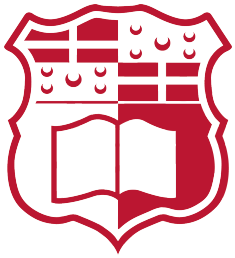


Variational Quantum Algorithms for Combinatorial Optimisation

Navigating the NISQ Era

Evan Camilleri

Supervised by Professor Tony John George Apollaro

Department of Physics

Faculty of Science

University of Malta

March, 2025

A dissertation submitted in partial fulfilment of the requirements for the degree of M.Sc. in Physics.

WWW.UM.EDU.MT

First edition, Thursday 10th July, 2025

To the Stars that Guide My Voyage:

My late parents, Yvonne and Silvio, who shaped my past,
My wife, Tanya, who enriches my present,
And my sons, Luke and Matthew, who inspire my future.

Acknowledgements

I pursued this Master's degree as part of a long-standing challenge I had set for myself several years ago. Coming from a background in classical computing and mathematics, completing this project would not have been possible without the impeccable mentorship and patience of Professor Tony John George Apollaro. Professor Apollaro was not only my supervisor. His unwavering support, insightful critiques, inspiring passion for academic excellence, and meticulous attention to detail helped me significantly throughout this dissertation.

I would also like to thank Dr Mirko Consiglio, who provided me with valuable insights during my journey and, together with Professor Apollaro, helped shape my research.

My appreciation also goes to the faculty and staff of the Department of Physics at the University of Malta and my colleagues at Quantum Malta. I am especially grateful to Professor André Xuereb, who accepted me as a student within the Department and recommended me to Professor Apollaro.

I must express my heartfelt gratitude to my late parents, who always encouraged me to aim high and remain committed to the projects I undertake. This journey was particularly challenging for me, as I experienced the loss of my father during this time. I am deeply thankful to my wife, Tanya, and my sons, Luke and Matthew, for their patience and unwavering support throughout this endeavour. I hope to have set an example for my sons, demonstrating the importance of perseverance and dedication to one's goals.

Abstract

Optimisation algorithms aim to solve a wide range of problems, including those related to physical systems and everyday challenges. Examples include optimising the shape or material distribution of structures to maximise strength and minimise weight, designing optimal investment portfolios, balancing risks and returns, planning efficient routes, optimising container placement on ships, and improving the manufacturing plan in factories. Optimising manufacturing plans in factories is the primary focus of my research. The goal of each optimisation problem typically involves reducing the costs by adjusting the algorithm's configuration.

For instance, in route optimisation, costs typically involve time and fuel, but constraints such as time windows or availability at destinations must also be considered. Incorporating more parameters can lead to solutions that better reflect real-world scenarios. However, increasing the number of parameters adds complexity to the problem, and more resources are required to achieve an optimal solution. In many practical applications, finding the exact optimal solution is often unnecessary. In many cases, finding a solution optimising the current status within an acceptable time frame is sufficient.

The challenge of optimising the manufacturing plan in factories, often referred to as the Job-Shop Scheduling Problem (JSSP), involves balancing several parameters. These include sale orders and their requested delivery dates, the expected arrival dates for raw materials derived from the bill of materials of the products listed in sale orders, and resource availability, both machines and human resources required in the production process. Additionally, the process is subject to several constraints, such as sequential dependencies—for example, producing one component before another—and process coordination, such as synchronising different production stages, where manufacturing must be completed before packaging can begin. Operators balance these parameters and constraints to minimise costs, such as reducing changes in the configuration of machines or reducing clean-up tasks between the manufacturing of different components and maximising resource utilisation while keeping up with promised delivery dates to customers.

The JSSP is a combinatorial optimisation problem that is NP-hard, meaning it becomes computationally difficult to compute as the number of parameters increases using classical computing methods. However, there are still various classical approaches to the problem, which result primarily in heuristic solutions. These include Integer Linear Programming (ILP), Dispatching Rules, Genetic Algorithms, or Simulated Annealing.

In this dissertation, I investigate the quantum computing algorithm Quantum Approximate Optimization Algorithm (QAOA) to address the JSSP problem.

Quantum computing is still in its infancy. Its foundation lies in quantum mechanics, which involves mathematical concepts such as linear algebra, complex numbers, and probability amplitudes. Key properties of quantum mechanics like

superposition, entanglement and quantum interference allow quantum computers to explore the solution space of a problem more effectively than classical computers. In this dissertation, I first examine the properties of quantum mechanics, which can help me investigate QAOA and study tools that I can use to program a quantum computer. Then, I explore the mathematical models used to represent the problem, in this case, the Job-Shop Scheduling Problem (JSSP).

JSSP involves several interdependent parameters. I expressed these parameters in mathematical models containing these interactions, some containing three or more parameters interacting with each other. The result was mathematical formulations involving problems with multivariate objective functions, which were then solved with QAOA. Another objective of my dissertation was to explore the possibility of reducing resource requirements while still achieving sufficiently good results based on relevant figures of merit.

In this dissertation, I investigate various configurations of QAOA, evaluating their success while factoring in the computational resources required for execution to achieve an optimal result. I propose a configuration of QAOA, which I call k-interaction Angle QAOA (ka-QAOA). I show that ka-QAOA performs comparably to other proposed QAOA configurations while reducing the computational resources required to achieve similarly good approximations.

While the results are promising, as parameters in problems increase, testing the different configurations of QAOA becomes a daunting task. More robust, error-free quantum computers are needed to model and solve real-world optimisation problems like JSSP. In the interim, we can still experiment with problems having few parameters to find optimal quantum algorithms.

Foreword

Developments in **Quantum Computing** reignited my passion for finding better solutions for complex optimisation problems, particularly the Job-Shop Scheduling Problem (JSSP), which was one of the most complex tasks I was assigned to solve as a software architect. Using classical algorithms, my team and I optimised the client's scheduling. We achieved a satisfactory improvement of thirty per cent over the client's existing manual procedures, solving the problem more efficiently - using less time, fewer resources, and with reduced errors.

While this was an achievement, it highlighted the limitations of classical algorithms when tackling combinatorial optimisation problems. My research into a better solution for the JSSP led me to quantum computing. While quantum computing is still in its infancy, it offers promising avenues to solve optimisation problems, particularly with Variational Quantum Algorithms (VQAs). These hybrid algorithms, which combine classical and quantum computing approaches, help bridge quantum concepts and real-world applications during the Noisy Intermediate-Scale Quantum (NISQ) era. The algorithms help leverage the power of noisy intermediate-scale quantum devices, making it possible to address previously infeasible problems. At the same time, the community awaits a high-fidelity multi-qubit quantum computer to achieve quantum advantage.

Contents

List of Figures

List of Tables

List of Abbreviations

am-QAOA Automorphic Angle QAOA xiv
AP Actual Probability xiv
AQC Adiabatic Quantum Computing 19
AR Approximation Ratio xiv
BQP Bounded-Error Quantum Polynomial 47
CUBO Cubic Unconstrained Binary Optimisation xv
JSSP Job-Shop Scheduling Problem ix
ka-QAOA k-interaction Angle QAOA x
ma-QAOA Multi Angle QAOA xiv
Max-Cut Maximum Cut xiv
nfev Number of Function Evaluations xiv
NFL No-Free Lunch 132
NISQ Noisy Intermediate-Scale Quantum xi
NP Non-Polynomial 30
P Polynomial 46
PUBO Polynomial Unconstrained Binary Optimisation xiv
QAA Quantum Adiabatic Algorithm 50
QAOA Quantum Approximate Optimization Algorithm ix
QKD Quantum Key Distribution 132
QUBO Quadratic Unconstrained Binary Optimisation xiv
SKU Stock Keeping Unit 43
TQA Trotterised Quantum Annealing 61
TSP Travelling Salesman Problem 28

1

Introduction

Following the success of the theory of quantum mechanics [1], scientists today can model and predict the behaviour of various physical systems. Building on this success, quantum computing has recently gained traction and received media attention due to the promise of solving highly complex problems. Quantum computing acts as a disruptor and a solution provider in communication security [2]. It promises to help researchers develop new drugs and materials [3] and solve complex problems like the Job-Shop Scheduling Problem (JSSP) [4] [5].

1.1 | Motivation

An increased number of publications related to developments in **Quantum Computing** and recent investments from corporations and governments all point to the importance of this new technology. Quantum computing promises to solve problems that are currently too complex for classical computing. Research in quantum technologies spans a wide variety of topics. This research includes the development of hardware to build quantum computers, the design of algorithms, cryptography, communication, sensing, error correction, material science, medicine and various applications in physics.

While quantum computing is still in its infancy, it offers promising avenues to solve optimisation problems, particularly with Variational Quantum Algorithms (VQAs). These hybrid algorithms, which combine classical and quantum computing approaches, help bridge quantum concepts and real-world applications during the NISQ era. The algorithms help leverage the power of noisy intermediate-scale quantum devices, making it possible to tackle problems previously considered infeasible. In the meantime, the community awaits the development of a high-fidelity, multi-qubit quantum computer capable of demonstrating quantum advantage.

Although much public interest in quantum computing stems from its risk of breaking Internet security, with big players investing both in developing quantum hardware and the so-called quantum-safe algorithms, its real challenges lie in solving real-world applications. Its applications extend to scientific problems that lead to discoveries in new medicine, better materials, or practical, real-world applications such as optimisation.

1.2 | Aims and Objectives

This research focuses on studying Variational Quantum Algorithms (VQAs) to solve combinatorial optimisation problems. I investigate various variants of the Quantum Approximate Optimization Algorithm (QAOA), a VQA, with different parameters and configurations. During this research, I found that JSSP can be formulated as a polynomial minimisation problem. However, since it remains NP-hard, I explore the application of quantum circuits with similar polynomial complexity. To streamline the approach, I specifically focus on cubic problems.

The ultimate goal of this study is to find an algorithm that achieves similar fidelity as existing quantum algorithms while reducing the computational resources required in iterations between the quantum and classical components of the hybrid algorithm.

1.3 | Document Structure

In **Section 2**, I introduce quantum computing concepts such as the qubit, the fundamental data processing unit in quantum computing, and its related visual concept, the Bloch Sphere. I then study several qubit properties that give qubits greater computational potential than classical bits. A quantum computer can be programmed through unitary operations coded in quantum gates, and I study several of these gates. Finally, I introduce the Ising model and the Adiabatic Theorem, the theoretical framework on which QAOA is based.

In **Section 3**, I study how to program combinatorial optimisation problems using QAOA on a quantum computer. I explain how I can represent the problems using graph theory with specific notation using Quadratic Unconstrained Binary Optimisation (QUBO) and Polynomial Unconstrained Binary Optimisation (PUBO) models. Then, I map these models onto an Ising model, which can subsequently be used to build an algorithm using quantum gates on a quantum computer. I define several combinatorial optimisation problems that can be programmed on a quantum computer using QAOA and explain problem complexity or $\mathcal{O}$-notation. I describe the QAOA algorithm, explaining its relation

to the Adiabatic Theorem and the Zassenhaus formula, through which I will later propose an improved variant of QAOA. Finally, I also study different configurations of QAOA, including the one I designed, k-interaction Angle QAOA (ka-QAOA), and the figures of merit used to evaluate the success of the QAOA method, comparing it to other configurations.

I present the respective results in **Section 4**. I study the Max-Cut problem using QAOA, as presented by the original QAOA paper [6], but also study the first higher order of the Zassenhaus formula. I then focus on the JSSP problem for QUBO and PUBO, also comparing both using various hypergraphs with a different number of vertices. I study the multiple configurations, including ka-QAOA and present conclusions on how it can reduce the number of function evaluations required by the VQA.

The convention adopted for indexes in this document is zero-based indexing, aligning with common practice in many programming environments. Therefore, sequences and arrays within these formulas will start with $x_0, x_1, x_2, \ldots$.

1.4 | Main Results

Results on the tests that I executed on various configurations and variants of QAOA show that while a solution may yield slightly better optimisation, it is not necessarily the better solution. The variant may require more resources, which might not be ideal for current quantum computer hardware. Results also show that while a variant may result in better optimisation for a particular problem, it is not necessarily a good solution for another optimisation problem.

In this dissertation, I show that k-interaction Angle QAOA (ka-QAOA) is an improved variant of QAOA, achieving high fidelity when compared to other variants, such as ma-QAOA, while requiring substantially fewer computational resources to compute the optimisation. As the complexity of the problem increases, ka-QAOA slightly reduces its approximation success compared to ma-QAOA. However, the resources required for computation of ka-QAOA remain low, which is ideal when the best possible approximation is not essential, especially given that the difference in approximation between other variants and ka-QAOA is minimal. This makes ka-QAOA a viable alternative in the current NISQ era, where the resources available to researchers and developers to execute quantum algorithms remain limited.

2

Background & Literature Overview

In a pivotal lecture at the 1981 *First Conference on the Physics of Computation*, Richard Feynman [7] highlighted the inadequacy of classical computers to simulate quantum phenomena, marking a seminal moment in the field. His quote, '*Nature is quantum, goddammit! So if we want to simulate it, we need a quantum computer*' captures the essential insight that propelled the development of quantum computing.

2.1 | Quantum Computing

In his 1985 paper, *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer* [8], physicist David Deutsch introduced the revolutionary concept of a quantum Turing machine. This paper was instrumental in establishing the principles that a quantum computer could potentially surpass classical computers in certain computational aspects, earning him recognition as the father of quantum computing.

2.1.1 | Quantum Bits

The fundamental data processing unit in classical computing is the *bit* [9], which exists in a 0 or 1 state. The equivalent unit in quantum computing is the *quantum bit*, or *qubit*. Benjamin Schumacher first coined the term qubit in 1995 [10]. While qubits also possess binary states, represented as $|0\rangle$ and $|1\rangle$, they exhibit additional states owing to the principle of *superposition* in quantum mechanics. A qubit can simultaneously occupy more than one state, as described by $|\psi\rangle$ in equation (2.1).

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

where:

α, β : complex coefficients that determine the probability amplitudes of the qubit's state

Section 2.2.1 discusses superposition in more detail.

2.1.2 | The Bloch Sphere

A visual interpretation of a qubit's state is often shown by the concept of the *Bloch sphere*, illustrated in Figure 2.1. Introduced by physicist Felix Bloch in 1946, the Bloch sphere provides a geometrical representation of the state space of a single qubit, mapping the complex coefficients α and β onto points on a sphere, as expressed in equation (2.2).

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{2.2}$$

Any qubit state on the Bloch sphere can be depicted as a point on the surface. The sphere poles correspond to the basis states $\sigma^z|0\rangle = |0\rangle$ and $\sigma^z|1\rangle = |1\rangle$, represented traditionally at the north and south poles, respectively. The other points on the sphere represent the superpositions of these basis states, with their positions determined by the values of α and β.

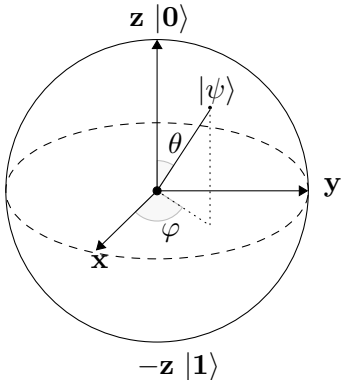


Figure 2.1: The Bloch Sphere

In the Bloch sphere representation, the polar angle, θ, (measured from the positive z-axis) and the azimuthal angle, φ, (measured in the x-y plane from the positive x-axis) correspond to the amplitudes and phases of the quantum state, respectively. This spherical representation is not just a visualisation tool; it's crucial for understanding qubit manipulation principles and quantum information processing dynamics.

2.2 | Qubit Properties

Qubits possess properties that distinguish them from classical bits. Unlike classical bits, which can represent 0 or 1 only, qubits can simultaneously represent a 0 **and** 1 value with varying probabilities. Additionally, unlike classical bits, qubits can exist in an entangled state, where the state of one qubit is intrinsically linked to the state of another.

2.2.1 | Superposition

A quantum state $|\phi\rangle$ is expressed as a linear combination of the orthonormal* basis vectors $|v_1\rangle$ and $|v_2\rangle$, such that $|\phi\rangle = c_1 |v_1\rangle + c_2 |v_2\rangle$, where c_1 and c_2 are complex coefficients, and $(|v_1\rangle, |v_2\rangle)$ forms an ordered orthonormal basis. This quantum state resides in a complex vector space known as a Hilbert space and is commonly referred to as a *state vector*. In the context of qubits, the standard computational basis is typically denoted as $|0\rangle$ and $|1\rangle$.

Due to the principle of *superposition* in quantum mechanics, a qubit can exist simultaneously in a combination of these basis states. This general quantum state is denoted by $|\psi\rangle$ and is defined in equation (2.1).

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

The state $|\psi\rangle$ represents a quantum state that exists in state 0 with a probability amplitude of α and in state 1 with a probability amplitude of β. Given that the total probability for a quantum system must sum to 1, the coefficients α and β in equation (2.1) must satisfy the normalisation condition given in equation (2.3).

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.3}$$

The concept of superposition is illustrated in Erwin Schrödinger's famous thought experiment, known as *Schrödinger's Cat* [11], demonstrating the idea of superposition. The bizarre nature of quantum phenomena is shown in this experiment, which also serves as a conceptual analysis clarifying superposition and quantum indeterminacy. In this case, the imaginary cat's life is entangled with a quantum superposition state, meaning the cat is simultaneously both alive and dead. Schrödinger designed this experiment to illustrate the challenges associated with applying quantum mechanics to large systems. In addition, this unique characteristic of qubits, which allows them to remain in several

*A set of vectors is *orthonormal* if each vector has unit length (i.e., is *normal*) and any pair of distinct vectors is orthogonal. Formally, $\langle v_i | v_j \rangle = 0$ for $i \neq j$, and $\langle v_i | v_i \rangle = 1$ for all i. Such a set forms an orthonormal basis for a Hilbert space.

possible superposition states, is a significant factor in the greater efficiency of quantum computers in computations.

2.2.2 | Entanglement

Quantum Entanglement, a term coined in 1935 by Erwin Schrödinger [12], refers to a phenomenon of a particle which is intrinsically correlated with another in such a way that both states affect each other even though they are far apart. Schrödinger was responding directly to another paper by Albert Einstein, Boris Podolsky, and Nathan Rosen, known as the EPR argument [13], who reasoned that quantum mechanics was incomplete. Einstein himself certainly did not have a high opinion of quantum entanglement. He even famously referred to it as *spooky action at a distance* because he could not fit it into the plan of a local theory with elements of reality.

This argument sparked many scientific advances relating to quantum mechanics. Subsequent research produced similar results each time and not only confirmed the existence of quantum entanglement but also underscored its critical importance in the broader context of quantum theory.

In quantum entangled systems, properties such as spin, position, or momentum of two entangled particles are correlated in a manner that defies classical explanation. However, after the state is determined by measuring the first particle, the state of the other particle is immediately determined, i.e. it *collapses*. This does not follow classical notions of locality and realism, but this has been confirmed experimentally [14].

A typical example of an entangled state in quantum mechanics is the set of Bell states, embodying a maximally entangled two-qubit system. The Bell states can be represented as shown in equation (2.4).

$$\begin{aligned} \left|\Phi^{+}\right\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \\ \left|\Phi^{-}\right\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \\ \left|\Psi^{+}\right\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right) \\ \left|\Psi^{-}\right\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right) \end{aligned} \tag{2.4}$$

where:

$\left\|\Phi^{+}\right\rangle, \left\|\Phi^{-}\right\rangle, \left\|\Psi^{+}\right\rangle, \left\|\Psi^{-}\right\rangle$	:	denotes a Bell state
$\left\|00\right\rangle, \left\|01\right\rangle, \left\|10\right\rangle, \left\|11\right\rangle$	:	symbolize the basis states of a two-qubit system in the σ^{z} basis

In these Bell states, equation (2.4), the first digit indicates the state of the first qubit, while the second digit corresponds to the state of the second qubit, with $|0\rangle$ and $|1\rangle$ representing the states of a single qubit. The coefficient $\frac{1}{\sqrt{2}}$ is a normalising factor, ensuring that the total probability of the system sums to one.

Entangled states are significant because each qubit's condition is interdependent. Knowing the state of one qubit immediately reveals the state of its counterpart, regardless of their physical separation. This demonstrates the principle of non-locality in quantum entanglement, challenging and extending our understanding of the natural world.

Entanglement is exploited in quantum algorithms such as Shor's algorithm for prime factorisation and Grover's algorithm for database search. These algorithms utilise entanglement to process data in ways unachievable by classical computing systems, allowing computations to be performed simultaneously on multiple interconnected qubits, leading to quantum parallelism.

2.2.3 | Measurement

Measurement in quantum mechanics is another concept distinct from classical measurement paradigms. Due to the superposition property (Section 2.2.1), the state of a qubit involves a probabilistic nature where outcomes are based on probabilities derived from the system's wave function. This wave function, representing a superposition of all possible states, *collapses* to a specific eigenstate of the measurement operator. The process, termed wave function collapse, is both instantaneous and irreversible. The outcome probabilities are determined by the absolute value squared of the wave function's amplitude in each possible state as per equation (2.1).

Moreover, quantum measurement gives rise to intriguing phenomena, such as the Heisenberg Uncertainty Principle, which states that certain pairs of properties of a quantum system, such as position and momentum, cannot be measured simultaneously with arbitrary precision. In entangled systems (Section 2.2.2), the measurement of one part can instantly influence the state of the other, a phenomenon that challenges classical notions of locality and causality.

When measuring a quantum state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, one actually measures an observable, with the measurement being performed in the computational basis, which corresponds to the eigenbasis of the Pauli-Z operator (σ^z). As a result, the state collapses to either $|0\rangle$ or $|1\rangle$, with probabilities of $|\alpha|^2$ to measure $|0\rangle$ or $|\beta|^2$ to measure $|1\rangle$. After the measurement, the state collapses to the corresponding outcome, with the measurement being the only *irreversible* operation in a quantum circuit, and the state is collapsed

to a classical outcome. In quantum circuits, the measurement of qubits is commonly represented by the symbol shown in Figure 2.2.

Figure 2.2: Circuit symbol for measuring a qubit

For example, in a simple random number generator circuit using qubits, the Hadamard gate transforms a qubit from a basis state (example $|0\rangle$) into an equal superposition of $|0\rangle$ and $|1\rangle$. Subsequent measurement then collapses each qubit to a random state of $|0\rangle$ or $|1\rangle$, effectively generating a random bit.

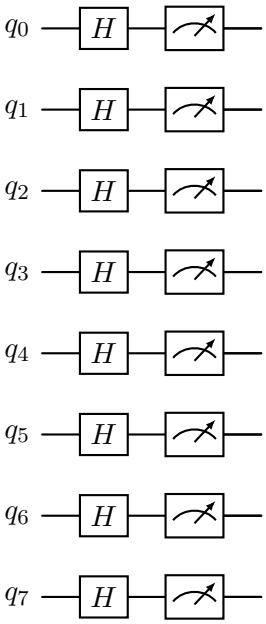


Figure 2.3: Eight-bit random number generator circuit

In the circuit in Figure 2.3, each qubit initially passes through a Hadamard gate (see Table 2.1), creating a state of uniform superposition state. Following this, a measurement is performed. The measurement results in either the state $|0\rangle$ or $|1\rangle$, each with a 50% probability, which translates to a classical bit value of 0 or 1, respectively. Consequently, all eight qubits independently produce a random output of classical bits. The overall circuit generates an 8-bit unsigned classical number, ranging from 0_{10} to 255_{10}. This constitutes a *true random number*, distinct from *pseudo-random numbers* generated by classical algorithms.

2.3 | Quantum Gates

Just as logic gates are the building blocks of algorithms in classical computing, so are quantum gates in quantum computing. Quantum algorithms are constructed using quantum gates acting on qubits and leveraging the principles of quantum mechanics, such as superposition (Section 2.2.1), entanglement (Section 2.2.2) and interference.

2.3.1 | Unitary Operations

When a quantum gate is applied to the quantum state $|\phi\rangle$, the state vector (Section 2.2.1) is transformed by multiplying it with a **unitary matrix** U corresponding to that gate, as shown in equation (2.5).

$$|\phi'\rangle = U\,|\phi\rangle \tag{2.5}$$

where:

$\|\phi\rangle$	:	the initial quantum state
$\|\phi'\rangle$	:	the state resulting from the application of the unitary operation
U	:	the Unitary Matrix or Operation

A unitary operation in quantum mechanics represents a mathematical operation corresponding to a rotation of the basis vectors. A unitary operation, described by matrix U, is unitary if its conjugate transpose $U^\dagger$ is also its inverse, i.e., $U^\dagger U = UU^\dagger = I$, where I is the identity matrix. This property ensures that the operation preserves the norm (or length) of any vector it acts upon in a complex vector space.

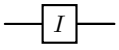
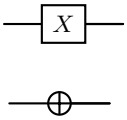
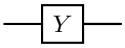
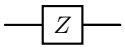
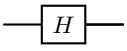
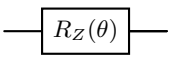
A unitary operation is also reversible. Since the inverse of a unitary matrix is always known, applying its inverse can undo any unitary operation. This property contrasts many classical operations, which can be irreversible (like AND and OR gates in classical computing). Simply put, information is not lost in quantum gates as can happen in classical gates, where certain operations might not be reversible. Non-reversible means that the original input cannot be derived from the output once performed. In contrast, the reversible nature of quantum gates allows for preserving and retrieving information from the quantum state.

Common examples of unitary operations, or quantum gates, which are defined in detail later in this section, include the Pauli gates (σ_x, σ_y, σ_z), the Hadamard gate (H), and the Controlled NOT (CNOT) gate. Unitary operations are the building blocks of quantum algorithms. However, while unitary operations are reversible and preserve information, they are distinct from quantum measurement, which is irreversible.

2.3.2 | Single-Qubit Gates

Single-qubit gates operate on individual qubits, forming the fundamental building blocks of quantum circuits. These gates focus on altering the state of a single qubit through operations like rotation and phase shifts. As in classical computing, these gates exploit quantum mechanical principles such as superposition, enabling the qubit to represent a combination of states rather than a fixed binary state. Single-qubit gates, Table 2.3.2, are essential for initialising and manipulating the states of qubits in a quantum system.

Table 2.1: Common Quantum Single-Qubit Gates and Their Properties

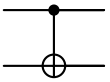
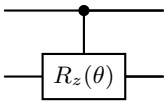
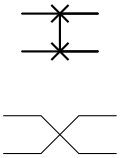
Gate	**Symbol**	**Matrix Representation**	**Effect on Qubits State**	**Circuit Representation**
Identity Gate	I	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	*Do nothing* gate: Leaves any qubit state unchanged	
Pauli-X Gate	X, σ_x, σ_1	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	Bit-flip gate: Flips $\|0\rangle$ to $\|1\rangle$ and vice versa	
Pauli-Y Gate	Y, σ_y, σ_2	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	Bit and phase-flip gate: Flips and adds a phase to the qubit state	
Pauli-Z Gate	Z, σ_z, σ_3	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	Phase-flip gate: Leaves $\|0\rangle$ unchanged, maps $\|1\rangle$ to $-\|1\rangle$	
Hadamard Gate	H	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	Superposition gate: Creates an equal superposition of $\|0\rangle$ and $\|1\rangle$	
Z Rotation Gate	$R_Z(\theta)$	$\begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$	Phase rotation gate: Rotates qubit around Z-axis by θ	

2.3.3 | Multi-Qubit Gates

Multi-qubit gates, Table 2.3.3, facilitate concurrent operations on multiple qubits, distinguishing them from the sequential nature of classical binary operations. This

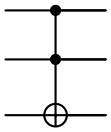
capability allows for more complex and efficient manipulation of quantum states than traditional computing paradigms by leveraging properties like superposition and entanglement. These properties *link* qubits' states to enable intricate computations. This interconnectedness is crucial for quantum algorithms, potentially accelerating problem-solving exponentially.

Table 2.2: Common Quantum Multi-Qubit Gates and Their Properties

Gate	**Symbol**	**Matrix Representation**	**Effect on Qubits State**	**Circuit Representation**
Controlled Pauli-X Gate	$CNOT$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	Conditional Flip Gate: Inverts the second qubit's state from $\|0\rangle$ to $\|1\rangle$ and vice versa, conditional on the first qubit being in the $\|1\rangle$ state.	
Controlled-Z Rotation Gate	$CRZ(\theta)$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{-i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$	Symmetric two-qubit, controlled phase rotation gate: applies a controlled phase shift to the two-qubit system, with the amount of phase shift determined by the angle θ, conditional on the first qubit being in the $\|1\rangle$ state.	
Swap Gate	$SWAP$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	State Exchange Gate: Swaps the states of the first and second qubits such that the state of the first qubit becomes the state of the second, and vice versa.	

Continued on next page

Table 2.2 – *continued from previous page*

Gate	Symbol	Matrix Representation	Effect on Qubits State	Circuit Representation
Toffoli Gate	$CCNOT$, CCX, $TOFF$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$	Conditional Flip Gate: Inverts the third qubit's state from $\|0\rangle$ to $\|1\rangle$ and vice versa, conditional on the first and second qubits being in the $\|1\rangle$ state.	

2.4 | Quantum Circuits

In classical computing, we connect several classical gates to form a circuit. Similarly, we can connect several quantum gates to form a quantum circuit. A quantum circuit would have a fixed number n of qubits, and we can represent it graphically using n horizontal parallel lines. Quantum gates would then be depicted on these lines. Computation starts with an initial state of the qubits, usually in the $|0\rangle$ state, and proceeds from left to right. Usually, at the end of each horizontal line, there would be a measurement symbol (Section 2.2.3), which collapses the quantum state into a classical state.

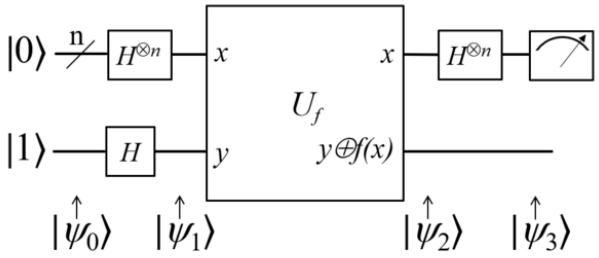


Figure 2.4: Deutsch Jozsa Algorithm Circuit
Credit: Wikimedia Commons

Quantum circuits allow us to build complex quantum algorithms. Figure 2.4 shows the Deutsch-Jozsa [15] circuit. This quantum algorithm is designed to solve a specific problem exponentially faster than any classical deterministic approach. Given a black box function (oracle) $f : \{0,1\}^n \to \{0,1\}$ that takes an n-bit binary input and produces a single-bit output (i.e. 0 or 1), the algorithm will determine if the function is *constant*, i.e. all outputs are the same (0 or 1) independent of the input, or if the function is *balanced*, i.e. producing 0 for half of the inputs and producing 1 for the other half.

2.4.1 | Noisy Intermediate-Scale Quantum (NISQ) Devices

In 2016, IBM released its 5-qubit cloud-based quantum computer, followed by 14-20 qubits in 2017 and 20-50 qubits in 2018. Then, in 2018, John Preskill [16] introduced the concept of Noisy Intermediate-Scale Quantum (NISQ) computers. Preskill [16] explains how 50-100 qubit quantum error-free computers are poised to surpass the capabilities of today's classical digital computers. While considering that a fault-tolerant quantum computer may still be a *distant dream*, Preskill [16] introduced the term NISQ: a quantum computer which is *intermediate-scale*, i.e. having a low number of qubits (from 50 qubits, a limit below which can be simulated by brute force algorithms on current supercomputers, to a few hundred qubits). These qubits are *noisy*, meaning that they are not perfect and easily affected by surrounding *noise*, resulting in possibly imperfect calculations.

There are two types of quantum computers. **Quantum Annealers** are a special quantum computing hardware technology designed to solve combinatorial optimisation problems. Quantum annealers operate by initially placing qubits into a superposition state for an initial Hamiltonian configuration for which its ground state is easy to prepare. This Hamiltonian evolves under the adiabatic principle, such that the initial Hamiltonian evolves into a complex Hamiltonian that encodes the cost function, ensuring that the system remains in its ground state. The ultimate purpose of such an evolutionary process is to reach an optimal or nearly optimal solution to the cost function of the problem through the exploitation of the adiabatic principle.

The other kind of quantum computer is **Gate-based Quantum Computers**. These quantum computers use quantum gates to build a quantum circuit and, therefore, create an algorithm based mainly on the rotation of qubits on the Bloch sphere to compute the algorithm. Quantum gates allow qubits to be placed together in a superposition and entangle qubits. The lifecycle of a qubit usually progresses from initialisation, for example, putting the qubit into a $|0\rangle$ state, a preparation phase, the most common being the Hadamard gate to put the qubit into an equal superposition of states, single or multiple qubit gates are then applied on the qubits' states, and finally, measurement is executed collapsing the qubit's quantum state into a classical value of 0 or 1.

The main target of researchers is to attain *quantum advantage* [17], a milestone for civilisation where a quantum computer proves to be more efficient in solving practical problems than classical computing. Due to their imperfect nature, it is unclear whether NISQ devices can attain this. Google claimed quantum advantage [18] [19], and so did The University of Science and Technology of China [20]. In 2019, Google claimed [18] that using their 54-qubit processor, they performed a task known as *random circuit sampling* in just 200 seconds. In contrast, they claimed that the task would take a classical supercomputer

approximately 10,000 years to perform, even though other entities renounced this last claim [21] [22], counter-claiming that the problem can be executed onto a classical computer in the region of two and a half days to a few dozen seconds.

2.4.2 | Circuit Depth

Noise is also an issue in classical computing; however, in quantum computers, noise can cause qubits to decohere, i.e. qubits lose their quantum state. Noise in quantum computers is caused by electronic interference, imperfections in the quantum hardware, and other unwanted interactions with the environment in which the physical qubits are embedded. Due to this fact, during the NISQ era, the complexity of quantum circuits is an essential consideration for physicists and developers. As a result of this, they aim to keep circuits at a *low-depth* (see Figure 2.5).

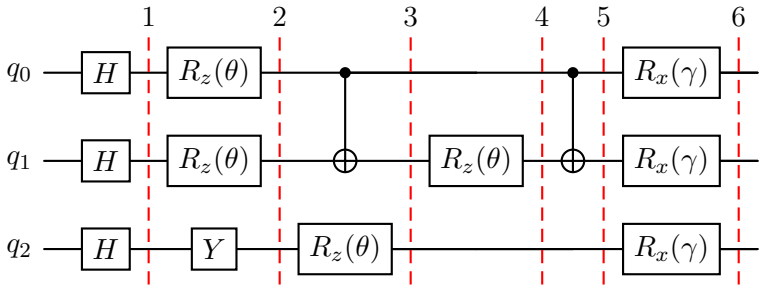


Figure 2.5: Sample quantum circuit with a depth of six.

A low-depth quantum circuit has the lowest number of sequential quantum gates acting on qubits, resulting in less noise affecting the qubits. Algorithms, such as QAOA (Section 3.5), are designed to keep their depth as low as possible to mitigate noise in NISQ devices.

2.5 | Physical Properties of Qubits

Qubits can be represented physically using different physical quantum systems, including electron spins, photon polarisation, trapped ions, superconducting circuits, quantum dots, and topological qubits.

2.5.1 | Spin

In classical mechanics, **spin angular momentum** can occur either when an object rotates on itself, i.e. on an axis, i.e. *rotational angular momentum*, or when it rotates around

another object, i.e. *orbital angular momentum*. If the object carries an electric charge, these movements cause a magnetic field. However, in the case of the rotational angular momentum, the object must have a dimension and mass distribution.

Quantum particles, such as electrons, do not have a dimension, but nonetheless, *magnetic properties* were detected in these particles. The source of this magnetic property is also called spin. Spin $\frac{1}{2}$ particles like electrons can point in one of two directions: *up* (spin $+\frac{1}{2}$) or *down* (spin $-\frac{1}{2}$), corresponding to the qubit states $|0\rangle$ and $|1\rangle$ respectively. The Stern-Gerlach experiment [23] (translated to English in [24]), illustrated in Figure 2.6, found that particles, such as electrons, showed a discrete pattern, rather than a continuous pattern, showing that the spin or the angular momentum of particles is quantised as opposed to the result that could be obtained by classical objects which show a continuous pattern.

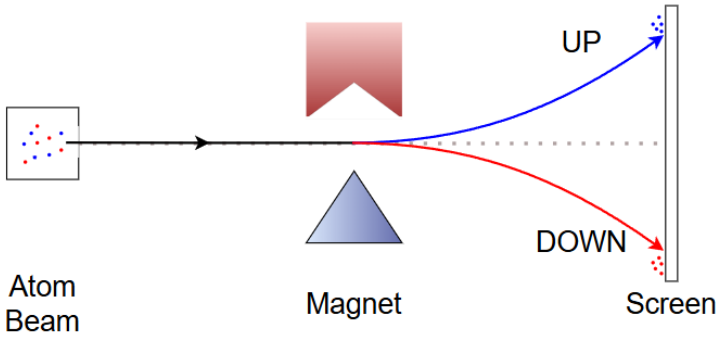


Figure 2.6: Stern-Gerlach Experiment showing the quantised property of the spin of quantum particles

The result is that a particle can exist in a superposition of two eigenstates: spin up $|\uparrow\rangle$ and spin down $|\downarrow\rangle$. Equation (2.6) therefore clearly represents the quantum state of the particle.

$$|\psi\rangle = \alpha\,|\uparrow\rangle + \beta\,|\downarrow\rangle \tag{2.6}$$

This data would not be enough to explain a system in classical mechanics. Since classical objects have a dimension, we need other parameters to define the state of objects. It is also essential to consider that rotations are intrinsically connected to the quantum mechanical properties of angular momentum. Since angular momentum is a vector quantity, the order of the sequence of applied rotations is essential as different orders may give different results; i.e. rotations do not commute.

The Schrödinger equation does not inherently predict spin in quantum mechanics. Therefore, scientists tried to add it to the respective equations. It was Paul Dirac [25],

who, while attempting to merge standard quantum mechanics and relativity, got, as part of his formula, a mathematical representation that acts like the properties of spin. While this is a valid mathematical explanation of spin, we still do not understand it physically. However, we know it is a property of quantum particles rather than an effect resulting from motion like that of classical objects.

2.5.2 | The Ising model

The Ising model, introduced by Ernst Ising in his 1924 thesis [26] (*English version* [27]) as an assignment given to him by physicist Wilhelm Lenz, examines a system's interaction behaviour represented by an n-dimensional lattice of points, each with a spin: spin-up $(+1)$ or spin-down (-1). Each particle's spin interacts only with its neighbours; the spin interaction happens only locally; it cannot interact with a spin that is not its neighbour. A spin may also interact with an external source. Equation (2.7) describes the classical classical Ising model.

$$H(\sigma) = -\sum_{\langle i,j \rangle} J_{ij}\sigma_i\sigma_j - \sum_j h_j\sigma_j \tag{2.7}$$

where:

$\langle i,j \rangle$	:	a pair of neighbouring spins
J_{ij}	:	the energy interactions between spin i and spin j
σ_i, σ_j	$\in$	$\{-1, +1\}$, the spin of i or j
h_j	:	the energy effecting spin j from an external source

The first part of the Hamiltonian in equation (2.7) $(-\sum_{\langle i,j \rangle} J_{ij}\sigma_i\sigma_j)$ represents an internal interaction between neighbouring spins, while the second part of the Hamiltonian $(-\sum_j h_j\sigma_j)$ represents an external source which interacts with the individual spin on the lattice.

Later, we will use the Ising model in quantum circuits, as optimisation problems can often be mapped onto Ising Hamiltonians.

2.5.3 | The Adiabatic Theorem

The Adiabatic Theorem is a concept in quantum mechanics rooted in classical thermodynamics that describes a process that evolves slowly and remains in its eigenstate. The Adiabatic theorem states that if a Hamiltonian $\hat{H}(t)$ starts from $\hat{H}_I$ at $t = 0$ evolving to $\hat{H}_F$ over a sufficiently long evolution time, then provided that the system starts in its

ground state of $\hat{H}_I$, during this Hamiltonian's evolution, the system is expected to remain in its instantaneous ground state. Born and Fock [28] stated that

> *A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum.*

In quantum computing, the **Adiabatic Quantum Computing (AQC)** paradigm leverages the principles of the Adiabatic Theorem by encoding a system, $\hat{H}(t)$, such that the ground state of a problem Hamiltonian, $\hat{H}_C$, is reached by initializing the system in the known ground state of another Hamiltonian, $\hat{H}_I$, and then slowly evolving $\hat{H}(t)$ from $\hat{H}_I$ to $\hat{H}_C$. As a result of the Adiabatic Theorem, the system will remain in its instantaneous ground state throughout the evolution and thus reach the ground state of the problem Hamiltonian, as defined in equation (2.8).

$$\hat{H}(t) = s(t)\hat{H}_I + (1 - s(t))\hat{H}_C \tag{2.8}$$

where:

$\hat{H}(t)$	:	the encoded system Hamiltonian
$s(t)$	:	a function which varies time t sufficiently slow from 1 to 0 over time T, s.t. $s(0) = 1$ and $s(T) = 0$
$\hat{H}_I$	:	the initial Hamiltonian with a known ground state
$\hat{H}_C$	:	the Cost, or problem, Hamiltonian for which we are looking for its ground state

AQC has various applications and is used in annealing quantum computers. Nonetheless, gate-based quantum computers can simulate annealing with algorithms such as QAOA (Section 3.5).

3

Quantum Algorithms for Combinatorial Optimisation

Optimisation problems involve finding the best solution from a solution space, typically by minimising or maximising an objective function which is subject to a number of constraints. These problems are prevalent across a wide range of real-world applications such as the JSSP, Financial Portfolio Optimisation, Resource Allocation, Network Optimisation, Traffic Management, Airline Scheduling, Material Science, Climate Modelling, Fluid Dynamics, and many others.

A significant subfield of optimisation is combinatorial optimisation, where the decision variables are discrete rather than continuous. These problems focus on selecting a solution from a finite, albeit often huge, solution space with several candidate solutions. Combinatorial optimisation algorithms usually traverse the solution space to identify the best outcomes.

3.1 | Problem Representation

In computer science, to solve problems, computer scientists use mathematical modelling to define a problem in a structured manner. To solve the problem, it must be modelled according to the available hardware. The problem model must include definitions for the process from input and output definitions, problem constraints and the respective objective function.

3.1.1 | Boolean Representation

Problems can be represented using logical expressions in binary values, namely 0 and 1. This classical mathematical representation is utilised in digital computing, or as we

are referring to it, *classical computing*, as it aligns with the binary nature of electronic circuitry. An example of a Boolean problem representation is the Half-Adder, shown in equation (3.1) and illustrated in Figure 3.1. It is a digital circuit that adds two single-bit binary numbers and outputs the sum and the carry bit.

$$\begin{aligned} S &= A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B) \\ C &= A \wedge B \end{aligned} \tag{3.1}$$

where:

A, B	:	the input binary variables
S	:	sum
C	:	carry

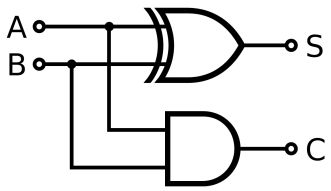


Figure 3.1: Half-Adder circuit. The top gate is an XOR gate, and the bottom is an AND gate in classical computing.

Credit: Wikimedia Commons

3.1.2 | Spin Representation

Problems can also be represented using spin-$\frac{1}{2}$ variables, commonly used to model qubits. These variables take on spin-up ($|\uparrow\rangle, +1$) or spin-down ($|\downarrow\rangle, -1$) states (Section 2.5.1). Equation (2.6) is an example of such a representation. The Up/Down states ($|\uparrow\rangle$, $|\downarrow\rangle$) are aligned along the z-axis of the Bloch Sphere (Section 2.1.2). Other examples of spin representation are the plus state ($|+\rangle$) and the minus state ($|-\rangle$), which are aligned along the x-axis.

$$
\begin{aligned}
|\uparrow\rangle &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\
|\downarrow\rangle &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\
|+\rangle &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad = \frac{1}{\sqrt{2}} \left(|\uparrow\rangle + |\downarrow\rangle \right) \\
|-\rangle &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad = \frac{1}{\sqrt{2}} \left(|\uparrow\rangle - |\downarrow\rangle \right)
\end{aligned}
\tag{3.2}
$$

This representation, equation 3.2, is essential in Quantum Computing, as quantum circuits can be built when a problem is in its spin representation. Mathematically, a spin can be operated upon using Pauli Matrices operators (Table 2.1), by which also spin-spin interactions can be modelled.

3.2 | Graph Theory

Graph Theory is the study of *mathematical structures* containing vertices, or nodes, and the interaction of these vertices through edges joining them. Graph Theory has its roots in 1735, when Swiss-German mathematician and physicist Leonhard Euler published a paper with a solution to *the Königsberg bridge problem* (an English translation of his paper with detailed commentary can be found in [29]). The city of Königsberg (Figure 3.2), now known as Kaliningrad, in Russia, consisted of two large islands (Kneiphof and Lomse) separated by the river Pregel flowing around them and which were connected to the mainland with seven bridges (of which five remain today). According to historical accounts, residents challenged themselves to create a path that crossed all seven bridges only once. After their attempts consistently failed, Euler's paper proved that the task was impossible from a mathematical perspective. While a brute-force solution is possible, as Euler himself noted, it is highly laborious. Anticipating the emergence of more complex, similar problems, Euler laid the foundation for a new branch of mathematics, now known as Graph Theory. In Graph Theory, a graph G is an ordered pair, as described by equation (3.3).

$$
G = (V, E) \tag{3.3}
$$

where:

V	:	a set of vertices (or nodes)
E	:	a set of edges that links or associates two vertices s.t. $E \subseteq \{x, y\}$, having $x, y \in V$

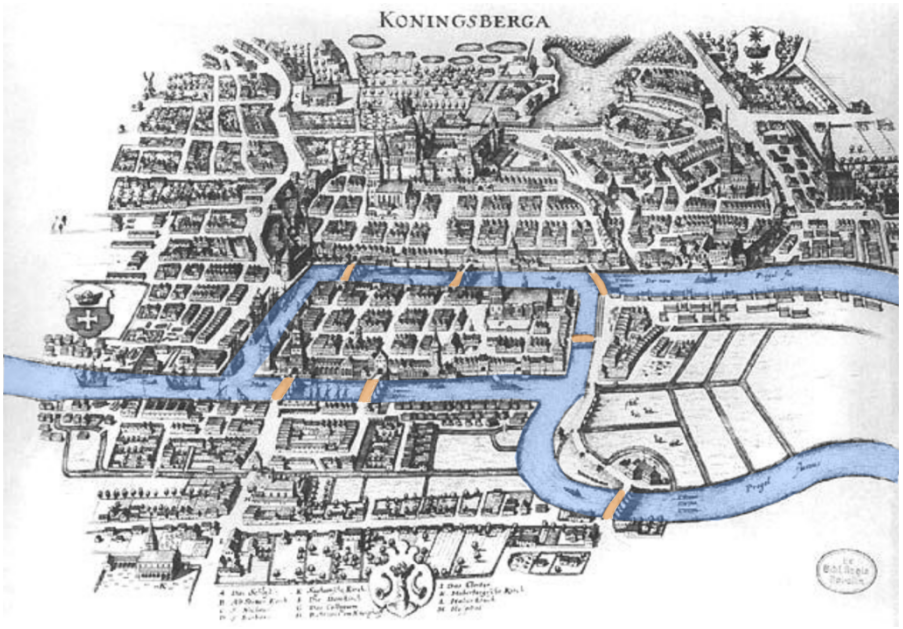


Figure 3.2: A map of Königsberg from Euler's days that displays the locations of the seven bridges

Credit: Wikimedia Commons

Euler represented landmasses using vertices and bridges using edges, as illustrated by Figure 3.3. He explained the Königsberg Bridge problem using the concept of vertex degrees*. Euler defined a **path** (known nowadays as *Eulerian path/trail*), which is a path that allows one to visit every edge exactly once and allows a revisit of vertices. Similarly, a **cycle** (known nowadays as *Eulerian cycle/circuit*) is a path which starts from a vertex, visits every edge exactly once and ends up in the initial vertex. Euler proved that the condition for an Eulerian cycle is that all vertices need an even degree (in the Königsberg Bridge problem, all vertices have an odd degree, node A in Figure 3.3 has a degree of 5. In contrast, all the other vertices have a degree of 3).

Since 1735, graph theory has evolved, and a number of graphs have been defined. These include but are not limited to, **directed graphs**, where edges are directed from one vertex to another; **weighted graphs**, in which a weight is given on the edge; and

*Vertex degree is the number of edges incident to the vertex

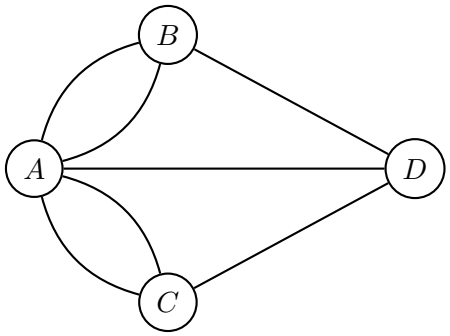


Figure 3.3: Graph representing the Seven Königsberg Bridge problem.

hypergraphs (Section 3.2.1), where an edge can join any number of vertices rather than just two vertices.

There are also some graphs named based on structure. Some of these include the **path graph** which is a graph which forms a single Euler Path; the **cycle graph** is a graph which forms a single Euler cycle; the **star graph** is a graph where one central vertex is connected to all other vertices which are not connected; and the **bipartite graph** is a graph whose vertices can be divided into two separate disjoint sets such that a vertex joins no two vertices in the same set.

Several graphs are named based on the degree of their vertices. These include **regular graphs** where each vertex has the same degree; **k-regular graphs** where each vertex has a degree k, where $k \in \mathbb{N}^+$; and a special case of the k-regular graphs are the **Cubic Graphs** where $k = 3$.

Graph Theory is used in real-world applications across different fields, including social networks, computer networks, data structures, artificial intelligence, genomics, molecular chemistry, supply chain management, airline networks, telecommunication networks, mobile networks, circuit design, quantum computing, sociology, combinatorics, and topology.

3.2.1 | Hypergraphs

A **hypergraph** is a graph generalisation where an edge, which, in the case of hypergraphs, is called a hyperedge, can join any number of vertices rather than just two vertices. A hypergraph H is an ordered pair, as shown in equation (3.4), with an example illustrated in Figure 3.4.

$$H = (V, E) \tag{3.4}$$

where:

V	:	a set of vertices (or nodes)
E	:	a set of hyperedges, where each hyperedge is a subset of V, such that $E \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$, where $\mathcal{P}(V)$ is a set of vertices member in V and excludes the empty set.

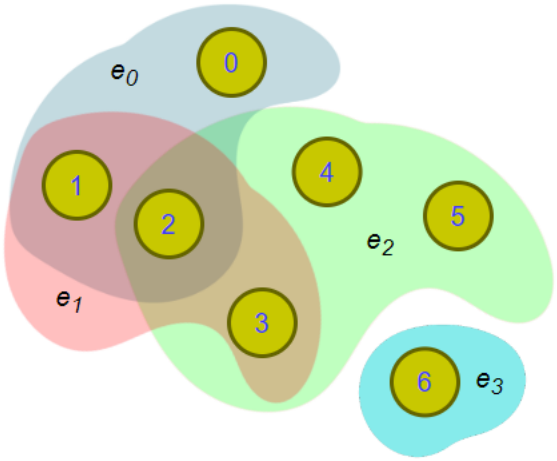


Figure 3.4: Hypergraph visual representation where numbered spheres represent the vertices and the coloured regions represent the hyperedges.

Many real-world applications require complex representations, and hypergraphs can model situations where more than two entities are involved in the same interaction. For example, a social network might use a hypergraph to model a group of people having a single interaction, whereas a traditional graph can only model interactions between two individuals.

3.2.2 | Automorphisms

By definition, two *simple* graphs G and H are considered to be **isomorphic** if there is a bijection[§] $\mathcal{O} : V(G) \to V(H)$ that preserves adjacency[¶] and non-adjacency[‖]. This means that for any two vertices u and v in G, the following condition, equation (3.5), holds.

$$u, v \in E(G) \iff \mathcal{O}(u), \mathcal{O}(v) \in E(H) \tag{3.5}$$

[§]A bijection is a mapping between two sets that is both one-to-one (injective, i.e. for different inputs we have different outputs but no duplicates) and onto (surjective, i.e. every possible output hs at least one input mapping into it), ensuring a perfect pairing between the elements of both sets.

[¶]Two vertices are adjacent if they are directly connected by an edge in a graph.

[‖]Two vertices are non-adjacent if there is no edge directly connecting them in the graph.

where:

$V(G)$	:	the vertex set of G
$V(H)$	:	the vertex set of H
$E(G)$	:	the edge set of G
$E(H)$	:	the edge set of H

In this case, we can use the notation $G \cong H$, meaning that graph G and graph H are isomorphic.

An **automorphism** of a graph is an isomorphism from the graph to itself. The main reason for the automorphism of graphs is that automorphisms describe the symmetries in graphs.

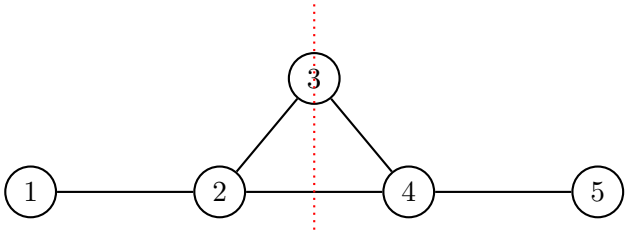


Figure 3.5: Sample graph with two automorphisms.

The red dotted line in the graph depicted by Figure 3.5 has been drawn to show the symmetry of the graph. The automorphisms of this graph are as follows:

$$\alpha = (3)(24)(15)$$
$$\epsilon = (1)(2)(3)(4)(5)$$

where:

α	:	an automorphism map across the line of symmetry shown by the red line; in this case (3) is mapped to itself, (24) means that it swaps (2) and (4), while (15) means that it swaps (1) and (5)
ϵ	:	the identity map; in this case each vertex v is mapped to itself

Therefore $Aut(G) = \{\epsilon, \alpha\} \cong \mathbb{Z}_2$

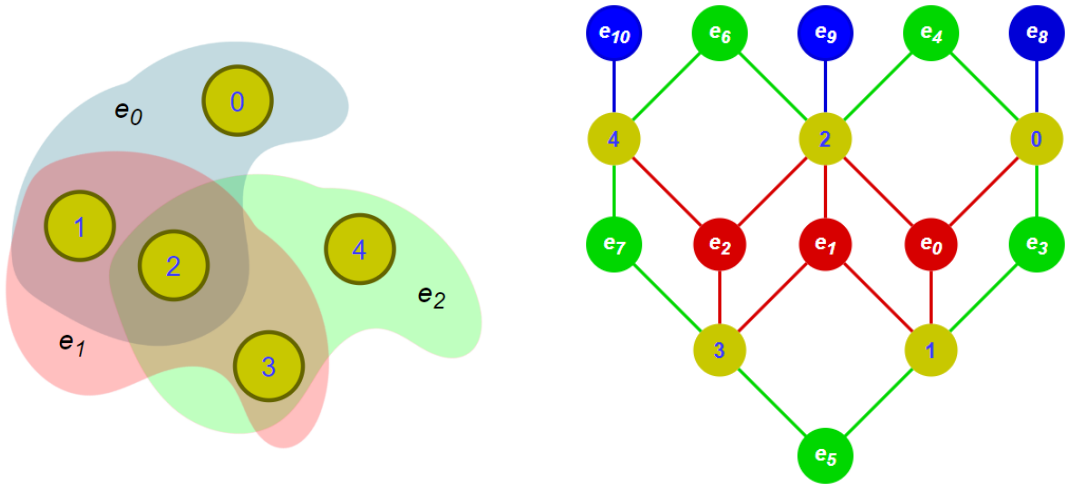
We will use Automorphisms of graphs in Section 4 to reduce the number of iterations of QAOA (Section 3.5).

3.2.3 | Automorphisms of Hypergraphs

Due to their complexity, extracting automorphisms of hypergraphs is a more laborious task than extracting automorphisms for *normal* graphs. One way to do this is to transform a hypergraph $\mathcal{H} = (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H})$ into a *normal* graph using the hypergraph's bipartite incidence graph $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$. The process follows:

1. Add to vertices $\{\mathcal{V}_\mathcal{G}\}$ of graph $\mathcal{G}$ all vertices $\{\mathcal{V}_\mathcal{H}\}$ of hypergraph $\mathcal{H}$
2. Add all edges $\{\mathcal{E}_\mathcal{H}\}$ of hypergraph $\mathcal{H}$ into the vertices $\{\mathcal{V}_\mathcal{G}\}$ of graph $\mathcal{G}$
3. Join vertices in graph $\mathcal{G}$ using edges $\{\mathcal{E}_\mathcal{G}\}$ by linking each vertex, original in $\{\mathcal{V}_\mathcal{H}\}$, to each vertex which in hypergraph $\{\mathcal{H}\}$ was originally an edge $\in \{\mathcal{E}_\mathcal{H}\}$
4. Extract the automorphism of the graph of incidence graph $\mathcal{G}$
5. Map the automorphism of the incidence graph $\mathcal{G}$ back to hypergraph $\mathcal{H}$ and thus get the automorphism of the hypergraph.

An example of the procedure is illustrated in Figure 3.6.



(a) 5 vertex hypergraph (b) Bipartite incidence graph

Figure 3.6: 5-vertex hypergraph and its related bipartite incidence graph.

3.3 | Combinatorial Optimisation

Combinatorial optimisation is a mathematical tool of discrete mathematics that can help to locate the optimal object in a finite set of objects. An example of a real-world combinatorial optimisation application is the Travelling Salesman Problem (TSP), which

determines the shortest possible route that visits a set of destinations on a map exactly once and returns to the starting point. The application of TSP is cardinal for services that entertain deliveries of goods, as such a tool can allow tracking vast distances and help save valuable time. In addition, combinatorial optimisation can be used to optimise stores with the best placement of goods and make recommendations in manufacturing by enhancing production planning.

TSP asks: *given a number of cities and the distance between each city, what is the shortest distance that one can take to follow the shortest path with the last stop being the city that one started in.* Menger [30] (*English* [31]) writes that in a set of $\mathcal{S} = (s_1, s_2, s_3, \ldots, s_n)$, one needs to find a permutation $\mathcal{P} = (p_1, p_2, p_3, \ldots, p_n)$, $n \in \mathbb{Z}$ and $1 \leq n \leq N$, that minimises the total distance travelled. This distance is represented as the sum shown in equation (3.6).

$$a_{i_1 i_2} + a_{i_2 i_3} + a_{i_3 i_4} + \cdots + a_{i_n i_1} \tag{3.6}$$

where:

a_{ij} $\in$ $\mathbb{R}$, representing the distances between cities i and j

The total possible permutations are $(n-1)!$, and the task is to find an efficient algorithm to identify the permutation yielding the minimum total distance.

3.3.1 | Quadratic Unconstrained Binary Optimisation (QUBO)

Quadratic Unconstrained Binary Optimisation (QUBO) is one of the main problems in combinatorial optimisation, which has applications in various practical fields such as finance, manufacturing, and machine learning. QUBO aims to find an optimal solution, using the minimum or min() function, for a quadratic polynomial using binary variables. The form of the QUBO formula is shown in equation (3.7).

$$\min \left(\sum_{i<j}^{n} Q_{ij} x_i x_j + \sum_{i=1}^{n} q_i x_i + c \right) \tag{3.7}$$

where:

Q_{ij}	:	coefficient of the quadratic terms
q_i	:	coefficient of the linear terms
x_i, x_j	:	binary variables
c	:	a constant term

Depending on the problem, one may not need to find the min() function but rather the max() function. This can be done by finding the minimum of the negative QUBO formula. These problems are defined in boolean variables, and their complexity may be such that the problem is Non-Polynomial (NP)-hard and that may scale exponentially as the number of variables increases. Real-world examples of QUBO problems are discussed in Section 3.3.9.

QUBO problems provide a formulation inherently tied to graph structures (see Section 3.2), involving binary variables and the pairwise interactions between them. QUBO problems can be represented by weighted graphs, where the nodes of the graph represent the binary variables and the edges reflect the interactions between them. In equation (3.7), each x_n corresponds to a node in the graph, the quadratic term interaction $Q_{ij}x_ix_j$ in the QUBO objective function corresponds to an edge in the graph between nodes i and j with weight Q_{ij}. The non-quadratic term q_ix_i refers to a self-loop on a single node x_i with a specific bias q_i. The constant c has no particular representation in the graph; it simply shifts the objective function by a constant amount.

3.3.2 | QUBO and the Ising model

QUBO is an expression containing binary operations taking the value of $+1$ for **true** or 0 for **false**. On the other hand, the Ising model (Section 2.5.2) is a lattice of spins, each of which can take the value of $+1$ for spin-up and -1 for spin-down. To process QUBO on a quantum computer, we cannot use bits (binary operations), binary operators or gates, but we need to use qubits (spin operations) and quantum gates. We represent a spin using the Pauli-Z operator σ_z.

The relationship between the classical bitwise value x and the quantum Pauli-Z operator σ_z can be expressed as equation (3.8).

$$x = \frac{\mathbb{1} - Z}{2} \tag{3.8}$$

This transformation in equation (3.8) aligns the Z eigenvalues with binary values. It maps $Z = 1$ (which corresponds to $|1\rangle$ in the computational basis) to bitwise $x = 0$ and $Z = -1$ (which corresponds to $|0\rangle$ in the computational basis) to bitwise $x = 1$. For example, let's assume that our cost function is in the form of:

$$C = x_0x_1 \tag{3.9}$$

where:

x_i = a classical bit

This cost function, equation (3.9), can now be represented using σ_z operators in a quantum circuit representation. It can be transformed into a QUBO Ising problem as shown in equation (3.10).

$$C = \left(\frac{\mathbb{1} - Z_0}{2}\right)\left(\frac{\mathbb{1} - Z_1}{2}\right) \tag{3.10}$$

The expanded QUBO Ising representation is therefore shown in equation (3.11).

$$C = \frac{1}{4}\left(\mathbb{1} - Z_0 - Z_1 + (Z_0 Z_1)\right) \tag{3.11}$$

where:

$\mathbb{1}$	=	the identity operator
Z_i	=	σ_z operator acting on the i-th qubit

The terms involving products of Z_i represent interactions between different qubits. This form suits quantum circuit implementations or quantum computing algorithms to solve QUBO problems. Figure 3.7 shows a partial quantum circuit corresponding to the cost function given in equation (3.11).

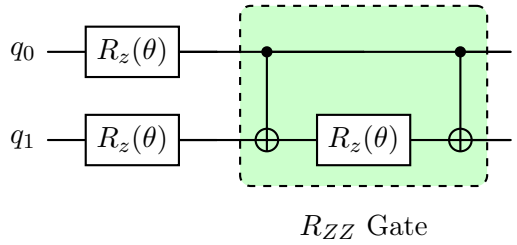


Figure 3.7: Partial quantum circuit for the simple QUBO Cost function in equation (3.11). The green area shows a Rotational ZZ gate ($R_{zz}(\theta)$), while white gates show Rotational Z gates ($R_z(\theta)$) (Section 2.3.2).

The Rotational ZZ gate $R_{ZZ}(\theta)$ is defined as:

$$\begin{aligned} R_{ZZ}(\theta) &= e^{-i\frac{\theta}{2}(Z\otimes Z)} \\ &= \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 & 0 & 0 \\ 0 & e^{i\frac{\theta}{2}} & 0 & 0 \\ 0 & 0 & e^{i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\theta}{2}} \end{bmatrix} \end{aligned} \tag{3.12}$$

3.3.3 | Polynomial Unconstrained Binary Optimisation (PUBO)

While some problems can be represented as QUBO, i.e. **Quadratic** Unconstrained Binary Optimisation problems, most problems in real life are PUBO, i.e. **Polynomial** Unconstrained Binary Optimisation problems, shown in equation (3.13). This is because the expressions resulting from these problems need interactions of more than two variables, making them polynomials.

$$\min \left(\sum_{p=1}^{m} \sum_{i_1<i_2<\cdots<i_p}^{n} Q_{i_1 i_2 \ldots i_p} \prod_{r=1}^{p} x_{i_r} + c \right) \tag{3.13}$$

where:

p	:	the degree of interaction, for example, linear, quadratic, cubic, etc.
$Q_{i_1 i_2 \ldots i_p}$	:	coefficient for each interaction term
x_{i_r}	:	binary variables
c	:	a constant term

The following example shows Cubic Polynomial Unconstrained Binary Optimisation (CUBO).

$$\min \left(\sum_{i<j<k}^{n} Q_{ijk} x_i x_j x_k + \sum_{i<j}^{n} Q_{ij} x_i x_j + \sum_{i=1}^{n} q_i x_i + c \right) \tag{3.14}$$

where:

Q_{ijk}	:	coefficient of the cubic terms
Q_{ij}	:	coefficient of the quadratic terms
q_i	:	coefficient of the linear terms
x_i, x_j, x_k	:	binary variables
c	:	a constant term

Similarly to QUBO problems, PUBO problems represent a formulation directly linked to hypergraphs (Section 3.2.1). PUBO problems can be represented by weighted hypergraphs, where the nodes of the graph represent the binary variables and the edges or hyperedges reflect the interactions between them.

3.3.4 | PUBO and the Ising model

Similar to how QUBO can be reformulated as an Ising model, which corresponds to a spin-$\frac{1}{2}$ Hamiltonian (Section 3.3.2), we can also similarly model a PUBO problem. Let's take, for example, a simple cubic cost function in the form of:

$$C = x_0 x_1 x_2 \tag{3.15}$$

where:

x_i = a classical bit

The given cost function can be represented using σ_z operators in a quantum circuit representation. It can be transformed into a PUBO problem as follows:

$$C = \left(\frac{\mathbb{1} - Z_0}{2}\right)\left(\frac{\mathbb{1} - Z_1}{2}\right)\left(\frac{\mathbb{1} - Z_2}{2}\right) \tag{3.16}$$

The expanded PUBO Representation is therefore:

$$C = \frac{1}{8}\left(\mathbb{1} - Z_0 - Z_1 - Z_2 + (Z_0 Z_1) + (Z_1 Z_2) + (Z_2 Z_0) - (Z_0 Z_1 Z_2)\right) \tag{3.17}$$

where:

$\mathbb{1}$ = the identity operator

Z_i = σ_z operator acting on the i-th qubit

The terms involving products of Z_i represent interactions between different qubits. This form suits quantum circuit implementations or quantum computing algorithms to solve PUBO problems. Figure 3.8 shows a partial quantum circuit corresponding to the cost function given in equation (3.17).

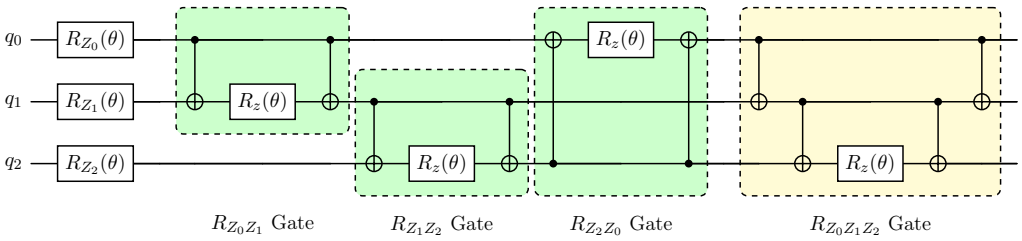


Figure 3.8: Partial quantum circuit for the simple PUBO cost function in equation (3.17), showing R_{zz} gates in the green areas and a R_{zzz} gate in the yellow area.

3.3.5 | Representing PUBO with QUBO

The PUBO formulation can be mapped into a QUBO problem [32] [33]. Before translating PUBO from binary to its spin representation, the procedure requires adding ancillary

bits to the input space and introducing a penalty term to the cost function, as shown in equation (3.18).

$$C(x) = \mathcal{J}x_0x_1x_2$$
$$C(Z) = \mathcal{J}\mathfrak{a}_0x_2 + 2Q(x_0x_1 - 2(x_0x_1 - 2\mathfrak{a}_0(x_0 + x_1) + 3\mathfrak{a}_0) \tag{3.18}$$

where:

$\mathcal{J}$	:	coefficient of the polynomial term, which in this case is a cubic term
x_i	:	the binary bits which form the PUBO binary function
$\mathfrak{a}_j$	:	an ancillary bit introduced to represent the PUBO by a QUBO formulation; if there are more than three terms, there will be more than one ancillary variable

The Python package QUBOvert [34] provides the functionality to transform a PUBO into a QUBO and introduce the necessary ancillary bits.

Stein et al. [35] study PUBO problems and their results indicate that PUBO formulations will outperform their QUBO representation when a problem is polynomial. The final goal of quantum computing is for multi-qubit hardware gates to be available to process PUBO more efficiently. Since QUBO formulations in a quantum circuit require the addition of ancillary qubits, the simulation models show a strong reduction in speed and the need for more memory since qubits are resource-hungry on classical computers. The authors believe that there is a need for more study on NISQ hardware of PUBO formulations and their QUBO representation.

3.3.6 | Number of Qubits Required in PUBO to QUBO mapping

Let's assume that we have a PUBO problem with three variables (cubic expression, or CUBO), then the maximum number of terms that we can get for a cost function that is similar to equation (3.19).

$$\sum_{i=0,i<j<k}^{n-1} \mathcal{J}_{ijk}\ x_i\ x_j\ x_k \tag{3.19}$$

where:

n	:	the number of variables in the PUBO expression
$\mathcal{J}$	:	a specific constant for that particular term
x	:	a boolean value

Without loss of generality, we will assume that $\mathcal{J} = 1\ \forall i, j, k$.

Since the aim is to transform the given PUBO problem into a QUBO problem, this requires the introduction of ancillary variables for its representation. Each term in equation (3.19) consists of three variables per term in the PUBO (CUBO) format, and the objective is to convert the expression into its QUBO analogue, i.e. having two variables per term. Here, I employ a strategy to reduce the number of combinations required for representation, thereby reducing the number of ancillary variables. Specifically, first, the combinations for half of the total variables ($\frac{n}{2}$) are considered, and then doubling the resulting number. Since there might be cases where n is an odd value, one can adapt this approach by considering the sum of the combinations of $\lceil \frac{n}{2} \rceil$, denoting the mathematical ceiling of the result, and $\lfloor \frac{n}{2} \rfloor$, denoting the mathematical floor of the result, to ensure an accurate transformation, as shown in equation (3.20).

$$C\left(\left\lceil \frac{n}{2} \right\rceil, 2\right) + C\left(\left\lfloor \frac{n}{2} \right\rfloor, 2\right) \tag{3.20}$$

where:

$C(n,k)$	=	binomial coefficient, i.e. $\left(\frac{n!}{k!(n-k)!}\right)$
n	=	the number of variables in the PUBO expression
k	=	the maximum number of variables that we want in our equation (i.e. 2)

For an even number of variables, the formula is thus simplified as follows:

$$= \left(\frac{\left(\frac{n}{2}\right)!}{2!\left(\frac{n}{2}-2\right)!}\right) + \left(\frac{\left(\frac{n}{2}\right)!}{2!\left(\frac{n}{2}-2\right)!}\right) = 2\left(\frac{\left(\frac{n}{2}\right)\left(\frac{n}{2}-1\right)}{2}\right) = \frac{n(n-2)}{4}$$
$$= \frac{n^2-2n}{4} \tag{3.21}$$

For an odd number of variables, the formula is therefore simplified as follows:

$$= \left(\frac{\left(\frac{n-1}{2}\right)!}{2!\left(\frac{n-1}{2}-2\right)!}\right) + \left(\frac{\left(\frac{n+1}{2}\right)!}{2!\left(\frac{n+1}{2}-2\right)!}\right) = \left(\frac{\left(\frac{n-1}{2}\right)\left(\frac{n-1}{2}-1\right)}{2}\right) + \left(\frac{\left(\frac{n+1}{2}\right)\left(\frac{n+1}{2}-1\right)}{2}\right)$$
$$= \frac{1}{8}((n-1)(n-3) + (n+1)(n-1)) = \frac{(n-1)^2}{4}$$
$$= \frac{n^2-2n+1}{4} \tag{3.22}$$

Therefore, I derive that the minimum number of ancillary variables required by the system represented by equation (3.20) can be represented through the equation (3.23).

$$\left\lceil \frac{n(n-2)}{4} \right\rceil \tag{3.23}$$

When similar CUBO expressions exist but not all terms are present, there might be even fewer ancillary variables.

3.3.7 | Pairs in the QUBO cost function

When converting a cost function from PUBO to QUBO, the resultant expression will contain both one-boolean variable terms and two-boolean variable terms. Subsequently, the algorithm is designed to translate these boolean variables into rotational Pauli-Z (σ^z) operators integrated into our quantum circuit.

Equation (3.24) represents a CUBO cost function, derived by applying equation (3.19) with $n = 3$. To convert this binary objective function into a spin-based cost function, the transformation in equation (3.8) is applied.

$$C(x) = x_0x_1x_2$$

$$C(Z) = \frac{1}{8}\left(\mathbb{1} - Z_0 - Z_1 - Z_2 + (Z_0Z_1) + (Z_1Z_2) + (Z_2Z_0) - (Z_0Z_1Z_2)\right) \tag{3.24}$$

. This results in the 3-qubit circuit, illustrated in Figure 3.9.

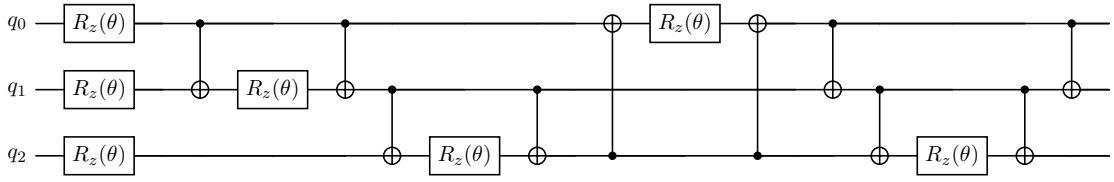


Figure 3.9: Partial quantum circuit for the PUBO Cost function (n=3) as described in equation (3.24).

Translating the CUBO cost function from equation (3.24) into a QUBO cost function using equation (3.18) yields the cost function shown in equation (3.25).

$$C(x) = 6x_3 + 2x_0x_1 - 4x_0x_3 - 4x_1x_3 + x_2x_3$$

$$C(Z) = 1\frac{3}{4} + \frac{1}{2}Z_0 + \frac{1}{2}Z_1 - \frac{1}{4}Z_2 - 1\frac{1}{4}Z_3 + \frac{1}{2}Z_0Z_1 - Z_0Z_3 - Z_1Z_3 + \frac{1}{4}Z_2Z_3 \tag{3.25}$$

This results in the 4-qubit circuit illustrated in Figure 3.10, with an additional qubit introduced due to the transformation requiring a single ancillary qubit.

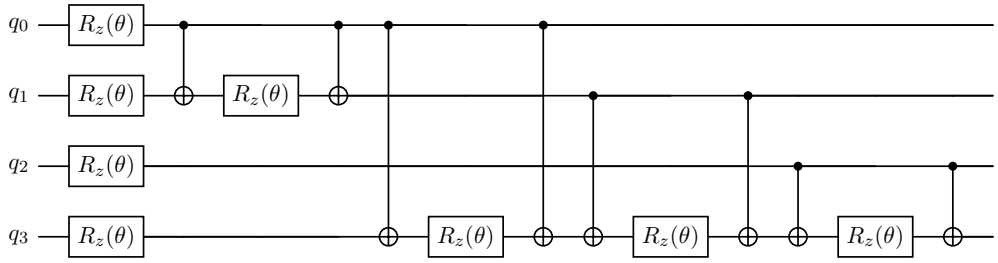


Figure 3.10: Partial quantum circuit for the QUBO Cost function (n=3) as described in equation (3.25).

This circuit can be designed in parallel, as shown in Figure 3.11, which illustrates its actual depth (Section 2.4.2).

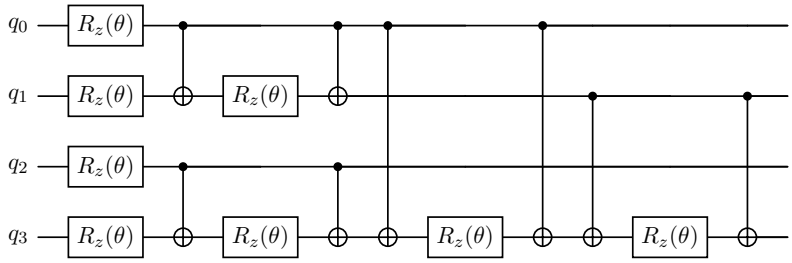


Figure 3.11: Partial quantum circuit for the QUBO Cost function (n=3) as described in equation (3.25), shown in parallel.

3.3.8 | Determining the number of R_{ZZ} Gates

Based on the information derived from the above and through deductive analysis of the initial cost function as referenced in equation (3.19), it is evident that for rotational ZZ gates, in the QUBO circuits, each non-ancillary qubit (i.e., a qubit directly corresponding to a boolean variable x_i in the PUBO problem) interacts with every ancillary qubit in the analogue QUBO representation. Furthermore, there is also an interaction between non-ancillary qubits themselves.

As a result, the expression for determining the quantity of rotational ZZ gates in the initial cost function (equation (3.19)) follows:

$$\#R_{ZZ(\theta)} = \left(n \left\lceil \frac{n(n-2)}{4} \right\rceil\right) + \binom{\lceil \frac{n}{2} \rceil}{2} + \binom{\lfloor \frac{n}{2} \rfloor}{2} \tag{3.26}$$

However, this approach to converting the CUBO cost function into a QUBO function involves segmenting the cost function into two halves, where the first $\lceil \frac{n}{2} \rceil$ variables interact with the last $\lfloor \frac{n}{2} \rfloor$ variables, and utilising the ancillary variables within each of these halves. This design can lead to duplicate interactions, manifesting from $n = 5$ onwards. In particular, the formula $\sum_{i=0,i<j<k}^{4} x_i x_j x_k$, will open as the following terms:

$$x_0x_1x_2 + x_0x_1x_3 + x_0x_1x_4 + x_0x_2x_3 + x_0x_2x_4 + x_0x_3x_4 + x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 \tag{3.27}$$

As evident from the terms in equation (3.27), except for the initial term $x_0x_1x_2$, the variables are distributed such that two of them reside on one side (either left or right) of the initial 3-term PUBO to QUBO partition, while the third variable is on the opposite side. As an exception, the first term facilitates the interaction between:

- variable x_0 with the ancillary variable created between x_1 and x_2
- variable x_1 with the ancillary variable created between x_0 and x_2
- variable x_2 with the ancillary variable created between x_0 and x_1

These terms generate three interactions among the three variables, although only one interaction for every three variables is required. Thus, the excess interactions are eliminated from equation (3.26). To achieve this, the quantity of 3-variable interactions within each half is assessed and then multiplied this count by 2, as shown below:

$$2 \cdot \left(\binom{\lfloor \frac{n}{2} \rfloor}{3} + \binom{\lceil \frac{n}{2} \rceil}{3} \right) \tag{3.28}$$

This will finally result in the formula to count all rotational ZZ for our cost function (equation (3.19)) as follows:

$$\#R_{ZZ(\theta)} = \left(n \left\lceil \frac{n(n-2)}{4} \right\rceil\right) + \binom{\lceil \frac{n}{2} \rceil}{2} + \binom{\lfloor \frac{n}{2} \rfloor}{2} - \left(2 \cdot \left(\binom{\lfloor \frac{n}{2} \rfloor}{3} + \binom{\lceil \frac{n}{2} \rceil}{3} \right)\right) \tag{3.29}$$

$$\#R_{ZZ(\theta)} = n \left\lceil \frac{n(n-2)}{4} \right\rceil + \binom{\lceil n/2 \rceil}{2} + \binom{\lfloor n/2 \rfloor}{2} - 2 \left(\binom{\lfloor n/2 \rfloor}{3} + \binom{\lceil n/2 \rceil}{3} \right) \tag{3.30}$$

$\forall n \in \mathbb{N}_e$ this formula can be expressed by removing all $\lceil \quad \rceil$ and $\lfloor \quad \rfloor$ symbols, therefore, getting:

$$\#R_{ZZ(\theta)} = \frac{2n^3 + 3n^2 - 14n}{12} \qquad \forall n \in \mathbb{N}_e \tag{3.31}$$

$\forall n \in \mathbb{N}_o$ this formula can be express by replacing $\left\lceil \frac{n(n-2)}{4} \right\rceil$ with the formula in equation (3.22), $\lceil \frac{n}{2} \rceil$ with $\frac{n+1}{2}$, and finally $\lfloor \frac{n}{2} \rfloor$ with $\frac{n-1}{2}$, therefore, getting:

$$\#R_{ZZ(\theta)} = \frac{2n^3 + 3n^2 - 14n + 9}{12} \qquad \forall n \in \mathbb{N}_o \tag{3.32}$$

which can finally be represented by

$$\#R_{ZZ(\theta)} = \left\lceil \frac{n(n-2)(2n+7)}{12} \right\rceil \tag{3.33}$$

Which scales on $\mathcal{O}(n^3)$.

3.3.9 | Combinatorial Optimisation Problems

Combinatorial optimisation (example TSP as per equation (3.6)) can be generalised as a problem defined by m clauses, each consisting of n bits. Each clause is a rule or constraint on a system for which one would like to find a lower or upper bound. The objective function is then found as follows:

$$C(x) = \sum_{\alpha=1}^{m} w_\alpha C_\alpha(x) \tag{3.34}$$

where:

m	:	the total number of α clauses
$C(x)$	:	the objective function of the system
$C_\alpha(x)$	:	the function relating to clause α
w_α	:	the weight assigned to clause α
x	=	a bit string $(x_1 x_2 \dots x_n)$

Each bit string x will result in 1 if clause α is satisfied else it will be 0. If one wants to minimise the object function of a system, then a clause is positive if it represents a cost or penalty, i.e. any quantity that we want to be present in the least possible or nothing at all.

On the other hand, in the same minimisation procedure, a clause is negative if it represents a desirable component, such as a benefit or a reward. A positive clause, therefore an undesirable clause, might be expenses such as production costs, while a negative clause, a desirable clause, might be revenue from production; thus, minimising a cost function with these clauses means reducing costs and increasing revenue. Each clause α can also have a weight w_α, which represents how much we would like this clause to be relevant to the overall objective function of the system.

In the following sections, I present examples of combinatorial optimisation problems.

3.3.9.1 | Max-Cut

In graph theory, the goal of Maximum Cut (Max-Cut) is to divide the vertices of a graph into a bipartite set while maximising the number of cuts, or *weight* of cuts, required in this separation. Mathematically, given an undirected graph $G = (V, E)$, with a vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and an edge set $E = \{e_{i,j}\} = \{(v_i, v_j) \mid v_i, v_j \in V\}$, each edge with weights $w_{ij}, \forall (i,j) \in E$, $w_{ij} = w_{ji}$, the Max-Cut problem's task is to partition the vertices of the graph into two sets S and $\bar{S}$ such that the total weight, equation (3.35), of edges between the two sets is maximised.

$$W(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} w_{ij} \tag{3.35}$$

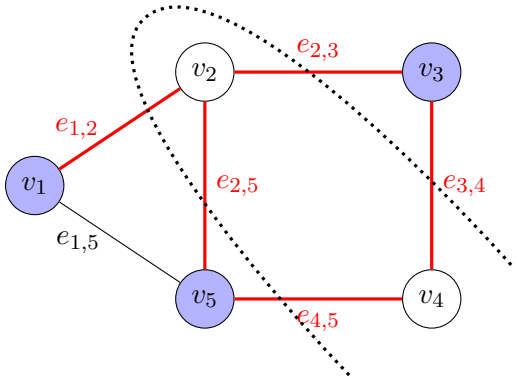


Figure 3.12: An example of a Max-Cut

In the example given by Figure 3.12, assuming that the graph is unweighted, so $w_{i,j} = 1$, then $S_1 = \{v_1, v_3, v_5\}$ and $S_2 = \{v_2, v4\}$ making the cut $e_{1,2} + e_{2,3} + e_{2,5} + e_{3,4} + e_{4,5}$, i.e. 5.

Max-Cut is part of *Karp's 21 NP-complete problems* [36]. As an NP-complete problem, it has no known algorithm which solves it in polynomial time, albeit its inverse problem, i.e. Min-Cut, has a polynomial time solution [37]. Max-Cut has applications in various domains, including machine learning, circuit layout design [38], and theoretical physics, where it is mathematically equivalent to the Ising model. It also arises in the analysis of social networks.

Using $z \in \{-1, +1\}$, the classical cost function for Max-Cut can be written as follows:

$$\textbf{Maximize } C(x) = \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - z_i z_j) \tag{3.36}$$

where:

w_{ij}	:	weight of the edge between vertices v_i and v_j
z_i, z_j	$\in$	$\{-1, +1\}, -1$ when vertex $\in S, 1$ when vertex $\in \bar{S}$

This mathematical formulation can be easily transformed into a Hamiltonian, suitable for quantum computation. The variables z_i can be mapped to spin variables Z_i, where Z_i is the Pauli-Z (σ^z) operator acting on the i-th qubit, as shown in equation (3.37).

$$z_i \to Z_i \tag{3.37}$$

Thus, the objective function becomes:

$$\begin{gathered} \sum_{(i,j)\in E} w_{ij} \left(\frac{1 - Z_i Z_j}{2} \right) \\ \sum_{(i,j)\in E} w_{ij} \left(\frac{1}{2} - \frac{1}{2} Z_i Z_j \right) \\ \sum_{(i,j)\in E} \frac{w_{ij}}{2} - \sum_{(i,j)\in E} \frac{w_{ij}}{2} Z_i Z_j \end{gathered} \tag{3.38}$$

As the first term is constant, it can be disregarded without affecting the optimisation outcome. Therefore, the final Hamiltonian for the Max-Cut problem can be written as follows:

$$H = - \sum_{(i,j)\in E} \frac{w_{ij}}{2} Z_i Z_j \tag{3.39}$$

The Max-Cut problem always yields a *degenerate* result, meaning that multiple distinct solutions achieve the same optimal value. This arises because the bitwise complement (i.e., applying a NOT operation) of any valid solution is also valid, leading to degeneracy. In this context, finding the optimal cut corresponds to finding the ground state (minimum energy state) of the associated Hamiltonian.

3.3.9.2 | Job-Shop Scheduling Problem (JSSP)

The **Job-Shop Scheduling Problem (JSSP)** is one of the most complex and challenging problems in combinatorial optimisation. The primary objective of JSSP is to find the most efficient way to schedule a number of jobs and their respective operations while considering several constraints. Usually, the goal of JSSP is to minimise completion time. Still, the cost function can also be designed to maximise profitability or any other metric important to the organisation performing the JSSP procedure.

The JSSP problem deals with a number of machines m $(M_1, M_2, \ldots, M_m)$ available to process n jobs $(J_1, J_2, \ldots, J_n)$, each with different known due dates. Each job consists of a number of ordered operations $(O_1, O_2, \ldots, O_n)$, each with known processing times and each assigned to specific machines. The main rules of any JSSP algorithm are usually:

- Each machine can process only one operation at any given time.
- There is a sequence that operations within a job must follow
- Operations are non-preemptable; once an operation starts, it has to be completed without interruption
- Tasks from different jobs are independent

Consider a JSSP optimisation problem involving two machines and two products (or SKUs). Each SKU yields a profit p and requires a single time unit t for production. The production process is handled by two employees, with each assigned to a distinct product. Consequently, the same product cannot be produced simultaneously on both machines, as no employee is available to operate two instances of the same product concurrently. The production planning Gantt chart would be structured as per Figure 3.13. The cost function for this problem can be formulated as per equation (3.40).

$$\begin{aligned} C(x) = \quad & p_1(q_{1,1}x_{1,1} + q_{1,2}x_{1,2}) + p_2(q_{2,1}x_{2,1} + q_{2,2}x_{2,2}) \\ & - w_1(x_{1,1}x_{2,1}) - w_2(x_{1,2}x_{2,2}) \\ & - w_3(x_{1,1}x_{1,2}) - w_4(x_{2,1}x_{2,2}) \end{aligned} \tag{3.40}$$

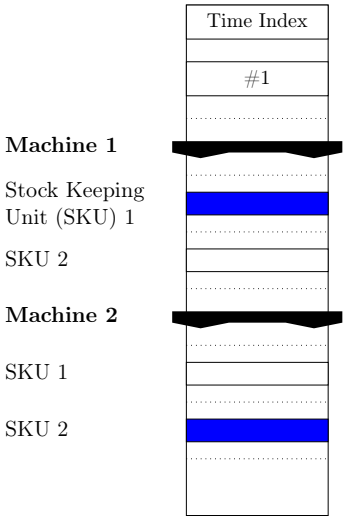


Figure 3.13: Simple Production Planning Gantt chart consisting of one time unit, two machines and two SKUs.

where:

$q_{s,m}$	=	the quantity produced for product SKU s in machine m
$x_{s,m}$	=	1 if quantity was produced for product SKU s in machine m, else 0
p_s	=	the profit for each unit of product SKU s
w_c	=	the weight constant for constraint c

and the constraints:

$-w_1(x_{1,1}x_{2,1}) - w_2(x_{1,2}x_{2,2})$	:	refers to the fact the two different products cannot be produced on the same machine at the same time; this is usually called the machine capacity constraint or the no overlap constraint
$-w_3(x_{1,1}x_{1,2}) - w_4(x_{2,1}x_{2,2})$	:	refers to the fact that, since two employees cannot do the same product, then, while one product is being manufactured on one machine, the same product cannot be manufactured on the other machine

A graph (Figure 3.14) can represent the constraints of this problem.

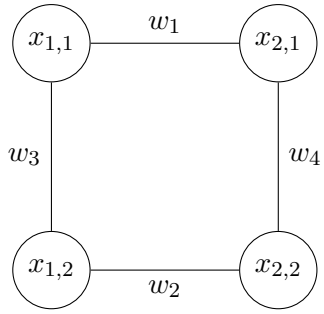


Figure 3.14: Graph showing the constraints of equation (3.40).

The constraint graph for this JSSP instance structurally resembles a Max-Cut problem, particularly in its partitioning objective. If the weights are similar (or the graph is unweighted, i.e. each weight equals 1), then one Max-Cut solution for this problem is shown in Figure 3.15.

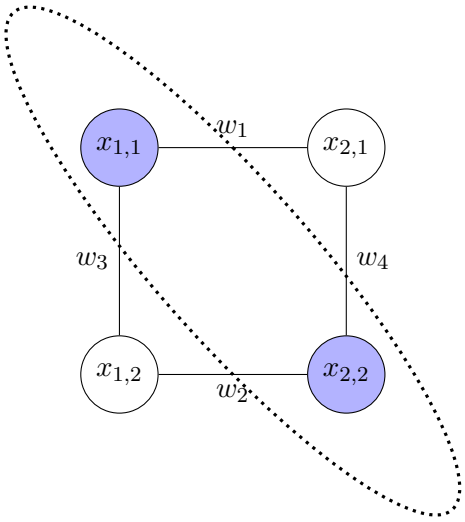


Figure 3.15: Graph showing the constraints of equation (3.40) with a depicted Max-Cut cut.

Assuming that $x_{s,m}$ is the quantity that can be produced in a single time index, then our Hamiltonian for our Cost function, $C(x)$ is actually:

$$
\begin{aligned}
H(z) = \quad & p_1(q_{1,1}(\frac{\mathbb{1}-Z_0}{2}) + q_{1,2}(\frac{\mathbb{1}-Z_1}{2})) + p_2(q_{2,1}(\frac{\mathbb{1}-Z_2}{2}) + q_{2,2}(\frac{\mathbb{1}-Z_3}{2})) \\
& - w_1((\frac{\mathbb{1}-Z_0}{2})(\frac{\mathbb{1}-Z_2}{2})) - w_2((\frac{\mathbb{1}-Z_1}{2})(\frac{\mathbb{1}-Z_3}{2})) \\
& - w_3((\frac{\mathbb{1}-Z_0}{2})(\frac{\mathbb{1}-Z_1}{2})) - w_4((\frac{\mathbb{1}-Z_2}{2})(\frac{\mathbb{1}-Z_3}{2}))
\end{aligned} \tag{3.41}
$$

where:

Z_0 = Pauli Z matrix for $x_{1,1}$
Z_1 = Pauli Z matrix for $x_{1,2}$
Z_2 = Pauli Z matrix for $x_{2,1}$
Z_3 = Pauli Z matrix for $x_{2,2}$

Assuming that the profits are equal, i.e., $p_1 = p_2 = 1$, this Hamiltonian becomes:

$$
\begin{aligned}
H(z) = \quad & \frac{1}{2}\left(q_{1,1} + q_{1,2} + q_{2,1} + q_{2,2} - (q_{1,1}Z_0) - (q_{1,1}Z_1) - (q_{2,1}Z_2) - (q_{2,2}Z_3)\right) \\
& - \frac{w_1}{4}((\mathbb{1} - Z_0 - Z_2 + Z_0Z_2)) - \frac{w_2}{4}((\mathbb{1} - Z_1 - Z_3 + Z_1Z_3)) \\
& - \frac{w_3}{4}((\mathbb{1} - Z_0 - Z_1 + Z_0Z_1)) - \frac{w_4}{4}((\mathbb{1} - Z_2 - Z_3 + Z_2Z_3))
\end{aligned} \tag{3.42}
$$

In production environments, it is common for requirements to include the production of multiple products with different quantities for sale orders, including a mix of products, some of which may be the same SKU, using multiple machines over a range of periods. This environment will also include additional constraints or parameters, such as the number of employees available at any time, cleaning or maintenance required on machines between different production runs, raw material requirements, different limited tools needed during production, and other factors. As constraints and parameters increase, more computational resources are required to solve JSSP. In a quantum computer, this would mean the number of qubits necessary to compute JSSP would increase linearly. Although this growth of resources is not exponential, it might still be beyond the capabilities of current NISQ machines.

The example above illustrates a two-parameter interaction, where Equation (3.42) represents a quantum Hamiltonian involving two interacting spin variables, Z_nZ_m. However, there might be scenarios where there is a constraint that the factory cannot handle the production of three components at once; for example, in a plastic injection moulding scenario, they have only two moulds for a specific component. In this case, one would have the following constraint:

$$C_x = \lambda\ x_{j_1 i_1 t}\ x_{j_2 i_2 t}\ x_{j_3 i_3 t}$$

This scenario creates a three-parameter interaction, which, as a polynomial problem, can be modelled with a hypergraph. The final quantum Hamiltonian, in this case, will have three interacting spin variables $Z_n Z_m Z_o$.

3.4 | Quantum Algorithms

In problem-solving, computer scientists study how algorithms perform in terms of computational complexity [39] as the amount of input information grows against the actual time for the algorithm to execute or the number of operations required. Algorithm performance is typically evaluated in terms of *big O notation* ($\mathcal{O}$) in dictating the asymptotic upper bound or order of approximation. For example the following notations: $\mathcal{O}(1)$, $\mathcal{O}(n)$, $\mathcal{O}(n \log n)$, $\mathcal{O}(n^2)$, $\mathcal{O}(2^n)$ quantify algorithmic efficiency.

An algorithm is considered to belong to the complexity class *Polynomial (P)* if its resource requirements grow at a rate bounded by a polynomial function of the input size. In other words, its complexity can be expressed as a polynomial in the size of the input. For example $f(n) = 3n^2 - 2n + 1$ exhibits a polynomial-time complexity of $\mathcal{O}(n^2)$.

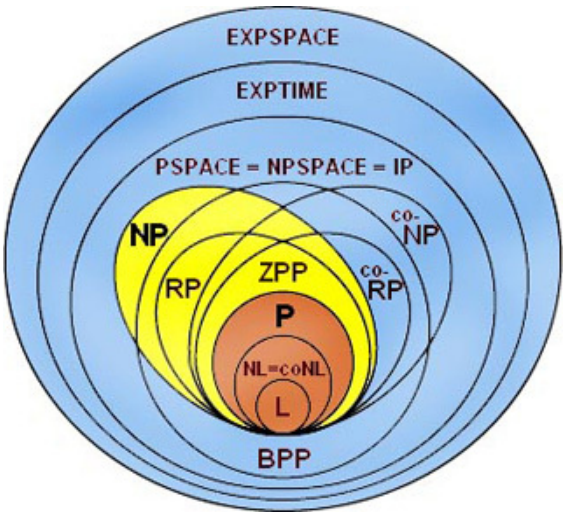


Figure 3.16: Complexity Classes
Credit: Brilliant Complexity Theory [40]

A problem belongs to *NP* if it cannot be solved in polynomial time, but whose answer, once found, can then be confirmed in polynomial time. In this case, the algorithm complexity grows exponentially with the size of the input, meaning that the required computational resources grow much faster than any polynomial function as the input parameters increase. For example, in the TSP (Section 3.3), the time complexity is

represented as $\mathcal{O}(n!)$. In the Max-Cut problem (Section 3.3.9.1), the time complexity is represented as $\mathcal{O}(2^n)$. A list of known computational complexity classes is available in the Complexity Zoo [41], which, at the time of publication of this dissertation, listed 550 complexity classes. Figure 3.16 shows a collection of well-known complexity classes and their relationships, illustrated as nested sets. The diagram highlights both deterministic and non-deterministic classes, as well as probabilistic ones, providing a general view of their relationship in terms of computational power.

Quantum algorithms are designed to solve problems that cannot be solved in polynomial time using classical algorithms. With the introduction of quantum algorithms, new complexity classes have been found that solve non-polynomial algorithms using a quantum speed-up.

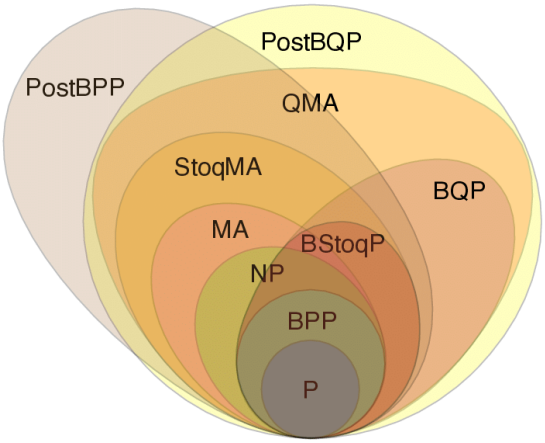


Figure 3.17: Classical and Quantum Complexity Classes
Credit: Albash and Lidar [42]

An example is **Bounded-Error Quantum Polynomial (BQP)** problems, a complexity class consisting of problems that can be solved in polynomial time by quantum computers with a small bounded probability of error. Examples of problems in BQP include Shor's algorithm* and Grover's algorithm§. Depending on its complexity, TSP algorithm can be classified under different quantum complexity classes [43]. Figure 3.17 illustrates the relationship between classical and quantum complexity classes, highlighting how classes such as BQP and QMA fit within the broader computational landscape.

*Shor's algorithm is a quantum algorithm that factors integers in quantum polynomial time (with the actual complexity class being BQP) with a complexity of $\mathcal{O}((\log N)^3)$.

§Grover's algorithm is a quantum algorithm designed to search in an unsorted database in $\mathcal{O}(\sqrt{N})$ time; its classical counterpart will search in $\mathcal{O}(N)$ time, making Grover's approach quadratically faster.

3.4.1 | Variational Quantum Algorithms (VQAs)

Variational Quantum Algorithms (VQAs) are a class of quantum algorithms designed to solve optimisation problems, including combinatorial optimisation, using parametrised quantum circuits. VQAs are hybrid algorithms, meaning that they are executed on both quantum computers, through a parametrised quantum circuit, and on classical computers, running an optimiser to minimise the cost function based on the outputs of the quantum circuit. The main advantage of VQAs is that they have a shallow circuit well-suited for NISQ devices, which have limited coherence times and gate fidelity. VQAs are primarily used in areas of quantum chemistry simulations [44], machine learning [45] and optimisation [46]. The most well-known and well-studied VQAs are the Variational Quantum Eigensolver (VQE)[¶] [47] and **Quantum Approximate Optimization Algorithm (QAOA)**, which is the main study in this dissertation (Section 3.5).

3.4.2 | Cost Landscape

Optimisation using VQAs has several limitations. These include **barren plateaus** [48], which are *flat areas* in the cost function where the search routine cannot decide on its next step. Another problem is **narrow gorges** [49], which are *narrow regions* near the global minima, making it difficult for the optimisation routine to locate the minima.

Visualising the VQA's cost landscape, as illustrated in Figure 3.18, can help understand the structure of the cost function. The landscapes of lower-dimensional cost functions are usually easy to visualise — for example, a line graph can represent a 2-D cost function. Several quantitative metrics [50] [51] can help to visualise a structure effectively, usually in a 3D landscape. Various visualisation techniques display a VQA's cost landscape, primarily scanning the landscape [52] by sampling several coordinates through a defined step size and studying the relation of the cost function at nearby points. It is essential to note that excessively distant coordinates, i.e., a low number of samples, will result in an incomplete landscape. In contrast, an excessive number of samples will incur significant computational costs.

[¶]The VQE is used for finding the ground state energy of quantum systems.

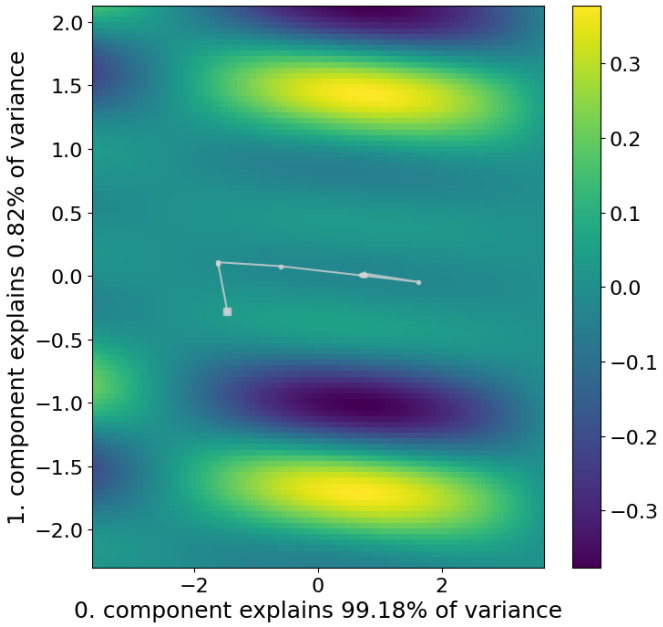


Figure 3.18: Cost landscape for the VQA of cost function $C(x) = \sum_{i=0}^{3}(-1)^i \cdot \prod_{j=0}^{2} x_{i+j}$. The yellow shades depict the maxima, while the blue shades are the minima. In this case, the VQA is the QAOA algorithm with $p = 1$ (Section 3.5). The path, in white, within the cost landscape is the path traversed by the QAOA algorithm as it attempts to find the solution. The path *finish-line* does not end in a maximum area (yellow), and in fact, QAOA fails to converge to the maximum in this instance.

During this research, I use ORQVIZ [53], a cost landscape library that helps visualise the landscape. The most commonly used visualisation of the cost landscape is a 2D heatmap scan (Figure 3.18), which, through the use of colour gradients, effectively conveys the topography of the landscape in three dimensions.

3.5 | Quantum Approximate Optimization Algorithm

The **Quantum Approximate Optimization Algorithm (QAOA)** [6] is a variational quantum algorithm designed to produce approximate solutions for combinatorial problems that are difficult to solve using classical algorithms. QAOA, illustrated in Figure 3.19, is a hybrid algorithm, i.e., it consists of a problem-specific ansatz[‖], or a quantum circuit,

[‖]The term *ansatz*, from the German word *Ansätz*, refers to a guess, or rather an initial estimate of a solution

running on a quantum device, and an optimiser, a classical algorithm running on a classical binary CPU.

The original QAOA paper [6], by Edward Farhi, Jeffrey Goldstone, and Sam Gutmann, submitted in 2014, aims to optimise a combinatorial optimisation problem, with Max-Cut being the specific problem tackled. In this paper, Farhi et al. [6] compare QAOA to Quantum Adiabatic Algorithm (QAA). They explain that QAA seeks exact solutions. In contrast, QAOA focuses on approximate solutions that scale better for practical implementation.

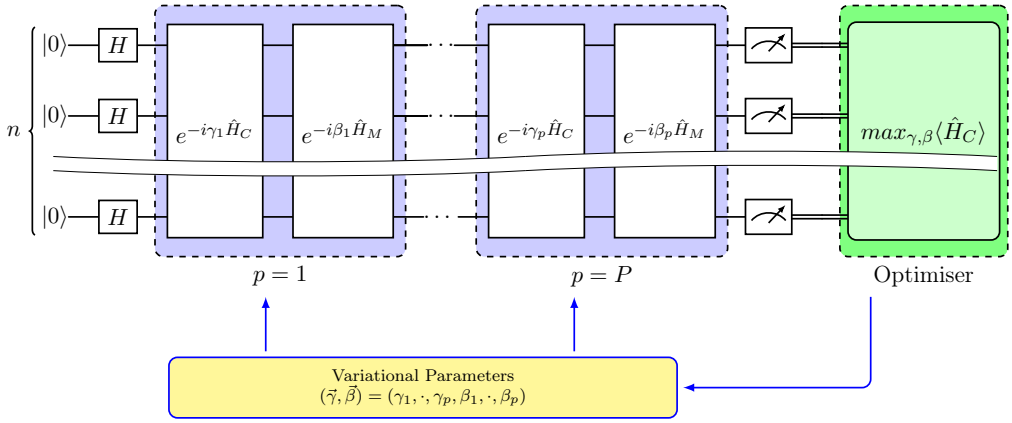


Figure 3.19: QAOA Circuit

The quantum part of QAOA consists of the following ansatz:

1. **Initial State Preparation:**
   - Qubits are initialised as $|0\rangle$
   - Qubits are then passed through a Hadamard gate to prepare them into a uniform superposition of all possible states in the computational basis. The system will be in the $|+\rangle^{\otimes n}$ state, where n is the number of qubits.
2. **Alternating Operator Layers:**
   The ansatz for QAOA contains alternating layers of two unitary operators: a cost function operator and a mixer operator. These layers are parametrised by angles later optimised by the classical optimiser algorithm, after which the layers are again executed until some condition is reached.
   - **Cost Function Operator ($\hat{U}_C(\gamma)$)**
     The cost function unitary operator is built from the problem's cost function

C, which is encoded into a Hamiltonian $\hat{H}_C$. Through this part of the circuit, the corresponding unitary operator $\hat{U}_C(\gamma) = e^{-i\gamma\hat{H}_C}$, where γ is a variational angle parameter, is applied to the quantum state. The operator is designed to minimise or maximise the cost function C through the variational parameter γ, which acts similarly to the time parameter in the adiabatic theory. This parameter controls the evolution of the quantum state and allows for fine-tuning of the state evolution to reach the lowest energy of the Hamiltonian $\hat{H}_C$.

- **Mixing Operator ($\hat{U}_M(\beta)$)**
  The mixing unitary operator $\hat{U}_M(\beta) = e^{-i\beta\hat{H}_M}$, where β is a variational angle parameter, helps transition between different quantum states in the superposition. It plays a role analogous to the initial Hamiltonian in adiabatic quantum computation, promoting exploration of the state space and helping the circuit escape local minima to reach an approximate value for the cost function.

These QAOA alternating layers operators are usually repeated for a fixed number of p times, with each layer containing both unitary cost function ($\hat{H}_C$) and mixer function ($\hat{H}_M$). Farhi et al. [6] specify that the approximation of the problem improves as p increases. While increasing p gives a better approximation, this complicates the quantum algorithm since NISQ devices are susceptible to decoherence errors on deep circuit depth [54]. For each call to the quantum computer, the algorithm needs a set of $2p$ angles, i.e. γ and β to produce the state $|\psi_p(\gamma,\beta)\rangle$, represented as follows:

$$\begin{aligned} |\psi_p(\gamma,\beta)\rangle &= \hat{U}_M(\beta_p)\hat{U}_C(\gamma_p)\dots\hat{U}_M(\beta_1)\hat{U}_C(\gamma_1)\,|s\rangle \\ &= e^{-i\beta_p H_B}e^{-i\gamma_p\hat{H}_C}\dots e^{-i\beta_1 H_B}e^{-i\gamma_1\hat{H}_C}\,|s\rangle \end{aligned} \tag{3.43}$$

where:

p	:	the number of layers in the QAOA circuit.
$\|s\rangle$	:	the initial state of the quantum circuit. In QAOA this is usually $\|+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}\|x\rangle$

3. **Measurement:**
   After applying both unitary operators using the initial parameters for γ and β, the first time the circuit is executed, or the optimised γ and β parameters in successive executions, the final quantum state is measured.

4. **Optimiser**
   After executing the layer and measuring the observable, the next step involves using a classical optimisation algorithm (Section 3.5.3.1) to refine each layer's variational parameters β and γ. The optimiser iteratively adjusts these parameters to minimise the expectation value of the problem Hamiltonian, thereby improving the circuit's approximation of the optimal solution.

This circuit is executed multiple times, and the most frequent measurement of the circuit's state gives a probable value for the binary variables in the QUBO (Section 3.3.1), or PUBO (Section 3.3.3), optimisation problem.

The acronym QAOA was reinterpreted in a paper by Hadfield et al. [55], this time as *Quantum Alternating Operator Ansatz* by extending the QAOA presented by Farhi et al. [6]. This generalised framework introduces more flexible families of parametrised operators, allowing the problem to solve a broader class of optimisation problems with hard constraints. The Quantum Alternating Operator Ansatz generalises the original QAOA by allowing problem-specific mixer operators and structured state spaces, thus allowing the algorithm to explore only the valid solutions that follow the rules of the problem. This makes it applicable to problems involving non-binary variables or feasibility constraints. The original QAOA is thus a special case within this broader ansatz, applicable primarily to unconstrained binary optimisation problems.

This dissertation primarily focuses on the original QAOA introduced by Farhi et al. [6], though it also reviews and discusses proposed variants of QAOA.

3.5.1 | QAOA relation to the Adiabatic Theorem

QAOA has been shown to be a discretised, gate-based analogue of the *QAA*, a continuous-time algorithm based on the Adiabatic Theorem [56]. QAA, just like *quantum annealing*, relies on slowly evolving parameters to transition a system from a known state into another state representing the problem into its optimal solution. Since QAA is an analogue algorithm, it cannot be directly executed using gate-based architecture.

In QAOA and QAA, Hamiltonians evolve from $\hat{H}_{initial}$ to $\hat{H}_{final}$. In QAOA, this evolution is discrete through parametric angle parameters (usually called γ and β). In this regard, QAOA takes from Trotterization [57], which is based on Trotter-Suzuki decomposition [58], a method used to approximate the evolution of complex Hamiltonians breaking them down into a sequence of simpler steps. The Trotter-Suzuki decomposition is based on the Lie product formula, named after Norwegian mathematician Sophus Lie.

Consider a Hamiltonian composed of two non-commuting parts, $\hat{H} = \hat{A} + \hat{B}$. The Lie product formula then approximates its exponential as follows:

$$e^{(\hat{A}+\hat{B})T} = \lim_{n\to\infty} \left(e^{\hat{A}t} e^{\hat{B}t}\right)^n \tag{3.44}$$

where:

T	:	total evolution time
n	:	number of Trotter steps
t	=	$\frac{T}{n}$

Since the limit of the formula in equation (3.44) is infinite, it must be truncated to implement it on a quantum computer, as shown in equation (3.45).

$$e^{-i(\hat{A}+\hat{B})T} \approx \left(e^{-i\hat{A}t} e^{-i\hat{B}t}\right)^n \tag{3.45}$$

This truncation introduces errors in the simulation, as follows, where $\mathcal{O}\left(\frac{T^2}{n}\right)$ is called the Trotter error, shown in equation (3.46).

$$e^{-i(\hat{A}+\hat{B})T} = \left(e^{-i\hat{A}t} e^{-i\hat{B}t}\right)^n + \mathcal{O}\left(\frac{T^2}{n}\right) \tag{3.46}$$

Stęchły [57] points out that while in AQC (Section 2.5.3), one starts from $\hat{H}_B$ and slowly moves to $\hat{H}_C$, QAOA *frantically* alternates between $\hat{H}_B$ and $\hat{H}_C$ choosing angles using a classical optimisation algorithm to mimic the AQC evolution until it reaches an approximation.

3.5.1.1 | The Zassenhaus formula

Similar to the Adiabatic Theorem, the Zassenhaus formula [59] is also another technique to express an exact exponential of sums of two non-commuting operators, as follows:

$$e^{A+B} = e^A e^B e^{-\frac{1}{2}[A,B]} e^{\frac{1}{6}([A,[A,B]]+[B,[B,A]])} \dots \tag{3.47}$$

While the Zassenhaus formula is not used directly in QAOA, it can be used to correct the Trotter approximation using higher orders. This first order of the Zassenhaus formula (QAOA-CD) was developed in parallel by Chandarana et al. [60] and Wurtz and Love [61], while the second order (QAOA-CD2) by Vizzuso et al. [62]. To implement first-order Trotterisation, one first needs to decompose the exponential $e^{-\frac{1}{2}[A,B]}$ from equation (3.47).

We want to use the Zassenhaus formula equation (3.47) at the first higher order in t, using $\hat{H}_A$ for the problem or cost function Hamiltonian and $\hat{H}_B$ for the mixer Hamiltonian, presented in equation (3.48).

$$e^{-it\left(\hat{H}_P+\hat{H}_M\right)} = e^{-it\hat{H}_P}e^{-it\hat{H}_M}e^{+\frac{t^2}{2}[\hat{H}_P,\hat{H}_M]} \tag{3.48}$$

If we evaluate for $t \ll 1$, we have the Suzuki-Trotter expansion:

$$e^{-it\left(\hat{H}_P+\hat{H}_M\right)} = e^{-it\hat{H}_P}e^{-it\hat{H}_M} \tag{3.49}$$

Hence, we need to decompose $e^{+\frac{t^2}{2}[\hat{H}_P,\hat{H}_M]}$ into quantum gates to design a quantum circuit which can implement the first higher order. Note that $\left[\hat{H}_P,\hat{H}_M\right]$ is not a hermitian operator, but it is anti-hermitian (skew-hermitian). Therefore,

$$\begin{aligned} C &= \left[\hat{H}_P,\hat{H}_M\right] &&= \hat{H}_P\hat{H}_M - \hat{H}_M\hat{H}_P &&\neq 0 \\ C^\dagger &= \left(\left[\hat{H}_P,\hat{H}_M\right]\right)^\dagger &&= \left(\hat{H}_P\hat{H}_M - \hat{H}_M\hat{H}_P\right)^\dagger = \hat{H}_M\hat{H}_P - \hat{H}_P\hat{H}_M &&= -C \end{aligned} \tag{3.50}$$

As a consequence, $e^{+\frac{t^2}{2}[\hat{H}_P,\hat{H}_M]}$ is a unitary operator, because if $P = -P^\dagger$, then e^P is unitary.

In our scenario

$$\hat{H} = \hat{H}_M + \hat{H}_P$$

$$\hat{H}_M = \sum_i h_i\hat{\sigma}_i^x \ , \ \hat{H}_P = \sum_{nm} t_{nm}\hat{\sigma}_n^z\hat{\sigma}_m^z$$

$$\left[\hat{H}_M,\hat{H}_P\right] = \left[\sum_i h_i\hat{\sigma}_i^x, \sum_{nm} t_{nm}\hat{\sigma}_n^z\hat{\sigma}_m^z\right]$$

$$\left[\hat{H}_M,\hat{H}_P\right] = \sum_{inm} h_i t_{nm}[\hat{\sigma}_i^x, \hat{\sigma}_n^z\hat{\sigma}_m^z]$$

Taking into consideration that

$$[A, BC] = B[A, C] + [A, B]C$$
$$[\hat{\sigma}_l^x, \hat{\sigma}_m^z] = -2i\hat{\sigma}_l^y\delta_{lm}$$

Then

$$\begin{aligned}\left[\hat{H}_M, \hat{H}_P\right] &= \sum_{inm} h_i t_{nm} (\hat{\sigma}_n^z [\hat{\sigma}_i^x, \hat{\sigma}_m^z] + [\hat{\sigma}_i^x, \hat{\sigma}_n^z] \hat{\sigma}_m^z) \\ &= -2i \sum_{inm} h_i t_{nm} \left(\hat{\sigma}_n^z \hat{\sigma}_m^y \delta_{im} + \hat{\sigma}_n^y \hat{\sigma}_m^z \delta_{in}\right) \\ &= -2i \sum_{nm} t_{nm} \left(h_m \hat{\sigma}_n^z \hat{\sigma}_m^y + h_n \hat{\sigma}_n^y \hat{\sigma}_m^z\right) \end{aligned} \tag{3.51}$$

This implies the need to apply a controlled operation based on $\sigma^z \otimes \sigma^y$, parametrised by an angle α. Similarly, a second controlled operation based on $\sigma^y \otimes \sigma^z$ must also be applied, using the same angle α**. Thus, the first higher order for the Zassenhaus formula, for the interaction between qubit n and qubit m, can be translated into the quantum circuit illustrated by Figura 3.20.

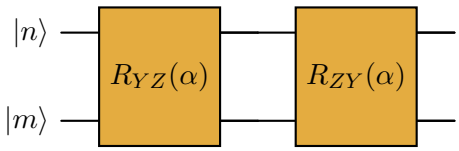


Figure 3.20: Partial quantum circuit showing the first higher-order of the Zassenhaus formula for $e^{\frac{t^2}{2}\hat{\sigma}_n^y \hat{\sigma}_m^z} + e^{\frac{t^2}{2}\hat{\sigma}_n^z \hat{\sigma}_m^y}$ as per equation (3.51), using sa-QAOA.

Chandarana et al. [60] claim that their implementation using the first higher-order Zassenhaus expansion, which they refer to as *Counterdiabatic QAOA* (CD-QAOA), performs better over the standard QAOA for various examples, including Max-Cut and the SK model [63]. Their results show that for QAOA circuits with a low number of layers (p), CD-QAOA converges faster to a solution than standard QAOA, by introducing just a single additional parameter α alongside the existing cost function parameter γ and mixer parameter β. Chandarana et al. [64] benchmarks their findings on Quantinuum's, Google's and IBM's quantum computers to tackle the protein folding problem on a tetrahedral lattice, obtaining a solution with low-depth circuits.

Vizzuso et al. [62] implement the second-higher order CD correction on QAOA. They claim that when the complexity of the cost function increases, CD-QAOA and their second-higher order implementation, CD2-QAOA, speed up the optimisation process. They find that this speed-up comes with a reduced number of layers required to arrive at a solution - an important factor to consider in the current NISQ era of quantum computers.

**In the single-angle version (sa-QAOA, Section 3.5), a single α_p parameter is used per layer; however, in the multi-angle version (ma-QAOA, Section 3.5.3.4), a distinct α_{g_p} parameter is assigned to each gate in each layer.

3.5.2 | Figures of Merit

To evaluate the success of QAOA across different configurations and in comparison to other algorithms, it is essential to use appropriate figures of merit. This section introduces the Approximation Ratio (AR) and Actual Probability (AP) metrics. Additionally, given that QAOA is a hybrid quantum-classical algorithm, this dissertation also considers the Number of Function Evaluations (nfev), which quantifies the number of times the quantum circuit has executed during the optimisation process.

3.5.2.1 | Approximation Ratio (AR)

The Approximation Ratio (AR) is the figure of merit that measures algorithm efficiency in approximation algorithms. The Approximation Ratio r is defined as follows:

$$\begin{aligned} \langle C \rangle &= \sum_i P(x_i) C(x_i) \\ r &= \frac{\langle C \rangle}{C_{\text{optimal}}} \end{aligned} \tag{3.52}$$

where:

$\{x_i\}$	:	the set of possible solutions for cost function C
$P(x_i)$	:	the probability of obtaining the solution x_i
C_{optimal}	:	the optimal solution for the cost function C, usually a known value found through a brute force algorithm

The AR measures the value of a cost function C relative to an optimal value. This indicates how close the algorithm's expected performance is to the best possible solution. The AR is a summary statistic that considers the probabilities of all possible solutions, making it a figure of merit more robust across all possible solutions weighted by their probabilities. According to Williamson and Shmoys [65, Definition 1.1], equation (3.52) follows the convention that AR $r \leq 1$ for maximisation problems, while $r \geq 1$ for minimisation problems. In both cases, $r = 1$ is the optimal solution, representing that the problem has been solved.

In terms of QAOA, if E is the expectation value of the Hamiltonian $\hat{H}_C$, given by equation (3.53).

$$E_p(\vec{\gamma}, \vec{\beta}) = \left\langle \psi_p(\vec{\gamma}, \vec{\beta}) \middle| \hat{H}_C \middle| \psi_p(\vec{\gamma}, \vec{\beta}) \right\rangle \tag{3.53}$$

where:

p	:	the number of layers in the QAOA circuit
ψ	:	the unitary operator which is represented by the QAOA circuit
γ, β	:	the parameters (angles) on the unitary operator ψ

The approximation ratio of the QAOA algorithm is now given by:

$$r = \frac{E_p(\vec{\gamma}^*, \vec{\beta}^*)}{C_{\text{optimal}}} \tag{3.54}$$

where:

γ^*, β^*	:	the set of parameters that maximise the expectation value E
C_{optimal}	:	the optimal solution for the cost function C. This refers to E_{min} or E_{max} depending if we are looking for a minimum or a maximum respectively.

3.5.2.2 | Actual Probability (AP)

Another figure of merit one can consider is **Actual Probability** (AP), i.e., the probability of obtaining a specific set of correct minimum (or maximum, depending on what we are looking for) solutions. However, AP can be sensitive to the solutions' accuracy. This may not account for near-optimal solutions that are still valuable in approximation algorithms.

3.5.2.3 | Comparing AR with AP

For example, consider two algorithms A and B, each with the following results:

Table 3.1: Shot Sample for Algorithm A

Minimum Value	Shots	% Occurrence
5	10	2%
4	40	8%
3	50	10%
2	100	20%
1	300	60%

Table 3.2: Shot Sample for Algorithm B

Minimum Value	**Shots**	**% Occurrence**
5	10	2%
4	40	8%
3	50	10%
2	400	80%
1	0	0%

AR can be calculated as follows:

$$AR = \frac{\sum_{i=1}^{n} \left(\text{Value}_i \times \frac{\%\ \text{Shots}_i}{100} \right)}{\text{Value}_{optimal}} \tag{3.55}$$

Algorithm A, with results presented in Table 3.1, achieves an AR of 1.72 and an AP of 60%, indicating that the solution reaches its minimum value of 1 at 60% of the time. Conversely, algorithm B, with results shown in Table 3.2, achieves a higher AR of 2.32 and an AP of 0%, since this latter algorithm does not get to the optimal value required, i.e. 1.

Table 3.3: Comparing Algorithm A with Algorithm B

Figure of Merit	**Algorithm A**	**Algorithm B**
AR	1.72	2.32
AP	60%	0%

Table 3.3 shows that AR results in a *low value* of 1.72 for Algorithm A and a *higher value* 2.32 for Algorithm B. Since this is a minimisation problem, AR indicates that Algorithm A is better.

AP is the percentage of the occurrence of the minimum value. Thus, AP shows no specific information on how much approximation was reached for both algorithms tested. While, in this case, AP also indicates that Algorithm A is better, it still does not give us any information on Algorithm B except that no minimum was reached.

3.5.2.4 | Number of Function Evaluations (nfev)

The Number of Function Evaluations (nfev) figure of merit tracks the number of times the optimisation algorithm evaluates the objective function during the optimisation process. If

two algorithms achieve comparable AR values, the one with fewer evaluations is considered more resource-efficient. A lower nfev indicates reduced resource usage compared to algorithms requiring more evaluations.

3.5.3 | QAOA Configurations: Optimization and Parameter Settings

Several studies have proposed modifications to the configuration of QAOA, including parameter tuning and novel extensions, intending to improve the algorithm's AR, nfev, or both.

3.5.3.1 | Classical Optimisation Algorithms

The classical algorithm in QAOA has the job to optimise the parameters β and γ angles (or multiple β's and γ's angles as in the case of multi-angle versions of the QAOA, Section 3.5.3.4). This is done by minimising a function and getting the most optimised parameters that minimise that function.

Most minimisation algorithms are based on the Newton Method (also known as the Newton-Raphson method [66]). The method iteratively approximates a non-quadratic function by a quadratic one that matches its value, gradient, and curvature at a chosen point. In the case of QAOA, the minimum depends on the selected initial angles. Starting from an initial guess, the function is approximated by a quadratic model. Since quadratic functions are easy to minimise, the minimum of this approximation is used as the new guess. This sequence is executed repeatedly until the parameter values converge to an optimal or near-optimal solution.

The quadratic approximation is given by:

$$g(x) = \frac{1}{2}(x-q)^T A(x-q) + (x-q)^T \Delta f(q) + f(q) \tag{3.56}$$

where:

q	:	the point where the original function is approximated
$f(q)$	:	the original function at point q
$(x-q)^T \Delta f(q)$	:	assures that the approximation has the same gradient as the original function at point q
$(x-q)^T A(x-q)$	:	the quadratic component of the approximation

Note that when $x = q$, we have $g(x) = g(q)$. A major drawback of the Newton Method is its reliance on the Hessian which may be unavailable, computationally expensive due

to its size, or not positive definite—leading to a quadratic approximation that does not possess a proper minimum (or maximum, when seeking a maximum).

To circumvent the problems caused by the Hessian, **quasi-Newton Methods** help find an approximation of the Hessian relatively quickly. The **BFGS** algorithm [67] [68] [69] [70], developed independently by four mathematicians, is one such algorithm.

Constrained Optimization BY Linear Approximation (COBYLA) algorithm, on the other hand, does not rely on the gradient as Newton method optimisers do. COBYLA uses a linear approximation of the function in the proximity of the selected point to evaluate a possible better next minimum or maximum point.

Bayesian Optimisation is a probabilistic algorithm that optimises black-box functions. It uses a Gaussian Process to model an objective function and selects the most promising points using an acquisition function to determine the next point to evaluate by exploration, sampling a point where the model is uncertain, or exploitation, sampling a point where the model predicts our solution. This is done by iterating several times until an approximation is used.

Powell is a gradient-free non-stochastic optimisation algorithm used particularly in algorithms that are noisy or computationally expensive.

Choosing one optimisation algorithm over another effects QAOA's ability to converge to a solution faster. The optimiser design may be subject to various advantages or disadvantages, such as simplicity and performance, therefore the speed of execution, especially for complex landscapes, adaptability in handling large datasets, the risk of some designs being stuck in local minima more than others and scalability for higher dimension problems amongst others. Pellow-Jarman et al. [71] concludes that BFGS performs well under realistic NISQ devices' noise levels. While COBYLA performs well in a noise-free state simulation, its performance is significantly degraded under noisy NISQ device environments. Tibaldi et al. [72] shows that Bayesian Optimisation is a robust method accounting for noise levels in sampling and NISQ machines.

In this dissertation, I use COBYLA and Powell optimisation algorithms.

3.5.3.2 | Different Mixers

Mixers in QAOA are responsible for exploring the solution space in the quantum state, with the transverse-field mixer, given by equation (3.57), being the most well-known, as it was proposed in the original QAOA [6] paper. Thanks to the Pauli-X (σ^x) operator, this mixer enables qubits to transition between computational basis states with different Hamming weights.

$$H_M = -\sum_i X_i \tag{3.57}$$

where:

X_i : Pauli-X operator applied to qubit i

There are studies of other mixers, such as Bartschi and Eidenbenz [73], who use Grover-like selective phase shift mixing operators; these apply a global reflection about the uniform superposition state and are compatible with the original QAOA, though less commonly used due to their non-local nature and implementation complexity.

3.5.3.3 | Initial Angles

As discussed in Section 3.5.3.1, the choice of initial point, points, or parameters in optimisation algorithms is vital to help converge faster to a solution. The same applies to quantum algorithms: choosing the angle parameters in QAOA is essential to help quantum algorithms converge faster to a solution.

One initialisation technique we will use in this dissertation is Trotterised Quantum Annealing (TQA) initialisation [74], which is parametrised by the Trotter time step. Sack and Serbyn [74] claim that TQA allows to circumvent the issue of false minima. TQA is based on the Adiabatic Theorem (equation (2.5.3)) and, using equation (2.8) $\hat{H}(t) = s(t)\hat{H}_I + (1 - s(t))\hat{H}_C$, the angles are set as $\beta = (1 - \frac{t}{T})\Delta t$ and $\gamma = (\frac{t}{T})\Delta t$. This is implemented in QAOA by setting the following initial angles per layer.

$$\gamma = \frac{i}{T}\Delta t, \quad \beta = \left(1 - \frac{i}{T}\right)\Delta t \tag{3.58}$$

where:

i = $1 \dots p$

Δt = $\frac{t}{T}$, time step

3.5.3.4 | Multi Angle QAOA (ma-QAOA)

In the QAOA, as presented by Farhi et al. [6], each layer p comprises a mixer and a cost function, and each one of these is assigned a different angle. Therefore, we use $2p$ angles in each iteration, after which a classical optimiser of choice optimises these angles.

Herrman et al. [75] present Multi Angle QAOA (ma-QAOA) where each rotational gate (R_Z in the cost function or R_X in the mixer) is given a separate angle which is also

optimised after each iteration. This approach will result in $p(g+n)$ angles, where g is the number of gates in the cost function and n is the number of qubits in the circuit since each qubit is assigned a single rotational X-gate (R_X) per layer in the mixer part of the circuit. This calculation assumes that the algorithm uses the standard mixer defined in Farhi et al.'s [6] QAOA implementation.

Since this approach needs more angles (i.e. classical parameters) for the classical optimiser, this also means that the classical algorithm will need more classical resources to find a solution. Nonetheless, although ma-QAOA requires more classical computational resources, the number of layers p in the QAOA implementation can be reduced to achieve a good approximation with shallower circuit depth than that of Farhi et al.'s [6] QAOA, making it a more suitable for NISQ devices. In ma-QAOA, the cost and the mixer operators are defined as follows:

$$\begin{aligned} \hat{U}_C(\gamma_p) = e^{-i\sum_{v=1}^{g}\gamma_{p,v}\hat{H}_{C,v}} &= \prod_{v=1}^{g} e^{-i\gamma_{p,v}\hat{H}_{C,v}} \\ \hat{U}_M(\beta_p) = e^{-i\sum_{w=1}^{n}\beta_{p,w}\hat{H}_{M,w}} &= \prod_{w=1}^{n} e^{-i\beta_{p,w}\hat{H}_{M,w}} \end{aligned} \tag{3.59}$$

where:

p	:	the number of layers in the QAOA circuit
g	:	the number of gates in the unitary cost operator $\hat{H}_C$
$\gamma_{p,g}$	:	$(\gamma_{1,1},\ \gamma_{1,2},\ \ldots,\ \gamma_{p-1,g},\ \gamma_{p,1},\ \gamma_{p,2},\ \ldots,\ \gamma_{p,g})$ the parameters (angles) in the unitary cost operator $\hat{H}_C$
n	:	the number of qubits in the circuit, and therefore the number of gates in the unitary mixer operator $\hat{H}_M$
$\beta_{p,n}$	:	$(\beta_{1,1},\ \beta_{1,2},\ \ldots,\ \beta_{p-1,n},\ \beta_{p,1},\ \beta_{p,2},\ \ldots,\ \beta_{p,n})$ the parameters (angles) in the unitary mixer operator $\hat{H}_M$

Herrman et al. [75] conclude that ma-QAOA converge to an optimal solution and that $\langle C\rangle_1^{ma} \geq \langle C\rangle$. In fact, QAOA is a special case of ma-QAOA. They found that it increases the approximation ratio in numerical optimisations for graphs that they studied. In their conclusion, they also mention that the drawback of this configuration, as mentioned above, is that the number of classical parameters is $p(g+n)$ instead of $2p$ in QAOA.

3.5.3.5 | Automorphic Angle QAOA (am-QAOA)

As a trade-off between the (*single angle*) QAOA proposed by Farhi et al. [6] and ma-QAOA proposed by Herrman et al. [75], Shi et al. [76] propose best-1sym-QAOA, max-sym-QAOA, and randgroup-QAOA in which they exploit the symmetries of the underlying graphs of the problems being studied. A *common* angle is used across all vertices falling under the same symmetry group, thus reducing the number of angles required for QAOA. In this dissertation, the implementation that exploits the symmetry of the underlying graph is referred to as Automorphic Angle QAOA (am-QAOA).

3.5.3.6 | k-interaction Angle QAOA (ka-QAOA)

In this dissertation, I present my original research proposing a novel configuration of QAOA, referred to as k-interaction Angle QAOA (ka-QAOA). This approach assigns a *common* angle to all terms in the cost function that involve the same number of interactions between Boolean variables in a PUBO. In equation (3.60), the parameter k denotes the number of interacting variables within each term of the cost function.

$$C'(x) = \sum_{i=0}^{n-k} (-1)^i \cdot \prod_{j=0}^{k-1} x_{i+j} \tag{3.60}$$

A single cost function may contain terms involving different numbers of variable interactions — for example, some terms may involve three variables, while others involve two. In the case of ka-QAOA, a distinct angle is assigned to the gates corresponding to three-variable interactions, another angle to two-variable interactions, and a separate angle to terms that involve no interaction with other variables. Note also that transforming the Boolean PUBO to spin notation for quantum circuits, equation (3.8), preserves the same k-local interactions. The mixer in ka-QAOA also continues to use a different angle parameter for each R_x gate, as is done in ma-QAOA.

As an example, consider a 4-qubit Cost Hamiltonian, equation (3.61), which translates to the circuit in Figure 3.21. It is immediately apparent that using fewer classical parameters will reduce the number of classical function evaluations. As I will show, tests indicate that it is done at a minimal cost to the Approximation Ratio and Actual Probability of the quantum algorithm.

$$C(x) = Z_0Z_1Z_2 - Z_0Z_1 - Z_0Z_2 + Z_0 - Z_1Z_2Z_3 + Z_1Z_3 + Z_2Z_3 - Z_3 \tag{3.61}$$

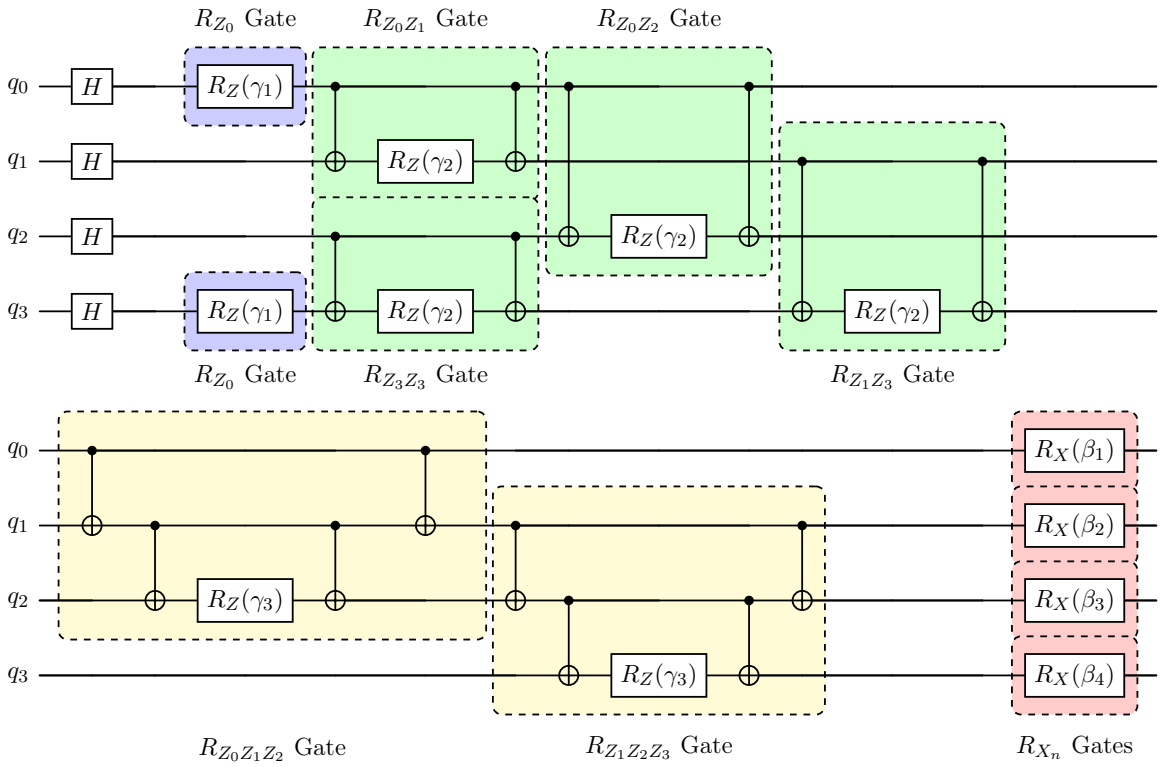


Figure 3.21: k-interaction Angle QAOA (ka-QAOA) four-qubit quantum circuit for equation (3.61) with $p = 1$ layers having parametrised angles γ_1 for R_{Z_m} (blue) gates, γ_2 for $R_{Z_mZ_n}$ (green) gates, γ_3 for $R_{Z_nZ_nZ_o}$ (yellow) gates, as well as β_1, β_2, β_3, and β_4 for R_{x_n} (red) gates.

3.6 | State Vector Simulator

The State Vector Simulator is a tool that runs on classical computers. It explores the quantum state of qubits and evolves them as each qubit passes through quantum gates. The system's state consisting of n qubits that reside in a 2^n dimensional Hilbert space is represented by:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle \tag{3.62}$$

where:

α_i : complex coefficients or amplitudes
$|i\rangle$: the basis states

The action of a quantum circuit yields:

$$|\psi_{new}\rangle = U\,|\psi_{current}\rangle \tag{3.63}$$

where:

U : the Unitary Matrix of the quantum gate

After applying all the gates in the quantum circuit, the final result will be an error-free representation of the final vector space ψ_{final}, giving us all the outcomes of the probabilities and the expectation values of the quantum operators. While this is a great tool to test quantum circuits, the resources required to simulate qubits increase exponentially ($\mathcal{O}(2^n)$), including the processing power and memory requirements of the classical computer.

3.7 | Development Environment

The quantum algorithms in this dissertation were developed using two libraries: Qiskit and Qulacs.

Qiskit [77] is a free and open-source software development kit (SDK) for Quantum Computing developed by IBM Research, giving developers the tools to generate and manipulate quantum programs and execute them on prototype quantum devices on the IBM Quantum Platform or simulators on a local computer. It uses a circuit model of universal quantum computation and can be utilised for any quantum hardware which follows such a model.

Qulacs [78] is another free library developed using C/C++, which can be used in Python to develop high-speed circuit simulations. Qulacs supplies only a state vector simulator with no possibility of executing a quantum circuit on physical hardware. Nonetheless, the library's speed, using GPUs, and efficiency are ideal for simulations in research environments.

4

Results & Discussion

In this section, I present the main results of my research. I evaluated the Quantum Approximate Optimization Algorithm (QAOA) as originally proposed by Farhi et al. [6]. Building on my research, I also tested QAOA for several variants, including the first higher-order of the Zassenhaus expansion (QAOA-CD), multi-angle QAOA (ma-QAOA), Automorphic QAOA (am-QAOA), and my novel proposal k-interaction Angle QAOA (ka-QAOA).

The source code for these results can be found on my GitHub at https://github.com/camillerievan. During these evaluations, I used several classical optimisation algorithms and various initial angle configurations.

Understanding the Tables and Figures

This section presents tables summarising the average number of function evaluations (*nfev*) (Section 3.5.2.4) for tests on various QAOA variants applied to hypergraphs of different node sizes n. Each value is accompanied by a subscript denoting the average Approximation Ratio (AR) (Section 3.5.2.1) achieved for that specific graph size n, layer count p, and QAOA variant. If no subscript is shown, an AR of 1 was attained, indicating the optimal value, as these tests aimed to maximise the cost function (Section 3.5.2.1).

This section also includes a series of charts that visually depict the Approximation Ratio (AR), Actual Probability (AP), and Number of Function Evaluations (nfev) obtained from these tests, providing insights into performance across the different configurations. The results for k-angle ●, single angle ◆ and automorphic angle ▲ have been shifted horizontally for better readability. The vertical lines represent the minimum and maximum results achieved, while the shaded areas represent the variance.

4.1 | Max-Cut

Several Max-Cut executions were performed across different graph types—both weighted and unweighted—with varying numbers of nodes, QAOA layers (p), and classical optimisation algorithms, using either the single-angle (sa-QAOA) or multi-angle (ma-QAOA) approach. The primary graphs used for Max-Cut executions were Erdős-Rényi graphs (also known as random internet graphs or binomial graphs), cycle graphs, random 3-regular graphs, and star graphs.

4.1.1 | QAOA

Section 3.5 discussed the Quantum Approximate Optimization Algorithm (QAOA) [6]. The QAOA quantum circuit is composed of a cost function operator $\hat{U}_C(\gamma)$ (shown in Listing 1) and a mixing operator $\hat{U}_M(\beta)$ (shown in Listing 2) which can be repeated into p layers. In sa-QAOA, each circuit layer has two parameters: one for the cost function and the other for the mixing operator.

```
for edge in self._graph.graph.edges:
    circuit.add_CNOT_gate(edge[0], edge[1])
    circuit.add_RZ_gate(edge[1], gamma)
    circuit.add_CNOT_gate(edge[0], edge[1])
```

Listing 1: Python code using the Qulacs library to include the Cost function operator, $\hat{U}_C(\gamma)$, into a quantum circuit.

```
for i in range(self._nodes_count):
    circuit.add_RX_gate(i, beta)
```

Listing 2: Python code using the Qulacs library to include the Mixer operator, $\hat{U}_M(\beta)$, into a quantum circuit.

4.1.2 | QAOA first higher-order

Section 3.5.1 discussed the Adiabatic Theorem and how QAOA is a result of equation (3.45), i.e. the Suzuki-Trotter decomposition. On the other hand, it also analysed how the Zassenhaus expansion, equation (3.47), can be used to correct the Trotter approximation using the expansion's higher orders.

In this section, I implement the first order of the Zassenhaus expansion $e^{-\frac{1}{2}[A,B]}$ into QAOA and test if the implementation achieves a better result than QAOA. I compare the results of sa-QAOA, as presented by Farhi et al. [6] to results obtained by executing the same problems using QAOA-CD [60]. To implement the first higher-order Trotterisation, one first needs to decompose the exponential $e^{-\frac{1}{2}[A,B]}$ (Section 3.5.1.1) as represented by the circuit in Figure 3.20 and coded in Listing 3.

```
for edge in self._graph.graph.edges:
    circuit.add_multi_Pauli_rotation_gate([edge[0], edge[1]], [3, 2], delta)
    circuit.add_multi_Pauli_rotation_gate([edge[0], edge[1]], [2, 3], delta)
```

Listing 3: Python code using the Qulacs library (Section 3.7) to incorporate the first-order term of the Zassenhaus expansion, $e^{-\frac{1}{2}[A,B]}$, into a quantum circuit.

4.1.3 | Erdős-Rényi Graph

The Erdős-Rényi graph [79] [80] is a graph defined as $G(n, M)$, where n is the number of nodes and M is the number of edges. For this example, I am generating an Erdős–Rényi–Gilbert graph [81], a variation of the Erdős-Rényi graph model. While the Erdős-Rényi model has a fixed number of edges, the number of edges in the Erdős–Rényi–Gilbert is based on a probability parameter, resulting in a graph definition of $G(n, \rho)$. Here, n is the number of nodes, and each edge is included in the graph based on a probability of ρ. An example is illustrated in Figure 4.1.

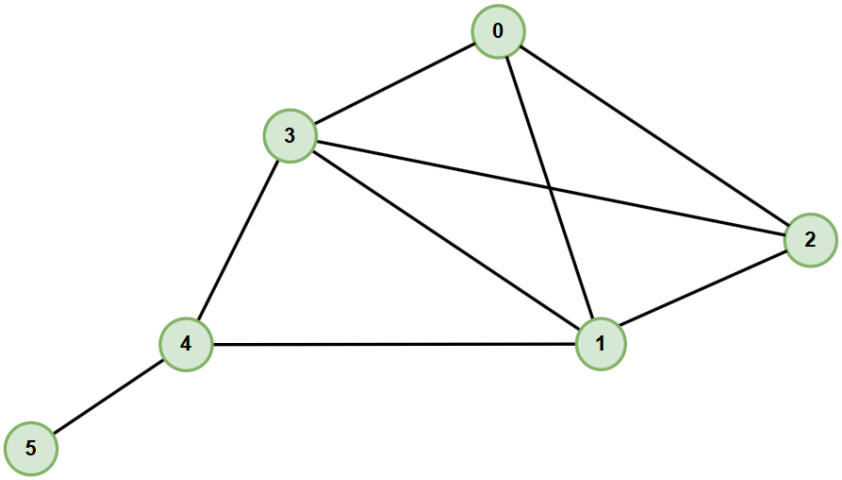
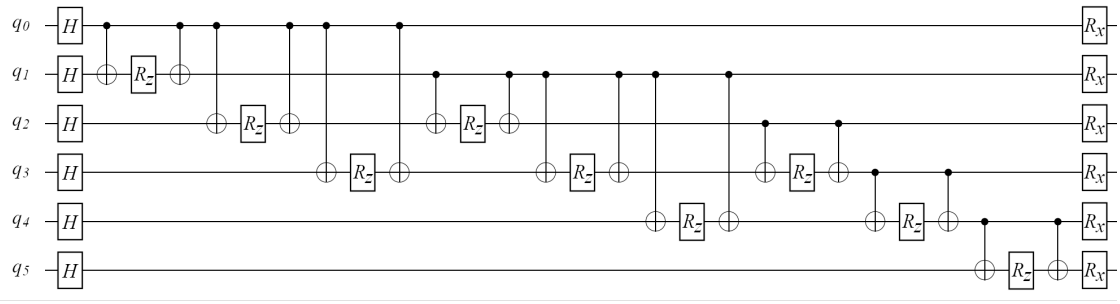


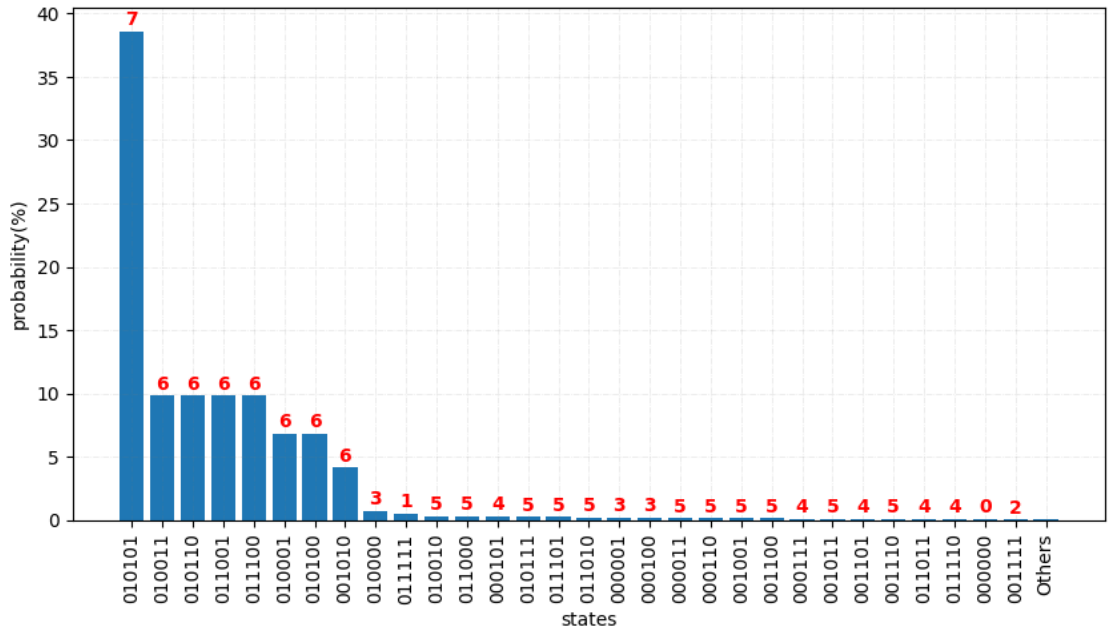
Figure 4.1: An unweighted Erdős-Rényi graph created with six nodes and nine edges $G(6, 9)$.

These types of random graphs are widely employed in network science applications, including social networks, epidemic modelling [82], percolation theory, gene regulatory networks, and financial systems, making them foundational tools for analysing complex systems. Erdős–Rényi graphs frequently appear in the literature on QAOA. For instance, Herrman et al. [75] evaluate ma-QAOA using collections of Erdős–Rényi graphs, while Amosy et al. [83] employ them to assess a method that leverages a fully connected neural network to improve QAOA initialisation by learning from previous executions.

The following charts present two tests, sa-QAOA as presented by Farhi et al. [6] (Figure 4.2), and the first higher-order QAOA, QAOA-CD (Figure 4.3). In both cases, the circuit is illustrated with $p = 1$, along with a bar chart displaying the output state probabilities for $p = 3$.

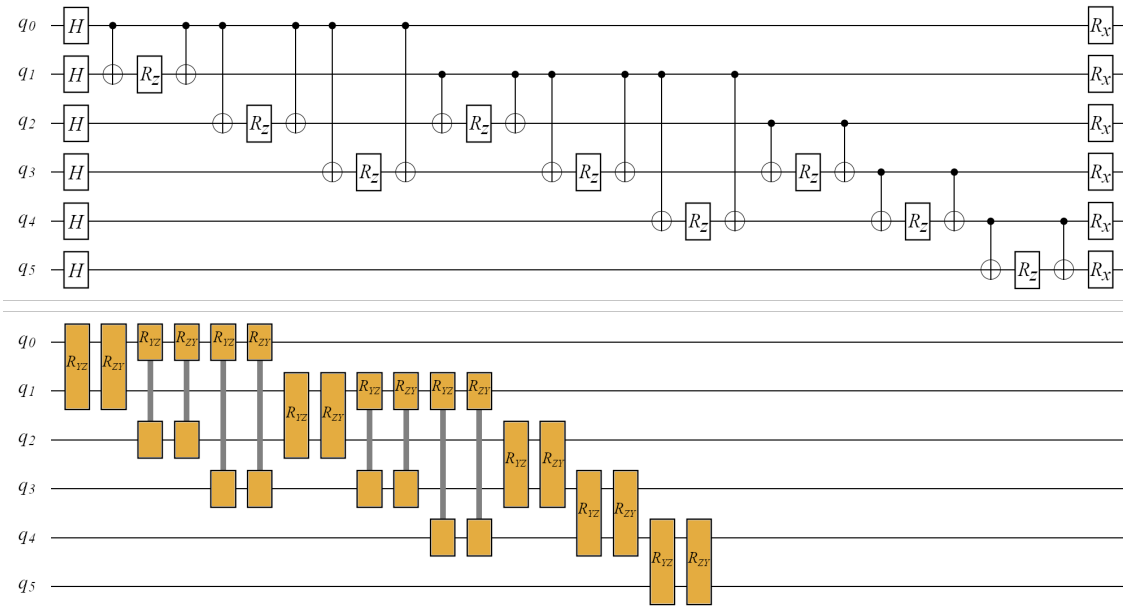


(a) sa-QAOA circuit representation for $p = 1$.

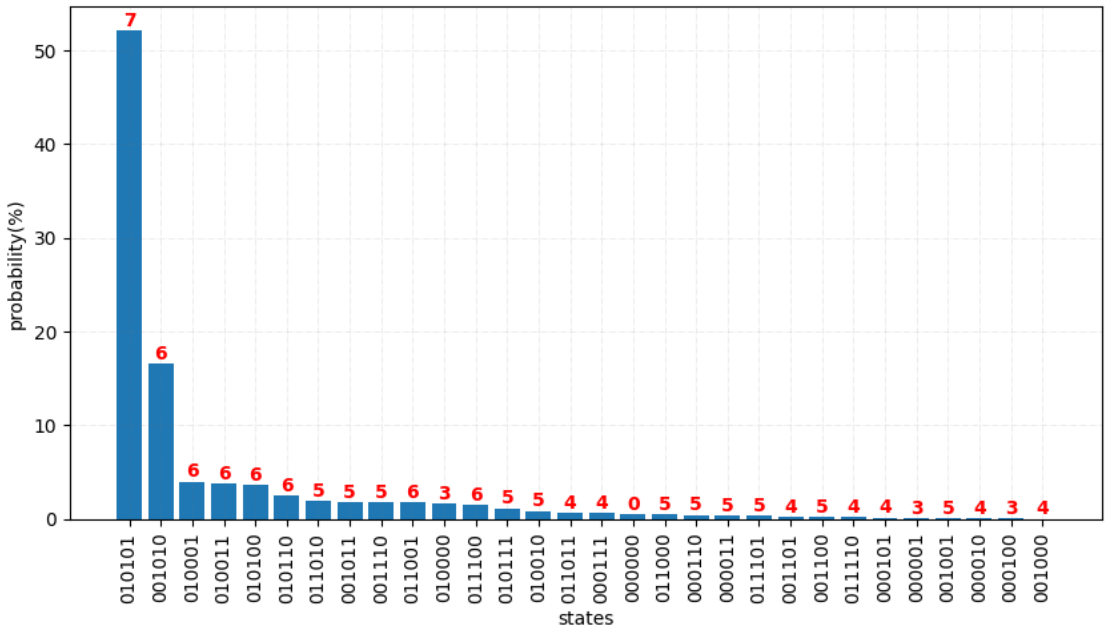


(b) Bar chart of state outcome probabilities for $p = 3$ for sa-QAOA.

Figure 4.2: Circuit diagram and probability distribution bar chart for sa-QAOA applied to the Erdős-Rényi graph shown in Figure 4.1. The result is degenerate, yielding a Max-Cut value of **seven** for **38.58%** of the time with **161** number of function evaluations for $p = 3$.



(a) QAOA-CD circuit representation for $p = 1$ (orange gates denote the *higher order* gates).



(b) Bar chart of state outcome probabilities for $p = 3$ for QAOA-CD.

Figure 4.3: Circuit diagram and probability distribution bar chart for QAOA-CD applied to the Erdős-Rényi graph shown in Figure 4.1. The result is degenerate, yielding a Max-Cut value of **seven** for **52.10%** of the time with **840** number of function evaluations for $p = 3$.

The results of the test done above are as follows:

Table 4.1: Comparing QAOA with First Higher-Order QAOA (QAOA-CD)

Figure of Merit	**QAOA**	**QAOA-CD**
AR	0.898	0.899
AP	38.58%	51.10%
nfev	158	840

The first higher-order QAOA outperforms standard QAOA when one compares both algorithms using AP. However, the difference in AR is negligible. Additionally, the first higher-order QAOA has a high number of Number of Function Evaluations (nfev), requiring more computational resources. As a result, I do not find it feasible to use the first higher-order QAOA when applied to Erdős-Rényi graphs. One must also consider that in a NISQ device, the fact that the first higher-order QAOA requires a deep circuit makes it more susceptible to errors during execution, further limiting its practicality.

4.1.4 | Star Graph

A Star Graph S_n consists of n vertices and $n-1$ edges. One central vertex has degree $n-1$, as it is connected to all other vertices in the graph. Each of the remaining vertices has a degree of 1 and is linked only to the central vertex by a single edge. Figure 4.4 shows an unweighted Star graph created with seven nodes and six edges.

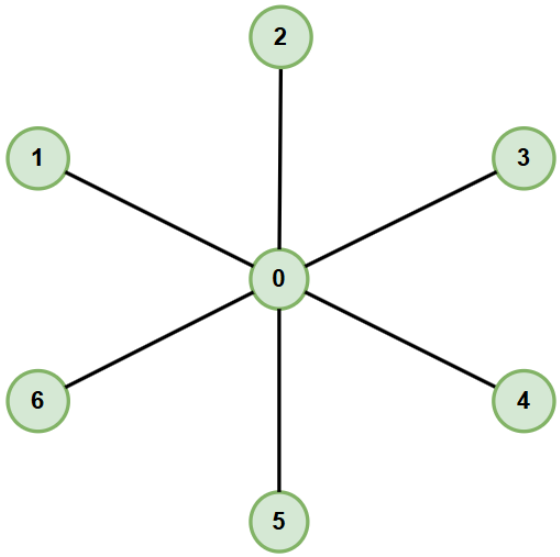
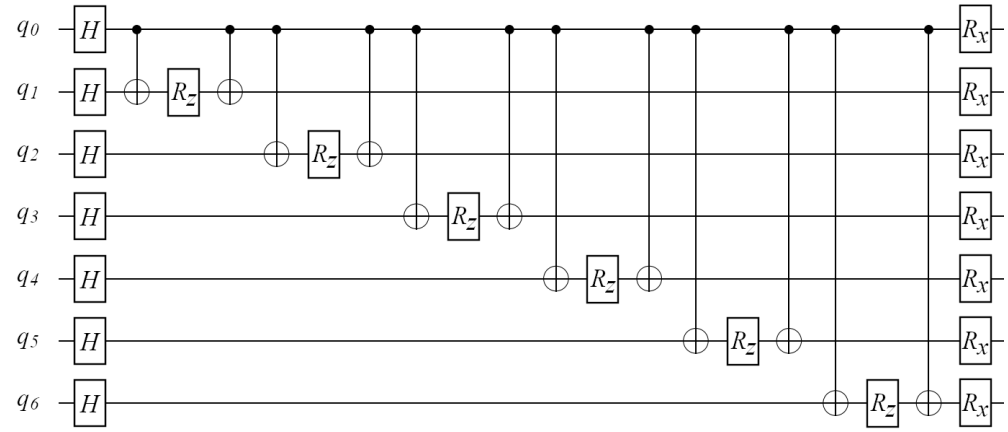


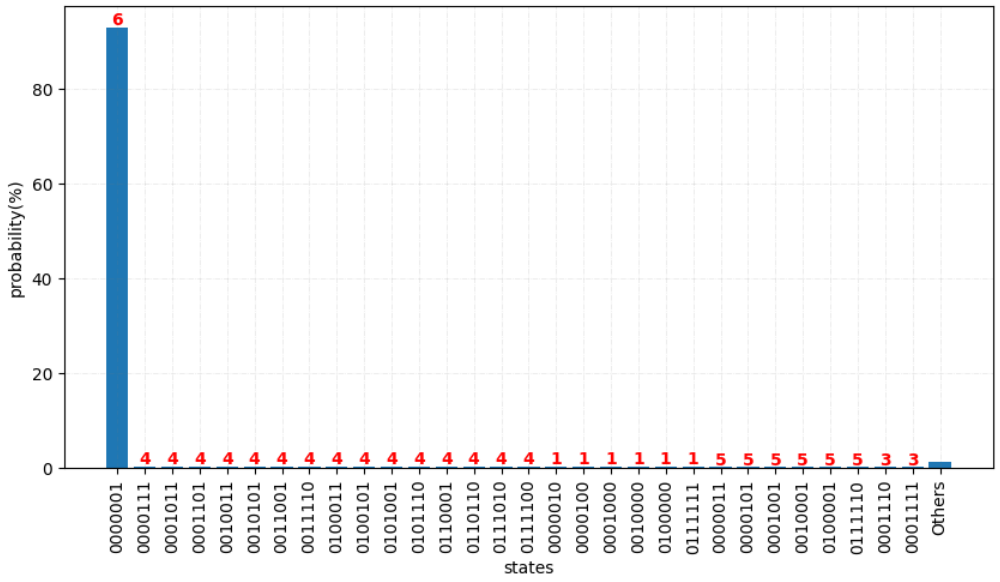
Figure 4.4: An unweighted Star graph created with seven nodes and six edges.

Star graphs are used to model computer and biological networks, as well as data structures, and are instrumental in the design of hierarchical tree structures [84]. These graphs are explicitly referenced in the QAOA literature, for example, Herrman et al. [75] employ them to evaluate ma-QAOA, while Ruan et al. [85] use Star graphs as a constraint-encoding operator to connect all feasible solutions with a central initial solution, enabling the incorporation of arbitrary constraints.

The following charts present two tests, sa-QAOA as presented by Farhi et al. [6] (Figure 4.5), and the first higher-order QAOA, QAOA-CD (Figure 4.6). In both cases, the circuit is illustrated with $p = 1$, along with a bar chart displaying the output state probabilities for $p = 3$.

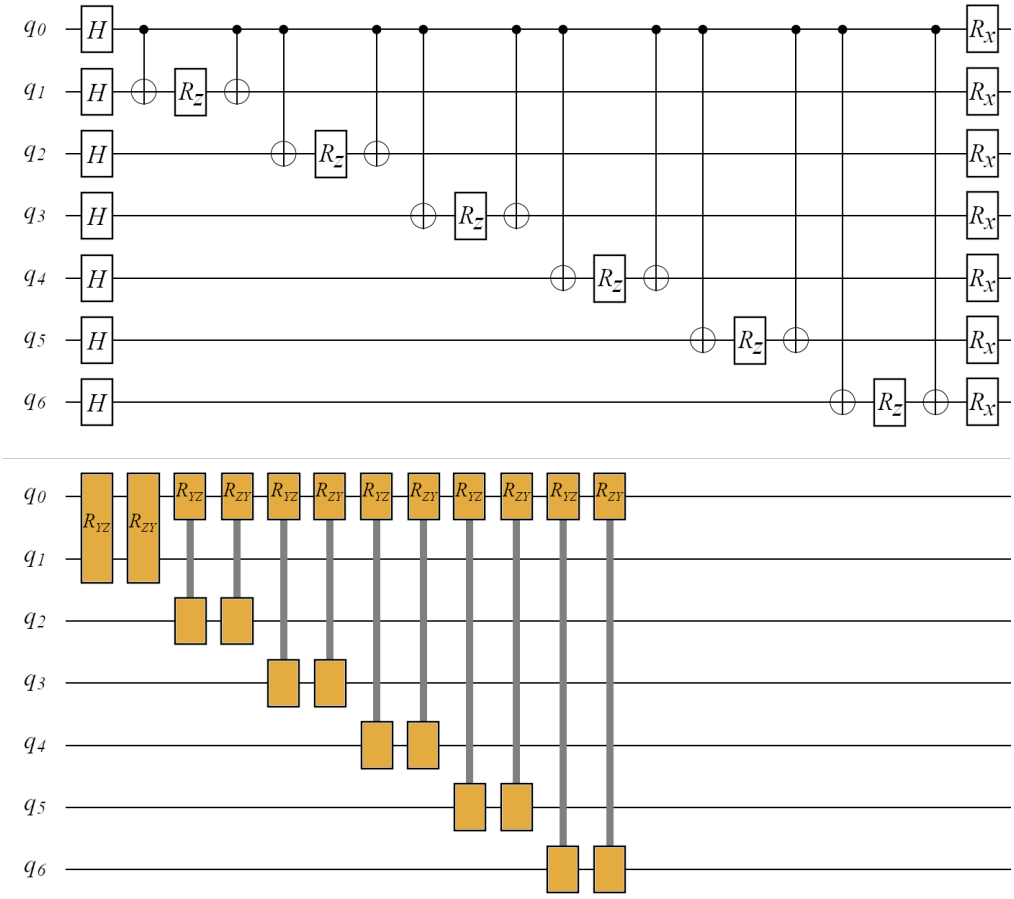


(a) sa-QAOA circuit representation for $p = 1$.

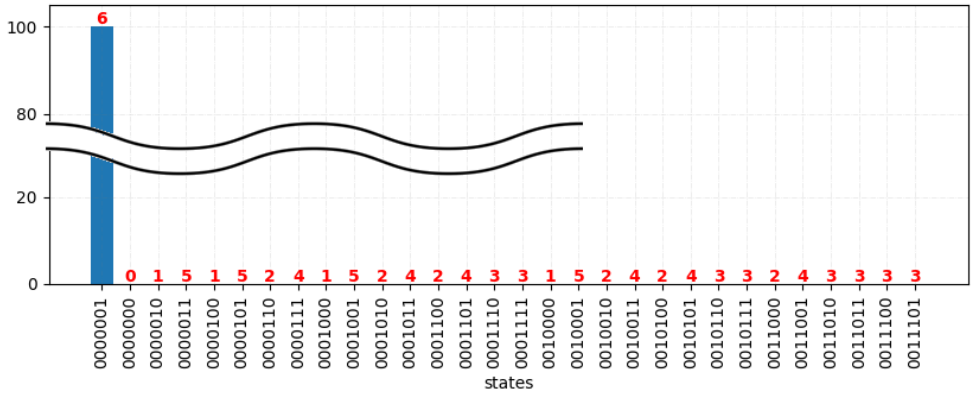


(b) Bar chart of state outcome probabilities for $p = 3$ for sa-QAOA.

Figure 4.5: Circuit diagram and probability distribution bar chart for sa-QAOA applied to the Star graph shown in Figure 4.4. The result gives a Max-Cut value of **six** for **92.88%** of the time with **336** number of function evaluations for $p = 3$.



(a) QAOA-CD circuit representation for $p = 1$ (orange gates denote the *higher order* gates).



(b) Bar chart of state outcome probabilities for $p = 3$ for QAOA-CD.

Figure 4.6: Circuit diagram and probability distribution bar chart for QAOA-CD applied to the Star graph shown in Figure 4.4. The result gives a Max-Cut value of **six** for **99.96%** of the time with **1340** number of function evaluations for $p = 3$.

Table 4.2: Comparing QAOA with First Higher-Order QAOA (QAOA-CD)

Figure of Merit	**QAOA**	**QAOA-CD**
AR	0.79	0.97
AP	92.88%	99.96%
nfev	336	1340

Table 4.2 shows that the first higher-order QAOA returns a slightly better result than QAOA when one compares both algorithms using AP. In this case, there is also a 22.8% difference in the AR, which is significant. The first higher-order QAOA still requires a high number of Number of Function Evaluations (nfev), necessitating more computational resources. When applied to star graphs, the first higher-order QAOA may be a candidate algorithm in the NISQ era. Again, one must also consider that in a NISQ device, the first higher-order QAOA requires a higher circuit depth, which makes it more susceptible to errors during execution, further limiting its practicality.

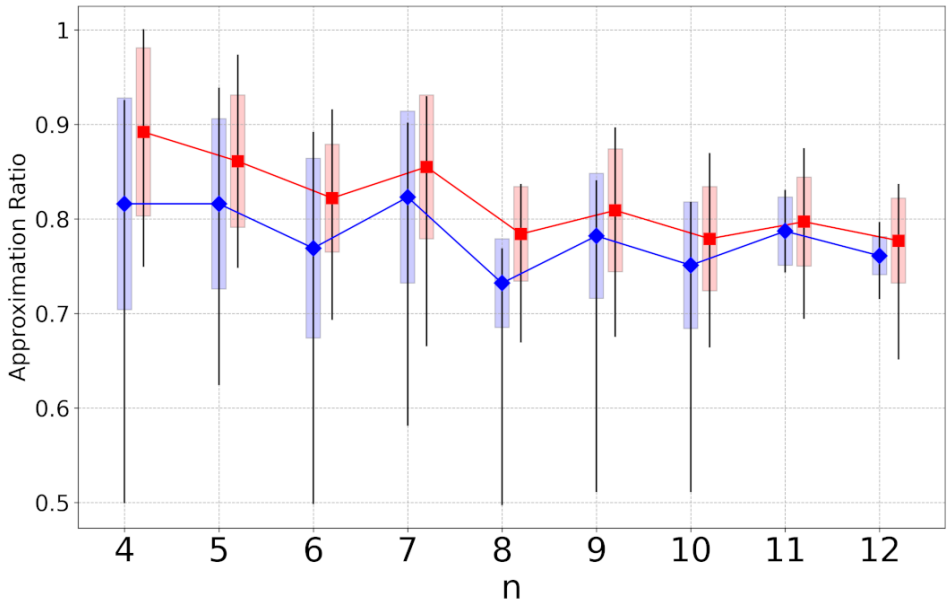
4.1.5 | Comparing QAOA and QAOA First Higher-Order

Table 4.3 presents the average number of function evaluations (*nfev*) (Section 3.5.2.4) obtained for sa-QAOA and the first higher-order QAOA (QAOA-CD) across forty-four graphs of different node sizes n. A total of 3200 tests were executed on these graphs that were made of four different configurations: Erdős-Rényi Graphs (Section 4.1.3, Cycle Graphs (Section 4.3.2), Random 3-Regular Graphs, and Star Graphs (Section 4.1.4).

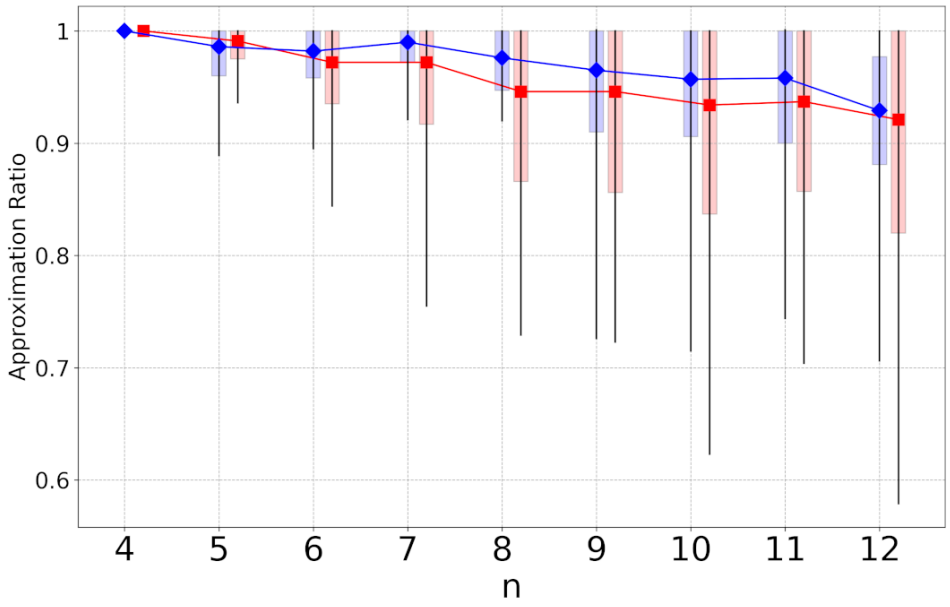
Refer to the explanations provided at the beginning of Section 4 for details on how to interpret the data in both the table and charts. An AR of 1 (shown without a subscript) indicates that the optimal value was found, as these tests aimed to maximise the cost function (Section 3.5.2.1), thereby achieving the optimal Max-Cut.

Table 4.3: Average number of function evaluations obtained after testing different unweighted graphs.

n	QAOA Variant	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
4	sa-QAOA	$58_{0.82}$	190	246	275	304
4	QAOA-CD	$114_{0.89}$	779	1318	2232	1312
5	sa-QAOA	$58_{0.82}$	$190_{0.92}$	$359_{0.96}$	$667_{0.97}$	$903_{0.99}$
5	QAOA-CD	$122_{0.86}$	$528_{0.92}$	$1107_{0.95}$	$2930_{0.98}$	$3236_{0.99}$
6	sa-QAOA	$49_{0.77}$	$187_{0.88}$	$430_{0.94}$	$785_{0.97}$	$1212_{0.98}$
6	QAOA-CD	$110_{0.82}$	$476_{0.92}$	$1185_{0.94}$	$1980_{0.97}$	$3617_{0.97}$
7	sa-QAOA	$48_{0.82}$	$198_{0.94}$	$502_{0.98}$	$678_{0.98}$	$1028_{0.99}$
7	QAOA-CD	$110_{0.86}$	$500_{0.94}$	$1175_{0.94}$	$2040_{0.97}$	$3182_{0.97}$
8	sa-QAOA	$49_{0.73}$	$182_{0.84}$	$452_{0.9}$	$866_{0.95}$	$1169_{0.98}$
8	QAOA-CD	$111_{0.78}$	$459_{0.87}$	$1086_{0.91}$	$2159_{0.93}$	$3498_{0.95}$
9	sa-QAOA	$56_{0.78}$	$190_{0.89}$	$438_{0.94}$	$820_{0.96}$	$1197_{0.97}$
9	QAOA-CD	$114_{0.81}$	$520_{0.91}$	$1036_{0.92}$	$2306_{0.94}$	$4360_{0.95}$
10	sa-QAOA	$47_{0.75}$	$176_{0.86}$	$439_{0.89}$	$977_{0.92}$	$1323_{0.96}$
10	QAOA-CD	$120_{0.78}$	$446_{0.85}$	$1104_{0.89}$	$1980_{0.92}$	$2924_{0.93}$
11	sa-QAOA	$55_{0.79}$	$224_{0.88}$	$518_{0.91}$	$848_{0.95}$	$1742_{0.96}$
11	QAOA-CD	$114_{0.8}$	$482_{0.89}$	$1136_{0.91}$	$2233_{0.95}$	$3721_{0.94}$
12	sa-QAOA	$55_{0.76}$	$225_{0.86}$	$534_{0.88}$	$969_{0.91}$	$1453_{0.93}$
12	QAOA-CD	$126_{0.78}$	$484_{0.86}$	$1229_{0.89}$	$2016_{0.91}$	$3080_{0.92}$



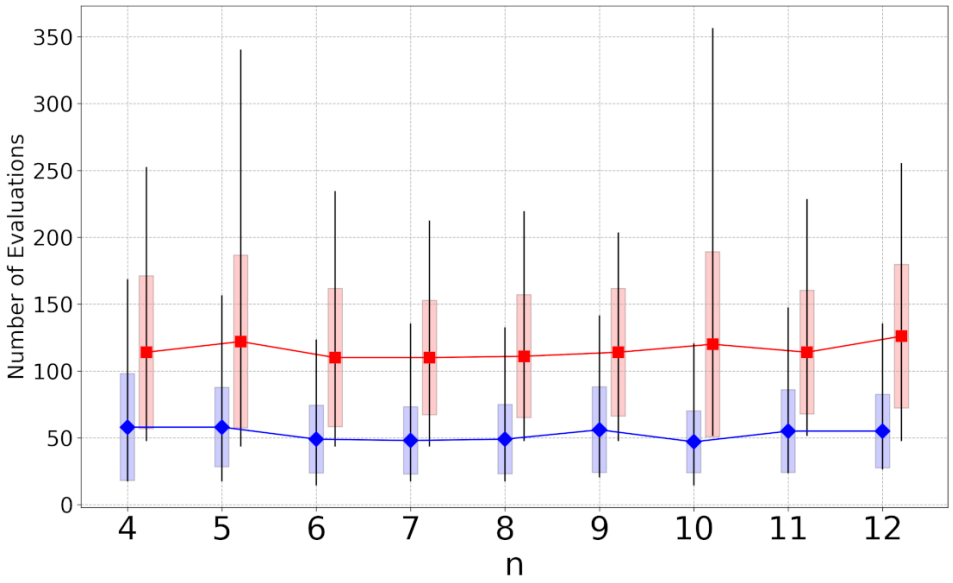
(a) Approximation Ratio for Max-Cut tests on graphs comparing sa-QAOA against QAOA-CD with $p = 1$.



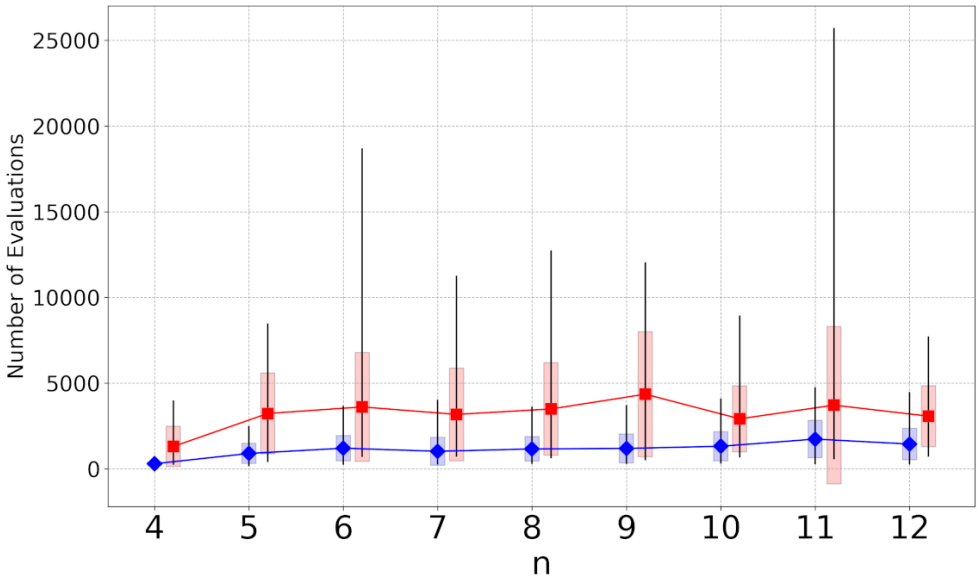
(b) Approximation Ratio for Max-Cut tests on graphs comparing sa-QAOA against QAOA-CD with $p = 5$.

Figure 4.7: Approximation Ratio for Max-Cut tests on graphs comparing sa-QAOA against QAOA-CD with $p \in \{1, 5\}$.

Legend: ■ QAOA-CD, ◆ sa-QAOA.



(a) Number of function evaluations for Max-Cut tests on graphs comparing sa-QAOA against QAOA-CD with $p = 1$.



(b) Number of function evaluations for Max-Cut tests on graphs comparing sa-QAOA against QAOA-CD with $p = 5$.

Figure 4.8: Number of function evaluations for Max-Cut tests on graphs comparing sa-QAOA against QAOA-CD with $p \in \{1, 5\}$.

Legend: ■ QAOA-CD, ◆ sa-QAOA.

QAOA of the first higher-order using the Zassenhaus expansion yields a better AR for one layer ($p = 1$) on the Max-Cut problem on the graphs that were examined. Nonetheless, QAOA-CD needs more resources than QAOA, at least for the graphs in the tests. Moreover, as the number of layers (p) increases, Approximation Ratio (AR) for sa-QAOA outperforms QAOA-CD while keeping a lower Number of Function Evaluations (nfev), therefore using less resources. Actually, Chandarana et al. [60] do say that *the QAOA-CD turns out to be a preferable algorithm for circuits of shorter depth.*

4.2 | Job-Shop Scheduling Problem

The Job-Shop Scheduling Problem (JSSP) (Section 3.3.9.2) targets to find an optimal solution to distribute jobs in a factory or manufacturing plant. After building the cost function (equation (3.40)) as a QUBO model, it is then converted into an Ising Hamiltonian (Section 3.3.2) and thereafter encoded into a quantum circuit. Each job is subject to several constraints, including a required sequence and the restriction that no two jobs may use the same machine simultaneously.

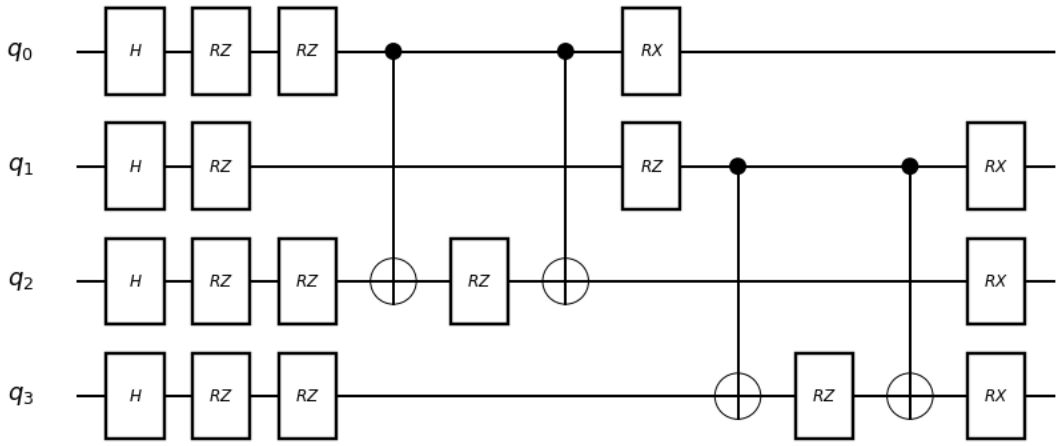
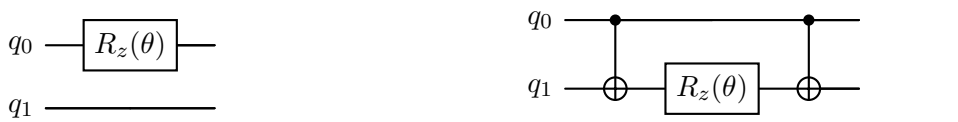


Figure 4.9: QAOA Quantum Circuit for the JSSP QUBO problem (Section 3.3.9.2), for $p = 1$.

Figure 4.9 illustrates a QAOA Quantum Circuit for a JSSP QUBO problem (Section 3.3.9.2) with $p = 1$. After executing the QAOA Circuit with $p = 5$, the output state is a quantum state whose amplitudes encode the probabilities of different candidate schedules. In this case, the bit strings 1001 or 0110 emerge as the most likely valid schedules. Each bit in the bit string corresponds to a binary variable x_{sm}, which can be decoded to represent a specific job-machine assignment.

4.2.1 | Quadratic Unconstrained Binary Optimisation

The Hamiltonian encodes both the objective function and constraints. It is built using Pauli operators with single-qubit terms (I and Z) to represent penalties or rewards for assigning jobs to machines, and two-qubit terms ($Z \otimes Z$) to enforce constraints by penalising invalid schedules. Both operators are shown in Figure 4.10.



(a) Z Rotation Gate $R_Z(\theta)$ on qubit 0.

(b) ZZ Rotation Gate $R_{ZZ}(\theta)$ on qubits 0 and 1.

Figure 4.10: Single and two-qubit rotational Z gates.

4.2.2 | Polynomial Unconstrained Binary Optimisation

Next, the the JSSP was expanded to include also time T as a parameter, thus having

- s: SKU (product, or job index) representing the specific product or job being scheduled ($1 \leq s \leq S$, where S is the total number of products/jobs).
- m: Machine Index, representing the machine on which a product/job is processed ($1 \leq m \leq M$, where M is the total number of machines).
- t: Time Index, representing the time slot during which a job is processed ($1 \leq t \leq T$, where T is the total number of time slots).

We can now include the observable term $Z_{s,m,t}$, indicating whether job s is scheduled on machine m at time t. The function **index**(s, m, t) (Listing 4) maps (s, m, t) to a unique index for the corresponding qubit:

```
def index(s, m, t):
    return ((s - 1) * (M * T) + (m - 1) * T + t) - 1
```

Listing 4: Mapping function for indexing.

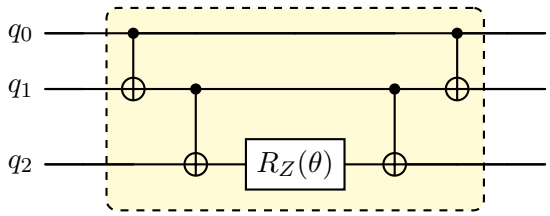
The following constraints were programmed into the Hamiltonian:

- A job cannot be on more than two machines at the same time:
  $\frac{1}{8}(1 - Z_{s,1,t} - Z_{s,2,t} - Z_{s,3,t} + Z_{s,1,t}Z_{s,2,t} + Z_{s,1,t}Z_{s,3,t} + Z_{s,2,t}Z_{s,3,t} - Z_{s,1,t}Z_{s,2,t}Z_{s,3,t})$

This constraint applies to all SKUs s and periods t, with m ranging over machines. This may result from factory constraints; for example, during the production of plastic injection products, the company has only two moulds for the SKU and, therefore, cannot produce the same item on three machines simultaneously.

- Two jobs cannot share a machine/time:
  $\frac{1}{4}(1 - Z_{s,m,t} - Z_{s',m,t} + Z_{s,m,t}Z_{s',m,t})$
  This constraint iterates over pairs of SKUs s and SKUs s', ensuring no overlap for the same machines m and period t.

In this JSSP QAOA example, we can see a three-qubit interaction ($Z_{s,1,t}Z_{s,2,t}Z_{s,3,t}$), and this shows the need for polynomial interaction between the qubits in quantum circuits using these type of JSSP. Figure 4.11 shows a circuit that can present this interaction.



$R_{Z_0Z_1Z_2}$ Gate

Figure 4.11: Three-qubit rotational Z gate.

The Rotational ZZZ gate $R_{ZZZ}(\theta)$ is defined as:

$$R_{ZZZ}(\theta) = e^{-i\frac{\theta}{2}(Z\otimes Z\otimes Z)} = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{i\frac{\theta}{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{i\frac{\theta}{2}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{-i\frac{\theta}{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{i\frac{\theta}{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-i\frac{\theta}{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{4.1}$$

Today's quantum computing hardware can execute single and two-qubit gate interactions [86]; various challenges arise to implementing three-qubit interactions or more. Various studies hint at the possibility of an efficient implementation of quantum gates between more than two qubits - one example is long-range Rydberg blockade

interactions [87]. Recently, more research is being done using quantum dot arrays [88], Ising-type interactions [89], and Superconducting Quantum Circuits [90], amongst others, to produce efficient hardware that can interact with three or more qubits. As observed in the study of the JSSP, including multi-qubit gates, those involving more than two qubits can enhance the efficiency of solving complex optimisation problems.

4.2.3 | PUBO to QUBO

In Section 3.3.5, we discussed how one could transform a PUBO problem into a QUBO problem by introducing ancillary qubits. Equation (3.18) can be used to transform a PUBO problem into a QUBO problem. In Section 3.3.6, we then calculated the maximum number of ancillary qubits that one would need (equation (3.23)). For example, in our tests, PUBO binary expression:

$$C(x) = -x_0x_1x_2 + x_0x_3 - x_2x_4x_5$$

having three-term interactions, results in

$$C(x) = 6x_6+2x_0x_1-4x_0x_6-4x_1x_6-x_2x_6+x_0x_3+6x_7+2x_4x_5-4x_4x_7-4x_5x_7-x_2x_7.$$

This result has a maximum of two-term interactions, and besides adding ancillary qubits, it increases circuit depth, which is not ideal for NISQ devices.

4.3 | Comparing variants of CUBO QAOA

This section presents a case study analysis of k-uniform hypergraphs. In a k-uniform hypergraph, each hyperedge connects exactly k vertices. In this analysis, we focus on the case where $k = 3$, thus having circuits with cubic interactions, and we examine two specific instances of Cubic Unconstrained Binary Optimisation (CUBO) problems: **path** 3-uniform hypergraphs and **cycle** 3-uniform hypergraphs.

These tests were conducted using BFGS and Powell classical optimisation algorithms (Section 3.5.3.1), each initialised with different angle parameters (Section 3.5.3.3), specifically $\frac{1}{p}$, $R(0 \to \frac{1}{p})$, $R(0 \to \frac{2\pi}{p})$, and TQA_{75} (Section 4.4.1).

4.3.1 | 3-Uniform Path Hypergraphs

We define a k-uniform path *sign-alternating* PUBO problem as follows.

$$C'(x) = \sum_{i=0}^{n-k} (-1)^i \cdot \prod_{j=0}^{k-1} x_{i+j} \tag{4.2}$$

where:

x_n	:	boolean variables
n	:	number of variables
k	:	number of connected terms

Since the terms x_n in equation (4.2) are boolean, then the cost function $C'(x)$ is referred to as a **boolean equation**.

These problems have been designed with *sign-alternating* formulas to introduce complexity and make them less trivial. Without the alternating signs, equation (4.2) becomes straightforward to maximise since there is only one maximum solution, i.e. setting each x_{i+j} term to 1, which will result in a maximum value of $n-k+1$ for the cost function $C'(x)$.

However, an alternating sign in the equation introduces some complexity in the optimisation. Setting $x_{i+j} = 1$ for a term might not be optimal as the expression could be negated by the $(-1)^i$ coefficient, introducing conflicting contributions to the total cost. From a mathematical perspective, this reformulates the problem from a linear sum of products into a polynomial with variable signs, thereby increasing its complexity. As a result, determining the function's maximum or minimum value requires more computational resources.

By setting $k = 3$, we specialise the path k-uniform sign-alternating PUBO problem into a path 3-uniform CUBO problem, which can be expressed as the following boolean equation:

$$\begin{aligned} C(x) &= \sum_{i=0}^{n-3} (-1)^i \cdot \prod_{j=0}^{2} x_{i+j} \\ &= \sum_{i=0}^{n-3} (-1)^i x_i x_{i+1} x_{i+2} \end{aligned} \tag{4.3}$$

where:

x_n	:	boolean variables
n	:	number of variables, or vertices in the hypergraph

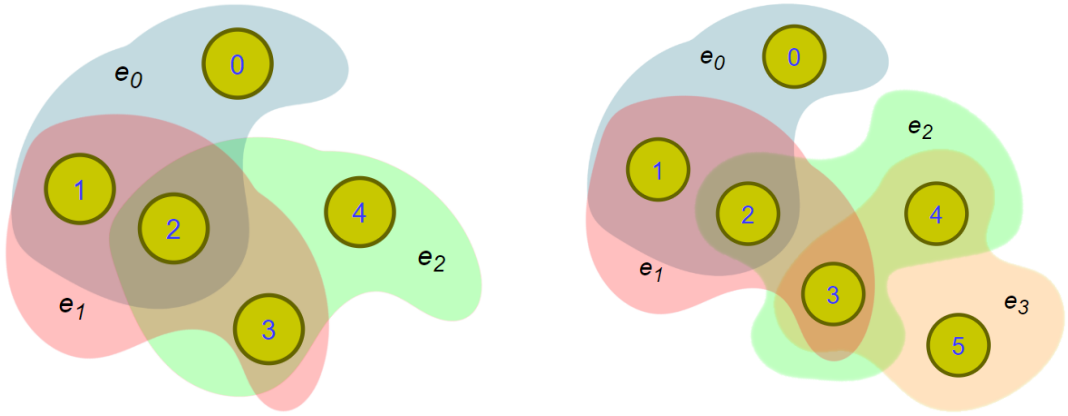
So, for example for $n = 5$, the boolean equation resolves to:

$$C'(x) = x_0x_1x_2 - x_1x_2x_3 + x_2x_3x_4 \tag{4.4}$$

while, for $n = 6$, the boolean equation resolves to:

$$C'(x) = x_0x_1x_2 - x_1x_2x_3 + x_2x_3x_4 - x_3x_4x_5 \tag{4.5}$$

Figure 4.12 shows the hypergraphs for the cost functions listed in equation (4.4) and equation (4.5).



(a) 5-vertex 3-uniform path hypergraph (b) 6-vertex 3-uniform path hypergraph

Figure 4.12: 5- and 6-vertex 3-uniform path hypergraphs representation.

Legend: ● *nodes* are the nodes from the boolean problem. The *coloured blobs* refer to hyperedges of the problem as defined by equation (4.3)

To develop the QAOA algorithm to find the maxima, we must derive the Hamiltonian, a spin problem for our hypergraph. Equation (4.3) captures the interactions amongst ternary relationships or dependencies of boolean variables. To transform this problem from a **boolean problem** into a **spin problem**, we substitute x_i with $\frac{1-Z_i}{2}$ (Section 3.1.2). The CUBO boolean problem will then be transformed into the following sign-alternating spin problem.

$$C(z) = \sum_{i=0}^{n-3} (-1)^i \frac{1 - Z_i}{2} \frac{1 - Z_{i+1}}{2} \frac{1 - Z_{i+2}}{2} \tag{4.6}$$

where:

Z_n	:	Pauli-Z gates
n	$\in$	$\mathbb{N}$, the number of variables

which can be simplified to:

$$
\begin{aligned}
C(z) \;=\; & \tfrac{1}{8}\sum_{i=0}^{n-3}(-1)^i\cdot(-Z_i - Z_{(i+1)} - Z_{(i+2)} && \text{(single term interaction)}\\
& \quad + Z_iZ_{(i+1)} + Z_iZ_{(i+2)} + Z_{(i+1)}Z_{(i+2)} && \text{(two term interaction)}\\
& \quad - Z_iZ_{(i+1)}Z_{(i+2)}) && \text{(three term interaction)}
\end{aligned}
$$

$$
\begin{array}{rccccccc}
 & \multicolumn{3}{c}{\text{(single term expansion)}} & \multicolumn{3}{c}{\text{(two term expansion)}} & \text{(three term expansion)}\\
= & \tfrac{1}{8}(-Z_0 & \cancel{-Z_1} & \cancel{-Z_2} & +Z_0Z_1 & +Z_0Z_2 & \cancel{+Z_1Z_2} & -Z_0Z_1Z_2\\
 & \cancel{+Z_1} & \cancel{+Z_2} & \cancel{+Z_3} & \cancel{-Z_1Z_2} & +Z_1Z_3 & \cancel{+Z_2Z_3} & +Z_1Z_2Z_3\\
 & -Z_2 & \cancel{-Z_3} & \cancel{-Z_4} & \cancel{+Z_2Z_3} & +Z_2Z_4 & \cancel{+Z_3Z_4} & -Z_2Z_3Z_4\\
 & +Z_3 & \cancel{+Z_4} & \cancel{+Z_5} & \cancel{-Z_3Z_4} & -Z_3Z_5 & \cancel{-Z_4Z_5} & +Z_3Z_4Z_5\\
 & & \vdots & & & \vdots & & \vdots\\
 & -Z_{n-4} & \cancel{-Z_{n-3}} & \cancel{-Z_{n-2}} & \cancel{+Z_{n-4}Z_{n-3}} & +Z_{n-4}Z_{n-2} & \cancel{+Z_{n-3}Z_{n-2}} & -Z_{n-4}Z_{n-3}Z_{n-2}\\
 & +Z_{n-3} & \cancel{+Z_{n-2}} & +Z_{n-1} & \cancel{-Z_{n-3}Z_{n-2}} & -Z_{n-3}Z_{n-1} & -Z_{n-2}Z_{n-1} & +Z_{n-3}Z_{n-2}Z_{n-1})
\end{array}
$$

$$
\begin{aligned}
= \; & \tfrac{1}{8}\left(-Z_0 + \left(\sum_{i=0}^{n-3}(-1)^{i+1}Z_i\right) + (-1)^nZ_{n-1} && \text{(single term interaction)}\right.\\
& +Z_0Z_1 + \left(\sum_{i=0}^{n-3}(-1)^iZ_iZ_{(i+2)}\right) - (-1)^nZ_{n-2}Z_{n-1} && \text{(two term interaction)}\\
& \left. + \left(\sum_{i=0}^{n-3}(-1)^{i+1}Z_iZ_{(i+1)}Z_{(i+2)}\right)\right) && \text{(three term interaction)}
\end{aligned}
\tag{4.7}
$$

where:

Z_i	:	Pauli-Z gates
n	$\in$	$\mathbb{N}$, the number of variables

Therefore, the number of terms required for:

- One Z-rotation gate is $2 + (n-3) - (2) + (1)^*$, i.e. n-2 terms.
- Two Z-rotation gates is $2 + (n-3) - (0) + (1)^*$, i.e. n terms.
- Three Z-rotation gates is $(n-3) - (0) + (1)^*$, i.e. n-2 terms.

The total count of terms in the above cost function, i.e., $(n-2) + (n) + (n-2)$, which results in $(3n-4)$ terms. To this, we add n for the number of terms in the mixer operator. Therefore, the total number of terms for our cost function in spin Hamiltonian is:

$$N_{terms} = 4(n-1) \tag{4.8}$$

*We add one since we are using 0-based indexes convention in equations.

Figure 4.13 represents the graph resulting from the transformation of the 5-vertex *boolean* hypergraph (Figure 4.12) into its respective **spin** hypergraph, adding **loops**, for single Z-terms, and **edges** for two and three Z-terms as per equation (4.7).

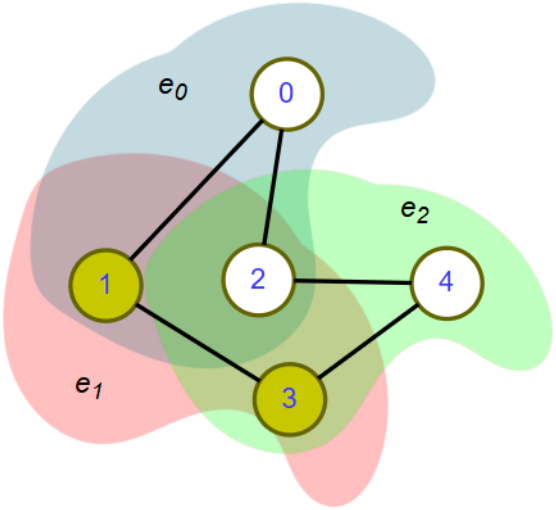


Figure 4.13: 5-vertex spin hypergraph representation. The coloured areas refer to 3-parameter interactions, cells connected by lines refer to 2-parameter interactions, and white cells refer to 1-spin parameters.

This cost function, equation (4.9), can now be designed as a quantum circuit, shown in Figure 4.14.

$$\begin{aligned} C(z) = \frac{1}{8}(&Z_0Z_1Z_2 - Z_0Z_1 - Z_0Z_2 + Z_0 - Z_1Z_2Z_3 + Z_1Z_3 + Z_2Z_3Z_4 \\ &- Z_2Z_4 + Z_2 - Z_3Z_4 + Z_4 - 1) \end{aligned} \tag{4.9}$$

This circuit has a circuit depth of 32 (excluding the Hadamard gates) and 22 CNOTS, none executed in parallel.

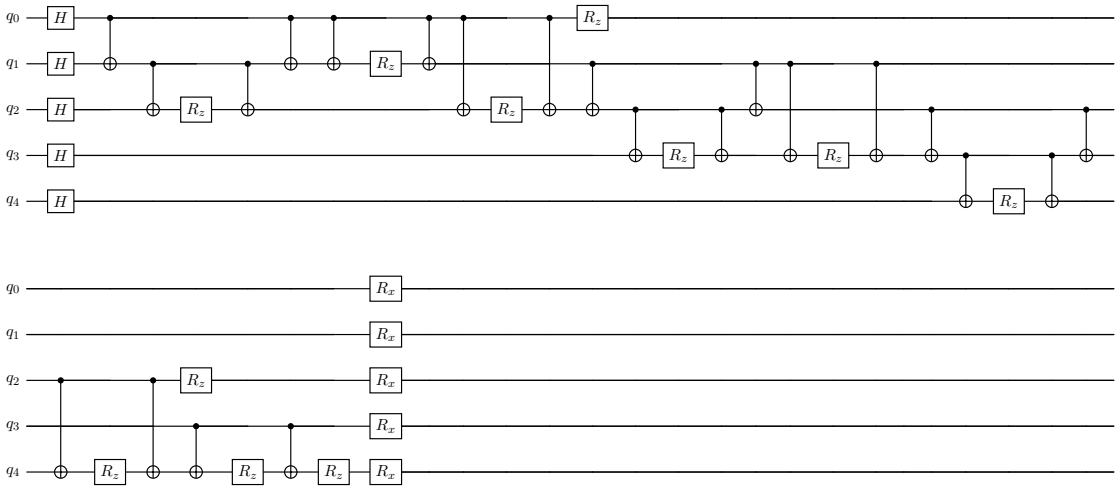


Figure 4.14: PUBO Quantum Circuit for a 5-vertex, $k = 3$ hypergraph with $p = 1$ layer. All R_z gates are Z-rotational gates with a given parameter γ ($R_z(\gamma)$), while all R_x gates are X-rotational gates with a given parameter β ($R_x(\beta)$).

For $p = 3$, the PUBO circuit results in an Approximation Ratio of **0.94**.

4.3.1.1 | PUBO to QUBO Circuit

Using equation (3.18), we can transform the PUBO Hamiltonian in equation (4.9) to QUBO Hamiltonian in equation (4.10) and represent it in a Quantum Circuit, as shown in Figure 4.15.

$$\begin{aligned} C'(z) = &\frac{1}{2}Z_0Z_1 - Z_0Z_5 + \frac{1}{2}Z_0 + \frac{1}{2}Z_1Z_2 - Z_1Z_5 - Z_1Z_6 + Z_1 - \frac{1}{4}Z_2Z_5 \\ &- Z_2Z_6 - \frac{1}{4}Z_2Z_7 + Z_2 + \frac{1}{4}Z_3Z_4 + \frac{1}{4}Z_3Z_6 - Z_3Z_7 + \frac{1}{4}Z_3 - Z_4Z_7 \\ &+ \frac{1}{2}Z_4 - \frac{3}{4}Z_5 - 1\frac{1}{4}Z_6 - \frac{3}{4}Z_7 + 4\frac{1}{4} \end{aligned} \tag{4.10}$$

This circuit can be reorganised with a circuit depth of 28 to 34 (excluding the Hadamard gates and depending on the parallelism involved) and 24 CNOTS, eight of which can be executed in parallel with another eight CNOTS, reducing the CNOT depth to 16.

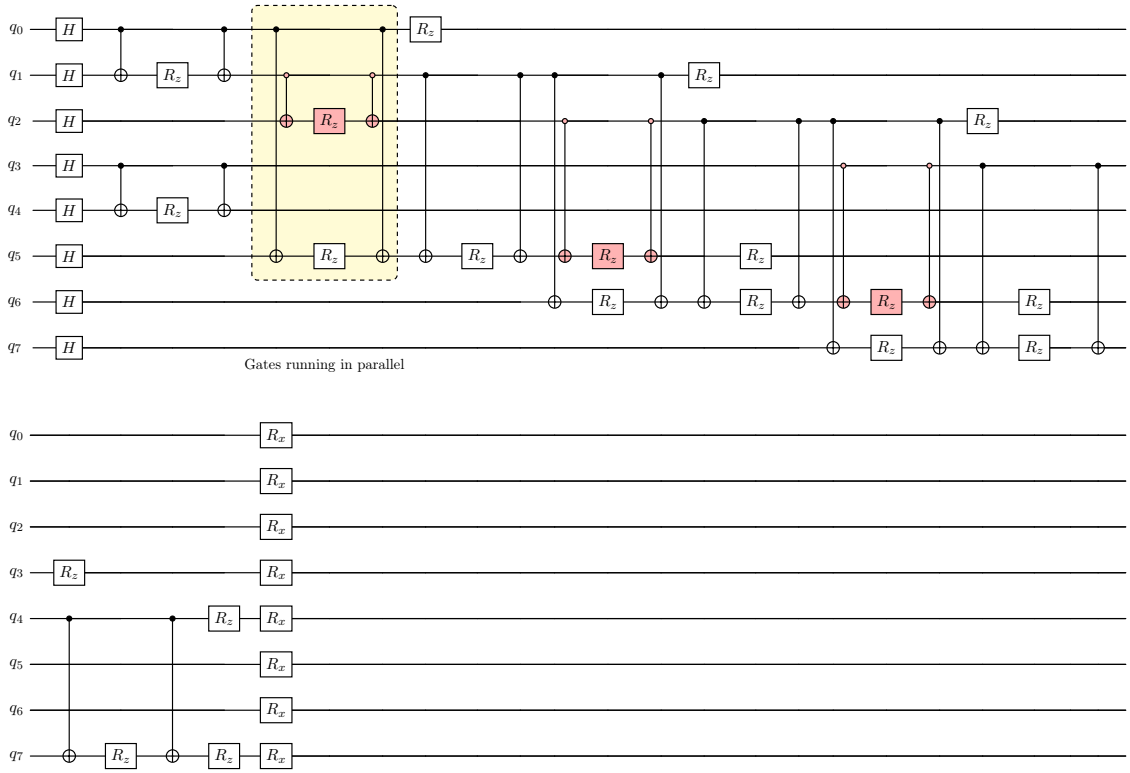
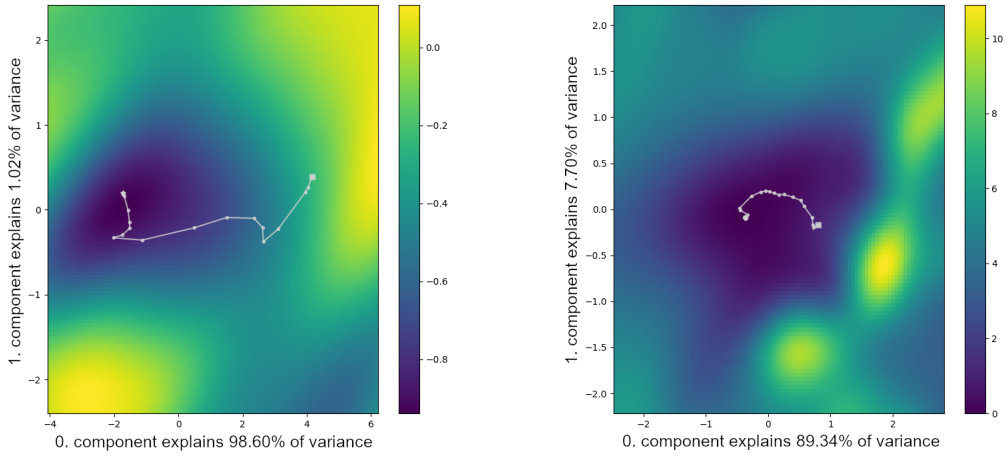


Figure 4.15: QUBO Quantum Circuit for a 5-vertex, $k = 3$ hypergraph with $p = 1$ layer. All R_z gates are Z-rotational gates with a given parameter γ ($R_z(\gamma)$), while all R_x gates are X-rotational gates with a given parameter β ($R_x(\beta)$). The gates in the yellow area show an example of gates running in parallel and superimposed in the circuit design. Other gates run parallel within the circuit, contributing to the overall quantum operations and simultaneously influencing the computation.

For $p = 3$, the QUBO circuit yields an exceptionally low **0.01** Approximation Ratio, indicating failure to find a minima due to barren plateau in the QAOA (Figure 4.16).

We immediately see that we require three ancillary qubits to implement the QUBO circuit, which follows the rule defined by equation (3.23) that states the maximum number of required ancillary qubits - we need less since this is a sign-alternating version of equation (4.3).

The landscape for the two algorithms, illustrated in Figure 4.16, shows how the PUBO circuit converges to a minimum, while the QUBO circuit ends up on a barren plateau.



(a) PUBO (b) QUBO

Figure 4.16: Cost landscape scans of the QAOA PUBO Hamiltonian (a) and QUBO Hamiltonian (b) with $p = 3$, also showing the path taken by the sa-QAOA algorithm as it tries to find the minima in each iteration. As one can see, the QUBO ends up in a barren plateau.

While the QUBO implementation can be more parallelised than the PUBO implementation and would thus require less depth, the number of ancillary qubits increases as **n** increases. As we saw in Section 4.3.1.1, the QUBO version does not yield a good AR when comparing it to the PUBO implementation.

4.3.1.2 | ma-QAOA Algorithm for 3-Uniform Path Hypergraphs

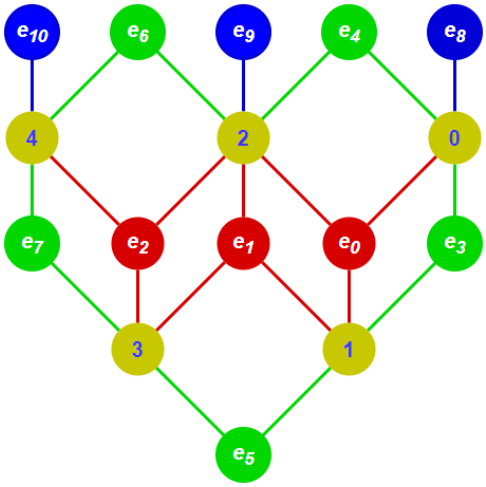
The ma-QAOA algorithm requires each rotational gate term to be assigned a separate angle parameter. Therefore, the number of angles required to run the ma-QAOA for the problem $N_{ma\angle} = 4(n-1)$ (equation (4.8)).

4.3.1.3 | am-QAOA Algorithm for 3-Uniform Path Hypergraphs

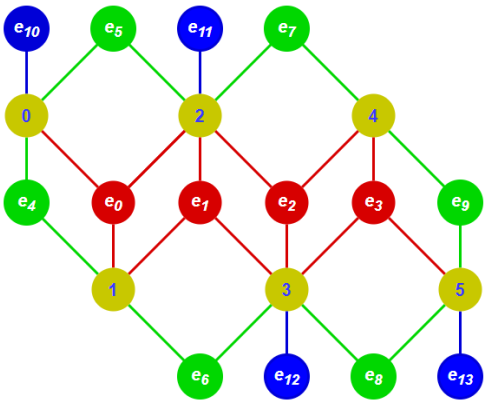
Since we are using the same problem, the cost function (equation (4.7)) remains the same; however, in am-QAOA, we reduce the number of angles by using the same angle for automorphic nodes and edges of the cost function.

To determine the automorphism of a hypergraph, we must first find its bipartite incidence graph (Section 3.2.3). By leveraging the automorphism of this graph, we can subsequently deduce the automorphism of the hypergraph. Figure 4.17 illustrates the bipartite incidence graphs of the 5- and 6-vertex spin hypergraphs (Figure 4.13). When

creating the bipartite incidence graph, we do not need to create a node for the hyperedge for the Z_i or Z_iZ_j nodes. However, creating a node for these edges simplifies our search for edge cycles and exploring automorphisms of hypergraphs. By representing each edge as a node, even trivial edges, we convert relationships into explicit entities that can be manipulated or examined more easily within the graph structure.



(a) Bipartite incidence graph for 5-vertex spin path hypergraph



(b) Bipartite incidence graph for 6-vertex spin path hypergraph

Figure 4.17: Bipartite incidence graph for the spin functions for the 5- and 6-vertex sign-alternating path hypergraphs.

Legend: ● *nodes* are the original nodes from our problem, ● *nodes* are the edges for single term interactions, ● *nodes* are the edges for two-term interactions, and ● *nodes* are the edges for three term interactions.

Due to the mirror image nature of the bipartite incidence graph of our problem, each node and edge form an isomorphism with the other side (mirror image) of the graph, giving us $(\frac{3n-4}{2},\ n \in \mathbb{N}_{even})$ or $(\frac{3n-1}{2},\ n \in \mathbb{N}_{odd})$ angles required to execute the am-QAOA. To get the total number of angles, equation (4.11), we need to add n for the mixer operator, for which we will use distinct angles for each R_x gate.

$$N_{\text{am}\angle} = \begin{cases} \frac{5}{2}n - 2 & \text{if } n \text{ is even} \\ \frac{5}{2}n - \frac{1}{2} & \text{if } n \text{ is odd} \end{cases} \tag{4.11}$$

where:

$n \in \mathbb{N}$ such that $n \geq 5$

This would be represented as a single equation as follows:

$$N_{am\angle} = \frac{5n + 3(n \bmod 2) - 4}{2} \tag{4.12}$$

where:

$n \in \mathbb{N}$ such that $n \geq 5$

4.3.1.4 | ka-QAOA Algorithm for 3-Uniform Path Hypergraphs

In ka-QAOA, we use just three angles (since $k = 3$) for the cost function (equation (4.7)). We use one angle for all $Z_iZ_jZ_k$ where $i < j < k < n$, another angle for Z_iZ_j where $i < j < n$ and another angle for Z_i where $i < n$. To this, we add n angles for the QAOA mixer, i.e. the size of the number of boolean parameters in the original problem. Therefore, the number of angles is:

$$N_{ka\angle} = n + 3 \tag{4.13}$$

where:

$n \in \mathbb{N}$

4.3.1.5 | Ratios of Different Angle Counts in QAOA

To illustrate the relationship between the number of angles in ma-QAOA and am-QAOA, we can plot their counts for $n \in \mathbb{N}, n \geq 5$ and examine their ratio, shown in equation (4.14).

$$Ratio_n = \begin{cases} \frac{5n-4}{8n-8} & \text{if } n \text{ is even} \\ \frac{5n-1}{8n-8} & \text{if } n \text{ is odd} \end{cases} \tag{4.14}$$

These indicate how the value of the original expression for the number of angles in am-QAOA changes in relation to the number of angles in ma-QAOA based on the parity of n. $n \mod 2$ (equation (4.12)) in the numerator introduces a condition that makes the ratio's value slightly vary between even and odd values of n, noticeable for smaller values of n. However, as n becomes large, its impact becomes negligible, and, therefore, the overall ratio approaches $\frac{5}{8}$.

$$Ratio_n \to \frac{5}{8} \tag{4.15}$$

On the other hand, the relation between ma-QAOA and ka-QAOA follows:

$$Ratio_n = \frac{n+3}{4n-4} \rightarrow \frac{1}{4} \tag{4.16}$$

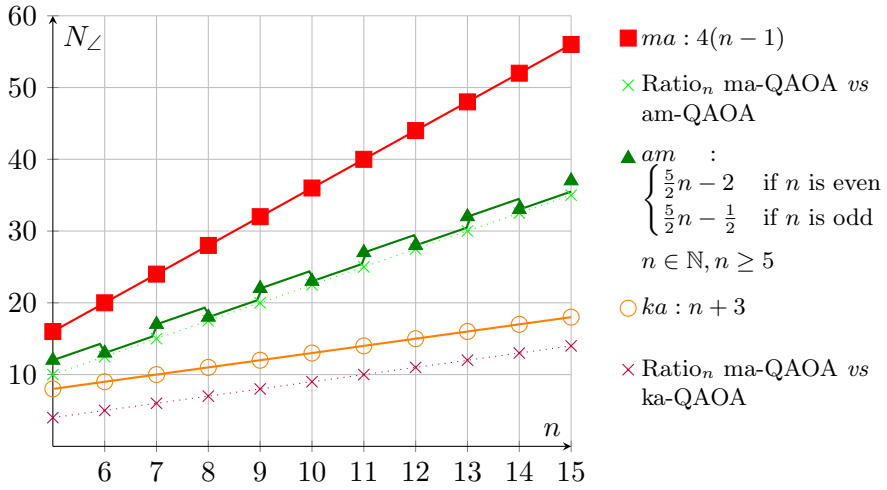


Figure 4.18: Ratio of angle count between ma-QAOA, am-QAOA and ka-QAOA for the 3-Uniform Path sign-alternating Hypergraphs.

Lines have been added to guide the eye.

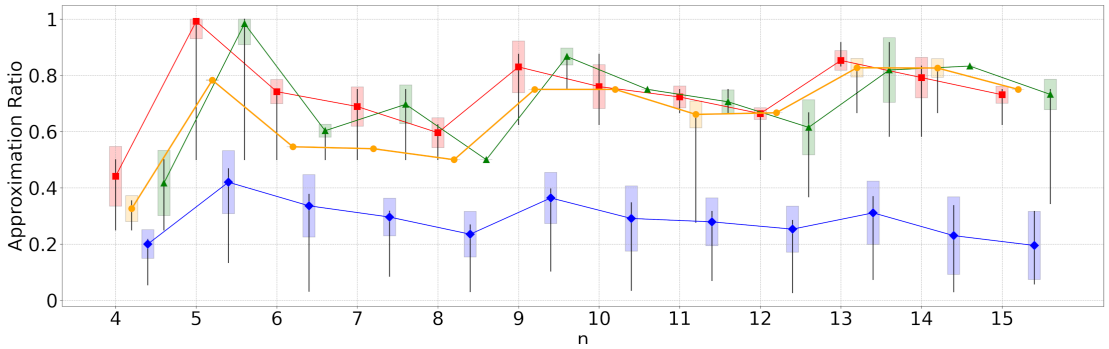
4.3.1.6 | QAOA Function Evaluations for 3-Uniform Path Sign-Alternating Hypergraphs

The Cost Function (equation (4.7)) was tested for graphs of size $n \in [4, 15]$ and with number of layers $p \in [1, 3]$ using sa-QAOA, ma-QAOA, am-QAOA and ka-QAOA.

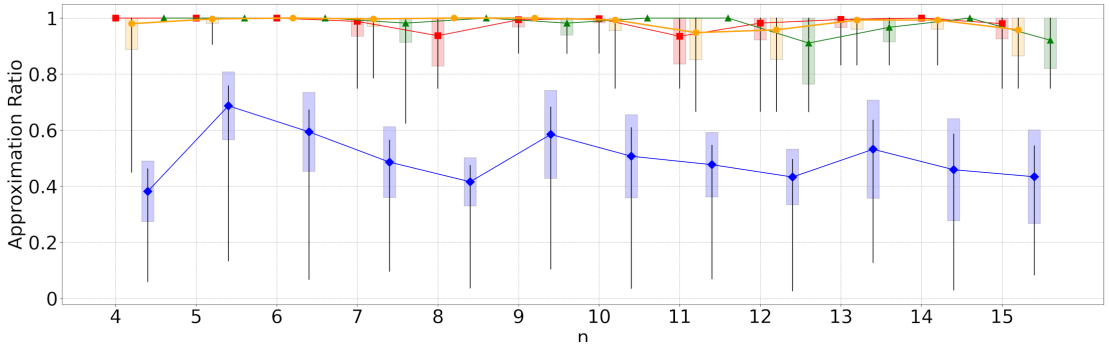
The following table and charts present the results for 3-Uniform Path Sign-Alternating Hypergraphs (Section 4.3.1). See the explanations at the beginning of Section 4 for guidance on interpreting the data in the table and charts.

Table 4.4: Average number of function evaluations for 3-Uniform Path Sign-Alternating Hypergraphs.

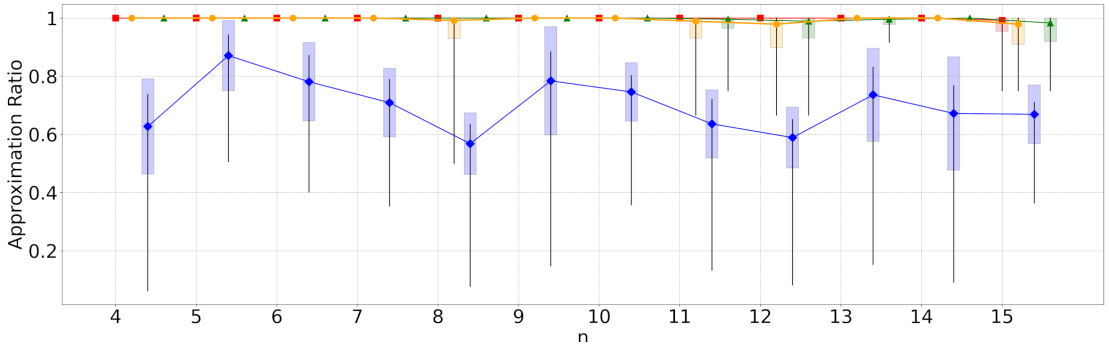
	sa-QAOA			ma-QAOA			am-QAOA			ka-QAOA		
n	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$
4	$46_{0.2}$	$210_{0.38}$	$485_{0.63}$	$770_{0.44}$	2526	4111	$645_{0.42}$	2045	3185	$445_{0.33}$	$1164_{0.98}$	1872
5	$59_{0.42}$	$195_{0.69}$	$469_{0.87}$	$1136_{0.99}$	3020	5447	$949_{0.98}$	2028	4391	$570_{0.78}$	1444	2660
6	$51_{0.34}$	$215_{0.59}$	$491_{0.78}$	$1543_{0.74}$	3749	7291	$1625_{0.6}$	2882	5039	$751_{0.55}$	1395	2776
7	$67_{0.3}$	$192_{0.49}$	$430_{0.71}$	$2014_{0.69}$	$5667_{0.99}$	9699	$1188_{0.7}$	$3822_{0.98}$	5994	$531_{0.54}$	1487	2628
8	$51_{0.23}$	$218_{0.42}$	$491_{0.57}$	$2618_{0.6}$	$7743_{0.94}$	13133	$1332_{0.5}$	3728	7071	$637_{0.5}$	1686	$3125_{0.99}$
9	$62_{0.36}$	$189_{0.59}$	$441_{0.78}$	$2699_{0.83}$	$7725_{0.99}$	15749	$1676_{0.87}$	$5096_{0.98}$	10974	$685_{0.75}$	2155	4244
10	$51_{0.29}$	$221_{0.51}$	$495_{0.75}$	$3794_{0.76}$	10861	19833	$1827_{0.75}$	5988	10427	$654_{0.75}$	$2089_{0.99}$	4268
11	$63_{0.28}$	$192_{0.48}$	$456_{0.64}$	$3632_{0.72}$	$11807_{0.93}$	22207	$2516_{0.71}$	7264	13352	$692_{0.66}$	$2410_{0.95}$	$4094_{0.99}$
12	$48_{0.25}$	$216_{0.43}$	$482_{0.59}$	$5055_{0.66}$	$13030_{0.98}$	25589	$2296_{0.62}$	$6810_{0.91}$	$12694_{0.99}$	$703_{0.67}$	$2605_{0.96}$	$4889_{0.98}$
13	$37_{0.31}$	$118_{0.53}$	$222_{0.74}$	$4690_{0.85}$	$15586_{0.99}$	28623	$2812_{0.82}$	$9560_{0.97}$	18161	$818_{0.83}$	$3284_{0.99}$	5263
14	$38_{0.23}$	$123_{0.46}$	$236_{0.67}$	$8252_{0.79}$	28590	60366	$3880_{0.83}$	11956	26379	$703_{0.83}$	$2046_{0.99}$	5096
15	$38_{0.19}$	$110_{0.43}$	$212_{0.67}$	$4982_{0.73}$	$17334_{0.98}$	$25170_{0.99}$	$3316_{0.73}$	$8715_{0.92}$	$14871_{0.98}$	$744_{0.75}$	$2321_{0.96}$	$4766_{0.98}$



(a) Approximation Ratio for 3-Uniform Path Sign-Alternating Hypergraphs with $p = 1$.



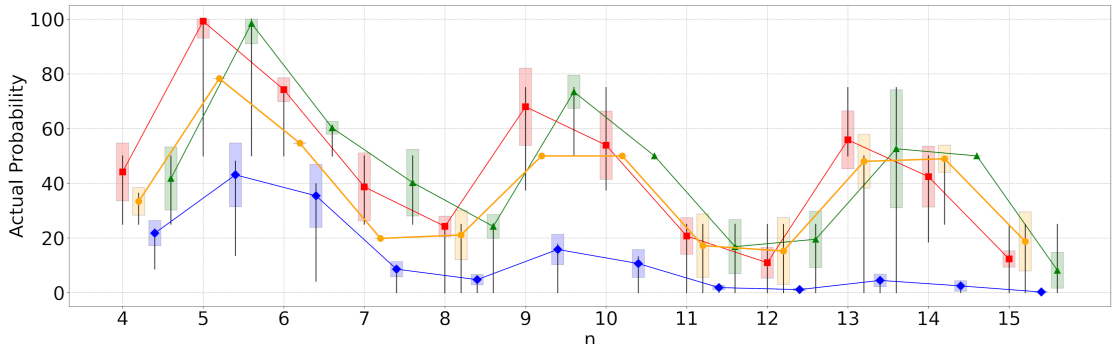
(b) Approximation Ratio for 3-Uniform Path Sign-Alternating Hypergraphs with $p = 2$.



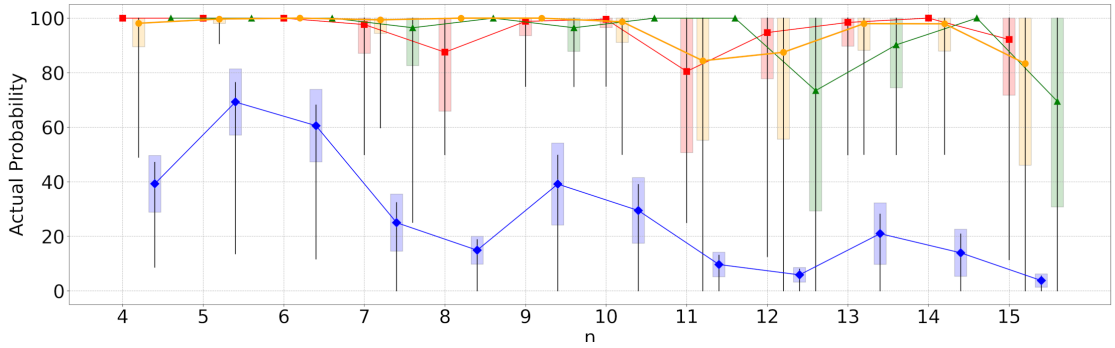
(c) Approximation Ratio for 3-Uniform Path Sign-Alternating Hypergraphs with $p = 3$.

Figure 4.19: Approximation Ratio for 3-Uniform Path Sign-Alternating Hypergraphs with $p \in \{1, 2, 3\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

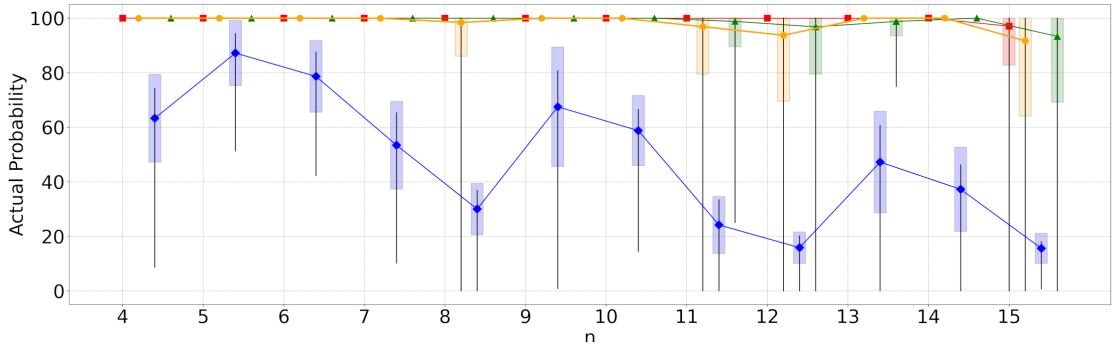
Legend: ■ Multi-angle, ● k-angle, ◆ Single angle, ▲ Automorphic angle.



(a) Actual Probability for 3-Uniform Path Sign-Alternating Hypergraphs with $p = 1$.



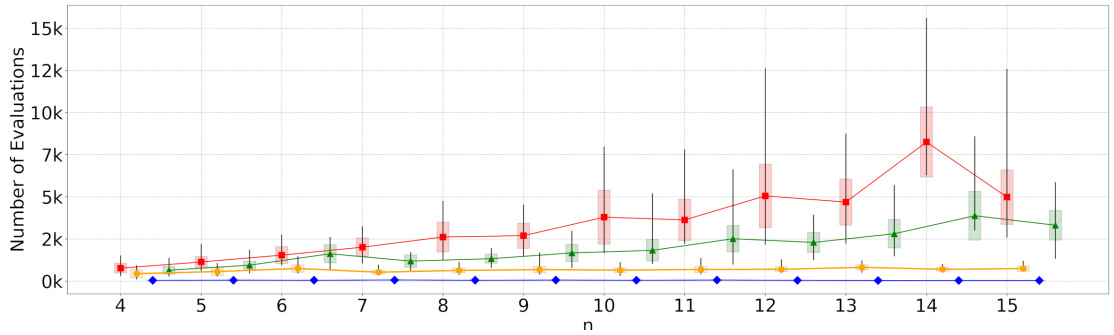
(b) Actual Probability for 3-Uniform Path Sign-Alternating Hypergraphs with $p = 2$.



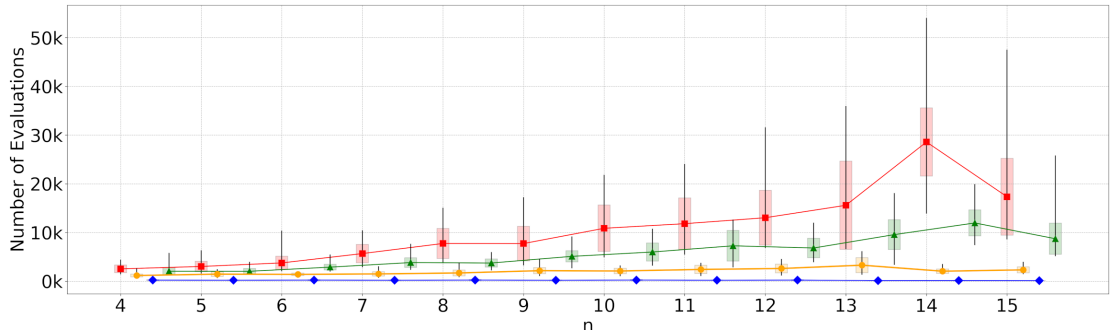
(c) Actual Probability for 3-Uniform Path Sign-Alternating Hypergraphs with $p = 3$.

Figure 4.20: Actual Probability for 3-Uniform Path Sign-Alternating Hypergraphs with $p \in \{1, 2, 3\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

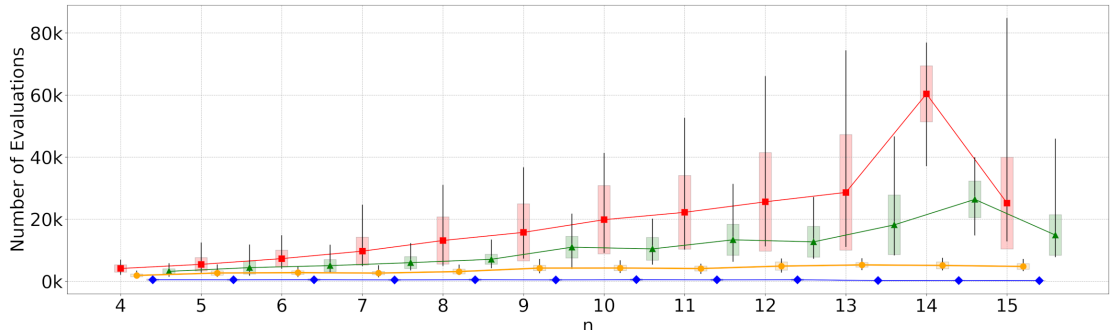
Legend: ■ Multi-angle, ● k-angle, ◆ Single angle, ▲ Automorphic angle.



(a) Number of function evaluations for 3-Uniform Path Sign-Alternating Hypergraphs with $p = 1$.



(b) Number of function evaluations for 3-Uniform Path Sign-Alternating Hypergraphs with $p = 2$.



(c) Number of function evaluations for 3-Uniform Path Sign-Alternating Hypergraphs with $p = 3$.

Figure 4.21: Number of function evaluations for 3-Uniform Path Sign-Alternating Hypergraphs with $p \in \{1, 2, 3\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

Legend: ■ Multi-angle, ● k-angle, ◆ Single angle, ▲ Automorphic angle.

Even at $p = 1$, experiments indicate that the Approximation Ratio (AR) of the ka-QAOA variant applied to 3-Uniform Path Sign-Alternating Hypergraphs is comparable to that achieved by am-QAOA and ma-QAOA, while providing a much better solution than sa-QAOA. Moreover, the Number of Function Evaluations (nfev) required to reach the optimisation required is considerably lower than that needed by am-QAOA and ma-QAOA. This reduction in *nfev* is noteworthy since this suggests that ka-QAOA can still obtain high-quality optimisations whilst utilising few resources in the process. This positions ka-QAOA as a suitable candidate during the NISQ era of quantum computation.

4.3.2 | 3-Uniform Cyclic Hypergraphs

The k-uniform cyclic *sign-alternating* PUBO boolean problem can be expressed as follows.

$$C(x) = \sum_{i=0}^{n-1} (-1)^i \cdot \prod_{j=0}^{k-1} x_{(i+j) \bmod n} \tag{4.17}$$

where:

x_n	:	boolean variables
n	:	number of terms, or vertices in the hypergraph
k	:	number of connected terms

As with the k-Uniform Path Hypergraphs (Section 4.3.1), these problems incorporate *sign-alternating* formulas to increase their complexity and ensure that they are not trivial. By setting $k = 3$, we specialise the cyclic k-uniform sign-alternating PUBO boolean problem into a cyclic 3-uniform sign-alternating CUBO boolean problem, which can be expressed as follows:

$$\begin{aligned} C'(x) &= \sum_{i=0}^{n-1} (-1)^i \cdot \prod_{j=0}^{2} x_{(i+j) \bmod n} \\ &= \sum_{i=0}^{n-1} (-1)^i x_i x_{(i+1) \bmod n} x_{(i+2) \bmod n} \end{aligned} \tag{4.18}$$

where:

x_n	:	boolean variables
n	:	number of variables, or vertices in the hypergraph

So, for example, for $n = 5$, the boolean equation resolves to equation (4.19) and is illustrated in Figure 4.22.

$$C'(x) = x_0x_1x_2 - x_1x_2x_3 + x_2x_3x_4 - x_3x_4x_0 + x_4x_0x_1 \tag{4.19}$$

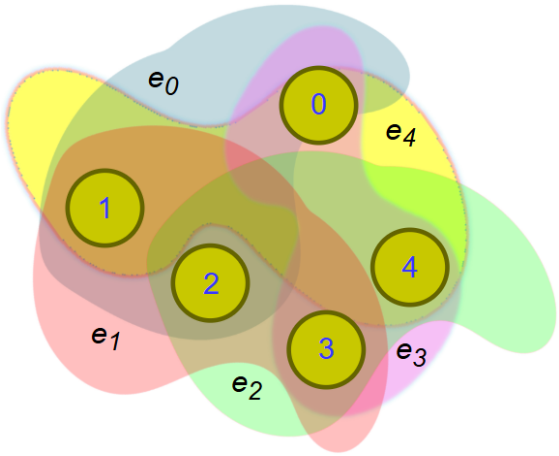


Figure 4.22: 5-vertex 3-uniform cyclic hypergraphs representation.

Legend: ● *nodes* are the nodes from the boolean problem. The *coloured blobs* refer to hyperedges of the problem as defined by equation (4.18)

while, for $n = 6$, the boolean equation resolves to:

$$C'(x) = x_0x_1x_2 - x_1x_2x_3 + x_2x_3x_4 - x_3x_4x_5 + x_4x_5x_0 - x_5x_0x_1 \tag{4.20}$$

4.3.2.1 | ma-QAOA Algorithm for 3-Uniform Cyclic Hypergraphs

Just as we did in Section 4.3.1.2, we substitute x_i with $\frac{1-Z_i}{2}$ in equation (4.18) to derive a spin problem for our hypergraph. The CUBO Boolean problem will thus be transformed into the following sign-alternating spin problem.

$$C(z) = \sum_{i=0}^{n-1}(-1)^i \frac{1-Z_i}{2}\frac{1-Z_{(i+1) \bmod n}}{2}\frac{1-Z_{(i+2) \bmod n}}{2} \tag{4.21}$$

where:

Z_n : Pauli-Z gates

This can be simplified to:

$$
\begin{aligned}
C(z) &= \tfrac{1}{8}\sum_{i=0}^{n-1}(-1)^i\cdot(-Z_i - Z_{(i+1) \bmod n} - Z_{(i+2) \bmod n} && \text{(single-term interaction)}\\
&= \quad +Z_iZ_{(i+1) \bmod n} + Z_iZ_{(i+2) \bmod n} + Z_{(i+1) \bmod n}Z_{(i+2) \bmod n} && \text{(two-term interaction)}\\
&\quad -Z_iZ_{(i+1) \bmod n}Z_{(i+2) \bmod n}) && \text{(three-term interaction)}
\end{aligned}
$$

$$
\begin{array}{lccc|ccc|c}
 & \multicolumn{3}{c|}{\text{(single-term expansion)}} & \multicolumn{3}{c|}{\text{(two-term expansion)}} & \text{(three-term expansion)}\\
= & \frac{1}{8}(-Z_0 & \cancel{-Z_1} & \cancel{-Z_2} & \cancel{+Z_0Z_1} & +Z_0Z_2 & \cancel{+Z_1Z_2} & -Z_0Z_1Z_2\\
 & +Z_1 & \cancel{+Z_2} & \cancel{+Z_3} & \cancel{-Z_1Z_2} & -Z_1Z_3 & \cancel{-Z_2Z_3} & +Z_1Z_2Z_3\\
 & -Z_2 & \cancel{-Z_3} & \cancel{-Z_4} & \cancel{+Z_2Z_3} & +Z_2Z_4 & \cancel{+Z_3Z_4} & -Z_2Z_3Z_4\\
 & +Z_3 & \cancel{+Z_4} & \cancel{+Z_5} & \cancel{-Z_3Z_4} & -Z_3Z_5 & \cancel{-Z_4Z_5} & +Z_3Z_4Z_5\\
 & & \vdots & & & \vdots & & \vdots\\
 & -Z_{n-4} & \cancel{-Z_{n-3}} & \cancel{-Z_{n-2}} & \cancel{+Z_{n-4}Z_{n-3}} & +Z_{n-4}Z_{n-2} & \cancel{+Z_{n-3}Z_{n-2}} & -Z_{n-4}Z_{n-3}Z_{n-2}\\
 & +Z_{n-3} & \cancel{+Z_{n-2}} & \cancel{+Z_{n-1}} & \cancel{-Z_{n-3}Z_{n-2}} & -Z_{n-3}Z_{n-1} & \cancel{-Z_{n-2}Z_{n-1}} & +Z_{n-3}Z_{n-2}Z_{n-1})\\
 & -Z_{n-2} & \cancel{-Z_{n-1}} & \cancel{-Z_0} & \cancel{+Z_{n-2}Z_{n-1}} & +Z_{n-2}Z_0 & \cancel{+Z_{n-1}Z_0} & -Z_{n-2}Z_{n-1}Z_0)\\
 & +Z_{n-1} & \cancel{+Z_0} & \cancel{+Z_1} & \cancel{-Z_{n-1}Z_0} & -Z_{n-1}Z_1 & \cancel{-Z_0Z_1} & +Z_{n-1}Z_0Z_1)
\end{array}
$$

$$
\begin{aligned}
= \quad & \tfrac{1}{8}\left(\left(\sum_{i=0}^{n-1}(-1)^{i+1}Z_i\right) - ((n \bmod 2)2Z_1\right) && \text{(single-term interaction)}\\
& + \left(\sum_{i=0}^{n-1}(-1)^{i}Z_iZ_{(i+2) \bmod n}\right) + ((n \bmod 2)2Z_0Z_1) && \text{(two-term interaction)}\\
& + \left(\sum_{i=0}^{n-1}(-1)^{i+1}Z_iZ_{(i+1) \bmod n}Z_{(i+2) \bmod n}\right)\Bigg) && \text{(three-term interaction)}
\end{aligned} \tag{4.22}
$$

where:

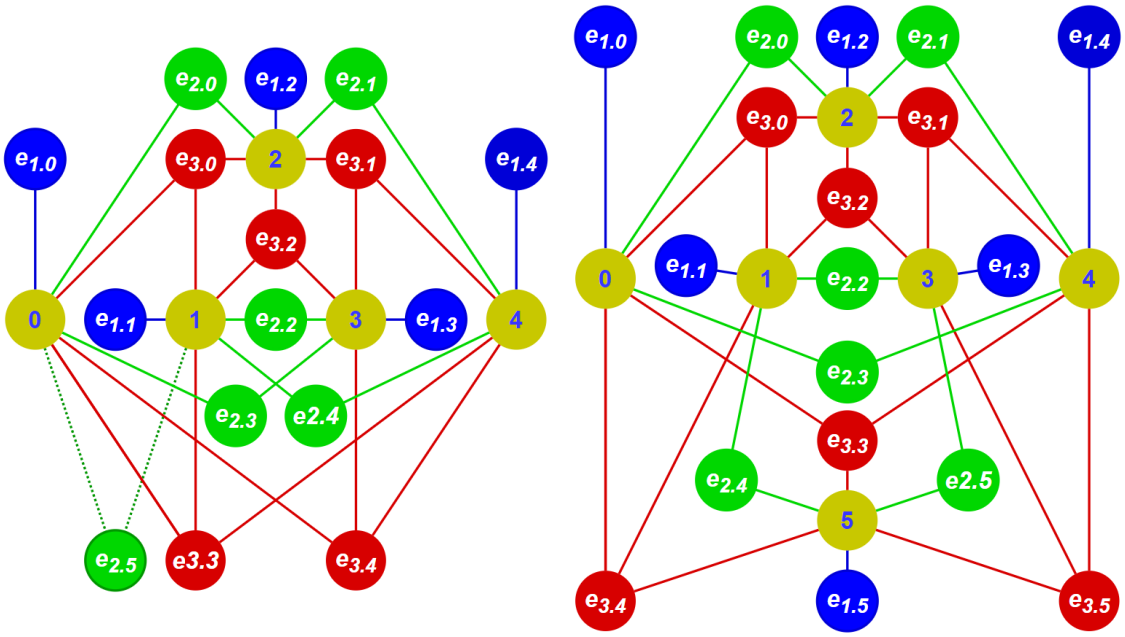
Z_i	:	Pauli-Z gates
n	$\in$	$\mathbb{N}$

Therefore, the number of terms required for single-term interaction is n (since the Z_1 term is already present in the summation), $n + (n \bmod 2)$ terms are required for the two-term interaction, and n terms are required for the three-term interaction, making the total of $3n + (n \bmod 2)$ terms required for the cost function. To this, we add n for the number of terms in the mixer operator. As explained before, ma-QAOA uses a separate angle per term in the Hamiltonian. Therefore, the number of angles required to run a ma-QAOA for the cyclic 3-Uniform *sign-alternating* hypergraph is:

$$N_{ma\angle} = 4n + (n \bmod 2) \tag{4.23}$$

4.3.2.2 | am-QAOA and ka-QAOA Algorithm for 3-Uniform Cyclic Hypergraphs

As in Section 4.3.1.3, we must again determine the automorphism of the spin graph defined by equation (4.22). We do this by first finding its bipartite incidence graph, illustrated in Figure 4.23.



(a) Bipartite incidence graph for 5-vertex cyclic hypergraph

(b) Bipartite incidence graph for 6-vertex cyclic hypergraph

Figure 4.23: Bipartite incidence graph for the 5- and 6-vertex cyclic hypergraphs.

Legend: ● *nodes* are the original nodes from the problem, ● *nodes* are the edges for single-term interactions, ● *nodes* are the edges for two-term interactions (note that edge $e_{2.5}$ in the 5-vertex graph is required because n, in this case 5, is odd), and ● *nodes* are the edges for three-term interactions.

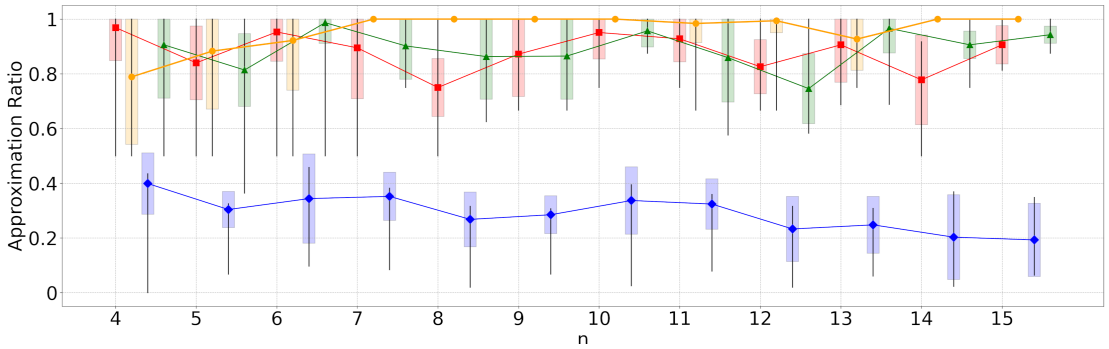
4.3.2.3 | QAOA Function Evaluations for 3-Uniform Cyclic Hypergraphs

The Cost Function (equation (4.22)) was tested for graphs of size $n \in [4, 15]$ and with number of layers $p \in [1, 3]$ using sa-QAOA, ma-QAOA, am-QAOA and ka-QAOA.

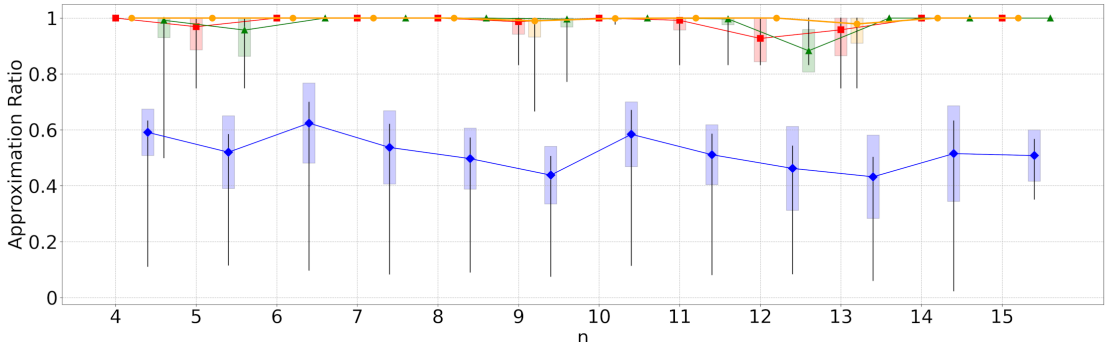
The following table and charts present the results for 3-Uniform Cyclic Sign-Alternating Hypergraphs (Section 4.3.2). See the explanations at the beginning of Section 4 for guidance on interpreting the data in the table and charts.

Table 4.5: Average number of function evaluations for 3-Uniform Cyclic Sign-Alternating Hypergraphs.

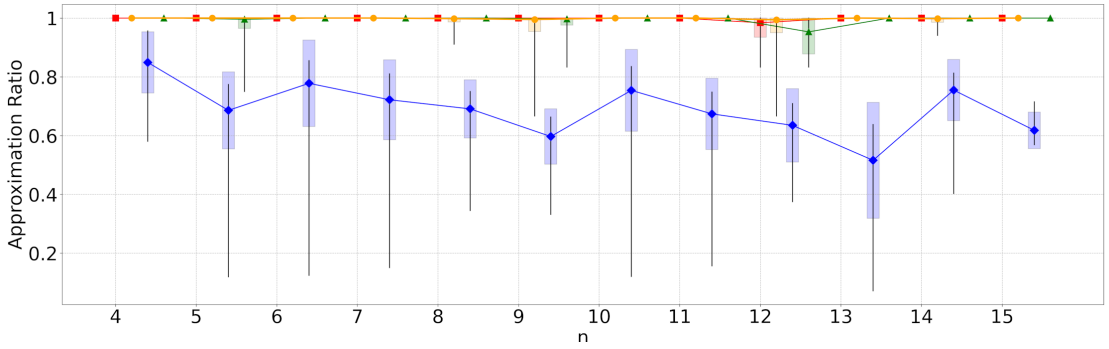
	sa-QAOA			ma-QAOA			am-QAOA			ka-QAOA		
n	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$
4	$46_{0.4}$	$247_{0.59}$	$563_{0.85}$	$916_{0.97}$	2551	4142	$703_{0.91}$	$1967_{0.99}$	3580	$333_{0.79}$	1121	1692
5	$63_{0.3}$	$198_{0.52}$	$397_{0.69}$	$1607_{0.84}$	$5022_{0.97}$	9014	$1112_{0.81}$	$3153_{0.96}$	5357	$331_{0.88}$	901	1853
6	$59_{0.34}$	$242_{0.62}$	$541_{0.78}$	$1897_{0.95}$	5683	10703	$1518_{0.99}$	3244	6123	$523_{0.92}$	1484	2600
7	$62_{0.35}$	$171_{0.54}$	$427_{0.72}$	$2612_{0.89}$	7442	12993	$1447_{0.9}$	4154	7218	445	1311	2563
8	$56_{0.27}$	$202_{0.5}$	$498_{0.69}$	$3006_{0.75}$	7846	16508	$2189_{0.86}$	5481	10184	503	1861	3199
9	$62_{0.28}$	$254_{0.44}$	$472_{0.6}$	$3768_{0.87}$	$10476_{0.99}$	18465	$2379_{0.86}$	5835	10953	612	$1456_{0.99}$	$3338_{0.99}$
10	$53_{0.34}$	$226_{0.58}$	$481_{0.75}$	$3882_{0.95}$	11598	22761	$2471_{0.96}$	7194	13606	700	2275	4066
11	$58_{0.32}$	$189_{0.51}$	$460_{0.67}$	$4561_{0.93}$	$13654_{0.99}$	25437	$3080_{0.86}$	7600	14762	$634_{0.98}$	1915	3887
12	$36_{0.23}$	$114_{0.46}$	$210_{0.64}$	$5138_{0.83}$	$13919_{0.93}$	$28563_{0.98}$	$2966_{0.75}$	$8905_{0.88}$	$17562_{0.95}$	$780_{0.99}$	2397	$4547_{0.99}$
13	$39_{0.25}$	$109_{0.43}$	$196_{0.52}$	$9902_{0.91}$	$30437_{0.96}$	60186	$4981_{0.97}$	14664	27811	$715_{0.93}$	$2007_{0.98}$	4753
14	$40_{0.2}$	$115_{0.52}$	$221_{0.75}$	$6106_{0.78}$	19744	42222	$4135_{0.91}$	16769	36723	847	2649	5869
15	$45_{0.19}$	$90_{0.51}$	$216_{0.62}$	$8933_{0.91}$	36233	79227	$5538_{0.94}$	18198	36552	766	2361	6256



(a) Approximation Ratio for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p = 1$.



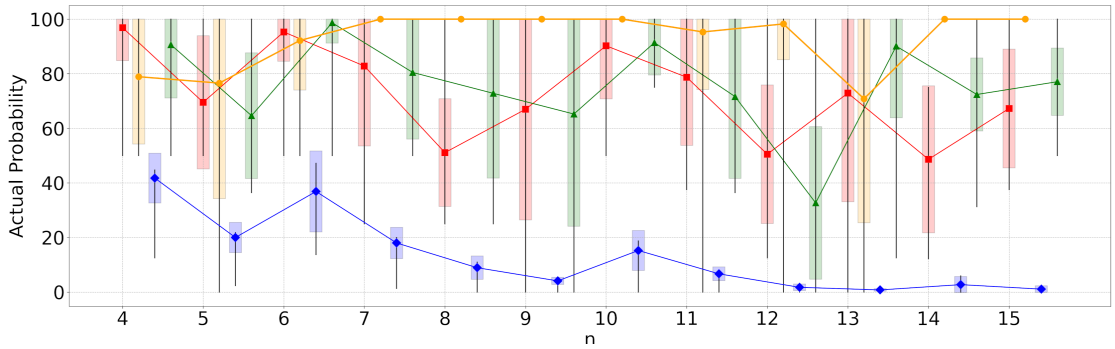
(b) Approximation Ratio for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p = 2$.



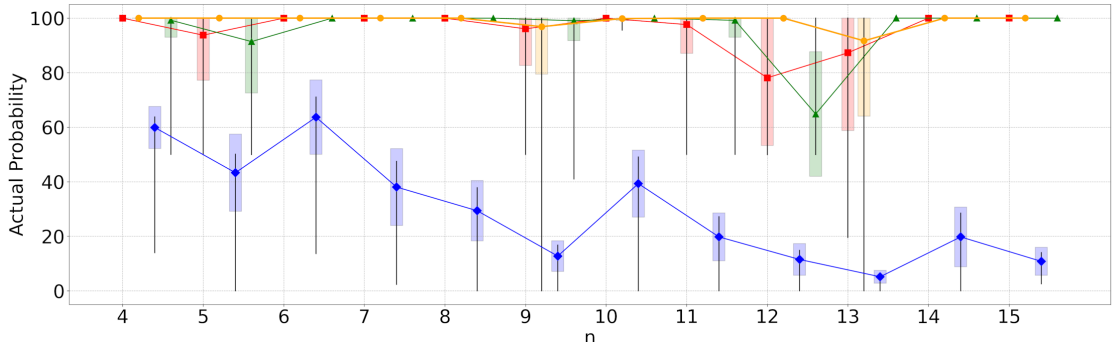
(c) Approximation Ratio for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p = 3$.

Figure 4.24: Approximation Ratio for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p \in \{1, 2, 3\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

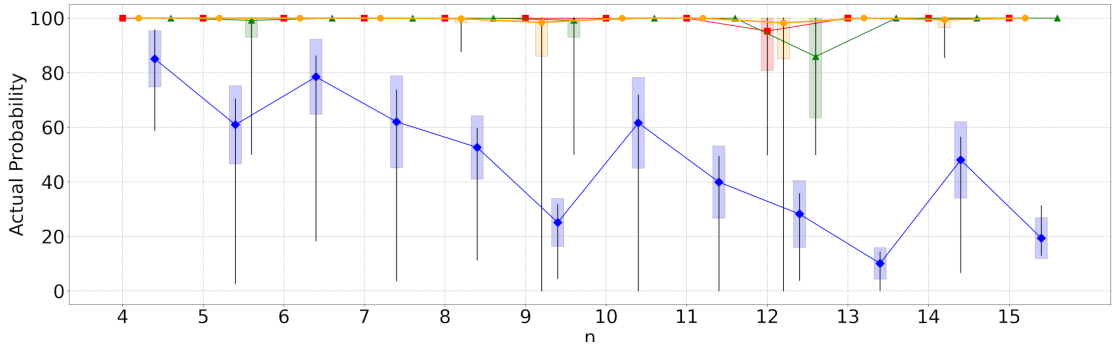
Legend: ■ Multi-angle, ● k-angle, ◆ Single angle, ▲ Automorphic angle.



(a) Actual Probability for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p = 1$.



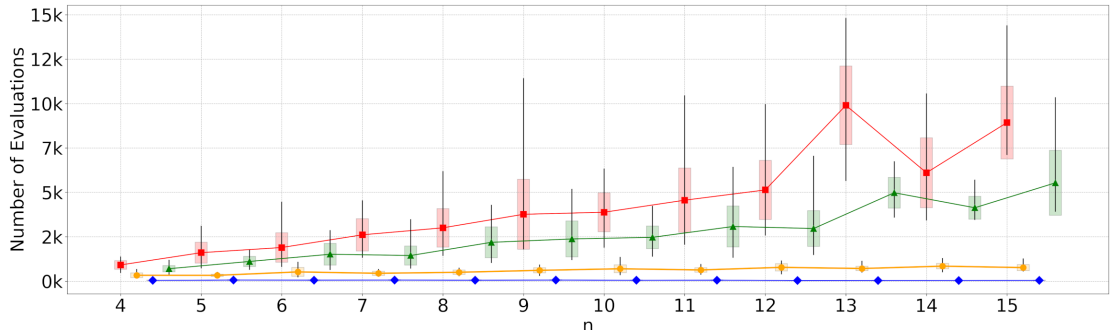
(b) Actual Probability for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p = 2$.



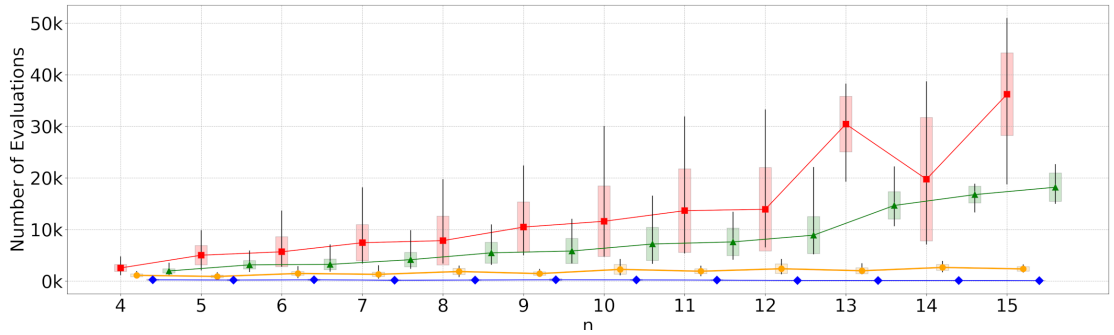
(c) Actual Probability for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p = 3$.

Figure 4.25: Actual Probability for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p \in \{1, 2, 3\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

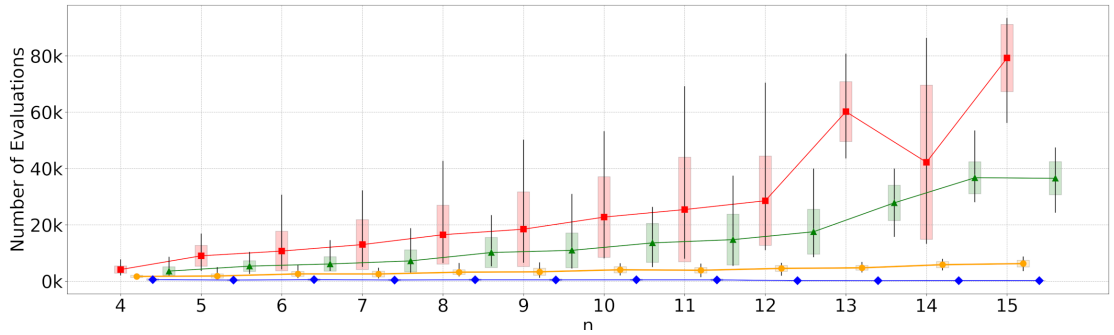
Legend: Multi-angle, k-angle, Single angle, Automorphic angle.



(a) Number of function evaluations for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p = 1$.



(b) Number of function evaluations for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p = 2$.



(c) Number of function evaluations for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p = 3$.

Figure 4.26: Number of function evaluations for 3-Uniform Cyclic Sign-Alternating Hypergraphs with $p \in \{1, 2, 3\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

Legend: Multi-angle, k-angle, Single angle, Automorphic angle.

Even at $p = 1$ experiments, the Approximation Ratio (AR) of the ka-QAOA variant shows that it is an ideal candidate for 3-Uniform Cyclic Sign-Alternating Hypergraphs. It gives slightly better results than those achieved by am-QAOA and ma-QAOA while providing a much better solution to sa-QAOA. Moreover, as in the case of the experiments on 3-Uniform Path Sign-Alternating Hypergraphs, the Number of Function Evaluations (nfev) required to reach the needed optimisation is considerably lower than that required by am-QAOA and ma-QAOA. The notable drop in *nfev* highlights ka-QAOA's ability to deliver strong optimisation performance with limited resource usage, positioning it as an effective approach for the constraints of the NISQ era.

4.3.3 | Function Evaluations for Random Coefficient (-1, +1, 0) Hypergraphs

To further evaluate the performance of k-interaction Angle QAOA (ka-QAOA) against sa-QAOA and ma-QAOA, tests were performed on forty-four 12-vertex hypergraphs ($n = 12$), as per equation (4.24), with randomly assigned coefficient values of -1, +1, or 0, and $p \in [1, 5]$. Since only a few graphs exhibited automorphism, am-QAOA was not tested.

$$C'(x) = \sum_{i=0}^{n-1} \mathcal{J}_i \prod_{j=0}^{2} x_{(i+j) \bmod n} \tag{4.24}$$

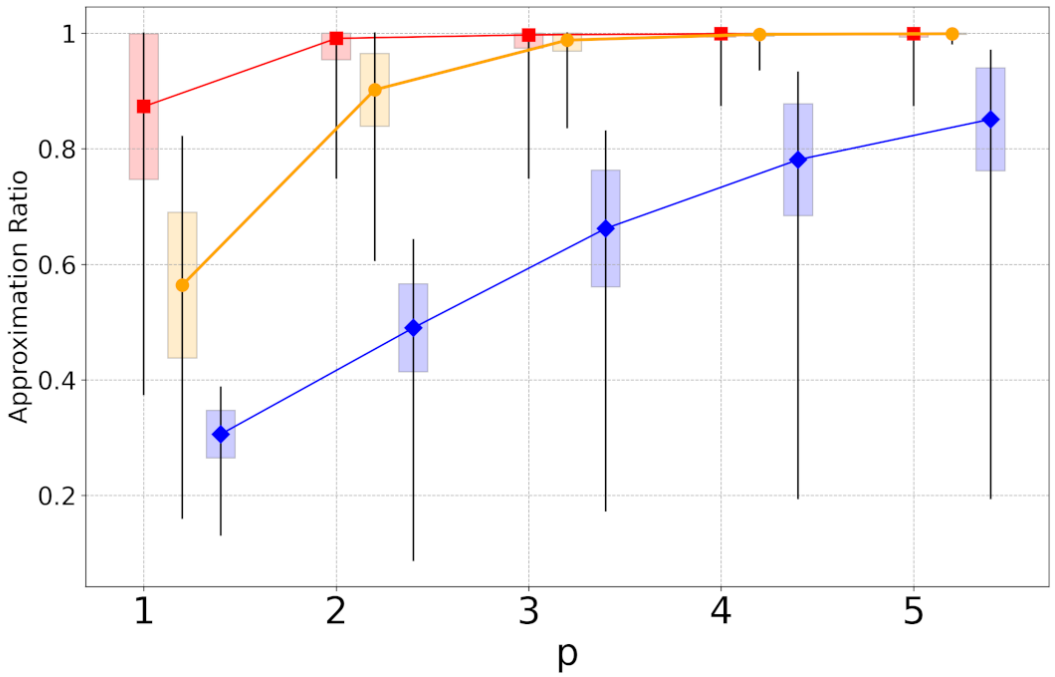
where:

x_n	:	boolean variables
n	:	number of variables, or vertices in the hypergraph
$\mathcal{J}_i$	:	i^{th} coefficient; a value -1, +1, or 0 assigned randomly

The following table and charts present the results for QAOA on these hypergraphs. See the explanations at the beginning of Section 4 for guidance on interpreting the data in the table and charts.

Table 4.6: Average number of function evaluations for random coefficient (-1, +1, 0) hypergraphs.

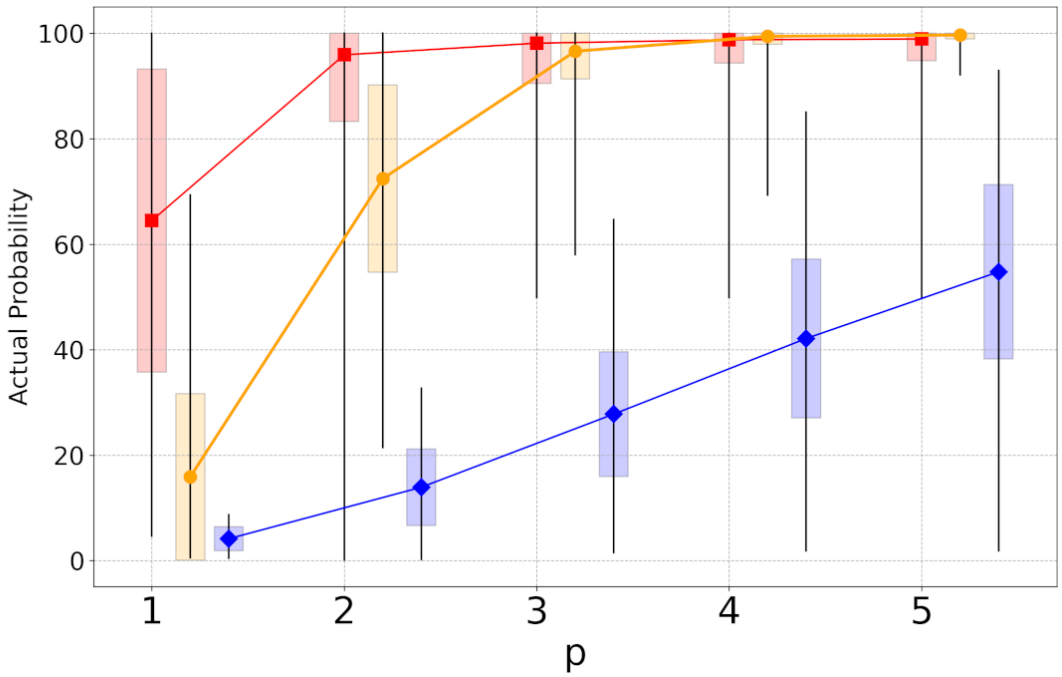
n	angle type	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
12	sa	$63_{0.29}$	$208_{0.49}$	$404_{0.66}$	$673_{0.78}$	$977_{0.85}$
12	ma	$5539_{0.87}$	$14531_{0.99}$	26929	40262	52488
12	ka	$895_{0.56}$	$4163_{0.9}$	$8995_{0.99}$	11740	14339



(a) Approximation Ratio for random coefficient (-1, +1, 0) hypergraphs with $n = 12$.

Figure 4.27: Approximation Ratio for random coefficient (-1, +1, 0) hypergraphs with $p \in \{1...5\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n = 12$, $k = 3$ and coefficient $\mathcal{J}_i \in \{-1, +1, 0\}$ (random sequence).

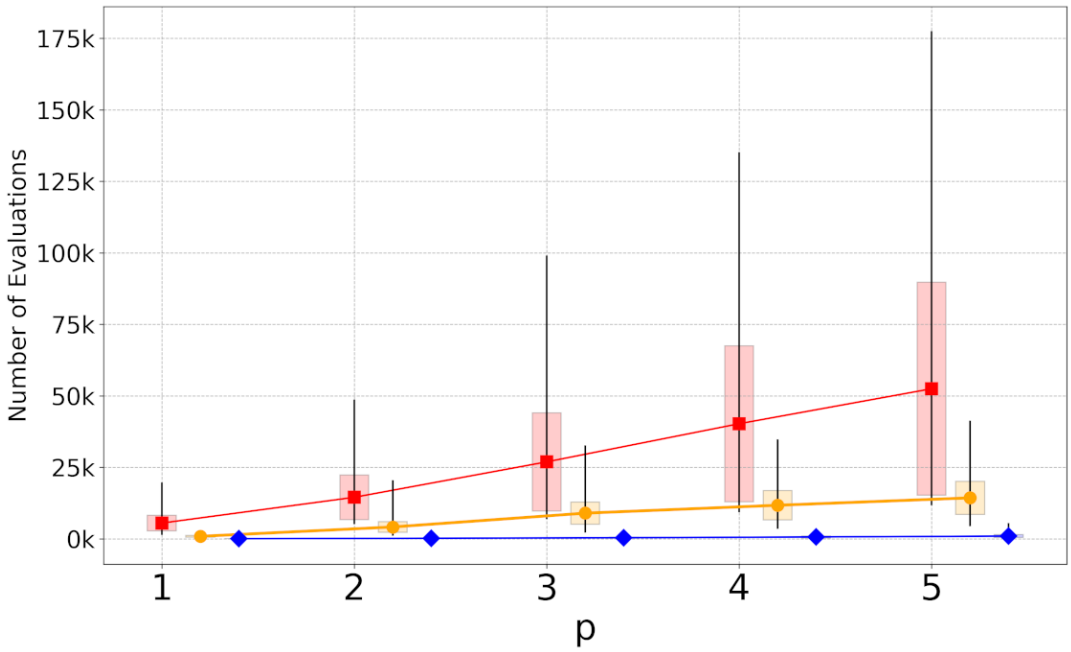
Legend: Multi-angle, k-angle, Single angle.



(a) Actual Probability for random coefficient (-1, +1, 0) hypergraphs with $n = 12$.

Figure 4.28: Actual Probability for random coefficient (-1, +1, 0) hypergraphs with $p \in \{1...5\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n = 12$, $k = 3$ and coefficient $\mathcal{J}_i \in \{-1, +1, 0\}$ (random sequence).

Legend: ■ Multi-angle, ● k-angle, ◆ Single angle.



(a) Number of function evaluations for random coefficient (-1, +1, 0) hypergraphs with $n = 12$.

Figure 4.29: Number of function evaluations for random coefficient (-1, +1, 0) hypergraphs with $p \in \{1...5\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n = 12$, $k = 3$ and coefficient $\mathcal{J}_i \in \{-1, +1, 0\}$ (random sequence).

Legend: Multi-angle, k-angle, Single angle.

For random coefficient (-1, +1, 0) hypergraphs with $n = 12$, the ka-QAOA variant starts to show a significant AR at $p = 2$, reaching the same levels of ma-QAOA at $p = 3$, while always providing a much better solution against sa-QAOA. Moreover, as in the case of the previous experiments, the Number of Function Evaluations (nfev) required to reach the optimisation required is considerably lower than that needed by ma-QAOA. This reduction in *nfev* is noteworthy since this suggests that ka-QAOA can still obtain high-quality optimisations whilst utilising few resources in the process, even for more complex hypergraphs.

4.3.4 | Function Evaluations for Random Integer Coefficient (-10 to 10) Hypergraphs

Finally, to evaluate the performance of k-interaction Angle QAOA (ka-QAOA) against sa-QAOA and ma-QAOA, tests were performed on eighteen 12 vertex hypergraphs ($n = 12$), as per equation (4.25), with randomly assigned integer coefficient values from -10 to 10 (zero included), and $p \in [1, 5]$. Since only a few graphs exhibited automorphism, am-QAOA was not tested.

$$C'(x) = \sum_{i=0}^{n-1} \mathcal{J}_i \prod_{j=0}^{2} x_{(i+j) \bmod n} \tag{4.25}$$

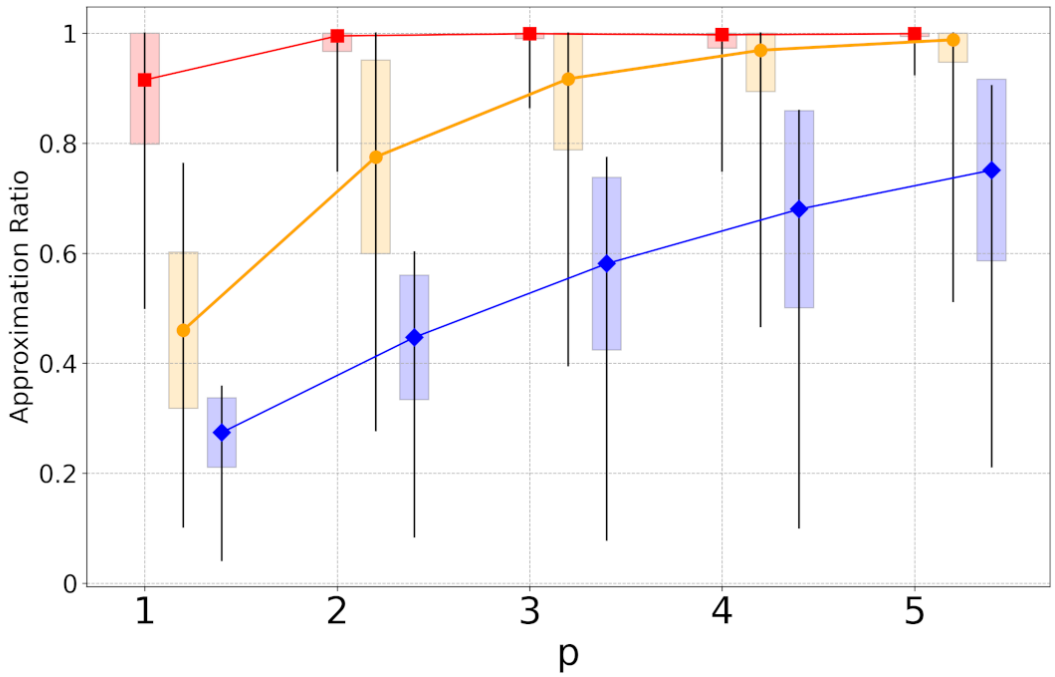
where:

x_n	:	boolean variables
n	:	number of variables, or vertices in the hypergraph
$\mathcal{J}_i$	:	i^{th} coefficient; an integer value from -10 to 10 (zero included) assigned randomly

The following table and charts present the results for QAOA on these hypergraphs. See the explanations at the beginning of Section 4 for guidance on interpreting the data in the table and charts.

Table 4.7: Average number of function evaluations for randomly assigned integer coefficient values from -10 to 10 hypergraphs.

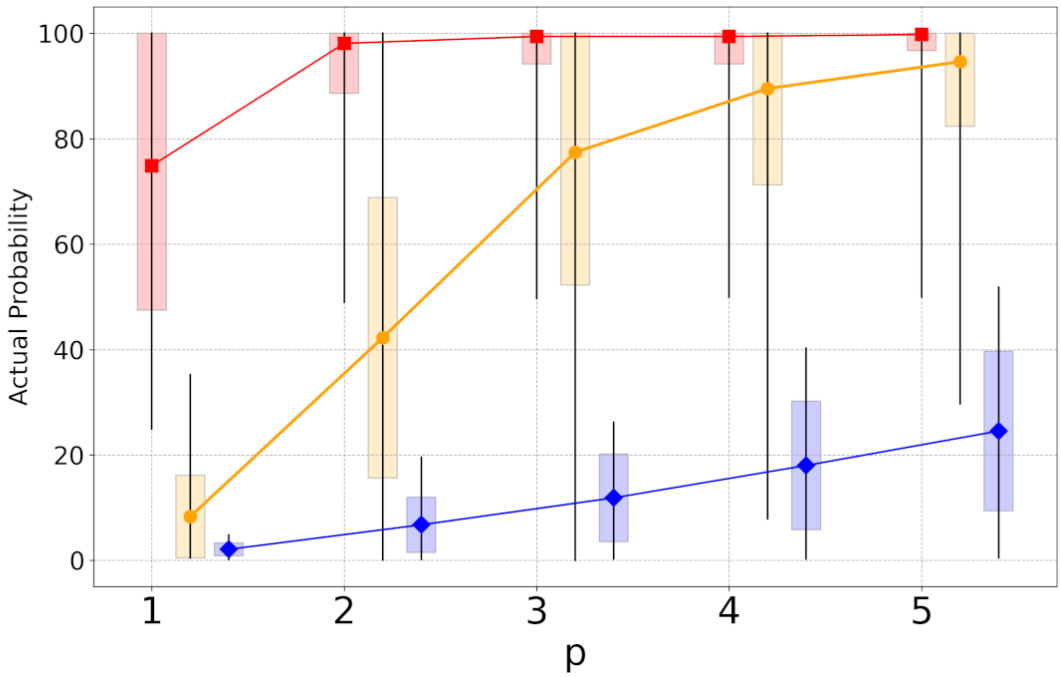
n	angle type	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
12	sa	$60_{0.21}$	$160_{0.41}$	$319_{0.56}$	$539_{0.67}$	$854_{0.74}$
12	ma	$4947_{0.91}$	13091	25610	43063	64230
12	ka	$1009_{0.45}$	$4407_{0.77}$	$11541_{0.92}$	$17442_{0.97}$	$23232_{0.99}$



(a) Approximation Ratio for randomly assigned integer coefficient values from -10 to 10 hypergraphs with $n = 12$.

Figure 4.30: Approximation Ratio for randomly assigned integer coefficient values from -10 to 10 hypergraphs with $p \in \{1...5\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n = 12$, $k = 3$ and coefficient $\mathcal{J}_i \in [-10, 10]$ (random sequence).

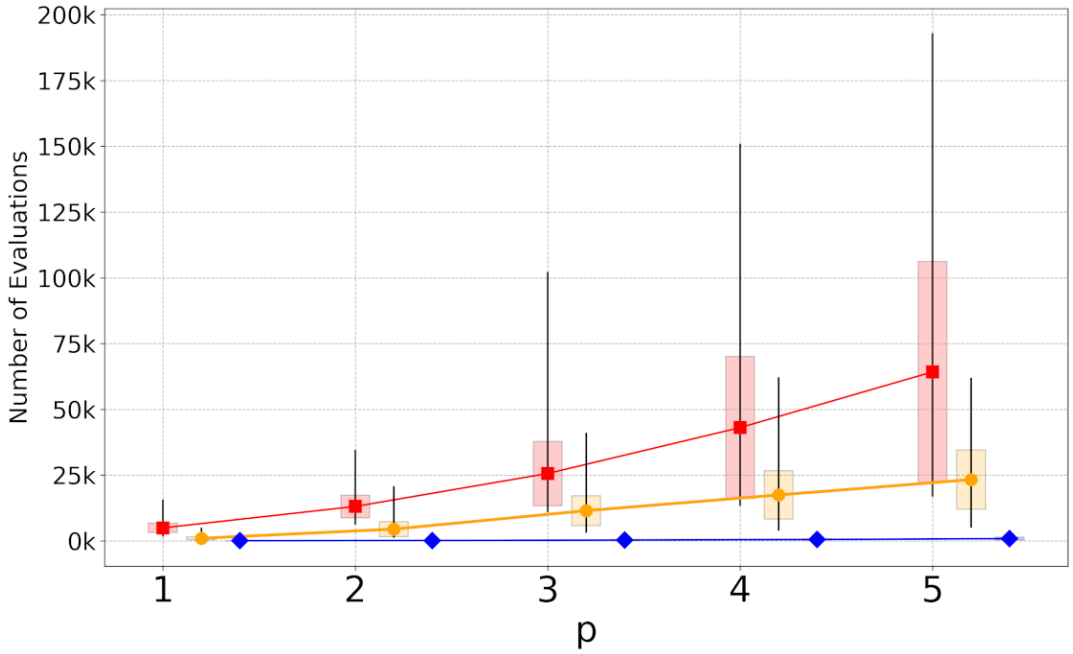
Legend: Multi-angle, k-angle, Single angle.



(a) Actual Probability for randomly assigned integer coefficient values from -10 to 10 hypergraphs with $n = 12$.

Figure 4.31: Actual Probability for randomly assigned integer coefficient values from -10 to 10 hypergraphs with $p \in \{1...5\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n = 12$, $k = 3$ and coefficient $\mathcal{J}_i \in [-10, 10]$ (random sequence).

Legend: ■ Multi-angle, ● k-angle, ◆ Single angle.



(a) Number of function evaluations for randomly assigned integer coefficient values from -10 to 10 hypergraphs with $n = 12$.

Figure 4.32: Number of function evaluations for randomly assigned integer coefficient values from -10 to 10 hypergraphs with $p \in \{1...5\}$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n = 12$, $k = 3$ and coefficient $\mathcal{J}_i \in [-10, 10]$ (random sequence).

Legend: Multi-angle, k-angle, Single angle.

For random coefficient [-10, 10] hypergraphs with $n = 12$, the ka-QAOA variant starts to show a significant AR at $p = 3$, reaching the same levels of ma-QAOA at $p = 4$, while again providing a much better solution against sa-QAOA. Moreover, as in the case of the previous results, the Number of Function Evaluations (nfev) required to reach the needed optimisation is considerably lower than that required by ma-QAOA. For example, the *nfev* for ka-QAOA at $p = 3$ is consistently lower than that of ma-QAOA at $p = 2$. This reduction in *nfev* is noteworthy since this suggests that ka-QAOA can still obtain high-quality optimisations whilst utilising few resources in the process, even for more complex hypergraphs, albeit requiring a slightly higher number of layers.

4.4 | Comparison of QAOA Configurations

4.4.1 | Initial Angle Parameters

As discussed in Section 3.5.3.3, the choice of the initial angles of QAOA can significantly impact the QAOA's performance. In the following charts and table, we refer to different initial angles of QAOA:

- $\frac{1}{p}$: Variable p is the layer number in the QAOA. Since one needs to initialise each layer with different angles, the initialisation for $\frac{1}{p}$ would mean that for the first layers, all angles would be initialised to 1, the second layer to $\frac{1}{2}$, the third layer to $\frac{1}{3}$, and so on.
- $R(0 \rightarrow \frac{1}{p})$: the principle is the same as for $\frac{1}{p}$, but in this case, for the initial parameter a random value from 0 to $\frac{1}{p}$ is selected. Therefore, for the first layers, all angles would be initialised to random numbers from 0 to 1, 0 to $\frac{1}{2}$ for the second layer, 0 to $\frac{1}{3}$ for the third layer, and so on.
- $R(0 \rightarrow \frac{2\pi}{p})$: this is the same principle as for $R(0 \rightarrow \frac{1}{p})$, except that the maximum number formula changes to $\frac{2\pi}{p}$ instead of $\frac{1}{p}$.
- TQA_{75}: Based on the approach of Sack and Serbyn [74] (Section 3.5.3.3), the cost function parameters γ are initialised to $\frac{i}{T}\Delta t$, while the mixer parameters β are initialised to $\left(1 - \frac{i}{T}\right)\Delta t$, where $i = 1 \ldots p$ and $\Delta t = \frac{t}{T}$. For our tests, for example in a 3-layer QAOA circuit, we are setting $\Delta t = 0.75$, therefore the first layer $\gamma = (\frac{1}{3})(\frac{3}{4}) = \frac{1}{4}$, $\beta = (\frac{2}{3})(\frac{3}{4}) = \frac{1}{2}$, the second layer $\gamma = (\frac{2}{3})(\frac{3}{4}) = \frac{1}{2}$, $\beta = (\frac{1}{3})(\frac{3}{4}) = \frac{1}{4}$, and the third layer $\gamma = (\frac{3}{3})(\frac{3}{4}) = \frac{3}{4}$, $\beta = (0)(\frac{3}{4}) = 0$.

The following table and charts present the results for unweighted 3-Uniform Sign-Alternating Hypergraphs (Section 4.3.1 and Section 4.3.2) on different initialisation angles. See the explanations at the beginning of Section 4 for guidance on interpreting the data in the table and charts.

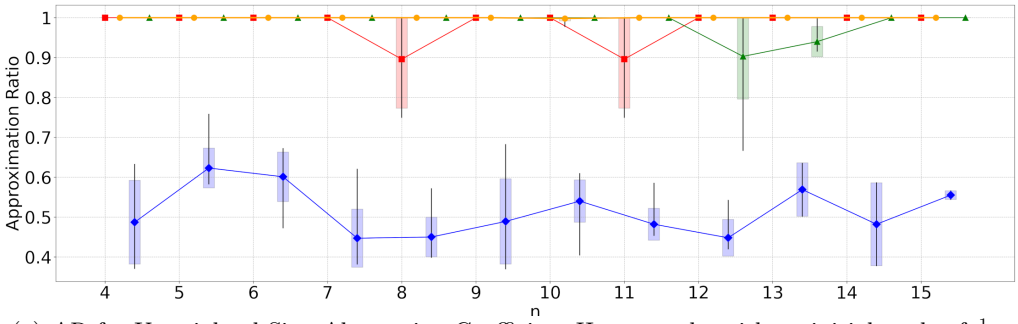
Table 4.8: Average number of function evaluations for Unweighted Sign Alternating Coefficient Hypergraphs segregated by different initial angles.

		sa-QAOA			ma-QAOA			am-QAOA			ka-QAOA		
n	initial angles	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$
4	$\frac{1}{p}$	$47_{0.33}$	$328_{0.49}$	$622_{0.73}$	$739_{0.73}$	1713	3430	$653_{0.53}$	1692	2631	$308_{0.43}$	1221	1950
4	$R(0 \to \frac{1}{p})$	$39_{0.31}$	$218_{0.53}$	$528_{0.84}$	$941_{0.67}$	2967	4473	$685_{0.66}$	$2273_{0.99}$	3668	$412_{0.65}$	1052	1694
4	$R(0 \to \frac{2\pi}{p})$	$54_{0.25}$	$206_{0.41}$	$438_{0.62}$	$841_{0.7}$	2661	4472	$617_{0.69}$	2042	3603	$347_{0.6}$	$1162_{0.97}$	1669
4	TQA_{75}	$41_{0.33}$	$184_{0.55}$	$562_{0.78}$	$786_{0.75}$	2447	3671	$772_{0.75}$	1812	3292	$501_{0.46}$	1180	1949
5	$\frac{1}{p}$	$74_{0.37}$	$280_{0.62}$	$586_{0.76}$	$1496_{0.98}$	3698	5757	$1121_{0.87}$	2431	3981	$391_{0.89}$	1138	1952
5	$R(0 \to \frac{1}{p})$	$62_{0.39}$	$195_{0.66}$	$404_{0.85}$	$1404_{0.86}$	$4585_{0.97}$	8704	$1090_{0.86}$	$2700_{0.95}$	$5743_{0.99}$	$370_{0.89}$	950	2125
5	$R(0 \to \frac{2\pi}{p})$	$55_{0.31}$	$162_{0.5}$	$374_{0.67}$	$1463_{0.91}$	$4191_{0.97}$	7599	$1080_{0.89}$	$2859_{0.98}$	5135	$431_{0.83}$	$1189_{0.99}$	2382
5	TQA_{75}	$57_{0.4}$	$171_{0.67}$	$427_{0.86}$	$1040_{0.96}$	3121	5635	756	2121	3886	$677_{0.68}$	1549	2570
6	$\frac{1}{p}$	$54_{0.36}$	$327_{0.6}$	$668_{0.81}$	$1487_{0.77}$	4353	6220	$1997_{0.81}$	3112	4549	$826_{0.56}$	1723	2558
6	$R(0 \to \frac{1}{p})$	$53_{0.41}$	$237_{0.67}$	$504_{0.86}$	$1922_{0.86}$	5520	11439	$1580_{0.8}$	3081	6417	$513_{0.77}$	1355	2902
6	$R(0 \to \frac{2\pi}{p})$	$51_{0.3}$	$189_{0.5}$	$471_{0.65}$	$1942_{0.86}$	4805	9404	$1454_{0.78}$	3070	5930	$587_{0.77}$	1333	2633
6	TQA_{75}	$64_{0.27}$	$183_{0.69}$	$457_{0.83}$	$1243_{0.87}$	3591	7025	$1327_{0.81}$	2975	4636	$738_{0.77}$	1474	2553
8	$\frac{1}{p}$	$41_{0.25}$	$293_{0.45}$	$571_{0.63}$	$2245_{0.69}$	$5122_{0.9}$	10284	$1588_{0.75}$	3960	7457	$638_{0.75}$	2282	3510
8	$R(0 \to \frac{1}{p})$	$53_{0.27}$	$235_{0.5}$	$484_{0.69}$	$3080_{0.69}$	$8949_{0.98}$	18991	$2001_{0.7}$	5297	9567	$525_{0.75}$	1582	2882
8	$R(0 \to \frac{2\pi}{p})$	$53_{0.21}$	$172_{0.37}$	$464_{0.53}$	$2879_{0.72}$	$8972_{0.98}$	16081	$1832_{0.69}$	5127	9734	$601_{0.75}$	1761	$3351_{0.99}$
8	TQA_{75}	$67_{0.29}$	$148_{0.52}$	$486_{0.69}$	$2822_{0.56}$	6580	10304	$1413_{0.56}$	3225	6389	$525_{0.75}$	1603	2964
10	$\frac{1}{p}$	$40_{0.32}$	$335_{0.54}$	$573_{0.76}$	$3326_{0.88}$	9724	13804	$1804_{0.82}$	6559	10126	$810_{0.87}$	3123	4878
10	$R(0 \to \frac{1}{p})$	$52_{0.34}$	$237_{0.62}$	$451_{0.82}$	$4235_{0.89}$	12963	25361	$2383_{0.87}$	7437	13676	$604_{0.87}$	2061	4049
10	$R(0 \to \frac{2\pi}{p})$	$48_{0.25}$	$169_{0.41}$	$468_{0.64}$	$4007_{0.9}$	12031	24202	$2282_{0.87}$	6864	12735	$696_{0.87}$	$2087_{0.99}$	4070
10	TQA_{75}	$70_{0.37}$	$180_{0.64}$	$501_{0.82}$	$3407_{0.7}$	8507	17174	$1883_{0.81}$	4758	9943	$633_{0.87}$	1604	3816

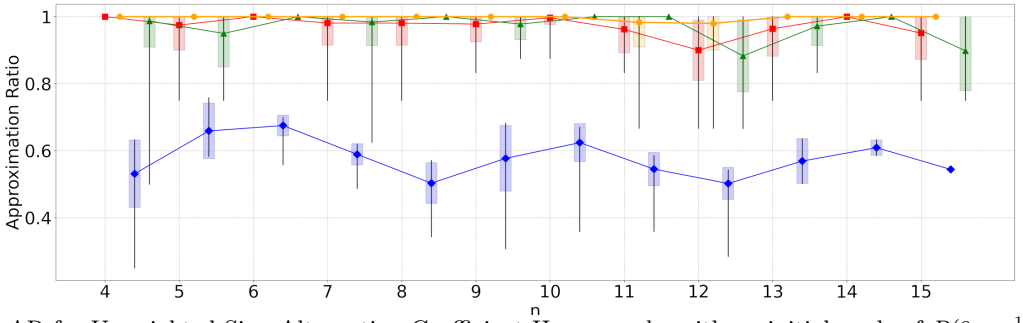
Continued on next page...

Table 4.8 – ***continued from previous page***

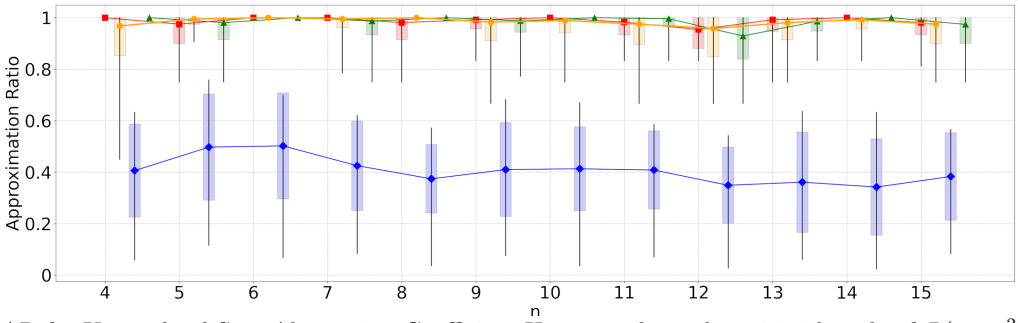
		sa-QAOA			ma-QAOA			am-QAOA			ka-QAOA		
n	initial angles	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$
11	$\frac{1}{p}$	$80_{0.29}$	$240_{0.48}$	$701_{0.66}$	$3228_{0.77}$	$9771_{0.9}$	19520	$2548_{0.77}$	8948	13337	$632_{0.83}$	2507	4187
11	$R(0 \to \frac{1}{p})$	$59_{0.33}$	$197_{0.54}$	$418_{0.73}$	$4551_{0.84}$	$14622_{0.96}$	29823	$2870_{0.85}$	7970	$15262_{0.99}$	$576_{0.83}$	$1705_{0.98}$	3691
11	$R(0 \to \frac{2\pi}{p})$	$51_{0.26}$	$171_{0.41}$	$388_{0.53}$	$4480_{0.83}$	$14554_{0.98}$	25215	$3081_{0.8}$	7351	15774	$700_{0.8}$	$2133_{0.97}$	$4052_{0.98}$
11	TQA_{75}	$59_{0.34}$	$165_{0.57}$	$398_{0.73}$	$3568_{0.85}$	9497	15802	$2457_{0.65}$	5154	9907	$778_{0.83}$	$2631_{0.93}$	4192
12	$\frac{1}{p}$	$42_{0.22}$	$304_{0.45}$	$493_{0.61}$	$4488_{0.81}$	10625	20066	$2215_{0.65}$	$7879_{0.9}$	$11624_{0.93}$	$862_{0.83}$	3738	4918
12	$R(0 \to \frac{1}{p})$	$43_{0.28}$	$165_{0.5}$	$378_{0.67}$	$5193_{0.75}$	$16743_{0.9}$	$33182_{0.97}$	$2679_{0.73}$	$8381_{0.88}$	$17245_{0.97}$	$660_{0.79}$	$2027_{0.98}$	4357
12	$R(0 \to \frac{2\pi}{p})$	$43_{0.21}$	$167_{0.35}$	$389_{0.51}$	$5106_{0.75}$	$14744_{0.95}$	30758	$2632_{0.75}$	$8464_{0.93}$	$17810_{0.98}$	$714_{0.83}$	$2401_{0.96}$	$4498_{0.96}$
12	TQA_{75}	$56_{0.29}$	$167_{0.5}$	$430_{0.66}$	$5530_{0.67}$	8759	17773	$2967_{0.51}$	$5954_{0.86}$	10633	$768_{0.83}$	$2111_{0.99}$	5432
13	$\frac{1}{p}$	$39_{0.08}$	$85_{0.57}$	$182_{0.73}$	$4004_{0.87}$	12740	23265	$2500_{0.87}$	$10738_{0.94}$	17112	$867_{0.83}$	2936	5360
13	$R(0 \to \frac{1}{p})$	$38_{0.34}$	$110_{0.57}$	$199_{0.73}$	$6690_{0.88}$	$23120_{0.96}$	45551	$3650_{0.92}$	$12116_{0.97}$	$24587_{0.99}$	$628_{0.9}$	1612	4799
13	$R(0 \to \frac{2\pi}{p})$	$38_{0.25}$	$126_{0.36}$	$231_{0.47}$	$6534_{0.87}$	$22273_{0.99}$	40025	$3606_{0.91}$	$11050_{0.99}$	22482	$823_{0.86}$	$2650_{0.98}$	4983
13	TQA_{75}	$37_{0.34}$	$100_{0.57}$	$178_{0.73}$	$6072_{0.82}$	13419	27381	$3348_{0.62}$	8462	12724	$856_{0.82}$	$5008_{0.98}$	5473
14	$\frac{1}{p}$	$34_{0.08}$	$100_{0.48}$	$199_{0.79}$	$5970_{0.79}$	12803	25377	$3360_{0.88}$	13190	27495	$792_{0.92}$	2135	5902
14	$R(0 \to \frac{1}{p})$	$40_{0.33}$	$115_{0.61}$	$221_{0.79}$	$6982_{0.86}$	28919	71373	$3748_{0.88}$	14374	32997	$711_{0.91}$	1961	5273
14	$R(0 \to \frac{2\pi}{p})$	$39_{0.11}$	$130_{0.34}$	$254_{0.61}$	$8347_{0.87}$	29679	56616	$3982_{0.88}$	14314	32620	$806_{0.92}$	$2731_{0.99}$	5384
14	TQA_{75}	$37_{0.35}$	$107_{0.61}$	$171_{0.79}$	$5311_{0.55}$	13841	26935	$6078_{0.79}$	15722	23032	$927_{0.92}$	2572	6604
15	$\frac{1}{p}$	$34_{0.08}$	$87_{0.56}$	$185_{0.71}$	$5440_{0.75}$	22482	24158	$2710_{0.76}$	10151	14225	$665_{0.88}$	2312	5280
15	$R(0 \to \frac{1}{p})$	$38_{0.22}$	$116_{0.54}$	$229_{0.71}$	$5797_{0.76}$	$19731_{0.95}$	$36565_{0.98}$	$3489_{0.77}$	$11476_{0.9}$	$21223_{0.95}$	$657_{0.88}$	1968	5225
15	$R(0 \to \frac{2\pi}{p})$	$44_{0.18}$	$100_{0.38}$	$221_{0.59}$	$5748_{0.75}$	$19154_{0.98}$	30322	$3292_{0.73}$	$9964_{0.97}$	$18354_{0.98}$	$803_{0.88}$	$2756_{0.97}$	$5940_{0.97}$
15	TQA_{75}	$39_{0.33}$	$100_{0.56}$	$168_{0.71}$	$3573_{0.69}$	11709	20171	$4451_{0.75}$	$6460_{0.81}$	12843	$1092_{0.88}$	$2164_{0.88}$	5032



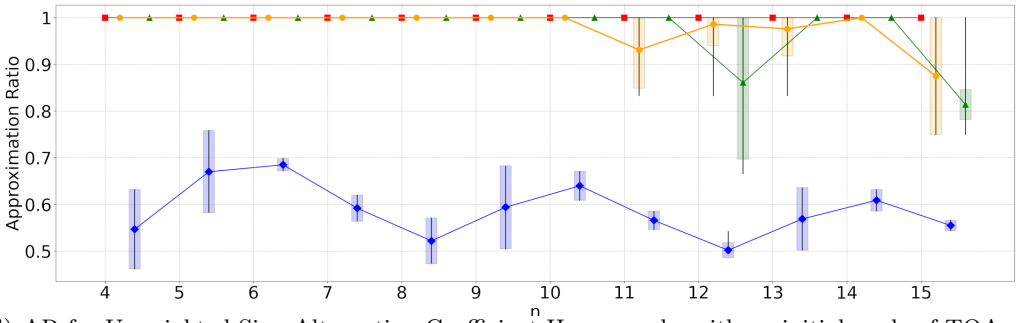
(a) AR for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of $\frac{1}{p}$.



(b) AR for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of $R(0 \rightarrow \frac{1}{p})$.



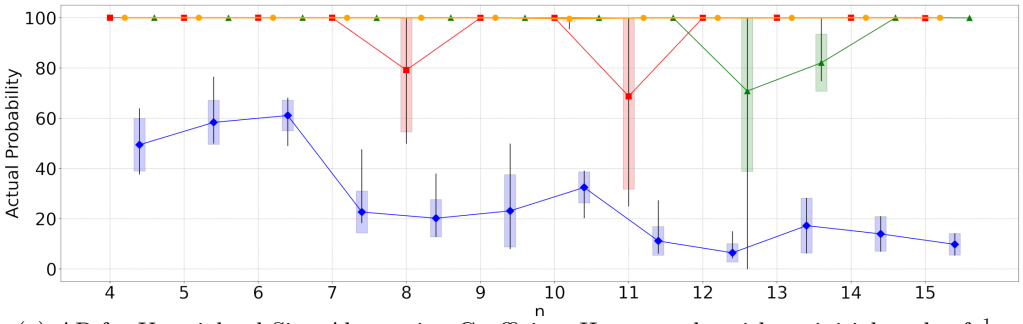
(c) AR for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of $R(0 \rightarrow \frac{2\pi}{p})$.



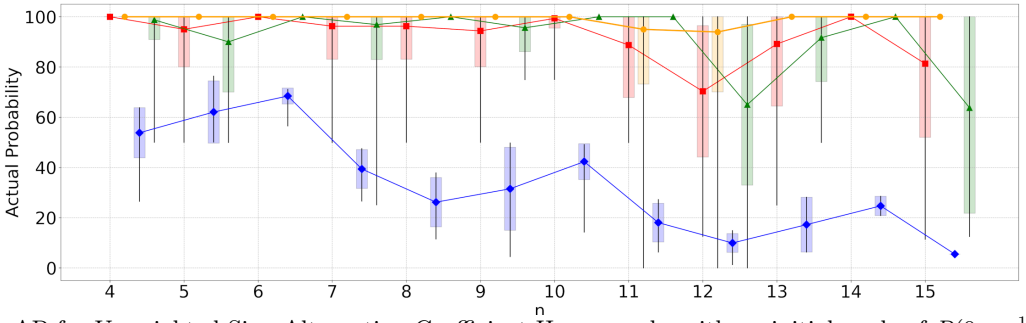
(d) AR for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of TQA_{75}.

Figure 4.33: Unweighted Sign Alternating Coefficient Graphs with different initial angles for $p = 2$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

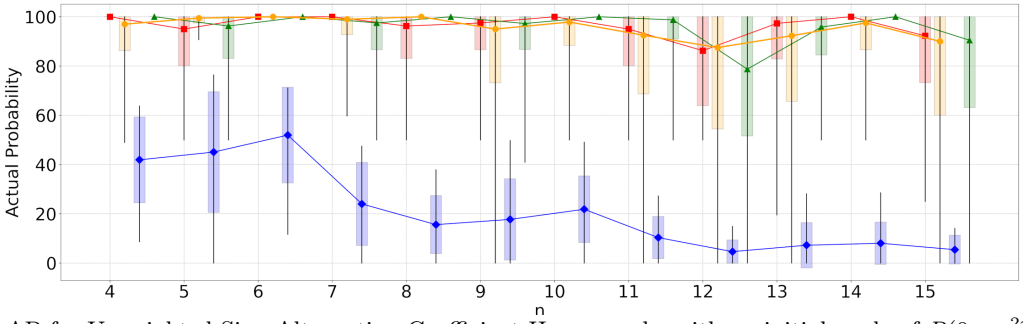
Legend: Multi-angle, k-angle, Single angle, Automorphic angle.



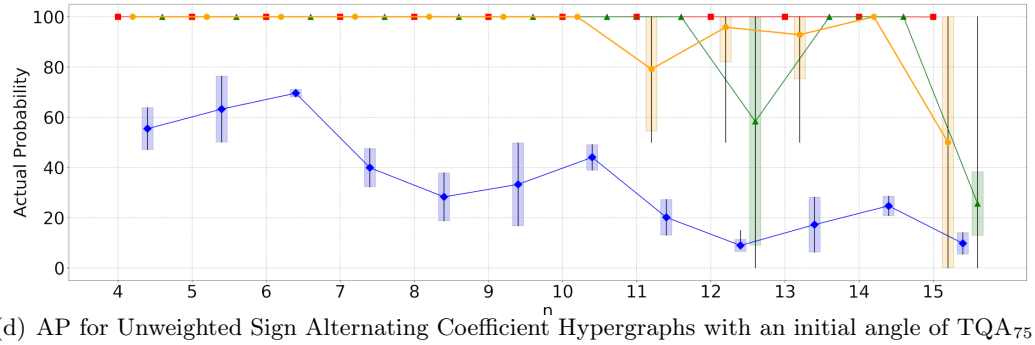
(a) AP for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of $\frac{1}{p}$.



(b) AP for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of $R(0 \rightarrow \frac{1}{p})$.



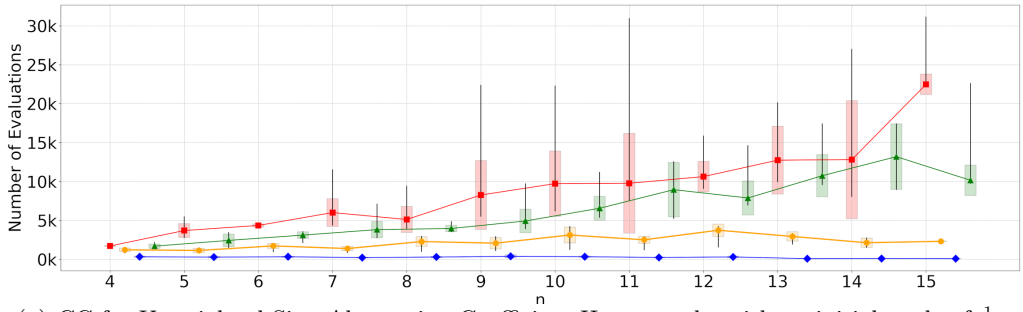
(c) AP for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of $R(0 \rightarrow \frac{2\pi}{p})$.



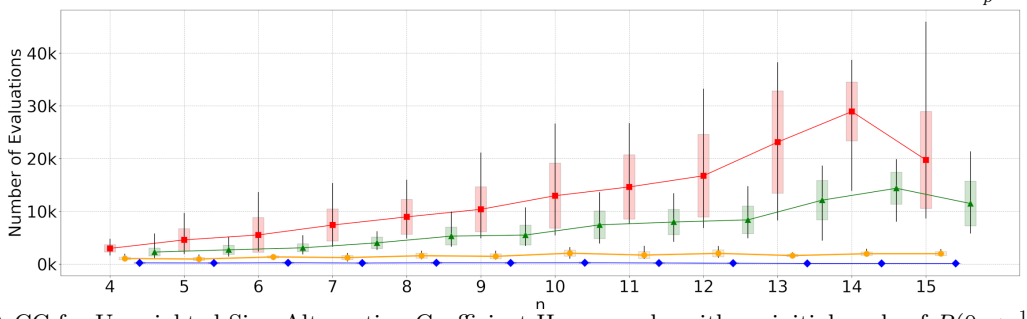
(d) AP for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of TQA_{75}.

Figure 4.34: Unweighted Sign Alternating Coefficient Graphs with different initial angles for $p = 2$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

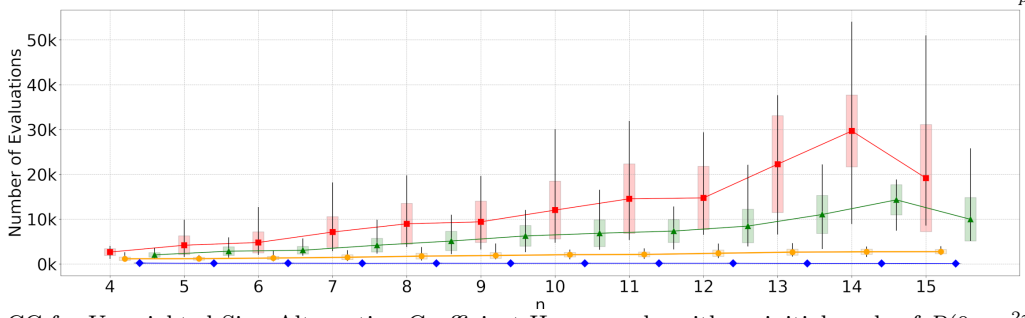
Legend: Multi-angle, k-angle, Single angle, Automorphic angle.



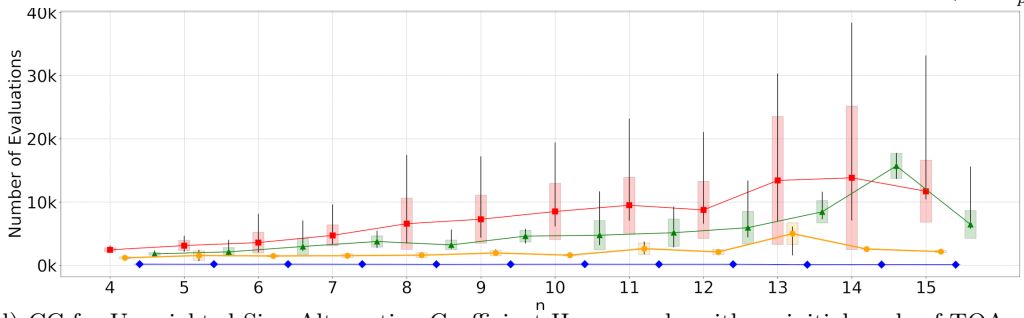
(a) CC for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of $\frac{1}{p}$.



(b) CC for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of $R(0 \rightarrow \frac{1}{p})$.



(c) CC for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of $R(0 \rightarrow \frac{2\pi}{p})$.



(d) CC for Unweighted Sign Alternating Coefficient Hypergraphs with an initial angle of TQA_{75}.

Figure 4.35: Unweighted Sign Alternating Coefficient Graphs with different initial angles for $p = 2$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

Legend: ■ Multi-angle, ● k-angle, ◆ Single angle, ▲ Automorphic angle.

The results show that TQA initialisation [74] returns comparable or slightly better AR with usually less function evaluations for $p > 1$ and all QAOA configurations. The initial values of the parameters are crucial since these angles help set the initial state of the evolution of the quantum state. Poorly chosen angles can lead to an optimisation process with local minima or even barren plateaus since the gradient would be negligibly slight, and the optimisation cannot continue.

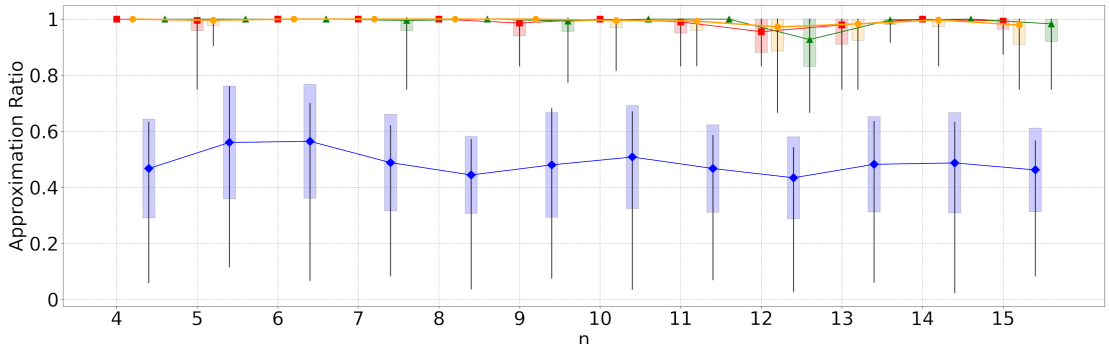
4.4.2 | Classical Optimisation Algorithms

As discussed in Section 3.5.3.1, QAOA, as a hybrid algorithm, comprises both a quantum circuit and a classical optimisation component. The execution alternates between the quantum and classical systems until convergence criteria are met and a minimum value is found or assumed. Classical optimisation algorithms BFGS and Powell (Section 3.5.3.1) were used in these tests.

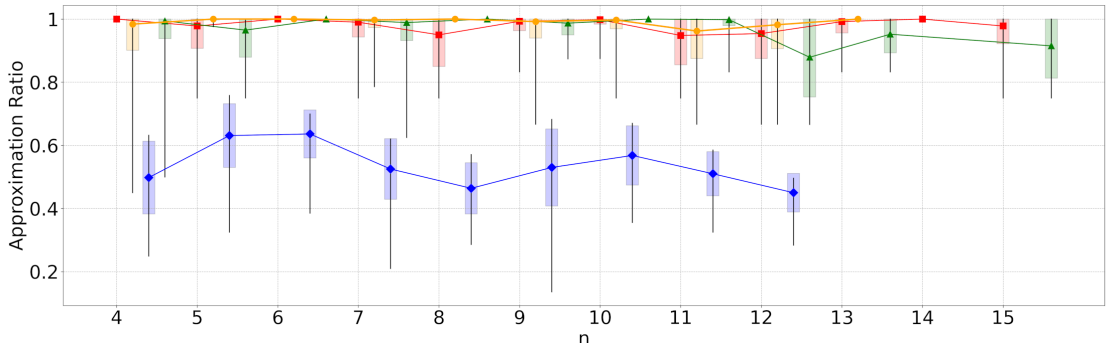
The following table and charts present the results for unweighted 3-Uniform Sign-Alternating Hypergraphs (Section 4.3.1 and Section 4.3.2) on different classical optimisation algorithms. See the explanations at the beginning of Section 4 for guidance on interpreting the data in the table and charts.

Table 4.9: Average number of function evaluations for Unweighted Sign Alternating Coefficient Hypergraphs segregated by different classical algorithms.

		sa-QAOA			ma-QAOA			am-QAOA			ka-QAOA		
n	classical algorithm	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$
4	BFGS	$41_{0.26}$	$135_{0.47}$	$291_{0.75}$	$920_{0.67}$	3084	5372	$700_{0.68}$	2232	4458	$320_{0.62}$	1032	1651
4	Powell	$48_{0.32}$	$284_{0.5}$	$664_{0.73}$	$797_{0.73}$	2211	3380	$658_{0.65}$	$1870_{0.99}$	2737	$430_{0.52}$	$1208_{0.98}$	1861
5	BFGS	$41_{0.31}$	$116_{0.56}$	$227_{0.77}$	$1796_{0.88}$	$5410_{0.99}$	9926	$1215_{0.92}$	3049	6399	$401_{0.87}$	1039	2613
5	Powell	$73_{0.39}$	$244_{0.63}$	$557_{0.78}$	$1117_{0.94}$	$3188_{0.98}$	5614	$919_{0.89}$	$2316_{0.97}$	3959	$481_{0.81}$	1252	2043
6	BFGS	$38_{0.31}$	$123_{0.56}$	$249_{0.76}$	$2442_{0.86}$	6522	14029	$1723_{0.79}$	3514	7626	$575_{0.77}$	1293	3031
6	Powell	$65_{0.36}$	$292_{0.64}$	$676_{0.79}$	$1286_{0.84}$	3632	5978	$1480_{0.8}$	2793	4354	$675_{0.71}$	1527	2482
8	BFGS	$36_{0.2}$	$121_{0.44}$	$241_{0.63}$	$3854_{0.69}$	12200	25311	$2346_{0.71}$	5827	12147	$530_{0.75}$	1354	2990
8	Powell	$64_{0.28}$	$263_{0.46}$	$646_{0.63}$	$2187_{0.66}$	$5151_{0.95}$	8526	$1409_{0.67}$	3871	6516	$594_{0.75}$	2025	$3265_{0.99}$
10	BFGS	$39_{0.24}$	$117_{0.51}$	$233_{0.74}$	$5291_{0.89}$	18139	36589	$2673_{0.87}$	9126	18191	$597_{0.88}$	1722	4156
10	Powell	$60_{0.36}$	$287_{0.57}$	$642_{0.76}$	$2966_{0.83}$	7083	12121	$1835_{0.84}$	5070	8312	$725_{0.87}$	2458	4174
11	BFGS	$37_{0.25}$	$116_{0.47}$	$201_{0.65}$	$5574_{0.85}$	$20583_{0.99}$	41914	$3510_{0.84}$	9545	21733	$598_{0.83}$	$1721_{0.99}$	$4022_{0.99}$
11	Powell	$74_{0.33}$	$236_{0.51}$	$612_{0.66}$	$3210_{0.81}$	$8018_{0.95}$	12967	$2371_{0.75}$	6164	9452	$702_{0.82}$	$2428_{0.96}$	3972
12	BFGS	$35_{0.22}$	$115_{0.43}$	$222_{0.6}$	$6777_{0.74}$	$21188_{0.95}$	45673	$3339_{0.74}$	$10704_{0.93}$	21927	$700_{0.83}$	$1936_{0.97}$	$4420_{0.99}$
12	Powell	$56_{0.28}$	$276_{0.45}$	$631_{0.61}$	$4089_{0.75}$	$8846_{0.95}$	$15918_{0.99}$	$2207_{0.65}$	$6150_{0.88}$	$11049_{0.95}$	$765_{0.82}$	$2882_{0.98}$	$4930_{0.99}$
14	BFGS	$39_{0.22}$	$119_{0.49}$	$229_{0.71}$	$7909_{0.85}$	29444	63046	$4007_{0.87}$	14363	31551	$775_{0.91}$	2347	5482
14	Powell				$4438_{0.63}$	7683	15360						
15	BFGS	$41_{0.19}$	$102_{0.46}$	$214_{0.65}$	$7699_{0.79}$	$33419_{0.99}$	67171	$4651_{0.81}$	$16137_{0.98}$	33323	$755_{0.88}$	$2341_{0.98}$	$5511_{0.99}$
15	Powell				$4673_{0.73}$	$15173_{0.98}$	$19873_{0.99}$	$3230_{0.73}$	$7991_{0.91}$	$12908_{0.98}$			



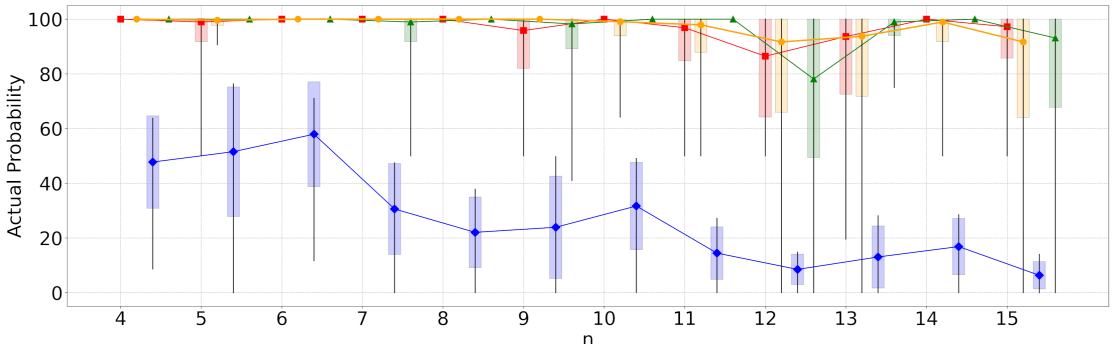
(a) AR for Unweighted Sign Alternating Coefficient Hypergraphs using BFGS classical optimisation algorithm.



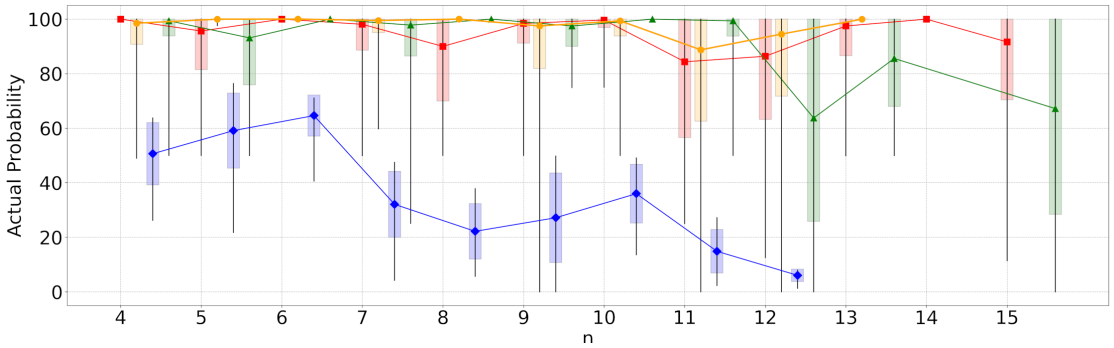
(b) AR for Unweighted Sign Alternating Coefficient Hypergraphs using Powell classical optimisation algorithm.

Figure 4.36: Unweighted Sign Alternating Coefficient Graphs with different classical optimisation algorithms for $p = 2$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

Legend: Multi-angle, k-angle, Single angle, Automorphic angle.



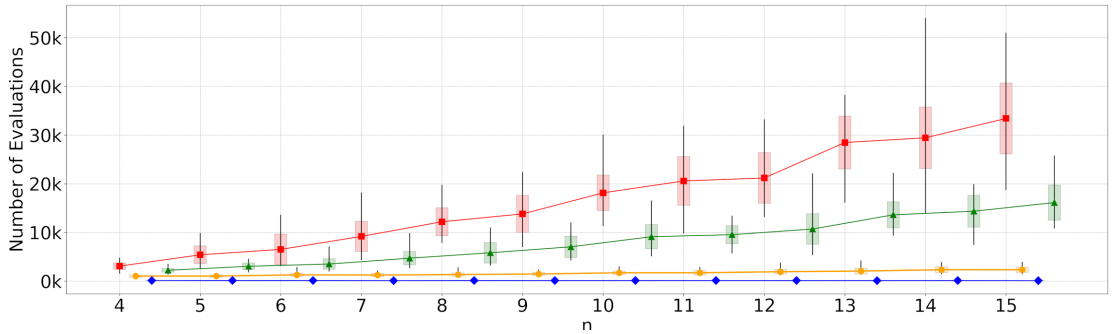
(a) AP for Unweighted Sign Alternating Coefficient Hypergraphs using BFGS classical optimisation algorithm.



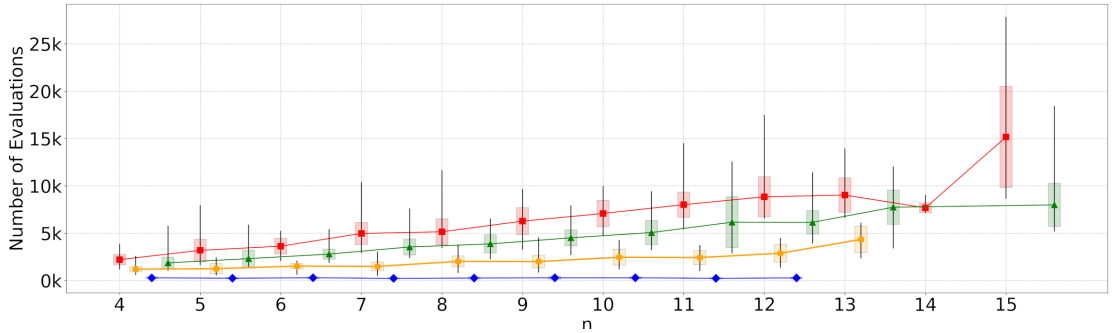
(b) AP for Unweighted Sign Alternating Coefficient Hypergraphs using Powell classical optimisation algorithm.

Figure 4.37: Unweighted Sign Alternating Coefficient Graphs with different classical optimisation algorithms for $p = 2$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

Legend: Multi-angle, k-angle, Single angle, Automorphic angle.



(a) CC for Unweighted Sign Alternating Coefficient Hypergraphs using BFGS classical optimisation algorithm.



(b) CC for Unweighted Sign Alternating Coefficient Hypergraphs using Powell classical optimisation algorithm.

Figure 4.38: Unweighted Sign Alternating Coefficient Graphs with different classical optimisation algorithms for $p = 2$ for the cost function $C(x) = \sum_{i=0}^{n-k} \mathcal{J}_i \prod_{j=0}^{k-1} x_{i+j}$ where $n \in [4, 15]$, $k = 3$. Coefficient $\mathcal{J}_i \in \{+1, -1\}$ (alternating sequence).

Legend: ■ Multi-angle, ● k-angle, ◆ Single angle, ▲ Automorphic angle.

The results show that while both optimisation methods deliver a comparable AR, the Powell algorithm requires fewer function evaluations for $p > 1$ on all QAOA variants that were tested except sa-QAOA. Ultimately, the specific graphs used for these tests may also affect this result. The decision to choose one algorithm over another depends on the particular characteristics of the problem being solved, including the complexity of the cost function and the trade-off between exploration and exploitation within the parameter space. Therefore, this does not necessarily mean that the Powell algorithm is always superior.

4.5 | Results and Insights

In Section 4.1, I analysed QAOA using the Zassenhaus expansion, which yielded improved AR on the Max-Cut problem for the graphs that were examined with a single-layer circuit ($p = 1$). Nonetheless, I do not find a potential for QAOA-CD, considering that the QAOA variant needs more resources than QAOA, at least for the graphs in the tests. Moreover, as the number of layers (p) increases, the Approximation Ratio (AR) for sa-QAOA outperforms QAOA-CD while keeping a lower Number of Function Evaluations (nfev), therefore using less computational resources.

Subsequent tests detailed in Section 4.3 conclude that ka-QAOA consistently maintains a high Approximation Ratio (AR) in comparison to the best-performing ma-QAOA. Notably, ka-QAOA requires significantly fewer function evaluations (*nfev*) needed to reach an acceptable AR. For instance, in Figure 4.19, the AR chart for *sign-alternating* 3-Uniform Path Hypergraphs, the AR for ka-QAOA at $p = 1$ nearly matches the AR for ma-QAOA, by $p = 2$ ka-QAOA reaches almost full potential with AR ≈ 1 matching the result achieved by ma-QAOA. This improvement is achieved in tandem with significantly lower *nfev* (Figure 4.21), which means that ka-QAOA requires fewer resources to compute.

A similar trend is observed for *sign-alternating* 3-Uniform Cyclic Hypergraphs (Figure 4.24). In this case, even at $p = 1$, ka-QAOA marginally outperforms ma-QAOA while still maintaining significantly lower *nfev* (Figure 4.26), therefore balancing resource consumption with performance obtained.

The analysis was further extended to random coefficient hypergraphs. The simulations were done on *random coefficient* [-1, 0, +1] (Section 4.3.3) and *random coefficient* $[-10, 10] \cap \mathbb{Z}$ (Section 4.3.4) in the formulations given by equation (4.24). Results show that ka-QAOA takes more layers to reach the AR of ma-QAOA, with $p = 3$ for random coefficient [-1, 0, +1] and $p = 5$ for random coefficient $[-10, 10] \cap \mathbb{Z}$ scenario. Despite this, the AR in both cases rapidly achieves an acceptable AR (AR $\gtrsim 0.8$) while still necessitating fewer

function evaluations (*nfev*) compared to ma-QAOA.

It is worth noting that, although sa-QAOA requires relatively few function evaluations, achieving satisfactory AR typically required a more significant number of layers, i.e. a larger p. Given that NISQ devices are susceptible to errors as the depth of the circuit increases [54], this characteristic renders the sa-QAOA approach less desirable in practical implementations.

Finally, comparing the effect of different initial angles strategies and classical algorithms on the QAOA further enriched my understanding of QAOA performance. As discussed in Section 4.4.1, I deduce that overall TQA initialisation [74] returns comparable or slightly better AR with usually less function evaluations for $p > 1$ and all QAOA configurations. Similarly, Section 4.4.2 shows that although both optimisation methods achieve comparable AR values, the Powell algorithm consistently requires fewer function evaluations for $p > 1$ across all tested QAOA variants, except for sa-QAOA.

These findings offer a comprehensive view of the trade-offs between the approximation quality and computational efficiency across various QAOA configurations, offering valuable insights for future research and practical applications in quantum optimisation. ka-QAOA shows potential in reducing resource requirements while maintaining a high Approximation Ratio (AR). In the current NISQ era, where quantum computing is both expensive and prone to errors, I have shown that ka-QAOA is a candidate that can help reduce the resource costs associated with quantum computation by minimising the resources required.

QAOA is an algorithm with several components and variants proposed in the literature. In this dissertation, I propose one more variant, ka-QAOA, which I showed is well suited for PUBO problems, and future research can focus on implementing other different configurations suggested in several studies to test the viability of the results in conjunction with ka-QAOA. One example might be integrating QAOA-CD or QAOA-2CD with ka-QAOA and exploring whether ka-QAOA can be effectively combined with parameter concentration effects [91], where optimal parameters tend to cluster around specific values across similar problem instances.

5

Conclusions

In recent years, solving combinatorial optimisation problems with quantum computing has garnered substantial attention, particularly following the work of Farhi et al. [6]. As problems targeted by quantum computing, such as combinatorial optimisation and quantum simulation, grow in complexity, classical methods increasingly struggle to scale, making further performance gains difficult. These discussions often centre around Moore's Law and the eventual physical limits of conventional computing hardware. Lloyd [92] analyses the theoretical boundaries imposed by the laws of physics on computation, while Preskill [16] discusses how classical methods struggle to simulate quantum systems as problem complexity increases. Variational Quantum Algorithms (VQAs) such as the Quantum Approximate Optimization Algorithm (QAOA) have emerged as promising alternatives even during this early NISQ era of quantum computation.

5.1 | Achieved Aims and Objectives

The primary aim of this study was to investigate VQAs for solving combinatorial optimisation problems. More specifically, I focus on developing a resource-efficient solution for JSSP suitable for the NISQ era. This dissertation demonstrates how JSSP can be formulated into a Hamiltonian for use in QAOA. It is then shown that, in real-world practical applications, JSSP is a PUBO problem. Subsequently, This study analyses both QUBO and PUBO, specifically CUBO, problems within the QAOA framework and study the efficiency of both problems under the sa-QAOA algorithm as proposed by Farhi et al. [6]. I extend this investigation to other variants of QAOA, specifically QAOA-CD, am-QAOA and ma-QAOA and a novel ka-QAOA, which I propose in this dissertation. Their performance is evaluated across various graph structures, revealing that ka-QAOA is

a promising candidate to serve as a NISQ era algorithm for achieving high-quality results while keeping computational resources to a minimum.

5.2 | Research Directions

Many other enhancements of QAOA are proposed in the literature; however, no single solution is universally optimal. The No-Free Lunch (NFL) concept, which originates from fields such as economics, machine learning, and optimisation [93], asserts that no universal solution guarantees the best performance every time. This research demonstrates that an effective solution for one graph does not necessarily produce an efficient solution for others. Consequently, further investigation is required to explore a range of solutions across diverse problems and graph types, aiming to identify optimal strategies for each context. Additionally, there is a potential to study other novel variations, including hybrid strategies, for example, ka-QAOA with QAOA-CD, to determine whether these can yield improved results, keeping the required computational resources to the minimum.

5.3 | The Quantum Potential

When I started this journey, it was immediately apparent that, as an emerging field, quantum computing is still years away from being deployed in everyday applications. Research institutions, commercial organisations (such as IBM, Google, Rigetti, Quantinuum and D-Wave), and governments are heavily investing in quantum computing, achieving significant improvements while laying out ambitious targets and timelines. Quantum computers are transitioning from experimental devices to real-world tools that provide useful applications, albeit limited to simple use cases. Given the substantial risk that future fault-tolerant quantum computers may pose to the current security infrastructure, investment is primarily directed towards cryptography, both in quantum-safe classical cryptography algorithms as well as actual research into quantum technologies to provide quantum-secure communication, especially for Quantum Key Distribution (QKD). However, the robustness of post-quantum cryptography against attacks from future quantum computers and today's classical ones remains uncertain [94]. Other primary applications of quantum computing being investigated at the moment include research in drug discovery and optimisation, which formed the primary focus of this research.

Throughout history, technological advancements have continually reshaped society. From the invention of the wheel to the revolutionary developments of the past centuries -

such as aeroplanes, computers, the internet, and mobile phones - society has undergone profound transformations. Were a person from a century ago to step into our present day, they might believe they had stumbled into a work of science fiction, so utterly transformed is the world they once knew. Imagine a scenario where a quantum computer in a hospital would design a medicine tailored to a patient's DNA and effectively specifically target their ailment. In manufacturing, quantum algorithms will continuously optimise the production process, improving efficiency and profitability, while advanced AI systems will handle labour-intensive jobs with minimal human intervention. This evolution might allow humanity to focus more on enriching life, scientific research, knowledge, and philosophy. Although that future has not yet arrived, we live in an exciting era where we can help shape our collective destiny.

References

[1] David Gross. A century of quantum mechanics. In Daniele C. Struppa and Jeffrey M. Tollaksen, editors, *Quantum Theory: A Two-Time Success Story*, pages 3–8, Milano, 2014. Springer Milan. ISBN 978-88-470-5217-8. 1

[2] Hoi-Kwong Lo, Marcos Curty, and Kiyoshi Tamaki. Secure quantum key distribution. *Nature Photonics*, 8(8):595–604, 2014. ISSN 1749-4893. doi: 10.1038/nphoton.2014.149. URL `https://doi.org/10.1038/nphoton.2014.149`. 1

[3] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. *Rev. Mod. Phys.*, 92:015003, Mar 2020. doi: 10.1103/RevModPhys.92.015003. URL `https://link.aps.org/doi/10.1103/RevModPhys.92.015003`. 1

[4] Davide Venturelli, Dominic J. J. Marchand, and Galo Rojo. Quantum annealing implementation of job-shop scheduling, 2016. URL `https://arxiv.org/abs/1506.08479`. 1

[5] David Amaro, Matthias Rosenkranz, Nathan Fitzpatrick, Koji Hirano, and Mattia Fiorentini. A case study of variational quantum algorithms for a job shop scheduling problem. *EPJ Quantum Technology*, 9(1):5, 2022. ISSN 2196-0763. doi: 10.1140/epjqt/s40507-022-00123-4. URL `https://doi.org/10.1140/epjqt/s40507-022-00123-4`. 1

[6] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014. 3, 49, 50, 51, 52, 60, 61, 62, 63, 67, 68, 69, 70, 74, 131

[7] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, 1982. doi: 10.1007/BF02650179. 5

[8] David Deutsch. Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proceedings of the Royal Society of London*, 400(1818):97–117, 1985. URL https://royalsocietypublishing.org/doi/10.1098/rspa.1985.0070. 5

[9] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x. 5

[10] Benjamin Schumacher. Quantum coding. *Phys. Rev. A*, 51:2738–2747, Apr 1995. doi: 10.1103/PhysRevA.51.2738. URL https://link.aps.org/doi/10.1103/PhysRevA.51.2738. 5

[11] E. Schrödinger. The present status of quantum mechanics. *Die Naturwissenschaften*, 23:807–812, 1935. doi: 10.1007/BF01491891. URL https://doi.org/10.1007/BF01491891. 7

[12] E. Schrödinger. Discussion of Probability Relations between Separated Systems. *Proceedings of the Cambridge Philosophical Society*, 31(4):555, January 1935. doi: 10.1017/S0305004100013554. 8

[13] A. Einstein, B. Podolsky, and N. Rosen. Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Physical Review*, 47(10):777–780, May 1935. doi: 10.1103/PhysRev.47.777. 8

[14] Alain Aspect, Jean Dalibard, and Gérard Roger. Experimental test of bell's inequalities using time-varying analyzers. *Phys. Rev. Lett.*, 49:1804–1807, Dec 1982. doi: 10.1103/PhysRevLett.49.1804. URL https://link.aps.org/doi/10.1103/PhysRevLett.49.1804. 8

[15] David Deutsch and Richard Jozsa. Rapid solutions of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992. doi: 10.1098/rspa.1992.0167. 14

[16] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2: 79, August 2018. ISSN 2521-327X. doi: 10.22331/q-2018-08-06-79. URL https://doi.org/10.22331/q-2018-08-06-79. 15, 131

[17] John Preskill. Quantum computing and the entanglement frontier, 2012. 15

[18] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell,

Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. doi: 10.1038/s41586-019-1666-5. 15

[19] Davide Castelvecchi. A truly remarkable breakthrough: Google's new quantum chip achieves accuracy milestone. *Nature*, 576(7787):456–460, 2024. doi: 10.1038/d41586-024-04028-3. URL `https://doi.org/10.1038/d41586-024-04028-3`. 15

[20] Hui Wang, Yuan-Chao Wei, Xiao Jiang, Jian-Yu Guan, Qing-Yu Zhao, Yu-Hao Deng, Huan Li, Si Chen, Hao Li, You-Qi Sheng, et al. Strong quantum computational advantage using a superconducting quantum processor. *arXiv preprint arXiv:2106.14734*, 2021. URL `https://arxiv.org/abs/2106.14734`. 15

[21] IBM Quantum. On 'quantum supremacy'. `https://www.ibm.com/quantum/blog/on-quantum-supremacy`, 2019. URL `https://www.ibm.com/quantum/blog/on-quantum-supremacy`. IBM Quantum Blog. 16

[22] Feng Pan, Keyang Chen, and Pan Zhang. Solving the sampling problem of the sycamore quantum circuits. *Phys. Rev. Lett.*, 129:090502, Aug 2022. doi: 10.1103/PhysRevLett.129.090502. URL `https://link.aps.org/doi/10.1103/PhysRevLett.129.090502`. 16

[23] Walther Gerlach and Otto Stern. Das magnetische moment des silberatoms. *Zeitschrift für Physik*, 9:349–352, 1922. doi: 10.1007/BF01326983. URL `https://doi.org/10.1007/BF01326983`. 17

[24] Martin Bauer. The stern-gerlach experiment, translation of: "der experimentelle nachweis der richtungsquantelung im magnetfeld", 2023. 17

[25] Paul A. M. Dirac. The quantum theory of the electron. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 117(778):610–624, 1928. doi: 10.1098/rspa.1928.0023. URL https://doi.org/10.1098/rspa.1928.0023. 17

[26] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1): 253–258, 1925. doi: 10.1007/BF02980577. 18

[27] Ernst Ising. Contribution to the theory of ferromagnetism, 2000. URL http://www.hs-augsburg.de/~harsch/anglica/Chronology/20thC/Ising/isi_fm00.html. Translated from the original German "Beitrag zur Theorie des Ferromagnetismus". 18

[28] M. Born and V. Fock. Beweis des adiabatensatzes. *Zeitschrift für Physik*, 51(3): 165–180, 1928. ISSN 0044-3328. doi: 10.1007/BF01343193. URL https://doi.org/10.1007/BF01343193. 19

[29] Norman Biggs, E. Keith Lloyd, and Robin J. Wilson. *Graph Theory, 1736-1936*. Oxford University Press, Oxford, 1976. ISBN 9780198539162. 23

[30] Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1): 96–115, 1927. URL http://eudml.org/doc/211191. 29

[31] Merrill M. Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956. doi: 10.1287/opre.4.1.61. URL https://doi.org/10.1287/opre.4.1.61. 29

[32] J. D. Biamonte. Nonperturbative k-body to two-body commuting conversion hamiltonians and embedding problem instances into ising spins. *Phys. Rev. A*, 77:052331, May 2008. doi: 10.1103/PhysRevA.77.052331. URL https://link.aps.org/doi/10.1103/PhysRevA.77.052331. 33

[33] Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1):155–225, 2002. ISSN 0166-218X. doi: https://doi.org/10.1016/S0166-218X(01)00341-9. URL https://www.sciencedirect.com/science/article/pii/S0166218X01003419. 33

[34] Joshua T. Iosue. Qubovert: A python package for converting combinatorial optimization problems into qubo and ising models, 2020. URL https://github.com/jtiosue/qubovert. 34

[35] Jonas Stein, Farbod Chamanian, Maximilian Zorn, Jonas Nüßlein, Sebastian Zielinski, Michael Kölle, and Claudia Linnhoff-Popien. Evidence that pubo outperforms qubo when solving continuous optimization problems with the qaoa. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, page 2254–2262, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701207. doi: 10.1145/3583133.3596358. URL https://doi.org/10.1145/3583133.3596358. 34

[36] Richard M. Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming*, 1972. URL https://api.semanticscholar.org/CorpusID:33509266. 41

[37] Clayton W. Commander. *Maximum cut problem, MAX-CUTMaximum Cut Problem, MAX-CUT*, pages 1991–1999. Springer US, Boston, MA, 2009. ISBN 978-0-387-74759-0. doi: 10.1007/978-0-387-74759-0_358. URL https://doi.org/10.1007/978-0-387-74759-0_358. 41

[38] Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988. doi: 10.1287/opre.36.3.493. URL https://doi.org/10.1287/opre.36.3.493. 41

[39] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. ISBN 9780521424264. URL http://doi.org/10.1017/CBO9780511804090. 46

[40] Brilliant Complexity Theory. Brilliant Complexity Theory. https://brilliant.org/wiki/complexity-theory/, n.d. Accessed: 2024-02-22. 46

[41] Complexity Zoo. Complexity Zoo. https://complexityzoo.net/, n.d. Accessed: 2024-02-22. 47

[42] Tameem Albash and Daniel A. Lidar. Adiabatic quantum computation. *Rev. Mod. Phys.*, 90:015002, Jan 2018. doi: 10.1103/RevModPhys.90.015002. URL https://link.aps.org/doi/10.1103/RevModPhys.90.015002. 47

[43] Florian Klug. Quantum algorithms for solving the traveling salesman problem. *SSRN Electronic Journal*, 2024. doi: 10.2139/ssrn.4836033. URL https://doi.org/10.2139/ssrn.4836033. Available at SSRN: https://doi.org/10.2139/ssrn.4836033. 47

[44] Bela Bauer, Sergey Bravyi, Mario Motta, and Garnet Kin-Lic Chan. Quantum algorithms for quantum chemistry and quantum materials science. *Chem. Rev.*, 120 (22):12685–12717, November 2020. ISSN 0009-2665, 1520-6890. doi: 10.1021/acs.chemrev.9b00829. 48

[45] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning. *arXiv preprint arXiv:2001.03622*, February 2020. 48

[46] David Amaro, Carlo Modica, Matthias Rosenkranz, Mattia Fiorentini, Marcello Benedetti, and Michael Lubasch. Filtering variational quantum algorithms for combinatorial optimization. *Quantum Sci. Technol.*, 7(1):015021, January 2022. ISSN 2058-9565. doi: 10.1088/2058-9565/ac3e54. 48

[47] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):–4213, 2009. doi: 10.1038/ncomms5213. URL `https://doi.org/10.1038/ncomms5213`. 48

[48] 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):1791, mar 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-21728-w. URL `https://doi.org/10.1038/s41467-021-21728-w`. 48

[49] Andrew Arrasmith, Zoë Holmes, M Cerezo, and Patrick J Coles. Equivalence of quantum barren plateaus to cost concentration and narrow gorges. *Quantum Science and Technology*, 7(4):045015, aug 2022. doi: 10.1088/2058-9565/ac7d06. URL `https://dx.doi.org/10.1088/2058-9565/ac7d06`. 48

[50] Daniel Müssig, Markus Wappler, Steve Lenk, and Jörg Lässig. Connecting the hamiltonian structure to the qaoa performance and energy landscape, 2024. URL `https://arxiv.org/abs/2407.04435`. 48

[51] Karin Eberhard. The effects of visualization on judgment and decision-making: a systematic literature review. *Management Review Quarterly*, 73(1):167–214, feb 2023. ISSN 2198-1639. doi: 10.1007/s11301-021-00235-8. URL `https://doi.org/10.1007/s11301-021-00235-8`. 48

[52] Lavinia Stiliadou, Johanna Barzen, Frank Leymann, Alexander Mandl, and Benjamin Weder. Exploring the cost landscape of variational quantum algorithms. In

Marco Aiello, Johanna Barzen, Schahram Dustdar, and Frank Leymann, editors, *Service-Oriented Computing*, pages 128–142, Cham, 2025. Springer Nature Switzerland. 48

[53] Manuel S. Rudolph, Sukin Sim, Asad Raza, Michal Stechly, Jarrod R. McClean, Eric R. Anschuetz, Luis Serrano, and Alejandro Perdomo-Ortiz. Orqviz: Visualizing high-dimensional landscapes in variational quantum algorithms, 2021. URL `https://arxiv.org/abs/2111.04695`. 49

[54] Ji Liu and Huiyang Zhou. Reliability Modeling of NISQ- Era Quantum Computers . In *2020 IEEE International Symposium on Workload Characterization (IISWC)*, pages 94–105, Los Alamitos, CA, USA, October 2020. IEEE Computer Society. doi: 10.1109/IISWC50251.2020.00018. URL `https://doi.ieeecomputersociety.org/10.1109/IISWC50251.2020.00018`. 51, 129

[55] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2), 2019. ISSN 1999-4893. doi: 10.3390/a12020034. URL `https://www.mdpi.com/1999-4893/12/2/34`. 52

[56] Lennart Binkowski, Gereon Koßmann, Timo Ziegler, and Rene Schwonnek. Elementary proof of qaoa convergence. *New Journal of Physics*, 2024. URL `http://iopscience.iop.org/article/10.1088/1367-2630/ad59bb`. 52

[57] Michał Stęchły. Quantum approximate optimization algorithm explained. `https://www.mustythoughts.com/quantum-approximate-optimization-algorithm-explained`, 2022. Accessed: 2024-06-22. 52, 53

[58] Masuo Suzuki. Generalized trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. *Communications in Mathematical Physics*, 51(2):183–190, 1976. doi: 10.1007/BF01609348. URL `https://sci-hub.se/10.1007/BF01609348`. [cited 2023 Jan 15]. 52

[59] Masuo Suzuki. On the convergence of exponential operators—the zassenhaus formula, bch formula and systematic approximants. *Communications in Mathematical Physics*, 57(3):193–200, 1977. doi: 10.1007/BF01614161. URL `https://sci-hub.se/10.1007/BF01614161`. [cited 2023 Jan 15]. 53

[60] Pranav Chandarana, Narendra N. Hegade, Iraitz Montalban, Enrique Solano, and Xi Chen. Digitized counterdiabatic quantum algorithm for protein folding. *Phys. Rev. Appl.*, 20:014024, Jul 2023. doi: 10.1103/PhysRevApplied.20.014024. URL https://link.aps.org/doi/10.1103/PhysRevApplied.20.014024. 53, 55, 69, 81

[61] Jonathan Wurtz and Peter J. Love. Counterdiabaticity and the quantum approximate optimization algorithm. *Quantum*, 6:635, January 2022. ISSN 2521-327X. doi: 10.22331/q-2022-01-27-635. URL https://doi.org/10.22331/q-2022-01-27-635. 53

[62] Mara Vizzuso, Gianluca Passarelli, Giovanni Cantele, and Procolo Lucignano. Convergence of digitized-counterdiabatic qaoa: circuit depth versus free parameters. *New Journal of Physics*, 26(1):013002, jan 2024. doi: 10.1088/1367-2630/ad1536. URL https://dx.doi.org/10.1088/1367-2630/ad1536. 53, 55

[63] Dmitry Panchenko. Introduction to the sk model, 2014. URL https://arxiv.org/abs/1412.0170. 55

[64] Pranav Chandarana, Narendra N. Hegade, Iraitz Montalban, Enrique Solano, and Xi Chen. Digitized counterdiabatic quantum algorithm for protein folding. *Phys. Rev. Appl.*, 20:014024, Jul 2023. doi: 10.1103/PhysRevApplied.20.014024. URL https://link.aps.org/doi/10.1103/PhysRevApplied.20.014024. 55

[65] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, Cambridge, UK, 2011. ISBN 978-0-521-11839-8. URL http://www.designofapproxalgs.com/. 56

[66] Joseph Raphson. *Analysis Aequationum Universalis*. Apud A. & J. Churchill, London, 1690. 59

[67] C.G. Broyden. The convergence of a class of double-rank minimization algorithms: 2. the new algorithm. *IMA Journal of Applied Mathematics*, 6:222–231, 1970. doi: 10.1093/imamat/6.3.222. 60

[68] R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13: 317–322, 1970. doi: 10.1093/comjnl/13.3.317. 60

[69] D. Goldfarb. A family of variable metric updates derived by variational means. *Mathematics of Computation*, 24:23–26, 1970. doi: 10.2307/2004873. 60

[70] D.F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24:647–656, 1970. doi: 10.2307/2004840. 60

[71] Aidan Pellow-Jarman, Ilya Sinayskiy, Anban Pillay, and Francesco Petruccione. A comparison of various classical optimizers for a variational quantum linear solver. *Quantum Information Processing*, 20(6):202, 2021. ISSN 1573-1332. doi: 10.1007/s11128-021-03140-x. URL https://doi.org/10.1007/s11128-021-03140-x. 60

[72] Simone Tibaldi, Davide Vodola, Edoardo Tignone, and Elisa Ercolessi. Bayesian optimization for qaoa. *IEEE Transactions on Quantum Engineering*, 4:1–11, 2023. doi: 10.1109/TQE.2023.3325167. 60

[73] Andreas Bartschi and Stephan Eidenbenz. Grover Mixers for QAOA: Shifting Complexity from Mixer Design to State Preparation . In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 72–82, Los Alamitos, CA, USA, October 2020. IEEE Computer Society. doi: 10.1109/QCE49297.2020.00020. URL https://doi.ieeecomputersociety.org/10.1109/QCE49297.2020.00020. 61

[74] Stefan H. Sack and Maksym Serbyn. Quantum annealing initialization of the quantum approximate optimization algorithm. *Quantum*, 5:491, July 2021. ISSN 2521-327X. doi: 10.22331/q-2021-07-01-491. URL https://doi.org/10.22331/q-2021-07-01-491. 61, 116, 123, 129

[75] Rebekah Herrman, Phillip C. Lotshaw, James Ostrowski, Travis S. Humble, and George Siopsis. Multi-angle quantum approximate optimization algorithm. *Scientific Reports*, 12(1):6781, 2022. doi: 10.1038/s41598-022-10555-8. URL https://doi.org/10.1038/s41598-022-10555-8. 61, 62, 63, 70, 74

[76] Kaiyan Shi, Rebekah Herrman, Ruslan Shaydulin, Shouvanik Chakrabarti, Marco Pistoia, and Jeffrey Larson. Multiangle qaoa does not always need all its angles. In *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, pages 414–419, 2022. doi: 10.1109/SEC54971.2022.00062. 63

[77] Ali Javadi-Abhari, David C McKay, Thomas Alexander, Luciano Bello, Michael J Biercuk, Lev S Bishop, Jun-Song Chen, Jerry M Chow, Colleen Claus, Andrew Cross, et al. Quantum computing with qiskit. *arXiv preprint arXiv:2405.08810*, 2023. 65

[78] Yasunari Suzuki, Yoshiaki Kawase, Yuya Masumura, Yuria Hiraga, Masahiro Nakadai, Jiabao Chen, Ken M. Nakanishi, Kosuke Mitarai, Ryosuke Imai, Shiro Tamiya, Takahiro Yamamoto, Tennin Yan, Toru Kawakubo, Yuya O. Nakagawa, Yohei Ibe, Youyuan Zhang, Hirotsugu Yamashita, Hikaru Yoshimura, Akihiro Hayashi, and Keisuke Fujii. Qulacs: a fast and versatile quantum circuit simulator

for research purpose. *Quantum*, 5:559, October 2021. ISSN 2521-327X. doi: 10.22331/q-2021-10-06-559. URL https://doi.org/10.22331/q-2021-10-06-559. 65

[79] Paul Erdős and Alfréd Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959. doi: 10.5486/pmd.1959.6.3-4.12. 69

[80] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960. doi: 10.1007/BF02024133. 69

[81] Edgar N. Gilbert. Random graphs. *Annals of Mathematical Statistics*, 30(4):1141–1144, 1959. doi: 10.1214/aoms/1177706098. 69

[82] Neriman Kartal and Şenol Kartal. Complex dynamics of covid-19 mathematical model on erdos renyi network. *International Journal of Biomathematics*, 16, 08 2022. doi: 10.1142/S1793524522501108. 70

[83] Ohad Amosy, Tamuz Danzig, Ohad Lev, Ely Porat, Gal Chechik, and Adi Makmal. Iteration-free quantum approximate optimization algorithm using neural networks. *Quantum Machine Intelligence*, 6(2):38, 2024. ISSN 2524-4914. doi: 10.1007/s42484-024-00159-y. URL https://doi.org/10.1007/s42484-024-00159-y. 70

[84] Navid Imani and Hamid Sarbazi-Azad. Properties of a hierarchical network based on the star graph. *Information Sciences*, 180(14):2802–2813, 2010. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2010.03.029. URL https://www.sciencedirect.com/science/article/pii/S0020025510001490. Including Special Section on Hybrid Intelligent Algorithms and Applications. 74

[85] Yue Ruan, Samuel Marsh, Xilin Xue, Xi Li, Zhihao Liu, and Jingbo Wang. Quantum approximate algorithm for np optimization problems with constraints, 2020. URL https://arxiv.org/abs/2002.00943. 74

[86] Teague Tomesh, Nicholas Allen, Daniel Dilley, and Zain Saleem. Quantum-classical tradeoffs and multi-controlled quantum gate decompositions in variational algorithms. *Quantum*, 8:1493, October 2024. ISSN 2521-327X. doi: 10.22331/q-2024-10-04-1493. URL https://doi.org/10.22331/q-2024-10-04-1493. 83

[87] L. Isenhower, M. Saffman, and K. Mølmer. Multibit CkNOT quantum gates via rydberg blockade. *Quantum Information Processing*, 10(6):755–771, 2011. ISSN

1573-1332. doi: 10.1007/s11128-011-0292-4. URL https://doi.org/10.1007/s11128-011-0292-4. 84

[88] Jiaan Qi, Zhi-Hai Liu, and Hongqi Xu. Scalable multi-qubit intrinsic gates in quantum dot arrays, 2024. URL https://arxiv.org/abs/2403.06894. 84

[89] Pascal Bassler, Matthias Zipper, Christopher Cedzich, Markus Heinrich, Patrick H. Huber, Michael Johanning, and Martin Kliesch. Synthesis of and compilation with time-optimal multi-qubit gates. *Quantum*, 7:984, April 2023. ISSN 2521-327X. doi: 10.22331/q-2023-04-20-984. URL https://doi.org/10.22331/q-2023-04-20-984. 84

[90] Xiao-Le Li, Ziyu Tao, Kangyuan Yi, Kai Luo, Libo Zhang, Yuxuan Zhou, Song Liu, Tongxing Yan, Yuanzhen Chen, and Dapeng Yu. Hardware-efficient and fast three-qubit gate in superconducting quantum circuits. *Frontiers of Physics*, 19 (5):51205, may 2024. ISSN 2095-0470. doi: 10.1007/s11467-024-1405-8. URL https://doi.org/10.1007/s11467-024-1405-8. 84

[91] V. Akshay, D. Rabinovich, E. Campos, and J. Biamonte. Parameter concentrations in quantum approximate optimization. *Physical Review A*, 104(1), July 2021. ISSN 2469-9934. doi: 10.1103/physreva.104.l010401. URL http://dx.doi.org/10.1103/PhysRevA.104.L010401. 129

[92] Seth Lloyd. Ultimate physical limits to computation. *Nature*, 406(6799):1047–1054, 2000. 131

[93] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi: 10.1109/4235.585893. 132

[94] Alvaro Cintas Canto, Jasmin Kaur, Mehran Mozaffari Kermani, and Reza Azarderakhsh. Algorithmic security is insufficient: A comprehensive survey on implementation attacks haunting post-quantum security, 2023. URL https://arxiv.org/abs/2305.13544. 132