} V \cdot r_i(t_k) \leq C \quad \forall t_k, \\ & r_i(t_k) \in \{0, 8, 16, 32, 48, 64\} \quad \forall i, k. \end{aligned}$$

or equivalently

$$\begin{aligned} \underset{\mathbf{r}}{\text{minimize}} \quad & U(\mathbf{r}) := \\ & w^{NC} \sum_{i=1}^{N} \left(\sum_{t_k=t_0}^{T} (V \cdot \delta_{ik} \cdot r_i(t_k) \cdot \Delta t) - e_i \right)^2 \\ & + w^{LV} \sum_{t_k=t_0}^{T} \left(\sum_{i=1}^{N} r_i(t_k) \right)^2 + \\ & w^{QC} \sum_{t_k=t_0}^{T} \frac{T-t_k}{T-t_0} \sum_{i=1}^{N} r_i(t_k) \\ \text{subject to} \quad & \sum_{i=1}^{N} V \cdot r_i(t_k) \leq C \quad \forall t_k, \\ & r_i(t_k) \in \{0, 8, 16, 32, 48, 64\} \quad \forall i, k. \end{aligned}$$

5.2.2 Binary Conversion

The cost function is clearly quadratic, which aligns well with the requirements for converting the problem into QUBO format. However, we also need to ensure it is binary, which means we must devise a binary representation of $r_i(t_k)$ without violating the quadratic nature of the cost function.

A potential binary encoding for the set $\{0, 8, 16, 32, 48, 64\}$ employs a three-qubit representation:

$$r_i(t_k) = x_{i,k,2} \cdot 16 \cdot \sum_{q=0}^{1} 2^q x_{i,k,q} + (1 - x_{i,k,2}) \cdot 8 \cdot x_{i,k,1} \tag{5.6}$$

where the first term encodes the values {16,32,48,64} and the second term accounts for the values {0,8}, with the qubit $x_{i,k,2}$ acting as a control. However, this encoding introduces terms like $x_{i,k,2}x_{i,k,1}$ which increase the order of the cost function. Given the challenge of finding a linear binary encoding for $r_i(t_k)$ within this set of values, we opt to simplify the set to {0,16,32,48}, which can be straightforwardly encoded using two qubits as:

$$r_i(t_k) = 16 \sum_{q=0}^{1} 2^q x_{i,k,q} \tag{5.7}$$

5.2.3 Incorporating Inequality constraints

The encoding in (5.7) preserves the second-order nature of the cost function. The final step to fully convert the problem into QUBO is incorporating the inequality constraint into the cost function. Normally, equality constraints are integrated into the cost function as squared error penalty terms. To include an inequality constraint, we first transform it into an equality constraint using a slack variable s, as suggested by [20] and [21]. Therefore, our inequality constraints are adjusted as follows:

$$\left(\sum_{i=1}^{N} V \cdot r_i(t_k) \right) + s_k = C \quad \forall k \tag{5.8}$$

Now, as an equality constraint, it can be integrated into the initial cost function as an additional term Q^P, where:

$$Q^P = \sum_{k=1}^{n} \left(\left(\sum_{i=1}^{N} V \cdot r_i(t_k) \right) + s_k - C \right)^2 \tag{5.9}$$

Here, n represents the number of timesteps in the horizon. The slack variables s_k are positive integers that represent the residual power capacity, thus the number of qubits required to encode a single slack variable is at most $\lceil \log_2(C) \rceil$, and consequently, for all slack variables, we need $n\lceil \log_2(C) \rceil$. For instance, with a capacity of 100kW and a 6 timestep horizon, an additional $6 \cdot 7 = 42$ classical variables would be needed. This represents a significant increase in qubit requirements. To avoid this increase, we employ an alternative method that bypasses the inequality constraints entirely, focusing on controlling the network load by adjusting accordingly the weight w^{LV} of the "Load Variation" cost function. The tighter the power capacity constraint, the greater the weight w^{LV} should be. Throughout the experiments, this method will be used, but for comparisons with classical algorithms, the slack variable implementation will be utilized to ensure fairness in comparison.

5.3 Implementation in simulators and cloud quantum hardware

The first test to verify our formulation's effectiveness will be the application of a full encoding VQA, complemented by our efficient encoding scheme.We will employ our custom-developed algorithm for the implementation of general encoding schemes for these experiments. We select a 2-body correlation encoding, where the ancilla consits of 2 qubits. This particular encoding scheme is chosen to capture at least the correlation between the two qubits ($x_{i,k,0}$ and $x_{i,k,1}$) that encode a single variable $r_i(t_k)$. This approach will be referred to as "Compressed VQA". The problem description relates to EVs that need to be fully charged with 50kW in 6 hours and EVs that require top-ups of 10kW in 1 hour. For illustrative purposes, two algorithms will be applied to a toy example with a Horizon of $T = 4$ hours, a timestep of $\Delta t = 1$ hour, a constant Voltage of $V = 240V$, and 4 EVs, where 3 require receiving 26.88 (full charge) kWh within 3 hours, and 1 requires 7.68 kWh within just 1 hour (top up). Recall that we have set the allowable currents to {0, 16, 32, 48}. Therefore, two possible configurations for the full charge would be {32, 32, 48} and {16, 48, 48}, whereas for the top-up, it is {32}. For this experiment, we will disregard the power capacity of the network. The experiment will be conducted using the Qiskit

library with a noise-free qasm-simulator. This toy example involves $4 \times 4 = 16$ classical variables $r_i(t_k)$, which can be encoded using 32 qubits for the full encoding algorithm. This count aligns with the maximum number of 32 qubits supported by the qasm simulator. Therefore, larger scale examples will be conducted using the Compressed VQA only. Figure 5.3 shows the comparison of our Compressed VQA with traditional full enconding VQA for a 2-layer ansatz. The schedule produced by each algorithm was:

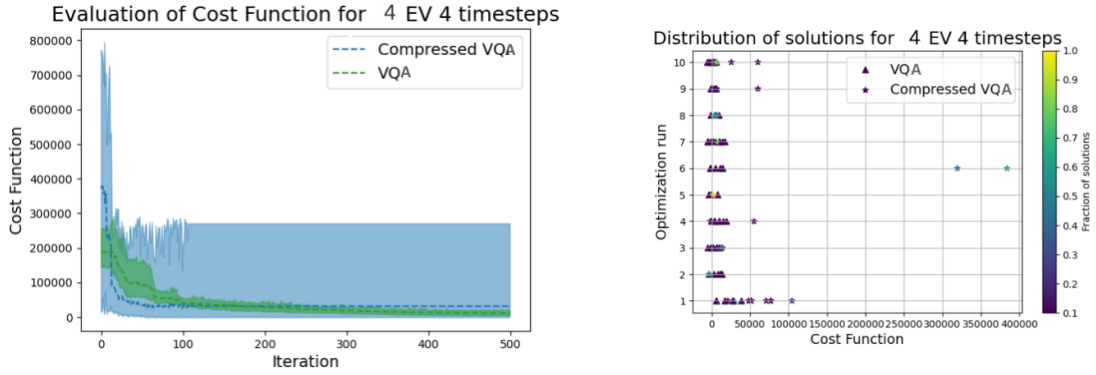


Figure 5.3: Comparison of the Full Encoding VQA against our qubit efficient approach, or Compressed VQA, for the Toy Example with a 2-Layer Ansatz across 10 experiments and 10000 circuit shots. The dashed line indicates the mean of all experiments. On the left, we observe the evolution of the cost function with respect to the optimizer iterations, and on the right, we see the distribution of solutions after convergence for each experiment. This toy example contains 32 binary variables and 32 qubits for the standard full enconding VQA whereas in our case only 6 qubits and a fraction of the corresponding gates!.

for Compressed VQA:

$$\mathbf{r} = \begin{bmatrix} 32 & 32 & 48 & 0 \\ 32 & 48 & 32 & 0 \\ 32 & 48 & 32 & 0 \\ 32 & 0 & 0 & 0 \end{bmatrix}$$

for full encoding VQA:

$$\mathbf{r} = \begin{bmatrix} 16 & 48 & 48 & 0 \\ 32 & 32 & 48 & 0 \\ 48 & 32 & 32 & 0 \\ 32 & 0 & 0 & 0 \end{bmatrix}$$

The Compressed VQA circuit employs 2 qubits for the ancilla and $\log_2(16/2) = 4$ for the register, summing up to 6 qubits in total. For the 2-Layer Ansatz, the number of necessary rotational R_Y gates for 1 qubit is 12, and the circuit demands 5 two-qubit $CNOT$ gates. In contrast, the traditional VQA approach, involving 32 qubits, requires 64 R_Y gates and 31 $CNOT$ gates.

5.3.1 Large Scale Example

Having confirmed that the model is effective for a toy example, we must now examine its performance on larger scale problems of few hundred or thousand binary variables. As previously mentioned, the use of traditional VQA is not feasible beyond this point as it would require hundreds or thousands of well controlled qubits, which even for the simulator case, would be impossible to handle; therefore, we will proceed exclusively with the qubit efficient approach. We will conduct experiments with a Horizon of 12 timesteps, each separated by half an hour, resulting in an 6-hour horizon, and involving 64 EVs. We will evaluate the results for varying numbers of circuit shots and varying numbers of ansatz layers. For the 2-Layer Ansatz, this example would require 1.536 classical variables, which means $2 + \log_2(1.536/2) = 12$ qubits, resulting in 24 R_Y gates and 11 $CNOT$ gates. The optimization parameters are as many as the R_Y gates. Additionally, we will set up a second large-scale example with 128 EVs and a 12-timestep horizon. This scenario would require 3.072 classical variables, necessitating $2 + \lceil \log_2(3.072/2) \rceil = 13$ qubits, which translates to 26 R_Y gates and 12 $CNOT$ gates. In Figure 5.4, we observe the outcomes for both examples. In each scenario, an increase in the number of circuit shots enhances the results, as expected. In contrast, increasing the number of ansatz layers does not result in significant improvements, and in the case of 128 EVs, it even diminishes the accuracy of the algorithm, a behaviour associated with the barren plateau phenomenon. In the leftmost plot of the figure 5.4, we observe the distribution of solutions derived from

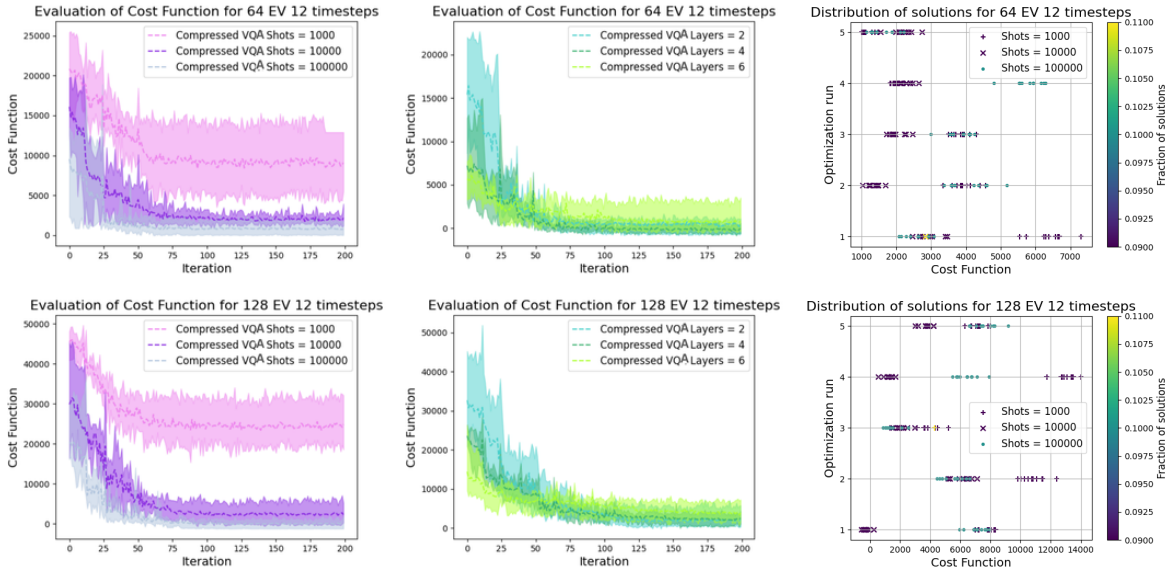


Figure 5.4: Left and middle plots present the evolution of the cost function with respect to the optimizers iteration for shots = 1000, 10000, 100000 (Left) and Layers = 2, 4, 6 (Middle) across 5 experiments. The dashed line indicates the mean of all experiments. Right plot shows the distribution of the solutions having acquired the optimal parameters. The **Upper plots** refer to a large-scale example with 64 EVs and a 12-timestep horizon which corresponds to 1.536 binary variables and 12 qubits, while the **Lower plots** pertain to a large-scale example with 128 EVs and a 12-timestep horizon which corresponds to 3.072 binary variables and 13 qubits.

the circuit using the optimized parameters for 5 different experiments. In contrast to the corresponding plot of the toy example in fig 5.3, where the largest proportion of the solution was associated with the optimal cost, here we obtain smaller clusters of solutions with suboptimal costs but relatively close to each other. The suboptimal and inconsistent solutions extracted from the post-optimized circuit can be explained by certain registers not being measured, resulting in random assignments for their associated binary variables.

5.3.2 Noisy Simulation

In this section, we investigate the performance of our encoding scheme under the effects of a noise model consisting of thermal relaxation errors, imperfect gate fidelities, and read-out errors. Thermal relaxation and decoherence can be characterized by the relaxation

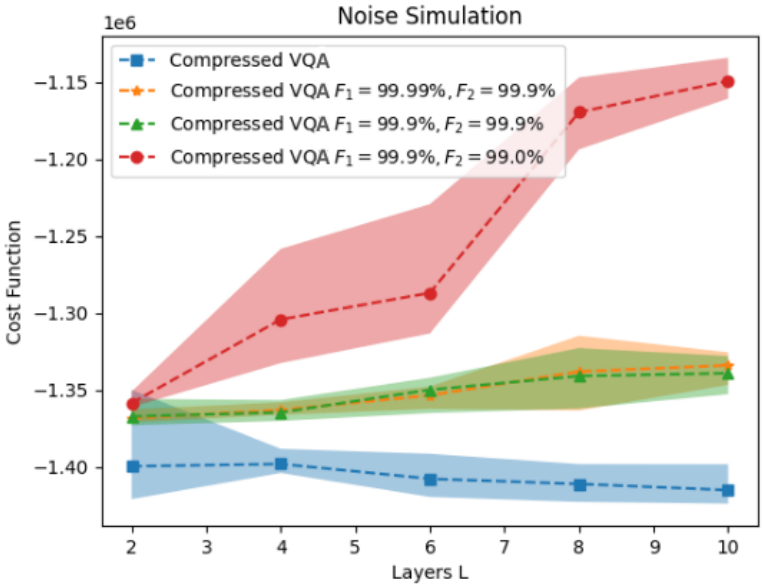


Figure 5.5: Comparison of the Compressed VQA for the Toy Example, in the presense of hardware noise and finite gate fidelities, across 5 Experiments with varying-layer Ansatz. We investigate how increases in circuit depth affect the outcomes across different gate fidelities. The noise model incorporates depolarization error, thermal relaxation error, and readout error. The readout error is fixed at 1% for all experiments, as well as the relaxation time ($T_1 = 50$ ms) and dephasing time ($T_2 = 70$ ms) both of which also remain constant throughout the experiments.

constants T_1 and T_2 respectively Gate errors are implemented via a depolarization channel that affects each qubit as it undergoes a gate operation. Readout error is the probability of obtaining an incorrect value of the qubit during measurement, i.e. reading a $|0\rangle$ when the qubit is in the $|1\rangle$ state and vice versa. In Figure 5.5, we observe the outcomes of applying various noise models with differing gate fidelities to the toy example. It is evident how an increase in circuit depth makes the algorithm increasingly susceptible to noise. In scenarios where noise is present, this vulnerability is pronounced. In the ideal case devoid of noise, increasing the circuit depth does not affect the outcome.

5.3.3 Execution in Cloud QPUs

In this section, we execute the qubit efficient quantum circuits on actual Quantum Processing Units (QPUs) provided by AWS. The experiment involves post-optimization sampling of solutions for the three examples we have previously run: the toy example of 4 EVs and 4 timestep horizon (32 variables), the large example of 64 EVs with a 12-timestep horizon (1536 variables), and the extra-large example of 128 EVs with a 12-timestep horizon (3072 variables). All the problem sizes are feasible only via the qubit compression approach and are to our knowledge the largest possible EV charging problems ever solved on NISQ devices. Standard VQAs or QAOA would require 32, 1536, and 3072 well controlled qubits to provide a solution.

As it is standard in the literature, we simulate the hybrid optimization process to acquire the optimized parameters/angles in a simulator and the feed this to the QPU. Following this, we execute the optimized circuit on the QPU and measure using 5,000 shots. We extract 10,000 solutions from the final circuit measurement and plot the cumulative distribution of the eventual cost function values returned by each solution. The available QPUs on the AWS platform for the challenge are "Lucy" by Oxford Quantum Circuits (OQC) with 8 qubits, and "Aria 1" by IonQ with 25 qubits. In the large scale experiments we could not use the "Lucy" device. We compare the performance of both these devices against the statevector simulator "SV1" by AWS in Figure 5.6

In addition to the solution sampling experiment, we also attempted to run a complete hybrid job using a real QPU. Hybrid jobs involve both classical and quantum operations and require multiple quantum tasks to be run until convergence is reached. To minimize costs and stay within the credit given to us, we selected the toy example to be executed as a hybrid job using "Lucy" by OQC as the QPU, as using IonQ would exceed our budget. Each iteration in the optimization procedure corresponds to a different quantum task. Therefore, we performed 60 iterations, with 5,000 shots per quantum task. The results are displayed in the figure 5.7 where it is evident that although the value of the cost function decreases over the iterations, it fails to converge to the optimal value of 0. This is primarily due to the noisy hardware and the insufficient number of shots.

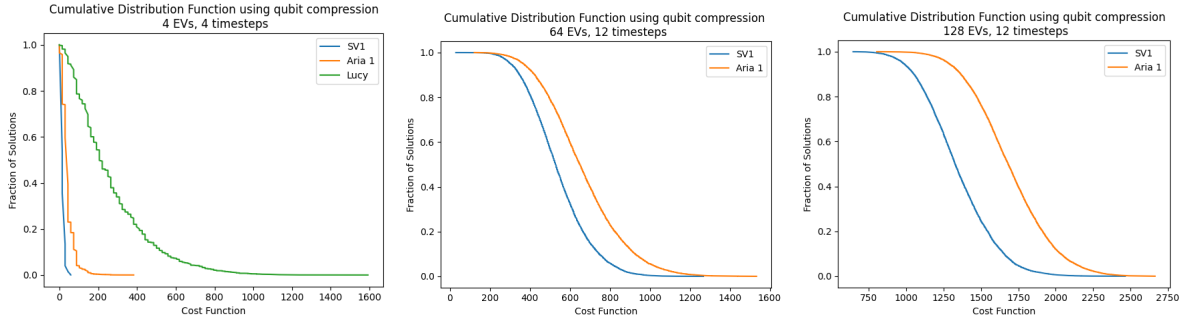


Figure 5.6: Comparison of the performance of our qubit efficient algorithm in QPUs against the statevector simulator for the toy example and 32 variables (left), and the large-scale example of 1536 variables (middle), and extra large-scale example of 3072 variables (right). Each curve represents the cumulative distribution of 10,000 solutions retrieved from the corresponding quantum circuit, measured with 5,000 shots. For example, the point (600, 0.3) on the graph indicates that 30% of the retrieved solutions have a cost function value greater than 600. The closer the curve of the cumulative distribution is to the y-axis, the better the result.

5.4 Comparison with Classical Algorithms

Finally, we will compare the outcomes of the qubit efficient quantum optimization approach with those from existing classical scheduling algorithms. The classical algorithms to be evaluated include the First-Come, First-Serve (FCFS), Earliest Deadline First (EDF), and Least Laxity First (LLF) algorithms. All algorithms will undergo testing in a straightforward simulation for the MPC formulation we follow here, involving 100 EVs and will be compared based on demand fulfillment and energy consumption. Specifically, the first experiment aims to assess the percentage of demand satisfaction under varying power capacities of the network. The second experiment will examine energy consumption given a fixed power capacity of the network. The qubit efficient quantum optimization solution, in short "quantum MPC", will predict outcomes over an 6-hour horizon, utilizing a circuit with 6 layers and 10,000 circuit shots for different numbers of qubits depending the size of the problem. It will undergo 15 experiments from 15 different starting points to avoid becoming trapped in a local minimum. Conducting multiple experiments is crucial due to the probabilistic nature of our quantum MPC. Thus, we

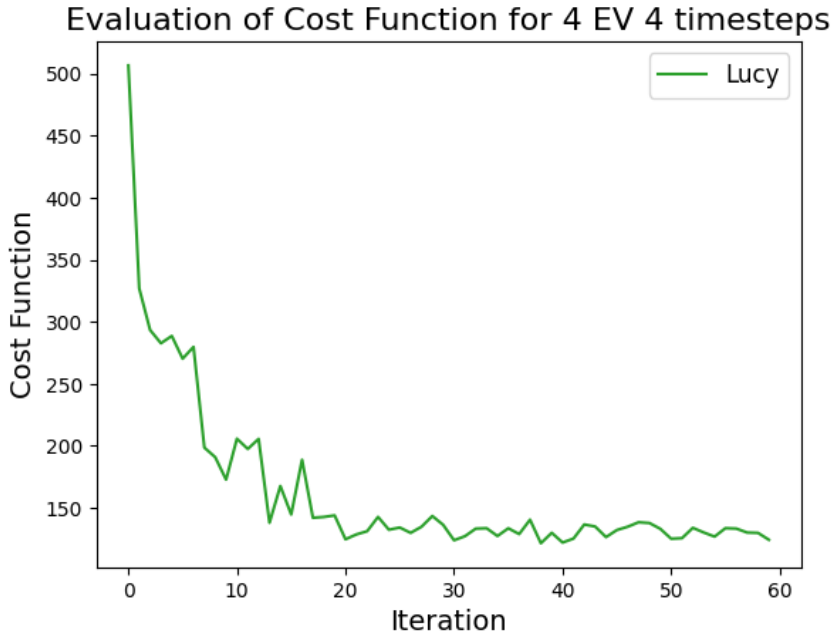


Figure 5.7: Evolution of the cost function over multiple iterations for the toy example involving 4 EVs and a 4-timestep horizon requiring 32 qubits if done with standard VQA or QAOA approaches. The experiment was conducted on AWS Braket using the "Lucy" QPU by Oxford Quantum Circuits comprising of only 8 qubits. It encompassed 60 iterations, with each iteration representing a new quantum task. For each quantum task, the quantum circuit was measured with 5,000 shots.

will examine its performance across both average and best case scenarios. In Figure 5.8, we show the results of this comparison. In terms of demand satisfaction, the plot reveals that in the best case scenario, our quantum MPC algorithm is more efficient under stricter power capacity limitations and converges to a 100% demand met percentage as quickly as the classical algorithms. In the average case scenario, while still more efficient at lower power capacities, it never fully converges to 100% demand met due to the various approximations it includes. In the case of energy consumption comparison, which is performed for a simulation of 100 EVs in a timespan of 24 hours with 1 Hour timestep, it is clearly evident that the quantum MPC, in both best case and average case, manages to reduce the peak of the cumulative power consumption. This is achieved by its ability of looking forward in its 6 hours Horizon and distribute the energy efficiently throughout all the timesteps

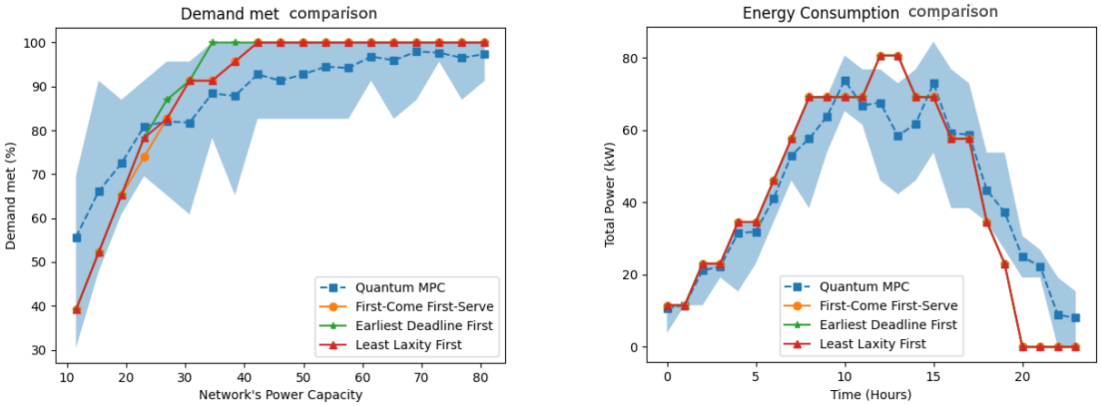


Figure 5.8: Demand met and energy consumption comparison for our qubit efficient quantum MPC solution, First-Come, First-Serve (FCFS), Earliest Deadline First (EDF), and Least Laxity First (LLF) classical algorithms. In the demand met comparison (left), we observe how the percentage of demand coverage for each algorithm increases with the maximum network power supply. In the energy consumption comparison (right), we note the total power consumed by all EVs at each timestep for a 24 hours time span. We observe our quantum solution to on par with the classical ones, and in certain instances even better, in spite the NISQ character of the hardware used, the latter attributed to the minimal enconding nature of our method.

5.5 Conclusion

In this work, we formulate the smart scheduling of EV charging problem as a QUBO problem and solved it using hybrid quantum-classical variational algorithm combined with our novel quantum encoding scheme which allowed us to implement solutions of few thousand binary variables with NISQ devices. We tested the accuracy of our minimal enconding approach with the full encoding for different problem sizes in simulators, and then extended our analysis to large problem sizes that are currently beyond the reach of standard VQAs or QAOA for NISQ. More specifically we solved for problems involving 64 EVs with a 12-timestep horizon, which corresponds to 1,536 binary variables, and 128 EVs with a 12-timestep horizon, equating to 3,072 variables. More specifically, we demonstrate solutions for a range of problems sizes up 128 EVs and a 12-timestep horizon which corresponds to 3.072 binary variables, using only 13 qubits (instead of 3072 qubits

required in standard QAOA or VQA approaches). We tested our solutions in simulators for a range of problem sizes, and parameters, and also run smaller instances on cloud quantum hardware provided by AWS for the challenge. We also test our solutions as against classical approaches and find good agreement for a range of parameter and problem sizes signifying the strenth of the minimal enconding techniques

Our results indicate that the qubit compression method is able to find classical solutions of similar quality to those obtained using the full encoding scheme for the EV charging problem at hand, and comparable to classical algorithms despite utilizing significantly fewer qubits. As the number of available qubits in NISQ devices becomes larger by the year, our qubit efficient approach open a possible avenue for demonstrating useful quantum advantage for EV charging problems in the near to mid term future.

Chapter 6

Conclusion and Future Work

In this thesis, we have explored the potential of hybrid quantum-classical algorithms, particularly focusing on their applications in the optimization of electric vehicle (EV) charging schedules. The integration of quantum computing principles with classical optimization techniques provides a promising approach to solving complex combinatorial problems more efficiently than traditional methods alone.

We began with a detailed introduction to the fundamental principles of quantum mechanics, establishing a strong foundation for understanding the behavior and manipulation of qubits, quantum gates, and entanglement. The thesis then moved to the Quadratic Unconstrained Binary Optimization (QUBO) problem, highlighting its relevance and challenges in various industrial applications.

The research presented innovative qubit-efficient encoding schemes that optimize the use of quantum resources, making the problem-solving process more efficient. These encoding schemes were rigorously tested through theoretical analysis and practical implementation on quantum simulators and cloud-based quantum hardware. The results demonstrated the efficacy of these methods in managing large-scale optimization problems.

A key application discussed was the smart scheduling of EV charging, a critical challenge in modern energy management systems. By applying hybrid quantum-classical algorithms, we showed that it is possible to achieve more efficient scheduling solutions, reducing computational complexity and improving scalability compared to classical approaches. The comparison between quantum and classical methods underscored the po-

tential advantages of quantum computing in practical, real-world scenarios.

As future work, we are aiming to explore the capabilities of qubit-efficient encoding schemes in a variety of other industry applications. These applications span across different sectors such as logistics, healthcare, finance, and telecommunications, where optimization problems are prevalent and can often be formulated as Quadratic Unconstrained Binary Optimization (QUBO) problems. By leveraging the qubit-efficient techniques developed in this thesis, we anticipate achieving significant improvements in computational efficiency and solution quality for complex, real-world optimization challenges. This exploration will not only validate the versatility and robustness of our encoding schemes but also potentially uncover new use cases where quantum computing can offer substantial advantages over classical methods.

Bibliography

[1] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996.

[2] Barry A. Cipra. An introduction to the ising model. *The American Mathematical Monthly*, 94(10):937–959, 1987.

[3] STEPHEN G. BRUSH. History of the lenz-ising model. *Rev. Mod. Phys.*, 39:883–893, Oct 1967.

[4] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution, 2000.

[5] David Amaro, Carlo Modica, Matthias Rosenkranz, Mattia Fiorentini, Marcello Benedetti, and Michael Lubasch. Filtering variational quantum algorithms for combinatorial optimization. *Quantum Science and Technology*, 7(1):015021, feb 2022.

[6] A.B. Finnila, M.A. Gomez, C. Sebenik, C. Stenson, and J.D. Doll. Quantum annealing: A new method for minimizing multidimensional functions. *Chemical Physics Letters*, 219(5):343–348, 1994.

[7] Suguru Endo, Zhenyu Cai, Simon C. Benjamin, and Xiao Yuan. Hybrid quantum-classical algorithms and quantum error mitigation. *Journal of the Physical Society of Japan*, 90(3):032001, 2021.

[8] Adam Callison and Nicholas Chancellor. Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond. *Phys. Rev. A*, 106:010101, Jul 2022.

[9] Xiao Yuan, Suguru Endo, Qi Zhao, Ying Li, and Simon C. Benjamin. Theory of variational quantum simulation. *Quantum*, 3:191, October 2019.

[10] Tobias Haug, Kishor Bharti, and M.S. Kim. Capacity and quantum geometry of parametrized quantum circuits. *PRX Quantum*, 2:040309, Oct 2021.

[11] V. Akshay, H. Philathong, M. E. S. Morales, and J. D. Biamonte. Reachability deficits in quantum approximate optimization. *Phys. Rev. Lett.*, 124:090504, Mar 2020.

[12] Mateusz Ostaszewski, Edward Grant, and Marcello Benedetti. Structure optimization for parameterized quantum circuits. *Quantum*, 5:391, January 2021.

[13] Hiroshi C. Watanabe, Rudy Raymond, Yu-Ya Ohnishi, Eriko Kaminishi, and Michihiko Sugawara. Optimizing parameterized quantum circuits with free-axis selection. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 100–111, 2021.

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

[15] 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.

[16] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1), Mar 2021.

[17] 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.

[18] Zachary J. Lee, Sunash Sharma, Daniel Johansson, and Steven H. Low. Acn-sim: An open-source simulator for data-driven electric vehicle charging research. *IEEE Transactions on Smart Grid*, 12(6):5113–5123, 2021.

[19] Zachary J. Lee, George Lee, Ted Lee, Cheng Jin, Rand Lee, Zhi Low, Daniel Chang, Christine Ortega, and Steven H. Low. Adaptive charging networks: A framework for smart electric vehicle charging. *IEEE Transactions on Smart Grid*, 12(5):4339–4350, 2021.

[20] Tomáš Vyskočil, Scott Pakin, and Hristo Djidjev. *Embedding Inequality Constraints for Quantum Annealing Optimization: First International Workshop, QTOP 2019, Munich, Germany, March 18, 2019, Proceedings*, pages 11–22. 01 2019.

[21] Hristo N. Djidjev. Quantum annealing with inequality constraints: the set cover problem, 2023.