

CLASSICAL-QUANTUM HYBRID  
OPTIMIZATION: THE CASE OF JOB SHOP  
SCHEDULING PROBLEM

**Master's thesis**

submitted in partial fulfillment of the requirements  
for the award of the degree  
“Masters in Mathematics”

of the Georg-August-Universität Göttingen

Within the Mathematical Data Science track of the Masters program Mathematics

submitted by

**Dhiraj Kumar**

Göttingen, August 21, 2024

**First Supervisor:** Prof. Dr. David Russell Luke

**Second Supervisor:** Dr. Christian Boehme

---

## **Declaration of Independence**

I hereby declare that I have written this thesis independently and only with the use of the sources and aids indicated.

Göttingen, August 21, 2024

---

## Acknowledgements

This thesis would not have been possible without the help and support of my supervisors, Prof. Luke and Dr. Boehme. I would also like to thank them for their trust and encouragement in doing a thesis on a topic I had no prior experience with. I am also thankful to Aasish Kumar Sharma for taking time out of his busy schedule to help me at various stages of the thesis. A big thanks to my dear friend and colleague Lourens van Niekerk for being there for me on this journey. And finally, to my family and friends for their love and support.

---

## Abstract

The Job Shop Scheduling Problem (JSSP) is a combinatorial optimization problem that is one of the most challenging and important scheduling problems. Finding the optimal schedule that reduces the makespan necessitates the use of dedicated algorithms, local search techniques, or metaheuristics. The combinatorial nature of these approaches makes solving them computationally intractable in the classical settings. As a result, in this thesis, we use a novel technique based on variational quantum heuristics to the Job JSSP. We employ the time-indexed JSSP instance representation to create a cost Hamiltonian that can be used in the Quantum Approximate Optimization Algorithm (QAOA) to find the optimal solution to a simple JSSP instance. We employed a quantum-classical hybrid technique to execute the QAOA on the high performance computing system. We also had a look at the mathematical building blocks of QAOA namely Quantum Adiabatic Algorithm, Trotterization, and the Non-Negative matrices which are essential to solve JSSP on gate-based quantum computers.

## Table of contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Motivation . . . . .	7
1.2	Related Work . . . . .	8
1.3	Contribution . . . . .	9
<b>2</b>	<b>Background</b>	<b>10</b>
2.1	Classical Job Shop Scheduling . . . . .	10
2.2	Hilbert Spaces . . . . .	15
2.2.1	More than one Hilbert Space . . . . .	20
2.3	Quantum Mechanics Postulates . . . . .	23
2.4	Quantum Computing . . . . .	28
2.4.1	Quantum bits . . . . .	28
2.4.2	Quantum Gates . . . . .	30
2.4.3	Quantum Circuits . . . . .	32
2.5	Adiabatic Algorithm . . . . .	33
2.6	Irreducible Matrices . . . . .	36
2.7	Trotterization . . . . .	42
2.8	QUBO Problems . . . . .	45
2.9	Variational Quantum Algorithms . . . . .	46
2.10	QAOA . . . . .	48
<b>3</b>	<b>Quantum Job Shop Scheduling Problem</b>	<b>53</b>
3.1	Problem representation . . . . .	53
3.2	QAOA for the JSSP . . . . .	54
<b>4</b>	<b>Implementation and Result</b>	<b>55</b>
<b>5</b>	<b>Conclusion and Discussion</b>	<b>56</b>
<b>6</b>	<b>Appendix</b>	<b>57</b>

## Notation

$\mathbb{H}; {}^2\mathbb{H}$	Hilbert Space; two-dimensional Hilbert space
$\mathbb{H}^*$	Dual space of $\mathbb{H}$
$A$	Operator and its matrix
ONB	Orthonormal Basis
$e_j$	Basis vectors
$Eig(A, \lambda)$	Eigenspace
$tr(A)$	Trace
$\mathbb{H}^A \otimes \mathbb{H}^B$	Tensor product
$\mathcal{U}(\mathbb{H})$	Set of unitary operators on $\mathbb{H}$
$L(\mathbb{H})$	Set of linear operator on $\mathbb{H}$
$\sigma(A)$ and $\rho(A)$	Spectrum and Spectral radius of operator A
$P_\lambda; P$	projection onto eigenspace $Eig(A, \lambda)$ ; orthogonal projection
$B(\mathbb{H})$	Set of bounded operator on $\mathbb{H}$
$B_{sa}(\mathbb{H})$	Set of bounded self-adjoint operator on $\mathbb{H}$
$\langle A \rangle_\psi$	Expectation on $A$ in state $ \psi\rangle$
$\mathbb{P}; \mathbb{P}_\psi$	Probability; Probability to be quantum state $\psi$
$D(\mathbb{H})$	Set of density operators on $\mathbb{H}$
$M_n$	set of $n \times n$ complex matrices

## Abbreviation

COP	Combinatorial Optimization Problem
JSSP	Job Shop Scheduling Problem
QAOA	Quantum Approximate Optimization Algorithm
NISQ	Noisy Intermediate-Scale Quantum
QEC	Quantum error correction
HPC	High-performance computing
BQP	Bounded error, Quantum, Polynomial-time
VQE	Variational Quantum Eigensolver
AQC	Adiabatic Quantum Computation
QAT	Quantum Adiabatic Theorem
AA	Adiabatic Assumption
SC	Strongly Connected
QUBO	Quadratic Unconstrained Binary Optimization
VQA	Variational Quantum Algorithm

# 1 Introduction

## 1.1 Motivation

The theory of classical computation has been around for more than a century and has been a cornerstone in the development of classical computers. Classical computers have revolutionized our world and equipped humans with computation power that is beyond the capabilities of the human mind. This newly found computational power enabled, including other things, finding solutions to many quantitative problems, in particular, optimization problems. However, even with recent developments in parallel computing and graphic rendering, some optimization problems stayed out of the reach of classical computers. Quantum computers, on the other hand, promise to solve many optimization problems that provide an asymptotic advantage over classical computing.

In the early 1980s, quantum computers were conceptualized by Richard Feynman to simulate large quantum mechanical systems[1]. This motivated serious development in quantum computation, which led to the development of quantum algorithms, in particular, Deutsch algorithm [2], Grovers' algorithm [3], and Shor's algorithm [4]. These algorithms showed, at least theoretically, that quantum computers can solve some problems much faster than classical computers. However, two key challenges need to be solved: First, is it possible to create a quantum computer that can perform computation in large Hilbert spaces with low enough errors to materialize quantum advantage? Second, is it possible to formulate the problem one wants to solve that is difficult for a classical computer but easy for a quantum computer? Although fault-tolerant quantum computers are still in the future, recent development of quantum algorithms in various fields like Machine learning [5], drug discovery [6], material science and Chemistry [7], and combinatorial optimization[8] showed that it is possible to tackle both the problems posed above albeit with limited numbers of qubits and limited connectivity of the qubits.

In this thesis, we will focus on a type of Combinatorial Optimization Problem (COP) called Job Shop Scheduling Problem (JSSP)[9]. JSSP are NP-hard problems [10], [11] which means that finding an optimal solution on a classical computer is not computationally feasible. Therefore, JSSP is the perfect candidate to be evaluated using a quantum computer. In JSSP there are  $n$  jobs that needs to be scheduled on  $m$  machine with the objective to minimize the total length of the schedule.

To solve JSSP, we will use Quantum Approximate Optimization Algorithm (QAOA) [8]. QAOA is a quantum algorithm designed to solve COP on a Gate-based hybrid computer, i.e., it uses both classical and quantum computers for its computation. The Gate-based approach can be seen as the extension of the classical computation and does not need advance knowledge in Quantum Mechanics. Use of hybrid computer arises due to the fact that the quantum computer available today are Noisy Intermediate-Scale Quantum (NISQ) [12] devices. As the name suggests, NISQ devices are susceptible to noises like gate noise, decoherence noise, measurement noise, etc. Decoherence destroys the superpositions where the information is stored. If the devices are noisy, then quantum algorithms that can run are short before they run into errors. Quantum error correction (QEC) has to be employed to mitigate the noise. QEC is an active field of research, and we will not dwell on that in this thesis.

The objective of this thesis is to solve a relatively simple JSSP to show how to

formulate the problem such that it can be solved on High-performance computing (HPC) system. We are not using an actual quantum computer. With the rapid development in the field quantum computing, new error correction methods, and growing number of qubits, it is possible that near future that much larger problem will be able to solve on quantum computer.

The thesis is divided into following sections: 1. Classical Job shop scheduling (2.1) 2. Hilbert Space (2.2) 3. Quantum mechanics postulates (2.3) 4. Quantum computing (2.4) 5. Adiabatic Algorithm (2.5) 6. Irreducible matrices (2.6) 7. Trotterization (2.7) 8. Qubo problems (2.8) 9. Variational quantum algorithms (2.9) 10. QAOA (2.10) and 11. Quantum Job Shop scheduling (3)

## 1.2 Related Work

The general strategy to solve JSSP<sup>1</sup> consists of two parts: (a) a search strategy that enumerates the candidate solutions, and (b) a search space reduction strategy that reduces the number of possible candidate solutions. There are different formulations of JSSP possible. Some examples are single-machine scheduling and parallel-machine with preemptive or without preemptive scheduling. All these formulations, and others, are discussed in the survey [13]. As JSSP is an NP-hard problem, not all formulations have feasible solutions. Those that have feasible solutions can be represented by a disjunctive graph or graph matrix as discussed in [14].

As JSSP has applications in various fields like manufacturing, logistics, computer architecture etc, various heuristic and metaheuristic algorithms have been developed which are discussed in [15], and [16]. The exact methods are mostly based on the branch-and-bound method<sup>2</sup> which heavily relies on strong lower bounds to stop the branching as early as possible to make the procedure computationally feasible on classical computers. The lower bounds for JSSP have been analyzed in [17] and [18].

Many approximate algorithms have also been used that prioritize the order in which jobs have to be performed. A summary of many such appropriate algorithms can be found in [19]. Various approximate algorithms with sequence-dependent time setup have been analyzed in [20]. In JSSP, there has been many instances when the same job competes for the same machine. This means there will always be one or more machines that act as a *bottleneck*. To tackle such situations, the shifting bottleneck heuristic is used as discussed in [21], and [22].

The approach that focuses more on the reduction of search space for the possible candidate solutions is called the constraint propagation approach and it is discussed in [23]. Another approach is a local search algorithm that starts with a candidate solution and iteratively chooses the best candidate from the neighbouring set that minimizes or maximizes the objective function. For local search methods see the survey [24].

For a detailed analysis of many metaheuristic algorithms like simulated annealing, tabu search, and genetic algorithms see the books [25], [26], and [9]. There are hybrid approaches as well like a local search approach combined with shifting bottleneck [27] or simulated annealing with tabu search [28]. Benchmarking plays an important tool to compare the efficiency of various algorithms. In the context of

<sup>1</sup>or any combinatorial search and optimization problems

<sup>2</sup>discussed in more detail in the thesis

JSSP, a review can be found in [29].

Coming to the quantum algorithm, Deutsch-Jozsa algorithm is the first quantum algorithm that is exponentially faster than any classical algorithm [2], although it is of little practical use. Shor’s algorithm [4]<sup>3</sup>, and Grover’s algorithm [3]<sup>4</sup> were among the first quantum algorithms that had practical application. To study the complexity of quantum algorithms, a new complexity class called ”Bounded error, Quantum, Polynomial-time” (BQP) was introduced in [30]. BQP refers to decision problems that can be solved with a bounded probability of error using a polynomial-sized quantum circuit.

The interest in quantum computing arises mainly due to two reasons: first is the relation:  $P \subseteq BPP \subseteq BQP \subseteq PSPACE \subseteq EXP$  where  $PSPACE$  are decision problems solvable in polynomial space,  $BPP$  are problems that can be solved using a randomized algorithm in polynomial time if bounded by probability error and  $EXP$  are decision problems solvable in exponential time [31]. The second reason is the so-called Solovay Kitaev theorem [32] that states that an arbitrary single-qubit gate may be approximated to some accuracy using a poly-logarithmic number of gates from a predefined universal discrete set. As a result, quantum circuits can be efficiently constructed.

Quantum computation by adiabatic evolution was proved in [33] and solved satisfiability problems and other combinatorial search problems. Adiabatic computation is polynomially equivalent to conventional quantum computing in the quantum gate model [34]. Quantum Annealing [35], [36] solves optimization problems by formulating them in terms of finding the ground state of Ising spin Hamiltonian [37]. Quantum annealing devices like D-Wave can solve a few small-size optimization problems, including JSSP [38], [39].

In the current quantum computing era, called the NISQ era, problems that can be encoded using quantum logic gates can be solved on gate-based quantum computers. Two important algorithms are Variational Quantum Eigensolver (VQE) [40] and QAOA [8]. QAOA has already been applied to many well-known COPs like Max-Cut [41], [42], Travelling Salesman Problem [43], Graph Coloring [44], and JSSP [45]. In the original paper of QAOA, it is proved that the quality of the solution increases as the quantum circuit depth increases. However, due to the issues with the NISQ devices, it is harder to produce such results on NISQ devices. [46] shows that it is very likely that once we have real quantum devices that are less noisy, the quality of results will improve significantly.

A detailed review of QAOA and its variants is studied in [47]. In the original QAOA paper, and in this thesis, the qubits are initialized in the state  $|+\rangle$  state along the x-axis of the Bloch sphere and tensorised  $n$  times. This has been generalized to initialize with separable initial states, which is QAOA-warmest [48] achieves faster convergence.

### 1.3 Contribution

In this thesis, we studied QAOA and implemented it for JSSP. We also discussed the mathematical proof of the QAOA and the Peron-Frobenius theorem, the foundation

<sup>3</sup>finds the prime factorization in polynomial-time

<sup>4</sup>achieved quadratic speedup in an unordered database search

on which the Adiabatic algorithm is built. We saw the type of mathematical entity that satisfies the Peron-Frobenius theorem, i.e., irreducible matrices. During the research, numerous books and research were read and the following consists of the list of the main reference for the respective sections. There are many citations within each section as well. If a diagram or statement does not have a citation that means it is taken from the following references:

- The section on the classical job shop scheduling (2.1) relies on the books *Scheduling - Theory, Algorithms, and Systems* by Michael L. Pinedo ([26]) and *Principles of sequencing and scheduling* by Kenneth R. Baker and Dan Trietsch[9].
- The section on mathematical background of Quantum Computing (2.2), Quantum mechanics postulates (2.3), and adiabatic algorithm (2.5) follows the books *Mathematics of Quantum Computing* by Wolfgang Scherer[49]. Some definition and theorem used are almost verbatim like 2.25, 2.26, and 2.27.
- The section on quantum computing (2.4) relies on *Quantum Computation and Quantum Information*[50] and *Quantum Computing: An Applied Approach*[51].
- The section on Irreducible matrices (2.6) uses results from the books *Non-negative Matrices* by Henryk Minc[52], and *Matrix Analysis* by Horn and Johnson[53].
- The section on Trotterization (2.7) follows the book *Intermediate Spectral Theory and Quantum Dynamics* by C.R.D.Oliveira [54]. The definition and theorem used are almost verbatim. The explanation is own contribution.
- The section on QUBO (2.8) relies on the book *Quantum Machine Learning: What Quantum Computing Means to Data Mining* and the paper [55].
- The section on VQA (2.9) relies on [40] and [47].
- The section on QAOA (2.10) relies on [8] and [47].
- The section on Quantum Job Shop Scheduling (3) follows from [45],[56] and [57].

## 2 Background

### 2.1 Classical Job Shop Scheduling

Scheduling is a decision-making process that allocates resources to optimize some pre-defined objective function. Scheduling problems, in general, consist of a finite number of **jobs** and **machines**.  $n$  denotes the number of jobs and  $m$  denotes the number of machines. If a job requires more than one processing step, or **operations**<sup>5</sup>, then  $(i, j)$  denotes the operation of job  $j$  on machine  $i$ . Each job has some data associated with it. In this thesis, we will focus on the processing time<sup>6</sup>  $p_{ij}$  of

<sup>5</sup>we will use task and operations interchangeably

<sup>6</sup>Other possibilities include Release date, Due date, and Priority factor

the jobs, where  $p_{ij}$  represents processing time of job  $j$  on machine  $i$ .

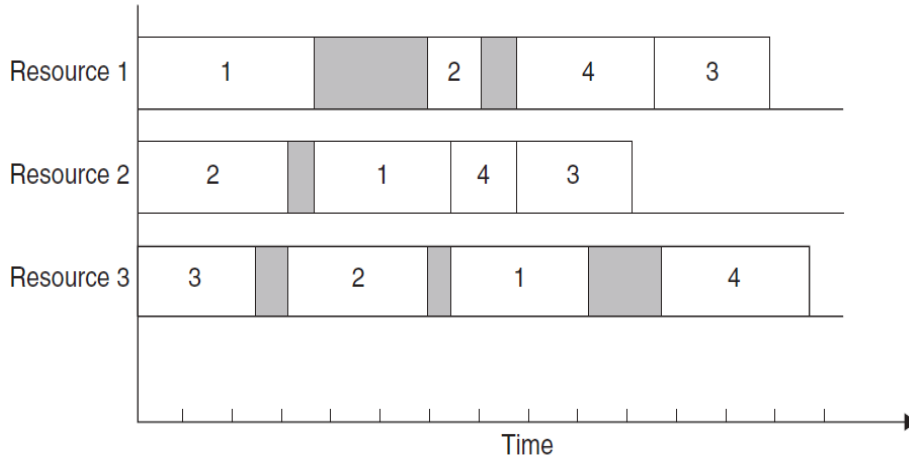


Figure 1: Gantt Chart is used to visualize and compare schedules.

The triplet  $\alpha|\beta|\gamma$  describes a scheduling problems. The  $\alpha$  represents the machine environment and has only one value. The  $\beta$  represents the constraints and can have no, single or several values. The  $\gamma$  represents the objective that needs to be minimized/maximize and has a single/multi value. Different variations of scheduling problems can be formed simply by varying the values in the triplet.

In this thesis, the  $\alpha$  will be Job Shop. Job shops are scheduling problems in which a set of different machines perform tasks of jobs. Each job is composed of an ordered list of tasks, from which every task requires a specific machine for a known processing time.

The  $\beta$  of Job shop considered in the thesis are defined below<sup>7</sup> [45]:

- i. Tasks are nonpreemptable, i.e., once started, it is necessary to keep the task on a machine until its completion.
- ii. Tasks of different jobs are independent,
- iii. Each task can be performed on a specific machine at a time,
- iv. Each machine can process only one job at a time.

The objective is to minimize the **makespan** ( $C_{max}$ ), i.e. the maximum completion time of all tasks. The makespan is the  $\gamma$  of the problem formulation.

JSSP belongs to a class of optimization problems called the COP. In COP, one searches for an optimum (combination of) object in a finite collection of objects [58]. The optimal object is called *optimal solution* whereas all the other objects in the finite collection are called *feasible solution*. In the context of JSSP with makespan as objective, every schedule that satisfies the constraints is a feasible solution and the schedule that has the minimum makespan is the optimal solution. As the number of jobs and machines increases, the set of feasible solutions increases exponentially,

<sup>7</sup>For a comprehensive list, see the reference.

making JSSP NP-hard problems [10], [11]. For instance, a (20, 10) job shop may have at most  $7.2651 \times 10^{183}$  possible solutions. Therefore, there is no standard procedure available to find the optimal solution. COPs are typically represented as a graph [58], and so is JSSP as discussed below. This lead to the disjunctive programming formulation of the JSSP with makespan objective and also describe a branch-and-bound algorithm to find the optimal solution which we discuss below.

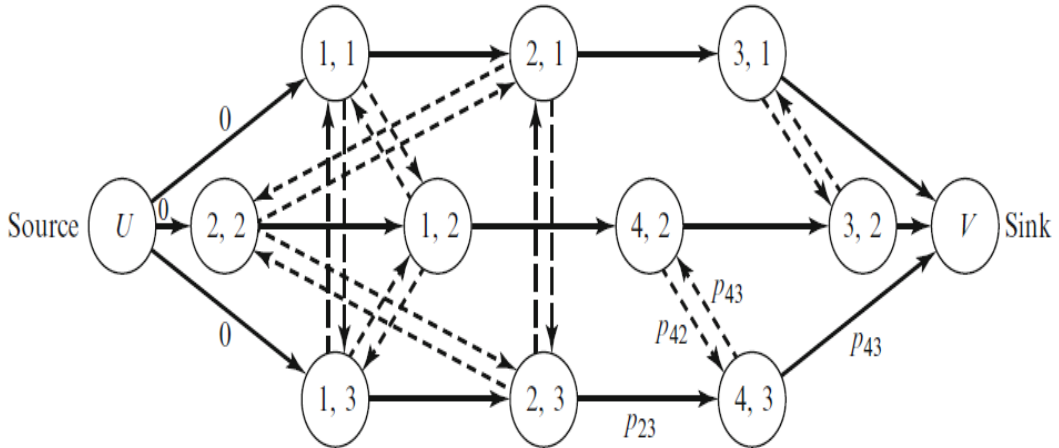


Figure 2: Directed graph for job shop with makespan as objective.

A disjunctive graph can be used to describe the minimization of the makespan in a job shop. Consider a directed graph  $G = (N, A, B)$  where  $N$  is the set of the graph nodes or operations  $(i, j)$  that must be performed on  $n$  jobs,  $A$  is a set of conjunctive arcs, and  $B$  is the set of disjunctive arcs. The arc  $(i, j) \rightarrow (k, j)$  forms a **conjunctive** (solid) arc if job  $j$  is processed on machine  $i$  before it is processed on machine  $k$ . Two operations that have to be processed on the same machine but belong to two different jobs are connected by **disjunctive** (broken) arcs. Disjunctive arcs go in opposite directions and form  $m$  cliques<sup>8</sup> of double arcs, one clique for each machine. All operations in the same clique have to be done on the same machine. The length of all the arcs represents the processing time. In addition, there are two dummy nodes called source  $U$  and sink  $V$ .  $U$  has  $n$  conjunctive arcs emanating from the first operations and  $V$  has  $n$  conjunctive arcs coming in from all the last operations. The arcs emanating from the source have a length of zero.

A **feasible** schedule selects one disjunctive arc from each pair, resulting in an acyclic-directed graph. This selection defines the sequence in which the operations are performed on that machine. Now, we will show that a clique selection must be acyclic: If there were a cycle within a clique, a feasible sequence of the operations on the corresponding machine would be impossible as such a cycle would correspond to an infeasible situation. For example, let  $(h, j)$  and  $(i, j)$  denote two consecutive operations belonging to job  $j$  and let  $(i, k)$  and  $(h, k)$  denote two consecutive operations belonging to job  $k$ . If under a given schedule, operation  $(i, j)$  precedes operation  $(i, k)$  on machine  $i$  and operation  $(h, k)$  precedes operation  $(h, j)$  on machine  $h$ , then the graph contains a cycle with four arcs, two conjunctive arcs and two disjunctive arcs from different cliques. Such a schedule is physically impossible.

<sup>8</sup>A clique refers to a graph in which any two nodes are connected; in this case, each connection within a clique consists of a pair of disjunctive arcs.

The longest path from  $U$  to  $V$  determines the makespan of a feasible schedule. This longest path consists of operations where the first starts at time 0 and the last finishes at the time of the final operation and satisfies the constraints defined in  $\beta$ . To minimize the makespan, select disjunctive arcs that minimize the length of the longest path which is called the *critical path*.

To present the disjunctive programming formulation, let  $y_{ij}$  denote the starting time of operation  $(i, j)$  and  $G = (N, A, B)$  be a directed graph as defined above. Then the following mathematical program minimizes the makespan:

$$\text{minimize } C_{max}$$

subject to

$$\begin{aligned} y_{kj} - y_{ij} &\geq p_{ij} && \forall (i, j) \rightarrow (k, j) \in A \\ C_{max} - y_{ij} &\geq p_{ij} && \forall (i, j) \in N \\ y_{ij} - y_{il} \geq p_{il} \text{ or } y_{il} - y_{ij} &\geq p_{ij} && \forall (i, l) \text{ and } (i, j), \quad i = 1, \dots, m \\ y_{ij} &\geq 0 && \forall (i, j) \in N \end{aligned}$$

Here, the first constraint assures that operation  $(k, j)$  cannot begin until operation  $(i, j)$  is finished. The third constraint, *disjunctive constraints*, guarantees that there is some ordering among operations of various jobs that must be executed on the same machine.

To obtain optimal solutions through the branch-and-bound methods, we consider a certain type of schedule.

#### Definition 2.1: Active Schedule

A feasible schedule is called **active** if it cannot be altered in any way such that some operation is completed earlier and no other operation is completed later.

It also follows from the definition that it is impossible to reduce the makespan of an active schedule without increasing the starting time of some operations. Therefore, an active schedule cannot have any idle time in which one or more operations of a job in queue could fit.

An operation becomes **schedulable** when all of its predecessors are scheduled. The order of precedence of the operations within the jobs determines how the next operations are decided. A **partial schedule** consists of operations with set beginning times. Given a partial schedule for each job shop situation, we may generate a distinct set of schedulable operations. Let

$PS(k)$  = a partial schedule containing  $k$  scheduled operations

$SO(k)$  = the set of schedulable operations at stage  $k$ , corresponding to a given  $PS(k)$

$s_j$  = the earliest time at which operation  $j \in SO(k)$  could be started

$f_j$  = the earliest time at which operation  $j \in SO(k)$  could be finished

For a given active partial schedule, the possible start time,  $s_j$ , for schedulable operation  $j$ , is determined by the completion time of the immediate predecessor of operation  $j$  and the latest completion time required by operation  $j$  on the machine. The larger of the two is  $s_j$ . The possible finish time,  $f_j$ , is  $s_j + p_j$ , where  $p_j$  represents processing time of operation  $j$ .

A systematic approach to generating active schedules works as follows.

**Algorithm 1: Generation of all Active Schedules.**

- Step 1. Let  $k = 0$  and begin with  $PS(k)$  as the null. Initially,  $SO(k)$  includes all operations with no predecessors.
- Step 2. Determine  $f^* = \min_{j \in SO(k)} \{f_j\}$  and the machine  $m^*$  that can realize  $f^*$ .
- Step 3. For each operation  $j \in SO(k)$  that requires machine  $m^*$  and for which  $s_j < f^*$ , create a new partial schedule in which operation  $j$  is added to  $PS(k)$  and started at time  $s_j$ .
- Step 4. For each new partial schedule created in Step 3, update the data set as follows:
- a) Remove operation  $j$  from  $SO(k)$ .
  - b) Create  $SO(k+1)$  by adding the direct successor of  $j$  to  $SO(k)$ .
  - c) Increase  $k$  by one.
- Step 5. Return to Step 2 for each partial schedule created in Step 3 and updated in Step 4, repeating until all active schedules have been generated.

The Algorithm 1 demonstrates a tree-structured method to schedule generation. The nodes in the tree represent partial schedules, and when a new operation is added to a partial schedule, the algorithm proceeds from one level of the tree to the next. If we create the tree in its entirety, it will list all active schedules. The enumeration tree could be the basis for an optimum-seeking approach using branch and bound. Even in a moderate-sized job shop problems, the computational cost of an enumeration tree by the branch-and-bound method is relatively high. The way to solve the computational problem is to use a suboptimal approach that generate one complete schedule. For this, one need a way to resolve conflicts between different partial schedule generated at step 3 of the algorithm 1, i.e., one bounds the growth of branching. The way to do it is through priority rule  $R$ . We will re-define the step 3 of the algorithm with priority rule:

**Algorithm 2: Generation of all Active Schedules.**

- Step 3. Calculate a priority index for each operation  $j \in SO(k)$  that requires machine  $m^*$  and has  $s_j < f^*$  using a specific priority rule. Find the operation with the smallest index and add it to  $PS(k)$  as soon as possible, creating only one partial schedule,  $PS(k+1)$ , for the following stage.

The following are some commonly used priority rules:

- Shortest processing time: select the next operation with least processing time.

- First come first served: select operation that comes first.
- Most work remaining: select the operation associated with a job that has most processes remaining.
- Least work remaining: select the operation associated with a job that has least processes remaining.

Which priority rule should be used depends on the objective and there is no one-fit-all solution.

JSSP has many heuristic algorithms as well. Some examples include Tabu search [59], Genetic algorithm [60], Simulated Annealing [61]. However, all these algorithms are computationally very expansive as well. Therefore, there is a growing interest in quantum computers as they have the potential to solve some classically intractable problems. QAOA is one such quantum algorithm that shows potential and is the focus of this thesis.

To understand and appreciate the quantum computer we need a mathematical background which is the objective of the remaining of this section.

## 2.2 Hilbert Spaces

### Definition 2.2: Inner Product Space

Let  $E$  be a complex vector space. A mapping  $\langle \cdot | \cdot \rangle : E \times E \rightarrow \mathbb{C}$  is called an **inner product** in  $E$  if for any  $x, y, z \in E$  and  $\alpha, \beta \in \mathbb{C}$  the following holds:

- $\langle x | y \rangle = \overline{\langle y | x \rangle}$ ;
- $\langle \alpha x + \beta y | z \rangle = \alpha \langle x | z \rangle + \beta \langle y | z \rangle$ ;
- $\langle x | x \rangle \geq 0$ ;
- $\langle x | x \rangle = 0$  implies  $x = 0$ .

A vector space with an inner product is called an **inner product space**.

This inner product induces a **norm**  $\|\cdot\|$  defined as  $\|x\| = \sqrt{\langle x | x \rangle}$  for  $x \in E$ .

An inner product space  $E$  is called **complete** if every Cauchy sequence in  $E$  converges to an element of  $E$  i.e. if for every  $\epsilon > 0 \exists M \in \mathbb{N}$  such that  $\|x_m - x_n\| < \epsilon \forall m, n > M$ .

### Definition 2.3: Hilbert Space

A complete inner product space is called a **Hilbert Space**  $\mathbb{H}$ .

### Definition 2.4

Two vectors  $\phi, \psi \in \mathbb{H}$  are called **orthogonal** if  $\langle \phi | \psi \rangle = 0$ .

Set  $\{\varphi_j \mid j \in I\} \subset \mathbb{H}$ :

- i. is **linearly independent** if for every finite subset  $\{\varphi_k\}$  and  $a_k \in \mathbb{C}$  with  $k = 1, 2, \dots, n$  the following

$$a_1\varphi_1 + a_2\varphi_2 + \dots + a_n\varphi_n = 0$$

holds only if  $a_1 = a_2 = \dots = a_n = 0$ .

- ii. is said to **span**  $\mathbb{H}$  if for every vector  $\varphi \in \mathbb{H}$  there are  $a_j \in \mathbb{C}$  with  $j \in I$  such that

$$\varphi = \sum_{j \in I} a_j \varphi_j$$

Set of vectors  $\{\varphi_j \mid j \in I\}$  that are linearly independent and spans  $\mathbb{H}$  forms the **basis** of  $\mathbb{H}$ .

In addition, if basis vectors  $\{e_j \mid j \in I\} \subset \mathbb{H}$  satisfy:

$$\langle e_j | e_k \rangle = \delta_{ij} := \begin{cases} 0, & \text{if } j \neq k \\ 1, & \text{if } j = k, \end{cases}$$

then they are called **orthonormal basis (ONB)**.

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

A Hilbert space  $\mathbb{H}$  is called **finite-dimensional** if it has finite number of basis vectors; otherwise  $\mathbb{H}$  is called **infinite-dimensional**.

Quantum computational process generally observes and manipulates observables such as spin or photon polarization. Therefore, it is sufficient to consider only finite-dimensional Hilbert space in quantum computing.

Let  $\{e_j\}$  be the ONB of  $\mathbb{H}$ . Then every vector  $\psi \in \mathbb{H}$  can be uniquely expressed as **basis expansion** of  $\psi$  as

$$\psi = \sum_i a_j e_j$$

If  $\psi = (a_1, \dots, a_n)$ , then we have  $a_j = \langle e_j | \psi \rangle$ . Furthermore,

$$\psi = \sum_j \langle e_j | \psi \rangle e_j \quad (1)$$

and

$$\|\psi\|^2 = \sum_j \langle e_j | \psi \rangle^2 \quad (2)$$

### Definition 2.5: Dual Space

The **dual space** of  $\mathbb{H}$  a vector space defined as:

$$\mathbb{H}^* := \{f : \mathbb{H} \rightarrow \mathbb{C} \mid f \text{ linear and continuous}\}.$$

Due to Riesz Representation Theorem [62] there exists a bijection between  $\mathbb{H}$  and its *dual space*  $\mathbb{H}^*$ . This bijection leads to the *bra* and *ket* notation first introduced

by Dirac. The **bra-vectors**  $\langle\psi| \in \mathbb{H}^*$  whereas **ket-vectors**  $|\varphi\rangle \in \mathbb{H}$ . Notice that  $\langle\varphi|\psi\rangle \in \mathbb{C}$ . Therefore, (1) can be re-written as:

$$|\psi\rangle = \sum_i |e_j\rangle \langle e_j|\psi\rangle \quad (3)$$

A linear map  $A: \mathbb{H} \rightarrow \mathbb{H}$  is called **operator**. The set of all operators is denoted by  $L(\mathbb{H})$ .

This leads to the following definition.

**Definition 2.6: Matrix Representation of Operator**

Let  $A$  be an operator on  $\mathbb{H}$  and  $\{|e_j\rangle\}$  be the ONB of  $\mathbb{H}$ . Then the  $(j, k)$  element of  $A$  in the basis  $\{|e_j\rangle\}$  is written as

$$A_{jk} := \langle e_j|Ae_k\rangle \quad (4)$$

The matrix  $(A_{jk})_{j,k=1,\dots,\dim(\mathbb{H})}$  is called the **matrix representation** of  $A$  in the basis  $\{|e_j\rangle\}$ .

Furthermore,  $A$  can be expressed as

$$A = \sum_{j,k} |e_j\rangle A_{jk} \langle e_k| \quad (5)$$

From here on,  $A$  will be used to denote both the operator and its matrix.

**Example 1**

Let  $\mathbb{H}$  be  $n$ -dimensional Hilbert Space and  $\{|e_j\rangle\}$  be its ONB. Then we can use the isomorphism  $\mathbb{H} \cong \mathbb{C}^n$  to see the standard basis in  $\mathbb{C}^n$ ,

$$|e_1\rangle = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \dots, |e_n\rangle = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Similarly, for the dual basis

$$\langle e_1| = [1 \ 0 \ \dots \ 0], \dots, \langle e_n| = [0 \ 0 \ \dots \ 1]$$

**Definition 2.7**

An operator  $A^*$  is called **adjoint** operator to  $A$  on  $\mathbb{H}$  if

$$\langle A^*\psi|\varphi\rangle = \langle\psi|A\varphi\rangle \quad \forall |\psi\rangle, |\varphi\rangle \in \mathbb{H} \quad (6)$$

If  $A^* = A$  then  $A$  is called **self-adjoint**.

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

$$\langle U\psi|U\varphi\rangle = \langle\psi|\varphi\rangle \quad \forall |\psi\rangle, |\varphi\rangle \in \mathbb{H} \quad (7)$$

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

An interesting property about the unitary operators is that their adjoint operator are their inverse and they do not change the norm.

### Definition 2.8: Spectrum

Let  $A \in L(\mathbb{H})$ . Vector  $|\psi\rangle \in \mathbb{H} \setminus \{0\}$  is called **eigenvector** of  $A$  if

$$A|\psi\rangle = \lambda|\psi\rangle.$$

and  $\lambda \in \mathbb{C}$  is called **eigenvalue**.

The set of all eigenvectors of  $A$  for a given eigenvalue  $\lambda$  of, along with zero vector is called **eigenspace** of  $\lambda$  and denoted by  $\text{Eig}(A, \lambda)$ .

An eigenvalue  $\lambda$  is called **non-degenerate** if  $\text{Eig}(A, \lambda)$  is one-dimensional. Otherwise,  $\lambda$  is called **degenerate**.

The set

$$\sigma(A) := \{\lambda \in \mathbb{C} \mid (A - \lambda \mathbf{1})^{-1} \text{ does not exist}\}$$

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

The **spectral radius** of  $A$  is defined as

$$\rho(A) = \sup\{|\lambda| : \lambda \in \sigma(A)\}$$

### Theorem 2.9

For  $A \in L(\mathbb{H})$ , let  $A|\psi\rangle = \lambda|\psi\rangle$ . Then

- (i) if  $A$  is self-adjoint, it has real eigenvalues.
- (ii) If  $A$  is unitary, it has eigenvalues with absolute value 1.

*Proof.* Let  $|\psi\rangle \in \mathbb{H}$  with  $|\psi\rangle \neq 0$ .

- (i) Let  $A$  be self-adjoint. Then

$$\lambda \langle \psi | \psi \rangle = \langle \psi | \lambda \psi \rangle = \langle \psi | A \psi \rangle = \langle A^* \psi | \psi \rangle = \langle A \psi | \psi \rangle = \langle \lambda \psi | \psi \rangle = \bar{\lambda} \langle \psi | \psi \rangle$$

where  $\bar{\lambda}$  is complex conjugate of  $\lambda$ . Therefore,  $\lambda$  is real.

- (ii) Let  $A$  be unitary. Then

$$\|\psi\| = \|A\psi\| = \|\lambda\psi\| = |\lambda| \|\psi\|$$

Therefore,  $|\lambda| = 1$ .

□

Self-adjoint operators are **diagonalizable**, that is, for every self-adjoint operator  $A$  there is an ONB  $\{|e_{j,\alpha}\rangle\}$  of  $A$  such that

$$A|e_{j,\alpha}\rangle = \lambda_j |e_{j,\alpha}\rangle.$$

This allows us to write self-adjoint operators in the **diagonal form** as follows

$$A = \sum_{j,\alpha} \lambda_j |e_{j,\alpha}\rangle \langle e_{j,\alpha}|. \quad (8)$$

**Definition 2.10: Projection**

An operator  $P \in L(\mathbb{H})$  satisfying  $P^2 = P$  is called a *projection*. If  $P^* = P$ , then  $P$  is called *orthogonal projection*.

**Theorem 2.11**

Let  $A$  be a self-adjoint operator as defined in 8. Let  $|e_{j,\alpha}\rangle$  with  $\alpha \in \{1, \dots, d_j\}$  be orthonormal eigenvectors of  $A$  where  $\alpha$  denotes the  $d_j$ -fold degeneracy of eigenvalues  $\lambda_j$ . Then

$$P_j = \sum_{\alpha=1}^{d_j} |e_{j,\alpha}\rangle \langle e_{j,\alpha}|$$

is the orthogonal projection onto the eigenspace  $Eig(A, \lambda_j)$  for the eigenvalue  $\lambda_j$ .

*Proof.* Using the definition of orthogonal projection, we have

$$\begin{aligned} P_j^* &= \left( \sum_{\alpha=1}^{d_j} |e_{j,\alpha}\rangle \langle e_{j,\alpha}| \right)^* = \sum_{\alpha=1}^{d_j} |e_{j,\alpha}\rangle \langle e_{j,\alpha}|^* \\ &= \sum_{\alpha=1}^{d_j} |e_{j,\alpha}\rangle \langle e_{j,\alpha}| \text{ since } |e_{j,\alpha}\rangle \text{ are eigenvectors of self-adjoint } A \\ &= P_j \end{aligned}$$

As  $|e_{j,\alpha}\rangle$  with  $\alpha \in \{1, \dots, d_j\}$  are ONB of eigenvectors of  $A$ , any eigenvector  $|\psi\rangle \in Eig(A, \lambda_j)$  can be written as

$$|\psi\rangle = \sum_{\alpha=1}^{d_j} |e_{j,\alpha}\rangle \langle e_{j,\alpha}|\psi\rangle = P_j |\psi\rangle$$

which shows that  $P_j$  is a projection onto  $Eig(A, \lambda_j)$ .  $\square$

Using the above, we can write any self-adjoint  $A$  in the form

$$A = \sum_{j,\alpha} \lambda_j |e_{j,\alpha}\rangle \langle e_{j,\alpha}| = \sum_j \lambda_j P_j$$

Further, for a  $|\psi\rangle$  with  $\|\psi\| = 1$ , the **orthogonal projection onto**  $|\psi\rangle$  is defined as

$$P_\psi := |\psi\rangle \langle \psi|.$$

**Definition 2.12: Norm of Operator**

$A \in L(\mathbb{H})$  is called a **bounded** if

$$\|A\| := \sup\{\|A\psi\| \mid |\psi\rangle \in \mathbb{H} \text{ and } \|\psi\| = 1\} < \infty$$

If  $\|A\|$  is defined then  $\|A\|$  is called **norm of the operator**  $A$ .

$B(\mathbb{H})$  denotes the set of all bounded operators on  $\mathbb{H}$ .  
 $B_{sa}(\mathbb{H})$  denotes the set of all bounded self-adjoint operators on  $\mathbb{H}$ .

### Definition 2.13: Commutator

An operator  $A \in B_{sa}(\mathbb{H})$  is called **positive** if  $\forall |\psi\rangle \in \mathbb{H}$

$$\langle \psi | A \psi \rangle \geq 0,$$

and written as  $A \geq 0$ . If the inequality is strict then it is called **strictly positive**.

Furthermore, **commutator** of two operators  $A$  and  $B \in L(\mathbb{H})$  is defined as

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

$A$  and  $B$  are said to **commute** if their commutator vanishes,  $[A, B] = 0$ .

### Definition 2.14: Trace

Let  $|e_j\rangle$  be ONB in  $\mathbb{H}$ . The **trace** of  $A$  is defined as

$$tr(A) := \sum_j \langle e_j | A e_j \rangle \stackrel{4}{=} \sum_j A_{jj} \quad (9)$$

The important property of the trace of a matrix is that it does not depend on the basis.

## 2.2.1 More than one Hilbert Space

Until now we discussed only one Hilbert Space which is enough to describe one qubit system which can be described in a two-dimensional Hilbert space. However, to represent more than one qubit system and mathematically define how it interacts, one needs more than one Hilbert Space.

Let  $\mathbb{H}^A$  and  $\mathbb{H}^B$  be two Hilbert spaces defined on complex vector space  $A$  and  $B$ , respectively. Then their **Tensor Product**  $\mathbb{H}^A \otimes \mathbb{H}^B$  is a linear combination of vectors of the form  $|\varphi\rangle \otimes |\psi\rangle$  where  $|\varphi\rangle \in \mathbb{H}^A$  and  $|\psi\rangle \in \mathbb{H}^B$ . That is

$$|\varphi\rangle \otimes |\psi\rangle : \mathbb{H}^A \times \mathbb{H}^B \rightarrow \mathbb{C} \quad (10)$$

The tensor product is a vector space over  $\mathbb{C}$ . To see that let  $\Psi_1, \Psi_2 \in \mathbb{H}^A \otimes \mathbb{H}^B$ , then

$$(a\Psi_1 + b\Psi_2)(\xi, \eta) := a\Psi_1(\xi, \eta) + b\Psi_2(\xi, \eta) \in \mathbb{H}^A \otimes \mathbb{H}^B$$

The null-map is the null-vector, and  $-\Psi$  is the additive inverse of  $\Psi$ .

Every vector  $|\Psi\rangle \in \mathbb{H}^A \otimes \mathbb{H}^B$  can be written as the linear combination of the ONBs  $\{|e_a\rangle\} \subset \mathbb{H}^A$  and  $\{|f_b\rangle\} \subset \mathbb{H}^B$  as follows

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

Now we can define the **inner product** on the tensor product

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

Similarly, the **norm** of  $|\Psi\rangle \in \mathbb{H}^A \otimes \mathbb{H}^B$  is defined as

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

and for any  $|\psi\rangle \in \mathbb{H}^A$  and  $|\phi\rangle \in \mathbb{H}^B$ ,

$$\|\psi \otimes \phi\| = \sqrt{\langle \psi \otimes \phi | \psi \otimes \phi \rangle} = \sqrt{\langle \psi | \psi \rangle \langle \phi | \phi \rangle} = \|\psi\| \|\phi\|$$

In the finite-dimension,  $\mathbb{H}^A \otimes \mathbb{H}^B$  is complete with the norm defined above and thus a Hilbert space.

In quantum computing, it is sufficient to consider Hilbert spaces that are defined on two-dimensional complex space. Therefore, we will use the notation  ${}^2\mathbb{H}$  to define such Hilbert spaces.

### Example 2

Let  $\mathbb{H}^A = \mathbb{H}^B = {}^2\mathbb{H} \cong \mathbb{C}^2$  with ONBs

$$\{|e_a\rangle\} = \{|f_b\rangle\} = \{|0\rangle, |1\rangle\} = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

Then  $\mathbb{H}^A \otimes \mathbb{H}^B \cong \mathbb{C}^4$  is

$$\{|e_a \otimes f_b\rangle\} = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

In general, for  $a_j, b_j \in \mathbb{C}$  and

$$|\phi_1\rangle = a_1 |0\rangle + b_1 |1\rangle = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix}, \quad |\phi_2\rangle = a_2 |0\rangle + b_2 |1\rangle = \begin{bmatrix} a_2 \\ b_2 \end{bmatrix}$$

we have,

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

In quantum computing, the tensor product for the  $n$  2-dimensional case can be defined as follows:

$${}^2\mathbb{H}^{\otimes n} := \underbrace{{}^2\mathbb{H} \otimes \dots \otimes {}^2\mathbb{H}}_{n \text{ times}} \quad (11)$$

The Hilbert Space  ${}^2\mathbb{H}^{\otimes n}$  is  $2^n$ -dimensional.

In quantum computing, it is conventional to start the counting from the right side. Therefore, the  $j+1$ -th factor in  ${}^2\mathbb{H}^{\otimes n}$  is  ${}^2\mathbb{H}_j$ . That is,

$${}^2\mathbb{H}^{\otimes n} = {}^2\mathbb{H}_{n-1} \otimes \dots \otimes \underbrace{{}^2\mathbb{H}_j}_{j+1 \text{ factor}} \otimes \dots \otimes {}^2\mathbb{H}_0$$

Further convention is to write every vector  $|x\rangle \in {}^2\mathbb{H}^{\otimes n}$  in its binary representation.

Let  $x \in \mathbb{N}_0$  with  $x < 2^n$ . Then the binary representation of  $x$  is written as

$$x = \sum_{j=0}^{n-1} x_j 2^j$$

where  $x_0, \dots, x_{n-1} \in \{0, 1\}$ . Then every vector  $|x\rangle \in {}^2\mathbb{H}^{\otimes n}$  is defined as

$$\begin{aligned} |x\rangle^n &:= |x\rangle := |x_{n-1} \dots x_1 x_0\rangle \\ &:= |x_{n-1}\rangle \otimes \dots \otimes |x_1\rangle \otimes |x_0\rangle = \bigotimes_{j=n-1}^0 |x_j\rangle \end{aligned}$$

We can represent the first and the last state vector in  ${}^2\mathbb{H}^{\otimes n}$  as

$$\begin{aligned} |0\rangle^n &= |00 \dots 0\rangle = \bigotimes_{j=0}^{n-1} |0\rangle \\ |2^n - 1\rangle^n &= |11 \dots 1\rangle = \bigotimes_{j=0}^{n-1} |1\rangle \end{aligned}$$

### Definition 2.15

The ONB in  ${}^2\mathbb{H}^{\otimes n}$  defined by  $|x\rangle = |x_{n-1}, \dots, x_0\rangle$  for  $x \in \{0, 1, \dots, 2^n - 1\}$  is called the **computational basis**.

In  ${}^2\mathbb{H}^{\otimes n}$  the computational basis is identical to the standard basis in  $\mathbb{C}^{2^n}$ .

The computational basis consists of **separable or product-states** (see 2.18). This is because each sub-system of a composite system (see 7) is in a pure state. For example, in example 2, if the system is in the state  $|10\rangle$ , then the first sub-system is in the pure state  $|1\rangle$ . The measurement of  $\sigma_z$  in his sub-system will result in  $-1$ . Similarly, the second sub-system is in the pure state  $|0\rangle$  and the measurement of  $\sigma_z$  in his sub-system will result in  $+1$ .

Now we see an example of another possible basis vectors

**Example 3**

**Bell basis** in  ${}^2\mathbb{H}^{\otimes 2}$  consists of

$$\begin{aligned} |\Phi^\pm\rangle &:= \frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle) \\ |\Psi^\pm\rangle &:= \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle) \end{aligned}$$

**2.3 Quantum Mechanics Postulates**

Quantum mechanics allows possible descriptions of the statistics of the quantum system. By **quantum system**, we mean any physical entity, such as atoms, electrons, or molecules, that exhibits quantum mechanical effects like superposition and entanglement<sup>9</sup>. There physical quantities are called **observable** that are associated with these physical entities. A **quantum state** contains all the necessary information to describe and to make predictions, upon measurement, about a quantum system. Due to the probabilistic nature of quantum systems, the mean value of an observable is usually considered.

Mathematically, states can either be described by a vector or an operator. This depends on whether the state is a pure state or a mixed state. We will describe these states as well as how to change and measure them in terms of Postulates.

**Postulate 1: (Observable and Pure States)**

An **observable**  $A$  on  $\mathbb{H}$  is represented by a self-adjoint operator.

If the mean value of  $A$  can be calculated using a **state vector**  $|\psi\rangle \in \mathbb{H}$  satisfying  $\|\psi\| = 1$  as

$$\langle A \rangle_\psi := \langle \psi | A \psi \rangle \quad (12)$$

then the quantum system is described by a **pure state**.

The mean value of the observable  $A$  using the diagonal form (8):

$$\begin{aligned} \langle A \rangle_\psi &\stackrel{12}{=} \langle \psi | A \psi \rangle \\ &\stackrel{8}{=} \langle \psi | \sum_j \lambda_j |e_j\rangle \langle e_j| \psi \rangle \\ &= \sum_j \lambda_j \langle \psi | e_j \rangle \langle e_j | \psi \rangle \\ &= \sum_j \lambda_j |\langle \psi | e_j \rangle|^2. \end{aligned}$$

These eigenvalues  $\lambda_j$  are the possible measurement results associated with  $A$  as stated in the following postulate.

<sup>9</sup>We will describe these quantum mechanical effects later in more detail.

**Postulate 2: (Measurement Probability)**

For an observable  $A$ , spectrum  $\sigma(A)$  contains all the possible measurement values.

For an observable  $A$  described by pure state  $|\psi\rangle \in \mathbb{H}$ , the probability of obtaining eigenvalue  $\lambda \in \sigma(A)$  is given by the projection  $P_\lambda$  onto the eigenspace  $Eig(A, \lambda)$ .

$$\mathbb{P}_\psi(\lambda) := \|P_\lambda |\psi\rangle\|^2$$

The set  $S_\psi := \{e^{i\alpha} |\psi\rangle \mid \alpha \in \mathbb{R}\}$  is called **ray** for each  $|\psi\rangle \in \mathbb{H}$  with  $\|\psi\| = 1$ . The same physical condition is described by each constituent of Ray. In other words, there is no physical difference between  $|\psi\rangle$  and  $e^{i\alpha} |\psi\rangle$ .

On the other hand, each unit vector in  $\mathbb{H}$  represents a physical state that characterizes a system using quantum mechanics which is the superposition principle.

**Definition 2.16: Superposition Principle**

Any linear combination of states that is normalized is again a state. That is, if  $|\varphi\rangle, |\psi\rangle \in \mathbb{H}$  then  $a|\varphi\rangle + b|\psi\rangle \in \mathbb{H}$  for  $a, b \in \mathbb{C}$  with  $|a|^2 + |b|^2 = 1$ .

A state that is an eigenvector of an operator associated with an observable  $A$  is also called an **eigenstate**. Let  $A$  be initially described by a pure state vector  $|\psi\rangle \in \mathbb{H}$ . Then the measurement of  $A$  yield  $\lambda_k \in \sigma(A)$  which can be seen as a preparation of the  $|\psi\rangle$  in the state  $|e_k\rangle \in \mathbb{H}$ . Another way to say this is that with the probability  $|\langle e_k | \psi \rangle|^2$ , the measurement ‘forces’ or ‘projects’ the object that was initially in the state  $|\psi\rangle$  into the eigenstate  $|e_k\rangle$ . It is described in the following postulate.

**Postulate 3: (Projection Postulate)**

Let the system be in the state  $|\psi\rangle$ . Then measurement has the following state transition

$$|\psi\rangle \xrightarrow{\text{measurement}} \frac{P_\lambda |\psi\rangle}{\|P_\lambda |\psi\rangle\|}$$

where  $P_\lambda$  is the projection onto the eigenspace of  $\lambda$ .

The state  $|\psi\rangle$  of a quantum system is also known as **wave function**.

Apart from measurement, a quantum state can be changed by applying unitary operators that are solutions to the famous **Schrödinger equation** (referred to as the initial value problem in the following) for a closed quantum system. It is formulated in the following postulate.

**Postulate 4: (Time Evolution)**

For an observable described by a pure state  $|\psi\rangle \in \mathbb{H}$ , the change, that is not caused by measurement, from the initial state  $|\psi(t_0)\rangle$  to the final state  $|\psi(t)\rangle$  is described by the time evolution operator  $U(t, t_0) \in \mathcal{U}(\mathbb{H})$ . That is

$$|\psi(t)\rangle = U(t, t_0) |\psi(t_0)\rangle.$$

The time evolution operator  $U(t, t_0)$  is the solution of the **initial value problem**

$$\begin{aligned} i \frac{d}{dt} U(t, t_0) &= H(t)U(t, t_0) \\ U(t_0, t_0) &= \mathbf{1} \end{aligned} \quad (13)$$

where  $H(t)$  is the self-adjoint operator known as **Hamiltonian**<sup>a</sup>. The quantum system is generated by the Hamiltonian.

<sup>a</sup>in this case time-dependent Hamiltonian

The total energy of the system is represented by the Hamiltonian 13. One can understand a quantum system's dynamics completely if one knows its Hamiltonian. However, determining a system's Hamiltonian can be challenging. For this thesis, we will take the Hamiltonian as given and not delve into the process of determining it.

Hamiltonian is a self-adjoint operator and has a spectral decomposition

$$H = \sum_{\lambda} \lambda |\psi\rangle \langle \psi| \quad (14)$$

where  $\lambda$  are eigenvalues corresponding to the normalized eigenvectors  $|\psi\rangle$ . The states  $|\psi\rangle$  are called **energy eigenstates** and  $\lambda$  is the energy of the state  $|\psi\rangle$ .

Quantum computing algorithms heavily rely on Hamiltonian and Time evolution operators. Unitary operators  $V$  applied on the quantum states are the building blocks of elementary gates. Unitary gates are used to construct quantum circuits in quantum computers and are implemented by creating a Hamiltonian that generates a time evolution  $U(t, t_0)$ . That is, one finds  $H(t)$  and  $t$  such that  $V = U(t, t_0)$ .

Now we will state the quantum mechanical postulates for the mixed state. A mixed state is a statistical mixture of several pure states<sup>10</sup>. It is defined mathematically as follows:

#### Postulate 5: (Mixed States)

A quantum state described by an operator  $\rho$  that has the following properties:

(i)  $\rho$  is self-adjoint

$$\rho^* = \rho.$$

(ii)  $\rho$  is positive

$$\rho \geq 0.$$

(ii)  $\rho$  has trace 1

$$\text{tr}(\rho) = 1.$$

is called **mixed state**. The operator  $\rho$  is called **density operator**. The set of **density operators** is denoted by  $D(\mathbb{H})$ .

<sup>10</sup>Some authors use the term mixed state for truly non-pure states, which is not the case. We are using the term in a general sense where pure states are special cases of mixed states.

All the quantum mechanical postulates are redefined for mixed state.

**Postulate 6: Generalization for Mixed State**

1. **Observables and Mixed States:** The mean value of an observable  $A$  in a mixed state  $\rho$  is

$$\langle A \rangle_\rho := \text{tr}(\rho A). \quad (15)$$

2. **Measurement Probability:** For an observable  $A$  described by mixed state  $\rho$ , the probability of obtaining eigenvalue  $\lambda \in \sigma(A)$  is given by the projection  $P_\lambda$  onto the eigenspace  $\text{Eig}(A, \lambda)$ .

$$\mathbb{P}_\rho(\lambda) := \text{tr}(\rho P_\lambda). \quad (16)$$

3. **Projection Postulate:** Let the system be in the state  $\rho$ . Then measurement has the following state transition

$$\rho \xrightarrow{\text{measurement}} \frac{P_\lambda \rho P_\lambda}{\text{tr} \rho P_\lambda} \quad (17)$$

where  $P_\lambda$  is the projection onto the eigenspace of  $\lambda$ .

4. **Time Evolution:** For an observable described by a mixed state  $\rho$ , every change that is not caused by measurement, from the initial state  $\rho(t_0)$  to the final state  $\rho(t)$  is described by the time evolution operator  $U(t, t_0) \in \mathcal{U}(\mathbb{H})$ . That is

$$\rho(t) = U(t, t_0)\rho(t_0).$$

The time evolution operator  $U(t, t_0)$  is the solution to the initial value problem as in the case of pure state.

Pure states  $|\psi\rangle \in \mathbb{H}$  are a special case of mixed states.

$$\rho_\psi := |\psi\rangle \langle \psi| = P_\psi \quad (18)$$

The probability to measure an eigenvalue  $\lambda_i$  of a observable  $A = \sum_i |e_i\rangle \lambda_i \langle e_i|$  can be calculated as follows:

$$\langle P_{e_i} \rangle_{\rho_\psi} = \text{tr}(\rho_\psi P_{e_i}) = \text{tr}(|\psi\rangle \langle \psi| e_i \langle e_i|) = \sum_k \langle e_k | \psi \rangle \langle \psi | e_i \rangle \langle e_i | e_k \rangle = |\langle e_i | \psi \rangle|^2,$$

and the expectation value

$$\langle A \rangle_{\rho_\psi} = \text{tr}(\rho_\psi A) = \text{tr}(|\psi\rangle \langle \psi| A) = \sum_{k,j} \langle e_k | \psi \rangle \langle \psi | e_i \rangle \lambda_i \langle e_i | e_k \rangle = \sum_i \lambda_i |\langle e_i | \psi \rangle|^2.$$

Interactions between a quantum system and its environment can transform pure states into true mixtures. This phenomenon is known as **decoherence**. One of the

most difficult difficulties in the practical application of quantum computing theory is avoiding decoherence over an extended time. We note again that knowledge of a state  $\rho$  or  $|\psi\rangle$  only allows claims about the statistics of the ensemble given by the state. Generally, one cannot accurately predict the behavior of individual objects within an ensemble. This is due to the **uncertainty** in measuring an observable  $A$  in the state  $\rho$ :

$$\Delta_\rho(A) := \sqrt{\left\langle \left( A - \langle A \rangle_\rho \mathbf{1} \right)^2 \right\rangle_\rho} \quad (19)$$

where  $\Delta_\rho(A)$  can be seen as the standard deviation from the mean value  $\langle A \rangle_\rho$ .

To describe the larger quantum system, we combine Hilbert spaces to form a larger composite system. The following postulate gives a formulation to construct the composite systems.

#### Postulate 7: Composite Systems

For  $n$  sub-systems  $\mathbb{H}^{A_j}$ , the **composite system** is represented as

$$\bigotimes_{j=1}^n \mathbb{H}^{A_j} = \mathbb{H}^{A_1} \otimes \dots \otimes \mathbb{H}^{A_n}$$

where  $\mathbb{H}^{A_j} \in {}^2\mathbb{H}$  for each  $j \in \{1, \dots, n\}$ . Here, we say that the system is in **composite state**.

To measure composite state we use the density operator  $\rho$  on the composite state  $\bigotimes_{j=1}^n \mathbb{H}^{A_j}$  as defined in Postulate 5. We can only measure the state of 1 sub-system at a time. For simplicity, we will define the measurement for 2 sub-systems of Hilbert space.

#### Definition 2.17: Partial Trace [63]

Let  $\mathbb{H}^A, \mathbb{H}^B \in {}^2\mathbb{H}$ . Then **partial trace** over  $\mathbb{H}^B$  is the map

$$tr^B : L(\mathbb{H}^A \otimes \mathbb{H}^B) \rightarrow L(\mathbb{H}^A)$$

For  $M \in L(\mathbb{H}^A \otimes \mathbb{H}^B)$ ,  $tr^B(M)$  is defined as

$$tr^B(M) := \sum_{j=1}^{d_B} (1_A \otimes \langle b_j |) M (1_A \otimes | b_j \rangle)$$

where  $|b_j\rangle$  is any orthogonal basis for  $\mathbb{H}_B$  and  $d_B$  is the dimension of  $\mathbb{H}_B$ . Partial trace  $tr^A(M)$  can be similarly defined.

Using the partial trace, we can define an operator  $\rho^A$  from the composite system's state  $\rho$ . This operator has properties of density operators and describes the state of the sub-system, say A when observed alone. Following measurement, the system

will collapse to the measured state. This new state will also be normalized, allowing for more quantum operations and measurements.

### Definition 2.18

A state  $\rho \in D(\mathbb{H}^A \otimes \mathbb{H}^B)$  that is composed of the sub-systems  $\mathbb{H}^A$  and  $\mathbb{H}^B$  is called **separable** or **product-state** if there exist states  $\rho_j^A \in D(\mathbb{H}^A)$ ,  $\rho_j^B \in D(\mathbb{H}^B)$  indexed by  $j \in I \subset \mathbb{N}$ , and a positive real numbers  $p_j$  satisfying

$$\sum_{j \in I} p_j = 1$$

such that

$$\rho = \sum_{j \in I} p_j \rho_j^A \otimes \rho_j^B.$$

Otherwise,  $\rho$  is called **entangled**.

Therefore, the entangled states are states that cannot be factored into individual states as shown in the example below.

### Example 4

The following is an example of separable state:

$$\frac{1}{2} (|00\rangle - |01\rangle + |10\rangle - |11\rangle) = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle)$$

and following is an entangled state:

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

## 2.4 Quantum Computing

### 2.4.1 Quantum bits

In classical computers, the *bit* is the smallest unit of information, represented by 0 or 1. In quantum computers, the smallest unit of information is the quantum bit or qubit. Before discussing its possible physical implementation, we first define a qubit as a mathematical object.

### Definition 2.19: Qubits

A **qubit** is a quantum mechanical system represented by two-dimensional Hilbert space  ${}^2\mathbb{H}$  and describes the quantum state of the quantum system. In  ${}^2\mathbb{H}$ , the orthonormal eigenvectors  $|0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ ,  $|1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$  of  $\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$  form a standard basis with 1 and  $-1$  as their corresponding eigenvalues<sup>a</sup>.

<sup>a</sup>Choosing orthonormal eigenvectors of  $\sigma_z$  is conventional.

In general, the qubit can be represented by a state vector  $|\psi\rangle$  or density vector  $\rho$  depending on whether it is in a pure state or mixed state.

The classical bits are either in one state or the other. By contrast, a qubit can exist in a continuum of state between  $|0\rangle$  and  $|1\rangle$  until it is measured. This is due to the *superposition principle* stated in the previous section.

Let a qubit be the following state

$$|\psi\rangle = a|0\rangle + b|1\rangle$$

where  $a, b \in \mathbb{C}$ . As mentioned in Postulate 2, when one measures a qubit, one can either observe the state  $|0\rangle$  with probability  $|a|^2$  or the  $|1\rangle$  with probability  $|b|^2$ .

Since  $|a|^2 + |b|^2 = 1$ , we can find  $\gamma, \varphi, \theta \in \mathbb{R}$  such that  $a = e^{i\gamma} \cos \frac{\theta}{2}$  and  $b = e^{i(\gamma+\varphi)} \sin \frac{\theta}{2}$ . Thus, a qubit state has the **general form**

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

As there is no difference between  $|\psi\rangle$  and  $e^{i\gamma} |\psi\rangle$ <sup>11</sup>, we can ignore the factor  $e^{i\gamma}$  and write

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle$$

The parameters  $\theta$  and  $\varphi$  represent a point on a three-dimensional unit sphere. This sphere is called **Bloch sphere**, as shown in figure 3, and it helps visualize the state of a single qubit.

Note that the whole of the Bloch sphere represents all possible states of a qubit. The two classical bits are represented by the north and south poles of the Bloch sphere.

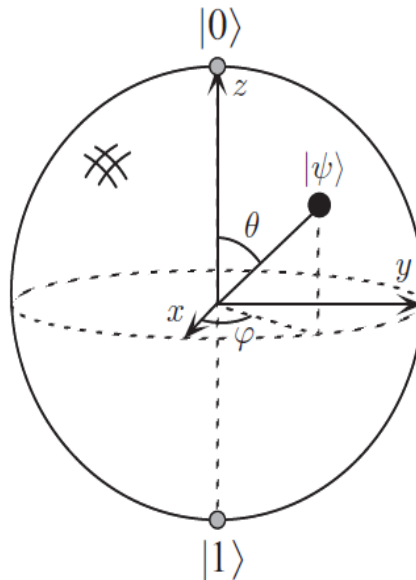


Figure 3: Bloch sphere representation of a qubit.

David DiVincenzo, in his seminal paper[64], describes the necessary conditions for the physical implementation of a quantum computation. The spin of an electron

<sup>11</sup>See the definition of Ray in the previous section

is one such physical implementation of the qubit. Bloch sphere can help us visualize the spin of an electron.

**Spin** of an electron is the internal angular momentum of an electron. It consists of three observables  $S_x, S_y, S_z$  corresponding to three directions in which electrons can move. These are grouped together to form the spin,  $\mathbf{S} = (S_x, S_y, S_z)$  and are defined in  ${}^2\mathbb{H}$ . These operators are defined as follows:

$$S_j = \frac{1}{2}\sigma_j \quad \text{for } j \in \{x, y, z\}$$

where  $\sigma_j$  are Pauli matrices defined below.

#### Definition 2.20: Pauli Matrices

The matrices  $\sigma_j \in \text{Mat}(2 \times 2, \mathbb{C})$  indexed by  $j \in \{x, y, z\}$  and defined as

$$\sigma_x := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_y := \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma_z := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

are called **Pauli Matrices**<sup>a</sup>.

<sup>a</sup>Some authors also include Identity matrix in Pauli Matrices.

Other physical manifestation possibilities are Trapped ions[65], polarized photons[66], Cold Trapped Ions[67] to name a few.

### 2.4.2 Quantum Gates

**Quantum gates**, also known as quantum operators, are used to manipulate qubits without collapsing their quantum states. Quantum gates operate linearly while preserving the norm, and take qubit from one state to another. **Quantum register** or **q-register** are the qubits on which quantum operations are performed.

The time evolution postulate of quantum mechanics (4) implies that all quantum gates must be unitary and are a solution to the initial value problem.

#### Definition 2.21

A **quantum n-gate** is a unitary operator

$$U : {}^2\mathbb{H}^{\otimes n} \rightarrow {}^2\mathbb{H}^{\otimes n}.$$

For  $n = 1$  a gate,  $U$  is called a *unary quantum gate* and for  $n = 2$  a *binary quantum gate*. All quantum gates are **reversible**, i.e., the output of the gate can be used to uniquely determine the input.

Table 1: Examples of Unary Quantum Gates.

Gate Name	Operator	Matrix	Symbol
Pauli-X	$X := \sigma_x$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$q - \boxed{X} -$

Pauli-Y	$Y := \sigma_y$	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	$q \text{ --- } \boxed{Y} \text{ ---}$
Pauli-Z	$Z := \sigma_z$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$q \text{ --- } \boxed{Z} \text{ ---}$
Hadamard	$H := \frac{\sigma_x + \sigma_z}{\sqrt{2}}$	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	$q \text{ --- } \boxed{H} \text{ ---}$
Phase-shift	$P(\alpha) :=  0\rangle\langle 0  + e^{i\alpha}  1\rangle\langle 1 $	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{bmatrix}$	$q \text{ --- } \boxed{P(\alpha)} \text{ ---}$
Phase-factor	$M(\alpha) := e^{i\alpha} \mathbf{1}$	$\begin{bmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\alpha} \end{bmatrix}$	$q \text{ --- } \boxed{M(\alpha)} \text{ ---}$
X-Rotation	$R_x(\alpha)$	$\begin{bmatrix} \cos(\frac{\alpha}{2}) & -i \sin(\frac{\alpha}{2}) \\ -i \sin(\frac{\alpha}{2}) & \cos(\frac{\alpha}{2}) \end{bmatrix}$	$q \text{ --- } \boxed{R_x(\alpha)} \text{ ---}$
Y-Rotation	$R_y(\alpha)$	$\begin{bmatrix} \cos(\frac{\alpha}{2}) & -\sin(\frac{\alpha}{2}) \\ \sin(\frac{\alpha}{2}) & \cos(\frac{\alpha}{2}) \end{bmatrix}$	$q \text{ --- } \boxed{R_y(\alpha)} \text{ ---}$
Z-Rotation	$R_z(\alpha)$	$\begin{bmatrix} e^{-i\frac{\alpha}{2}} & 0 \\ 0 & e^{-i\frac{\alpha}{2}} \end{bmatrix}$	$q \text{ --- } \boxed{R_z(\alpha)} \text{ ---}$

Table 2: Examples of Binary and 3-qubit Quantum Gates.

Gate Name	Operator	Matrix	Symbol
Controlled-NOT	$CX, CNOT$	$\begin{bmatrix} \mathbf{I} & 0 \\ 0 & X \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	$q_0 \text{ --- } \bullet \text{ ---}$ $q_1 \text{ --- } \oplus \text{ ---}$
SWAP	$SWAP$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$q_0 \text{ --- } \times \text{ ---}$ $q_1 \text{ --- } \times \text{ ---}$
Controlled Z	$CZ$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	$q_0 \text{ --- } \bullet \text{ ---}$ $q_1 \text{ --- } \boxed{Z} \text{ ---}$
Toffoli	$CCNOT$	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$	$q_0 \text{ --- } \bullet \text{ ---}$ $q_1 \text{ --- } \bullet \text{ ---}$ $q_2 \text{ --- } \oplus \text{ ---}$

Barenco et al. [68] demonstrated the universality of a set of one-bit quantum gates and two-bit CNOT gates, i.e., any unitary operations on any number of bits  $n$  can be described as compositions of these gates.

### 2.4.3 Quantum Circuits

Quantum circuits function similarly to conventional circuits, with quantum wires connecting quantum gates that control qubits and make measurements.

Let's look at a of a quantum circuit diagram 4 to understand it better. The circuit is read from left to right.  $q_0$  and  $q_1$  are two input qubits whereas  $c$  is a classical bit that is required to save the result of the measurement. Since in the example there are two qubits, we need two classical bits to store the result. The horizontal lines corresponding to qubits are called quantum wires. These are not physical wires, rather they may correspond to the passage of time or the movement of physical particles like photons or electrons. It is conventional to assume that qubits are in computational basis state as inputs, usually  $|0\rangle$ . The vertical dashed lines are used to separate each operation for better understanding and are not part of the circuit per se.

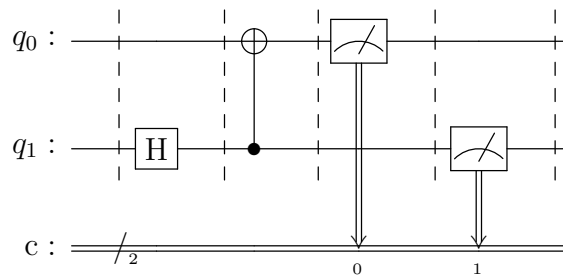


Figure 4: Example of Quantum Circuit.

Now, we will see what is happening in the circuit:

1. Qubits are initialized in the computational basis state.
2. Hadamard gate is performed on  $q_1$ .
3. CNOT gate is performed with  $q_1$  as the control gate and  $q_0$  as target.
4.  $q_0$  is measured and stored in 0-th bit.
5.  $q_1$  is measured and stored in 1-th bit.

Although there are many similarities between classical and quantum circuits, there are a few differences as well.

1. Measurement is implicit in a classical circuit whereas in quantum circuits there is a special emphasis on measurement.
2. It is possible to join wires in a classical circuit. As this operation is not reversible, it is not allowed in quantum circuits.
3. Quantum circuits are *acyclic*, i.e., there is no feedback from one part of the circuit to another.
4. It is possible to copy a bit. However, it is not possible to create a copy of an arbitrary qubit. This is known as **No-cloning theorem**.

## 2.5 Adiabatic Algorithm

Let  $\Psi(t_0)$  be the eigenstate of a quantum system at some initial time  $t_0$ . As time increases, the quantum system will evolve according to the Schrödinger equation to reach the final eigenstate  $\Psi(t_1)$  at some final time  $t_1$ . There are two ways to perform this evolution on a quantum computer. The first way is to use a finite number of gates  $U_1, \dots, U_N$  and apply them on an initial state  $\psi(t_0)$  to produce the final state  $\psi(t_1)$ . For a pure state, this can be seen as:

$$|\Psi(t_1)\rangle = U_N U_{N-1} \dots U_2 U_1 |\Psi(t_0)\rangle$$

The second way is using Adiabatic Quantum Computation (AQC). AQC uses a time-dependent Hamiltonian  $H(t)$  to generate a time evolution  $U(t_1, t_0)$  that transforms the initial pure state  $|\Psi(t_0)\rangle$  into the final state  $|\Psi(t_1)\rangle$ . This is possible due to Postulate (4).

More precisely, let the system be initially in the  $j$ th eigenstate  $|\Phi_j(t_0)\rangle$  of the initial time-independent Hamiltonian  $H_{t_0}$  at time  $t_0$ , and let it evolve according to the time evolution generated by  $H(t)$  until  $t_1$  to reach the  $j$ th eigenstate  $|\Phi_j(t_1)\rangle$  of the final time-independent Hamiltonian  $H_{t_1}$ .  $U(t_1, t_0) |\Phi_j(t_0)\rangle$  denotes the time-evolved state. The Quantum Adiabatic Theorem (QAT) 2.27 tells us that under certain conditions the eigenstate of the final Hamiltonian and time-evolved-state can be made arbitrarily close to each other.

### Definition 2.22

Let  $E_j(t)$  and  $|\Phi_j(t)\rangle$  denotes the  $j$ th 'instantaneous' eigenvalue and 'instantaneous' eigenvector, respectively, of the time-dependent Hamiltonian  $H(t)$ , where the index  $j \in I \subset \mathbb{N}_0$  and  $H(t) := H(t_0 + tT)$  with  $t \in [0, 1]$  and  $T := t_1 - t_0$ .

$t$  can be seen as a tuning parameter of the speed of the change of the Hamiltonian and  $T$  as the total time.

Similarly, we define  $U(t) := U(t_0 + tT, t_1)$ .

Generally,  $|\Phi_j(t_1)\rangle$  does not coincide with the state reached by time evolution  $U(t_1, t_0)$  from the initial state  $|\Phi_j(t_0)\rangle$ . That is

$$|\Phi_j(t_1)\rangle \neq U(t_1, t_0) |\Phi_j(t_0)\rangle$$

and the goal of the adiabatic method is to design  $H(t)$  such that, at least for a particular  $j$ , the difference between  $U(t_1, t_0) |\Phi_j(t_0)\rangle$  and  $|\Phi_j(t_1)\rangle$  is as small as possible. QAT 2.27 provides a bound on this difference as explained below. Before stating the theorem, we need to state some assumptions and definitions.

### Definition 2.23: Gap of Eigenvalue

Let  $H(t) \in B_{sa}(\mathbb{H})$ . The **gap** of the eigenvalue  $E_j(t)$  of  $H(t)$  is defined as

$$g_j : [0, 1] \longrightarrow \mathbb{R}$$

$$t \longmapsto g_j(t) := \min \left\{ |E_k(t) - E_j(t)| \mid k \in I \setminus \{j\} \right\}$$

**Definition 2.24: Adiabatic Assumption (AA)**

$H(t) \in B_{sa}(\mathbb{H})$  satisfies adiabatic assumptions if

- (i)  $H(t)$  is at least twice differentiable with respect to every  $t \in (0, 1)$ .
- (ii)  $H(t)$  has purely discrete spectrum  $\{E_j(t) | j \in I \subset \mathbb{N}_0\}$  for every  $t \in [0, 1]$ .
- (iii) The eigenvalues  $E_j(t)$  of  $H(t)$  do not cross, that is,  $H(t)$  has non-zero gap.

**Definition 2.25: Adiabatic Generator**

Let  $H(t) \in B_{sa}(\mathbb{H})$  satisfies the AA 2.24, and let  $\{P_i(t) | i \in I\}$  the projections onto the eigenspaces of  $H(t)$ . For each  $j \in I$  and total time  $T$  the **adiabatic generator**  $H_{A,j}$  is defined as

$$\begin{aligned} H_{A,j} : [0, 1] &\rightarrow B_{sa}(\mathbb{H}) \\ t &\mapsto H_{A,j}(t) := TH(t) + i[\dot{P}_j(t), P_j(t)]. \end{aligned}$$

where  $\dot{P}_j(t)$  is the derivative of  $P_j(t)$ .

**Definition 2.26: Adiabatic Intertwiner**

Let  $H_{A,j}$  be an adiabatic generator 2.25. For each  $j \in I$  the **adiabatic intertwiner**  $U_{A,j} : [0, 1] \rightarrow \mathcal{U}(\mathbb{H})$  is the solution of the initial value problem

$$\begin{aligned} iU_{A,j}(t) &= H_{A,j}(t)U_{A,j}(t) \\ U_{A,j}(0) &= \mathbf{1} \end{aligned}$$

**Theorem 2.27: Quantum Adiabatic Theorem (QAT)**

Let  $H(t) \in B_{sa}(\mathbb{H})$  satisfies the AA 2.24, and let  $\{P_i(t) | i \in I\}$  be the set of projections onto the eigenspaces of  $H(t)$ . Let  $g_j$  defines the gap for  $H(t)$  for a given  $j \in I$  2.23 and, let  $U(t)$  and  $U_{A,j}$  be as defined in 2.22 and 2.26 and let  $T$  be the total time.

Then for all  $j \in I$  and  $t \in [0, 1]$  we have

$$\|(U_{A,j}(t) - U(t))P_j(t_0)\| \leq \frac{C_j(t)}{T},$$

where

$$C_j(t) := \frac{\|\dot{H}(t)\|}{g_j(t)^2} + \frac{\|\dot{H}(0)\|}{g_j(0)^2} + \int_0^t \left( \frac{\|\ddot{H}(u)\|}{g_j(u)^2} + 10 \frac{\|\dot{H}(u)\|^2}{g_j(u)^3} \right) du.$$

QAT provides an upper-bound of the difference between the final state reached by applying instantaneous eigenvector ( $U(t)$ ) on the initial state and the final state reached by applying time-evolution ( $U_{A,j}(t)$ ) on the initial state. Therefore, if one

can find a Hamiltonian that satisfies the AA, one can solve optimization problems using the following algorithm [33],[69]. It can be seen that in order to have better result one can either have large total time  $T$  value and/or large gap  $g$ .

Now that we have the necessary conditions in which one can use the QAT to find the upper bound, we can define the adiabatic algorithm.

**Algorithm 3: Adiabatic Algorithm**

1. Find a Hamiltonian  $H_B$  that has an eigenstate  $|\psi_0\rangle$  with the lowest eigenvalue.
2. Find a Hamiltonian  $H_C$  that encodes the optimization problem one is trying to solve.
3. Construct a time-dependent Hamiltonian  $H(t)$  that satisfies the AA 2.24. It can be written as:

$$H(t) = \left(1 - \frac{t}{t_1}\right) H_B + \left(\frac{t}{t_1}\right) H_C$$

where  $t_1$  is the final time,  $t \in [0, T]$  and  $T$  is the total evolution time.  $H(t)$  is known as transitional Hamiltonian and it encapsulates the adiabatic evolution path.

4. Prepare the quantum system in the initial state  $|\psi_{t_0}\rangle$ .
5. Evolve the system from  $t = t_0$  to the final time  $t = t_1$  with the time evolution  $U(t_1, t_0)$  as mentioned in the Postulate (4).
6. Observe the final state  $U(t_1, t_0) |\psi_{t_0}\rangle$  which will correspond to the lowest eigenvalue of  $H_C$ .

According to the Adiabatic Algorithm, if one starts in a highest/lowest eigenvalue of  $H_B$  then the system will evolve into a system that will have the highest/lowest eigenvalue of  $H_C$ , provided the process runs for a long enough time and the gap of eigenvalue remains large. This is due to AA (iii).

Although the Adiabatic Algorithm requires continuous state evolution (AA (i)), it is possible to emulate this process on a gate-based quantum computer by Trotterizing  $\hat{U}(t)$  into sufficiently small steps. This involves decomposing  $\hat{U}(t)$  into a sequence of small steps using the Trotter-Suzuki formula:

$$\hat{U}(t) \approx \prod_{k=0}^{T-1} \exp \left[ -i\hat{H}(k\Delta\tau)\Delta\tau \right] = \prod_{k=0}^{T-1} \exp \left[ -if(k\Delta\tau)\hat{H}_C\Delta\tau \right] \exp \left[ -ig(k\Delta\tau)\hat{H}_B\Delta\tau \right] \quad (20)$$

where  $\Delta\tau =: t/T$ ,  $f(t) = \left(1 - \frac{t}{t_1}\right)$ , and  $g(t) = \left(\frac{t}{t_1}\right)$ . We observe that as  $k$  increases,  $f(k\Delta\tau)$  increases while  $g(k\Delta\tau)$  decreases. Consequently, the time steps of  $\hat{H}_C$  will decrease linearly, while those of  $\hat{H}_B$  will increase linearly. A detailed mathematical description of Trotterization is discussed in Subsection 2.7.

In the next sub-section, we will look into the kind of matrices (Hamiltonian) that satisfy the AA and the uniqueness of the solution.

## 2.6 Irreducible Matrices

Irreducible matrices play a central role in the proof of the Perron-Frobenius Theorem 2.40. The Perron-Frobenius Theorem demonstrates the existence of a unique maximum eigenvalue for a non-negative irreducible matrix. This result is crucial in showing that irreducible matrices satisfy the AA, which explains why the Adiabatic Theorem can be applied in QAOA at every iteration.

### Definition 2.28: Non-Negative and Positive Matrices

Let  $M_n$  be the set of all  $n \times n$  matrices and  $A = [a_{ij}] \in M_n$  with  $i, j \in \{1, \dots, n\}$ . Then  $A \geq 0$  is **non-negative matrix** if all  $a_{ij}$  are real and non-negative. Similarly,  $A > 0$  is **positive matrix** if all its entries  $a_{ij}$  are real and positive.

### Definition 2.29: Irreducible

A matrix  $X$  is said to be **cogredient** to a matrix  $Y$  if there exists a permutation matrix  $P$  such that  $X = P^T Y P$ . A non-negative matrix  $A \in M_n$ ,  $n \geq 2$ , is called **reducible** if it is cogredient to a matrix of the form

$$\begin{bmatrix} B & C \\ 0 & D \end{bmatrix}$$

where  $B$  and  $D$  are square submatrices. Otherwise,  $A$  is **irreducible**.

From the definition of irreducible matrices, it is clear that a matrix that has either a row or a column full of 0s cannot be irreducible.

To prove the Perron-Frobenius Theorem, we state some properties of Non-negative matrices without proving them. Let  $A, B \in M_n$  then the following holds:

$$|A^m| \leq |A|^m \quad \forall m = 1, 2, \dots \quad (21)$$

$$\text{If } 0 \leq A \leq B, \text{ then } 0 \leq A^m \leq B^m \quad \forall m = 1, 2, \dots \quad (22)$$

$$\text{If } |A| \geq |B|, \text{ then } \|A\| \geq \|B\| \quad (23)$$

$$\|A\| = \||A|\| \quad (24)$$

where  $\|\cdot\|$  is the matrix norm, which for a matrix  $A \in M_n$  is  $\max \|Ax\|$  such that  $\|x\| = 1$ .

### Theorem 2.30

For  $A, B \in M_n$ , if  $|A| \leq B$ , then  $\rho(A) \leq \rho(|A|) \leq \rho(B)$ .

*Proof.* Let  $|A| \leq B$  then  $A \leq |A| \leq B$

$$\stackrel{22}{\implies} A^m \leq |A|^m \leq B^m$$

$$\stackrel{21}{\implies} |A^m| \leq |A|^m \leq B^m$$

$$\stackrel{23}{\implies} \|A^m\| \leq \| |A|^m \| \leq \|B^m\| \text{ and}$$

$$\stackrel{24}{\implies} \|A^m\|^{\frac{1}{m}} \leq \| |A|^m \|^{\frac{1}{m}} \leq \|B^m\|^{\frac{1}{m}}$$

for  $m = 1, 2, \dots$ . The inequality follows since  $A$  and  $B$  are non-negative matrices.

Now let  $m \rightarrow \infty$  and apply 6.2, we get  $\rho(A) \leq \rho(|A|) \leq \rho(B)$ .  $\square$

### Corollary 2.31

Let  $A, B \in M_n$ . If  $0 \leq A \leq B$ , then  $\rho(A) \leq \rho(B)$ .

*Proof.* Follows from 2.30.  $\square$

### Theorem 2.32

Let  $A \in M_n$  and suppose that  $A > 0$ . Then

$$\min_{1 \leq i \leq n} \sum_{j=1}^n a_{ij} \leq \rho(A) \leq \max_{1 \leq i \leq n} \sum_{j=1}^n a_{ij}$$

*Proof.* We define  $\alpha = \min_{1 \leq i \leq n} \sum_{j=1}^n a_{ij}$  and construct a new matrix  $B$  with  $A \geq B \geq 0$  such that  $\sum_{j=1}^n b_{ij} \equiv \alpha \forall i = 1, 2, \dots$ .

To see if such a construct is possible, consider the following two cases:

1. if  $\alpha = 0$ , set  $B = 0$
2. if  $\alpha > 0$ , set  $b_{ij} = \alpha a_{ij} (\sum_{j=1}^n a_{ij})^{-1}$

From 6.3, we have  $\rho(B) = \alpha$ .

And from 2.31 we have  $\rho(B) \leq \rho(A)$ . Similarly, the upper bound also follows.  $\square$

### Corollary 2.33

Let  $A \in M_n$  and suppose  $A \geq 0$  and  $\sum_{j=1}^n \forall i = 1, 2, \dots, n$  then  $\rho(A) > 0$ . That is,  $\rho(A) > 0$  if  $A > 0$  or if  $A$  is irreducible and non-negative.

*Proof.* Follows from 2.32.  $\square$

### Definition 2.34: Perron Vector

Let  $A \in M_n$  and suppose  $A > 0$ . Then the vector  $x$  is called **Perron vector** if  $Ax = \rho(A)x$ ,  $x > 0$ , and  $\sum_{i=1}^n x_i = 1$

### Theorem 2.35

Let  $A \in M_n$  and  $A \geq 0$  then  $\rho(A)$  is an eigenvalue of  $A$  and  $\exists x > 0$  such that  $Ax = \rho(A)x$ .

*Proof.* For any  $\epsilon > 0$ , let  $A(\epsilon) \equiv [a_{ij} + \epsilon] > 0$ .

To show the existence of  $x > 0$ , let  $x(\epsilon)$  be the Perron vector of  $A(\epsilon)$ . By definition of Perron vector we have  $x(\epsilon) > 0$  and  $\sum_{i=1}^n x(\epsilon)_i = 1$ .

Now, define the set of vectors  $\{x(\epsilon) : \epsilon > 0\}$ . We see that this set is contained in the compact set  $\{x : x \in \mathbb{C}^n, \|x\|_1 \leq 1\}$ . Therefore, by Bolzano–Weierstrass theorem and Monotone-convergence theorem [70], there a monotone decreasing sequence  $\epsilon_1, \epsilon_2, \dots$  with  $\lim_{k \rightarrow \infty} \epsilon_k = 0$  such that  $\lim_{k \rightarrow \infty} x(\epsilon_k) \equiv x$  holds.

$x(\epsilon_k)$  are also Perron vectors, therefore,  $x(\epsilon_k) > 0 \forall k = 1, 2, \dots$ , and the following holds

$$x = \lim_{k \rightarrow \infty} x(\epsilon_k) \geq 0 \quad \text{and} \quad x \neq 0$$

as

$$\sum_{i=1}^n x_i = \lim_{k \rightarrow \infty} \sum_{i=1}^n x(\epsilon_k)_i \equiv 1.$$

From 2.30 it follows that  $\rho(A(\epsilon_k)) \geq \rho(A(\epsilon_{k+1})) \geq \dots \geq \rho(A) \forall k = 1, 2, \dots$ , which means the sequence of real numbers  $\{\rho(A(\epsilon_k))\}_{k=1,2,\dots}$  exists and  $\rho \geq \rho(A)$ .

Now as  $x(\epsilon)$  is a Perron vector, we have

$$\begin{aligned} Ax &= \lim_{k \rightarrow \infty} A(\epsilon_k)x(\epsilon_k) = \lim_{k \rightarrow \infty} \rho(A(\epsilon_k))x(\epsilon_k) \\ &= \lim_{k \rightarrow \infty} \rho(A(\epsilon_k)) \lim_{k \rightarrow \infty} x(\epsilon_k) = \rho x \end{aligned}$$

and that  $x \neq 0$  (as shown above). This implies that  $\rho$  is an eigenvalue of  $A$ .  $\square$

### Theorem 2.36

Let  $A \in M_n$  and  $A > 0$  then  $\rho(A)$  is an eigenvalue of algebraic multiplicity 1.

*Proof.* We will prove this by contradiction.

Using Schur Triangularization theorem 6.4,  $A$  can be written as

$$A = U\Delta U^*$$

where  $U$  is unitary matrix,  $\Delta$  is an upper triangular matrix whose main diagonal entries are eigenvalues of  $A$ .

By 2.35,  $\rho = \rho(A)$  is an eigenvalue of  $A$ . Let the algebraic multiplicity of  $\rho$  be  $k > 1$ . Then using Schur Triangularization theorem, the main diagonal entries of  $\Delta$  can be written as  $\rho, \dots, \rho, \lambda_{k+1}, \dots, \lambda_n$ .

However, by 6.5 we have

$$L = \lim_{m \rightarrow \infty} [\rho(A)^{-1}A]^m \tag{25}$$

$$= U \lim_{m \rightarrow \infty} \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \frac{\lambda_{k+1}}{\rho} & & \\ & 0 & & & \ddots & \\ & & & & & \frac{\lambda_n}{\rho} \end{bmatrix}^m U^* \tag{26}$$

$$= U \lim_{m \rightarrow \infty} \begin{bmatrix} 1 & & & & \\ & \ddots & & & * \\ & & 1 & & \\ & & & 0 & \\ 0 & & & & \ddots \\ & & & & & 0 \end{bmatrix} U^* \quad (27)$$

where the diagonal entry 1 is repeated  $k$  times and the diagonal entry 0 is repeated  $n - k$  times. This implies that  $L$  has rank at least  $k$ . However, due to 6.6,  $L$  is a positive matrix of rank 1. Therefore,  $k > 1$  is not possible.  $\square$

**Theorem 2.37**

Let  $A \in M_n$  and  $A \geq 0$ . Then  $A$  is irreducible  $\iff (I + A)^{n-1} > 0$

*Proof.* We will prove  $A$  is reducible if and only if  $(I + A)^{n-1} > 0$  has at least one 0 entry.

$\Rightarrow$  Assume  $A$  is reducible. By definition, there exists a permutation matrix  $P$  such that  $A$  can be written as a block upper triangular matrix:

$$A = P \begin{bmatrix} B & C \\ 0 & D \end{bmatrix} P^T = P \tilde{A} P^T$$

where  $B, C, 0$  and  $D$  are block matrices.

Notice that  $\tilde{A}^2, \tilde{A}^3, \dots$ , and  $\tilde{A}^{n-1}$  all have the 0 block of the same size in the lower left of  $\tilde{A}$ . Now consider

$$\begin{aligned} (I + A)^{n-1} &= (I + P \tilde{A} P^T)^{n-1} \\ &= (P[I + \tilde{A}]P^T)^{n-1} \quad \text{since } P P^T = P P^{-1} = I \\ &= P(I + \tilde{A})^{n-1} P^T \quad \text{since } P^{n-1} = P \\ &= P \left[ I + (n-1)\tilde{A} + \binom{n-1}{2} \tilde{A}^2 + \dots + \binom{n-1}{n-1} \tilde{A}^{n-1} \right] P^T \end{aligned}$$

and notice that all of the terms in the square brackets have an 0 block in the lower left. Therefore,  $(I + A)^{n-1}$  is reducible and hence it has at least one 0 element.

$\Leftarrow$  Conversely, assume  $(I + A)^{n-1}$  has at least one zero entry. This implies that there exists at least one pair  $(i, j)$  with  $i \neq j$   $(i, j)$  element of  $(I + A)^{n-1}$  is 0.

Consider a graph  $G(A)$  associated with  $A$ . Then each element in  $A$  corresponds to a directed edge. As  $(I + A)^{n-1}$  has a zero entry, this means there does not exist a path between  $i$  and  $j$  node of at most  $n - 1$  length. This implies that the graph can be partitioned into at least two non-empty sets  $S$  and  $T$  with no edge between them.

Now, we can rearrange the elements of  $A$  using some permutation matrix  $P$  such that  $A$  can be transformed into a block upper triangular matrix as follows

$$\tilde{A} = P^T A P = \begin{bmatrix} B & C \\ 0 & D \end{bmatrix}, \quad B \in M_r, \quad 0 \in M_{n-r, r}$$

Therefore,  $A$  is reducible.

From this we conclude that  $A$  is irreducible if and only if  $(I + A)^{n-1} \neq 0$ .  $\square$

**Theorem 2.38**

Let  $A \in M_n$  and let  $\lambda_1, \dots, \lambda_n$  be the eigenvalues of  $A$  (including multiplicities). Then  $\lambda_1 + 1, \dots, \lambda_n + 1$  are the eigenvalues of  $I + A$  and  $\rho(I + A) \leq 1 + \rho(A)$ . If  $A \geq 0$ , then  $\rho(I + A) = 1 + \rho(A)$ .

*Proof.* For  $i \in \{1, \dots, n\}$ , let  $\lambda_i \in \sigma(A)$  be an eigenvalue of  $A$  with multiplicity  $k$ . Then  $\lambda_i$  is a root of the characteristic equation

$$p_A(t) := \det(tI - A), \text{ i.e., } p_A(\lambda_i) = \det(\lambda_i I - A) = 0$$

with multiplicity of  $k$ .

Since  $\det(tI - A) = \det[(t+1)I - (A+I)]$ , therefore  $\lambda_i + 1$  is a root of  $p_{A+I}(t) = \det[tI - (A+I)]$ , i.e.,  $\lambda_i + 1$  is an eigenvalue of  $I + A$ .

Therefore,

$$\rho(I + A) = \max_{1 \leq i \leq n} |\lambda_i + 1| \leq \max_{1 \leq i \leq n} |\lambda_i| + 1 = \rho(A) + 1.$$

Now assume that  $A \geq 0$ . This means that  $I + A \geq 0$  and therefore by 2.35, we have that  $\rho(I + A)$  is an eigenvalue of  $I + A$ . Using the proof above, this means  $1 + \rho(A)$  is an eigenvalue of  $I + A$ . Therefore, we conclude  $\rho(I + A) = 1 + \rho(A)$ .  $\square$

**Theorem 2.39**

Let  $A \in M_n$ . Further assume that  $A \geq 0$  and  $A^k > 0$  for some  $k \geq 1$ , then  $\rho(A)$  is an algebraically simple eigenvalue of  $A$ , i.e.,  $\rho(A)$  has an algebraic multiplicity of 1.

*Proof.* We will prove this by contradiction.

Let  $\lambda_1, \dots, \lambda_n$  are the eigenvalues of  $A$ . Then  $\lambda_1^k, \dots, \lambda_n^k$  are the eigenvalues of  $A^k$  since  $AAx = A(\lambda x) = \lambda Ax = \lambda^2 x$ , which can be extended to  $A^k$ .

As  $A$  is non-negative, it follows from 2.35 that  $\rho(A)$  is an eigenvalue of  $A$ .

Now, assume  $\rho(A)$  has multiplicity greater than 1. Then  $\rho(A)^k = \rho(A^k)$  will have a multiplicity eigenvalue of  $A^k$ . However, since  $\rho(A^k)$  is a positive matrix, by 2.36  $\rho(A^k)$  is a simple eigenvalue which is a contraction. Therefore,  $\rho(A)$  is simple eigenvalue of  $A$ .  $\square$

**Theorem 2.40: Perron-Frobenius Theorem**

Let  $A$  be an  $n \times n$  complex matrix and suppose that  $A$  is irreducible and non-negative. Then

- (a)  $\rho(A) > 0$ ;
- (b)  $\rho(A)$  is an eigenvalue and  $\exists x > 0$  such that  $Ax = \rho(A)x$ ; and
- (c)  $\rho(A)$  is an algebraically simple eigenvalue of  $A$ .

*Proof.* (a) follows directly from Corollary 2.33.

(b) follows from 2.35.

(c) 2.38 shows that if  $\rho(A)$  is a multiple eigenvalue of  $A$  then  $1 + \rho(A) = \rho(I + A)$  is a multiple eigenvalue of  $I + A$ .

However, since  $A$  is non-negative and irreducible implies that  $I + A \geq 0$ . Therefore, by 2.37 we get  $(I + A)^{n-1} > 0$ .

Finally, by using 2.39,  $1 + \rho(A)$  is an algebraically simple eigenvalue of  $A$ .  $\square$

### Definition 2.41: Strongly Connected Graph

A directed graph  $\Gamma$  is called **strongly connected** (SC) if between every pair of distinct nodes  $P_i, P_j$  in  $\Gamma$  there is a directed path of finite length that begins at  $P_i$  and ends at  $P_j$ .

### Theorem 2.42

Assume  $A \in M_n$  and  $A \geq 0$ . Let  $P_i, P_j \in \Gamma(A)$  be two nodes. Then there exists a directed path of length  $m$  in  $\Gamma(A)$  from  $P_i$  to  $P_j$  if and only if  $[A^m]_{ij} \neq 0$ . Here,  $[A]_{ij}$  means the  $(i, j)$ -th element of  $A$ .

*Proof.* We will prove it by induction.

For  $m = 1$  the claim follows trivially.

Let  $m = 2$ . Since  $[A^2]_{ij}$  has length 2, it can be written as follows:

$$[A^2]_{ij} = \sum_{k=1}^n [A]_{ik} [A]_{kj} = \sum_{k=1}^n a_{ik} \cdot a_{kj}$$

$[A^2]_{ij} \neq 0 \iff$  there exists at least one  $k$  for which both  $a_{ik}$  and  $a_{kj}$  are not zero  
 $\iff$  there is a path of length 1 from  $P_i$  to  $P_k$  and of length 1 from  $P_k$  to  $P_j$   
 $\iff$  there is a path of length 2 from  $P_i$  to  $P_j$ .

Now, let us suppose that the claim holds for the case  $m = q$ . Then following as above:

$$[A^{q+1}]_{ij} = \sum_{k=1}^n [A^q]_{ik} [A]_{kj} = \sum_{k=1}^n [A^q]_{ik} \cdot a_{kj}$$

$[A^{q+1}]_{ij} \neq 0 \iff$  there exists at least one  $k$  for which both  $[A^q]_{ik}$  and  $a_{kj}$  are not zero  
 $\iff$  there is a path of length  $q$  from  $P_i$  to  $P_k$  and of length 1 from  $P_k$  to  $P_j$   
 $\iff$  there is a path of length  $q + 1$  from  $P_i$  to  $P_j$ .

$\square$

### Theorem 2.43

Let  $A \in M_n$  and  $A \geq 0$ . Then  $A$  has property SC iff  $(I + A)^{n-1} > 0$ .

*Proof.* Using binomial expansion, we have

$$(I + A)^{n-1} = I + (n-1)A + \binom{n-1}{2}A^2 + \cdots + \binom{n-1}{n-2}A^{n-1}$$

Then  $(I + A)^{n-1} > 0$

$\iff \exists$  a node  $(i, j)$  with  $i \neq j$  such that at least one of  $A, A^2, \dots, A^{n-1}$  has a positive  $(i, j)$  entry

$\stackrel{2.42}{\iff}$  there is some directed path in  $\Gamma(A)$  from  $P_i$  to  $P_j$ .

$\iff \Gamma(A)$  property SC. □

### Theorem 2.44

$A$  is irreducible  $\iff A$  has property SC.

*Proof.* Direct consequence of 2.37 and 2.43 □

### Theorem 2.45

Suppose  $A$  is an irreducible non-negative matrix and let  $D$  be a diagonal matrix with positive entries. Then  $A + D$  is irreducible

*Proof.* Since  $A$  is irreducible then  $A$  has property SC. Adding a diagonal matrix  $D$  will not remove any path between any nodes. □

## 2.7 Trotterization

In this section, we look into unitary evolution groups and their role in defining Stone's theorem. Stone's theorem establishes a one-to-one correspondence between self-adjoint operators and unitary evolution groups. We conclude this section with the proof of the Trotter-Suzuki formula, also known as the Trotter product formula. The Trotter product formula enables the use of the adiabatic algorithm on a gate-based quantum computer, as quantum gates are, by definition, unitary operators.

### Definition 2.46

A map  $G : \mathbb{R} \rightarrow B(\mathbb{H})$  is **unitary evolution group** on  $\mathbb{H}$  if  $G(t)$  is unitary operator onto  $\mathbb{H}$  and  $G(t+s) = G(t)G(s), \forall t, s \in \mathbb{R}$ .

$t$  and  $s$  play the role time in quantum mechanics.

### Definition 2.47

If  $G(t)$  is a unitary evolution group, the operator  $T$  defined by

$$\text{dom}(T) := \left\{ \xi \in \mathbb{H} : \exists \lim_{h \rightarrow 0} \frac{1}{h}(G(h) - \mathbf{1})\xi \right\},$$

that is,  $\xi \in \text{dom}(T)$  iff  $t \mapsto G(t)\xi$  is differentiable at  $t = 0$ ,

$$T\xi := i \lim_{h \rightarrow 0} \frac{1}{h}(G(h) - \mathbf{1})\xi, \quad \xi \in \text{dom}(T),$$

is called the **infinitesimal generator** of  $G(t)$ .

As mentioned in quantum mechanics postulate 4, it is common to construct a unitary evolution group from a self-adjoint operator (or Hamiltonian). Unitary evolution groups result in the time evolution of quantum states, which are solutions of Schrödinger equations. The following two theorems provide the mathematical description of this process, which we will state without proof.

**Theorem 2.48**

If  $T$  is self-adjoint, there exists a strongly continuous unitary evolution group  $U(t)$  for which  $T$  is its infinitesimal generator. Then  $U(t) = e^{-itT}$ ,  $t \in \mathbb{R}$ .

**Theorem 2.49: Stone theorem**

If  $U(t)$  is a measurable unitary evolution group on  $\mathbb{H}$ , then its infinitesimal generator  $T$  is self-adjoint, that is,  $U(t) = e^{-itT}$ .

Let  $T, S \in \mathbb{H}$  be two self-adjoint operators such that  $T + S$  is also self-adjoint, with  $\text{dom}(T + S) = \text{dom}T \cap \text{dom}S$ . The question we are interested in is: what is the relationship between  $e^{-it(T+S)}$  and the individual unitary evolution groups  $e^{-itT}$  and  $e^{-itS}$ ? This question arises in QAOA when we approximate the sum of operators in 2.55. The following theorem shows that it is possible to write  $e^{-it(T+S)}$  in terms of the individual unitary evolution groups.

**Theorem 2.50: Trotter Product Formula**

Suppose that  $T, S \in \mathbb{H}$  are self-adjoint operators such that  $T + S$  is also self-adjoint, with  $\text{dom}(T + S) = \mathcal{D} := \text{dom}(T) \cap \text{dom}(S)$ . Then, for each  $t \in \mathbb{R}$ , we have

$$e^{-it(T+S)} = s - \lim_{n \rightarrow \infty} (e^{-\frac{t}{n}T} e^{-\frac{t}{n}S})^n.$$

*Proof.* We will first consider the two important points in the proof:

1. For  $0 \neq h \in \mathbb{R}$  and  $\xi \in \mathcal{D}$ , let

$$u_h(\xi) := \frac{1}{h} (e^{-itT} e^{-itS} \xi - e^{-it(T+S)} \xi). \quad (28)$$

Using the definition of unitary evolution group, we see that the domain  $\mathcal{D}$  is left invariant by  $e^{-isT}$ ,  $e^{-isS}$  and  $e^{-is(T+S)}$ ,  $\forall s \in \mathbb{R}$ . Consider the following identity for  $\xi \in \mathcal{D}$

$$u_h(\xi) = \frac{(e^{-itT} - 1)}{h} \xi + e^{-itT} \frac{(e^{-itS} - 1)}{h} \xi - \frac{(e^{-it(T+S)} - 1)}{h} \xi$$

This implies that  $\lim_{h \rightarrow 0} u_h(\xi) = 0$ . Further, let  $u_0(\xi) := 0, \xi \in \mathcal{D}$ .

2. For  $n \in \mathbb{N}$  and expanding 28, we get

$$\begin{aligned} (e^{-itT/n} e^{-itS/n})^n - (e^{-it(T+S)/n})^n &= (e^{-itT/n} e^{-itS/n})^n - e^{-it(T+S)} \\ &= \sum_{j=0}^{n-1} (e^{-itT/n} e^{-itS/n})^j \\ &\quad \times [e^{-itT/n} e^{-itS/n} - e^{-it(T+S)/n}] (e^{-it(T+S)/n})^{n-1-j} \end{aligned}$$

From 1. above we see that  $u_h : \mathcal{D} \rightarrow \mathbb{H}$  is linear and bounded for each  $h$ . Further, we see that for fixed  $\xi \in \mathcal{D}$ ,  $u_h$  is continuous as a function of  $h$ . This gives the pointwise convergence  $u_h(\xi) \rightarrow 0$  as  $h \rightarrow \infty$  which implies that there exists  $c(\xi) > 0$  for which  $\|u_h(\xi)\| \leq c(\xi) \forall h \in \mathbb{R}$ .

Since  $T + S$  is self-adjoint, it is closed. Therefore, the domain  $\mathcal{D}$  with the graph norm<sup>12</sup>  $\|\cdot\|_{T+S}$  forms a Banach Space. Now, by applying the Uniform Boundedness Principle 6.7 to the family  $u_h : (\mathcal{D}, \|\cdot\|_{T+S}) \rightarrow \mathbb{H}$ , there is  $C > 0$  so that

$$\|u_h(\xi)\| \leq C\|\xi\|_{T+S} \quad \forall h \in \mathbb{R}, \xi \in \mathcal{D}.$$

Now, introduce the map for each fixed  $\xi \in \mathcal{D}$ ,

$$\mathbb{R} \ni t \mapsto \xi_t := e^{-it(T+S)}\xi \in (\mathcal{D}, \|\cdot\|_{T+S}).$$

Then using the fact that  $e^{it(T+S)}$  is unitary as well, we have

$$\begin{aligned} \|\xi_t - \xi_s\|_{T+S}^2 &= \|\xi_t - \xi_s\|^2 + \|(T+S)\xi_t - (T+S)\xi_s\|^2 \quad (\text{from the definition of graph norm}) \\ &= \|e^{-it(T+S)}\xi - e^{-is(T+S)}\xi\|^2 + \|(T+S)e^{-it(T+S)}\xi - (T+S)e^{-is(T+S)}\xi\|^2 \\ &= \|e^{it(T+S)}e^{-it(T+S)}\xi - e^{it(T+S)}e^{-is(T+S)}\xi\|^2 \\ &\quad + \|(T+S)e^{it(T+S)}e^{-it(T+S)}\xi - (T+S)e^{it(T+S)}e^{-is(T+S)}\xi\|^2 \\ &= \|\xi - e^{-it(s-t)(T+S)}\xi\|^2 + \|(T+S)\xi - e^{-it(s-t)(T+S)}(T+S)\xi\|^2 \end{aligned}$$

This converges to 0 as  $s \rightarrow t$ , that is, the map  $t \mapsto \xi_t$  into  $(\mathcal{D}, \|\cdot\|_{T+S})$  is continuous. Thus, for fixed  $t \in \mathbb{R}$ , the compactness of the interval  $[-|t|, |t|]$  imply that

$$J_{\xi,t} = \{\xi_s : |s| \leq |t|\}$$

is a compact set in  $(\mathcal{D}, \|\cdot\|_{T+S})$  as continuous images of compact set is again compact. Hence  $J_{\xi,t}$  is totally bounded in  $(\mathcal{D}, \|\cdot\|_{T+S})$  as compactness implies totally boundedness. Therefore, the triangular inequality and the above uniform boundedness conclude that the restriction to the continuous family of linear operators  $u_h : J_{\xi,t} \rightarrow \mathbb{H}$  converges to 0 uniformly as  $h \rightarrow 0$ . That is  $\max_{|s| \leq |t|} \|u_h(\xi_s)\| \rightarrow 0$  as  $h \rightarrow 0$ .

Let  $h = t/n$  and note that  $(n-1-j)/n \leq 1$ . Thus, by 2. above we have,

$$\begin{aligned} \left\| (e^{-itT/n}e^{-itS/n})^n \xi - e^{-it(T+S)}\xi \right\| &\leq \max_{|s| \leq |t|} \|n [e^{-itT/n}e^{-itS/n} - e^{-it(T+S)/n}] e^{-is(T+S)}\xi\| \\ &\leq |t| \max_{|s| \leq |t|} \left\| \frac{1}{h} [e^{-ihT}e^{-ihS} - e^{-ih(T+S)}] e^{-is(T+S)}\xi \right\| \\ &= |t| \max_{|s| \leq |t|} \|u_h(\xi_s)\| \end{aligned}$$

which goes to 0 as  $h \rightarrow 0$ , that is, as  $n \rightarrow \infty$ . This implies

$$(e^{-itT/n}e^{-itS/n})^n \xi \rightarrow e^{-it(T+S)}\xi, \quad \forall \xi \in \mathcal{D}.$$

Since the involved operators are unitary, this convergence extends to the closure of  $\mathcal{D}$ , that is, it holds on  $\mathbb{H}$ .  $\square$

<sup>12</sup>Graph norm of  $T$  on  $\text{dom}T$  is  $\|\xi\|_T := (\|T\xi\|^2 + \|\xi\|^2)^{1/2}$

## 2.8 QUBO Problems

Until now, we have seen most of the mathematics required to describe QAOA. However, the question remains: how to formulate the optimization problem such that it can run on quantum computers? To answer this question, one needs to understand two things: 1. Ising Models and 2. Quadratic unconstrained binary optimization (QUBO) problems<sup>13</sup>.

**Ising Models**[55] provide a framework for representing a physical system using a regular lattice arrangement of molecules. This representation is not limited to molecules; in this thesis, we focus on the lattice arrangement of qubits. Each node in the lattice is assigned a two-valued variable. Depending on whether the variable takes the value  $+1$  or  $-1$ , the molecule at that node is said to have spin up or spin down<sup>14</sup>. This two-valued variable is referred to as the **spin**  $\sigma_i$  for node  $i$ . We assume that only neighboring spins interact with each other. A **configuration** of the lattice refers to a specific set of values assigned to all the spins. If there are  $N$  nodes, there will be  $2^N$  possible configurations.

The optimal lattice configuration is given by[71]

$$\arg \min_{\sigma} (\sigma J \sigma + h^T \sigma)$$

where  $\sigma_i \in \{-1, +1\}$ ,  $J$  is an operator that describes the interactions, and  $h$  is called the external magnetic field. The following Hamiltonian represents the Ising objective function:

$$H_I = \sum_{i,j} J_{i,j} \sigma_i^z \sigma_j^z + \sigma_i h_i \sigma_i^z$$

where  $\sigma_i^z$  is the Pauli Z operator acting of qubit  $i$ .

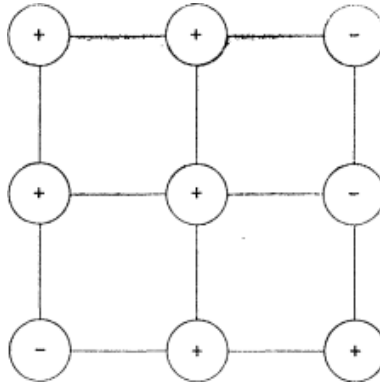


Figure 5: A possible configuration of a finite square lattice.

What physicists call the Ising model, mathematicians call QUBO problem[37]. In a QUBO problem[72], the vector of unknowns  $\mathbf{x} = (x_1, \dots, x_n)$  is represented by decision variables taking discrete binary values, so that  $x \in \{0, 1\}^n$ . Moreover, a QUBO problem is defined by a square symmetric matrix  $\mathbf{Q} \in \mathbb{R}^{n \times n}$ . Given the cost function

$$C(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} = \sum_{i,j=1}^n Q_{ij} x_i x_j, \quad (29)$$

<sup>13</sup>It is also known as Unconstrained Binary Quadratic Programming (UBQP)

<sup>14</sup>Other interpretations are also possible, but for our discussion, this interpretation suffices.

a QUBO problem aims to find the optimal vector  $x^*$  such that

$$x^* = \arg \min_{\mathbf{x} \in \{0,1\}^n} C(\mathbf{x}) \quad (30)$$

QUBO can also be defined as maximization problems instead of minimization ones by simply inverting the sign of the cost function  $C(\mathbf{x})$ .

QUBO instances can also be seen as a one-to-one correspondence with Ising models[37]. Ising problems replace the original QUBO variables  $x \in \{0, 1\}^n$  with Ising variables  $\sigma \in \{-1, 1\}^n$ , such that  $\sigma_i = 2x_i - 1$  for  $i = 1, \dots, n$ . The final Ising Hamiltonian, which depends on  $\sigma$ , is equivalent to Eq. 29 except for a constant irrelevant to the optimization. A more detailed explanation of the relationship between QUBO and Ising models can be found in [73], [74].

## 2.9 Variational Quantum Algorithms

The quantum computers of the current generation are called NISQ devices [12]. NISQ devices have limited number of qubits with short lifespans and high error rate, making it challenging to run a quantum algorithm on these devices successfully. Variational Quantum Algorithms (VQAs) [40] provide the quantum advantage on NISQ devices. VQAs are optimization-based approaches that are similar to neural networks in machine learning. They use a hybrid approach, by running the parameterized quantum circuit on the quantum computers and uses classical computers to optimize the parameters. This approach keeps the quantum circuits short, mitigating the noise issue. It also faces challenges like accuracy and efficiency.

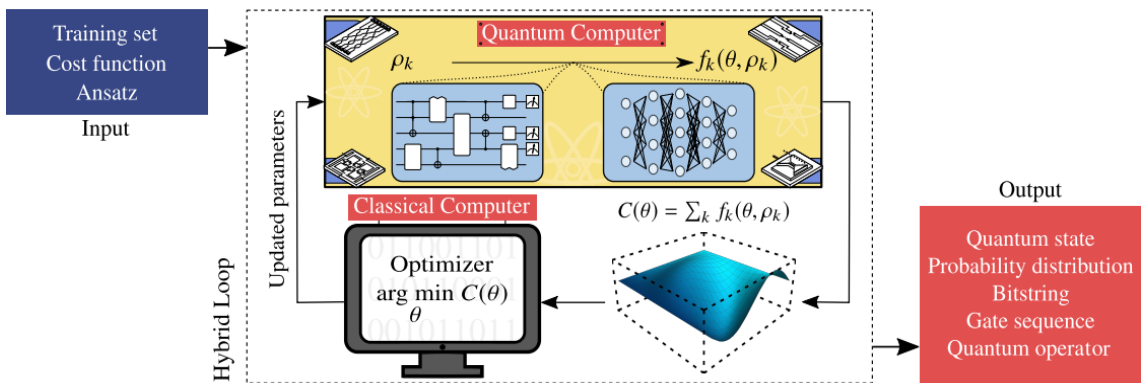


Figure 6: Schematic diagram of VQA.

VQAs have two main components: 1. cost (or loss) function and 2. ansatz. VQAs use a quantum computer to estimate the cost function  $C(\theta)$  while leveraging the power of classical optimizers to find the parameter  $\theta$ . Let's look at each component in more detail now.

1. The cost (or loss) function encodes the solution to an optimization problem. Without the loss of generality, the cost function can be written as

$$C(\theta) = f(\{\rho_k\}, \{O_k\}, U(\theta)), \quad (31)$$

where  $f$  is a function determined by the task at hand,  $U(\theta)$  is the parametrized unitary,  $\{\rho_k\}$  are the input states, and  $\{O_k\}$  are set of observables.

The cost function should have certain desired properties. First, it must be *faithful*, meaning that the minimum of  $C(\theta)$  corresponds to the problem's solution. Second, it must be possible to *efficiently* estimate  $C(\theta)$  by making measurements on quantum computers and performing post-processing on classical computers<sup>15</sup>. Third, it must be *operationally meaningful*, such that smaller cost values indicate better solutions. Finally, it should be *parameterizable*.

2. Ansatz can be seen as a parametrized quantum circuit. The structure of ansatz generally depends on the problem at hand. In some cases, problem-specific information can be used to design an ansatz. These are called *problem-inspired* ansätze. However, generic ansatz architectures also exist. For the cost function 31, the parameters  $\theta$  can be encoded in a unitary operator  $U(\theta)$  that is applied to the input states of the quantum circuit.  $U(\theta)$  can be generically expressed as the product of  $L$  unitaries applied sequentially

$$U(\theta) = U_L(\theta_L) \dots U_2(\theta_2)U_1(\theta_1)$$

Each individual  $U_i(\theta_i)$  may also contain unparameterized components.

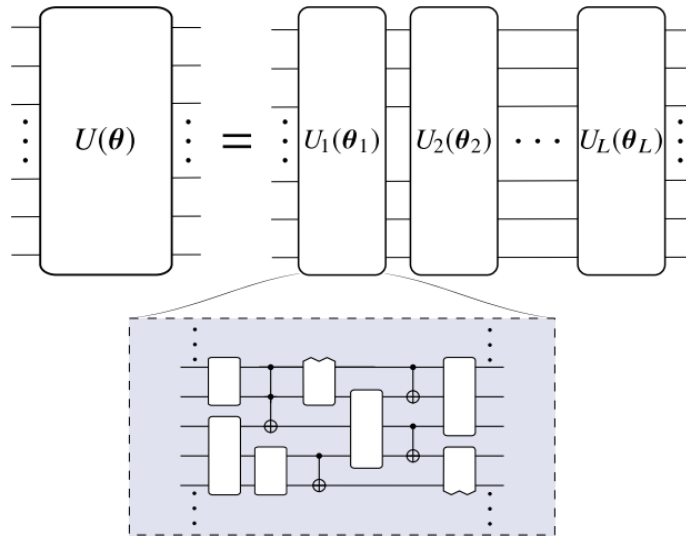


Figure 7: Schematic diagram of an ansatz.

VQAs rely on variational theory [75]. **Variational theory** is a method for evaluating and improving predictions about the shapes of wave functions. It is used to calculate the lowest expectation value of a specific observable, typically the ground state energy, within a trial wave function. This trial wave function is called variational because it is parameterized by a set of values used to fit a generic wave function to the system and find the minimum expectation value. This process is defined in terms of a Hamiltonian  $\hat{H}$  and a trial wave function  $|\psi_k\rangle$  to determine the ground state energy of the system,  $E_0$ , which is bounded as follows:

$$E_0 \leq \frac{\langle \psi | \hat{H} | \psi \rangle}{\langle \psi | \psi \rangle} \quad (32)$$

<sup>15</sup>The implicit assumption is that it should not be efficiently computable on classical computers.

Variational algorithms aim to minimize the Hamiltonian's expectation value by finding a parametrized  $|\psi_k\rangle$ . This is done by approximating the eigenvector  $|\psi_k\rangle$  with the lowest eigenvalue,  $E_0$ , by iteratively improving the ansatz. The ansatz consists of the initial trial wave function and the first set of parameters. How these parameters are chosen depends on the context of the problem to be solved.

The choice to form an ansatz as a variational problem allows the use of quantum computers. The optimization problem can be encoded as a Hamiltonian can be written as the sum of unitary operators.

As mentioned before, the optimization of the parametrized quantum circuit happens using classical algorithms. The optimization problem associated with VQAs is itself NP-hard problem [76]. Other challenges include the stochastic environment due to finite measurement possibilities, hardware noise, and barren plateaus [77]. This led to the development of quantum-aware optimizers. However, the choice of the optimal optimizer is a topic of active debate, therefore, we will just give certain examples without going into details. Classical optimizers that are used in VQAs can be broadly classified:

1. Based on Gradient descent methods. Some examples include Adam [78], Adaptive Optimizer [79], Stochastic Gradient for hybrid approach [80].
2. Other methods. Some examples include Meta-learning [81], Simultaneous Perturbation Stochastic Approximation [82]

## 2.10 QAOA

The QAOA is a quantum algorithm that can be used to find approximate solutions for COPs [8]. It was first introduced as a VQA capable to find approximate solutions to the MaxCut problem, and suitable to be run on NISQ devices. Before explaining the algorithm, we need some terminology.

### Definition 2.51: Objective Function

Let

$$C(z) = \sum_{\alpha=1}^m C_{\alpha}(z)$$

be the objective function of the COP which is specified by  $n$  bits and  $m$  clauses.  $z = z_1 z_2 \cdots z_n$  is the bit string.  $C_{\alpha}(z) = 1$  if  $z$  satisfies clause  $\alpha$  and 0 otherwise.

We have defined objective function as Boolean satisfiability problem (B-SAT) [83]. By definition objective function is an operator with non-negative values. Therefore, it is diagonal in the computational basis vector  $|z\rangle$  in a  $2^n$ -dimensional Hilbert space. This means the objective function can also be defined in terms of the unitary operator.

**Definition 2.52: Cost Hamiltonian**

Let  $C$  be the objective function in  $\mathcal{L}(\mathbb{H})$ . Then  $C$  is called **cost Hamiltonian** and using Postulate 4 it can be defined as a unitary operator

$$\hat{U}_C(\gamma) = e^{-i\gamma C} = \prod_{\alpha=1}^m e^{-i\gamma C_\alpha}$$

where  $\gamma$  is a variational parameter that lie between 0 and  $2\pi$ .

**Definition 2.53: Mixer Hamiltonian**

Let  $B$  be defined as the sum of all single-bit Pauli X ( $\sigma^x$ ) operators

$$B = \sum_{j=1}^n \sigma_j^x = \sigma_1^x \otimes I^{\otimes(n-1)} + I_1 \otimes \sigma_2^x \otimes I^{\otimes(n-2)} + \dots + I^{\otimes(n-1)} \otimes \sigma_n^x$$

Then  $B$  is an irreducible self-adjoint operator and is called **mixer Hamiltonian**. Using Postulate 4 one can define  $B$  in terms of a unitary operator

$$\hat{U}_B(\beta) = e^{-i\beta B} = \prod_{j=1}^n e^{-i\beta \sigma_j^x}$$

where  $\beta$  is a variational parameter which runs from 0 to  $\pi$ .

These layers are analogous to the exponentiated operators on the right-hand side of Eq. (20). However, instead of following predefined  $f$  and  $g$  functions, the parameters  $\gamma_k$  and  $\beta_k$  are trained variationally. In this sense, QAOA can be regarded as a discretized version of the Adiabatic Algorithm and a special case of VQE.

**Definition 2.54: Problem Hamiltonian**

Let  $H(t) \in \mathcal{L}(\mathbb{H})$  be a time-dependent Hamiltonian defined as the convex combination of the Mixer and Cost Hamiltonian

$$H(t) = (1 - s(t))B + s(t)C$$

where  $s(0) = 0$  and  $s(T) = 1$  with  $T$  being the total running time.  $H(t)$  is called the **problem Hamiltonian**.

QAOA uses Trotterization (2.50) so that NISQ devices can be used to solve the optimization problem. As NISQ devices can only maintain the state of a given qubit in the range of a microsecond, it is imperative to truncate the adiabatic algorithm. This is what QAOA is: a truncated version of an adiabatic algorithm for NISQ devices. This truncation gives us the approximate result that is the whole objective of QAOA.

**Definition 2.55: Truncated Hamiltonian**

Let  $H$  be the problem Hamiltonian. Then the truncated Hamiltonian in terms of unitary operator is

$$U(T) \approx \prod_{k=1}^p \exp[-iH(k\Delta t)\Delta t]$$

where  $U(T)$  represents the evolution operator from 0 to  $T$ . Using Trotterization (2.50), for two non-commuting operators it becomes:

$$U(T) \approx \prod_{k=1}^p \exp[-i(1 - s(k\Delta t))B\Delta t] \exp[-is(k\Delta t)C\Delta t]$$

where  $B$  and  $C$  are mixer and cost Hamiltonian respectively.

Furthermore, time-dependent components are redefined:  $(1 - s(k\Delta t))\Delta t$  as  $\beta_k$  and  $s(k\Delta t)\Delta t$  as  $\gamma_k$  which give us the unitary operator forms of the mixer and cost Hamiltonian respectively.

With  $\beta(= (\beta_1, \beta_2, \dots, \beta_k))$  and  $\gamma(= (\gamma_1, \gamma_2, \dots, \gamma_k))$  as variational parameters, Variational Quantum Algorithm (VQA) [40] can be employed to solve for  $\beta$  and  $\gamma$  which in-return solve the optimization problem.

**Definition 2.56: QAOA Ansatz**

Let  $|s\rangle$ , the *initial state*, be the uniform superposition over computational basis states and  $z$  as defined in 2.52

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle.$$

Then the QAOA ansatz is formed by alternating applying the unitary operators of cost and mixer Hamiltonian on the initial state

$$|\gamma, \beta\rangle = \hat{U}_B(\beta_p)\hat{U}_C(\gamma_p) \cdots \hat{U}_B(\beta_1)\hat{U}_C(\gamma_1) |s\rangle$$

where  $p$  defines the number of iterations.

Note that the initial state is the highest energy eigenstate of the mixer Hamiltonian.

Now we have all the tools to solve and show the convergence in QAOA.

**Theorem 2.57: QAOA**

Consider the objective function in (2.51). Let  $c^* = \max_z C(z)$  be its optimal value and  $F_p(\gamma, \beta)$  be the expectation of cost Hamiltonian in the final state of the QAOA ansatz truncated at  $p$ . Then

$$\lim_{p \rightarrow \infty} \max_{\gamma, \beta} F_p(\gamma, \beta) = c^*$$

*Proof.* By the definition of expectation of a Hamiltonian in a quantum state, we

have

$$F_p(\gamma, \beta) = \langle \gamma, \beta | C | \gamma, \beta \rangle$$

where  $C$  is the cost Hamiltonian and  $|\gamma, \beta\rangle$  is the final quantum state of QAOA ansatz.  $F_p$  is an eigenvalue of the cost function.

Let  $M_p$  be the maximum of  $F_p$  over the angles

$$M_p = \max_{\gamma, \beta} F_p(\gamma, \beta)$$

Due to the Perron-Frobenius theorem (2.40), we know that  $M_p$  will be positive. Therefore, as the value of  $p$  in the QAOA ansatz increases there are only two possibilities:

1. We will find quantum state  $|\gamma, \beta\rangle$  that results in an increase in  $F_p$  (and ultimately in  $M_p$ ).
2. Or we will find quantum state  $|\gamma, \beta\rangle$  that result in a decrease in  $F_p$ . However, one can always choose  $\gamma_p$  and  $\beta_p$  such that  $M_p$  stays the same.

This results in the following inequality

$$M_p \geq M_{p-1}$$

The above inequality along with Adiabatic Algorithm (3) completes the proof

$$\lim_{p \rightarrow \infty} M_p = \max_z C(z)$$

where  $z$  is quantum state after measurement  $F_p$  as per Postulate (2).  $\square$

Now we will bring everything together and define the QAOA algorithm[47]:

#### Algorithm 4: QAOA Algorithm

1. Let  $\hat{H}_C$  be the cost Hamiltonian such that its highest energy state encodes the solution to the optimization problem. Let  $\hat{H}_B$  mixer Hamiltonian that does not commute with  $\hat{H}_C$ .
2. Initialize the quantum circuit in the state  $|s\rangle$  which corresponds to the highest energy state of  $\hat{H}_B$ :

$$|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \quad (33)$$

where  $n$  is the number of qubits.

3. Construct the ansatz and apply the unitaries:

$$\hat{U}_C(\gamma) = e^{-i\gamma\hat{H}_C} = \prod_{i=1, j < i}^n R_{Z_i Z_j}(-2\omega_{ij}\gamma) \quad (34)$$

$$\hat{U}_B(\beta) = e^{-i\beta\hat{H}_B} = \prod_{i=1}^n R_{X_i}(2\beta) \quad (35)$$

where  $\gamma$  and  $\beta$  are variational parameters,  $R_X$  and  $R_Z$  are the rotation gate and  $R_Z$  is between two  $CNOT$  gates.

- Let  $p \geq 1$  be the number of QAOA layers. Now, initialize the  $2p$  variational parameters  $\gamma = (\gamma_1, \gamma_2, \dots, \gamma_p)$  and  $\beta = (\beta_1, \beta_2, \dots, \beta_p)$  with  $\gamma_k \in [0, 2\pi)$  and  $\beta_k \in [0, \pi)$  for  $k = 1, \dots, p$ . The final stage output of the circuit is

$$|\psi_p(\gamma, \beta)\rangle = e^{-i\beta_p \hat{H}_B} e^{-i\gamma_p \hat{H}_C} \dots e^{-i\beta_1 \hat{H}_B} e^{-i\gamma_1 \hat{H}_C} \quad (36)$$

- Calculate the expectation value of the  $\hat{H}_C$  with respect to the ansatz state  $|\psi_p(\gamma, \beta)\rangle$ .

$$F_p(\gamma, \beta) = \langle \psi_p(\gamma, \beta) | \hat{H}_C | \psi_p(\gamma, \beta) \rangle \quad (37)$$

- Use a classical optimization algorithm to iteratively update  $\gamma$  and  $\beta$  to find the optimal set of parameters  $(\gamma^*, \beta^*)$  such that the expectation value  $F_p(\gamma, \beta)$  is maximized:

$$(\gamma^*, \beta^*) = \arg \max_{\gamma, \beta} F_p(\gamma, \beta) \quad (38)$$

At the end of the algorithm, the approximation ratio  $\alpha$  will be given by:

$$\alpha = \frac{F_p(\gamma^*, \beta^*)}{C_{\max}}, \quad (39)$$

and the state  $|\psi_p(\gamma^*, \beta^*)\rangle$  will encode the solution to the optimization problem. It is worth noting that QAOA is often initialized with the mixer Hamiltonian's greatest energy eigenstate rather than its ground state. Nonetheless, the adiabatic theorem is still valid in such instances.

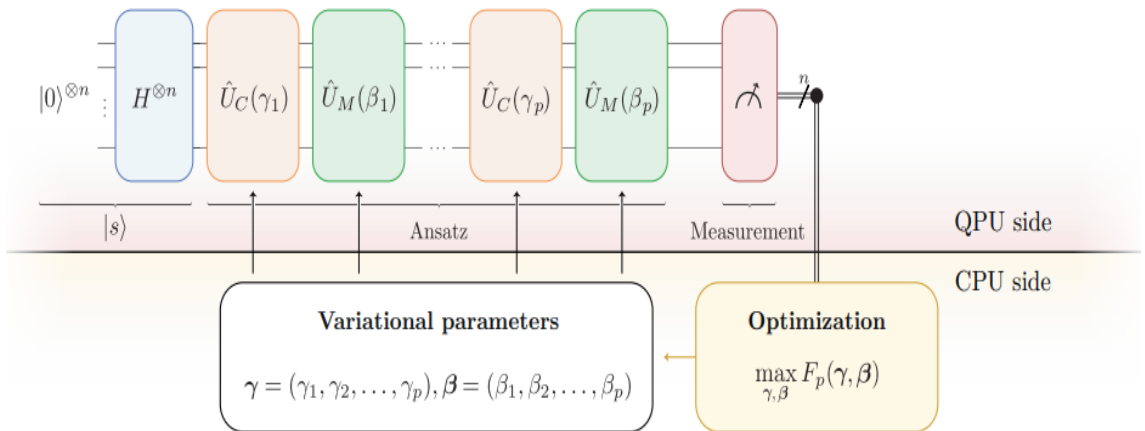


Figure 8: QAOA workflow with  $p$  layers

Now let's have a look at what exactly happens when the unitary operator of the cost Hamiltonian is applied on a 2-qubit system:

**Example 5: Implementing  $U_C(\gamma)$  on 2-qubits**

For a  $U_C(\gamma)$  gate, we should expect the result:

$$\begin{aligned}
 |x_1x_2\rangle &\xrightarrow{U_C(\gamma)} e^{-i\gamma C_2} e^{-i\gamma C_1} |x_1x_2\rangle \\
 &= e^{-i\gamma C_2} e^{-i\gamma C_1} (\alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + \alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle) \\
 &= e^{-i\gamma C_2(00)} e^{-i\gamma C_1(00)} \alpha_0\beta_0 |00\rangle + e^{-i\gamma C_2(01)} e^{-i\gamma C_1(01)} \alpha_0\beta_1 |01\rangle \\
 &\quad + e^{-i\gamma C_2(10)} e^{-i\gamma C_1(10)} \alpha_1\beta_0 |10\rangle + e^{-i\gamma C_2(11)} e^{-i\gamma C_1(11)} \alpha_1\beta_1 |11\rangle \\
 &= e^{-i\gamma^0} e^{-i\gamma^0} \alpha_0\beta_0 |00\rangle + e^{-i\gamma^0} e^{-i\gamma^0} \alpha_0\beta_1 |01\rangle \\
 &\quad + e^{-i\gamma^0} e^{-i\gamma^1} \alpha_1\beta_0 |10\rangle + e^{-i\gamma^1} e^{-i\gamma^1} \alpha_1\beta_1 |11\rangle \\
 &= \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + e^{-i\gamma} \alpha_1\beta_0 |10\rangle + e^{-2i\gamma} \alpha_1\beta_1 |11\rangle \\
 &= \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + R(\gamma)\alpha_1\beta_0 |10\rangle + R^2(\gamma)\alpha_1\beta_1 |11\rangle
 \end{aligned}$$

### 3 Quantum Job Shop Scheduling Problem

#### 3.1 Problem representation

As previously stated, job shops are scheduling problems in which a set of machines perform jobs. Each job consists of an ordered list of operations with each operation requiring a specified machine with a defined processing time. Here, we use the following formulation of JSSP[45].  $\mathcal{J} = \{j_1, \dots, j_J\}$  represents the set of  $J$  jobs, each with  $O_j$  operations.  $\mathcal{O}_j = \{o_{j1}, \rightarrow \dots \rightarrow o_{jO_j}\}$  represents an ordered list of operations for job  $j$ . Each operation must be performed on a specific and different machine from a set of  $M$  machines,  $\mathcal{M} = \{m_1, \dots, m_M\}$ , and only one operation can be performed by a machine at a given time. The objective is to minimize the makespan.

To solve JSSP using QAOA, we have to formulate it as a QUBO problem 2.8. In particular, we need to define the JSSP constraints as binary clauses and represent each operation as a time-indexed binary variable. The time-indexed binary variable is defined as

$$x_{k,t} = \begin{cases} 1 & \text{if operation } o_k \text{ starts at time } t \\ 0 & \text{otherwise} \end{cases} \quad (40)$$

where  $t$  is bounded by a makespan  $T$ <sup>16</sup>, and the index  $k$  represents the position of an operation in a list of all operations of all the jobs:

$$\left[ \underbrace{o_{11}, \dots, o_{1O_1}}_{j_1}, \underbrace{o_{21}, \dots, o_{2O_2}}_{j_2}, \dots, \underbrace{o_{J1}, \dots, o_{JO_J}}_{j_J} \right] = \left[ \underbrace{o_1, \dots, o_{k_1}}_{j_1}, \underbrace{o_{k_1+1}, \dots, o_{k_2}}_{j_2}, \dots, \underbrace{o_{k_{J-1}+1}, \dots, o_{k_J}}_{j_J} \right] \quad (41)$$

<sup>16</sup>the makespan  $T$  has to be estimated but for this thesis we assume it to be given.

Let us now define different JSSP constraints as binary variables. Let  $x$  be the input binary string.

The first constraint is that the operation must start only once. This can be expressed as:

$$h_1(x) = \sum_k \left( \sum_t x_{k,t} - 1 \right)^2 \quad (42)$$

The second constraint is that at any time only one operation can run on a given machine. This can be expressed as:

$$h_2(x) = \sum_m \left( \sum_{\substack{k \neq k' \\ 0 \leq t, t' \leq T \\ 0 < t' - t < \ell_k}} x_{k,t} x_{k',t'} \right) \quad (43)$$

where  $\ell_k$  is the processing time of operation  $k$ .

The third constraint is that the original order of the operations is kept for all the operations for every job in a given instance. This can be expressed as:

$$h_3(x) = \sum_{n=1}^J \left( \sum_{\substack{k_{n-1} < k < k_n \\ t + \ell_k > t'}} x_{k,t} x_{k+1,t'} \right) \quad (44)$$

If the values of all three objectives are equal to zero, all of the constraints are satisfied, indicating that there is a feasible schedule with a makespan less than or equal to  $T$ .

### 3.2 QAOA for the JSSP

Now that we have the QUBO formulation of the JSSP, we can do a simple transformation to derive the Cost Hamiltonian that is required by QAOA.

As seen in the section on QAOA, if we can define the problem objective function as a sum of a finite number of constraints  $C_\alpha(x)$  where  $x \in \{0, 1\}^R$  is a binary string of length  $R$ ,

$$C(x) = \sum_{\alpha} C_{\alpha}(x) \quad (45)$$

then QAOA can be used to find an approximate solution. To do that we need to convert the constraints into quantum Hamiltonians by replacing the binary variables  $x_r$ ,  $r \in \{1, 2, \dots, R\}$ , with spin variables  $s_r$ ,

$$x_r = \frac{1 - s_r}{2} \quad (46)$$

Clearly,  $s_r = 1 \implies x_r = 0$  and  $s_r = -1 \implies x_r = 1$ . Therefore,  $s_r$  can be represented by the Pauli-Z matrix  $\sigma_r^z$ .

As a result, we obtain a cost Hamiltonian

$$\hat{H}_C = C(\sigma^z) \quad (47)$$

The goal of QAOA is to find optimal parameters  $\gamma^*$  and  $\beta^*$ , so that the expected value

$$F_P(\gamma^*, \beta^*) = \langle \psi_p(\gamma^*, \beta^*) | \hat{H}_C | \psi_p(\gamma^*, \beta^*) \rangle \quad (48)$$

is maximized (or minimized).

Finally, to solve JSSP using QAOA we have to convert the binary variables to spin variables in constraints (42) to (44) using the formula (46). We can then consider the objective function represented by the formula

$$\hat{H}_C(\sigma^z) = h_1(\sigma^z) + h_2(\sigma^z) + h_3(\sigma^z) \quad (49)$$

as the cost Hamiltonian (47). The JSSP solution can then be calculated by embedding this cost Hamiltonian into the QAOA.

## 4 Implementation and Result

Due to the high computational demands of quantum simulation, this thesis only employs a toy example. The example consists of two machine, three jobs, each with two operations, where each operation requires one unit of processing time. The makespan of the optimal schedule is three units. The toy example was executed on a single node of the HPC system of the GWDG<sup>17</sup> which had 384 GB of RAM. The large memory requirement is due to the exponential growth of the quantum states as the number of qubits increases, as well as to store the intermediate results. The QAOA algorithm took approximately 3 minutes to compute the optimal schedule and used 15 qubits to simulate the problem. The QAOA parameters,  $\beta$  and  $\gamma$ , were initialized to 0, and the parameter  $p$  was set to 1.

We use the **QASM** simulator provided by Python package **qiskit** [84], [85], which is developed by IBM. Due to unavailability of a real quantum computers, quantum simulators are used to simulate the quantum computers on the classical computers. Qiskit is the most popular choice to simulate quantum algorithm in Python.

Figure 9 presents the optimal schedule as determined by the QAOA algorithm. The Gantt chart illustrates the sequence in which the jobs should run to achieve the optimal makespan. Figure 10 displays the disjunctive graph corresponding to the schedule.

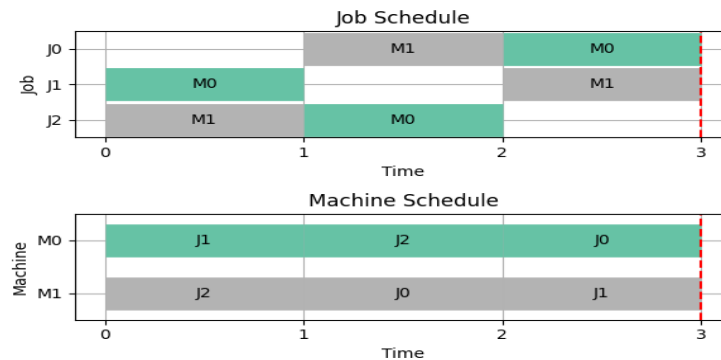


Figure 9: Job shop schedule using QAOA algorithm.

<sup>17</sup>Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

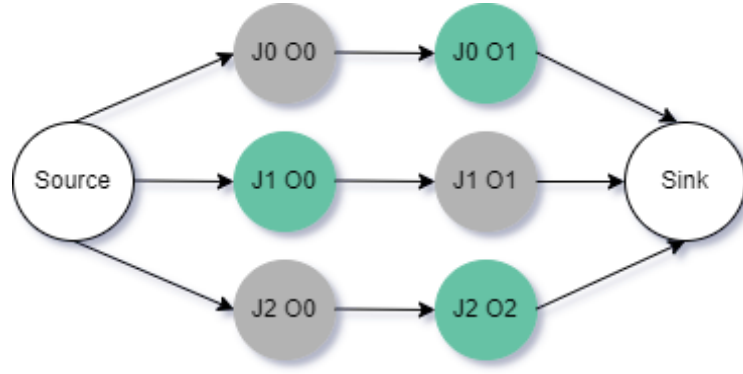


Figure 10: Disjunctive graph of the schedule. J represents the job, and O represents the operation. The numerical value after J or O represents the first or the second job or operation. The nodes with the same colour run on the same machine.

To check the results obtained using the quantum algorithm, we used the Python package **docplex.cp**, developed by IBM. Figure 11 illustrates the optimal schedule, which is identical to the one produced by QAOA which proves that the schedule created by QAOA is in fact the optimal schedule. It took less than one second to find the optimal schedule on a laptop with 8GB RAM.

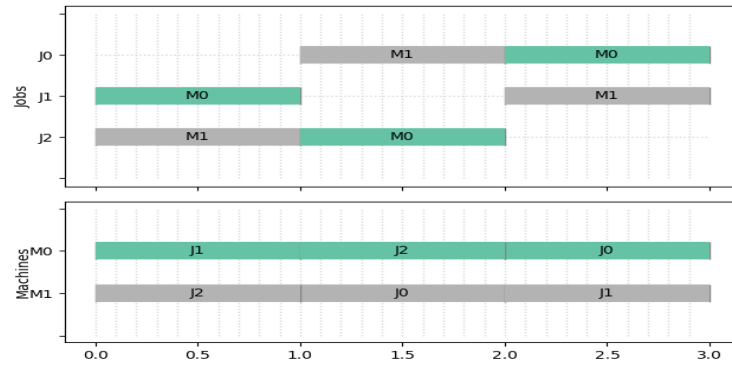


Figure 11: Job shop schedule using classical algorithm.

The attempts to solve a problem with three machine, three jobs, each with three operations, didn't result in convergence to the optimal result even though it took more than 5 hours<sup>18</sup> to find the solution. The QAOA algorithm was using 33 qubits to simulate this problem. However, the Python class **QasmSimulator** of qiskit saves probabilities of 32 qubits by default which explains why the convergence of this problem didn't result in finding the solution. Another reason is the high memory utilization to simulate qubits of classical systems [86].

The entire code is available on [gitlab](#) and is inspired by [SEQUOIA-Demonstrators](#).

## 5 Conclusion and Discussion

This thesis presents the successful implementation of the QAOA to solve the JSSP. We also establish a mathematical foundation by defining the underlying theorems on

<sup>18</sup>in one case around 11 hours

which QAOA is based, specifically the adiabatic algorithm and trotterization. Additionally, we examine the mathematical entities (matrices) that satisfy the AA and prove the Frobenius-Perron theorem, which is an essential result from the perspective of the uniqueness of the solution. The classical branch-and-bound algorithm for solving JSSP is also discussed, along with a review of the postulates of quantum mechanics that forms the foundation of quantum computing.

Our results indicate that, even for a small scheduling problem, classical algorithms solve the JSSP significantly faster than the hybrid QAOA algorithm. The following are some of the reasons for this discrepancy in the result:

- The use of quantum simulators on classical computers. Quantum algorithms are designed to leverage the properties of the quantum hardware which is still in the nascent stage with limited numbers of real qubits available.
- Classical algorithms are designed to take advantage of the parallelization through multi-threading or multi-processing. As discussed in [86], running a quantum algorithm on more than one computer node is still an area field of research. The same paper also highlights that quantum algorithm run significantly faster on a GPU as compared to CPUs.
- The limited availability of the quantum-aware optimizers that are part of VQAs. The advancements in the optimizers will enable rapid and efficient training of the parameters and help avoid local minima [87].
- The present quantum error correction methods are another factor. Improvement in the error correction methods in the hybrid approach that will lead to more accurate solutions at a faster rate.

Given the ongoing research in quantum computing, there is a high possibility that some, if not all, of the issues mentioned above will be tackled, improving the viability of quantum computers.

## 6 Appendix

The appendix contains results that were used in proving theorems and lemmas in this thesis but didn't prove.

### Theorem 6.1

Let  $\|\cdot\|$  be a matrix norm. Then for  $A \in M_n$

$$\rho(A) \leq \|A\|$$

### Theorem 6.2

Let  $\|\cdot\|$  be a matrix norm on  $M_n$ . Then for  $A \in M_n$

$$\rho(A) = \lim_{k \rightarrow \infty} \|A^k\|^{1/k}$$

**Theorem 6.3**

Let  $A \in M_n$  and suppose that  $A \geq 0$ . If the row sums of  $A$  are constant, then  $\rho(A) = \|A\|_\infty$

**Theorem 6.4: Schur Triangularization theorem**

Given  $A \in M_n$  with eigenvalues  $\lambda_1, \dots, \lambda_n$  in any prescribed order, there is a unitary matrix  $U \in M_n$  such that

$$U^*AU = T = [t_{ij}]$$

is upper triangular, with diagonal entries  $t_{ii} = \lambda_i$ ,  $i = 1, \dots, n$ . That is, any square matrix  $A$  is unitarily equivalent to a triangular matrix whose diagonal entries correspond to the eigenvalues of  $A$  in a predetermined order. Furthermore, if  $A \in M_n(\mathbb{R})$  and if all eigenvalues of  $A$  are real, then  $U$  may be chosen to be real and orthogonal.

**Theorem 6.5**

Let  $A \in M_n$  and suppose  $A > 0$ . Then

$$\lim_{m \rightarrow \infty} [\rho(A)^{-1}A]^m = L$$

where  $L = xy^T$ ,  $Ax = \rho(A)x$ ,  $A^T y = \rho(A)y$ ,  $x > 0$ ,  $y > 0$ , and  $x^T y = 1$ .

**Corollary 6.6**

If  $A \in M_n$  and  $A > 0$ , then  $L = \lim_{m \rightarrow \infty} [\rho(A)^{-1}A]^m$  is a positive matrix of rank 1.

**Theorem 6.7: Uniform Boundedness Principle**

Let  $\mathcal{B}$  be Banach space,  $\mathcal{N}$  be a normed vector space, and  $B(\mathcal{B}, \mathcal{N})$  be the set of bounded linear operators from  $\mathcal{B}$  to  $\mathcal{N}$ . Then for any family of operators  $\{T_\alpha\}_{\alpha \in J}$  in  $B(\mathcal{B}, \mathcal{N})$  so that, for each  $\xi \in \mathcal{B}$ ,

$$\sup_{\alpha \in J} \|T_\alpha \xi\| < \infty$$

satisfies  $\sup_{\alpha \in J} \|T_\alpha\| < \infty$  for some indexing set  $J$ .

## References

- [1] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, Jun 1982.
- [2] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439:553 – 558, 1992.

- [3] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.
- [4] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.
- [5] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, Dec 2019.
- [6] Y. Cao, J. Romero, and A. Aspuru-Guzik. Potential of quantum computing for drug discovery. *IBM Journal of Research and Development*, 62(6):6:1–6:20, 2018.
- [7] Alán Aspuru-Guzik, Anthony D. Dutoi, Peter J. Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, 2005.
- [8] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.
- [9] Kenneth R. Baker and Dan Trietsch. *Principles of sequencing and scheduling*. Wiley series in operations research and management science. John Wiley, Hoboken, 2nd ed edition, 2019.
- [10] M. R. Garey, D. S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 2024/07/21/ 1976. Full publication date: May, 1976.
- [11] Yu.N. Sotskov and N.V. Shakhlevich. Np-hardness of shop-scheduling problems with three jobs. *Discrete Applied Mathematics*, 59(3):237–266, 1995.
- [12] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.
- [13] Jacek Blazewicz, Moshe Dror, and Jan Weglarz. Mathematical programming formulations for machine scheduling: A survey. *European Journal of Operational Research*, 51(3):283–300, 1991.
- [14] Jacek Błażewicz, Erwin Pesch, and Małgorzata Sterna. The disjunctive graph machine representation of the job shop scheduling problem. *European Journal of Operational Research*, 127(2):317–331, 2000.
- [15] Jacek Błażewicz, Wolfgang Domschke, and Erwin Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1):1–33, 1996.
- [16] A.S. Jain and S. Meeran. Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, 113(2):390–434, 1999.

- [17] Peter Brucker and Bernd Jurisch. A new lower bound for the job-shop scheduling problem. *European Journal of Operational Research*, 64(2):156–167, 1993. Project Management and Scheduling.
- [18] J. Carlier and E. Pinson. Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78(2):146–161, 1994. Project Management and Scheduling.
- [19] R. Haupt. A survey of priority rule-based scheduling. *Operations-Research-Spektrum*, 11(1):3–16, Mar 1989.
- [20] Christian Artigues, Pierre Lopez, and Pierre-Dimitri Ayache. Schedule generation schemes for the job-shop problem with sequence-dependent setup times: Dominance properties and computational analysis. *Annals of Operations Research*, 138(1):21–52, Sep 2005.
- [21] Joseph Adams, Egon Balas, and Daniel Zawack. The shifting bottleneck procedure for job shop scheduling. *Management science*, 34(3):391–401, 1988.
- [22] Ferdinando Pezzella and Emanuela Merelli. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, 120(2):297–310, 2000.
- [23] Ulrich Dorndorf, Erwin Pesch, and Toàn Phan-Huy. Constraint propagation and problem decomposition: a preprocessing procedure for the job shop problem. *Annals of Operations Research*, 115(1):125–145, 2002.
- [24] Robert Johannes Maria Vaessens, Emile Hubertus Leonardus Aarts, and Jan Karel Lenstra. Job shop scheduling by local search. *Informatics Journal on computing*, 8(3):302–317, 1996.
- [25] Jacek Błażewicz, Klaus H Ecker, Erwin Pesch, Günter Schmidt, and Jan Weglarz. *Handbook on scheduling: from theory to applications*. Springer Science & Business Media, 2007.
- [26] Michael L. Pinedo. *Scheduling - Theory, Algorithms, and Systems*. Springer, Berlin, Heidelberg, 2016.
- [27] Egon Balas and Alkis Vazacopoulos. Guided local search with shifting bottleneck for job shop scheduling. *Management science*, 44(2):262–275, 1998.
- [28] Chao Yong Zhang, PeiGen Li, YunQing Rao, and ZaiLin Guan. A very fast ts/sa algorithm for the job shop scheduling problem. *Computers & operations research*, 35(1):282–294, 2008.
- [29] Jelke J Van Hoorn. The current state of bounds on benchmark instances of the job-shop scheduling problem. *Journal of Scheduling*, 21(1):127–128, 2018.
- [30] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 11–20, 1993.

- [31] Lance Fortnow and John Rogers. Complexity limitations on quantum computation. *Journal of Computer and System Sciences*, 59(2):240–252, 1999.
- [32] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191, 1997.
- [33] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution, 2000.
- [34] Dorit Aharonov, Wim Van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review*, 50(4):755–787, 2008.
- [35] Sergio Boixo, Tameem Albash, Federico M Spedalieri, Nicholas Chancellor, and Daniel A Lidar. Experimental signature of programmable quantum annealing. *Nature communications*, 4(1):2067, 2013.
- [36] Travis S Humble, Alex J McCaskey, Ryan S Bennink, Jay Jay Billings, EF DAzevedo, Blair D Sullivan, Christine F Klymko, and Hadayat Seddiqi. An integrated programming and development environment for adiabatic quantum optimization. *Computational Science & Discovery*, 7(1):015006, 2014.
- [37] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2, 2014.
- [38] Davide Venturelli, Dominic JJ Marchand, and Galo Rojo. Quantum annealing implementation of job-shop scheduling. *arXiv preprint arXiv:1506.08479*, 2015.
- [39] Krzysztof Kurowski, Jan Wglarz, Marek Subocz, Rafał Różycki, and Grzegorz Waligóra. Hybrid quantum annealing heuristic method for solving job shop scheduling problem. In *Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part VI 20*, pages 502–515. Springer, 2020.
- [40] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, August 2021.
- [41] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X*, 10:021067, Jun 2020.
- [42] Gavin E Crooks. Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv preprint arXiv:1811.08419*, 2018.
- [43] Matthew Radzihovsky, Joey Murphy, and S Mason. A qaoa solution to the traveling salesman problem using pyquil, 2019.
- [44] Zsolt Tabi, Kareem H El-Safty, Zsófia Kallus, Péter Hága, Tamás Kozsik, Adam Glos, and Zoltán Zimborás. Quantum optimization for the graph coloring problem with space-efficient embedding. In *2020 IEEE international conference on quantum computing and engineering (QCE)*, pages 56–62. IEEE, 2020.

- [45] Krzysztof Kurowski, Tomasz Pecyna, Mateusz Slysz, Rafał Różycki, Grzegorz Waligóra, and Jan Wglarz. Application of quantum approximate optimization algorithm to job shop scheduling problem. *European Journal of Operational Research*, 310(2):518–528, 2023.
- [46] Daniel Stilck França and Raul Garcia-Patron. Limitations of optimization algorithms on noisy quantum devices. *Nature Physics*, 17(11):1221–1227, 2021.
- [47] Kostas Blekos, Dean Brand, Andrea Ceschini, Chiao-Hui Chou, Rui-Hao Li, Komal Pandya, and Alessandro Summer. A review on quantum approximate optimization algorithm and its variants. *Physics Reports*, 1068:1–66, 2024. A review on Quantum Approximate Optimization Algorithm and its variants.
- [48] Reuben Tate, Jai Moondra, Bryan Gard, Greg Mohler, and Swati Gupta. Warm-Started QAOA with Custom Mixers Provably Converges and Computationally Beats Goemans-Williamson’s Max-Cut at Low Circuit Depths. *Quantum*, 7:1121, September 2023.
- [49] Wolfgang Scherer. *Mathematics of Quantum Computing*. Springer Cham, 2020.
- [50] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.
- [51] Jack D Hidary. *Quantum computing: An applied approach*. Springer Nature, Cham, Switzerland, September 2021.
- [52] Henryk Minc. *Nonnegative Matrices*. Wiley-Interscience, 1988.
- [53] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.
- [54] César R. Oliveira. *Intermediate Spectral Theory and Quantum Dynamics*. Birkhäuser Basel, 2008.
- [55] STEPHEN G. BRUSH. History of the lenz-ising model. *Rev. Mod. Phys.*, 39:883–893, Oct 1967.
- [56] David Amaro, Matthias Rosenkranz, Nathan Fitzpatrick, Koji Hirano, and Mattia Fiorentini. A case study of variational quantum algorithms for a job shop scheduling problem. *EPJ Quantum Technology*, 9(1):5, Feb 2022.
- [57] Jiachen Zhang, Giovanni Lo Bianco, and J. Christopher Beck. Solving job-shop scheduling problems with qubo-based specialized hardware. *Proceedings of the International Conference on Automated Planning and Scheduling*, 32(1):404–412, Jun. 2022.
- [58] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Number v. 1 in Algorithms and Combinatorics. Springer, 2003.
- [59] Mauro Dell’Amico and Marco Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41(3):231–252, Sep 1993.

- [60] M. Gen, Y. Tsujimura, and E. Kubota. Solving job-shop scheduling problems by genetic algorithm. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1577–1582 vol.2, 1994.
- [61] N. Jawahar S. G. Ponnambalam and P. Aravindan. A simulated annealing algorithm for job shop scheduling. *Production Planning & Control*, 10(8):767–777, 1999.
- [62] D. Werner. *Funktionalanalysis*. Springer-Lehrbuch. Springer Berlin Heidelberg, 2011.
- [63] Jonas Maziero. Computing partial traces and reduced density matrices. *International Journal of Modern Physics C*, 28(01):1750005, January 2017.
- [64] David P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, 48(9-11):771–783, 2000.
- [65] R. Blatt and C. F. Roos. Quantum simulations with trapped ions. *Nature Physics*, 8(4):277–284, Apr 2012.
- [66] E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409(6816):46–52, Jan 2001.
- [67] J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Phys. Rev. Lett.*, 74:4091–4094, May 1995.
- [68] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, November 1995.
- [69] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, April 2001.
- [70] R.G. Bartle and D.R. Sherbert. *Introduction to Real Analysis*. Wiley, 2000.
- [71] P. Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Elsevier Science, 2014.
- [72] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28(1):58–81, Jul 2014.
- [73] Fred Glover, Gary Kochenberger, and Yu Du. A tutorial on formulating and using qubo models, 2019.
- [74] Apostolos Chalkis, Thomas Kleinert, and Boro Sofranac. Qubo dual bounds via sdp plane projection method, 2023.
- [75] P.W. Atkins and R.S. Friedman. *Molecular Quantum Mechanics*. OUP Oxford, 2011.

- [76] Lennart Bittel and Martin Kliesch. Training variational quantum algorithms is np-hard. *Phys. Rev. Lett.*, 127:120502, Sep 2021.
- [77] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1), November 2018.
- [78] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [79] Jonas M. Kübler, Andrew Arrasmith, Lukasz Cincio, and Patrick J. Coles. An Adaptive Optimizer for Measurement-Frugal Variational Algorithms. *Quantum*, 4:263, May 2020.
- [80] Ryan Sweke, Frederik Wilde, Johannes Meyer, Maria Schuld, Paul K. Faehrmann, Barthélémy Meynard-Piganeau, and Jens Eisert. Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum*, 4:314, August 2020.
- [81] Max Wilson, Rachel Stromswold, Filip Wudarski, Stuart Hadfield, Norm M. Tubman, and Eleanor G. Rieffel. Optimizing quantum heuristics with meta-learning. *Quantum Machine Intelligence*, 3(1):13, Apr 2021.
- [82] J.C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.
- [83] Tasniem Nasser Alyahya, Mohamed El Bachir Menai, and Hassan Mathkour. On the structure of the boolean satisfiability problem: A survey. *ACM Comput. Surv.*, 55(3), mar 2022.
- [84] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023.
- [85] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024.
- [86] Christian Boehme, Lourens van Niekerk, Dhiraj Kumar, Aasish Kumar Sharma, Tino Meisel, and Martin Leandro Paleico. A comparison of hpc based quantum computing simulators using quantum volume. In *(Upcoming paper) Tagungsband der Informatik*, Lecture Notes in Informatics. Gesellschaft für Informatik e.V., 2024.
- [87] Xiu-Zhe Luo, Jin-Guo Liu, Pan Zhang, and Lei Wang. Yao.jl: Extensible, Efficient Framework for Quantum Algorithm Design. *Quantum*, 4:341, October 2020.