

DIPLOMA THESIS

# Resource efficient quantum-classical computing and applications in scheduling and optimization problems

School of Electrical & Computer Engineering  
Technical University of Crete



*Author:*

Nikolaos Venetis

*Thesis Supervisor:*

Prof. Dimitris G. Angelakis

*Thesis Examiners:*

Prof. Dionysios Christopoulos

Prof. Thrasyvoulos Spyropoulos

---

June, 2025

# Abstract

This thesis investigates the application of hybrid quantum-classical algorithms to solve Job Shop Scheduling Problems (JSSP), a class of NP-hard combinatorial optimization problems. The work begins by introducing the foundational principles of quantum mechanics and quantum computing, highlighting their unique characteristics that hold the potential to revolutionize computational paradigms.

Then, teleportation is presented to demonstrate how these quantum properties can be harnessed in practice in real quantum algorithms. The study then explores the use of Quadratic Unconstrained Binary Optimization (QUBO) as a framework for representing and solving combinatorial optimization problems using both classical and quantum computational resources.

State-of-the-art quantum approaches, such as the Quantum Approximate Optimization Algorithm (QAOA) and the Variational Quantum Eigensolver (VQE)—implemented with a Hardware-Efficient Ansatz (HEA)—are discussed in depth. These algorithms are tested on benchmark problems including Max-Cut and Subset Sum, establishing a foundation for their application to more complex scheduling problems.

A particular instance of the JSSP is then formulated mathematically, with all necessary constraints rigorously defined. This formulation is translated into a QUBO-compatible representation to enable its execution on quantum backends. Initial experiments using QAOA and HEA are conducted on small problem instances to assess the correctness and feasibility of the proposed formulation.

Recognizing the limitations of current quantum hardware—particularly when scaling beyond toy problems—the thesis introduces qubit-efficient encoding schemes as a strategy to address scalability challenges inherent in conventional quantum approaches. These schemes are evaluated using standard problems such as Subset Sum before being applied to a scaled-up version of the JSSP, approaching instances that resemble real-world scheduling tasks.

Finally, real quantum hardware accessed via cloud platforms is used to run selected problem instances, providing insight into the behavior of the algorithms under realistic noise conditions. The findings suggest that while solving multi-constraint combinatorial

---

problems with compact QUBO formulations remains challenging, the continued evolution of quantum hardware offers promising potential. This work contributes to the advancement of quantum computing techniques for addressing complex scheduling problems and lays the groundwork for future research in this domain.

# Acknowledgments

I would like to sincerely thank Professor Dimitris G. Angelakis, my thesis supervisor, for his invaluable guidance and support in bringing this work to completion. I am also grateful to the Center for Quantum Technologies at the National University of Singapore for their support during my visit there. This experience, combined with the daily interactions with our collaborators through joint groups, greatly enriched this research.

# Contents

<b>1</b>	<b>Introduction</b>	<b>10</b>
<b>2</b>	<b>Background in Quantum Computing</b>	<b>13</b>
2.1	Qubit Definition . . . . .	13
2.1.1	What is a Qubit? . . . . .	13
2.1.2	Mathematical Notation . . . . .	14
2.1.3	Linear Algebra Concepts Needed . . . . .	15
2.1.4	Bloch Sphere Representation . . . . .	17
2.2	Single Qubit Gates . . . . .	18
2.3	Two Qubit Gates . . . . .	21
2.4	Quantum Entanglement . . . . .	22
2.5	Hamiltonian Definition . . . . .	23
2.6	Teleportation . . . . .	23
<b>3</b>	<b>Classical to Hybrid Optimization</b>	<b>26</b>
3.1	QUBO Problems . . . . .	28
3.2	Ising Model . . . . .	30
3.3	From QUBO to Ising . . . . .	31
3.4	Adiabaticity and Quantum Annealing . . . . .	32
3.5	Hybrid Quantum-Classical Optimization . . . . .	33
3.6	Algorithms . . . . .	35
3.6.1	Quantum Approximate Optimization Algorithm . . . . .	35
3.6.2	Hardware Efficient Ansatz . . . . .	38
3.7	Application of Quantum Algorithms in Specific Problems . . . . .	39
3.7.1	Max-Cut and Subset Sum . . . . .	39
3.7.2	Max-Cut using QAOA and HE Ansatz . . . . .	41
3.7.3	Subset Sum using QAOA and HE Ansatz . . . . .	45
<b>4</b>	<b>Quantum Optimization in Scheduling Problems</b>	<b>48</b>
4.1	Scheduling Problems . . . . .	49

4.2	The Manufacturing Production Scheduling Problem . . . . .	50
4.2.1	Problem Introduction . . . . .	50
4.2.2	Introducing the Disjunctive QUBO Model . . . . .	50
4.2.3	Mathematical Formulation . . . . .	52
4.2.4	Algorithmic Approach . . . . .	57
4.2.5	Result Validation . . . . .	70
<b>5</b>	<b>Qubit Efficient Quantum Optimization for Scheduling Problems</b>	<b>71</b>
5.1	Qubit Efficient Schemes explained . . . . .	74
5.2	Qubit Efficient Formulation applied . . . . .	77
5.2.1	Subset Sum Problem . . . . .	77
5.2.2	Manufacturing Production Scheduling Problem using Qubit Efficient Schemes. . . . .	81
5.2.3	Execution on Cloud QPUs . . . . .	95
<b>6</b>	<b>Conclusion and Future Work</b>	<b>99</b>

# List of Figures

1.1	Moore's law plateau. . . . .	11
2.1	Representation of Hilbert space for one qubit. The figure above shows a qubit that has a 64% chance of being measured 0 and a 36% chance of being measured 1. . . . .	15
2.2	Bloch sphere. We can notice that $ \psi\rangle$ can be measured in three different orthonormal basis. . . . .	18
2.3	Identity gate representation. . . . .	19
2.4	Pauli-X gate representation. . . . .	19
2.5	Pauli-Y gate representation. . . . .	19
2.6	Pauli-Z gate representation. . . . .	20
2.7	Hadamard gate representation. . . . .	20
2.8	Teleportation circuit. . . . .	24
2.9	Bob operations according to Alice's outcome. . . . .	25
3.1	Table of some known penalties. . . . .	29
3.2	Spins on a lattice representation. . . . .	30
3.3	Ground state and first excited state of a Hamiltonian. . . . .	32
3.4	Quantum tunneling, a property of phasing through a thermal jump. . . . .	33
3.5	Big picture of a VQA. . . . .	34
3.6	Multi-layer QAOA with depth=p . . . . .	36
3.7	$R_{ZZ}$ gate. . . . .	37
3.8	Single rotation $R_Z$ gate. . . . .	37
3.9	Hardware Efficient ansatz first full layer and second layer's single rotation gates with N qubits. . . . .	38
3.10	A random graph and its Max-Cut. . . . .	40
3.11	A 5-node graph with a simple and visible solution. . . . .	42
3.12	Ansatz for QAOA and Max-Cut with 5 nodes. . . . .	42



3.13	QAOA converges fast after 40 iterations in the optimal point. Using such a shallow circuit with only 1-Layer and a small amount of total gates makes our algorithm find the optimal solution with some probability but not fully converging to the optimal solution with close to total accuracy. . . . .	43
3.14	Probabilities of the measurements are distributed on the optimal solutions in a better way, and that is why cost function has a value close to 5 when Max-Cut is 5. 10 experiments are conducted and we can see that the best one is really close to optimal. . . . .	43
3.15	It is noticeable that the cost function converges fast. The optimal solutions are found with just one layer and all other solutions have zero probability of being measured in a noiseless simulation. . . . .	45
3.16	Distribution indicates that optimal solution is found with a big probability and that the cost function converges. . . . .	46
3.17	Distribution indicates that optimal solution is found with a big probability but smaller than the one-layer and that the cost function converges. . .	46
3.18	Distribution indicates that almost always the optimal solution is found even with one layer and that the cost function almost instantly converges to the optimal value of the cost function. . . . .	47
4.1	The first figure shows the ordering of the operations of each Job and in which machine each operation can be assigned. In the second figure we see a potential schedule that satisfy the constraints and has a certain makespan.	51
4.2	From the matrices created to represent the scheduling problem to qubits used as input for the quantum hardware. . . . .	54
4.3	Snippet of pseudo-code of Augmented Lagrangian method. . . . .	57
4.4	A toy example of 4 tasks and 2 different Jobs. Tasks of different jobs are denoted with a different color. . . . .	57
4.5	Ordering is denoted by the indices of the Tasks so ID 1 must be finished prior to the start of ID 2. . . . .	58
4.6	A possible inactivity window and how it would be handled by the optimization algorithm. . . . .	58
4.7	The outline of qubits representing our classical variables. . . . .	58
4.8	Cost function of 1-Layer QAOA using 8 qubits. The value it converges is sub-optimal, the depth of the circuit is not enough to capture the complexity of the problem. . . . .	60
4.9	Gantt chart produced. The solution of the 1-Layer QAOA is not in the feasible set because task ordering constraint is violated. . . . .	60

4.10	Cost function for 12-Layer QAOA. More iterations are needed for the cost function to converge but a better solution is produced. . . . .	61
4.11	Gantt chart produced. The optimal solution that lies in the feasible set is depicted. . . . .	62
4.12	Cost function for 20-Layer QAOA. In this cost function that correlations between qubit are sufficiently expressed by the deepness of the circuit we can compare with the 12-Layer one and we can observe we reach a worse solution. . . . .	63
4.13	Gantt chart produced. The solution is feasible but it is sub-optimal so a timestep is not exploited as it should. . . . .	63
4.14	First Lagrange iteration. A constraint is violated here so the corresponding qubits are penalized accordingly using a different QUBO matrix. . . . .	64
4.15	Last Lagrangian iteration. Algorithm terminates, indicating a feasible solution has been found. . . . .	65
4.16	Gantt chart produced by the last iteration that both lays in the feasible set and is optimal. . . . .	65
4.17	First iteration cost function. A feasible solution is not reached so more iterations are needed. . . . .	66
4.18	Last iteration cost function where all the constraints are met. . . . .	67
4.19	Gantt chart for 3-Layer HEA. A feasible solution has been reached but it is sub-optimal. . . . .	68
4.20	Cost function for QAOA and HEA. . . . .	69
4.21	A promising solutions has been reached. In a more complex scenario with 4 total constraint the optimal solution was met. . . . .	69
5.1	Cost function of 12 variables Subset Sum. . . . .	78
5.2	Cost function for different encodings on Subset sum. . . . .	79
5.3	Cost function for Subset Sum with a 50 variable vector . . . . .	80
5.4	Cost function for 100 variables. . . . .	81
5.5	Bitstring conversion after simplification. . . . .	82
5.6	Gantt chart. . . . .	83
5.7	HEA of 7 qubits from a starting total of 50 qubits and a total of 3-Layers. . . . .	84
5.8	Cost function for different encoding schemes. The problem consists of 50 classical variables and 47 constraints. . . . .	84
5.9	Gantt Chart for minimal encoding with a ratio of satisfied constraints reaching 70%. . . . .	85

5.10 3 ancilla qubit solution Gantt chart with a total of 75% ratio of satisfied constraints. . . . .	86
5.11 5 ancilla qubit solution Gantt chart with a ratio of around 91%. . . . .	86
5.12 4 Layer 5 ancilla Gantt chart reaching an 83% ratio of satisfied constraints. . . . .	87
5.13 Google-OR Tools solution Gantt chart showing the optimal solution we should get. . . . .	88
5.14 Gantt Chart of the best experiment with minimal encoding . . . . .	89
5.15 Gantt chart for 3-ancilla encoding schemes. . . . .	89
5.16 Gantt chart for 5-ancilla qubits encoding scheme reaching an 83% ratio of satisfied constraints. . . . .	90
5.17 Gantt Chart for the 4-Layered experiment. . . . .	91
5.18 Cost function for 28 tasks problem. This experiment consists of 200 classical variables and a total of around 200 constraints to be satisfied. . . . .	92
5.19 Minimal encoding for 200 variables. . . . .	92
5.20 3 ancilla qubits Gantt chart. . . . .	93
5.21 3-Layers 5 ancilla qubits Gantt chart. . . . .	94
5.22 4-Layered 5-ancilla encoding scheme gantt chart. . . . .	95
5.23 Small scale experiment CDF. . . . .	96
5.24 Medium scale experiment CDF. . . . .	97
5.25 Large scale experiment CDF. . . . .	97

# Chapter 1

## Introduction

### A never-ending quest

Since the dawn of civilization, humans—driven by curiosity and innovation—have engaged in a relentless pursuit of progress. In the era of the Technological Revolution, this pursuit has become one of humanity’s primary objectives. Over the past four centuries, technological advancement has accelerated dramatically: from the use of the abacus in ancient times to the development of modern central processing units (CPUs) capable of executing millions of computations in mere fractions of a second.

A key observation fueling this rapid growth is Moore’s Law, formulated by Gordon Moore in 1965, which states that the number of transistors on a microchip doubles approximately every two years [4]. This exponential growth in computational power has been accompanied by a corresponding decrease in transistor size. However, as transistor dimensions approach the nanometer scale, classical physics begins to break down. At such scales, quantum mechanical effects—including quantum decoherence and tunneling—begin to dominate, introducing probabilistic behavior and undermining the deterministic operation of classical transistors.

In response to these physical limitations, quantum computing emerged as a promising alternative. The foundational idea was introduced by Richard Feynman in 1981 [5], who proposed that quantum systems could simulate other quantum systems more efficiently than classical computers. This concept was later formalized by David Deutsch in 1985 [6], who developed the notion of a universal quantum computer.

Unlike classical computers, which process bits in definite states (0 or 1), quantum computers utilize quantum bits (qubits) that can exist in superpositions of states. Leveraging phenomena such as superposition and entanglement, quantum computers can explore multiple computational paths simultaneously, potentially offering exponential speed-ups for specific classes of problems.

The long-term goal of the field is to achieve quantum supremacy—the point at which quantum computers can solve certain computational problems more efficiently than any known classical algorithm. This milestone was reportedly reached by Google in 2019 [7], demonstrating that quantum machines may soon tackle real-world NP-hard problems previously deemed intractable for classical systems.

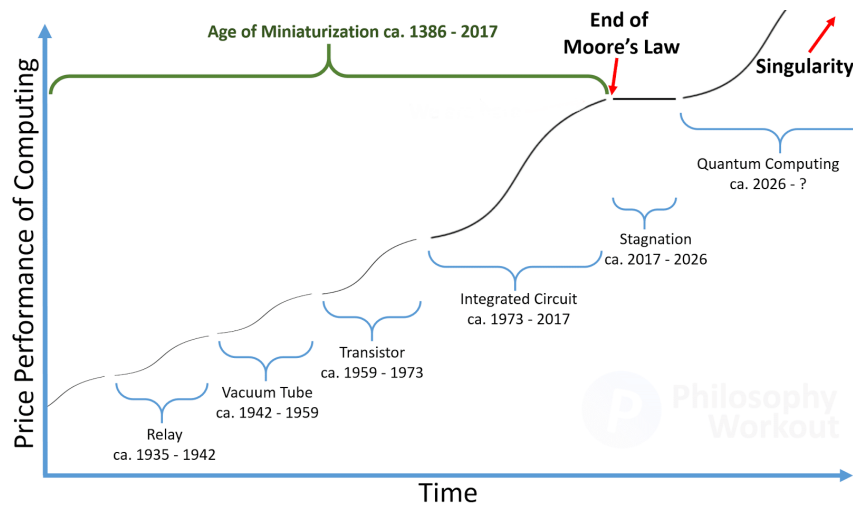


Figure 1.1: Moore's law plateau.

## Quantum computing breakthroughs

After **Richard Feynman** and **David Deutsch's** innovative ideas, which pioneered the basis of quantum computing more breakthroughs were made in the field of quantum mechanics. First of the was in 1985, a Deutsch's formulation of a universal quantum computer, capable of performing any computation achievable by a classical computer but with exponentially greater efficiency for certain problems. Then the first practical algorithm breakthrough came in 1994 when **Peter Shor** developed Shor's algorithm, which showed that quantum computers could factorize large integers exponentially faster than classical methods, threatening modern encryption systems. Lastly around the same time **Lov Grover** introduced an algorithm which provided quadratic speedup for unstructured search problems. All those quantum breakthroughs transitioned quantum computing from a theoretical fantasy, to a field of immense practical, scientific and computational potential

## Quantum technology affecting optimization

---

Optimization is an extremely highly active of research in both classical and quantum engineering. Optimization is defined as the way of finding the best solution, minimizing a function, choosing from every possible input the function can take. Quantum computing excels at calculating every outcome simultaneously and also at finding the outcome producing the minimum energy, a fact that can be translated as "minimizing a function". There are several fields that quantum optimization can come into play. For transportation, the most efficient routes can be computed quickly solving combinatorial problems exponentially faster than classical computers. In finance, portfolio optimization can be done, improving investment strategies. Lastly, scheduling optimization problems, a sector that we aim to tackle in this thesis, can be solved, finding the optimal solution exponentially faster than classical computers.

All these applications and optimization algorithms may in theory work but quantum hardware cannot tackle real-world large-scale problems yet, because it is in really early stages yet.

# Chapter 2

## Background in Quantum Computing

This chapter presents the theoretical background necessary to understand the foundational concepts of quantum computing, with a particular focus on quantum optimization. It begins by introducing the fundamental unit of quantum information—the qubit—including a detailed examination of its mathematical formulation and physical interpretation. The Bloch sphere will be employed as a visual and conceptual tool to represent the state of a single qubit, highlighting the differences between classical bits and quantum states.

Following the definition of a qubit, the chapter will explore quantum operations, also known as quantum gates. Both single-qubit and two-qubit gates will be analyzed, with an emphasis on their matrix representations and circuit-level implementations. This section will provide essential examples such as the Pauli gates, Hadamard gate, and controlled-NOT (CNOT) gate, which serve as building blocks for more complex quantum circuits.

The chapter concludes with a discussion of quantum entanglement—one of the most intriguing and non-classical phenomena in quantum mechanics. To illustrate its significance and practical applications, the Quantum Teleportation Protocol will be examined. This protocol not only demonstrates the power of entanglement but also exemplifies the principles underlying quantum information transfer.

### 2.1 Qubit Definition

#### 2.1.1 What is a Qubit?

Qubit is the quantum analogous for the bit in classical computing. Classical bit can only take two values **0** or **1**. Qubit, being the fundamental unit in quantum computing, can take a combination of values between **0** and **1** called a superposition of states. After a measurement is performed quantum state collapses in one of the two states, but pre-

measurement quantum wave behavior properties lead to a probabilistic model from which we can derive interesting results.

### 2.1.2 Mathematical Notation

The most widely used notation to represent qubit states and operations between one or more qubits is Dirac notation which was introduced by **Paul Dirac** in 1920s. A different name for Dirac notation is Braket notation. Ket represents a vector  $|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$  and bra represents its conjugate transpose  $|\psi\rangle^\dagger = \langle\psi| = [a, b]$ .

The inner product of a bra and a ket is denoted as  $\langle\psi|\phi\rangle$  and equals a scalar value. The inner product of two same vectors equals one because they are parallel to each other.

The simplest and most accurate mathematical tool to represent qubits is Hilbert space denoted as  $\mathcal{H}$ . The Hilbert space is a high-dimensional space used to describe systems with one or more qubits. The simplest Hilbert space is for one qubit and consist of two orthonormal vectors

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

that form the basis state of the space.

The difference from classical computing is that in quantum computing a qubit can be in both states simultaneously with some probability. This property is called superposition and it is denoted as

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle.$$

As described earlier the superposition collapses to just one state after the measurement either  $|0\rangle$  or  $|1\rangle$ . The  $\alpha$  and  $\beta$  denote the root of the probability of the qubit to collapse for each state, respectively.

So if a superposition is described as

$$\sum_{i=0}^n c_i |\psi_i\rangle \quad \text{then} \quad \sum_{i=0}^n c_i^2 = 1.$$

For one qubit we can represent superposition in a circle. Knowing the state the qubit is, operations can be performed to change the state of the qubit, meaning changing the probability of it collapsing in a state after measurement.



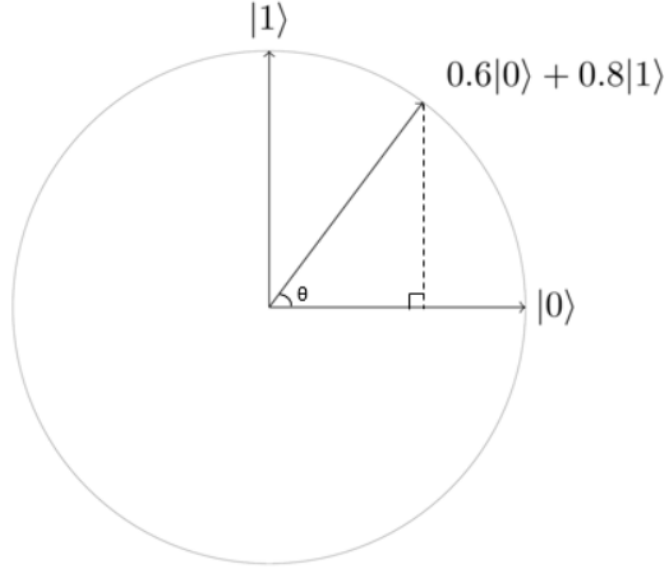


Figure 2.1: Representation of Hilbert space for one qubit. The figure above shows a qubit that has a 64% chance of being measured 0 and a 36% chance of being measured 1.

In the above figure if we take a measurement of the qubit the probability of it being at state  $\mathcal{P}(|1\rangle) = 0.36$  when the complementary probability is  $\mathcal{P}(|0\rangle) = 0.64$ .

### 2.1.3 Linear Algebra Concepts Needed

#### Hermitian Conjugate

The Hermitian conjugate denoted as  $(\dagger)$  is the transpose of the complex conjugate of a matrix, so for a matrix  $M$

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

If the dagger of a matrix is equal to the matrix itself then the matrix is called Hermitian operator or self-adjoint operator.

$$M^\dagger = M$$

#### Eigenvalues and Eigenvectors

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

In the formula above  $M$  denotes a matrix and  $|v_i\rangle \forall i \in \mathbb{N}$  are the eigenvectors with the corresponding eigenvalues being  $\lambda_i$ . The number of the eigenvectors and eigenvalues is found by the dimensions of the matrix. If  $M \in \mathbb{C}^{n \times n}$  then eigenvalues are  $n$  and unique

eigenvalues are at most  $n$ . To find those eigenvalues one must solve the "characteristic equation"  $\rightarrow \det(M - \lambda I) = 0$ .

### Commutator

Let  $A, B \in \mathbb{C}^{n \times n}$  then their commutator is:

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

if the commutator is equal to zero then the two operators commute and vital properties can be derived.

### Unitary Operator

Let  $U \in \mathbb{C}^{n \times n}$  then  $U$  is a unitary operator if

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

An important property of a unitary operator is that it can be rewritten as

$$U = e^{iH} \text{ where } H \text{ is a Hermitian operator.}$$

### Projective Operator

Let  $P \in \mathbb{C}^{n \times n}$  be an operator. For it to be called projective then :

$$P^2 = P \text{ and } P = P^\dagger$$

Projective operators are mutually orthogonal and their summation gives the identity matrix. Their purpose is to act on a state space and projecting it onto a subspace.

For  $P = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$  if we act on  $|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$  we get :

$$P|\psi\rangle = \begin{bmatrix} \alpha \\ 0 \end{bmatrix} = \alpha|0\rangle$$

which is the projection of  $|\psi\rangle$  in the basis  $|0\rangle$ .

## Tensor Product

Tensor product is a mathematical operation between two vectors or matrices.

Let  $A \in \mathbb{C}^{n \times m}$  and  $B \in \mathbb{C}^{i \times j}$  then their tensor product is denoted as:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1m}B \\ a_{21}B & a_{22}B & \cdots & a_{2m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}B & a_{n2}B & \cdots & a_{nm}B \end{pmatrix} \in \mathbb{C}^{ni \times jm}$$

where  $a_{kl}$  denotes the element of A in the k-th row and the l-th column and B denotes the B matrix.

Tensor products in quantum computing are really helpful, their properties let us create state spaces with more than one qubit and represent accurately in a mathematical way.

### 2.1.4 Bloch Sphere Representation

Qubits and in general particles like photons can have both a wave-like behavior and a particle-like behavior, this is called wave-particle duality. Before measuring a qubit its behavior is given by a wave function but after the measurement this behavior collapses and the particle is measured in a certain state. The measurement is done on the spin of the particle. To better represent the spin of a qubit/particle in a 3D space Bloch sphere is used.

Bloch Sphere is a geometric visualization of a quantum state as a point in a sphere. The north and south pole of the sphere represent  $|1\rangle$  and  $|0\rangle$  and the equator represents an equal superposition of the two.

Because of the states representing spins states are also denoted as:

$$|0\rangle = |\downarrow\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = |\uparrow\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Another notation for  $|0\rangle = |z_{-}\rangle$  because it is in the z-axis looking downwards in the Bloch sphere.  $|1\rangle$  is represented as  $|z_{+}\rangle$ .

To better understand the Bloch representation we rewrite the qubit's superposition as:

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

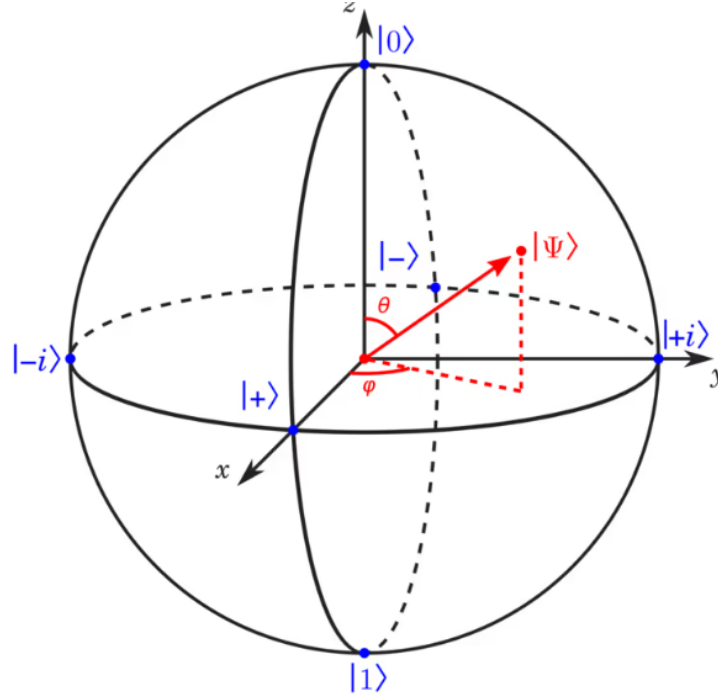


Figure 2.2: Bloch sphere. We can notice that  $|\psi\rangle$  can be measured in three different orthonormal basis.

## 2.2 Single Qubit Gates

The fundamental way of performing operations in quantum computing are qubit gates, a fact that defines their importance to quantum mechanics. For an matrix operator to be considered a gate, the matrix must be unitary.

The process is done by multiplying the unitary matrix with the qubit, and the result is some sort of rotation in the Bloch sphere. After the multiplication its important to note that the rotated qubit keeps its norm intact and that by acting on the rotated qubit with the inverse of the matrix we reverse the process.

$$\text{If } U |\psi\rangle = |\phi\rangle \text{ then } U^\dagger |\phi\rangle = |\psi\rangle$$

In quantum mechanics there are gates that are well-known and crucial for the field. The most well-known gates are the Pauli gates named after Wolfgang Pauli. These gates are represented by Hermitian matrices and work as follows:

- **Identity matrix**

Identity matrix is the first Pauli matrix denoted as  $\mathcal{I}$  and when acting on a qubit it does no rotation.

$$\mathcal{I}|0\rangle = |0\rangle \text{ and } \mathcal{I}|1\rangle = |1\rangle$$



Figure 2.3: Identity gate representation.

- **Pauli X**

Pauli-X or  $\sigma_x$  flips the state between  $|0\rangle$  and  $|1\rangle$ .

It is represented as:

$$X = \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

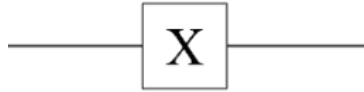


Figure 2.4: Pauli-X gate representation.

- **Pauli Y**

Pauli-Y or  $\sigma_y$  applies a bit-flip and a phase-flip to the qubit. It is represented as:

$$Y = \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

and when acting on  $|0\rangle$  or  $|1\rangle$  it works as follows:

$$Y|0\rangle = i|1\rangle \text{ and } Y|1\rangle = -i|0\rangle$$



Figure 2.5: Pauli-Y gate representation.

- **Pauli Z**

Pauli-Z or  $\sigma_z$  applies a phase-flip on  $|1\rangle$  but leaves  $|0\rangle$  intact. This is rotating a qubit around z-axis. It is represented as:

$$X = \sigma_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

and when acting on  $|0\rangle$  or  $|1\rangle$  it works as follows:

$$Z |0\rangle = |0\rangle \text{ and } Z |1\rangle = -|1\rangle$$

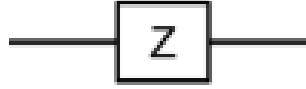


Figure 2.6: Pauli-Z gate representation.

- **Hadamard Gate**

A very important gate is Hadamard gate. Its very important feature is that it takes  $|0\rangle$  and  $|1\rangle$  and when acting on them it creates an equal superposition with a phase difference. It is represented as:

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

and when acting on the basis states it produces:

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

and

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

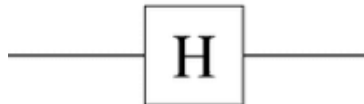


Figure 2.7: Hadamard gate representation.

- **Rotation gates**

Lastly there are gates that rotate qubits around x,y and z axis based on an angle  $\theta$ .

Every matrix  $R_x, R_y, R_z$  makes a rotation around the corresponding axis and their matrix representation is as follows:

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

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

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

## 2.3 Two Qubit Gates

Two qubit gates can be created by using the tensor product of single qubit gates. This way also multi-qubit gates can be created using the same principle. Two-qubit gates can be classified in local and non-local gates.

- **Local Operators**

Local operators are the ones that can be factorized in a tensor-product of single-qubit gates. For example having the two-qubit gate:

$$X \otimes Z |00\rangle = X |0\rangle \otimes Z |0\rangle = |0\rangle \otimes |1\rangle = |01\rangle$$

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

- **Non-local Operators**

Non-local operators are two-qubit gates that involve both qubit in the operation. This process creates entanglement between the two qubits, a principle that has crucial properties for quantum mechanics leading to computational speed-ups in quantum algorithms. It is important to note that those non-local operators cannot be factorized.

$$U_{non-local} \neq U_1 \otimes U_2$$

One of the most crucial non-local operators is the Controlled Not gate or CNOT. The CNOT acts on two qubits, the first qubit works as the control qubit and the second as the target qubit. If the control qubit is  $|0\rangle$  then the target qubit stays intact, else the target qubit changes its state.

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

CNOT gate can also be expressed as:

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

where  $P_0 = |0\rangle\langle 0|$  and  $P_1 = |1\rangle\langle 1|$

## 2.4 Quantum Entanglement

Entanglement is one of the most interesting and counter-intuitive quantum phenomena. This process makes two qubits correlated in a way that their state cannot be described independently of the other. The distance between them does not affect this connection or the speed that the information is transferred, a fact that defies laws of physics as we know them. After measuring one of the qubits the other's state is instantly determined cause of the entanglement.

The easiest and most common way of creating entanglement is using a CNOT gate on two qubits where the first is on a superposition. This quantum circuit is one of the well-known Bell states and can be describes as:

$$\begin{aligned} |\Phi^+\rangle &= \text{CNOT}(H \otimes I) |00\rangle = \text{CNOT}\left(\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |10\rangle\right) \rightarrow \\ &= \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \end{aligned}$$

The other three Bell states are denoted as follows:

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

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

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



Entanglement helps quantum algorithms tackle with problems in a more efficient way. This is achieved because entanglement helps quantum system to process and explore multiple quantum states simultaneously so crucial speed-ups can achieved in certain algorithms. Entanglement enables the teleportation of quantum information in an instant which helps creating more efficient communication systems and also improve classical encryption. Lastly, dependencies between variables can be better represented as entangled qubits leading to faster results.

## 2.5 Hamiltonian Definition

In quantum mechanics the Hamiltonian of a quantum system is a mathematical operator that represents the total energy of the system. The Hamiltonian plays a vital role in Schrodinger's equation.

- **Time Independent Schrodinger's equation**

The Hamiltonian denoted as  $\hat{H}$  acts on a quantum state  $|\psi\rangle$  and returns the energy of the levels of a system. Energy plays the role of the eigenvalue of the system.

$$\hat{H} |\psi\rangle = E |\psi\rangle$$

- **Time Dependent Schrodinger's equation**

This equation describes how the quantum system(quantum state  $|\psi\rangle$ ) evolves over time.

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

As Hamiltonian is a metric of energy it mainly consists of the kinetic and potential energy of the quantum system.

## 2.6 Teleportation

For this really important algorithm that indicates how important quantum entanglement is we will introduce Alice and Bob. The concept of the algorithm is to transfer the qubit  $|\psi\rangle$  from Alice to Bob.

To do that the circuit bellow is created and will be further analyzed.

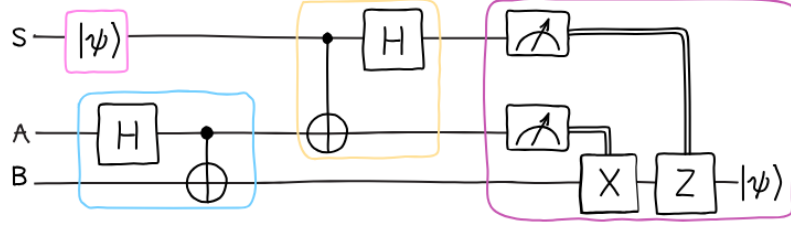


Figure 2.8: Teleportation circuit.

First of all a Bell state is created by Alice and Bob and entanglement between them is established. Alice and Bob are at  $|00\rangle$  state before entanglement.

$$\text{CNOT}(\text{H} \otimes \text{I}) |00\rangle = \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle.$$

After entanglement between Alice and Bob are established they are transferred to different parts of the world.

For the operations to be made easier we represent  $|\psi\rangle = a|0\rangle + b|1\rangle$ .  
The whole system becomes:

$$\frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle)$$

Now Alice performs a non-local operation with the qubit to be teleported, so a CNOT with Alice as the target qubit is applied.

The system becomes:

$$\frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|110\rangle + b|101\rangle)$$

Then a Hadamard gate is applied to the first qubit or the qubit to be teleported and the whole system becomes:

$$|\Psi\rangle = \frac{1}{2}(a|000\rangle + a|100\rangle + a|011\rangle + a|111\rangle + b|010\rangle - b|110\rangle + b|001\rangle - b|101\rangle)$$

If we do the right groupings we can see in an easier way the outcomes that can be produced.

$$|\Psi\rangle = \frac{1}{2}(|00\rangle (a|0\rangle + b|1\rangle) + |01\rangle (a|1\rangle + b|0\rangle) + |10\rangle (a|0\rangle - b|1\rangle) + |11\rangle (a|1\rangle - b|0\rangle))$$

So it can be easily concluded that if we measure Alice's qubits then Bob's qubit will collapse in one of those four states.

If Alice's qubits are measured at  $|00\rangle$  then the teleportation is done and Bob has the right qubit. The probability of it happening is 0.25. Nevertheless, if Alice's qubits have a different measurements then local operations can be done to Bobs qubit so he gets the right outcome. Bellow there is a table of the operations that must be made according to Alice's outcome.

Alice measures:	Bob performs:
$ 00\rangle$	$I$
$ 01\rangle$	$X$
$ 10\rangle$	$Z$
$ 11\rangle$	$ZX$

Figure 2.9: Bob operations according to Alice's outcome.

This algorithm shows us what quantum mechanics can achieve through properties like quantum entanglement. The teleportation algorithm's protocol is called LOCCSE. The starting letters denote "local operation", because after entanglement we can produce the qubit to be teleport just with local operations. "Classical Communication" is needed because after Alice's measurement the outcome of the measurement must be communicated classically so Bob gets the right result. Lastly "shared entanglement" is the most obvious part of the protocol as Alice and Bob have performed a non-local operation and they are entangled.

# Chapter 3

## Classical to Hybrid Optimization

In this chapter, we explore the intersection between mathematical models from physics and classical computing, with a particular emphasis on their relevance to quantum computing and hybrid optimization techniques. We begin by examining the Ising Model, a well-established physical model used to describe magnetic interactions in statistical mechanics. This model, originally developed to understand ferromagnetism, has since found profound applications in computational contexts due to its ability to encode binary optimization problems.

Parallel to this, we introduce the Quadratic Unconstrained Binary Optimization (QUBO) model, which is widely employed in classical combinatorial optimization. QUBO formulations express a broad range of NP-hard problems in a unified mathematical structure, making them a central focus in both classical and quantum optimization research. By analyzing the structural similarities between the Ising Model and QUBO, we demonstrate how these equivalences enable a natural mapping of classical problems onto quantum frameworks. This mapping plays a critical role in positioning QUBO as a quantum-friendly representation, especially within hybrid quantum-classical algorithms.

Following this theoretical groundwork, we review several fundamental quantum properties—such as the adiabatic theorem, and quantum tunneling—that contribute to the theoretical advantage of quantum computing over its classical counterpart. These phenomena enable quantum algorithms to explore solution spaces more efficiently and to perform certain computational tasks with reduced complexity.

To bring theory into practice, the chapter proceeds to implement and evaluate two prominent hybrid quantum algorithms: the Quantum Approximate Optimization Algorithm (QAOA) and the Variational Quantum Eigensolver (VQE). Both are variational algorithms that leverage classical optimization techniques to guide quantum state evolution. In our implementation, we adopt a Hardware-Efficient Ansatz (HEA), which is designed to be compatible with the constraints of near-term quantum hardware (also

---

known as Noisy Intermediate-Scale Quantum, or NISQ, devices).

These algorithms are tested on two benchmark combinatorial optimization problems like Subset Sum and Max-Cut, illustrating their performance, convergence behavior, and sensitivity to quantum circuit depth and parameter selection. Through these experiments, we aim to highlight both the potential and current limitations of hybrid quantum-classical approaches in solving real-world optimization problems.

Classical optimization is a wide and open research field with problems in finance, scheduling, materials science and even finding the best route in combinatorial problems. In general, optimization is about finding and minimizing/ maximizing the value of the problem's cost function, sometimes under certain constraints.

Quantum optimization is an open field of immense potential and computational power. There are two main ways to perform quantum optimization, one of them is quantum annealing and the other one is a circuit-based optimization.

The first one takes advantage of the adiabatic theorem of quantum mechanics and evolves the quantum system in a way that it stays on its "ground state". The ground state is the energy state of the system with the minimum energy, so with the right problem formulation it is also the minimization of the cost function. Nevertheless, quantum annealers are limited to the optimization problems they can solve due to their hardware construction.

The second approach, which is the one we will use on this thesis too, is circuit-based optimization. This approach uses problem specific or universal circuits to calculate the cost function of a certain problem faster than a classical computer. Then this cost function is optimized by a classical computer. This process is done iteratively until the best solution is found. Circuit-based approaches can perform universal operations, so they can tackle a large number of optimization problems. Nevertheless, the problem is that there are limitations considering the problem's size as there are upper-bounds that can achieve stable results.

In this thesis the optimization problems we will deal with are binary optimization problems. Variables take values of zero or one, and the goal of the optimization algorithm is to find the optimal bit-string of the variables that minimize the cost function. Those combinatorial binary optimization problems are considered to be NP-Hard, meaning classical computers cannot solve them if the size of the problem is sufficiently large. Classical optimization has come with solutions as approximation algorithms or heuristics that find near optimal solutions but even then scalability is not on their side, so quantum algorithms come into play.

### 3.1 QUBO Problems

Combinatorial Optimization is an active research area pursued by research communities. These problems are about making yes or no choices in a large set of decisions and then yielding an objective function value. QUBO which stand for Quadratic Unconstrained Binary Optimization is a model that fits combinatorial optimization exceptionally for a large amount of problems. Also, QUBO models can be easily translated to Ising models which plays a prominent role in physics. Some of the problems that are easily solved when modeled like QUBO are the max-cut and the subset-sum problem that we will also deal with in this thesis.

QUBO problems are combinatorial optimization problems where variables can take only binary values. The goal is to form a binary vector called bitstring to minimize a quadratic function. The bitstring that minimizes the cost function is called optimal bitstring and is denoted as  $\mathbf{x}^*$ .

The mathematical notation behind  $\mathbf{x}^*$  is:

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

where  $C(\mathbf{x})$  is the value of the cost function and  $\mathbf{Q}$  is the QUBO matrix, with  $Q \subseteq \mathcal{C}^{n \times n}$  where  $n$  is the length of the vector  $\mathbf{x}$ .

To further analyze the cost function we do some mathematical calculations.

We get:

$$C(\mathbf{x}) = \sum_{(i,j), i \neq j} q_{ij} x_i x_j + \sum_i q_{ii} x_i$$

where  $i, j \leq n$  and  $q_{ij}$  being the element of  $\mathbf{Q}$  in the  $i$ -th row and  $j$ -th column.

This holds true because the values on the diagonal are equal to their roots due to their binary definition.

$$x_i^2 = x_i$$

$$\text{if } x_i^2 = 0, \text{ then } x_i = 0$$

$$\text{if } x_i^2 = 1, \text{ then } x_i = 1.$$

It can be noticed that QUBO's complexity grows exponentially, because for  $n$  variables the solution to be examined are  $2^n$ . The standard procedure of finding a solution are exact solvers like Branch and Bound but it can be easily concluded that exact search methods will fail for significantly large problems. This need of solving bigger problems has created approximate algorithms that find close-to-optimal solutions such as CPLEX, Gurobi and SCIP. Also heuristics and metaheuristics such as Simulated annealing and Path Relinking have been created and produce better results for bigger problems.

Lastly we will explain what the term Unconstrained means. The original problem may have constraints, but those constraints are included to the cost function multiplied by a sufficiently big penalty.

Classical Constraint	Equivalent Penalty
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$
$x = y$	$P(x + y - 2xy)$

Figure 3.1: Table of some known penalties.

To better grasp the idea of penalties lets say we want to minimize a cost function  $C(x_1, x_2)$  but also there is a constraint of  $x_1 \leq x_2$ .

$$\min_{x_1, x_2} C(x_1, x_2)$$

$$\text{s.t } x_1 \leq x_2$$

is equal to:

$$\min_{x_1, x_2} C(x_1, x_2) + P(x_1 + x_1x_2)$$

where  $P$  is a sufficiently large number.

## 3.2 Ising Model

The Ising model is a mathematical model in physics that was originally created from Ernst Ising to understand phase shifts in a system of interacting spins on a lattice.

The Ising model consists of spins that interact only with their nearest neighbor on a lattice and take only discrete values of -1 or 1. To describe the energy of the Ising model we use the Hamiltonian which is given as:

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

In the Hamiltonian above  $s_i$  represents the spin in the  $i$ -th position,  $h_i$  represents an external magnetic field affecting spin  $i$ .  $J_{ij}$  is the interaction strength between neighboring spins. There are two types of interaction.

- **Ferromagnetic Interaction**

In this type of interaction  $J < 0$  and the spins tend to align with each other, meaning having the same direction.

- **Antiferromagnetic Interaction**

In this type of interaction  $J > 0$  and the spins tend to have an opposite direction.

- **No Interaction**

If  $J = 0$  then there is no interaction between the spins.

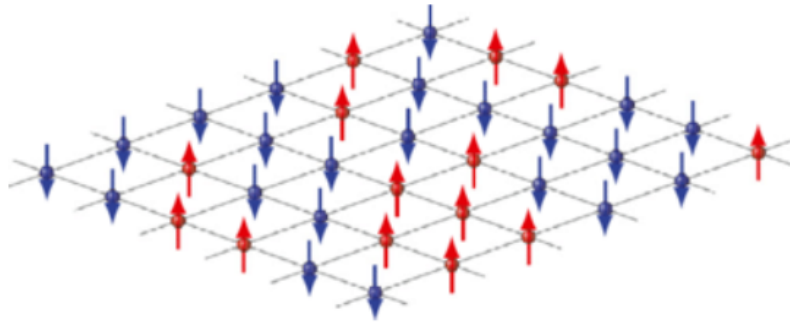


Figure 3.2: Spins on a lattice representation.

The last thing we must do is converting the classical Hamiltonian of the Ising model to the quantum one. To do that we have to substitute the variables that take values -1 or 1 with an operator that has those as eigenvalues. This operator is Pauli-Z. So:  
Classical:



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

Quantum counterpart:

$$H_{ising_{quantum}} = \sum_i \sum_j J_{ij} Z_i \otimes Z_j + \sum_i h_i Z_i$$

where  $Z_i$  is Pauli-Z gate acting on i-th qubit.

This modeling of the system has many applications in energy minimization. It represents complex systems through binary variables and can be manipulated so that an energy minimization can be achieved through specific spin configuration. We can see there is a profound analogy to optimization problems and to QUBO models through easy modifications.

### 3.3 From QUBO to Ising

With the help of the Ising model real-world problems can be translated into a framework of binary variables and interaction. This makes the translation of optimization problems to Ising really easy.

It is easily noticeable why Ising model is of interest to us as it also has direct similarity to QUBO. So, by mapping an Ising to a QUBO problem or vice-versa we can solve optimization problems by leveraging laws of physics.

For the mapping translation it is evident that the difference between QUBO and Ising problems is that the first one has variables  $x_i$  that takes values equal to either 0 or 1 but the second one has variables  $s_i$  that takes values either -1 or 1.

So to transform them we make the transformation below:

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

So by substituting this to the cost function of a QUBO we get:

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

$$C(s) = \frac{1}{4} (\sum_{i,j} Q_{ij} s_i s_j + 2 \sum_i \sum_j Q_{ij} s_i + \sum_{i,j} Q_{ij} + \sum_i Q_{ii})$$

In the equation above we have an offset that we do not mind subtracting because when we search for the optimal solution adding a scalar to the cost function will make no difference. So the Hamiltonian becomes:

$$C(s) = \frac{1}{4}(\sum_{i,j} Q_{ij} + 2\sum_i Q_{ii}) \text{ where } J_{ij} = \frac{Q_{ij}}{4} \text{ and } h_i = \frac{\sum_j Q_{ij}}{2}$$

## 3.4 Adiabaticity and Quantum Annealing

### Adiabatic Theorem

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

This theorem states that if we start from the ground state of a known Hamiltonian  $H_0(0)$  at  $t=0$  and we evolve the system slowly enough through time to end up in the problem specific Hamiltonian  $H_1(\varphi)$  when  $t = \varphi$  then the energy of the system will stay on the ground state so the optimization problem will be minimized. To summarize, the time-dependent Hamiltonian becomes :

$$H(t) = (1 - \frac{t}{\varphi})H_0 + \frac{t}{\varphi}H_1$$

where  $\varphi$  is the total time of the evolution.

The unitary evolution operator for the above Hamiltonian is:

$$U = e^{-i\frac{Ht}{\hbar}}$$

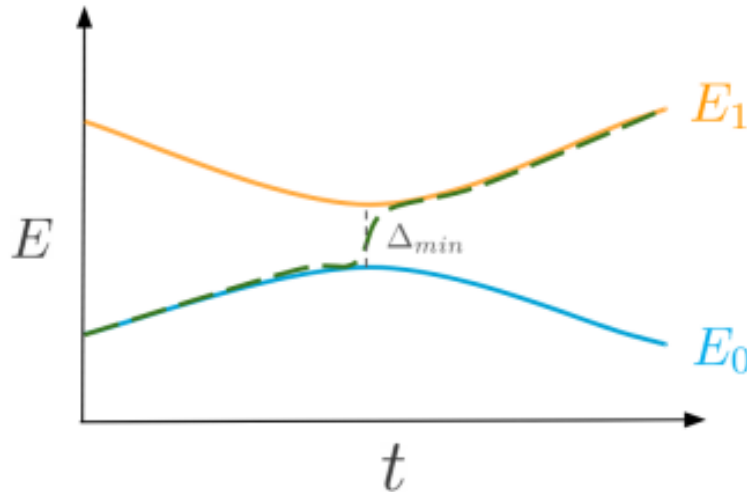


Figure 3.3: Ground state and first excited state of a Hamiltonian.

In the figure above we can see the ground state and the first excited state of a Hamiltonian. Quantum annealers must evolve the system slowly enough so that the Hamiltonian

does not jump from the ground state to the excited state. The total time is a function inversely proportional to the minimum energy gap, denoted as  $\Delta_{min}$ . Specifically,  $T = \mathcal{O}(\frac{1}{\Delta_{min}^2})$

### Quantum tunneling

Another principle of quantum annealers that has been proven to be extremely beneficial for quantum optimization is quantum tunneling. According to this principle, a system of particles can traverse through energy barriers, a fact that seems impossible in classical physics. This principle of just phasing through energy barriers without the need of a thermal jump makes one of the greatest obstacles of optimization disappear. The system cannot be stuck in local minima with quantum tunneling so finding a global minimum of the Hamiltonian is easier compared to classical optimization algorithms.

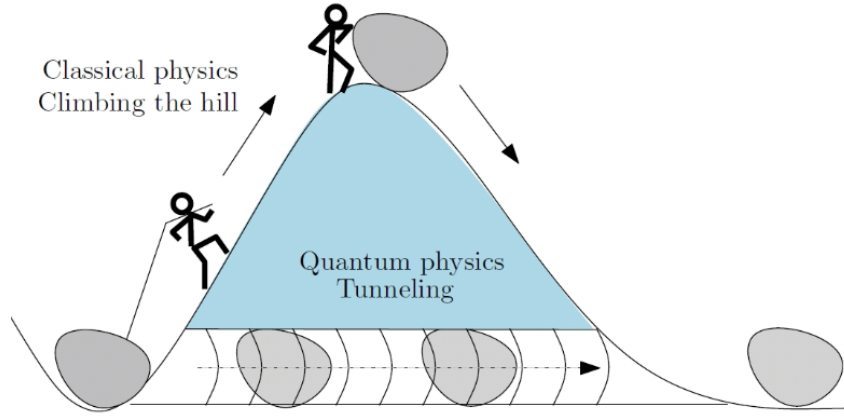


Figure 3.4: Quantum tunneling, a property of phasing through a thermal jump.

Quantum annealers can easily and naturally find the optimal solution in optimization problems but with the current hardware they have enough downsides. Annealers are really vulnerable to noise and thus their scalability is a problem. Also the connectivity is an issue for the qubits, it is really hard to represent fully connected graphs even for small scales.

## 3.5 Hybrid Quantum-Classical Optimization

Quantum annealers faced hardware limitations, such as limited connectivity and scalability, prompting the exploration of alternative quantum computing paradigms. Variational Quantum Algorithms (VQAs) emerged as a prominent approach—not solely to address

hardware constraints—but primarily because they rely on parameterized quantum circuits, which are optimized through classical feedback loops. This hybrid structure enables the solution of complex optimization problems on noisy intermediate-scale quantum (NISQ) devices.

The hybrid nature of the algorithms is because of their need to use both quantum and classical computational power. Depending on the problem the quantum part of the algorithm (e.g circuit, ansatz) can vary. The big picture is that a parameterized quantum circuit is used to compute a cost function, then a classical optimizer is optimizing the parameters to produce the optimal solution.

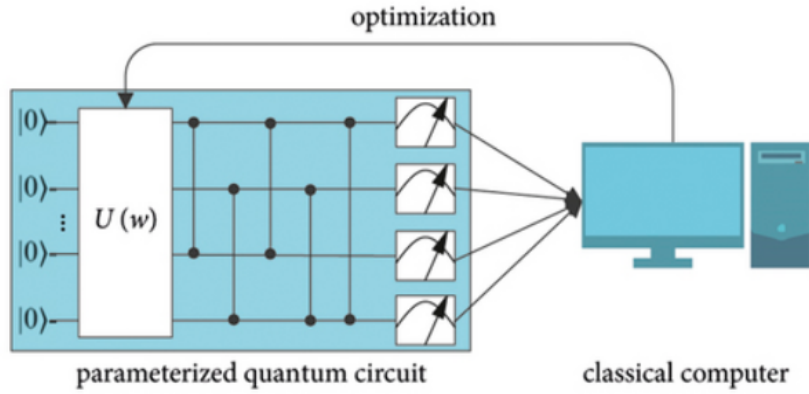


Figure 3.5: Big picture of a VQA.

First step of the algorithm is to define a cost function  $C$ , which is the problem's function to be minimized. Then a quantum circuit called ansatz acts on a starting state  $|\psi_0\rangle$  and produces an outcome based on the parameter  $\theta$ .

If we denote the ansatz as an oracle, which is a parameterized unitary operator  $\hat{U}(\theta)$  then the outcome is:

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

After that, the cost function is computed as:

$$C(\theta) = \langle \psi(\theta) | H | \psi(\theta) \rangle$$

This cost function is fed to the classical optimizer, which tries to minimize its value through the change of the angle  $\theta$ . After a certain amount of iterations when the termination criteria are met we are left with the optimal angle  $\theta^*$  which minimizes the cost function and returns the desired state  $|\psi^*\rangle$ .

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

Lastly we measure state  $|\psi^*\rangle$  and if its the optimal state we get the optimal bitstring  $\mathbf{x}^*$  with a big probability. State can also be expressed as:

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

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

So to take the right solution of this superposition  $|a^*|^2$  must be close to 1.

The classical optimizers used to optimize the angles for VQAs can be both gradient-free and gradient-based optimizers. There are several optimizers that have been specifically fabricated to optimize VQAs. Nevertheless, NP-Hardness of the problems are not eliminated, so optimizing the angle parameters is still considered to be an NP-Hard problem for classical optimizers.

## 3.6 Algorithms

Below two commonly used hybrid quantum-classical approximate algorithms are presented and then used to solve two well-known problems, the Max-Cut and Subset-Sum problem.

### 3.6.1 Quantum Approximate Optimization Algorithm

QAOA or Quantum Approximate Optimization Algorithm is a hybrid algorithm designed to solve combinatorial optimization problems. QAOA works like a quantum annealer to solve QUBO problems taking advantage of the Trotterization theorem [8]. The difference from VQAs is that there is a set of parameters  $\gamma, \beta$  and the ansatz is fabricated according to the problem.

The trotterization theorem allows us to break down the exponential of the sum of non-communicating Hamiltonians to grasp how QAOA mimics quantum annealers.

Quantum annealing Unitary operator:

$$|\psi(t)\rangle = e^{it(A(t)H_0+B(t)H_1)} |\psi_0\rangle$$

Using the trotterization theorem we get:

$$|\psi(t)\rangle = e^{itA(t)H_0} e^{itB(t)H_1} |\psi_0\rangle$$

Now it is visible that if we replace  $tA(t)$  and  $tB(t)$  with the parameters  $\gamma$  and  $\beta$  we get the Hamiltonian of the QAOA:

$$|\psi(\gamma, \beta)\rangle = \prod_{p=1}^P e^{-i\beta_p H_0} e^{-i\gamma_p H_1} |\psi_0\rangle$$

where  $p$  is the the number of ansatz's repetitions which is called depth.

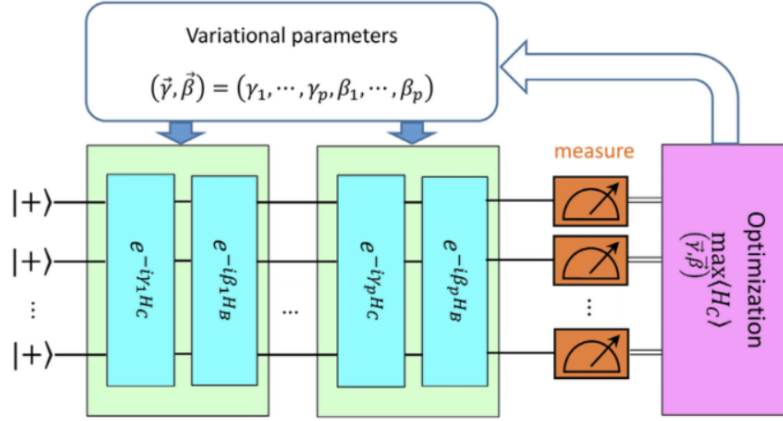


Figure 3.6: Multi-layer QAOA with depth= $p$

In the figure above Cost function is given by:

$$C(|\psi\rangle) = \langle\psi| H_C |\psi\rangle$$

and the problem is said to be maximization which is equal to the minimization of cost function multiplied with -1.

The two Hamiltonians applied above have certain names and uses.

- **Cost Hamiltonian**

The cost Hamiltonian is a problem specific Hamiltonian designed according to the optimization problem. So, for different problems cost Hamiltonian and the corresponding circuit called ansatz may vary.

Having  $H_{ising}$  as our cost function we can implement the ansatz.

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

which using operators is:

$$H_{ising_{quantum}} = \sum_i \sum_j J_{ij} Z_i \otimes Z_j + \sum_i h_i Z_i$$

So for interactions between pairs of qubits an  $R_{ZZ}(\theta)$  gate is used to represent an entangling gate that acts on two qubit to ensure correlation and it is defined as:

$$R_{ZZ}(\theta) = e^{-i\theta Z_i \otimes Z_j}$$

$$\text{where } \theta = 2\gamma_p J_{ij}$$

For the diagonal elements of the Ising/QUBO matrix or the  $\sum_i h_i Z_i$  we have single-qubit rotation around z-axis with just  $R_z$  gates.

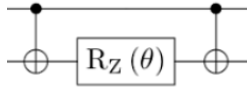


Figure 3.7:  $R_{ZZ}$  gate.

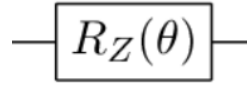


Figure 3.8: Single rotation  $R_Z$  gate.

#### • Mixing Hamiltonian

The mixing Hamiltonian is not problem specific so it a certain configuration of quantum gates for every problem. This configuration ensures exploration through the solution space. It consists of single-qubit gates that act to every qubit and are rotations around x-axis.

$$H_M = \sum_i X_i$$

So the unitary operation applied is:

$$U_M = e^{-i\beta \sum_i X_i}$$

which is a rotation around x-axis for all qubits simultaneously.

The existence of the mixing Hamiltonian ensures that QAOA will balance exploration and exploitation to solve the combinatorial problem.

### 3.6.2 Hardware Efficient Ansatz

Hardware Efficient ansatz was created to give solution in cases where Quantum Approximate Optimization Algorithms are lacking. QAOA have problem specific ansatzes that in a lot of case are not compatible with the quantum hardware. Also, in QAOA instances deep circuits are needed for exact solutions to be produced, a fact that makes the circuits susceptible to noise and decoherence as noise is analogous to circuit deepness.

So Hardware Efficient ansatz is a more generic type of circuit that is not problem specific and it is designed to be compatible with current quantum hardware. To capture correlation and single-qubit operations, Hardware efficient ansatz is tailored with alternating single-qubit gates and gates that create entanglement.

Hardware efficient ansatz works exactly like QAOA. Cost function is calculated and then classical optimization is giving a better solution for the circuit's parameter  $\theta$ . Typically, HEA consist of single rotation gates around y-axis and a CNOT or CZ entangling gates alternating n times where n is the amount of layers.

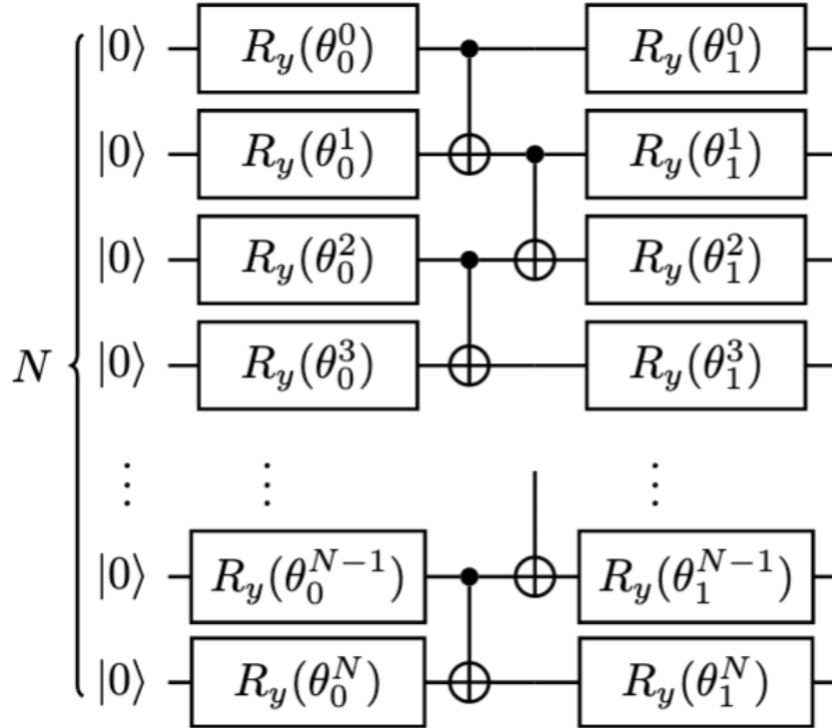


Figure 3.9: Hardware Efficient ansatz first full layer and second layer's single rotation gates with  $N$  qubits.

For complex scenarios more layers can be added to increase expressibility of the al-



gorithm and produce better solution. There is an optimal point in choosing the number or layers, beyond this point the deepness of the circuit becomes too large. This leads to the unitary transformations to become highly random and approach a uniform distribution. Gradient of the cost function this way becomes exponentially small as the depth increases, so it is difficult for the algorithm to optimize its solution. This phenomenon is called Barren plateaus .

## 3.7 Application of Quantum Algorithms in Specific Problems

### 3.7.1 Max-Cut and Subset Sum

- **Max-Cut Problem**

One of the most common optimization problems that can be easily expressed as QUBO are Max-Cut problems. Max-Cut is the division of a graph into two sub-graphs. The essence of the problem is trying maximizing the number of edges that are involved in the cut.

To formulate the problem we give values to the nodes of the graph. If  $i$ -th node belongs to the sub-graph A then  $x_i = 0$ , else  $x_i = 1$ .

So, if  $x_i$  and  $x_j$  belongs to different groups then the edge between them must be add to the cut.

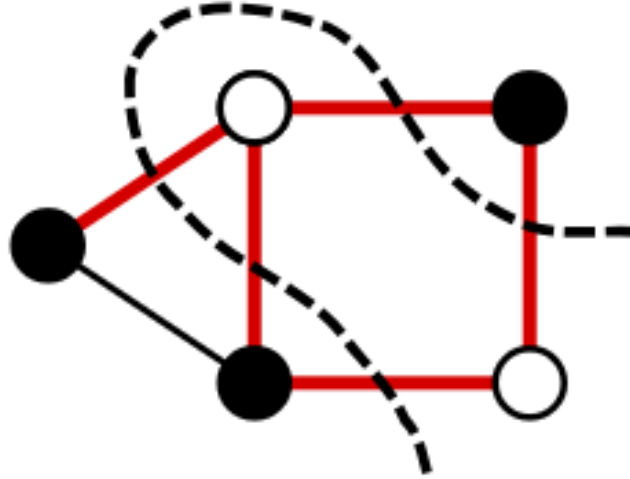


Figure 3.10: A random graph and its Max-Cut.

Formulation of the cost function:

The function:  $f(\mathbf{x}) = x_i(1 - x_j) + x_j(1 - x_i)$   
 takes values of 1 if the nodes belong to different sides of the cut.

So:

$$\begin{aligned} \text{maximize}_x C(\mathbf{x}) &= \sum_{\langle i|j \rangle} x_i + x_j - 2x_i x_j \\ \text{s.t } x_i &\in \{0, 1\} \forall i. \end{aligned}$$

It is clear what with this formulation if  $x_i$  and  $x_j$  belong to different sub-graphs then their edge is added to the cut and the cost function is increased by one. So if every edge has the same weight then if we maximize the cost function we maximize the edges that are in the cut.

We can also notice that this problem can be expressed directly by a QUBO matrix. The quadratic terms are the non-diagonal elements and the non-quadratic terms can be squared and represented by the diagonal elements of the matrix.

- **Subset Sum Problem**

In the Subset Sum problem we have a vector  $\mathbf{a}$  and a scalar value  $S$ . The goal of the algorithm is to find a subset of the elements of  $\mathbf{a}$  that are as close to  $S$  as possible.

To accomplish that we can multiply  $\mathbf{a}$  with a binary vector  $\mathbf{x}$  that takes values 0,1. After that we subtract  $S$  and the value closest to zero is the optimal value, so the

bitstring  $\mathbf{x}$  producing that value is the optimal bitstring.

So cost function is :

$$C(\mathbf{x}) = \mathbf{a}^T \mathbf{x} - S$$

To make this cost function optimizable we square it so it has an optimal point.

So the final problem becomes:

$$\text{minimize}_{\mathbf{x}} C(\mathbf{x}) = (\mathbf{a}^T \mathbf{x} - S)^2$$

$$\text{s.t } x_i \in \{0, 1\} \forall i.$$

Below we shall see a Max-Cut and a Subset sum example to better grasp the concept of VQAs for those problems but also for combinatorial optimization problems in general.

#### 3.7.2 Max-Cut using QAOA and HE Ansatz

In this section we use the two different algorithms to work on a the Max-Cut problem. We delve deeper into the algorithms solution by manipulating circuit's deepness to find better solution in non-exploding times.

Lastly we try to tailor a problem with a solution that can be easily validated, so we can check algorithms accuracy and convergence.

So the graph created is the following:

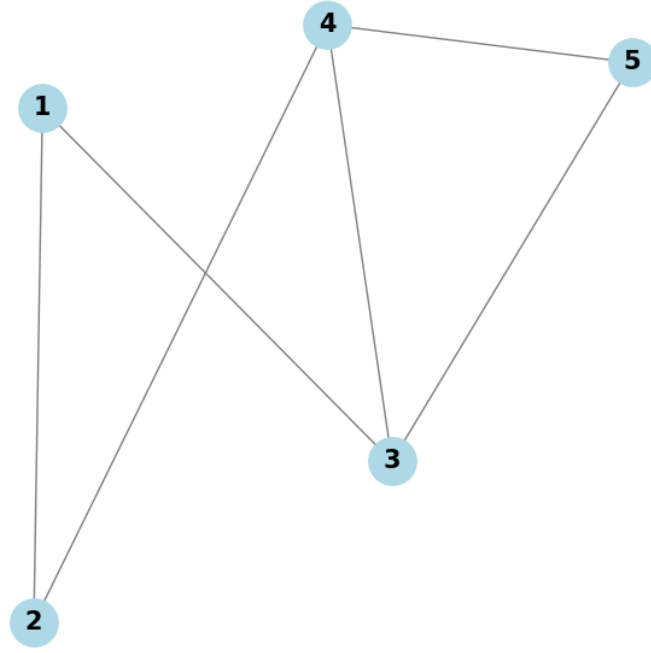


Figure 3.11: A 5-node graph with a simple and visible solution.

In this graph we see that the visible solution is the one that 1,4 or 2,3 in the same sub-graph and leaves the rest together.

#### QAOA testing

- **One layer.**

With one layer in the problem specific quantum circuit we get:

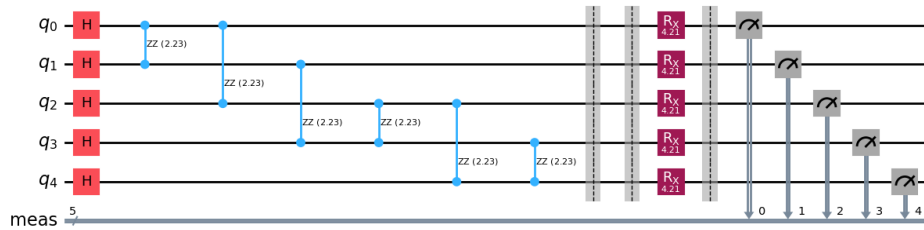


Figure 3.12: Ansatz for QAOA and Max-Cut with 5 nodes.

It can be notice that entangling gates have been used only where edges exist. For

### 3.7. APPLICATION OF QUANTUM ALGORITHMS IN SPECIFIC PROBLEMS

example, nodes 1 and 4 do not have an edge connecting them so  $q_0$  and  $q_3$  representing those nodes do not have an entangling gate.

And the corresponding results are:

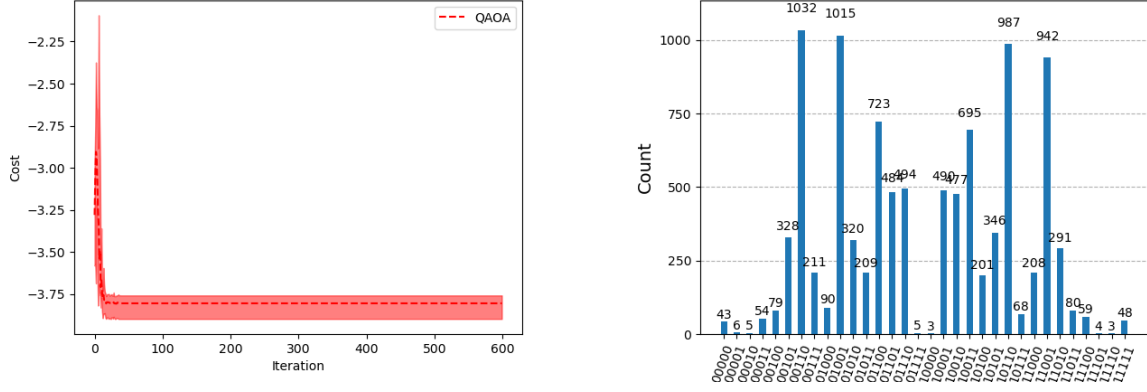


Figure 3.13: QAOA converges fast after 40 iterations in the optimal point. Using such a shallow circuit with only 1-Layer and a small amount of total gates makes our algorithm find the optimal solution with some probability but not fully converging to the optimal solution with close to total accuracy.

- **Five layers.**

For five layers, the one-layer circuit is repeated five times and the results are depicted below:

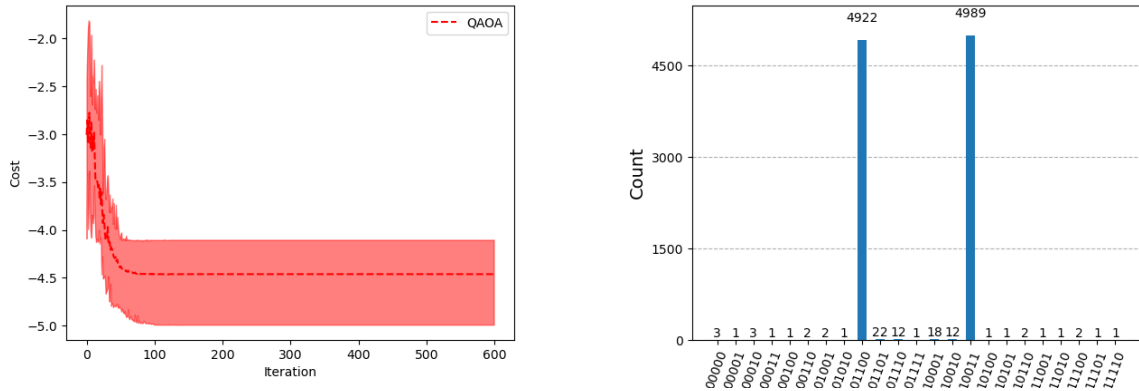


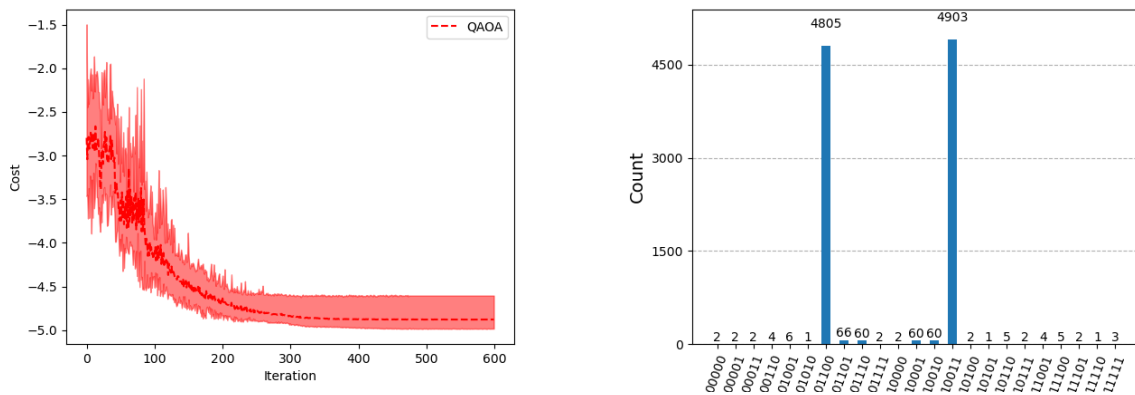
Figure 3.14: Probabilities of the measurements are distributed on the optimal solutions in a better way, and that is why cost function has a value close to 5 when Max-Cut is 5. 10 experiments are conducted and we can see that the best one is really close to optimal.

### 3.7. APPLICATION OF QUANTUM ALGORITHMS IN SPECIFIC PROBLEMS

Theoretically, with a infinite layer QAOA circuit we can reach the optimal point with certainty when the problem can be depicted by a partially or fully connected graph.

- **Twenty layers.**

If we create a 20-layer circuit we can imagine that there is a big computational scaling and that run-time of the algorithm becomes non-efficient for a small problem like that. Nevertheless, the accuracy of the solution becomes really good making non-optimal solutions almost disappear.



To conclude QAOA produces better solutions when the number of layers becomes larger until the Barren plateaus are reached. Run-times explode for even small amounts of qubits when large numbers of layers are used. So, for this problem we can see that both algorithms reach a "ground truth" solution but the 5-layered one is sufficient not to say better.

#### Hardware Efficient ansatz.

- **One layer.**

Hardware Efficient Ansatz does not encourage exploration. So we can see that after finding an optimal solution it stays on that solution without exploring more. This can make the solution stack in local minima easier.

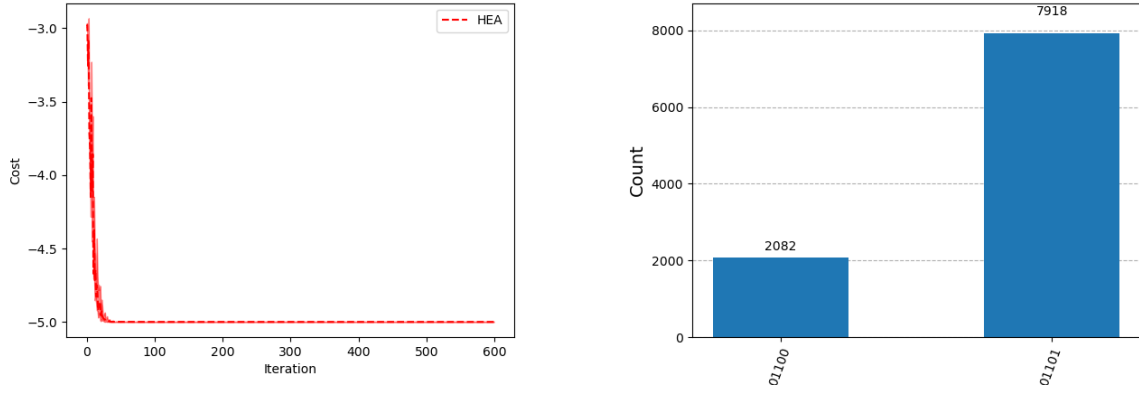


Figure 3.15: It is noticeable that the cost function converges fast. The optimal solutions are found with just one layer and all other solutions have zero probability of being measured in a noiseless simulation.

We do not need to do any more experiments with Hardware Efficient Ansatz, the solution provided by one-layer is sufficient.

#### 3.7.3 Subset Sum using QAOA and HE Ansatz

For the Subset-Sum problem we will also tailor a problem so the best solution is visible.

$$a = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \text{ and scalar } S = 8, \text{ So } |\mathbf{x}^*\rangle = |111111\rangle$$

#### QAOA testing

- One Layer QAOA





### 3.7. APPLICATION OF QUANTUM ALGORITHMS IN SPECIFIC PROBLEMS

---

- For a simple problem that the optimal solution can be found with one layer, adding complexity with more layers creates more noise so the solution can become actually worse.
- When adding more layers, the optimization parameters are more so the classical optimizer may find it more difficult to explore the parameter space and provide the optimal set of parameters.

#### Hardware Efficient Ansatz.

In VQA using HEA the algorithm does not encourage exploration. If the a minimum is found the algorithm is prone to stack in this minimum. Subset Sum is a rather easy problem with an easy to navigate landscape. This makes HEA working extremely fast providing the best solution almost instantly in scenarios like that.

#### One Layer HEA.

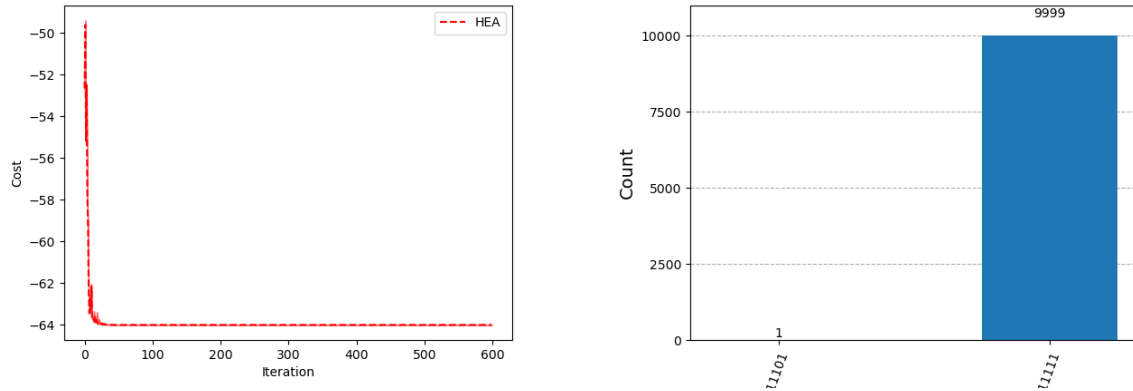


Figure 3.18: Distribution indicates that almost always the optimal solution is found even with one layer and that the cost function almost instantly converges to the optimal value of the cost function.

## Chapter 4

# Quantum Optimization in Scheduling Problems

In this chapter, we present the core contribution of this thesis, which focuses on applying quantum algorithms to complex scheduling problems. We begin by introducing the general class of scheduling problems, which play a central role in a wide range of industrial and operational contexts. These problems involve the allocation of limited resources over time to perform a set of tasks, often under strict constraints and with the objective of optimizing performance metrics such as total completion time, lateness, or resource utilization.

Among the various types of scheduling problems, we concentrate on a particularly challenging subclass known as the Job-Shop Scheduling Problem (JSSP). JSSP encompasses a broad family of optimization problems in which multiple jobs, each composed of a sequence of operations, must be scheduled on a set of machines. Each operation is constrained to a specific machine and must follow a predefined order. Due to the combinatorial explosion of possible schedules, JSSP is widely recognized as an NP-hard problem, and as such, remains computationally intractable for large instances when approached using classical methods alone.

To investigate the potential of quantum computing in this domain, we construct and study a manufacturing scheduling problem that falls within the JSSP framework. This problem models a realistic industrial scenario in which various tasks must be scheduled efficiently across multiple machines, reflecting both precedence constraints and machine availability.

The remainder of the chapter is dedicated to exploring how quantum algorithms—specifically, those designed for optimization such as QAOA—can be employed to address this manufacturing scheduling problem. We translate the problem into a form suitable for quantum computation, typically through a QUBO (Quadratic Unconstrained Binary Optimization)

formulation, which facilitates implementation on current quantum hardware or simulators.

Through this investigation, we aim to assess both the computational complexity and the solvability of various problem instances. By evaluating the performance of quantum algorithms on different scales and configurations, we explore the feasibility of extending quantum approaches to real-world scheduling challenges and large-scale data environments. This serves as a critical step toward understanding the limitations and capabilities of near-term quantum devices in solving practically relevant industrial optimization problems.

## 4.1 Scheduling Problems

Scheduling is used in a broad number of instances to allocate resources with the ultimate goal of minimizing the total time or resources a certain procedure requires. The instances scheduling problems can be used, is in hospitals to allocate nurses or doctors to shifts, in computers, to allocate CPU resources to different tasks, or even to how to allocate runways in an airport to deal with takeoffs and landings.

Generally a scheduling problem can be described as follows:

- **Jobs:** There is a set of Jobs that must be completed.
- **Tasks:** A Job consists of certain Tasks that must be done in certain order.
- **Resources:** To finish the set of Tasks and in that way complete a Job requirement there is a finite set of resources that can be used.

A scheduling's problem goal is to allocate tasks to resources without violating the constraints introduced in an optimal way. The optimality is decided by some measure performance, typically the total time needed.

In general, there is a big amount of categories that a scheduling problem can fall into. This is decided by the characteristics of the scheduling problem, for example there can be deterministic scheduling or uncertain scheduling. Our problem falls under the umbrella of JSSP which are multiple machines scheduling problems. The fact that multiple machines exist and a Job does not undergo the same sequence of machines to be produced provides flexibility but also complexity(bigger amount of variables).

This manufacturing problem is a special case of scheduling problem. Special and really specific constraints are introduced as task ordering, inactivity windows, need-by-date and even campaigning. Those constraints are introduced to make the problem fit in

real world applications where certain products must be created on specific deadlines or where manufacturing lines do not work without periods of inactivity.

## 4.2 The Manufacturing Production Scheduling Problem

### 4.2.1 Problem Introduction

The big picture of the manufacturing problem is how tasks that belong to jobs can be assigned to machines, without violating any constraints so a certain number of jobs is completed in the minimum possible time. This problem is considered to JSSP which is a NP-Hard problem. Nevertheless, the mathematical formulation and how it can be converted into a QUBO are of significant importance. Having this QUBO formulation and even tackling toy problems can set the grounds for bigger real life problems when the hardware is ready.

Undoubtedly, the manufacturing problem is a struggle for every production factory that wants to maximize its efficiency and its reward in a certain amount of time, utilizing every machine at its maximum and respecting inactivity windows like vacations and weekends for its employees.

### 4.2.2 Introducing the Disjunctive QUBO Model

**Problem definition disjunctive model is used.**

JSS problem is defined as a problem of  $N$  Jobs and a finite set of machines  $M$ . For convenience we refer to the problem as  $n \times m$  problem. For each job a vector that represents the order of  $J$  job through the machines is given. This vector  $[\sigma_1^J, \dots, \sigma_m^J]$  indicates the operations (tasks) of job  $J$ . For each set of job  $i$  and machine  $j$ , we are given an non-negative integer  $p_{i,j}$  which represents the time needed for the  $i$ -th job to finish at  $j$ -th machine. There is also a constraint that each machine can process at most one job at a time and once a job starts on a machine it must finish without interruption. The final objective is to find a schedule that minimizes the makespan of a set of Jobs in a finite set of Machines.

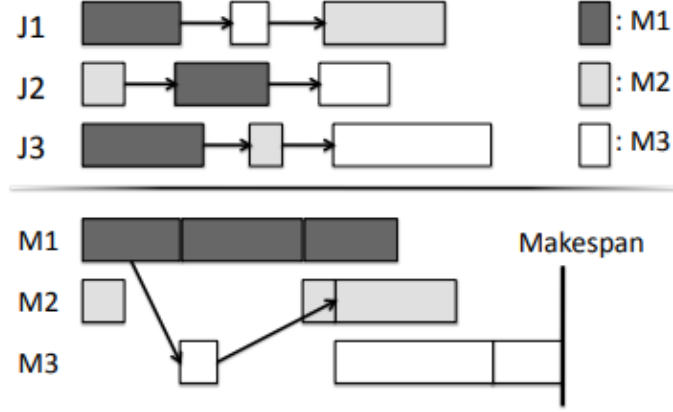


Figure 4.1: The first figure shows the ordering of the operations of each Job and in which machine each operation can be assigned. In the second figure we see a potential schedule that satisfy the constraints and has a certain makespan.

### Model Formulation

The following Mixed Integer problem is based on Manne. First of all we present the decision variables:

**Starting time:**

$$x_{ij} = c, \quad c \in \mathbb{N}$$

for j-th job in i-th machine

**Precedence:**

$$z_{ijk} = \begin{cases} 1, & \text{if job j precedes job k on machine i} \\ 0, & \text{otherwise} \end{cases}$$

So the complete disjunctive model:

- **min**  $C_{max}$  Optimization is done over the total makespan, so the objective is to minimize the starting time + operating time of the last job.
- **Constraints**

$$- x_{ij} \geq 0$$

which states that the starting timestep of each job must be bigger than zero.

$$- x_{\sigma_h^j, j} \geq x_{\sigma_{h-1}^j, j} + p_{\sigma_{h-1}^j, j}$$

which is called the precedence constraints. And states that if two operations are from the same job, if *operation<sub>h</sub>* precedes *operation<sub>h-1</sub>* then, starting time + operating time of *operation<sub>h-1</sub>* must be smaller than starting time of *operation<sub>h</sub>*. This way operations are done in the right order.

$$- x_{ij} \geq x_{ik} + p_{ik} - Vz_{ijk}$$

which is constraint that ensures that if job k precedes job i, and they are processed in the same machine no collision will happen. The last term indicates that if j precedes k then z variable which is a switch subtracts V, a sufficiently large value, so the inequality is always fulfilled.

$$- x_{ik} \geq x_{ij} + p_{ij} - V(1 - z_{ijk})$$

this constraint is "symmetric" to the one above ensuring the other case.

$$- C_{max} \geq x_{\sigma_m^j, j} + p_{\sigma_m^j, j}$$

which ensures that the total makespan is bigger than the last jobs starting time + operation time.

$$- z_{ijk} \in \{0, 1\}$$

which states that z can only take binary flag values.

To conclude we can understand that there is a direct correlation of the disjunctive model and our problem. So we can create our formulation on the disjunctive model formulation. The difference between MIP and QUBO is that the first model has integers as inputs and the later one has only binary variables as inputs. Nevertheless, we can create an encoding to make integer  $\rightarrow$  binary.

### 4.2.3 Mathematical Formulation

To start with the problem we introduce a variable that represents the starting time of every task of a certain job. This variable will be denoted as  $S_i^J$  with i representing the task and J representing the Jobs.

## 4.2. THE MANUFACTURING PRODUCTION SCHEDULING PROBLEM

---

After dividing the total time into timesteps we observe that if for example  $i$ -th task starts at timestep 3 then:

$$S_i^J = 3$$

Now, we make this integer representation binary, because the problem we solve is a binary optimization problem so:

$$S_i^J = 11$$

Finally a matrix is created that represents the starting time of each task and that it should be optimized. This matrix has a size of:

$$\#tasks \times \log(\#timesteps) .$$

Then the variable  $X_{i,m}$  is introduced.

$$X_{i,j} = \begin{cases} 1, & \text{if task } i \text{ operates on machine } j \\ 0, & \text{otherwise} \end{cases}$$

So a matrix  $X$  is created that has a size of:

$$\#tasks \times \#machines.$$

Then a different set of variables is needed called sequencing variables.

$$Y_{i,j} = \begin{cases} 1, & \text{if task } i \text{ precedes task } j \\ 0, & \text{otherwise} \end{cases}$$

So a matrix  $Y$  is created that has a size of:

$$\#tasks \times \#tasks.$$

Lastly a set of variables is used to handle inactivity windows.

$$Z_{i,l} = \begin{cases} 1, & \text{if task } i \text{ starts before inactivity window } l \\ 0, & \text{if task } i \text{ starts after inactivity window } l \end{cases}$$

So a matrix  $Y$  is created that has a size of:

## 4.2. THE MANUFACTURING PRODUCTION SCHEDULING PROBLEM

$$\#tasks \times \#inactivitywindows.$$

To create the input variables that will have a one-to-one correlations with the qubits we use as input we must flatten, into a vector, all those matrices and then create a single vector with all those variables. In essence, those sizes above represent a worst-case scenario of scalability and do not correspond in real scaling.

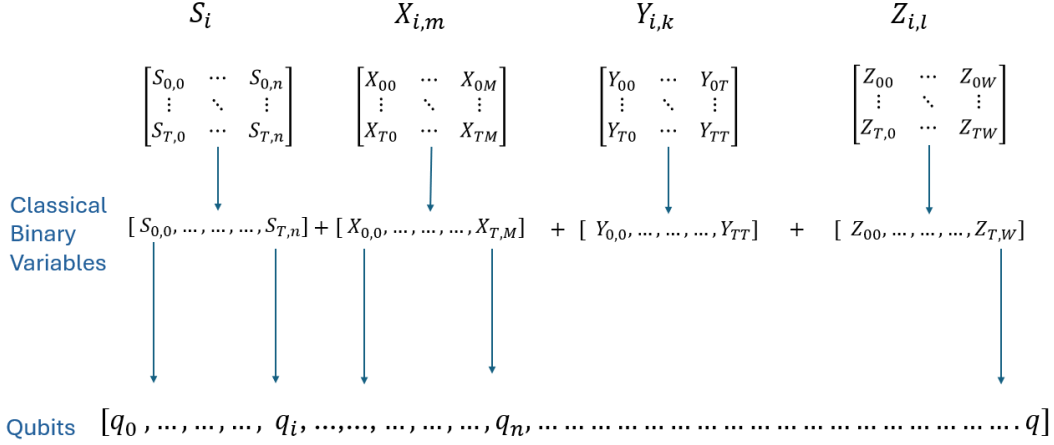


Figure 4.2: From the matrices created to represent the scheduling problem to qubits used as input for the quantum hardware.

### Parameters

To have a good representation of the problem we introduce some parameters that will help us create the constraints of the problem in an accurate way.

- $p_i^J$ : Processing time of task  $i$  of job  $J$ .
- $b_i^J$ : Brand of task  $i$  job  $J$ . If two tasks  $i,j$  have a different brand and they are executed on the same machine one after the other then a cleaning time of the machine is introduced.
- $b_i^{J*}$ : Brand of task  $i$  job  $J$ . If two tasks  $i,j$  have same brand but different strength key(sub-bran measurement) and they are executed on the same machine one after the other then a cleaning time of the machine is introduced.
- $\delta_{jk}$ : which is a flag if task  $j$  and  $k$  belong to different brands.
- $\delta_{jk}^*$ : which is a flag if task  $j$  and  $k$  belong to same brands but have a different strength key (sub-brand measurement).
- $ur_m$ : which indicates the capacity the machine works. If  $ur_m = 0.9$  then process time takes a total of  $p_i^J(2 - ur_m)$ .



## 4.2. THE MANUFACTURING PRODUCTION SCHEDULING PROBLEM

---

- $D^J$ : which indicates a date that job J must be completed.
- $M$ : is a sufficiently large value to activate or deactivate inequality constraints.
- $[m_{m,s}, m_{m,e}]$ : represents the starting time and ending time of a maintenance window for machine m.
- $t_c$ : time required for cleaning if two tasks belong to different brand.
- $t_c^*$ : time required for cleaning if two tasks belong to different strength key.

The binary optimization problem is described as follows:

**Minimize**     $C_{max}$

$$C_{max} \geq S_i^J + p_i^J(2 - ur_m) - M(1 - X_{i,m}) \quad \forall i. \quad (4.1)$$

$$S_k^J \geq S_i^J + p_i^J(2 - ur_m) - M(1 - X_{i,m}) \quad \forall i, k \in j, \forall j \in J \quad (4.2)$$

$$\sum_m X_{i,m} = 1 \quad \forall i \quad (4.3)$$

$$D^j \geq S_i^j + p_i^j(2 - ur_m) - M(1 - X_{i,m}) \quad \forall i, j. \quad (4.4)$$

$$S_k^J \geq S_i^J + p_i^J(2 - ur_m) + \delta_{i,k}t_c + (1 - \delta_{i,k})\delta_{i,k}^*t_c^* - M(1 - Y_{i,k}) - M(2 - X_{i,m} - X_{k,m}) \quad \forall i \neq k. \quad (4.5)$$

$$S_i^J \geq S_k^J + p_k^J(2 - ur_m) + \delta_{i,k}t_c + (1 - \delta_{i,k})\delta_{i,k}^*t_c^* - M(1 - Y_{i,k}) - M(2 - X_{i,m} - X_{k,m}) \quad \forall i \neq k. \quad (4.6)$$

$$S_i^J + p_i^J(2 - ur_m) \leq m_{m,s} + M(1 - X_{i,m}) + M(1 - Z_{i,l}) \quad \forall i, l \quad (4.7)$$

$$S_i^J \geq m_{m,e} - M(1 - X_{i,m}) - MZ_{i,l} \quad \forall i, l \quad (4.8)$$

In the following lines we will try to delve deeper into the inequality constraints:

- **Makespan Constraint (4.1)**

This constraint ensures that every task is finished before the makespan time. This flag for the machine indicates that this inequality holds only for the machine m, so we can use the proper utilization rate.

- **Task Order Constraint (4.2)**

This constraint is about tasks precedence, so they follow the right order. Last term plays the same role as before.

- **Single Machine Assignment Constraint** (4.3)

Constraint is about every task running at exactly one machine.

- **Need by Date Constraint** (4.4)

Above constraint ensures that if there is a need for a certain Job to be completed by a certain timestep it will be fulfilled.

- **No Overlapping/Campaigning Constraint** (4.5) (4.6)

This ensures that if task i starts before task k it will be finished first before k's initialization and vice-versa. The terms with the deltas apply cleaning times if needed, term with Y ensures precedence and last term ensures that tasks should run at the same machine.

- **Maintenance/Holiday Constraint** (4.7) (4.8)

Last two constraints ensure that a task will end before an inactivity window and will start after the inactivity window's ending time. The terms using X variables ensure the machine is right and the last terms check if the task should start after or before an inactivity window.

### Augmented Lagrangian Method

To handle all this inequalities the intuition behind it is to make them equalities. To do that a slack variable is added to every inequality and the constraint is  $s_i \geq 0$ .

So, if we have a constraint:

$$\text{s.t } \mathbf{a}_i^T \mathbf{x} - b_i \leq 0$$

then we convert it to equality:

$$\mathbf{a}_i^T \mathbf{x} - b_i + s_i = 0 \text{ with } s_i \geq 0$$

This way of handling inequalities uses slack variables as additional input variables, a fact that makes slack variables really inefficient for quantum computing because the qubits we can use are up to 20.

To handle this problem another method is introduced called Augmented Lagrangian method. To handle the constraint we try to minimize the cost function plus a penalized second-order Taylor approximation of the constraints. If a constraint is violated then the penalty of this particular constraint is increased and the optimization is done again.

**Input:** Initial  $\mu$ , initial  $\lambda$

**1. Repeat**

**2.**  $x \leftarrow \arg \min_x f(x) + \sum_i (\lambda_i c_i(x) + \frac{\mu}{2} \|c_i(x)\|^2)$

**3. for**  $i$  **in**  $1, \dots, n$  **do**

**4.** **if**  $c_i(x) > 0$  **then** **If Constraint is violated**

**5.**  $\lambda_i \leftarrow \lambda_i + \mu \cdot c_i(x)$  **Increase the penalty**

**6.** **end if**

**7. end for**

**8.**

**9.**  $\mu \leftarrow \rho \mu$

**10. Until all constraints are satisfied**

$c(x) := a^T x - b$

Solve using a quantum algorithm

Figure 4.3: Snippet of pseudo-code of Augmented Lagrangian method.

Using this method is like using the Newton method for the constraints to reach the feasible set in less time using less variables.

#### 4.2.4 Algorithmic Approach

To construct the problem we create the QUBO function for every term of our unconstrained cost function and then add them all together in a general QUBO model this model is then optimized, and the optimal bitstring produced is decoded and translated into a gantt graph of the initial problem.

To visualize the problem better we use graphs of the toy example:

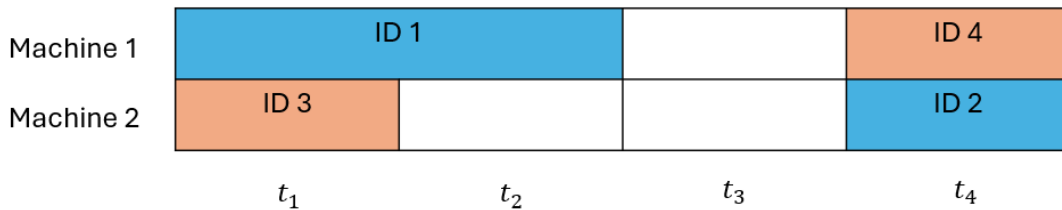


Figure 4.4: A toy example of 4 tasks and 2 different Jobs. Tasks of different jobs are denoted with a different color.

We can also visualize some of the constraints in this example like task-ordering:

## 4.2. THE MANUFACTURING PRODUCTION SCHEDULING PROBLEM

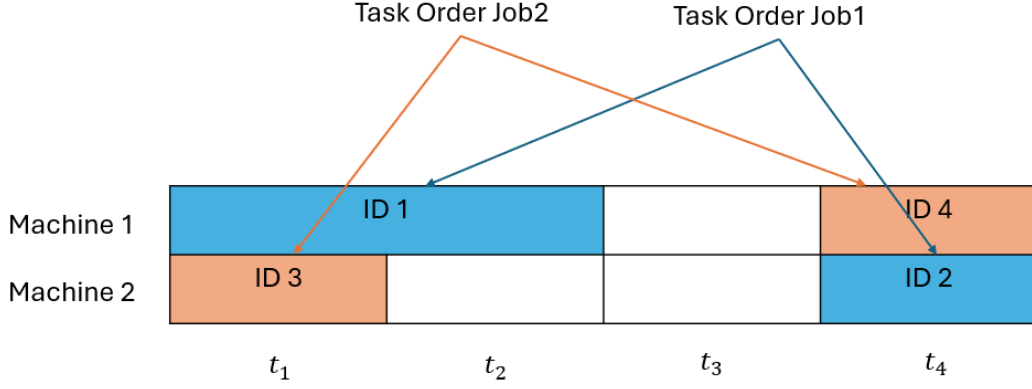


Figure 4.5: Ordering is denoted by the indices of the Tasks so ID 1 must be finished prior to the start of ID 2.

In the above picture we can also visualize no overlapping constraint as we can see that for a task to start the one running before should be finished. Also a possible inactivity window can be illustrated as follows:

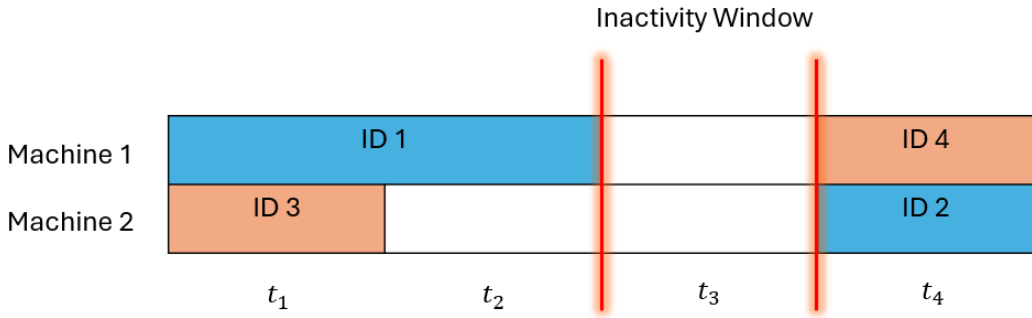


Figure 4.6: A possible inactivity window and how it would be handled by the optimization algorithm.

Lastly to visualize how a bitstring is produced and decoded there is a bitstring illustrated and analyzed below:

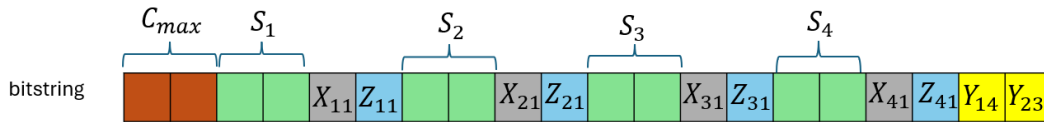


Figure 4.7: The outline of qubits representing our classical variables.

In the bitstring above:

- **C<sub>max</sub>**:  $C_{max}$  indicates the total time the production will take. For example, if  $C_{max} = 11$  then total time will be four timesteps.

## 4.2. THE MANUFACTURING PRODUCTION SCHEDULING PROBLEM

---

- $S_i$ : This set of variables corresponds to the starting time of the  $i$ -th task. For example, if  $S_1 = 00$  then the first task will start at timestep zero.
- $X_{i,m}$ : This is a binary variable denoting if machine  $m$  is used by the  $i$ -th task or not.
- $Z_{i,l}$ : If the value of this variable is equal to one this means that the  $i$ -th variable has finished before the inactivity window denoted by  $l$ .
- $Y_{i,j}$ : Those are tuples that denote that a pair of tasks is used by the same machine, so there is a potential of overlapping and conflict.

### QAOA approach

To implement this problem we started by creating small examples to ensure that every constraint can be satisfied. The two main constraints that come to play in every scheduling optimization problem of that category is the order constraint problem and the no overlapping/ campaigning constraint. The other limitations are too problem specific and help us construct a real world problem but on a research level validation of those constraints is crucial.

#### Task order constraint

To start evaluating task order constraint we created a really simple example with two tasks, one machine and one job. This simple example alone consists of 8 qubit when the maximum number of qubits is around 25-30.

Using QAOA we created different depth circuit to see the minimum depth circuit that can tackle with the problem. Different experiments were conducted using from 1 up to 20 layers.

There is proof of convergence for QAOA algorithms when circuit depth goes to infinity but for problems that can be well represented by undirected graphs.

- **1-Layer QAOA**

The cost function of the problem is depicted bellow. Convergence does happen but the problem is not expressed in the best way possible so the value that it converges to is not the optimal.

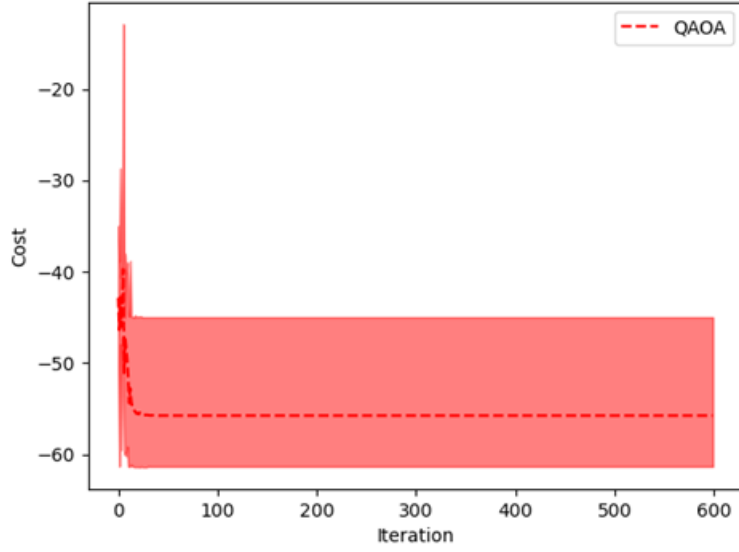


Figure 4.8: Cost function of 1-Layer QAOA using 8 qubits. The value it converges is sub-optimal, the depth of the circuit is not enough to capture the complexity of the problem.

The Gantt chart produced by the solution is depicted bellow. Here we can notice that the solution is not only sub-optimal but it is not even in the feasible set, meaning that the solution is not satisfied.

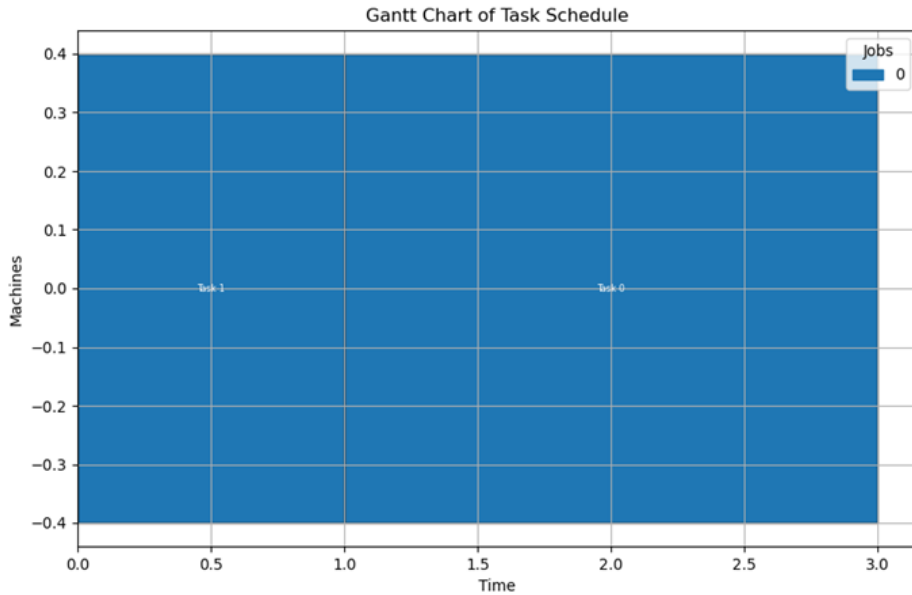


Figure 4.9: Gantt chart produced. The solution of the 1-Layer QAOA is not in the feasible set because task ordering constraint is violated.

- **12-Layer QAOA**

With 12 layers we can again see that the cost function does reach convergence. This time by seeing and analyzing the Gantt chart we see that we actually reach the optimal solution of the problem.

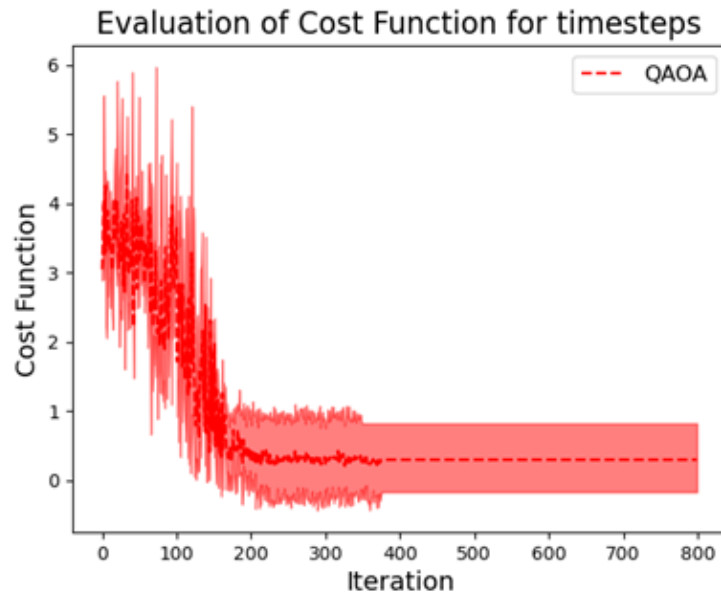


Figure 4.10: Cost function for 12-Layer QAOA. More iterations are needed for the cost function to converge but a better solution is produced.

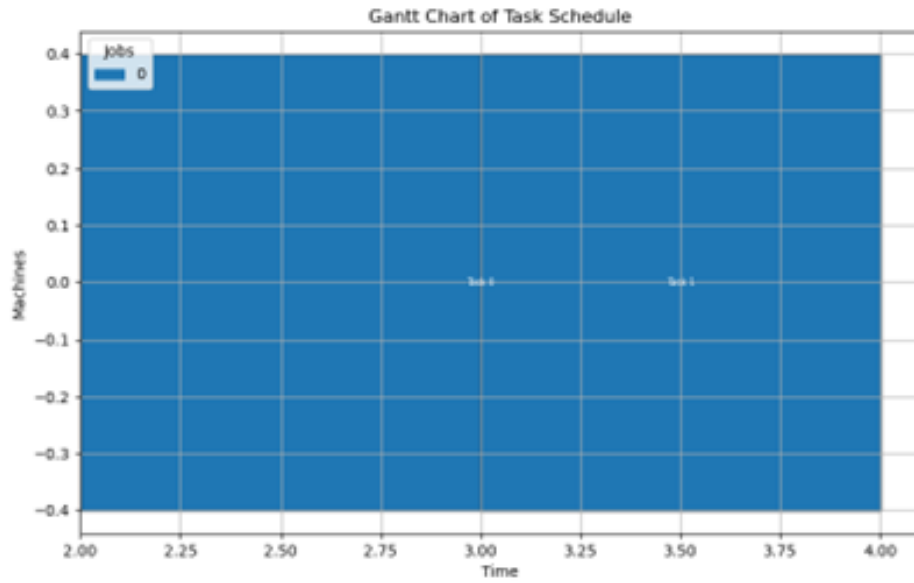


Figure 4.11: Gantt chart produced. The optimal solution that lies in the feasible set is depicted.

- **20-Layer QAOA**

It is observed that the algorithm converges again but as we see in the Gantt chart the solution lays in the feasible set but is sub-optimal.



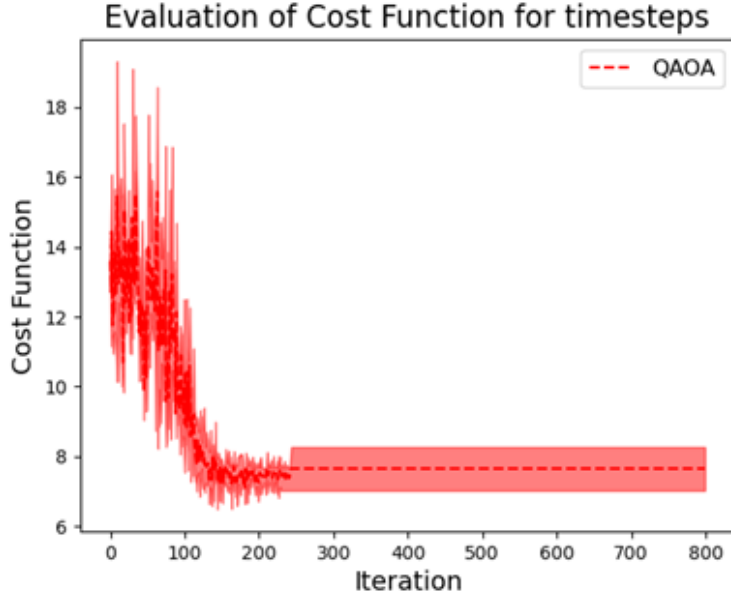


Figure 4.12: Cost function for 20-Layer QAOA. In this cost function that correlations between qubit are sufficiently expressed by the deepness of the circuit we can compare with the 12-Layer one and we can observe we reach a worse solution.

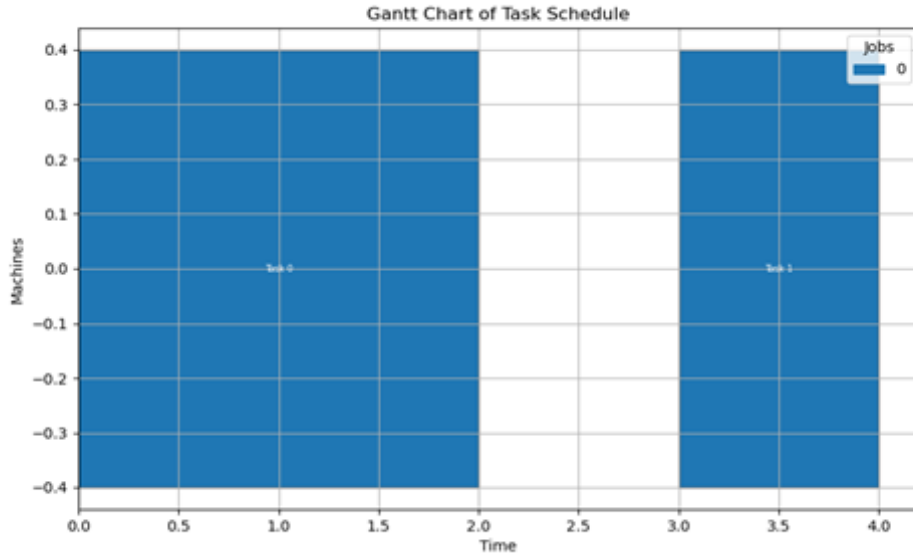


Figure 4.13: Gantt chart produced. The solution is feasible but it is sub-optimal so a timestep is not exploited as it should.

To conclude, the problem even with 20-Layers produce even sub-optimal solutions. This can be due to several reasons. First of all, this problem specific ansatz that expresses

really well problems that can be depicted by graphs does not express well this JSSP problem even for simple instances. With QAOA there is no point in creating and trying to solve bigger problems as we see that solutions are not optimal when time complexity for 20-Layers is already big enough.

### Hardware Efficient Ansatz Approach

In this section, we see how a generic ansatz like Hardware efficient ansatz deals with the problem. We also use the Augmented Lagrangian method so lagrangian multipliers are updated iteratively when a constraint is not satisfied.

#### Task order constraint

Same toy example is tackled here.

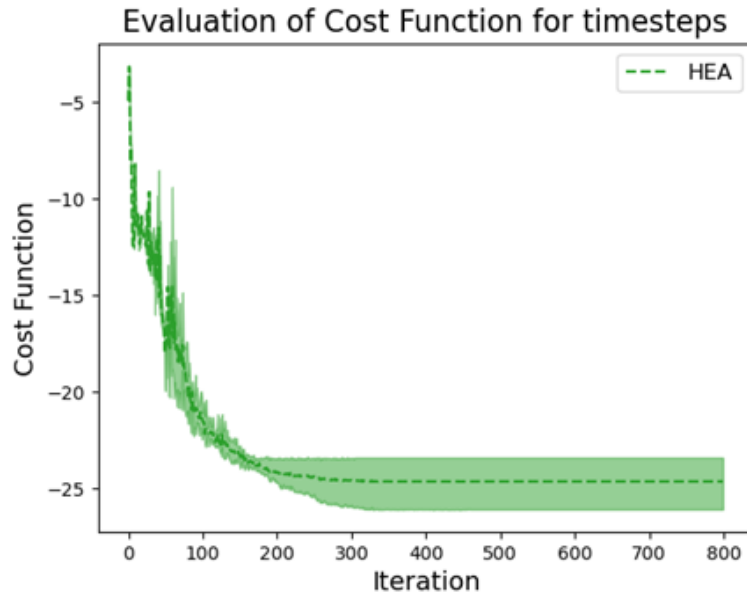


Figure 4.14: First Lagrange iteration. A constraint is violated here so the corresponding qubits are penalized accordingly using a different QUBO matrix.

First Lagrange iteration converges but the value is not feasible so another iteration is performed to reach a solution that lays in the feasible set.

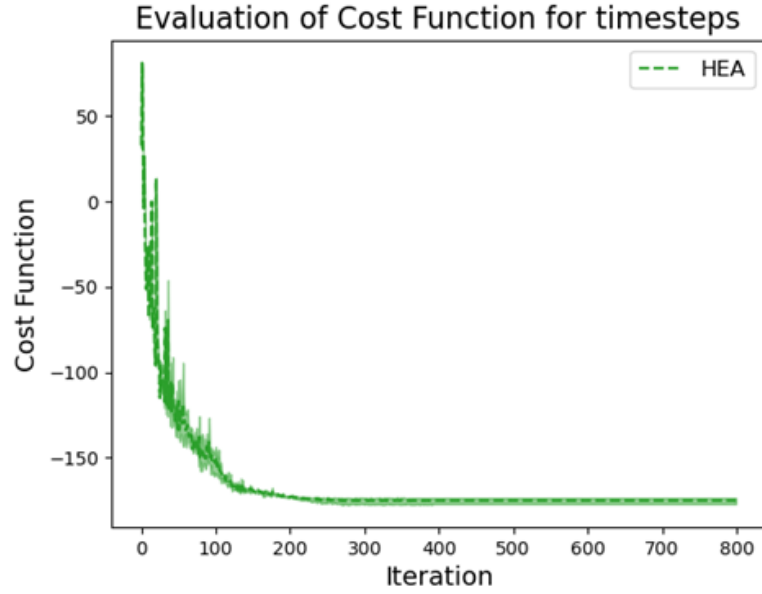


Figure 4.15: Last Lagrangian iteration. Algorithm terminates, indicating a feasible solution has been found.

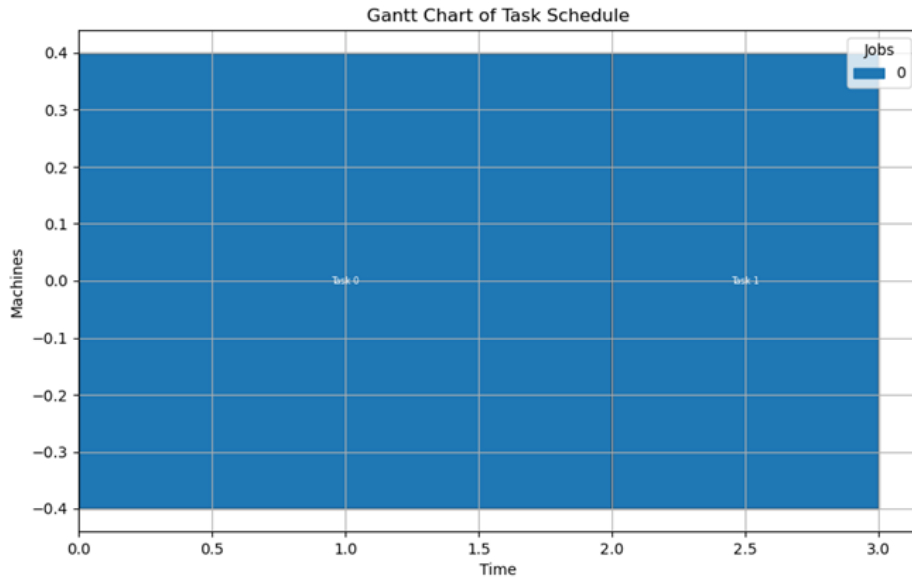


Figure 4.16: Gantt chart produced by the last iteration that both lays in the feasible set and is optimal.

This experiment was run several times. Every time a feasible solution was produced and most of the times the solution was also the optimal one. Time was minimal for this problem so we can see that there is potential to scale it up. Also we can conclude that

## 4.2. THE MANUFACTURING PRODUCTION SCHEDULING PROBLEM

---

the formulation of the problem in general is in the right track as the algorithm looks and finds the right answer.

### Task Order Constraint with Overlapping/Campaigning Constraint

For this toy example we used 3 tasks with 2 machines and 2 jobs, in a time window of 4 timesteps. The total number of variables is 12 which corresponds to 12 qubits. This is not considered a small problem for quantum computers.

The problem was run several times and in all of them the solution was in the feasible set. This ensures that the formulation is good and simple problems are well expressed.

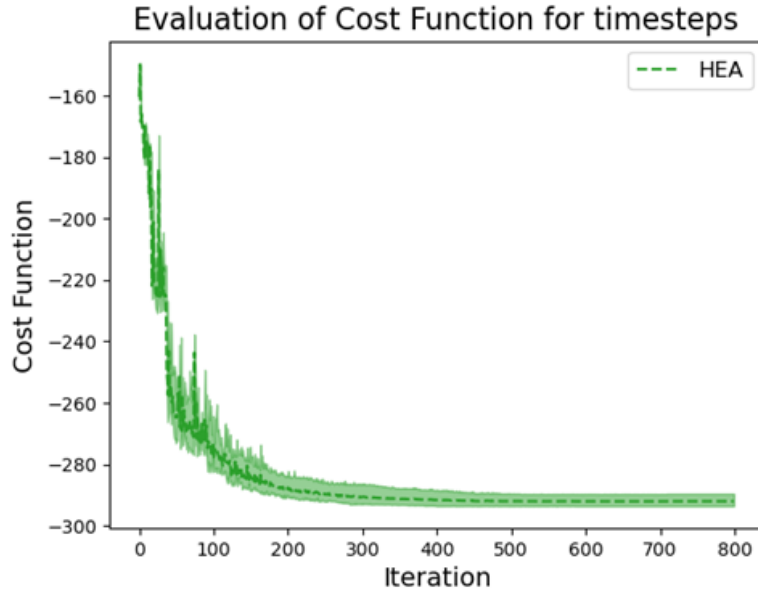


Figure 4.17: First iteration cost function. A feasible solution is not reached so more iterations are needed.

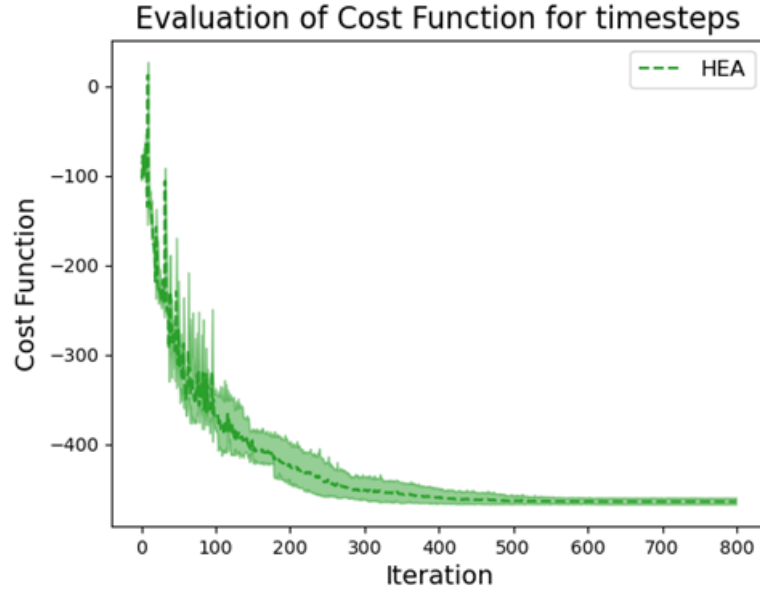


Figure 4.18: Last iteration cost function where all the constraints are met.

We see that cost functions converge but for example in the first iteration the solution is not feasible. In those bigger problems we can find and stack in local minima or even find global minimum but even then understand that it lays outside the feasible set. Problem has a lot of hyper-parameters that can be sub optimally tuned and also QUBO problems cannot tackle problems with too many constraints efficiently when created to also minimize the size of classical variables. Nevertheless, for this simple toy example we reach every time a feasible solution so we reach the terminating condition for the Lagrangian model as depicted in the gantt chart below.

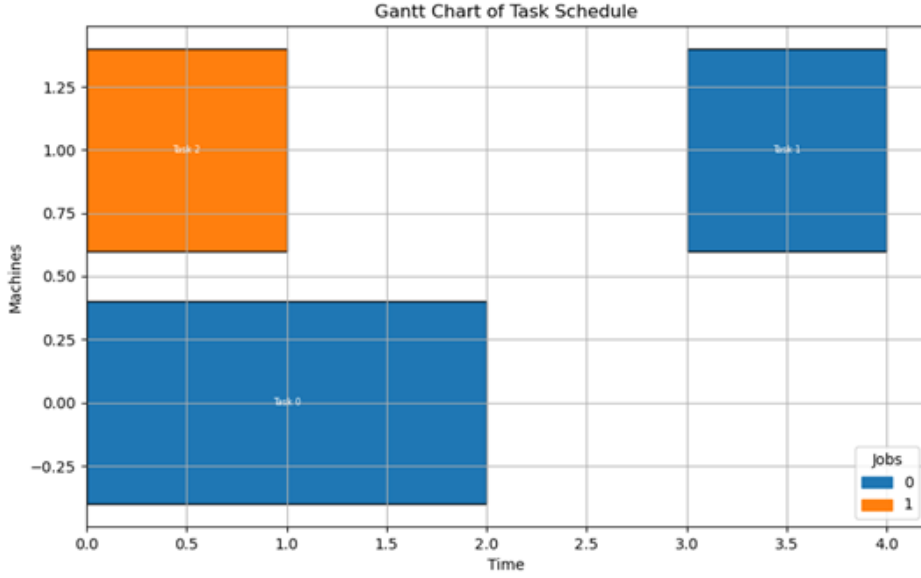


Figure 4.19: Gantt chart for 3-Layer HEA. A feasible solution has been reached but it is sub-optimal.

### Every constraint except inactivity window constraint.

In this section, an example is created to ensure that every constraint is satisfied. Inactivity window constraint is not included in the formulation cause with this variable minimization approach it adds a lot of complexity to the algorithm and good solutions cannot be captured.

For this example the total classical variables are 16. The total number of tasks is 4 with 2 machines, 2 jobs and a total number 4 timesteps. A 3-Layer HEA is used and we also use a 20-Layer QAOA to depict that QAOA cannot capture good solutions.

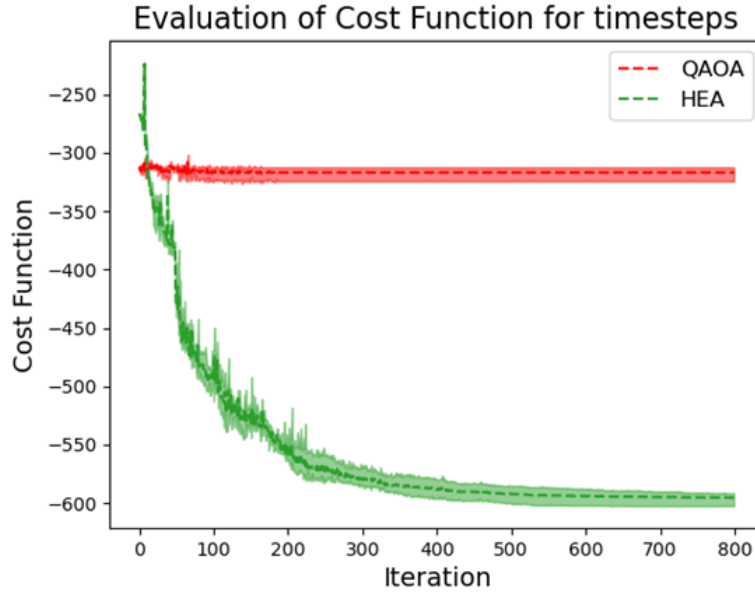


Figure 4.20: Cost function for QAOA and HEA.

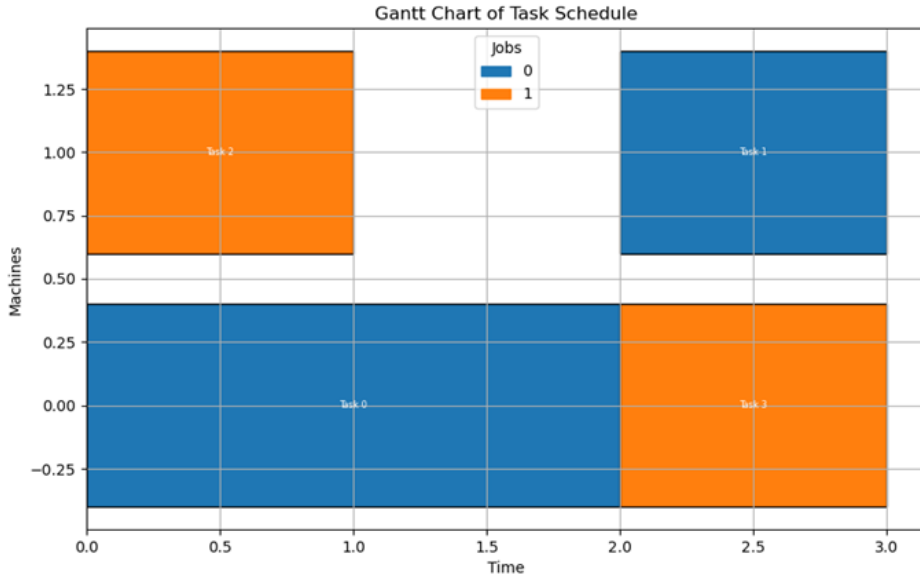


Figure 4.21: A promising solutions has been reached. In a more complex scenario with 4 total constraint the optimal solution was met.

We can observe that the QAOA does not even converge when the HEA works well for even a more complicated example. Algorithm was run multiple times and most of them produced a ground truth result depicted in the above gantt chart. Nevertheless,

scalability has its limits with classical hybrid approaches and we cannot tackle bigger state spaces in logical time just yet.

In the next chapter a simplification is made and with a qubit efficient scheme, bigger problems can be formulated with a non one-to-one mapping. Interesting results are produced with hybrid algorithms that gives hope for quantum approaches in the future.

### 4.2.5 Result Validation

In the whole section we formulated a complex problem as QUBO, adding all the constraints of the problem in the cost function to be optimized. Then we used classical quantum optimization approaches to tackle the problem and saw some promising results for toy problems. In the next years with more qubits coming into play there may be ways to solve bigger, close to real JSSP and take advantage of quantum mechanics special properties to solve NP-Hard problems in even polynomial time.



## Chapter 5

# Qubit Efficient Quantum Optimization for Scheduling Problems

In the following chapter, we extend the theoretical foundations and problem formulations developed earlier by focusing on practical implementations and experimental evaluations. A central challenge in applying quantum algorithms to real-world optimization problems lies in the limited number of qubits and high noise levels characteristic of current quantum hardware. To address this constraint, we introduce a set of qubit-efficient encoding schemes aimed at representing problem instances more compactly, while preserving their essential computational structure.

These encoding methods are designed to reduce the number of qubits required to represent scheduling problems—particularly instances of the Job-Shop Scheduling Problem (JSSP)—without sacrificing the expressiveness or correctness of the optimization formulation in a severe way. Alongside these encoding schemes, we also apply problem simplifications to reduce dimensionality, eliminate redundant constraints, and focus on core substructures of the original problem. This dual approach allows us to significantly scale up the size of testable instances, targeting problem formulations with a larger number of variables than would otherwise be tractable on near-term quantum devices.

With these optimized formulations in place, we proceed to implement and solve a series of problem instances using both quantum simulators and actual quantum hardware. The hardware runs are performed on available quantum processors provided through cloud-based platforms, offering valuable insight into the real-world performance and reliability of current quantum technologies. These tests allow us to evaluate the impact of noise, circuit depth, and device-specific limitations on solution quality.

Finally, we present and analyze the results of these experiments, focusing on key performance indicators such as solution fidelity, convergence behavior, runtime and of course feasibility. Comparisons are made between different encoding approaches. These

---

results contribute to our broader objective: evaluating the practical viability of quantum computing for industrial-scale optimization and understanding how quantum resources can be leveraged most effectively in the presence of real-world constraints.

Next we will try analyze and delve into the material published by [9],[10]:

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

Ioannis D. Leonidas, Alexander Dukakis, Benjamin Tan, Dimitris G. Angelakis,2024. *Qubit efficient quantum algorithms for the vehicle routing problem on quantum computers of the NISQ era*. Adv. Quantum Technol. 20, 2300309.

The ability of solving QUBO problems is a motivating fact that makes quantum computing an open research field with great potential. Nevertheless, quantum hardware is not sufficiently developed yet, so it is difficult to tackle real world problems with big inputs and data.

Quantum algorithms like QAOA and Hardware Efficient ansatz use a one-to-one mapping between classical variables and qubits. This encoding is used to fully exploit quantum hardware's properties like superposition and entanglement which better simulates correlations between qubits. The limitations of today's quantum hardware though makes scalability of those problems really small.

The paper analyzed in this chapter proposes solution to examine bigger state spaces with the hardware provided. The already proposed and used one-to-one mapping is referred as complete or full encoding. This full encoding takes every possible solution and translates it in a quantum basis state  $|\mathbf{x}_i\rangle$ , which leads in having a quantum state of  $2^{n_c}$  basis states where  $n_c$  is the number of classical variables.

So the quantum circuit and the optimization algorithms are producing a superposition which is :

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

This  $\mathbf{a}(\theta)$  represents a vector of the probabilities of every state of the  $2^{n_c}$  possible states. This quantum computing principal of testing multiple classical solutions simultaneously gives a computational advantage to quantum computing. Nevertheless, this one-to-one mapping has a trade-off of making the possible solutions and the Hilbert space immensely large for a small amount of input variables. So in this chapter by

---

introducing qubit-efficient encoding we try to minimize the qubits needed for binary optimization problems. The idea behind the encoding is, instead of representing the entire set of classical variables, we can divide the classical bitstring to subsets and use quantum superposition to evaluate all the possible solutions of all possible subsets simultaneously, to then compose a classical solution.

To put this idea in action firstly we need to divide the  $n_c$  classical variables into  $R$  subgroups. We can encode each classical variable the way we want, meaning each subgroup can have any number of binary variables and each classical binary variable can be a part of many subgroups. For simplicity, in this chapter we will use disjoint subgroups which have the same amount of variables on them. So we choose a number of variables each subgroup will have and we denote it as  $n_a$ . To be able to access and enumerate the different subgroups we use also qubits called register qubits.

The number  $R$  of subgroup depends only to the number of classical variables and to the number of the variables contained to each subgroup. To calculate  $R$ , we use the simple formula :

$$R = \frac{n_c}{n_a}$$

To denote each subgroup we use a number of qubits which is  $n_r = \lceil \log(R) \rceil$ . If a subgroup consists of just two binary variables then it will also have 2 ancilla qubits which will denote the values of the variables.

To better understand the encoding:

Let there be 8 classical variables that are separated into 4 subgroups so there are 2 ancilla qubits representing the values of the subgroup and 2 qubits representing the index of the subgroup.

Then for example:  $|x\rangle = |0001\rangle = |00\rangle |01\rangle$  which indicates that the values of the second subgroup are 00.

So the total qubits needed for this representation is:

$$n_q = n_a + n_r$$

And the quantum state that represents the encoding is :

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

where  $x_a^i$  denotes the value of the ancilla for register  $r$ .

## 5.1 Qubit Efficient Schemes explained

For this encoding scheme there are corner cases to consider. For example full or complete encoding is when the number of ancillas is equal to the number of classical variables  $n_a = n_c$ . Another corner case is for ancilla qubits equal to 1. In this case, the maximum minimization is done for qubits and the number of qubits used is:

$$n_q = 1 + \lceil \log(n_c) \rceil$$

In this corner case though, if there is a correlation between qubits it is not expressed and handled in the quantum computation process. So, in this case we do not take advantage of the entanglement principle of quantum mechanics. This approach is called minimal encoding, and the trade-off presented is that it encodes efficiently only statistically independent classical variables.

So, the idea is to increase the number of ancillas, and this way we can capture  $n_a$ -body correlations between classical variables. These correlations make sure that if there is dependency between classical variables it will be taken into consideration to finally produce the optimal bitstring. Concluding, we determine the number of ancilla qubits depending on the classical dependency between variables reaching full encoding when there is complete dependency.

### Cost Function

The cost function for QUBO problems is :

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

Nevertheless, in this approach the cost function is straight forward because the  $\mathbf{x}_i$  represents the whole bitstring  $|\mathbf{x}_i\rangle$ , so the complete classical solution is depicted. On the other hand to produce the cost function of a qubit efficient scheme a different process is needed. A qubit efficient formulation creates bitstrings that do not represent the whole solution. To create a full solution you need to take one ancilla measurement from each register and concatenate them into one.

So a solution like  $|\mathbf{x}_\alpha^i\rangle |r\rangle$  does not depict a full classical solution but a part of it. The cost function of a qubit efficient formulation is computed as follows:

$$\begin{aligned} C_\theta &= \sum_{\mathbf{x}} Pr_\theta(\mathbf{x}) \mathbf{x}^T \mathbf{Q} \mathbf{x} \\ &= \sum_{\mathbf{x}} Pr_\theta(\mathbf{x}) \sum_{i,j=1}^{n_c} x_i Q_{i,j} x_j \\ &= \sum_{i,j=1}^{n_c} Q_{i,j} \sum_{\{\mathbf{x} | x_i = x_j = 1\}} Pr_\theta(\mathbf{x}) \end{aligned}$$

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

In those equations  $\mathbf{x}$  represents the concatenation of the complete classical bitstring. The bitstring given  $x_i$  and  $x_j$  is equal to 1 is part of the subspace where those variables take certain values with  $2^{n_c-2}$  possible solutions.  $P_{i,j}^{1,1}$  represents the probability of both  $x_i$  and  $x_j$  being 1 where  $P_i^1$  represents  $x_i$  being 1. This representation of the cost function can be applied in various scenarios, and in every encoding scheme. Different encoding schemes result to different translation and calculation of the probabilities above.

## Minimal encoding scheme

The minimal encoding scheme is the simpler scenario of encoding that requires the least amount of qubits for the same amount of classical variables. The approach of minimal encoding represents the corner case where,  $n_\alpha = 1$  so every register group contains only one binary variable. So the total number of the subgroups is  $R = \log(n_c)$  and the total number of qubits needed is :

$$n_q = 1 + \lceil \log(n_c) \rceil$$

For the minimal encoding we can derive a formula to represent the quantum state as follows:

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

In this formula  $|\phi_i\rangle_r$  is the register group and the  $|0\rangle_\alpha$ ,  $|1\rangle_\alpha$  is the value of the register group (ancilla) multiplied with the corresponding probability according to measurements. This particular encoding encapsulates each classical variable in a register group and explores how it's value affects the cost function. The probability distribution:

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

The above formula indicates that to find the probability of each bitstring we multiply the probabilities of every register group having a specific value. The probability of  $x_i$  having a certain value like one is:  $Pr(x_i = 1) = |\beta_i(\theta)|^2$  and the probability of it being zero is the complementary:  $Pr(x_i = 0) = |1 - \beta_i(\theta)|^2 = |\alpha_i(\theta)|^2$ .

A simple example to understand the later formulation is having 4 classical variables. If we use minimal encoding we need 4 register groups to represent every classical variable,

so the number of register qubits will be two. In this representation for example bitstring  $|0\rangle|00\rangle$  represents the first group out of 4 and this group has a value of zero. So:

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

represents the classical bitstring:

$$\mathbf{x} = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}$$

In conclusion, minimal encoding does not allow us to capture correlations between qubits which makes it's use limited. Also to capture all the possible ways a bitstring can formulate we need at least 4 measurements, so we do not find possibilities for every classical bitstring simultaneously.

To calculate the cost function for minimal encoding we use the formula above. Because correlations cannot be captured, we consider variables independent, so the joint probability of two variables is just the product of the probabilities. So  $P_{i,j}^{1,1}$  is  $|\beta_i(\theta)|^2 |\beta_j(\theta)|^2$  when  $i \neq j$  and  $P_i^1$  is  $|\beta_i(\theta)|^2 |\beta_i(\theta)|^2$  when  $i = j$ .

To visualize the formula it becomes:

$$C_\theta = \sum_{i,j=1}^{n_c} Q_{i,j} |\beta_i(\theta)|^2 |\beta_j(\theta)|^2 (1 - \delta_{i,j}) + \sum_{i=1}^{n_c} Q_{i,i} |\beta_i(\theta)|^2 |\beta_i(\theta)|^2$$

## n-Body Correlation

There can be included more than one ancilla qubits in a group. This indicates that more qubits are needed in comparison to minimal encoding but some correlations between variables can be captured. For simplicity we will explain a case with 2 ancilla qubits in each register subgroup.

To measure an entire bitstring  $|\psi(\theta)\rangle$  that corresponds to a full classical solution:

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

where P is the set of pairs i,j. The computation of probabilities on this encoding is different from the minimal encoding but the intuition is the same. So for example if we want to calculate the probability of the i-th classical variable which lies on the r-th register group.

- If  $i$ -th variable is the first of the group:

$$Pr(x_i = 1) = |c_i(\theta)|^2 + |d_j(\theta)|^2$$

- If  $i$ -th variable is the second ancilla of the group:

$$Pr(x_i = 1) = |b_i(\theta)|^2 + |d_j(\theta)|^2$$

Now if we want to calculate probability of  $i \neq j$  and  $Pr_{i,j}^{1,1}$  then:

- If they are on different register groups:

$$Pr_{i,j}^{1,1}(\theta) = Pr(x_i = 1)Pr(x_j = 1)$$

- If they are on the same register group:

$$Pr_{i,j}^{1,1}(\theta) = |d_r(\theta)|^2$$

The conclusion of the encoding schemes is that like every minimization in information and connectivity in order to simplify a problem there exists an accuracy trade-off. Like every simplification we can notice that in order to encapsulate bigger problems we create formulations that do not capture correlations between every qubit. Intuitively we know that this makes our solution less accurate and possibly sub-optimal. Nevertheless, for specific problems that variables are almost independent qubit efficient schemes can work well and produce promising results. Qubit efficient schemes though bridge the gap between classical and quantum scalability in variables with the current quantum hardware, so they are a promising and interesting research area.

## 5.2 Qubit Efficient Formulation applied

### 5.2.1 Subset Sum Problem

To check the validity of our algorithm the first thing is to test it on simple examples. A simpler problem than a JSSP is the subset sum. The goal of the algorithm is to choose a subset of a vector  $\mathbf{a}$  that is as close as it can to a scalar  $S$ .

So cost function is :

$$C(\mathbf{x}) = \mathbf{a}^T \mathbf{x} - S$$

To make this cost function optimizable we square it so it has an optimal point.

So the final problem becomes:

$$\text{minimize}_{\mathbf{x}} C(\mathbf{x}) = (\mathbf{a}^T \mathbf{x} - S)^2$$

$$\text{s.t. } x_i \in \{0, 1\} \forall i.$$

## First Experiment

Firstly, a small vector was created with 12 variables to check the quality of different encoding schemes compared with the full encoding. For every encoding 5 different experiments were conducted and both the mean and the variance of every cost function was captured.

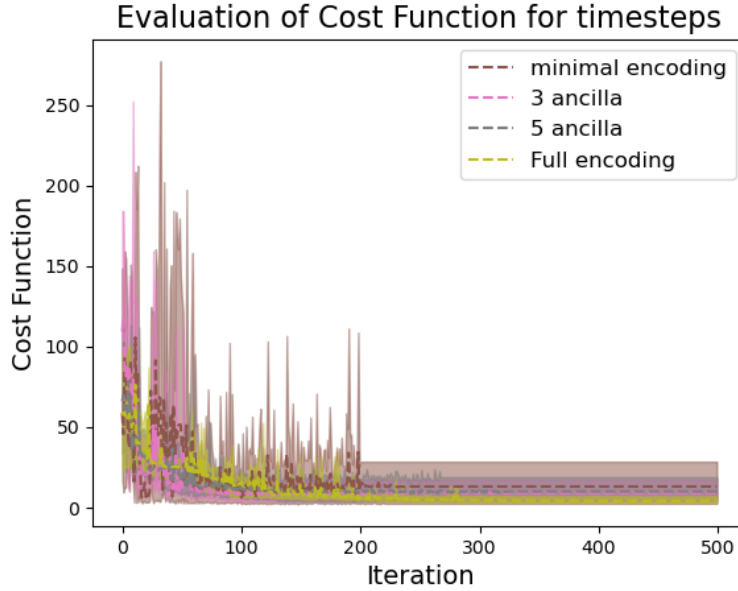


Figure 5.1: Cost function of 12 variables Subset Sum.

In the plot above we can see four different encoding schemes. First of all the minimal encoding that uses one ancilla qubit, and in total for 12 variables it needs only 5 qubits. Then, a 3-ancilla scheme and a 5-ancilla scheme is used needing 5 and 7 qubits. The last is the full encoding that captures correlations between all variables because the qubits and the classical variables used have a one-to-one mapping. The other encodings capture less correlations and dependencies.

Nevertheless, this problem is fairly easy and we can capture the best solution even with minimal encoding as seen in the plot. It is easily observed though that the mean of the experiments and the smoothness of the cost function is better when we capture more



dependencies between qubits. Reaching the optimal points with minimal encoding seems promising, but in the solution space there are several solutions that are global minima.

## "Full Encoding limits" Experiment

The next experiment is a 20 variables experiment. This is the maximum amount of qubits that we can reach with the full-encoding scheme in logical time on a normal laptop. Again different encodings are tested and evaluated.

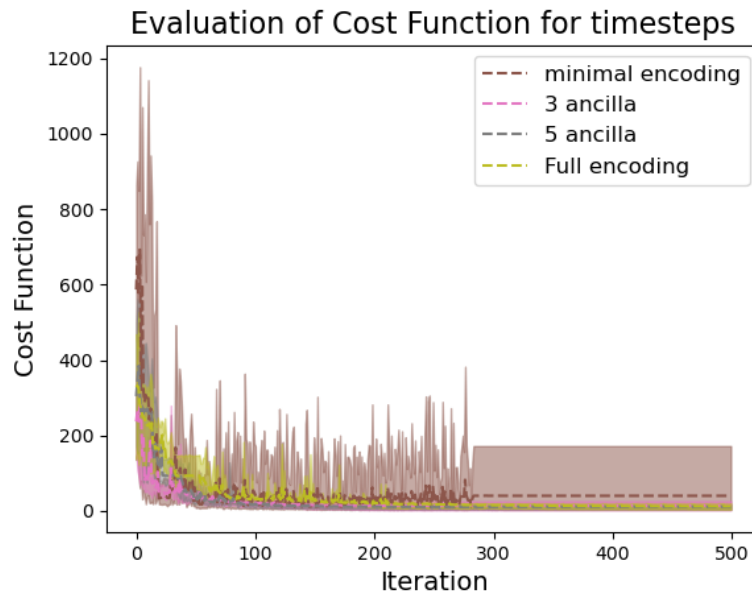


Figure 5.2: Cost function for different encodings on Subset sum.

It is easily noticeable here that in 5 experiments all encoding schemes reach the optimal value but the reliability of encoding schemes with more ancillas is better. The variance that we notice in minimal encoding starts to become an issue if we want to produce optimal solutions consistently in every run.

## 50 variables Experiment

In this experiment we do not use full encoding any more and we use a 10-ancilla encoding scheme.

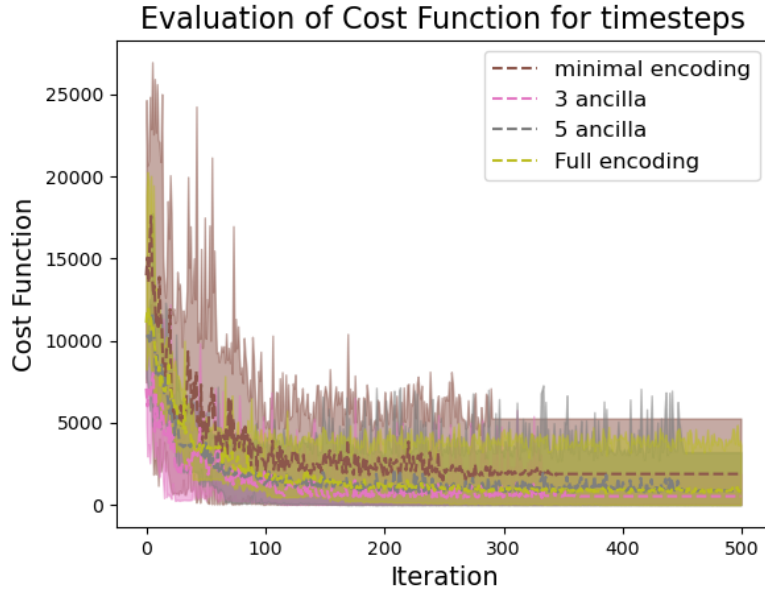


Figure 5.3: Cost function for Subset Sum with a 50 variable vector

As expected still minimal encoding has the biggest variance so single experiments are less reliable, however even with 50 variables we are able to produce optimal or near-optimal solutions even with minimal encoding. All the other encoding schemes seem to work in a similar way having the same variance and mean solution. We would expect in such a problem with fully dependent variables to produce slightly better solutions with more ancillas, but maybe this is due to the vector being fairly easy.

## Last Experiment.

In this experiments we pushed the variables to 100 classical variables. Four encoding schemes where tested.

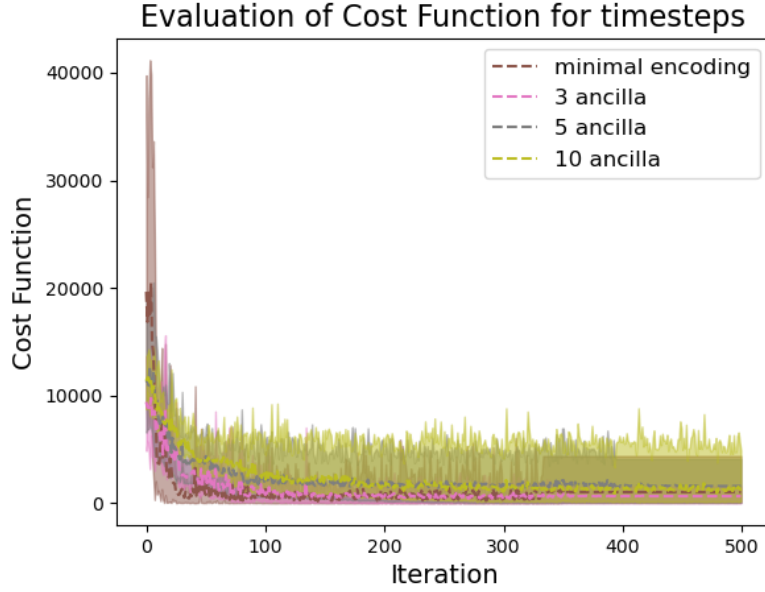


Figure 5.4: Cost function for 100 variables.

The variance of all experiments is fairly big. We still capture the optimal solution in all encoding schemes. For those experiments we project in a solution space which has  $2^{100}$  possible solutions so maybe the variance is noticed in the experiments due to poor exploration in the solution space. The total amount of measurements we use is 100,000 so some "instances" may never be explored which adds some randomness in the final solution and some variance in the cost function between experiments.

The fact that minimal encoding in big experiments captures better solutions with less variance is difficult to explain and counter intuitive. More ancillas in a register means that dependencies between classical variables are captured in the quantum circuit so we have more information to work with. Nevertheless, the amount of measurements and the poor exploration of the solution space and the complexity of the circuit in 10-ancilla groups may introduce a hardness which is not suitable for a fairly easy combinatorial problem as Subset-Sum. Reaching full-encoding and using the suitable measurements though should yield the best solution for problems that can be represented from a fully connected or a dense graph.

### 5.2.2 Manufacturing Production Scheduling Problem using Qubit Efficient Schemes.

In the chapter before we explored how far we can get into scaling with the more "classical" quantum approaches. However, it was visible that using the current quantum hardware

we could tackle only toy examples that do not correspond to real world problems. In this chapter we are going to explore if by using qubit compression we can deal with problems big enough to be translated into real world problems. We will then validate the results to examine feasibility and convergence.

The output to be validated will be a Gantt graph with the schedule solution, a cost function and a ratio of constraints satisfied if the problem's solution is infeasible.

### Problem simplification

The mathematical formulation we created to deal with the exact problem at hand has too many constraints and it is really complicated for a QUBO formulation. So, we created a simplified version of the problem with the only constraints being the overlapping/campaigning constraint. For large problems those constraints are still a lot but we will see how the algorithm react when putting more tasks and Jobs into play.

The simplification made does not allow machine alternation, meaning that every task can only be assigned to a certain machine. Also there are no inactivity windows or vacation dates meaning that every timestep in the Horizon is a feasible starting and ending time for each task. These simplifications capture a problem close to the one we are solving but have less constraints and a smaller amount of variables because  $X_{i,j}$  and  $Z_{i,j}$  are missing. So the bitstring of the solution is modified as follows:

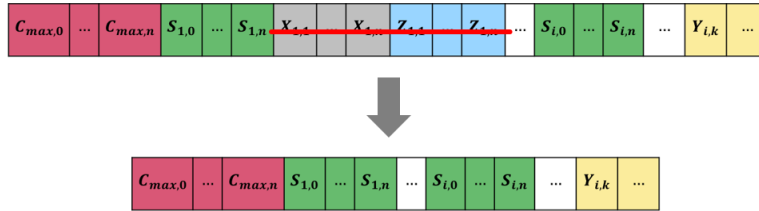


Figure 5.5: Bitstring conversion after simplification.

So we will start by increasing the tasks, the Jobs and the Horizon to reach an inflection point. The output will resemble the Gantt graph bellow and we will be able to conclude if the solution is sufficiently good or not.

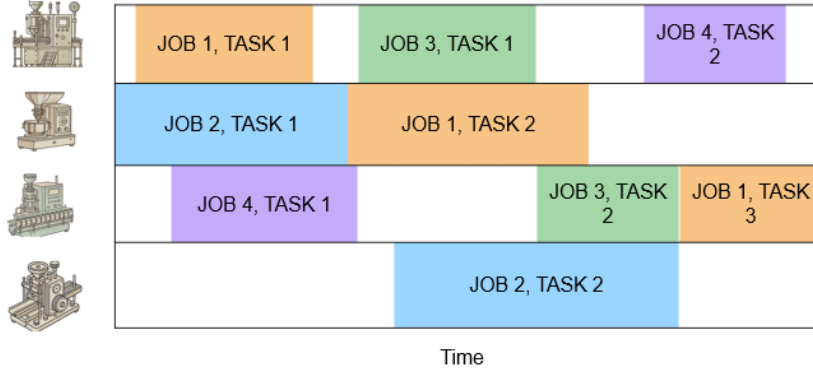


Figure 5.6: Gantt chart.

To minimize the input string we will start by deciding how many ancilla qubits we will use. Then the total amount of qubits will be determined to encode the amount of starting variables. The amount of ancilla qubits should be determined in way to not lose too much information from variable dependencies and also to not use an excessive amount of qubits that the HEA cannot converge in logical amounts of time.

To validate the solutions we produce we had to come up with some kind of reliable metric. To do that it is believed that the cost function alone is not a sufficient metric. QUBO formulation adds all the constraints in the cost function penalizing the ones that are violated with a big enough penalty. Nevertheless, in those multi-constraint problems fulfilling one constraint may lead to violating another due to how cost function is formulated. Even when fine-tuning by trial and error is made in the hyper-parameters using QUBO for JSSP is challenging so cost function as a metric is not enough in our opinion.

The first metric used is a simple feasibility metric that captured the ratio of constraints satisfied.

Then using a Google-OR Tools algorithm we produce the optimal solution. This algorithm is a MILP formulation that does not encapsulate the constraints in the cost function but instead the constraints are used like a classical optimization problem. We feed this optimal solution bitstring in a VQA circuit to produce the corresponding cost function value of the QUBO and then compare the Gantt Chart produced by the Google-OR tools with the quantum solutions.

### First Experiments.

The first experiment conducted had 50 classical variables. Even this amount of variables cannot be tackled by "classical" quantum algorithms, so we used the encoding schemes that allow as to bridge the scalability gap between classical and quantum ap-

proaches. The problem above corresponds to 12 tasks that belong to 4 Jobs and 4 machines. In those experiments we used different encoding schemes like minimal encoding, 3 and 5 ancilla qubits and also different circuit depths like 3 or 4 layers that we noticed produce the best results. As before the solution space our problem is projected into has  $2^{50}$  possible solutions, so to explore the solution space we need a big amount of measurements.

To visualize the circuits used for the experiments we produce a circuit for a 7 qubits and 3 layers experiment:

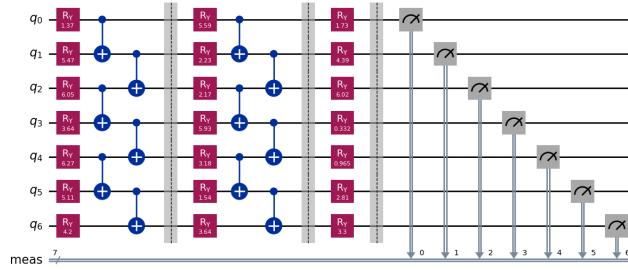


Figure 5.7: HEA of 7 qubits from a starting total of 50 qubits and a total of 3-Layers.

To produce a solution as reliable as possible we used 300,000 measurements to both sufficiently explore the solution space and to run the algorithm in logical time.

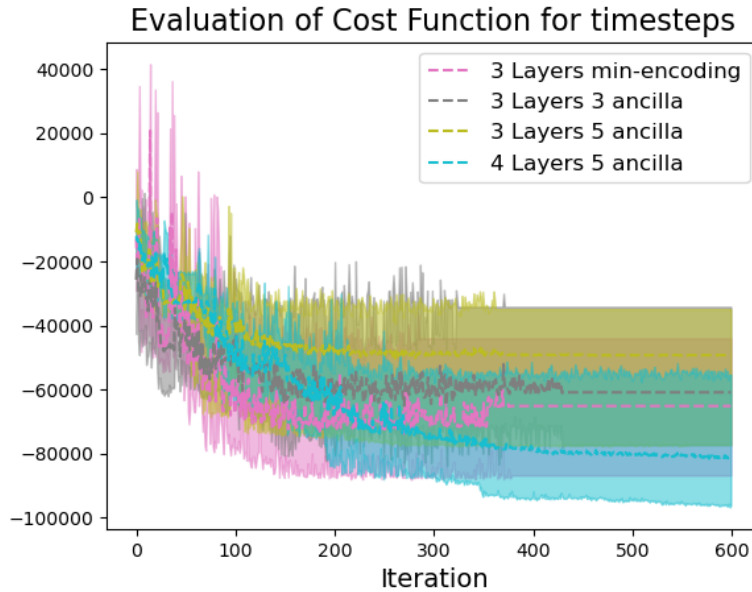


Figure 5.8: Cost function for different encoding schemes. The problem consists of 50 classical variables and 47 constraints.

In the above cost function we see some kind of convergence in all the encoding schemes but there is big variance between the experiments. So, to have a better visualization of the quality of each approach we will produce the Gantt chart of Google-OR tools and compare it with the Gantt Chart produced by the encoding schemes.

- **Minimal-Encoding**

In the minimal encoding the total amount of qubits is 7. The Gantt Chart produced has a ratio of 70% of constraints satisfied with a total of 47 constraints and the quality of the solution is not good as seen bellow:

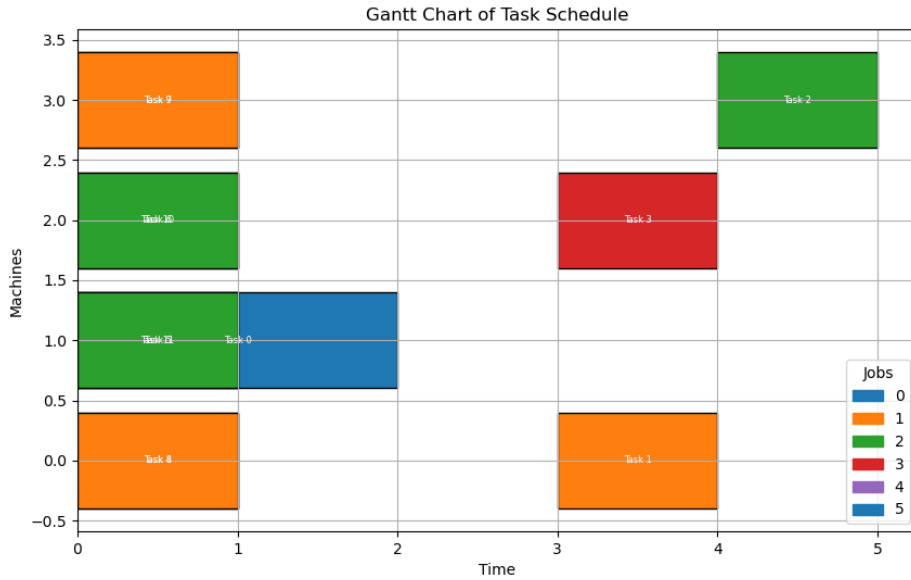


Figure 5.9: Gantt Chart for minimal encoding with a ratio of satisfied constraints reaching 70%.

- **3 ancilla qubits**

The ratio of satisfied constraints here is better with a total of 75% of constraints satisfied but we see that the tasks are stack to certain starting times. Its hard to intuitively explain this fact, probably this is a local minima of the algorithm and the solution is stack there. The quantum properties like quantum tunneling could help in a different landscape but with those penalties introduced in every unsatisfied constraint gives as a really noisy and non-smooth landscape which is really difficult to navigate.

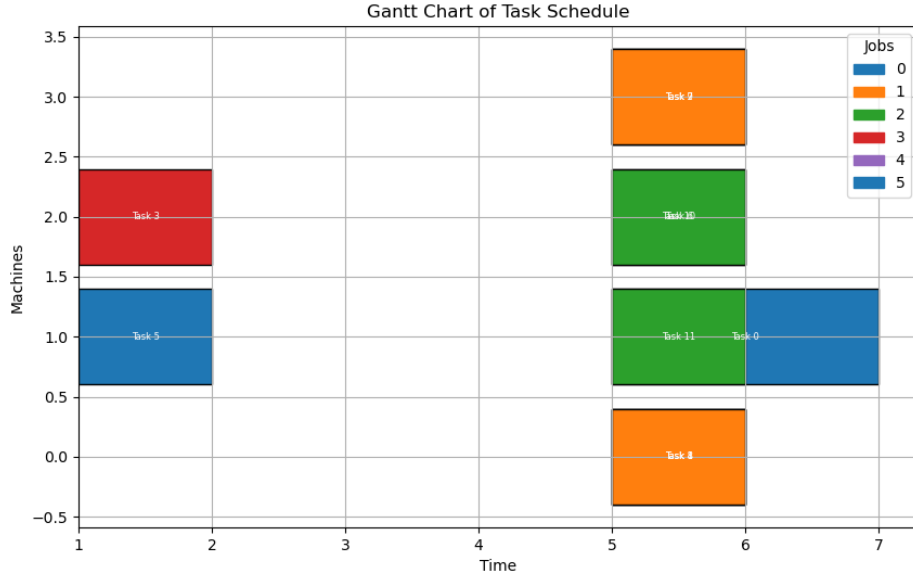


Figure 5.10: 3 ancilla qubit solution Gantt chart with a total of 75% ratio of satisfied constraints.

### • 5 ancilla qubits

In this encoding scheme a total of 9 qubits is used and a total of 91% of constraints is satisfied and the tasks are more evenly spread in the Gantt Graph. It seems that the solution is promising in this Gantt Graph, the solution is not random but kind of sophisticated. The formulation is not ideal for such a number of constraints so the quality of this solutions is deemed good.

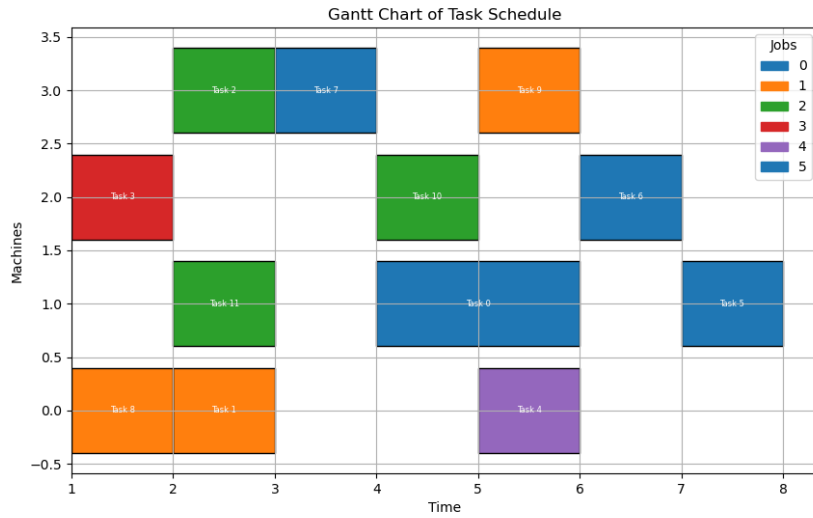


Figure 5.11: 5 ancilla qubit solution Gantt chart with a ratio of around 91%.



- **5 ancilla qubits 4-Layers used.**

Making the depth of the circuit bigger does not make the solution better. The total ratio of satisfied constraints is 83% and we can notice from the Gantt Graph that the solutions worse.

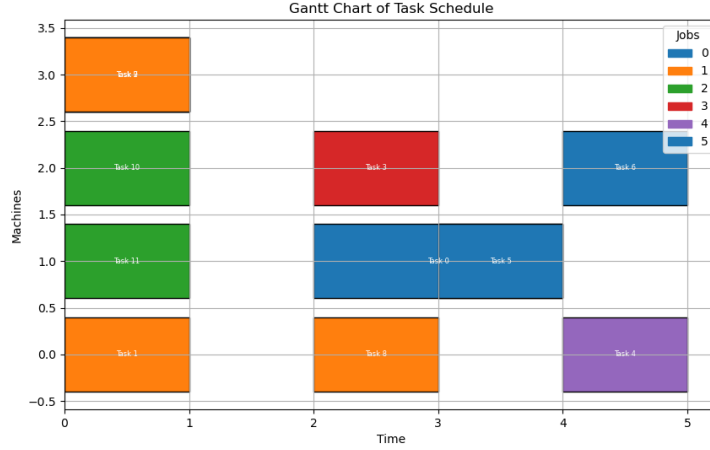


Figure 5.12: 4 Layer 5 ancilla Gantt chart reaching an 83% ratio of satisfied constraints.

The fact that a worse cost function value produces a solution worse than a better cost function value is counter intuitive. This is due to QUBO adding all the constraints to the cost function. There are a lot of hyper parameters to be tuned to both minimize the total time of the schedule and to satisfy all the constraints. The parameters are not optimally tuned so solutions which push all the tasks at the start have a smaller cost function even when violating some constraints. From the hyper-parameters used those produce the most reliable solutions, so we kept on using them.

### Medium Size experiments.

In this section an experiment with 100 classical variables is conducted. This corresponds to 18 tasks, 6 Jobs and 6 machines. Same experiments as above are done with 1, 3 and 5 ancilla qubits. The Google-OR Tools solution produced for the above problem is the one below:

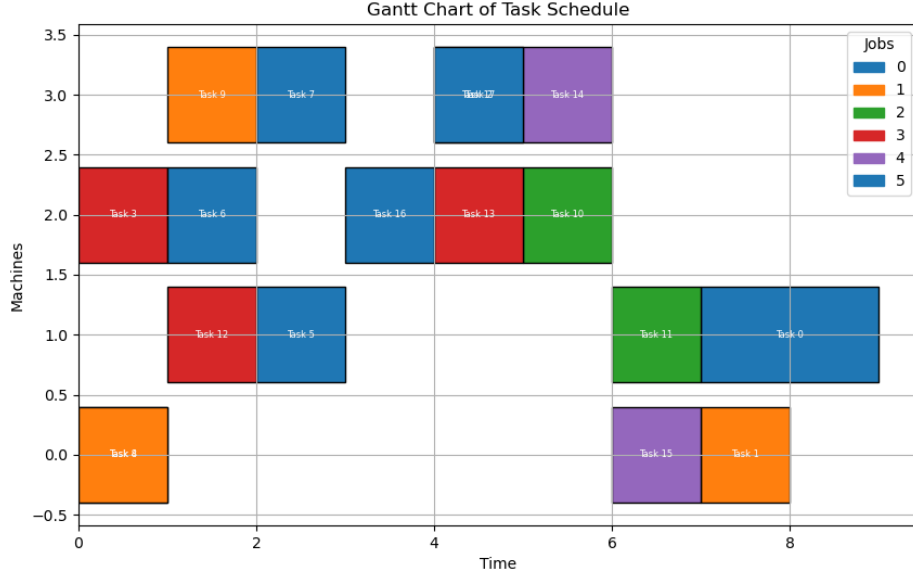


Figure 5.13: Google-OR Tools solution Gantt chart showing the optimal solution we should get.

In this experiment we are already in higher spaces than quantum computing can handle without qubit compression. Those problems start becoming difficult for our QUBO formulation and the solution becomes worse. We can see in the cost function that every encoding converges to a different value so we know for sure they are stuck in local minima and that even with enforcing constraints with big hyper-parameters we do not get to feasible solutions.

### • Minimal Encoding

For minimal encoding we use 1 ancilla qubit so the total amount of qubits is  $n_c = n_a + n_r$  which is  $n_c = 8$ . The total amount of measurements used for these experiment are 600,000.

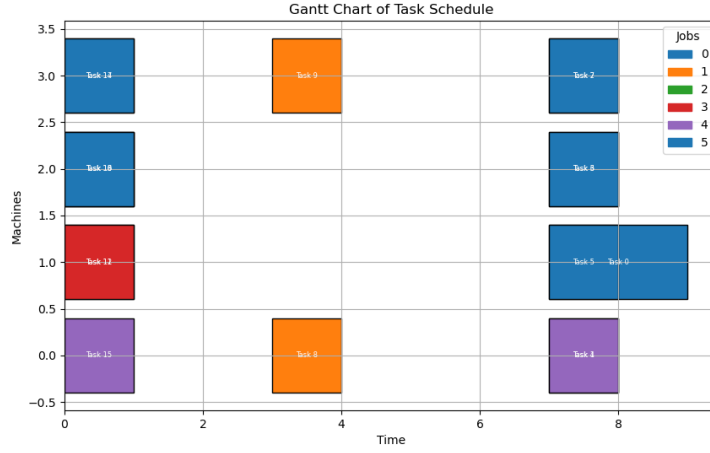


Figure 5.14: Gantt Chart of the best experiment with minimal encoding

We can see that starting time of tasks is stuck to 1 or 5 prioritizing the minimization of the starting time without caring about enforcing constraints even with high hyper-parameters.

### • 3 ancilla qubits

For 3-ancilla encoding the total number of qubits is  $n_c = 8$  and the total amount of measurements used is 600,000.

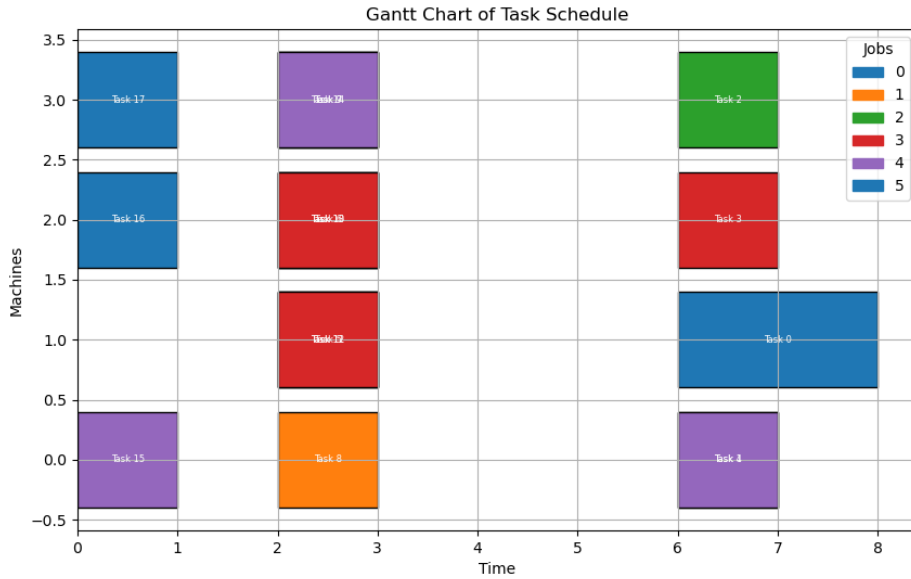


Figure 5.15: Gantt chart for 3-ancilla encoding schemes.

In this experiment, we notice that even if the cost function is reaching a higher point of convergences it seems to capture some correlation between tasks and some spreading of starting times is beginning to happen.

### • 5 ancilla qubits

The 5-ancilla qubit encoding scheme consists in total of  $n_c = 10$  qubits. The amount of measurements is the same and the Gantt chart is depicted bellow:

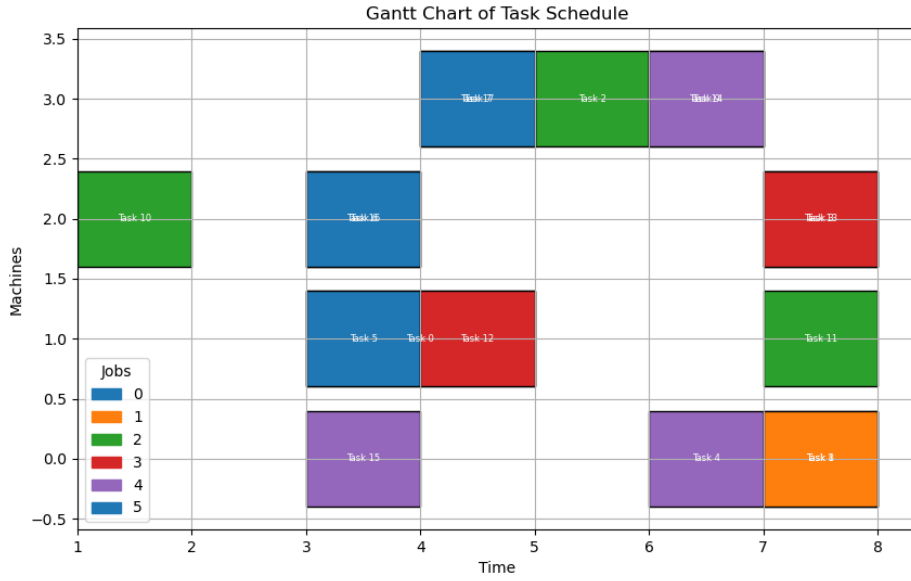


Figure 5.16: Gantt chart for 5-ancilla qubits encoding scheme reaching an 83% ratio of satisfied constraints.

The results seem to capture more of the constraints with a total ratio of satisfied constraints reaching 83%. Nevertheless, the result stacks to certain points of time.

### • 4-Layers, 5 ancilla qubits.

This experiment took the most time as it consists of 10 total qubits but the amount of entangling and single rotation gates is bigger due to circuit's bigger depth.

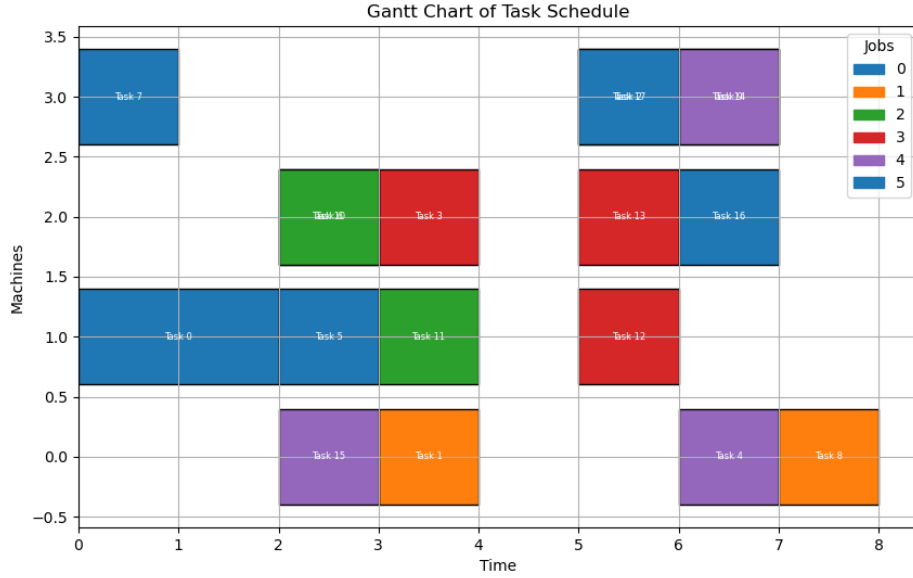


Figure 5.17: Gantt Chart for the 4-Layered experiment.

Here we see the best result yet with a feasibility ratio of 86% reaching a solution that tries to enforce the constraints.

### Large Size Experiment.

In this experiment we try to capture the inflection point. This is the point where the quality of the solution starts becoming worse and worse. The measurements that has to be done to capture better solutions and the amount of qubits that must be used, makes the total runtime of the algorithm unrealistic. It shows that there is a gap between both the scalability and the quality of solutions that must be bridged between classical and quantum optimization yet. This is a logical outcome because quantum computing is still evolving, nevertheless some promising results have been produced that quantum computing may be able to tackle difficult problems in the future.

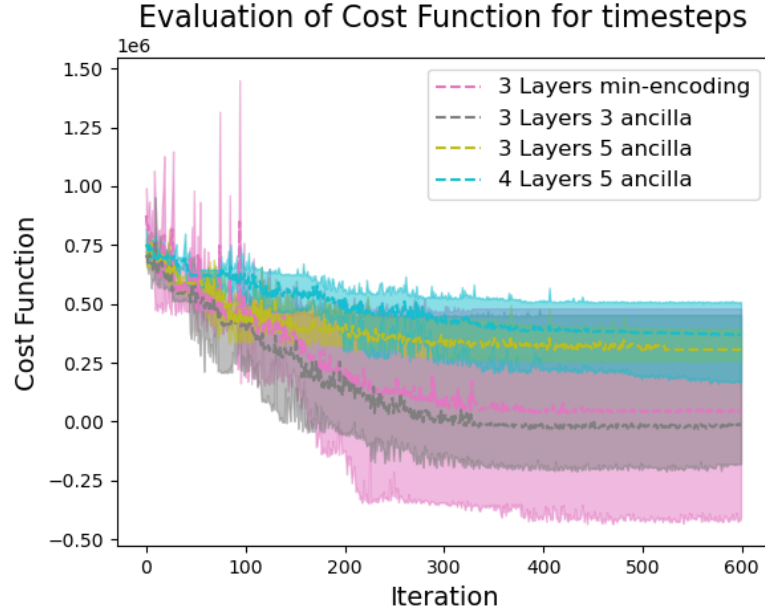


Figure 5.18: Cost function for 28 tasks problem. This experiment consists of 200 classical variables and a total of around 200 constraints to be satisfied.

- **Minimal Encoding**

The total measurements here were 1,000,000 and the runtime was 10 minutes per experiment even when the total number of qubits was only  $n_c = 9$ .

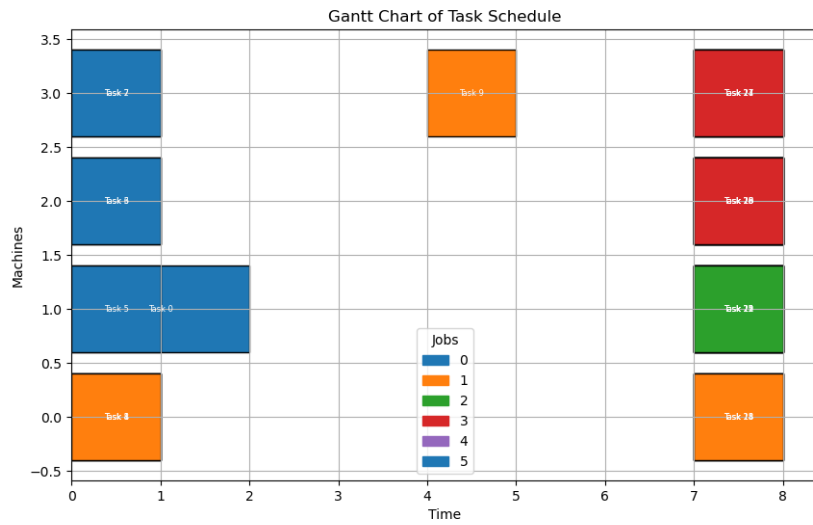


Figure 5.19: Minimal encoding for 200 variables.

- 3 ancilla qubits

In this experiment the total amount of qubits where  $n_c = 10$  making the total runtime up to 1.5 hours. The same problem seems to appear where the minimization of the cost function prioritizes the minimization of the total runtime without capturing the constraints even when they are heavily penalized.

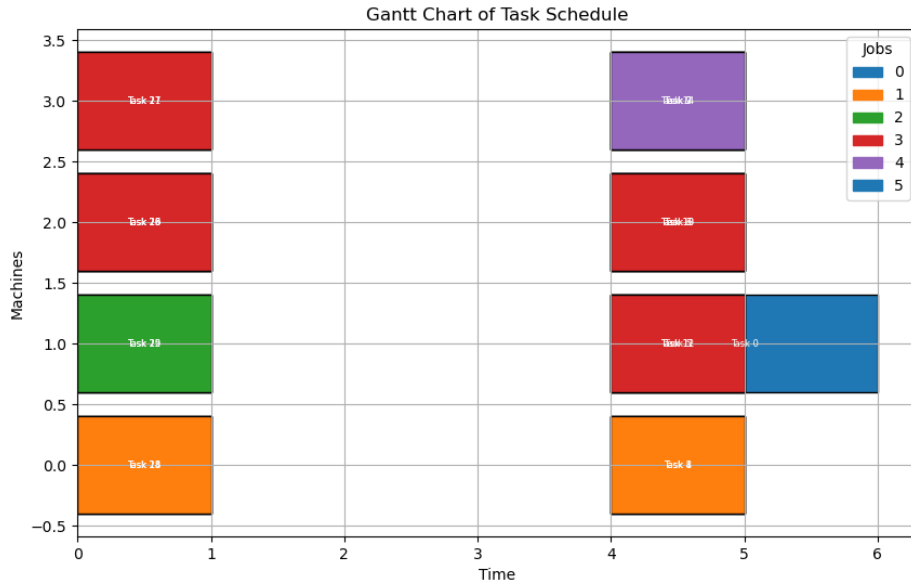


Figure 5.20: 3 ancilla qubits Gantt chart.

- 5 ancilla qubits

A bigger spreading in starting times is happening as before but the total ratio of satisfied constraints is 71% making the solution infeasible.

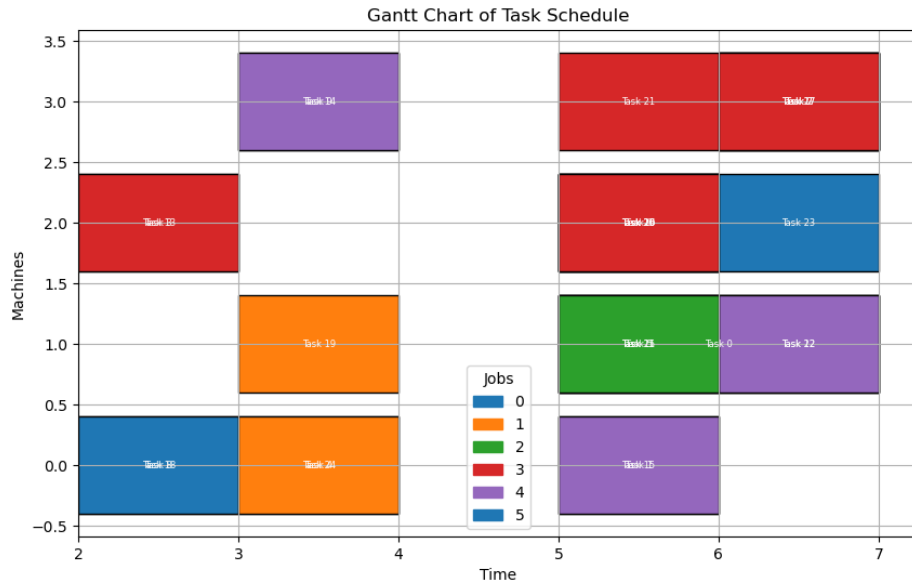


Figure 5.21: 3-Layers 5 ancilla qubits Gantt chart.

- **4-Layers, 5 ancilla qubits.**

Here we notice a total ratio of 75%. Task ordering constraint seems to be forced but the overlapping constraint is lacking according to the violated constraints and the Gantt Chart below.



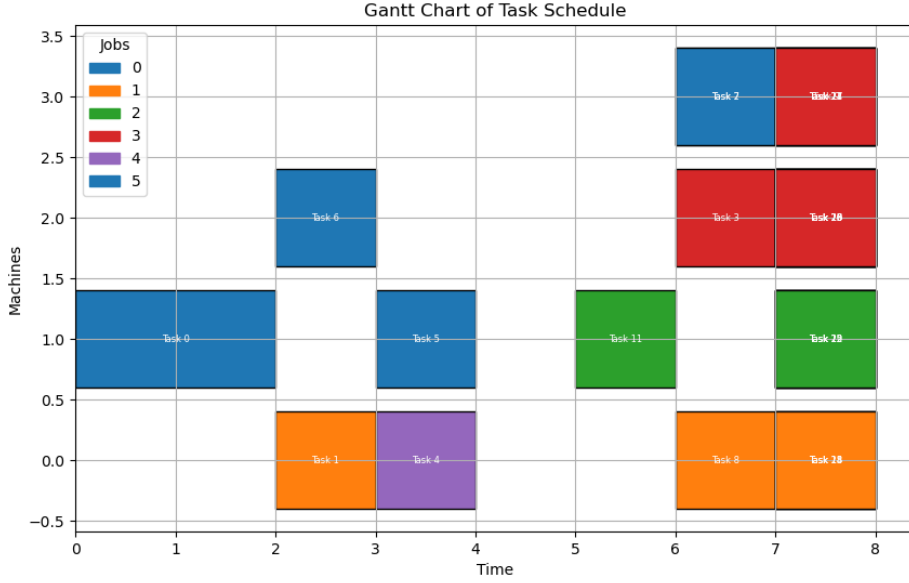


Figure 5.22: 4-Layered 5-ancilla encoding scheme gantt chart.

Even if some of the ratios for 5-ancilla qubits are acceptable 71% – 75% we can notice that this amount of constraints is not solvable from our formulation. QUBO formulation seems to struggle solving multi-constraint problem both in theory and in practice. Trying to force 200 constraints in a QUBO matrix using addition creates a huge and not-smooth landscape for the cost function which makes the solution stack in local minima even with quantum mechanics properties like quantum tunneling.

### 5.2.3 Execution on Cloud QPUs

To evaluate the solution we need to also use real QPUs and not only noiseless simulators. So in this chapter we used some real QPUs from IBM Cloud like Fez, Marrakesh, Sherbrooke, Strasbourg, Brussels and Aachen and compared those solutions produced with the noiseless simulator ones.

## Small Scale Experiment.

Due to IBM Cloud services being an expensive tool we were able to run only the last iteration of the experiment. After that we took the simulator solution and we compared it with the real QPUs experiments using a normalization. The tools used to compare

the cost functions is a Cumulative Distribution Function (CDF). In all the experiments a 3-Layer, 5-ancilla circuit is used that seemed to produce the best results.

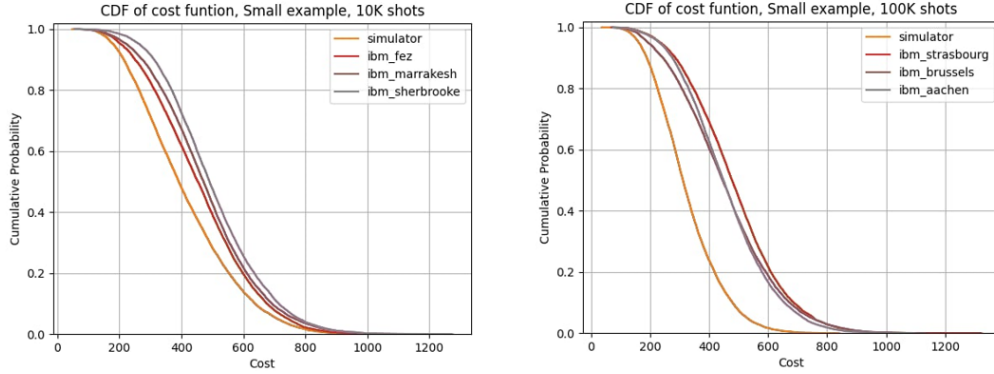


Figure 5.23: Small scale experiment CDF.

The quality of the solution is given by comparing how close is the CDF of the QPUs to the simulator solution. The solution is normalized, so using more shots seems to be further from the simulator solution but actually the simulator solutions with less shots is worse.

### Medium Scale Experiment.

The second experiment is the one having around 100 classical variables. For this one we deemed that 10k shots would not be able to produce an acceptable solution so we used only 100k shots and the results are presented bellow.

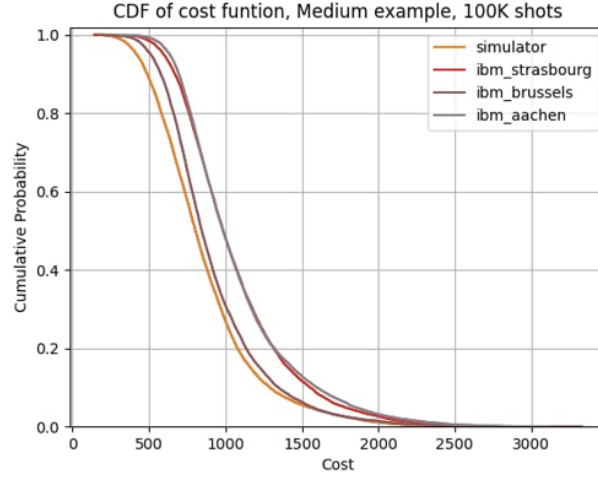


Figure 5.24: Medium scale experiment CDF.

## Large Scale Experiment.

The last experiment was the large scale one with around 200 classical variables and 200 constraints corresponding to 28 tasks. We were not able to produce good enough solutions for this problem even on simulators. Nevertheless, this chapter is not about the quality of the simulator solutions, its about testing how close to that solution we are able to reach with quantum machines. So when algorithms reach optimality reliably so does our quantum hardware. That being stated the real hardware solutions for this problem were the ones bellow:

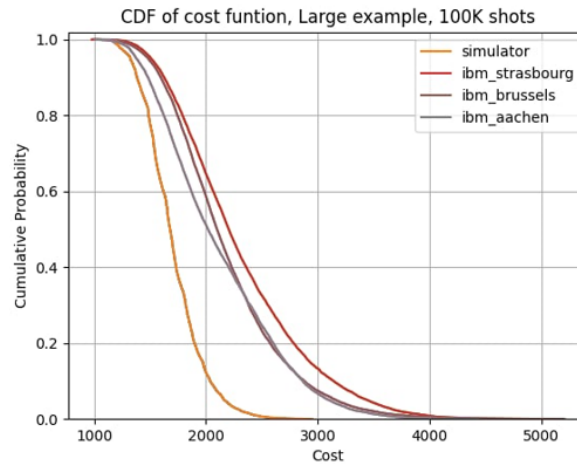


Figure 5.25: Large scale experiment CDF.

It can be noticed that in the medium scale experiment the QPUs were able to produce better results. This is due to having less qubits and less gates. Noise is more impact-full with more gates especially 2-qubit ones because the probability of error becomes bigger.

## Chapter 6

# Conclusion and Future Work

In this thesis, we investigated the potential of quantum computing for solving the Job Shop Scheduling Problem (JSSP), one of the most challenging NP-hard combinatorial optimization problems. The work comprised a full mathematical modeling of the JSSP, its conversion into a quantum-suitable formulation, and the development and evaluation of quantum algorithms tailored to this setting.

Initially, the JSSP was formulated as a Mixed-Integer Linear Program (MILP), incorporating all necessary inequality constraints. The disjunctive model, a widely accepted approach in the literature for such scheduling problems, was employed due to its ability to minimize the number of decision variables. However, since quantum algorithms natively operate on binary variables, a transformation from the MILP formulation to a Quadratic Unconstrained Binary Optimization (QUBO) model was performed. This conversion is essential for compatibility with current quantum optimization techniques.

Following the construction of the QUBO model, several quantum algorithms were developed and tested, including Variational Quantum Algorithms (VQAs) and the Quantum Approximate Optimization Algorithm (QAOA). These were initially validated using toy models—small-scale problem instances solvable by hand—to ensure algorithmic correctness. To enforce feasibility in the face of constraint violations, an Augmented Lagrangian Method was incorporated. This method iteratively penalized constraint violations, allowing the algorithms to explore feasible regions more effectively. Among the techniques applied, VQAs demonstrated a higher success rate in achieving optimal or near-optimal solutions in these toy examples.

Once validated, the algorithms were scaled to tackle larger problem instances in an effort to approach real-world applicability. Due to current hardware limitations in quantum computing—particularly with respect to the number of qubits and noise levels—a qubit compression encoding scheme was implemented. This method trades off some pre-

---

cision for scalability by reducing the quantum resource requirements. Problems involving up to 30 tasks were tested, yielding promising results, especially in instances with fewer constraints. However, as the number of constraints increased, the performance of the algorithms began to deteriorate, likely due to the difficulty in navigating the complex feasible space defined in the QUBO format.

Despite these challenges, the entire modeling approach was consistently guided by the principle of variable minimization, a critical factor given current hardware limitations. It is anticipated that as quantum hardware evolves—both in terms of qubit count and noise reduction—more complex formulations involving a larger number of variables can be explored, potentially improving both accuracy and performance.

To assess the practicality of the proposed methods, experiments were also conducted on real quantum hardware. The results obtained were encouraging: the solutions derived from real quantum processors closely approximated those from classical simulators, suggesting that noise, while present, did not critically impair solution quality. This reinforces the potential of quantum computing as a viable tool for tackling hard optimization problems.

In conclusion, while classical methods struggle with the computational demands of NP-hard problems like the JSSP, quantum computing offers a fundamentally different paradigm that may eventually scale more effectively. The results of this thesis serve as a preliminary step toward understanding the capabilities and limitations of current quantum approaches. Although scalability remains a significant hurdle, techniques such as qubit compression and advancements in quantum algorithm design have enabled us to demonstrate meaningful progress. With ongoing developments in quantum hardware and algorithmic strategies, future research can build on this work to further push the boundaries of quantum optimization in complex scheduling problems.

## Future work

There are several avenues for future exploration:

- **Improved Constraint Handling:** The Augmented Lagrangian method could be enhanced with adaptive penalty updates or hybridized with classical post-processing to better manage constraint satisfaction, because with this approach we notice that the problem can diverge from the feasible set too.
- **Alternative Encodings:** There are some alternative encoding schemes like a different Two-Body encoding schemes that could provide better solutions.
- **Incorporate more classical processing:** Using some classical tools and incor-

---

porating some classical pre-processing and post processing could help create better solutions.

- **Hybrid Quantum-Classical Methods:** Further development of hybrid algorithms combining classical solvers with quantum subroutines could enable solving larger problem instances more effectively.

Ultimately, this work contributes to the growing body of evidence suggesting that quantum computing, while still nascent, holds promise for addressing computationally intractable problems in operations research and beyond.

# Bibliography

- [1] Krzysztof Kurowski, Tomasz Pecyna, Mateusz Słysz, Rafał Różycki, Grzegorz Waligóra, Jan Weglarz, *Application of quantum approximate optimization algorithm to job shop scheduling problem*, European Journal of Operational Research, 2023.
- [2] Jozef Gruska *POWER OF QUANTUM ENTANGLEMENT*, Faculty of Informatics, Masaryk University.
- [3] Fred Glover, Gary Kochenberger, Yu Du, *Quantum Bridge Analytics I: A Tutorial on Formulating and Using QUBO Models*, University of Colorado
- [4] G. E. Moore, “Cramming More Components onto Integrated Circuits,” *Electronics*, vol. 38, no. 8, pp. 114–117, 1965.
- [5] R. P. Feynman, “Simulating Physics with Computers,” *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, 1982.
- [6] D. Deutsch, “Quantum Theory, the Church–Turing Principle and the Universal Quantum Computer,” *Proceedings of the Royal Society of London. Series A*, vol. 400, no. 1818, pp. 97–117, 1985.
- [7] F. Arute *et al.*, “Quantum Supremacy Using a Programmable Superconducting Processor,” *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
- [8] E. Farhi, J. Goldstone, and S. Gutmann, “A Quantum Approximate Optimization Algorithm,” arXiv:1411.4028 [quant-ph], 2014
- [9] B. Tan, M. A. Lemonde, S. Thanasilp, J. Tangpanitanon, and D. G. Angelakis, Qubit-efficient encoding schemes for binary optimisation problems, *Quantum*, vol. 5, p. 454, 2021.
- [10] I. D. Leonidas, A. Dukakis, B. Tan, and D. G. Angelakis, *Qubit efficient quantum algorithms for the vehicle routing problem on quantum computers of the NISQ era*, Adv. Quantum Technol., vol. 20, 2300309, 2024.



- [11] Maxime Dupont, Nicolas Didier, Mark J. Hodson, Joel E. Moore and Matthew J. Reagor, “Calibrating the Classical Hardness of the Quantum Approximate Optimization Algorithm,” University of California.
- [12] Maxime Dupont, Nicolas Didier, Mark J. Hodson, Joel E. Moore and Matthew J. Reagor, “An entanglement perspective on the quantum approximate optimization algorithm,” University of California.
- [13] D. Azses, M. Dupont, B. Evert, M. J. Reagor, and E. G. Dalla Torre, “Navigating the noise-depth tradeoff in adiabatic quantum circuits,” arXiv preprint arXiv:2209.11245 (2023).
- [14] M. Dupont, B. Evert, M. J. Hodson, B. Sundar, S. Jeffrey, Y. Yamaguchi, D. Feng, F. B. Maciejewski, S. Hadfield, M. S. Alam, Z. Wang, S. Grabbe, P. A. Lott, E. G. Rieffel, D. Venturelli, and M. J. Reagor, “Quantum-Enhanced Greedy Combinatorial Optimization Solver,” *Science Advances* **9**, eadi0487 (2023).
- [15] M. Dupont and B. Sundar, “Extending relax-and-round combinatorial optimization solvers with quantum correlations,” *Physical Review A* **109**, 012429 (2024).
- [16] F. B. Maciejewski, S. Hadfield, B. Hall, M. Hodson, M. Dupont, B. Evert, J. Sud, M. S. Alam, Z. Wang, S. Jeffrey, B. Sundar, P. A. Lott, S. Grabbe, E. G. Rieffel, M. J. Reagor, and D. Venturelli, “Design and execution of quantum circuits using tens of superconducting qubits and thousands of gates for dense Ising optimization problems,” arXiv preprint arXiv:2308.12423 (2023).
- [17] M. Dupont, B. Sundar, B. Evert, D. E. Bernal Neira, Z. Peng, S. Jeffrey, and M. J. Hodson, “Benchmarking Quantum Optimization for the Maximum-Cut Problem on a Superconducting Quantum Computer,” *Physical Review Applied* **23**, 014045 (2025).
- [18] B. Sundar and M. Dupont, “Qubit-efficient quantum combinatorial optimization solver,” arXiv preprint arXiv:2407.15539 (2024).
- [19] F. B. Maciejewski, B. G. Bach, M. Dupont, P. A. Lott, B. Sundar, D. E. Bernal Neira, I. Safro, and D. Venturelli, “A Multilevel Approach For Solving Large-Scale QUBO Problems With Noisy Hybrid Quantum Approximate Optimization,” *Proc. IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. [pages], (2024). *Note: Also available as arXiv preprint arXiv:2408.07793 (2024).*
- [20] W.-Y. Ku and J. C. Beck, “Mixed Integer Programming Models for Job Shop Scheduling: A Computational Analysis,” *Computers & Operations Research* **73**, 165–173 (2016).

- [21] K. Jun, “QUBO formulations for a system of linear equations,” *Results in Control and Optimization* **14**, 100380 (2024).