Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Quantum Approximate Optimisation Algorithm for Classic Timetabling Problems

Author:
Can Orman

Supervisor:
Roberto Bondesan

Submitted in partial fulfillment of the requirements for the MSc degree in MSc Artificial Intelligence Applications and Innovation of Imperial College London

September 2024

Abstract

The university timetabling problem (UTP) is a critical issue in higher education institutions for ensuring optimal resource allocation. Finding an optimal timetable is challenging as the UTP is classified as NP-Hard, requiring exponential computation power to solve large problem instances. Such problems are classically solved using dedicated algorithms, local search strategies, or metaheuristics; these approaches often yield acceptable results, but they heavily rely on classical computational power and may give suboptimal results from being trapped within local minimas. An emmerging field within quantum computing, namely adiabatic computing offers a promising speedup for solving combinatorial problems by leveraging the adiabatic theorem, stating that a quantum system remains in its instantaneous eigenstate if changes to the system's Hamiltonian occur slowly enough. Through this theorem, The Quantum Approximate Optimization Algorithm (QAOA), is able to solve combinatorial optimisation problems by a sequence of parameterised quantum operations to gradually transform an initial state into an approximate solution. This paper presents a formulation of a simplified exam scheduling problem and demonstrates how it can be transformed into a graph colouring problem, which is further reformulated as a quadratic unconstrained binary optimisation (QUBO) problem suitable to be solved by the QAOA.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

The university timetabling problem (UTP) is a well-known combinatorial optimisation challenge; there are many variants of the UTP but common problem instances typically involve scheduling a series of courses, students, rooms, and timeslots while adhering to various constraints [1]. Constraints in the context of UTP are requirements or rules that timetables must satisfy for the timetable to be feasible, constraints may vary but typically involve ensuring no class conflicts, assigning rooms with sufficient capacity for classes, and others. Designing a timetable that optimally satisfies these constraints is a type of problem known as combinatorial optimisation, a problem known to be NP-hard. Each combinatorial problem is defined by a finite set of possible solutions, where the goal is to find the optimal solution within the defined set. In principle, the optimal solution to combinatorial optimisation problems can be found through brute force or enumeration strategies, but in practice, this is frequently impossible. This is because most combinatorial problems fall into the category of NP-Hard or NP-Complete; as the answer to such problems involves a selection of components within a defined set, each component added to the set increases the search space of the problem by an exponential degree (e.g. $O(2^n)$ for binary problems or $O(n!)$ for permutations problems such as the UTP) [2]. This rapid growth makes it computationally infeasible to examine each possible solution for practical-sized problems.

A common workaround to the computational intractability of combinatorial problems is to utilise stochastic optimisation techniques. Metaheuristics such as evolutionary algorithms and simulated annealing, for example, have proven to yield satisfactory results and in some cases optimal solutions [3]. Despite their success in the field of combinatorial optimisation, there are two common pitfalls with stochastic search methods, firstly metaheuristics often converge prematurely to suboptimal solutions, particularly in large optimisation problems where solution landscapes are complex with many local optima. The second issue comes from the fact that stochastic methods may be overly reliant on classical computing power, which is bounded by the physical limits of microcontrollers and power issues [4].

An emerging field within quantum computing, adiabatic computing, shows great

promise in overcoming the shortcomings of classical solutions in solving combinatorial optimisation problems. Adiabatic quantum computing is based on the principles of quantum mechanics, where solutions to combinatorial optimisation problems are found through evolving a quantum system from an initial ground state to the ground state of the problem Hamiltonian that encodes the solution. This process known as adiabatic evolution leverages the adiabatic theorem, which states that the quantum system will remain in the instantaneous ground state provided its Hamiltonian changes sufficiently slowly with time. At the end of the adiabatic evolution, if the ground state is maintained throughout the process of evolution then the measurement of the final state will yield the optimal solution of the combinatorial optimisation problem.

This paper will focus on a practical application of the Quantum Approximation Optimisation Algorithm (QAOA), an algorithm inspired by adiabatic computation, to solve a variant of the UTP. We first present a literature review on QAOA, outlining its core structure and mechanics as well as discussing recent works evaluating the QAOA and the suitability of QAOA for UTP. We then formulate the Exam Timetabling Problem (ETP), as a variant of the UTP and demonstrate how this can be encoded into a quadratic unconstrained binary optimisation problem suitable for QAOA. The methodology of implementing the QAOA is subsequently proposed as well as the results in the following section. Finally, the evaluation of the QAOA and further discussion on the scalability of the proposed methodology and current implementations of NISQ devices is discussed in the last section.

1.2 Literature Review

The QAOA, introduced by Farhi et al [5] represents a significant advancement in the field of quantum computing, introducing an algorithm inspired by the process of adiabatic evolution and quantum annealing. The algorithm is an algorithmic framework derived as an approximation to the Quantum Adiabatic Algorithm (QAA) introduced earlier in 2000 [6]. The goal of the QAOA is to produce approximate solutions to optimisation problems specified by an objective function that operates on n qubits with m constraints; a key integer parameter p additionally determines the quality of the approximation by defining the depth of iterations performed in QAOA.

The QAOA is structured around two key unitary operations, the objective function operator $U(C, \gamma)$ and the mixing operator $U(B, \beta)$, also known as the final and initial Hamiltonian respectively. The objective function operator $U(C, \gamma)$ encodes the optimisation problem by acting on a quantum state such that the eigenvalues correspond with the objective function values of potential solutions. This unitary operation rotates the quantum state based on the specific cost function and an angle parameter γ, where the parameter γ determines how strongly the problem constraints influence the state evolution. The mixing operator $U(B, \beta) = e^{-i\beta B}$ is composed of a sum of single-qubit Pauli-X matrices $B = \sum_{j=1}^{n} \sigma_x^{(j)}$. This operator initiates the QAOA by creating a superposition state and facilitates exploration by enabling transition between states. The mixing operation applies rotations determined by a

parameter β, which controls how much the quantum state shifts or "mixes" between the objective function operator $U(C, \gamma)$. Through these two unitary operators, the QAOA evolves the quantum state by alternating between the application of $U(C, \gamma)$ and $U(B, \beta)$ with p layers. This sequence:

$$|\gamma, \beta\rangle = U(B, \beta_p)U(C, \gamma_p) \cdots U(B, \beta_1)U(C, \gamma_1) |s\rangle ,$$

iteratively transforms the initial uniform superposition state into one that encodes high-quality solutions. We can observe that parameter p defines the number of alternating layers between the two unitary operations, where each layer applies a controlled evolution, guided by tunable angles γ and β, to progressively enhance the overlap of the quantum state with the optimal solution. The algorithm scales linearly with p, and as p approaches infinity, the optimal solution is guaranteed to be found.

There are several distinguishing differences between the QAOA and its inspired counterpart QAA. The most important distinction is that the QAOA adiabatic evolution is defined by the discrete parameter p whereas the QAA evolves continuously and is parameterised by time t. This discrete step nature makes QAOA suitable for gate-based quantum computers including current-era Noisy Intermediate-Scale Quantum (NISQ) devices, and provides flexibility in applications as researchers have the option to limit discrete steps p so that the depth of the circuit is kept within the capabilities of the quantum hardware accessible. The use of classical parameters γ and β additionally makes the QAOA a hybrid algorithm, combining quantum computation for state preparation and adiabatic evolution with classical optimisation for parameter tuning. The classical parameters of QAOA therefore require classical parameter optimisation strategies to run effectively, this can involve utilising metaheuristics like evolutionary algorithms, using gradient-based models, or other advanced techniques to tune γ and β efficiently.

The inception of QAOA sparked research into analysing its performance on various combinatorial problems, beginning with the Max-Cut problem in the original paper which demonstrates that the QAOA had an approximation of at least 0.6924 for the Max-Cut problem when p=1. This performance was not as competitive as the current best classical algorithm for the Max-Cut problem, namely the Goemans-Williamson algorithm [7] which provided an approximation ratio of approximately 0.878 for general Max-Cut problems using semidefinite programming. However as we consider the QAOA at higher depths of p, the results between QAOA and classical methods become increasingly interesting and indecisive. Basso's 2022 paper for example [8], provides an in-depth analysis of the QAOA applied to the Max-Cut problem on large-girth D-regular graphs and its comparison to classical algorithms. It develops iterative methods to evaluate the performance of QAOA at various depths, presenting an approach that can outperform classical counterparts at high depths, such as p=11 or beyond. Hastings' 2019 paper, however, challenges the assumption that the Quantum Approximate Optimisation Algorithm (QAOA) consistently outperforms classical algorithms [9]. By analysing bounded-depth optimisation approaches, he demonstrates that for problems like Max-3-Lin-2 and Max-Cut, local classical algorithms can achieve similar or better results than the one-step QAOA, particularly on triangle-free graphs.

Despite empirical findings and comparisons, Farhi claims in a further study [10] that there are findings and complexity theory arguments that indicate that the QAOA has the potential to hold quantum supremacy. Even the most shallow QAOA circuit (p=1) outputs a distribution that cannot be efficiently calculated by classical computers without collapsing the polynomial hierarchy. While the debate of whether QAOA will hold quantum supremacy is inconclusive, the potential in QAOA for near-time applications is clear. Lucas [11] outlines how various NP-Complete and NP-Hard problems can be mapped onto the Ising Model and equivalently the Quadratic Unconstrained Binary Optimisation (QUBO) problem, a crucial step in defining the cost function operator for QAOA. The paper includes formulations for all 21 NP-complete problems identified by Karp, including the Travelling Salesman Problem, Graph Colouring and Job Sequencing.

While there is extensive literature on QAOA performance on Max-Cut, the research on the practical use cases for QAOA is limited due to quantum hardware limitations on current NISQ-era devices. However as quantum hardware continues to expand, the capabilities of QAOA could become sufficiently powerful to solve meaningfully large combinatorial optimisation problems. Additionally, while the Max-Cut problem serves as a useful benchmark for evaluating QAOA against classical algorithms, it is an unconstrained problem. To determine whether QAOA holds advantages over classical optimisation algorithms, it is essential to apply it to more commonly encountered constrained optimisation problems in real-world applications. In this paper, we look at the UTP as a starting point to examine the QAOA in a more constrained problem formulation. The UTP is a natural fit for evaluating QUBO as it can be classically modelled as a graph colouring problem and we may easily adjust the complexity of the UTP by reducing or adding custom constraints. We initially explore the QAOA in solving our simplified formulated version of the exam timetabling problem (ETP), a variant of the UTP discussed in the following chapter.

Chapter 2

Problem Formulation

2.1 Simplified Exam Scheduling Problem

The exam timetabling problem involves assigning exams to predefined timeslots while ensuring that no student is scheduled for overlapping exams. The problem can be represented by defining a set of exams $E = \{e_1, e_2, \ldots, e_n\}$ and a set of timeslots $T = \{t_1, t_2, \ldots, t_k\}$. Each exam e_i must be assigned to a timeslot t_j such that no conflicts occur.

A conflict exists if two exams, e_i and e_j, share at least one student. This constraint ensures that if e_i and e_j have common students, they cannot be assigned to the same timeslot. The objective is to allocate all exams into the minimum number of timeslots while satisfying these constraints. Formally, we aim to find a function $f : E \to T$ such that for any two conflicting exams e_i and e_j, $f(e_i) \neq f(e_j)$. The goal is to minimize $|T|$, the total number of timeslots used, while adhering to the conflict constraints.

It is possible for the optimisation to include additional criteria, such as balancing the distribution of exam load across days or minimising the number of consecutive exams in back-to-back timeslots. Such preferences may be represented by adding additional constraints in the problem formulation that softly penalises timetables that violate these preferences. However, with the limitation of QAOA hardware in mind, we opt for a simpler formulation that solely considers the minimum feasible timetables with all exams assigned non-overlapping timeslots.

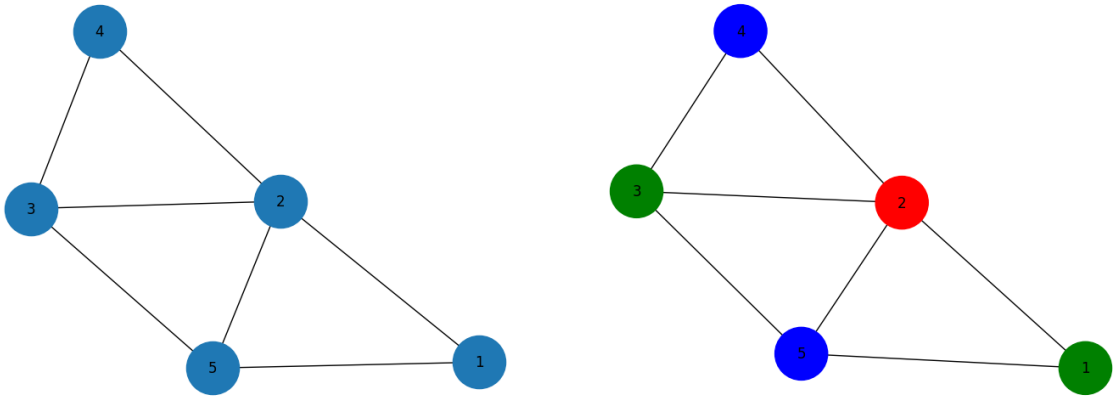
2.2 Graph Representation and the Graph Colouring Problem

The exam timetabling problem can be represented as an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node $v_i \in V$ represents an exam, and an edge $e_{ij} \in E$ exists between nodes v_i and v_j if and only if exams i and j share at least one student. This edge indicates that these exams cannot be scheduled at the same time due to the common students who must attend both.

The goal is to find an optimal or feasible colouring $c : V \to \{1, 2, \ldots, k\}$, where

k is the total number of available timeslots, such that adjacent nodes v_i and v_j (i.e., connected by an edge e_{ij}) are assigned different colours (timeslots), $c(v_i) \neq c(v_j)$. The objective is to minimise k while ensuring that all constraints are met, therefore optimising the schedule to reduce the number of timeslots.

This formulation can be mapped to a graph colouring problem as demonstrated in Figure 2.1, where the colouring constraints represent scheduling conflicts, and the optimisation criterion seeks the minimal chromatic number of the graph. The graph colouring problem evaluated through finding the minimum k colours required to satisfy all constraints is formally known as the chromatic number problem. Satisfying these constraints in the context of ETP ensure that no student is assigned overlapping exams, and the colouring scheme determines the non-conflicting assignment of exams to timeslots.



(a) Unsolved Graph Colouring Problem **(b)** Solved Graph Colouring Problem

Figure 2.1: Comparison of Unsolved and Solved Graph Colouring Problems

It is important to note that an explicit objective function cannot be written for the chromatic number problem. This is because an explicit objective function to directly minimise the number of colours without specifying k in advance is difficult to formulate as k itself is the solution to the optimisation process. Therefore to solve the ETP, the chromatic number problem can be indirectly reformulated through an iterative reduction of k in the k-colouring problem. The k-colouring problem specifically aims to achieve a valid colouring with exactly k colours. By starting with an upperbound of k (such as the number of vertices/exams represented in the graph), it is possible to therefore iteratively test whether the graph is k-colourable by decrementing k until the smallest k results in a valid solution. Once the smallest k resulting in a feasible solution is found, the optimal timetable for the ETP can be produced using k timeslots where each exam that is assigned to a colour is assigned to a corresponding timeslot. For example in the solved ETP presented in Figure 2.1b, a schedule with zero conflicts may be created by assigning exams 1 and 3 to timeslots 1, exams 4 and 5 to timeslot 2, and exam 2 to timeslot 3.

Chapter 3

Methodology

3.1 QUBO Encoding

The Quadratic Unconstrained Binary Optimisation (QUBO) encoding is a mathematical optimisation framework defined by the objective function $y = x^T Q x$, where x is a binary decision variable vector, and Q represents a symmetric matrix of constants. The QUBO serves as a formulation model for various combinatorial optimisation problems including the graph colouring problem and is an essential step in using the QAOA to solve combinatorial optimisation problems. This is because the objective function formulation in QUBO problems are represented as a quadratic polynomial over binary variables (0 or 1), which naturally maps to the quantum states $|0\rangle$ and $|1\rangle$ of qubits.

Within the context of ETP, we can reformulate the ETP as a graph colouring problem as outlined in Section 2.2 to encode the ETP as a QUBO. For this problem, we refer to Glover et. al. [12] methodology to encode the ETP from a graph colouring problem into a QUBO formulation. This QUBO representation has proven to be efficient for a wide variety of colouring instances with hundreds of nodes as demonstrated in Kochenberger, et. al. [13]. We start by producing the equivalent graph colouring problem based on the ETP through $G = (V, E)$, where V is the set of n nodes and E is the set of edges, and K is the number of available colours. We then introduce a binary decision variable $x_{i,k}$ for each node i and colour k, defined as:

$$x_{i,k} = \begin{cases} 1 & \text{if node } i \text{ is assigned colour } k, \\ 0 & \text{otherwise.} \end{cases}$$

where each binary variable $x_{i,k}$ corresponds to a possible node and colour assignment, and node i is assigned colour k if the binary variable is set to one. The result is a vector of binary variables of size $(i \times k)(i \times k)$ where all possible node and colour assignment are represented by a corresponding binary decision variable. The main benefit of the QUBO formulation is its capacity to encode problem constraints as penalty terms within the objective function, eliminating the need to explicitly define constraint sets. This is done by applying a penalty constant P to the coefficients of binary variables such that variables that enforce constraints are given a

higher weight to maximise the chance that the optimal QUBO solution satisfies all constraints.

To apply the penalty constraints to the relevant binary decision variables, a QUBO matrix Q of size $(i \times k)(i \times k)$ is used to represent the coefficients of each binary variable $x_{i,k}$. The QUBO formulation of the graph colouring problem introduces two primary constraints that may be encoded through applying penalty constants:

1. **Node Colour Assignment**: Each node must be assigned exactly one colour, expressed as:
$$\sum_{k=1}^{K} x_{i,k} = 1, \quad \forall i \in V.$$
To enforce this in the QUBO formulation, a penalty term is added to the Q matrix to term penalise any deviation from this constraint, formulated as:
$$P\left(\sum_{k=1}^{K} x_{i,k} - 1\right)^2.$$

2. **Adjacency Constraint**: For an edge (i, j) connecting nodes i and j, the constraint that if node i is assigned colour k, then node j should not be assigned the same colour k can be formulated as:
$$x_{i,k} \cdot x_{j,k} = 0, \quad \forall (i,j) \in E, \quad k \in \{1, \ldots, K\}.$$
the penalty term $P/2$ is applied to entries in Q corresponding to adjacent nodes with the same colour, resulting in a higher cost for QUBO where adjencency constraints are violated.
$$\frac{P}{2}\left(x_{i,k} \cdot x_{j,k}\right).$$

These constraints are incorporated as penalty terms within the QUBO objective function where the P are penalty coefficient enforces the node colouring constraint and P penalises violations of adjacency constraints. The result is the final QUBO Q matrix which encodes the objective function to be minimised:

$$\text{Minimise: } \sum_{i=1}^{nK}\sum_{j=1}^{nK} Q_{i,j} x_i x_j.$$

where minimising the QUBO results the satisfaction of both node colour assignment and adjacency constraints and the QUBO binary variable assignments can be extracted to formulate the colour assignment for each node in the graph colouring problem, and subsequently the ETP. To gain a definitive understanding, of the QUBO encoding, we revisit the graph colouring problem example in Figure 2.1 to understand how each node and edge constraint maps onto the QUBO with corresponding penalty terms. Consider a $K = 3$ graph colouring problem $G = (V, E)$ with $V = \{1, 2, 3, 4, 5\}$ representing five nodes, and the edge set E defined as:

$$E = \{(1,2),(1,5),(2,3),(2,4),(2,5),(3,4),(3,5)\}.$$

Given a penalty constant where $P = 4$ following Q matrix can then be constructed:

$$\begin{bmatrix}
-4 & 4 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
4 & -4 & 4 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
4 & 4 & -4 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
2 & 0 & 0 & -4 & 4 & 4 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 \\
0 & 2 & 0 & 4 & -4 & 4 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 \\
0 & 0 & 2 & 4 & 4 & -4 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 \\
0 & 0 & 0 & 2 & 0 & 0 & -4 & 4 & 4 & 2 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 4 & -4 & 4 & 0 & 2 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 4 & 4 & -4 & 0 & 0 & 2 & 0 & 0 & 2 \\
0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & -4 & 4 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 4 & -4 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 4 & 4 & -4 & 0 & 0 & 0 \\
2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & -4 & 4 & 4 \\
0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 4 & -4 & 4 \\
0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 4 & 4 & -4
\end{bmatrix}$$

The QUBO matrix for the graph colouring problem exhibits a block-diagonal structure for P-related terms, where positive coefficients penalise invalid configurations and negative coefficients encourages valid assignments. Each (3×3) block represents the three possible colour assignments for a single node. The diagonal block entries consisting of $P = 4$ and $-P = -4$ entries enforce the single node colour constraint, and the off diagonal block entries apply a penalty of $P/2 = 2$ to discourage assignment multiple colours to the same node where a conflict is defined. We can observe how minimising the QUBO objective function minimises the penalties corresponding to both conflicts, resulting in a graph colouring node assignment where all constraints are fully satisfied.

Interestingly, we may also observe how the size of matrix Q scales with the number of nodes i and colours k. This is important to note, as the size of the matrix Q determines the amount of qubits required to encode the QUBO as a cost Hamiltonian. Specifically, the number of qubits required to formulate a graph colouring problem with i nodes and K colours is $i \times K$. For example the graph colouring problem disucssed in Figure 2.1 requires 15 qubits to be encoded. It is also worth noting that the number on conflicts or edges in a graph colouring problem does not effect the number of qubits required to represent the problem.

3.2 QAOA Initialisation

In this study, we implement the QAOA using the *QAOA* class from the Qiskit Algorithms library `qiskit_algorithms.minimum_eigensolvers.QAOA` using AerSimulator

to simulate noise-free quantum simulations. The *QAOA* class provides a high-level abstraction for constructing and executing QAOA circuits, this shifts the focus of QAOA implementation through the initialisation of parameter configurations rather than implementing the entire QAOA circuit itself. The *QAOA* class serves as a minimum eigensolver for finding approximate solutions to optimisation problems formulated as finding the minimum eigenvalue of a cost Hamiltonian. As discussed in Section 1.2 it integrates the quantum circuit construction and classical parameter optimisation and solves problem instances by alternatively applying the Cost Hamiltonian and a Mixer Hamiltonian for a predefined number of layers p.

While the *QAOA* class implicitly handles the QAOA circuit, we will briefly outline its methodology for completeness. In the context of solving the Graph Colouring Problem using QAOA, the QUBO formulation is transformed into a Cost Hamiltonian to guide the quantum system toward optimal solutions. The binary variables representing node-colour assignments are mapped to quantum states using Pauli Z operators, where x_i is expressed as $\frac{1-Z_i}{2}$. Substituting this into the QUBO objective function converts it into a sum of terms involving Z_i and Z_iZ_j, resulting in the Cost Hamiltonian H_C. This Hamiltonian encodes the penalties for constraint violations, such as assigning more than one colour to a node or the same colour to adjacent nodes. During QAOA, the system evolves under H_C and a Mixer Hamiltonian, with variational parameters γ and β adjusted to minimise the expectation value of H_C. This iterative process, combining quantum evolution and classical optimisation, enables the identification of feasible colour assignments that satisfy the graph colouring constraints.

In order to initialise the QAOA, the initial cost Hamiltonian, mixing Hamiltonian, and variational parameters β and γ must be defined. As discussed in section 3.1, the objective function of the cost Hamiltonian is defined through the QUBO Q matrix. Using the DOcplex library, this model can then be converted inito a QUBO format that may be used for the QUBO solver by invoking the `from_docplex_mp(mdl)` function. Additionally in the context of utilising the *QAOA* class, defining the mixing Hamiltonian is unnecessary because of the QUBO representation used. The role of the mixing Hamiltonian is to enable transitions between different states of the solution space, but since the QUBO formulates the combinatorial problem that already spans the entire feasible solution space, this is not needed for the QAOA success. As a result, the standard QAOA approach which is implicitly done through the *QAOA* class is to use a simple X-gate-based mixing operator that drives transitions between 0 and 1 for each qubit, without needing a problem-specific or complex mixing Hamiltonian.

For the variational parameters β and γ, a classical optimiser must be selected to initialise these values. There are many classical optimisers available for QAOA offering various strengths and varying levels of performance within various environments. Gradient-free optimisers such as Nelder-Mead foe example, excels in problems with a smooth and unimodal solution space [14], and gradient optimisers such as Adam are suited for high dimensional space problems and noisy gradients [15]. Within the context of ETP, we have chosen COBYLA (Constrained Optimisation BY Linear Approximations) optimiser for parameter tuning. COBYLA is a gradient-free

optimisation algorithm that iteratively refines parameter estimates by approximating the objective function with linear models. COBYLA is particularly well-suited for QAOA for solving the ETP due to its ability to handle noisy, non-differentiable objective functions without requiring gradient information, which aligns with the characteristics of the QUBO constraint encoding for graph colouring problems with high constraints (many edges/conflicts) [16]. Additionally, its efficiency in low-dimensional parameter spaces makes it a practical choice for optimising the QAOA's variational parameters (γ, β) at low circuit depths and for smaller problem instances. As all simulations will be conducted through a quantum simulator, we are limited using low circuits depths ($p = 1$) and to testing small graph colouring problem instances, making COBYLA a strong choice for small problems.

Finally, one last parameter, the penalty constant P must be initialised for the QAOA to encode the graph colouring constraints in the QUBO. In our experiments, we will explore how varrying levels of P interacts with the classical optimiser COBYLA and the overall performance of the QAOA

3.3 Interpreting QAOA Results and Evaluation Metrics

Similar to the QUBO Matrix Q, the QAOA output is a binary array of length $n \times K$ where each element corresponds to whether a particular colour corresponds with a specific node. Every group of K binary variables represents one of each possible colour assignments to a node of the graph colouring problem, and a valid QAOA result will assign exactly one binary variable with a value of 1 for every k group of variables, satisfying the node-colour assignment conflict. Using this binary array, the timetable for the ETP problem may be constructed by grouping variables assigned with 1 across all binary variables that corresponds with the same node, and subsequently assigning those binary variables to timeslots, similar to the methodology outlined in Section 2.1. The resulting schedule can be used to represent the optimal timetabling for the ETP problem.

In our experiment, we will focus on using the QAOA to solve various problem instances later outlined in Chapter 4. The problem instances are small to accommodate for the limitations of simulating a quantum environment using classical hardware, therefore rather than the standard approximation ratio to measure the performance of QAOA, the runtime will be used to measure how long the QAOA takes to produce the optimal solution. This is because since the problem instances are simple, the QAOA will likely result in perfect solutions for all problem instances, so the approximation value will not provide a meaningful result. Finally, we measure how varying levels of the penalty constant P may effect the QAOA performance as well as its interaction with the local optimiser COBYLA.

Chapter 4

Experimental Results

Quantum computing simulations were performed on a machine with an Intel Core Ultra 9 185H (2.5 GHz, 16 Cores, 22 logical processors), NVDIA GeForce RTX 4060 Laptop GPU, 32GB RAM, running Qiskit 1.3.0 on Python 3.13.0, using AerSimulator for noise-free simulations.

4.1 Problem Instances

Table 4.1: Problem Instances for the Exam Timetabling Problem

Instance ID	Number of Exams	Conflicts	Timeslots	Qubits Required
1	2	1	2	4
2	3	2	2	6
3	4	2	2	8
4	3	3	3	9
5	5	3	2	10
6	4	5	3	12
7	5	7	3	15
8	4	6	4	16
9	5	9	4	20

To test the performance of the QAOA, a synthetic dataset $etp\_problems$ containing nine problem instances of the ETP was constructed, each defined by a set number of timeslots, a list of exams, and conflicts between specific exam pairs. The instances vary in complexity, featuring between 2 to 5 exams and 2 to 4 available timeslots. The characteristics of each problem instance is shown in Table 4.1, where the number of conflicts denote the number of exam pairs that cannot be scheduled simultaneously due to shared student enrolments. The problem instances become progressively more complex, requiring more qubits encode each problem instances using a QUBO representation as the number of exam and timeslots required for eaach problem increases. An example encoding of a problem instance as well as a

the graph colouring representation for each problem instance is shown in the appendix. The dataset includes simple scenarios such as problem instance 3 which may be formulated a disconnected graph, and problem instance 8 which may be formulated as a fully connected graph.

4.2 Penalty Constant Tuning and Results

This section presents the results of experiments investigating the impact of varying the penalty constant P on the run time of the QAOA when solving the graph colouring problem. The penalty constant P was applied to the QUBO matrix, with the values $P = 0.1$, 1, 2, 5, and 50 being tested. For each P value, the QAOA was executed three times across nine graph instances (I1 to I9), and the run times were recorded in seconds. If the solver was unable to find a solution within 10 minutes, a dash ('-') was recorded. Table 4.2 displays the run times for each combination of P value, run, and graph instance, along with the average run time for each P value.

Table 4.2: QAOA Runtime Results for Different P Values and Runs Across Instances

P Value & Run	I1	I2	I3	I4	I5	I6	I7	I8	I9
P=0.1, Run 1	4.02	5.33	8.76	7.63	20.03	24.33	239.77	351.13	-
P=0.1, Run 2	3.93	5.28	8.32	4.59	20.11	25.16	259.59	352.49	-
P=0.1, Run 3	8.31	13.37	8.02	6.20	20.60	34.23	170.66	338.87	-
P=0.1, Avg	**5.42**	**8.66**	**8.37**	**6.81**	**20.25**	**27.91**	**223.34**	**347.50**	**-**
P=1, Run 1	0.36	0.28	0.44	1.09	1.61	5.34	64.25	87.55	-
P=1, Run 2	0.35	0.83	0.58	1.07	2.15	4.76	37.97	442.43	-
P=1, Run 3	0.23	0.24	0.51	0.83	1.54	5.30	111.88	118.11	-
P=1, Avg	**0.31**	**0.45**	**0.51**	**1.00**	**1.77**	**5.13**	**71.37**	**216.03**	**-**
P=2, Run 1	0.20	0.32	0.53	0.94	1.42	7.61	57.43	213.98	-
P=2, Run 2	0.19	0.27	0.50	0.90	1.48	5.42	52.45	537.41	-
P=2, Run 3	0.21	0.34	0.85	2.07	1.65	26.53	187.31	108.85	-
P=2, Avg	**0.20**	**0.31**	**0.63**	**1.30**	**1.52**	**13.19**	**99.06**	**286.75**	**-**
P=5, Run 1	0.25	1.17	0.62	2.56	1.57	13.57	342.13	515.30	-
P=5, Run 2	0.23	0.47	0.43	2.82	1.52	15.21	130.78	78.04	-
P=5, Run 3	0.24	0.32	0.52	1.53	1.60	13.33	202.03	252.31	-
P=5, Avg	**0.24**	**0.65**	**0.52**	**2.30**	**1.56**	**14.04**	**224.98**	**281.88**	**-**
P=50, Run 1	1.83	0.23	0.51	0.91	1.39	45.79	-	-	-
P=50, Run 2	1.30	0.23	5.83	7.37	19.41	4.53	42.80	65.70	-
P=50, Run 3	1.81	1.84	5.18	13.73	6.03	4.73	-	-	-
P=50, Avg	**1.65**	**0.77**	**3.84**	**7.34**	**8.94**	**18.35**	**-**	**-**	**-**

Chapter 5

Discussion and Analysis

5.1 QAOA Results Overview and Analysis

The results presented in Table 4.2 show that the QAOA was able to reliably solve all instances of the ETP problems within feasible runtimes, with the exception of problem instance 9, where no QAOA run produced a solution within the 10-minute cut-off threshold. This outcome is expected, as problem instance 9 requires 20 qubits to encode the QUBO objective function into a cost Hamiltonian, compared to 16 qubits for problem instance 8. Since the experiments were conducted in a noiseless quantum simulator environment, the increase from 16 to 20 qubits results in a 16-fold increase in the size of the state vector. Specifically, the state vector for 16 qubits contains $2^{16} = 65,536$ complex amplitudes, whereas for 20 qubits, the state vector contains $2^{20} = 1,048,576$ complex amplitudes, resulting in the $\frac{2^{20}}{2^{16}} = 16$ factor increase. While problem instance 9 is likely feasible given a higher cutoff threshold (1-2 hours), we can observe how the increase in state vector size significantly pushes the limits of what can be efficiently simulated on the classical hardware used in this experiment. In a an environment using real quantum hardware, we can expect the QAOA to solve all problem instances by several orders of magnitude faster, likely solving each instance within milliseconds [17]. However, this comes with the additional challenges from current NISQ devices such as gate errors, decorherence, and noise. Further analysis of the QAOA's performance for more complex problem instances and the reliability of quantum hardware is discussed in later sections.

Apart from problem instance 9, the QAOA was able to produce results for all problem instances, but the QAOA performance varied significantly as different values of the penalty constant was applied to the QUBO encoding. When the smallest penalty constant $P = 0.1$ was used, the average run times were consistently higher across all instances compared to all other values of P. For example, the average runtime of the solver when $P = 0.1$ was 5 to 9 seconds for simple problem instances (I1 to I4) but was solved within a second for almost all other values of P. Despite this, the solver for $P = 0.1$ resulted in the most consistent results, having low deviations in runtime even for more complicated problem instances (e.g. runtimes 351.13, 352.49, 347.50 for I8). This suggests that the initialisation of β and γ parameters determined by the classical optimiser COBYLA largely does not impact the perfor-

mance of the QAOA due to the flatter solution landscape from a smaller penalty constant. Overall while a low penalty constant may provide more consistent and reliable results, a low penalty constant does not sufficiently enforce the constraints imposed by the QUBO formulation, leading to longer computation times.

Conversely, when a greater penalty constant such as $P = 5$ and $P = 50$ was used to encode the QUBO, the results show an overall decrease in runtime compared to $P = 1$ but a significant increase in runtime variability. The variability of runtime is less pronounced for simple problem instances (I1 to I4) but become more apparent for the most complex instance (I6 to I8) where the solvers performance degraded notably. For example when $P = 50$, the solver for example was unable to find solutions within the 10 minute cut-off threshold for two runs when solving instance 7 and 8, and resulted in unexpectedly high runtimes for easier problem instances such as taking 45.79 seconds to solve instance 6. Interestingly, the high variance in runtime performance when the penalty function is high also led to unexpectedly short runtime performances in certain runs; the shortest runtime for instance 7 and 8 was 42.80 and 65.70 seconds respectively when encoding the QUBO with a penalty constant of $P = 50$.

The variability of results can likely be attributed to a combination of the QAOAs stochastic nature and the interaction between a high penalty constant and the COBYlA optimiser. COBYLA behaves like a local optimiser and can sometimes get trapped in local minima, especially for high-dimensional or steep parameter landscapes. This behaviour makes it more suited for problems where the parameter landscape is smooth or where a good initial guess is available. When the penalty constant is increased, the cost function becomes dominated by constraint terms, which may distort the optimisation landscape and create steep regions around feasible solutions. The optimiser may struggle to navigate these steep regions and become trapped in a local minimum where constraints are only partially satisfied, this is likely why the QAOA with $P = 50$ was not able to solve problem instances 7 and 8 for some runs. On the other hand if the randomly initialised QAOA parameters (β and γ) happen to be close to optimal values, the optimiser can quickly find the correct solution. With a steep landscape created by a high penalty constant, starting close to a feasible region can result in rapid convergence, which explains the surprisingly fast runtimes of the QAOA for problem instances 7 and 8 with $P = 5$ and $P = 50$.

Overall, the results suggests there is a trade-off between high and low values of the penalty constant P, where lower values of P offer more reliable results and higher values of P results in faster but increasingly inconsistent results. The overall fastest runtime comes from $P = 1$ and $P = 2$, where $P = 2$ had the fastest average runtime when solving the simplest problem instances (I1 and I2) and $P = 1$ had the fastest average runtime when solving more complex instances (I3 to I8). The moderate values of P appear to balance the enforcement of constraints with computational efficiency, suggesting an optimal range for penalty constants in QAOA. These findings suggest that the choice of penalty constant P is critical when applying QAOA to the graph colouring problem and subsequently the ETP. Future work could explore adaptive methods for selecting P dynamically based on problem characteristics or initial trial runs.

5.2 Larger Combinatorial Problems and NISQ environments

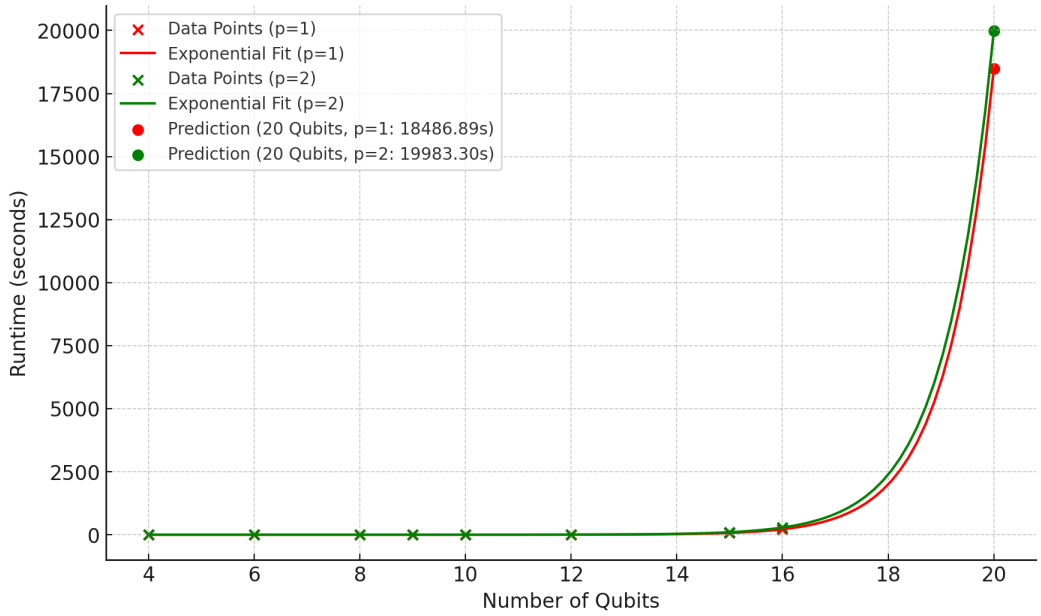


Figure 5.1: QAOA Runtime vs Number of Qubits for $p = 1$ and $p = 2$.

The QAOA was able to produce results for all problem instances except for instance 9 within the 10 minute threshold cut-off threshold. However it is likely that the solver would be able to find a solution for problem instance 9 given a longer cut-off threshold. In Figure 9, we plot the exponential fit for QAOA runtime performances based on the runtime of each problem instance and the corresponding number of required qubits to represent the problem in the QUBO. As expected the runtime increases exponentially with the number of qubits, with a approximated runtime of 308.1 and 333.05 minutes for values of $p = 2$ and $p = 1$ respectively.

The difference in runtime performance between penalty values $p = 1$ and $p = 2$ in the Quantum Approximate Optimisation Algorithm (QAOA) experiments is primarily attributed to the effect of penalties on the complexity of the cost landscape in Quadratic Unconstrained Binary Optimisation (QUBO) problems. Higher penalty values, such as $p = 2$, impose stricter constraints compared to lower values like $p = 1$. This stricter enforcement can create a more complex optimisation landscape characterised by steeper gradients and an increased number of local minima. Consequently, COBYLA may encounter greater difficulty in converging to optimal solutions, necessitating additional iterations and thereby increasing the overall runtime. Moreover, higher penalties may result in deeper quantum circuits to encode the constraints, further contributing to longer execution times due to increased computational complexity. In contrast, lower penalties like $p = 1$ enforce constraints less rigidly, producing a smoother landscape that facilitates more efficient navigation by the optimiser, potentially leading to faster convergence and reduced runtime.

In practical implementations of QAOA on real quantum hardware, noise introduces significant challenges that are absent in noiseless simulations. Noise in quantum environment may appear in various sources, including gate errors, measurement errors, and decoherence, all of which degrade computational accuracy [17]. Gate errors accumulate with increasing circuit depth, causing deviations from the true optimal solution and requiring the optimiser to perform additional iterations to compensate. Decoherence, which results in the loss of quantum states over time, limits the feasible circuit depth, reducing the quality of solutions for larger problem instances.

Furthermore, noise introduces variability in results, making the convergence process less consistent and more difficult to achieve. Higher penalty values may exacerbate these challenges by increasing the sensitivity of the optimisation landscape to noise, whereas moderate penalties may offer a balance between enforcing constraints and maintaining noise resilience. While error mitigation techniques can help alleviate some of these issues, they introduce additional computational overhead. While noiseless simulators provide an idealised baseline for performance, real quantum hardware imposes practical constraints that can significantly increase runtime and compromise solution quality.

5.3 Further Practical Applications for The Graph Colouring Problem

We have demonstrated how university scheduling problems can be transformed into the graph colouring problem and subsequently solved using QAOA. However this is only one possible area of application for the graph colouring problem, the QAOA, designed for solving combinatorial optimisation problems, has broad applicability to real-world challenges beyond exam timetabling, particularly through its potential to address the graph colouring problem. In telecommunications, QAOA can optimise frequency assignment by efficiently searching for low-conflict colourings in large, complex networks, thereby reducing interference between adjacent transmitters. In logistics and transportation, QAOA can enhance vehicle routing and traffic flow management by assigning routes or time slots with minimal overlap. Similarly, in computer science, QAOA can be leveraged for register allocation by finding optimal mappings of variables to a constrained set of registers, mitigating conflicts in computation. In circuit design, QAOA can optimise wire layouts by identifying configurations that minimise interference. The strength of QAOA lies in its ability to explore large solution spaces efficiently, offering the potential for quantum advantage in problems where classical algorithms face computational bottlenecks. Therefore, by solving graph colouring problems more effectively, QAOA provides a versatile tool for tackling a wide range of industrial optimisation tasks that require conflict minimisation and resource allocation

Chapter 6

Conclusion

This paper explored the application of the Quantum Approximate Optimisation Algorithm (QAOA) to the classic University Timetabling Problem (UTP) by encoding it as a graph colouring problem and reformulating it as a Quadratic Unconstrained Binary Optimisation (QUBO) problem. The findings demonstrate that QAOA is capable of solving small instances of the exam timetabling problem effectively, offering promising insights into the viability of quantum algorithms for real-world combinatorial optimisation tasks.

Experimental results indicated that the choice of the penalty constant P significantly influences QAOA's performance. Moderate values of P produced optimal trade-offs between runtime efficiency and solution reliability, while extremely low or high values led to either longer computation times or inconsistent results. The use of the COBYLA optimiser further highlighted the challenges of navigating complex optimisation landscapes, especially as problem size increased. Though the QAOA was limited by hardware constraints in simulating larger problem instances, the results suggest that quantum hardware advancements could unlock more substantial benefits in solving larger, real-world timetabling challenges.

Beyond the UTP, the versatility of the QAOA for solving graph colouring problems offers potential applications in various fields, such as telecommunications, logistics, and circuit design. As quantum computing technology evolves, addressing current challenges like gate errors, decoherence, and noise will be critical to realising the full potential of QAOA in solving complex optimisation problems more efficiently than classical methods.

Future research could focus on enhancing the scalability of QAOA, exploring adaptive methods for penalty constant selection, and benchmarking performance on larger datasets using real quantum hardware. By advancing these areas, QAOA holds the potential to revolutionise the field of combinatorial optimisation, offering speedups and improved solutions for problems that are currently intractable with classical techniques.

Bibliography

1. Burke EK and Petrovic S. Recent research directions in automated timetabling. European Journal of Operational Research 2002; 140:266–80. DOI: 10.1016/S0377-2217(02)00068-1
2. Carter MW and Laporte G. Recent developments in practical examination timetabling. *Practice and Theory of Automated Timetabling*. Ed. by Burke EK and Ross P. Springer, 1996 :3–21. DOI: 10.1007/3-540-61469-2_44
3. Kirkpatrick S, Gelatt CD, and Vecchi MP. Optimization by simulated annealing. Science 1983; 220:671–80
4. Kurowski K, Pecyna T, Slysz M, Różycki R, Waligóra G, and Wglarz J. Application of quantum approximate optimization algorithm to job shop scheduling problem. European Journal of Operational Research 2023; 310:518–28. DOI: https://doi.org/10.1016/j.ejor.2023.03.013. Available from: https://www.sciencedirect.com/science/article/pii/S0377221723002072
5. Farhi E, Goldstone J, and Gutmann S. A Quantum Approximate Optimization Algorithm. arXiv preprint arXiv:1411.4028 2014. Available from: https://arxiv.org/abs/1411.4028
6. Farhi E, Goldstone J, Gutmann S, and Sipser M. Quantum Adiabatic Evolution Algorithm for Finding Minimum Eigenvalues. arXiv preprint arXiv:quant-ph/0001106 2000. Available from: https://arxiv.org/abs/quant-ph/0001106
7. Goemans MX and Williamson DP. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. Journal of the ACM 1995; 42:1115–45. DOI: 10.1145/227683.227684
8. Basso J, Farhi E, Marwaha K, Villalonga B, and Zhou L. The Quantum Approximate Optimization Algorithm at High Depth for MaxCut on Large-Girth Regular Graphs and the Sherrington-Kirkpatrick Model. arXiv preprint arXiv:2110.14206v3 2022 Jul. Available from: https://arxiv.org/abs/2110.14206v3
9. Hastings MB. Classical and Quantum Bounded Depth Approximation Algorithms. arXiv preprint arXiv:1905.07047 2019. Version 2, August 1, 2019. Available from: https://arxiv.org/abs/1905.07047
10. Farhi E and Harrow AW. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. arXiv preprint arXiv:1602.07674v2 2019 Oct. Available from: https://arxiv.org/abs/1602.07674v2

11. Lucas A. Ising formulations of many NP problems. Frontiers in Physics 2014 Feb; 2:5. DOI: 10.3389/fphy.2014.00005. Available from: https://www.frontiersin.org/articles/10.3389/fphy.2014.00005/full

12. Glover F, Kochenberger G, and Du Y. A Tutorial on Formulating and Using QUBO Models. arXiv preprint arXiv:1811.11538 2019. Available from: https://arxiv.org/abs/1811.11538

13. Kochenberger GA, Glover F, Alidaee B, and Rego C. An Unconstrained Quadratic Binary Programming Approach to the Vertex Coloring Problem. Annals of Operations Research 2005; 139:229–41

14. Ronayne JG and Kendon V. Performance of classical optimization algorithms on the quantum approximate optimization algorithm. Quantum Information Processing 2021; 20:1–21

15. Mohseni M, Liu AY, and Neven H. Comparing classical optimizers for QAOA on noisy quantum devices. arXiv preprint arXiv:2307.10149 2023

16. Nannicini G. Performance of hybrid quantum/classical variational heuristics for combinatorial optimization. arXiv preprint arXiv:1805.12037 2018 Dec. Available from: https://arxiv.org/abs/1805.12037

17. Singhal S, Srivastava V, Rohith P, Jain P, and Bhowmik D. Performance Comparison for Quantum Approximate Optimization Algorithm (QAOA) across Noiseless Simulation, Experimentally Benchmarked Noisy Simulation, and Experimental Hardware Platforms. *2024 8th IEEE Electron Devices Technology Manufacturing Conference (EDTM)*. 2024 :1–3. DOI: 10.1109/EDTM58488.2024.10512142