QUANTUM ALGORITHMS FOR PRODUCTION AND SCHEDULING OPTIMIZATION

A Thesis by

Joyce Chiam Ziyi

Bachelor of Science, Wichita State University, 2021

Submitted to the Department of Industrial, Systems, and Manufacturing Engineering
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

December 2022

QUANTUM ALGORITHMS FOR PRODUCTION AND SCHEDULING OPTIMIZATION

The following faculty members have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Industrial Engineering.

Saideep Nannapaneni, Committee Chair

Deepak Gupta, Committee Member

Gamal Weheba, Committee Member

Krishna Krishnan, Committee Member

Rajeev Nair, Committee Member

DEDICATION

I dedicate this thesis to my family, friends, research advisor and colleagues who have helped me in various ways throughout the course of my thesis.

ACKNOWLEDGEMENTS

I would like to acknowledge my research advisor, Dr Saideep Nannapaneni, for his continuous support and patience towards me throughout my research experience. I want to also thank Sima Esfandiarpour and Ramkumar Harikrishnakumar for also providing guidance and assistance throughout this journey. Special thanks to Great Plains Industries for providing me a platform to learn so much more about production scheduling in the real world.

The Department of Industrial, Systems, and Manufacturing Engineering of Wichita State University has been extremely helpful in providing a positive and conducive work environment for me to successfully carry out my thesis. I would not have been able to come this far in my education if it weren't for the support and encouragement of the faculty members.

ABSTRACT

Quantum computing has become an emerging field of interest due to its computational efficiency in solving combinatorial optimization problems. This rapid-growing paradigm of computation harnesses the principles of quantum mechanics such as superposition and entanglement, to converge towards an optimum solution faster than classical computation. In our advanced developing era, we are facing challenges of storing and processing exponentially growing data. Quantum computation has proven its value in rectifying such issues by its ability of storing large sizes of data and processing them in a parallel manner. One of the widely used quantum optimization algorithms is the Quantum Approximate Optimization Algorithm (QAOA), a variational hybrid quantum-classical algorithm. In this paper, we use an active learning-based Bayesian optimization approach to identify appropriate quantum circuit parameters. A variety of industrial processes are modeled using combinatorial optimization with the ultimate objective of either the improving overall efficiency or minimizing the total cost. However, these problems are known to be computationally intractable. In the realm of combinatorial optimization, a single-machine scheduling problem with multiple jobs and operations is notoriously a hard problem. In this paper, we discuss the application of QAOA and active learning approach to solve a single-machine scheduling problem, and compare the solution obtained using classical analysis. Another one of the common problems is solving linear systems of equations, in which there are already many existing classical algorithms that can be implemented. A linear system of equations is a vital concept in the realm of science and engineering. As the field of data analytics grow tremendously, the sizes of datasets can be computationally expensive and time consuming. Hence, quantum linear regression algorithms such as the HHL algorithm can supply exponential speedups to solve these questions. To prove the flexibility and breadth of such quantum applications, this paper will also discuss how a quantum linear regression algorithm can be used to solve production scheduling problems.

TABLE OF CONTENTS

Chapter Page

1 INTRODUCTION 1

2 BACKGROUND 6

2.1 *Active Learning and Bayesian Optimization* 6
2.2 *Quantum Approximate Optimization Algorithm (QAOA)* 8
2.3 *Quantum Linear Regression* 10
2.4 *Uncertainty Propagation* 12

3 METHODOLOGY 14
3.1 *Quantum Approximate Optimization Algorithm* 14
3.2 *HHL Algorithm* 16

4 CASE STUDY AND RESULTS 18
4.1 *Case Study 1: Job Shop Scheduling* 18
4.2 *Case Study 1: Production Scheduling* 24
4.2.1 2x2 Example 24
4.2.2 3x3 Example 25
4.2.3 4x4 Example 27
4.3 Uncertainty Propagation 29

5 CONCLUSION 30

BIBLIOGRAPHY 31

LIST OF TABLES

Table		Page
1	Results of Each Acquisition Function for 20 Trials	22
2	Solution Comparison for each Example	28

LIST OF FIGURES

Figure		Page
1	Illustrative quantum circuit of QAOA	9
2	Process flowchart of HHL algorithm	10
3	Framework of HHL Algorithm	11
4	Process flowchart of uncertainty propagation	13
5	Flowchart of QAOA Algorithm	15
6	Frequency of Computing Minimum Value for each Acquisition Function	21
7	Performance of LCB	22
8	Performance of EI	23
9	Performance of PI	23
10	Factory Network of Workstations with Branching Probabilities	27
11	Results of model prediction KDE	29

LIST OF SYMBOLS

α	Alpha
β	Beta
γ	Gamma
ψ	Psi
θ	Theta
μ	Micron
σ	*Sigma*
λ	Lambda
∞	Infinity
$	US Dollar

CHAPTER I

INTRODUCTION

Quantum computing is the new horizon in the computing world that has been a popular topic of discussion. Instead of classical bits, qubits (or qbits) are used to perform computation. It was first introduced by Farhi [1] while Choi [2] provides a good overview of its fundamentals and applications. This advanced development of quantum computing uses the principle of quantum mechanics of these qubits – superposition and entanglement. These phenomena enable high-speed computation and also efficient data storage. The functionality and flexibility of quantum computing also can be applied to a large field of problem sets and compute optimal solutions with minimal effort. One of the widely used quantum optimization algorithm is the Quantum Approximate Optimization Algorithm (QAOA). One of the most significant drivers of the quantum computing industry is the promise of improving combinatorial optimization, such as the MAX-CUT problem [3]. One of the most significant drivers of the quantum computing industry is the promise of improving combinatorial optimization. This paper discusses the application of QAOA on one of the hardest industrial processes to solve in the recent decade – job shop scheduling problem (JSSP). It is a well-known, combinatorial optimization problem in the realm of Industrial Engineering. Such problem is prevalent in almost all manufacturing and production industries. It involves the process of different job assignments to corresponding machines with variating factors impacting the overall schedule and minimize overall production costs.

QAOA is a hybrid quantum-classical variational algorithm that uses a combination of quantum and classical analyses to efficiently solve combinatorial optimization problems. The QAOA algorithm has been used to solve a variety of scheduling problems such as aircraft tail assignment, job-shop scheduling , bike-sharing systems, and wireless scheduling [4, 5, 6]. QAOA uses the quantum gate-based paradigm for analysis. In the quantum gate paradigm (e.g. IBM Q), the quantum computations are carried out through the application of quantum gates (e.g. Pauli X, Y, and Z gates, H, rotation gates, and CNOT). Some of these gates are associated with parameters (e.g. rotation angles in the rotation gates). The performance of the quantum analysis depends on the quantum gate parameters. The quantum analysis in QAOA identifies the best solution for a given set of gate parameters, and the quality of the solutions depends on these gate parameters. The role of the classical analysis in QAOA is in the identification of appropriate gate parameters that result in the optimal solution. Since the gate parameters are continuous variables, a mix of gradient-based and gradient-free classical algorithms have been to identify the optimal gate parameters. In this paper, we seek to demonstrate the application of active learning (sometimes referred to a sequential training data selection) and Bayesian optimization for efficient estimation of quantum gate parameters [7]. The Bayesian optimization approach is useful when optimizing expensive-to-evaluate black-box functions as it intelligently chooses the inputs where the expensive function needs to be evaluated, does not require the calculation of gradients, and can be used for global optimization analysis. In the Bayesian optimization analysis, a Gaussian Process (GP) model is used as a surrogate for the expensive function. The input-output data for training the GP are intelligently selected using an acquisition learning function; the trained GP is later used for optimization analysis. More details regarding Bayesian optimization and active learning are given in Section 2. A similar work was performed by Wang [8], which introduced Quantum Approximate Bayesian Optimization Algorithm (QABOA). The algorithm integrated Bayesian optimization with QAOA but focused on global optimization of continuous variable functions. In this paper, we focus on combinatorial optimization problems (discrete variables).

In this paper, a special case of a JSSP is discussed – single-machine scheduling, where the ultimate goal is to generate an optimal schedule to carry out jobs at different times on the same machine. Several papers have proven that JSSP can be solved using classical algorithms, such as genetic algorithm , stochastic neural networks, Memetic algorithm (MA), and working time window non-delay scheduling algorithm (WT-NSA) [9, 10, 11, 12]. Amaro [13], on the other hand, studies the application of several variational quantum algorithms for a comprehensive job shop scheduling problem, including QAOA, variational quantum eigensolver (VQE), variational quantum imaginary time evolution algorithm (VarQITE) and the filtering variational quantum eigensolver (F-VQE). It was concluded in Amaro's study that F-VQE was the fastest, most consistent algorithm. Our paper aims to still utilize QAOA due to its ease of use and simple approach. Quantum annealing also proved to be applicable to a job shop scheduling problem by using more of a hybrid and heuristic method [14]. In this paper, as our initial work, we consider the single-machine scheduling problem to demonstrate the proposed approach of QAOA with active learning and Bayesian optimization. The rest of the paper is organized as follows – Section 2 provides a brief background to active learning, Bayesian optimization, and QAOA. Section 3 discusses the proposed methodology and Section 4 illustrates the proposed methodology for a schedule with two jobs, where each job is comprised of two operations each.

Linear regression is a model that can predict the value of a variable based on the value of the other variable. The variable that is being predicted is called the dependent variable whilst the variable that is being based on is called the independent variable. There are several methods to build a linear regression model, such as gradient descent and least square methods. Linear regression is widely used in many fields because its models are easy to compute and interpret, providing users a straightforward route to acquiring future predictions [15].

Production scheduling is a vital component within a manufacturing facility and doing it right can help the company save a lot of money and square footage. Operations Research is an important tool to utilize when it comes to production scheduling [16]. It helps allocate labor, plan material flow, schedule jobs, purchase materials and many other applications. It is an integration of manufacturing, planning, supply chain and management. In layman terms, it is an established process of controlling and planning work within any forms of production process at an optimized manner. With the rise of Industry 4.0, more interchangeable ideas need to be shared among the scheduling field, since both these aspects can greatly impact a corporation's efficiency and performance [17]. Genetic programming has been applied for production scheduling [18] and even the use of genetic algorithms for single machine preventative planning and production scheduling, as shown by Sortrakul [19] and Bierwirth [20]. Kumral [21] attempted a simulated annealing approach for mine production scheduling, where the main objectives are to minimize cost deviation from required tonnage, contents and fluctuations. Evidently, there are several established algorithms that can be implemented to solve production scheduling problems.

In this study, quantum linear regression and uncertainty propagation are introduced in attempt to efficiently schedule production and forecast demand. To make the 2x2 matrix experiment more realistic, production scheduling practices will be made reference to Great Plains Industries, a manufacturing company based in Wichita that supplies industrial flow meters all over the world. Flow meters are a type of tool that can measure the speed of liquid, fuel or solvent flowing through a pipe. The flow causes the turbine in the meter to rotate, sending pulses to record the flow rate. Hence, the volumetric flow rate is proportional to the rotational speed of the turbine blades. Although the data used in this paper is not identical to what GPI produces, realistic and well-approximated numbers are being used to reflect the functionality of this study. On the other hand, to prove the applicability of HHL algorithm, a 3x3 matrix production operations example will be referenced from a textbook.

Hybrid quantum linear regression equation algorithm has been proven to be useful and efficient based on several literature reviews. Lee [22] experimented the modified HHL algorithm on IBM Quantum Experience using four qubits. Liu [23], on the other hand, made improvements to the original HHL algorithm based on other results obtained by other scholars. However, the authors suggest further improvements to be made to the algorithm for better performance, such as obtaining higher-precision eigenvalue information, since traditional classical computers were used where space and time were limited. Zhang [24] also simulated improvements for the HHL algorithm using Qiskit. Multivariate polynomial systems can also be solved using the HHL algorithm but there are limitations in which the approach that had been used [25].

At the end of the paper, we will implement uncertainty propagation to predict excess inventory based on the unpredictability of its input parameters – which in this case is the cycle time. By using this method to make predictions, we can prepare for the uncertain fluctuation of inventory, and in a case of a spurge within the prediction model, companies can better prepare and make necessary adjustments to mitigate its large impact on overall production efficiency.

CHAPTER II

BACKGROUND

2.1 *Active Learning and Bayesian Optimization*

Active learning is an emerging learning algorithm which can perform better than traditional methods with substantially less amount of data. The algorithm itself proactively chooses data it wants to learn from to generate desire outputs. Melnikov [26] introduced the creation of new quantum experiments using active learning in automated laboratories and Gal [27] studied the improvement of image data using active learning approaches. In this paper, active learning is used within the Bayesian Optimization analysis to train a Gaussian Process surrogate model, which is later used for global optimization analysis. The effectiveness of Bayesian optimization has been proven through simulations and real experiments in an active learning framework [28]. Bayesian optimization provides an elegant framework for approaching problems that resemble the scenario to find the global minima in the smallest number of steps. Snoek [29] explains the good practices for Bayesian optimization of machine learning algorithms, especially when it comes the selection of acquisition functions.

Active learning is capable of solving optimization problems with limitations of limited sampling of inputs, unknown derivative and getting caught in local minima. Firstly, a surrogate function is formed based on sampled points that approximates the objective function. The surrogate function is updated accordingly to obtain more promising regions of the minima. Surrogate functions are usually represented by Gaussian Processes (GP), like a probability distribution. An inexpensive process that returns several functions with probabilities attached to them, instead of integers. Another representation of the surrogate function the Tree Parzen Estimators (TPE). It selects the value of y^* is that is a bit higher than the best observed so that the data points can be separated into 2 clusters – better than y^* and worse than y^*. In this paper, the analysis implements the Gaussian Process to acquire optimal values of hyperparameters due to its flexibility and tractability.

The surrogate function is updated with an acquisition function, also known as the selection function. This drives the proposition of new points to test, in an exploration and exploitation trade-off. Exploitation refers to exploiting known and promising spots. Exploration is to sample in locations with high uncertainty to ensure the whole area is being explored. The acquisition function needs to be balanced. This function is being used to update the surrogate model by maximizing the function and obtaining the corresponding hyperparameter and its objective function score. A high-level illustration of the entire workflow is shown in Figure 1.

There are three widely used acquisition functions that are utilized for active learning in Bayesian Optimization – (1) Probability of Improvement (PI), (2) Expected Improvement (EI), and (3) GP Lower Confidence Bound (GP-LCB). In this paper, we will test all three acquisition functions to determine the one that works best with the optimization problem. However, Wilson [30] also provides a good guide on maximizing acquisition functions. Probability of Improvement (PI) focuses on the maximizing the probability of improving over the best current value, x_t^+. Based on the GP, the strategy can be computed as:

$$-PI(x) = -P(f(x) \geq f(x_t^+) + k) \tag{1}$$

Expected Improvement (EI) maximizes the expected improvement over the current best value, which is in the form:

$$-EI(x) = -E[f(x) - f(x_t^+)] \tag{2}$$

GP Lower Confidence Bound (GP-LCB) constructs acquisition functions that minimize regret over the optimization process by exploiting lower confidence bounds (upper, when considering maximization):

$$LCB(x) = \mu_{GP}(x) + k\sigma_{GP}(x) \tag{3}$$

Active learning can be performed as such:

1. Initialize a Gaussian Process surrogate function prior distribution.
2. Choose several data points x such that the acquisition function a(x) operating on the current prior distribution is maximized.
3. Evaluate the data points x in the objective cost function c(x) and obtain the results, y.
4. Update the Gaussian Process prior distribution with the new data to produce a posterior.
5. Repeat steps 2–5 for several iterations.
6. Interpret the current Gaussian Process distribution to find the global minima.

2.2 *Quantum Approximate Optimization Algorithm (QAOA)*

QAOA is a variational algorithm that uses operators to perform a quantum analysis, that prepares a quantum state $|\psi(\beta_{opt}, \gamma_{opt})$ which encodes the personal best solution. The values of these parameters then enter the classical optimizer where a new set of values are generated to improve the overall optimal value. The optimal solution is identified after performing both the quantum and classical analysis until there are no further improvements observed. The outer loop of QAOA is an optimization problem for a classical computer to identify the optimal hyperparameters of rotation angles. In the initial place, Bayesian optimization approach is supposed to take place as the classical optimizer, where a surrogate function is constructed to guide the search of the optimal hyperparameters. The unique aspect of QAOA that sets it apart from other quantum algorithms is the Hamiltonian approach. Part of the procedure when applying QAOA is to derive the initial Hamiltonian, H_i, and final Hamiltonian, H_C, which are also known as mixer Hamiltonian and cost Hamiltonian. Tibaldi [31] also incorporated Bayesian optimization as a classical framework for QAOA and provides a more comprehensive perspective on how the algorithm works.

In the quantum circuit, Hadamard gates are applied in the beginning to ensure superposition of qbits, which assigns equal possibilities to both states for each qbit. Then, the cost and mixer operators, U_C (γ_p) and U_M (β_p) are applied based on the Hamiltonians that were formulated previously. In this theory, p is a positive integer where in which when the limit p $\rightarrow \infty$, QAOA is capable of finding the global optimum value of any classical combinatorial optimization problem. Depending on the problem complexity, a lower value of p can also result in the correct solution. The p¬-level QAOA algorithm has 2p parameters (γi and βi,i=1,2,3. . . p) which corresponds to the angles of the cost Hamiltonian and mixer Hamiltonian that are applied during each stage of the iteration. For $e^{i\beta X}$, an Rx gate with an angle of 2β is applied, for $e^{i\gamma Z}$, Rz gate with an angle of $2\gamma$is applied, while for $e^{i\gamma Z_1 Z_2}$, an Rz gate is applied in between 2 CNOT gates on the second qubit (target). Once the quantum analysis is carried out, the output is sent through the classical optimizer, where active learning generates improved values of γ and β for the next iteration. Figure 1 shows an illustration of the overall QAOA.

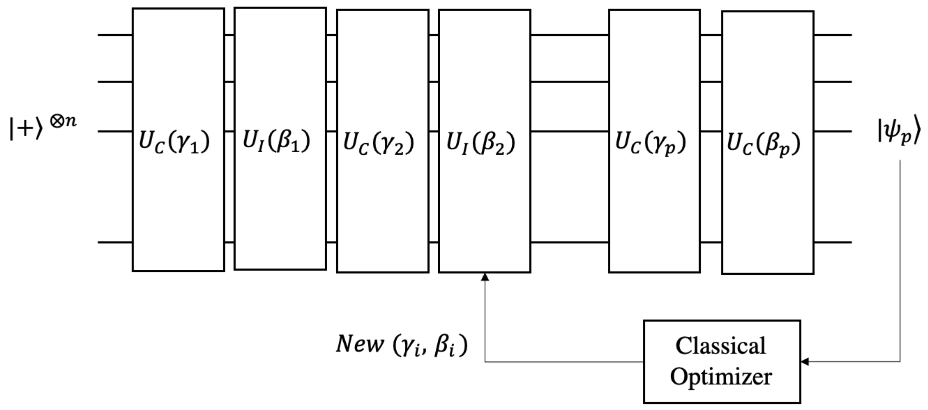

Figure 1: Illustrative quantum circuit of QAOA

2.3 *Quantum Linear Regression*

Linear regression is a form of curve fitting analysis where the main goal is to generate a function that best fits a set of data points. This approach is widely used in various fields to model and study the relationship between variables because regression analyses can predict output values for inputs that are unknown. With the emergence of quantum computing, there has been many efforts to perform linear regression using quantum algorithms – knowing quantum algorithms have higher computational efficiency and speed as compared to classical computers. In this paper, the Harrow-Hassidim-Llyod (HHL) quantum algorithm is discussed and implemented into a real-life example of production scheduling.

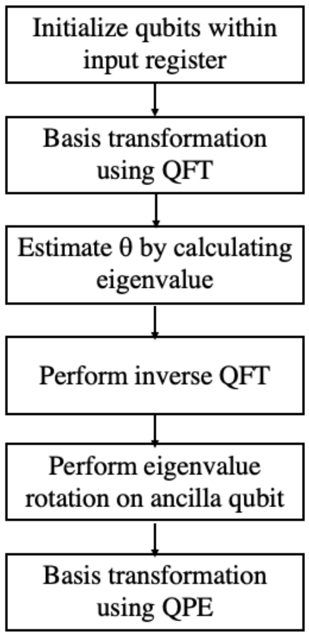
Initialize qubits within input register
Basis transformation using QFT
Estimate θ by calculating eigenvalue
Perform inverse QFT
Perform eigenvalue rotation on ancilla qubit
Basis transformation using QPE

Figure 2: Process flowchart of HHL algorithm

The HHL algorithm was developed by Aram Harrow, Avinatan Hassidim, and Seth Lloyd [32]. This algorithm is able to approximate a function of the solution vector in a relatively short period of time (O(logN)). This can be done using 3 main steps, which are quantum phase estimation (QPE), eigenvalue inversion rotation, and inverse QPE. There are also three quantum registers that encompasses multiple qubits respectively. The input register stores a binary representation of eigenvalues of a matrix, the clock (or solution register) comprises the solution vector while the ancilla register contains extra qubits that are needed for the analysis to be carried out.

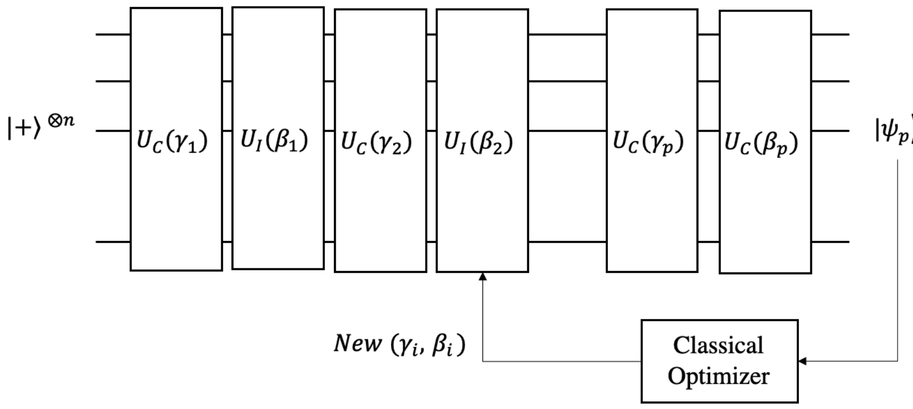

Figure 3: Framework of HHL Algorithm

The first step in the HHL algorithm is to initialize the qubits within the input register by loading the data $|b>$. After that, as mentioned, QPE is implemented to calculate the eigenvalue corresponding to its eigenvector in order to estimate θ. However, we cannot estimate θ in the computational basis. Thus, principles of quantum Fourier transform (QFT) are utilized to transform θ to a Fourier basis and then back to its original basis by using inverse QFT. The Fourier basis is basically an angular representation; hence it must be used when evaluating θ, whilst after implementing inverse QFT, the value is represented as an integer. The mathematical form of the unitary gate is:

$$U = e^{iAt} := \sum_{j=0}^{N-1} e^{i\lambda_j t} < u_j|u_j >= 1, \forall i \tag{4}$$

The next portion is to perform an eigenvalue rotation on the ancilla qubit. Each rotation angle is dependent on the eigenvalue, also referred as the qubits in clock register.

$$\sum_{j=0}^{N-1} b_j|\lambda_j >_{nl} |u_j >_{nb} (\sqrt{1 - \frac{C^2}{\lambda_j^2}}|0> + \frac{C}{\lambda_j}|1> \tag{5}$$

After that, inverse QPE is carried out so the outcome can be in the form of a computational basis.

$$\sum_{j=0}^{N-1} b_j|0>_{nl}|u_j>_{nb}(1-\frac{C^2}{\lambda_j^2})|0>+\frac{C}{\lambda_j}|1> \tag{6}$$

2.4 *Uncertainty Propagation*

When uncertainty is present, it is fairly difficult to predict the future, but such information can benefit many fields on a large scale. With uncertainty propagation though, we are able to build a model prediction based on the nature of its inputs and parameters. It has been proven that other methods of propagating uncertainty can be implemented, such as probabilistic logic sampling in Bayesian networks [33]. There are a total of four cases that can be analyzed, and each of them are unique on its own.

The first case is simple - when there are both deterministic inputs and model parameters, the output will also be deterministic. The second case is when there are uncertain inputs but deterministic model parameters. Since uncertainty is present in the model, the output will be uncertain and is represented as a distribution. There are three straightforward steps to case 2 – generate sample from the input distributions, obtain model prediction at each generated samples, and then model the distribution of output. There are two ways to represent the output sample, one is to use a histogram, but in this paper, we will use a non-parametric method known as Kernel Density Estimation (KDE). A kernel is basically a function that needs to satisfy two conditions: area under the curve must equal to 1, and it should be symmetric. It can be mathematically written as:

$$f(y)=\frac{1}{n}\sum_{i=1}^{n}K(y-y_i^*) \tag{7}$$

In case 3, the output is also estimated in a distribution using KDE because of the uncertain model parameters, but deterministic inputs. Similarly, the model prediction is calculated for each set of parameter values, then projected into a KDE distribution. Last but not least, when both the inputs and model parameters are uncertain, we will compute the output values for each set of values that has been generated from their corresponding distributions. The values of each model prediction are then represented as a KDE distribution. Therefore, uncertainty propagation is a straightforward and easy method to apply. It is evident that the concept can be easily interpreted, hence, it can be applied to many real-life problems where uncertainty is a major factor.

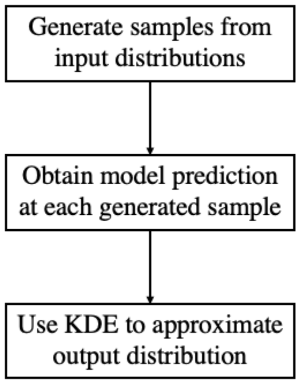
Generate samples from input distributions
Obtain model prediction at each generated sample
Use KDE to approximate output distribution

Figure 4: Process flowchart of uncertainty propagation

CHAPTER III

METHODOLOGY

3.1 *Quantum Approximate Optimization Algorithm*

The general form of a single-machine scheduling problem consists of k number of jobs, while the jobs branch out into n number of operations. Each operation has a processing time of p and it is assumed that all operations must be completed within a time t. The objective function is to minimize overall production cost and adhere to the corresponding constraints. The single machine scheduling problem is often solved under various constraints. The first constraint that needs to be taken account of is precedence constraint, where the operations associated with a given job should follow an order of sequence to be run on a single machine. The second constraint is to ensure that only one operation (related to any job) is carried out at a time, since there is only a single machine available in this case. The third constraint considered to ensure that an operation can done once and only once. Logically, an operation would not need to be processed a second time since it would have already been completed in the first run. The last constraint relates to operational deadlines. In a real-world, some operations will need to be done before or after a specific time. This is commonly observed when the required personnel are unavailable all the time. Therefore, some operations need to be completed when such personnel are available. This constraint will bring in the reality of manufacturing processes to the analysis to provide a more realistic schedule. Moreover, every operation is associated with a cost. The cost can be a constant across time or may change with time.

The first step in solving such combinatorial optimization problems is formulate a QUBO [5]. Since the scheduling problem is associated with constraints, the constraints needs to be incorporated into the objective function using the penalty method such that when the constraints are violated, a penalty term is added. Given the QUBO, we construct the QAOA circuit (both the cost Hamiltonian and the mixer Hamiltonian) for a given value of p (defined in Section 2.2). For a given value of p, we will have p parameters associated with cost and mixer Hamiltonians resulting in 2p number of parameters. Here, we build a Gaussian process model connecting the quantum gate parameters and the cost associated with the best solution (obtained from QAOA). To begin with, we select the initial values of gate parameters using a design of experiments (DOE) approach. Using the initial data, we construct a GP model, and subsequent input-output data are selected using an acquisition function. We continue adding new training data until no significant improvement is observed in the trained GP model. The trained GP model is then used to identify the optimal gate parameters. After identifying the optimal gate parameters, we run the quantum circuit to obtain the optimal solution and also associated cost. Fig 5 provides a step-by-step workflow of QAOA.

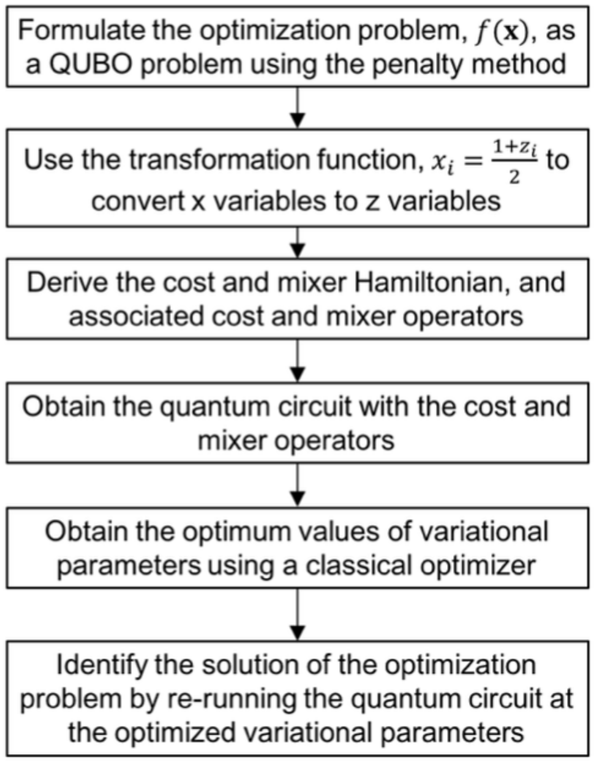
Formulate the optimization problem, $f(\mathbf{x})$, as a QUBO problem using the penalty method
Use the transformation function, $x_i = \frac{1+z_i}{2}$ to convert x variables to z variables
Derive the cost and mixer Hamiltonian, and associated cost and mixer operators
Obtain the quantum circuit with the cost and mixer operators
Obtain the optimum values of variational parameters using a classical optimizer
Identify the solution of the optimization problem by re-running the quantum circuit at the optimized variational parameters

Figure 5: Flowchart of QAOA Algorithm

3.2 *HHL Algorithm*

Qiskit itself has a built-in function for the HHL algorithm, and it has been used by many researchers in their work. Therefore, the analysis in this paper will also implement the built-in function and compare the results with the ones computed classically. Since there is a broad field of industrial engineering problems in the real world, the complexity is parallel to the size of the matrix. Hence, in this paper, we will attempt to implement HHL algorithm on a 2x2 and 3x3 production example.

To solve a system of linear equations using quantum computation, the problem needs to be first encoded into a comprehendible quantum language. The problem can be rescaled by normalizing a and b and map them to their corresponding quantum states $|\mathrm{b}\rangle$ and $|\mathrm{x}\rangle$. Hence the rescaled problem will be in a form of $\mathrm{A}|\mathrm{x}\rangle = |\mathrm{b}\rangle$.

There are two requirements that need to be adhered to in order to apply the HHL algorithm – the A matrix must be symmetric, and the b vector must be normalized. Therefore, the necessary transformation and adjustments must be made prior to running the algorithm to solve the equations. Most matrices that are computed from real-world problems is highly likely not symmetrical, so by simply multiplying the equations a common coefficient, we can achieve matrix symmetry without complicating the process. For the b vector, it needs to be normalized in a way that by adding all the squared values within the vector, it will equal to 1.

With the two arrays being transformed into a form that the algorithm can comprehend, the built-in function will compute an unnormalized solution vector. To obtain the final answer, a normalization analysis is done.

Note that in the HHL algorithm, the input register represents 2 states. Hence, for a 3x3 matrix, despite having only 3 values within the b vectors, we will still need to consider 2 qubits in the input register, which totals up to 4 states. To account for this, dummy values are considered to transform the matrix into a 4x4 dimension. The fourth variable is added while forcing it to equal to 0. Here is an example:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ 0 \end{bmatrix}$$

However, most matrices are not naturally symmetrical. In order to force a matrix to be symmetrical, the matrix is multiplied by its transpose. When doing so, the b vector must also be multiplied by the A transpose. A simple example below illustrates how this would work:

$$A^T A = \text{Symmetrical Matrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 4 & 1 \\ 0 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 2 & 4 & 2 \\ 3 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 14 & 12 & 13 \\ 12 & 18 & 11 \\ 13 & 11 & 13 \end{bmatrix}$$

As observed, the matrix is now symmetrical. The same is done here to transform the matrix into a 4x4 matrix by adding a dummy variable. For an asymmetrical 4x4 matrix, similarly, it needs to be multiplied by its transpose to force symmetry, in order for the HHL algorithm to comprehend and further make an analysis.

CHAPTER IV

CASE STUDY AND RESULTS

4.1 *Case Study 1: Job Shop Scheduling*

As a case study to demonstrate the proposed approach, we consider scheduling two jobs on a single machine. We denote the two jobs as j_1 and j_2. The first job j_1 is composed of two operations denoted as O_{11} and O_{12}, while the second job j_2 is also composed of two operations of O_{21} and O_{22}. We assume each operation (either associated with j_1 or j_2) takes 1 unit of time. Across the two jobs, we have four operations and since each operation takes one unit of time, it takes four units of time to complete both the jobs. We consider precedence constraints associated with j_1 and j_2 that O_{i2} should occur after O_{i1} where i=1,2. In addition to precedence constraints, we also consider an operational deadline associated with O_{12} that it needs to be completed before t=3. In this analysis, we are assuming that there are 2 jobs. Job 1 branches out into 2 operations – O_{11} and O_{12}, while job 2 branches out into 2 operations – O_{21}, O_{22}. At first, both these jobs are merged into 1 single job where there are all 4 operations, O_i where i=1,2,3,4,. Each operation has a processing time of pi, which we assume is 1 unit of time for all operations. Since there are 4 operations and 4 time slots, it can be justified that there are 16 combinations, which can also be referred to as random variables.

Operation can only be done once and only once and one operation can be carried out per time slot.

$$\sum_{t=0}^{3} x_{it} = 1, \forall i \tag{8}$$

$$\sum_{i=1}^{4} x_{it} = 1, \forall t \tag{9}$$

For the operational deadline constraint, it is assumed that O2 must be done by t =

3, meaning the operation has to start by t = 2.

$$\sum_{k=0}^{2} x_{2k} = 1 \tag{10}$$

And for the precedence constraint, it is enforced in the equation below that O2 must be done after O1 is completed, as well as O4 must be done after O3. The assumption here is that all operations will complete in 1 unit of time before proceeding to the next operation.

$$x_{10}x_{21} + x_{10}x_{22} + x_{10}x_{23} + x_{11}x_{22} + x_{11}x_{23} = 1 \tag{11}$$

$$x_{30}x_{41} + x_{30}x_{42} + x_{30}x_{43} + x_{31}x_{42} + x_{31}x_{43} = 1 \tag{12}$$

With these constraints, we can rule out a few variables that will definitely result in a value of 1. For instance, in the precedence constraint, operation 1 and 3 will never be assigned to last time slot – 3, while operation 2 and 4 will never be assigned to the first time slot. Hence, $x_{13}, x_{33}, x_{20}, x_{40} = 0$. In the operational deadline constraint, we know that operation 2 will never be assigned to time slot 3 because it has a deadline of completing by time slot 2. Hence, $x_{23} = 0$.

Since operation 2 will be assigned to time slot 2 or lower, it can be concluded that operation 1 cannot be assigned to time slot 2. Hence, $x_{12} = 0$. Lastly, it can induce that operation 4 will definitely be assigned to the last time slot because operation 1 and 2 cannot be done at time slot 4, while operation 3 must be done before operation 4. Hence, $x_{41}, x_{42} = 0$. With the analysis, we are left with only 7 possible combinations.

The overall problem is then narrowed down with the final two constraints: An operation must only be done once and only once.

$$x_{10} + x_{11} = 1 \tag{13}$$

$$x_{21} + x_{22} = 1 \tag{14}$$

$$x_{30} + x_{31} + x_{32} = 1 \tag{15}$$

Only one operation can be carried out per time slot.

$$x_{10} + x_{30} = 1 \tag{16}$$

$$x_{11} + x_{21} + x_{31} = 1 \tag{17}$$

$$x_{22} + x_{32} = 1 \tag{18}$$

The cost function that will be minimized can be derived as $\sum_{i=3}^{3} c_t x_{it}$ while taking into account of the analysis done above. The value of c_i depends on which time slot the operation is being carried out. The cost of carrying out operation 1 at time slot 1 is \$3, while at time slot 2 is \$5, and lastly at time slot 3 is \$7. On the other hand, for operation 2, the cost for each of the same time slots are \$4, \$6, \$8. And operation 3 costs \$7, \$5, \$4 respectively. The expansion of the cost function are follows:

$$c(x) = 3x_{10} + 5x_{11} + 7x_{12} + 4x_{20} + 6x_{21} + 7x_{22} + 7x_{30} + 5x_{31} + 4x_{32} \tag{19}$$

Next, the constraints mentioned above need to be shifted to the objective function. In order to do that, the constraint equation needs to be forced to be equal to 1 and squared to ensure non-negativity. Moreover, a penalty value, η must be applied. The overall QUBO minimization objective function is:

$$\begin{aligned}&3x_{10} + 5x_{11} + 7x_{12} + 4x_{20} + 6x_{21} + 7x_{22} + 7x_{30} + 5x_{31} + 4x_{32} + \eta(x_{10} + x_{11} - 1)^2 + \\ &\eta(x_{21} + x_{22} - 1)^2 + \eta(x_{30} + x_{31} + x_{32} - 1)^2 + \eta(x_{10} + x_{30} - 1)^2 + \eta(x_{22} + x_{32} - 1)^2 + \\ &\eta(x_{11} + x_{21} + x_{31} - 1)^2\end{aligned} \tag{20}$$

The next step is to perform a linear transformation to convert the x variables into z variables. This is a very vital step because the binary strings must take values of 1 or -1 in order to utilize the quantum Ising Hamiltonian approach. This transformation can be carried out using $x_i = \frac{1+z_i}{2}$. Each combination is represented as a qubit in the circuit. Next, from the converted function with z variables, the cost and mixer operators are formed. The quantum circuit is then computed with all corresponding gates to the coefficient in the expanded objective function.

Once the quantum portion of the analysis is completed, the initial cost value is passed through the classical optimizer where active learning is implemented to obtain improved values of and . By using EI and PI as the acquisition function in the active learning portion of the algorithm, the results fluctuated every trial. However, LCB produces accurate results more accurately. With the new improved values that were generated by using active learning, the overall optimal cost is computed - 13. The algorithm optimized the problem and determined that the string that produced the optimal cost is $|1010001>$. This means that $x_{10}, x_{21}, x_{32} = 1$, where O_1 is assigned to the first time slot, O_2 is assigned to time slot 1, O_3 is assigned to time slot 2, and based on the first analysis, O_4 is already assigned to the last time slot.

It was determined that LCB produces consistent and accurate results after running the three algorithms on 10 iterations and measuring its performance rate. Table shows the results obtained and it can be observed that LCB produced the correct solution 80% of the time, whilst PI and EI have a performance rate of only 15% and 55% respectively.

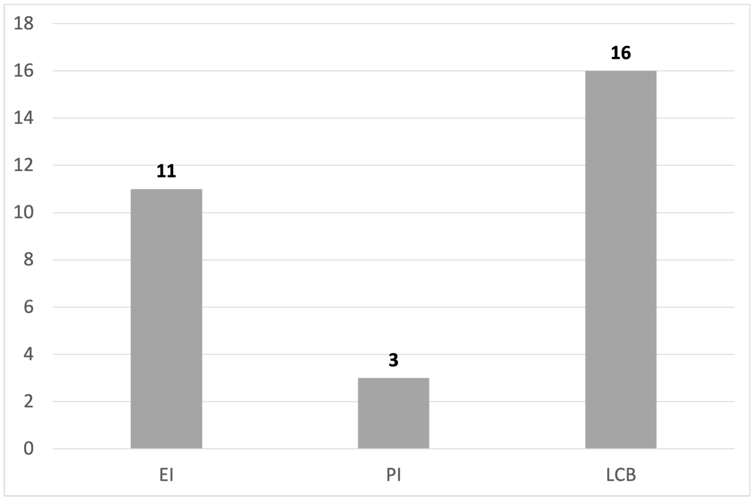

Figure 6: Frequency of Computing Minimum Value for each Acquisition Function

Table 1: Results of Each Acquisition Function for 20 Trials

PI	EI	GP-LCB
109	13	109
109	13	108
109	19	13
15	13	13
109	109	13
109	109	13
109	109	13
109	13	13
109	13	13
109	13	13
13	13	13
109	109	13
109	109	13
109	109	13
13	13	13
109	19	109
109	13	13
109	13	13
13	13	15
109	109	13

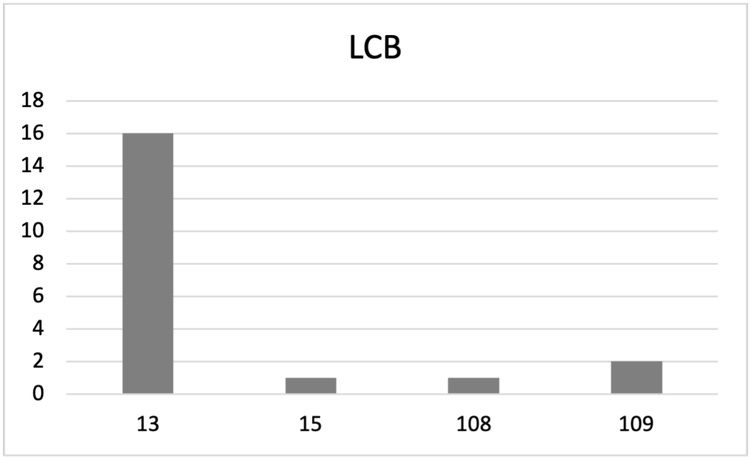

Figure 7: Performance of LCB

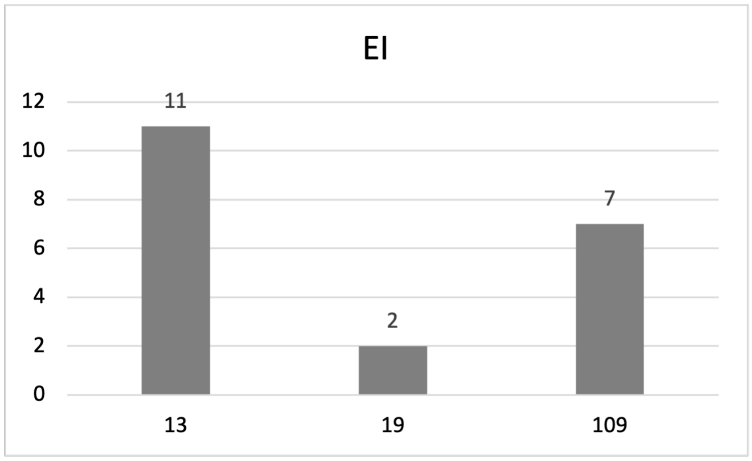

Figure 8: Performance of EI

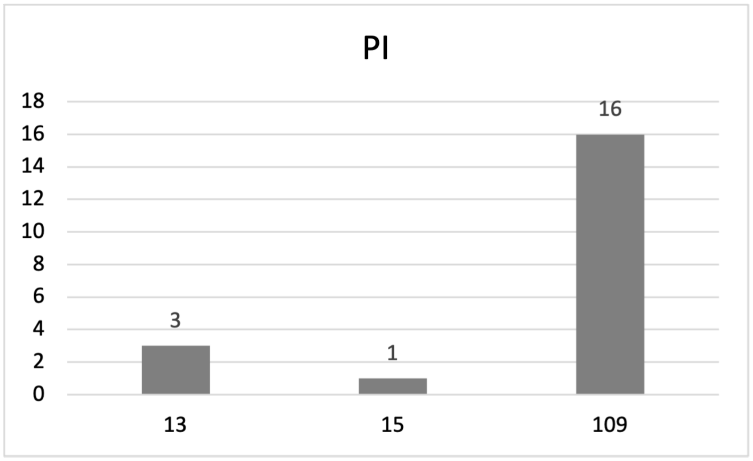

Figure 9: Performance of PI

4.2 *Case Study 1: Production Scheduling*

4.2.1 2x2 Example

Operators typically build about 9 meters an hour, which can be translated into a cycle time of 0.15 meters per minute. The meters will then be transported to another work cell to be assembled on another part. There is approximately 6 meters that are being used per hour. Similarly, it can be computed as a cycle time of 0.1 meters per minute. Additionally, there is an inventory constraint on meters, meaning there is a limitation on the number of meters that can be stocked in the warehouse. In this case, there can only be an excess of 30 meters from production to be stored as inventory. We also determined the work hours per day to be 9 hours, which equates to 540 minutes.

x_1 is defined as the time spent in minutes building meters, while x_2 is the time spent in minutes being assembled to its main assembly. That being said, by multiplying the x variables with cycle time, we are able to obtain the amount of time that needs to be planned for each building and using the meters. Adding on the inventory constraint, equation 1 is formulated. To ensure that the total time spent sums up to the total work hours a day, equation 2 is created as well.

$$\begin{cases} 0.15x_1 - 0.1x_2 = 30 \\ x_1 + x_2 = 540 \end{cases} \tag{21}$$

The equations need to be transformed so that the HHL algorithm can be comprehended. The A matrix must be symmetrical whilst the b vector must be normalized. Therefore, with the necessary transformations, we have the following:

$$A = \begin{bmatrix} -1.5 & 1 \\ 1 & 1 \end{bmatrix} \quad , \quad b = \begin{bmatrix} -300 \\ 540 \end{bmatrix}$$

By solving this classically, $x_1 = 336$ minutes and $x_2 = 204$ minutes. This means that there should be at least 336 minutes, or 5.6 hours allocated to building meters in order to meet the demand and adhere to the inventory restrictions.

To solve this problem using a quantum computer, a built-in Python function of the HHL algorithm is implemented and ran on a simulator. The solution computed using this function will come out unnormalized. Hence, another normalization step needs to be done to acquire the final answer. With quantum computation, the final answer of the x variables came out to be $x_1 = 289.9$ minutes and $x_2 = 216.8$ minutes. There is a slight discrepancy between the solution computed using the quantum computer and classically by hand. However, it has been prevalent that quantum computing has not reached 100 percent accuracy yet, but the time it takes for the solution to be computed is much shorter. It may not be observed in this example, but as complexity increases, the computational difference will surface.

This sort of information can assist Value Stream Leaders and Production Schedulers in not only production planning, but also capacity planning, line balancing, inventory management, warehouse operations and even maintaining a Kanban system for material flow.

4.2.2 3x3 Example

Most of the meters that are manufactured would need to be tested for functionality and conformal configuration. Hence, the 2x2 example can be further expanded into a 3x3 example with a total of three variables. Similarly, x_1 is defined as the time spent in minutes building meters, while x_2 is the time spent in minutes being assembled to its main assembly. The third variable, x_3, represents the time it takes to test a single unit of meter. Based on real-life time study data, the average cycle time for a full cycle of meter functional testing is approximately 3 minutes, which can be converted into 0.05 meters per minute. With the comprehensive consideration of the cycle time for building meters, assembling to main assembly as well as testing, a demand of 50 meters is still required to meet demand. Hence, the first equation is computed.

The constraint will be carried over from the 2x2 example where there is a limit to inventory space, which is represented as the second equation. Lastly, the total time spent on building, assembling and testing should not exceed one work shift, which is equivalent to 540 minutes. The testing time is divided in half because 50% of time, operators would be performing a value-added task while the tester is running. This was observed while conducting a time study analyzing the value-added and non-value added touch time of materials during the entire process. This constraint is represented as the third equation. The combined equations are as follows:

$$\begin{cases} 0.15x_1 - 0.1x_2 + 0.05x_3 = 30 \\ 0.15x_1 - 0.1x_2 = 30 \\ x_1 + x_2 + \frac{x_3}{2} = 540 \end{cases} \tag{22}$$

This set of equations are then converted into a comprehendible format for the HHL algorithm to understand. The matrices are as follows:

$$A = \begin{bmatrix} -1.5 & 1 & 0.5 \\ 1.5 & -1 & 0 \\ 10 & 10 & 5 \end{bmatrix} \quad , \quad b = \begin{bmatrix} 500 \\ 30 \\ 5400 \end{bmatrix}$$

The first issue with the above the A matrix is that it is not symmetrical. Hence, by multiplying the A matrix with A transpose, a symmetrical matrix can be formed. To normalize the b vector, it is multiplied by the A transpose matrix as well, computing the following matrices.

$$A = \begin{bmatrix} 7.66 & 1.25 & 35.5 \\ 1.25 & 3.25 & 5 \\ 35.5 & 5 & 225 \end{bmatrix} \quad , \quad b = \begin{bmatrix} 54795 \\ 54470 \\ 28050 \end{bmatrix}$$

HHL algorithm cannot solve a 3x3 example directly. Hence, a fourth dummy variable will need to be added, converting this set of equations into a 4x4 example, This can be done by indicating the fourth dummy variable is equal to 0 as the fourth equation.

$$A = \begin{bmatrix} 7.66 & 1.25 & 35.5 & 0 \\ 1.25 & 3.25 & 5 & 0 \\ 35.5 & 5 & 225 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad , \quad b = \begin{bmatrix} 54795 \\ 54470 \\ 28050 \\ 0 \end{bmatrix}$$

4.2.3 4x4 Example

In a case of optimizing production scheduling within a factory network of workstations, there are many factors that can play into a real-life situation. Each workstation can represent different tasks and as the product flows to the next step, it can either proceed to the following workstation or return to the previous workstation for rework. Each branch has its respective probabilities as well as an arrival rate for each corresponding workstation. An example of a production flow is shown in figure 10.

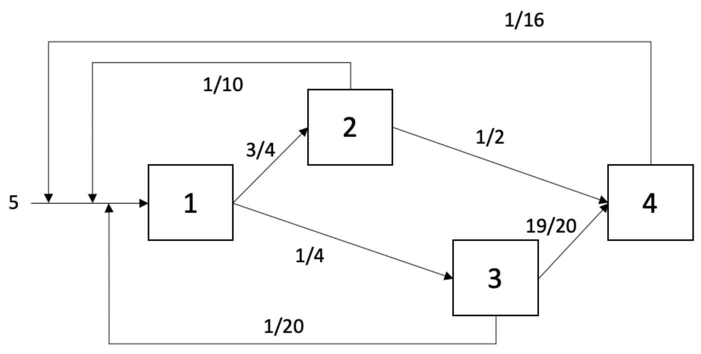

Figure 10: Factory Network of Workstations with Branching Probabilities

There are 4 different arrival rates, depicted as λ_1 for workstation 1, λ_2 for workstation 2, λ_3 for workstation 3 and λ_4 for workstation 4.

$$\begin{cases} \lambda_1 - 0.1\lambda_2 - 0.05\lambda_3 - 0.063\lambda_4 = 5 \\ -0.75\lambda_1 + \lambda_2 = 0 \\ 0.25\lambda_1 + 0.9\lambda_2 - \lambda_3 = 0 \\ 0.5\lambda_2 + 0.95\lambda_3 - \lambda_4 = 0 \end{cases} \tag{23}$$

The flowchart can then be converted into A matrix and b matrix. Similar to the 3x3 example, the A matrix needs to be symmetrical by multiplying its transpose by both the A matrix and b vector respectively. Since this is already a 4x4 matrix, there is no need to add a dummy variable because the HHL algorithm can comprehend the problem.

$$A = \begin{bmatrix} 1.02 & -0.85 & 0.21 & -0.04 \\ -0.85 & 1.56 & 0.71 & 0.50 \\ 0.21 & 0.71 & 1.87 & -0.50 \\ -0.04 & 0.50 & -0.50 & 2.15 \end{bmatrix} \quad , \quad b = \begin{bmatrix} 5 \\ -0.5 \\ -0.25 \\ -0.31 \end{bmatrix}$$

The 2x2, 3x3 and 4x4 examples were all ran on classical and quantum computers to make contrasts and comparisons on the solution. Table **??** shows an overview comparison between the quantum and classical solution. The outcomes are relatively consistent and accurate, which produced an accuracy rate of 97.8%.

Table 2: Solution Comparison for each Example

	2x2 Example	3x3 Example	4x4 Example
Quantum Solution	0.5982357	0.3113363	8.3615551
Classical Solution	0.6362197	0.3134811	8.3613517

4.3 Uncertainty Propagation

Using the first equation from the 2x2 example, we can conduct uncertainty propagation to predict potential excess inventory. In this case, we assume the value of y to be the model prediction, so we will not be using the value of 30 from the linear solver part. The first step in doing this is to determine which case this problem falls under. Knowing that the input parameters, which are the cycle times, are uncertain at times and varies across different builds of flow meters, we will use case 2 of uncertainty propagation.

Upon conducting time studies, the cycle times for different operators vary, but not by much. Hence, it is assumed that the x variables follow a uniform distribution with respective upper and lower bounds.

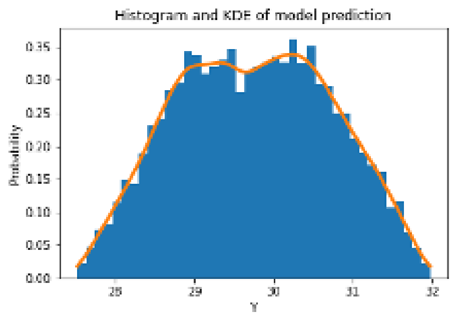

Figure 11: Results of model prediction KDE

As observed in the histogram and KDE of the model prediction, the mean is 29.75 while the standard deviation is 0.98. This means that there is a flexible inventory space of 28 to 31 meters for this example. With this knowledge, companies can better prepare for extra inventory or make capacity planning adjustments based on the model prediction. This does not only save time but also can make a positive financial impact to the company.

CHAPTER V

CONCLUSION

The paper demonstrated solving a single machine scheduling using QAOA and active learning with seven decision variables. Despite the low problem complexity, it has been justified that the algorithm is capable of solving a machine scheduling problem. The methods that were discussed in this paper are applicable to combinatorial optimization problems in various other fields. From our analysis, we observed that of the acquisition functions used for active learning, LCB was comparatively more reliable and produced more accurate results. The results obtained from the proposed approach match well with those of the classical analysis. Following our initial work in this paper, as future work, we seek to study the scalability of the proposed approach in terms of the number of decision variables, and also the complexity of QAOA circuit. We will also investigate the application of the proposed approach to more complex job-shop scheduling problems, and also implement this approach on real quantum hardware. This paper is also the preliminary step to prove the application of quantum algorithms on industrial engineering related problems. The example used here is fairly simplistic but easy to prove its functionality and reliability. With this study, we also cover the uncertainty of certain parameters within production planning and how we can use an active learning approach to predict the future. This can help provide more time and opportunities to develop a solution before making an immediate impact. The drawback of this case study is the lack of external validity and applications to a broader and more complicated production scheduling problem. However, with the mentioned concept and steps, most scheduling problems can be solved using quantum linear regression and can be further modeled using uncertainty propagation. More research can be done to generalize a quantum algorithm specific to production scheduling in the future, where the application can be straightforward and one dimensional so that all the end user needs to do is to just key in the required inputs.

REFERENCES

REFERENCES

[1] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[2] J. Choi and J. Kim, "A tutorial on quantum approximate optimization algorithm (qaoa): Fundamentals and applications," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 138–142, IEEE, 2019.

[3] Q. Wang and T. Abdullah, "An introduction to quantum optimization approximation algorithm," 2018.

[4] P. Vikstål, M. Grönkvist, M. Svensson, M. Andersson, G. Johansson, and G. Ferrini, "Applying the quantum approximate optimization algorithm to the tail-assignment problem," *Physical Review Applied*, vol. 14, no. 3, p. 034009, 2020.

[5] R. Harikrishnakumar, S. E. Borujeni, A. Dand, and S. Nannapaneni, "A quantum bayesian approach for bike sharing demand prediction," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 2401–2409, IEEE, 2020.

[6] R. Harikrishnakumar and S. Nannapaneni, "Smart rebalancing for bike sharing systems using quantum approximate optimization algorithm," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 2257–2263, IEEE, 2021.

[7] J. Kirschner, I. Bogunovic, S. Jegelka, and A. Krause, "Distributionally robust bayesian optimization," in *International Conference on Artificial Intelligence and Statistics*, pp. 2174–2184, PMLR, 2020.

[8] Y. Wang, "A quantum approximate bayesian optimization algorithm for continuous problems," in *IIE Annual Conference. Proceedings*, pp. 235–240, Institute of Industrial and Systems Engineers (IISE), 2021.

[9] 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*, vol. 2, pp. 1577–1582 vol.2, 1994.

[10] F. Y.-P. Simon and Takefuji, "Stochastic neural networks for solving job-shop scheduling. i. problem representation," in *IEEE 1988 International Conference on Neural Networks*, pp. 275–282 vol.2, 1988.

[11] S. Hasan, R. Sarker, D. Essam, and D. Cornforth, "Memetic algorithms for solving job-shop scheduling problems," *Memetic Computing*, vol. 1, no. 1, pp. 69–83, 2009.

[12] K. Ploydanai and A. Mungwattana, "Algorithm for solving job shop scheduling problem based on machine availability constraint," *International Journal on Computer Science and Engineering*, vol. 2, no. 5, p. 2010, 1919.

[13] D. Amaro, M. Rosenkranz, N. Fitzpatrick, K. Hirano, and M. Fiorentini, "A case study of variational quantum algorithms for a job shop scheduling problem," *EPJ Quantum Technology*, vol. 9, no. 1, p. 5, 2022.

[14] K. Kurowski, J. Wglarz, M. Subocz, R. Różycki, and G. Waligóra, "Hybrid quantum annealing heuristic method for solving job shop scheduling problem," in *International Conference on Computational Science*, pp. 502–515, Springer, 2020.

[15] X. Su, X. Yan, and C.-L. Tsai, "Linear regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 3, pp. 275–294, 2012.

[16] G. Buxey, "Production scheduling: Practice and theory," *European Journal of Operational Research*, vol. 39, no. 1, pp. 17–31, 1989.

[17] M. Parente, G. Figueira, P. Amorim, and A. Marques, "Production scheduling in the context of industry 4.0: review and trends," *International Journal of Production Research*, vol. 58, no. 17, pp. 5401–5431, 2020.

[18] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: a survey with a unified framework," *Complex & Intelligent Systems*, vol. 3, no. 1, pp. 41–66, 2017.

[19] N. Sortrakul, H. L. Nachtmann, and C. R. Cassady, "Genetic algorithms for integrated preventive maintenance planning and production scheduling for a single machine," *Computers in Industry*, vol. 56, no. 2, pp. 161–168, 2005.

[20] C. Bierwirth and D. C. Mattfeld, "Production scheduling and rescheduling with genetic algorithms," *Evolutionary computation*, vol. 7, no. 1, pp. 1–17, 1999.

[21] M. Kumral and P. Dowd, "A simulated annealing approach to mine production scheduling," *Journal of the Operational Research Society*, vol. 56, no. 8, pp. 922–930, 2005.

[22] Y. Lee, J. Joo, and S. Lee, "Hybrid quantum linear equation algorithm and its experimental test on ibm quantum experience," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[23] X. Liu, H. Xie, Z. Liu, and C. Zhao, "Survey on the improvement and application of hhl algorithm," in *Journal of Physics: Conference Series*, vol. 2333, p. 012023, IOP Publishing, 2022.

[24] M. Zhang, L. Dong, Y. Zeng, and N. Cao, "Improved circuit implementation of the hhl algorithm and its simulations on qiskit," *Scientific Reports*, vol. 12, no. 1, pp. 1–12, 2022.

[25] J. Ding, V. Gheorghiu, A. Gilyén, S. Hallgren, and J. Li, "Limitations of the macaulay matrix approach for using the hhl algorithm to solve multivariate polynomial systems," *arXiv preprint arXiv:2111.00405*, 2021.

[26] A. A. Melnikov, H. Poulsen Nautrup, M. Krenn, V. Dunjko, M. Tiersch, A. Zeilinger, and H. J. Briegel, "Active learning machine learns to create new quantum experiments," *Proceedings of the National Academy of Sciences*, vol. 115, no. 6, pp. 1221–1226, 2018.

[27] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *International Conference on Machine Learning*, pp. 1183–1192, PMLR, 2017.

[28] S. Bansal, R. Calandra, T. Xiao, S. Levine, and C. J. Tomlin, "Goal-driven dynamics learning via bayesian optimization," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 5168–5173, 2017.

[29] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[30] J. Wilson, F. Hutter, and M. Deisenroth, "Maximizing acquisition functions for bayesian optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[31] S. Tibaldi, D. Vodola, E. Tignone, and E. Ercolessi, "Bayesian optimization for qaoa," *arXiv preprint arXiv:2209.03824*, 2022.

[32] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical review letters*, vol. 103, no. 15, p. 150502, 2009.

[33] M. Henrion, "Propagating uncertainty in bayesian networks by probabilistic logic sampling," in *Machine intelligence and pattern recognition*, vol. 5, pp. 149–163, Elsevier, 1988.