QUBIT EFFICIENT QUANTUM OPTIMIZATION AND APPLICATIONS IN ROUTING AND SCHEDULING PROBLEMS

by

IOANNIS D. LEONIDAS

A THESIS SUBMITTED FOR THE DEGREE OF M.Sc.

TECHNICAL UNIVERSITY OF CRETE

2024

Supervisor:

Associate Professor Dimitris G. Angelakis

Examiners:

Professor Retired Demosthenes Ellinas

Professor Thrasyvoulos Spyropoulos

Acknowledgements

Contents

Abstract

This thesis presents a novel approach for solving route and scheduling problems of the Quadratic Unconstrained Binary Optimization (QUBO) type using a novel quantum algorithm developed with collaborators at the Centre for Quantum Technologies in Singapore. The algorithm allows the mapping of classical binary variables to $\log_2(N) + 1$ qubits allowing for implementation of industrial level problems in near term quantum computers. We start the work by attacking a problem from the shipping industry known as the Vehicle Routing Problem with Time Windows (VRPTW), which we first cast in QUBO format. We then study how to adapt the qubit efficient algorithm to the problem at hand for different parameter regimes and constraints and design the relevant quantum circuits. We run the circuits on simulators first and pick the optimal ones to implement on real quantum computers on the cloud using superconducting and ion-based qubits from provided by AWS (IONQ, Riggetti) and IBMQ. We demonstrate that is possible to solve problem instances of 128 and 3964 classical variables using only 8 and 13 qubits, well beyond capabilities of standard approaches based on the quantum approximate optimization algorithm (QAOA). We benchmark our results with the standard binary-to-qubit mappings used in QAOA and standard commercial solvers such as Gurobi find excellent agreement. Next, we introduce a novel reinforcement learning (RL) enhancement algorithm that can be used on top of our qubit efficient encoding to further enhance the quality of solutions obtained. In the final two chapters, we formulate as QUBO and then solve two more optimization problems from the aviation industry, namely the Tail Assignment Problem (TAP) and the Flight Gate Assignment (FGA) to test the effectiveness of the enhanced algorithm in tackling problems up to 25000 classical variables. Our simulator results show that using the enhanced RL pipeline one can find solutions belonging to the top 1% of the solution space.

Author research milestones

I. D. Leonidas, A. Dukakis, B. Tan, and D. G. Angelakis, **"Qubit efficient quantum algorithms for the vehicle routing problem on NISQ processors"**, 2023. arXiv: 2306.08507 [quant-ph].

Accepted to be presented at the American Physical Society (APS) March Meeting 2024, Session: K49.00004 : Qubit efficient quantum algorithms for the vehicle routing problem on NISQ processors, Authors : I. D. Leonidas, A. Dukakis, B. Tan, and D. G. Angelakis

Accepted at Advanced Quantum Technologies journal: Paper with title "Qubit efficient quantum algorithms for the vehicle routing problem on quantum computers of the NISQ era".

List of Tables

List of Figures

1 Introduction

Quadratic Unconstrained Binary Optimization (QUBO) problems are a thoroughly researched category within optimization problems. Their relevance is underscored by the ability to reframe numerous industry-specific optimization problems into QUBO format [1]–[11]. Solving industry relevant optimization problems not only enhances a company's resource utilization, but also contributes to environmental sustainability by lowering the carbon emissions from operations [12]–[14].

The field of quantum computing has recently made strides, leading to the creation of tools that utilize emerging quantum techniques for finding approximate answers to QUBO problems [15], [16]. This includes Quantum Annealing (QA) [17], [18] and various variational methods like the Quantum Approximate Optimization Algorithm (QAOA) [19]–[24] or parameterized quantum circuits optimized for hardware efficiency [25]–[29]. Nonetheless, the application of these quantum algorithms in real-world industry-scale problems is hampered by the limitations of current Noisy Intermediate Scale Quantum (NISQ) devices [15], [30]–[32]. To overcome this, significant effort has been directed towards reducing the number of binary variables required in solving these problems. This includes restructuring classical problems to lessen binary variable requirements [33]–[37], adopting divide-and-conquer strategies to deconstruct problems or quantum circuits [38]–[51], and employing various qubit encoding techniques [52]–[55], among other strategies [56]–[62].

Recent advancements in quantum computing have catalyzed the development of inno-

vative algorithms, significantly enhancing the capability to solve complex combinatorial optimization problems. These advancements are epitomized by algorithms such as the Quantum Enhanced Greedy Solver [63], Reinforcement Learning Quantum Local Search (RL-QLS) [64], and the integration of reinforcement learning with quantum-inspired algorithms [65].

The Quantum Enhanced Greedy Solver, developed by Dupont et al. [63], represents a pivotal step in quantum algorithmic design. It utilizes an iterative quantum heuristic optimization approach, showing performance on par with classical greedy algorithms even amidst quantum hardware noise. This algorithm's implementation on a programmable superconducting quantum system has demonstrated superior performance compared to classical counterparts, indicating a clear quantum enhancement in output quality. It has shown promising results in solving Sherrington-Kirkpatrick Ising spin glass problems, thus opening new avenues in the quantum computing field for addressing complex optimization tasks.

In the domain of quantum local search, the work by Liu et al. [64] introduces the RL-QLS approach. This method enhances the efficiency of Quantum Local Search by training a reinforcement learning agent for improved sub-problem selection, beyond random selection. The integration of RL techniques with NISQ algorithms demonstrates the RL agent's effectiveness in boosting the average approximation ratio for fully-connected random Ising problems. This research signifies a promising direction for merging RL into quantum computing, thereby enhancing the performance of optimization tasks.

Furthermore, the work by Beloborodov et al. [65] discusses the application of reinforcement learning in enhancing quantum-inspired algorithms. Their approach involves a reinforcement learning agent in conjunction with a quantum-inspired algorithm to solve the Ising energy minimization problem, equivalent to the Maximum Cut problem. This integration allows for the sampling of high-quality solutions with a higher probability and outperforms both baseline heuristics and black-box hyperparameter optimization

approaches.

In this thesis, we apply a qubit efficient encoding scheme proposed by Tan et al. [55], called *minimal encoding*, to solve variationally optimization problems of industry relevant sizes. In particular we solve 3 optimization problems, namely the Vehicle Routing Problem with Time Windows (VRPTW), the Tail Assignment Problem (TAP) and the Gate Assignment Problem (FGA). For VRPTW we used different quantum computers developed by industry leading companies through the cloud to solve problem instances of up to 3964 classical variables. Results show the effectiveness of the qubit efficient encoding scheme in solving large optimization problems in the NISQ era, but also it's downside in finding clusters of approximate solutions rather than the global optimum solution. To further increase the problem sizes we introduce a Reinforcement Learning (RL) based enhancement of the quality of solutions obtained by the optimization with minimal encoding. As a consequence, we were able to tackle problems of up to 25000 classical variables using up to 16 qubits in quantum simulators for the quantum circuit and a single laptop GPU for the RL enhancement algorithm.

This thesis is organized as follows : **Background section:** We explain the basics of QUBO and quantum optimization as well as classical and quantum-classical algorithms for solving optimization problems. **Qubit Efficient Route Optimization:** This chapter first explains the qubit efficient encoding scheme introduced in [55] and shows how we can apply it to the Vehicle Routing Problem with Time Windows in order to tackle larger problem sizes using less qubits in real quantum hardware from three industry leading companies. We also introduce a reinforcement learning enhancement algorithm designed to further optimize the solutions found by the quantum optimization part. **Qubit Efficient Optimization for the Tail Assignment Problem:** In this chapter we formulate the TAP as a QUBO based on [1] and explain how we generate TAP instances of specific size. In the last section of this chapter we show results after applying the RL enhanced pipeline for a plethora of problem sizes and discuss the approximation

ratio obtained. **Qubit Efficient Optimization for the Flight Gate Assignment Problem:** This chapter explains the problem and formulation as introduced in [66] and then shows how we can map this problem to a QUBO form. We also explain the procedure of generating QUBO matrices of specific size and provide details about the problem instances considered in this work. Then we apply the RL enhanced pipeline to solve GAS problems and discuss the results. **Appendix:** This section contains a brief extrapolation on the resources needed to solve the optimization problems considered in this thesis using the IBM Cairo quantum computer. Also it provides pseudocodes for the main algorithms for QUBO optimization that were used in this thesis.

2 Background

2.1 Quadratic Unconstrained Binary Optimization

Quadratic Unconstrained Binary Optimization (QUBO) is the problem of finding an assignment $\mathbf{x^*}$ of n binary variables that is minimal with respect to a cost function quadratic in the binary variables where

$$C(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} = \sum_{i,j=1}^{n} Q_{i,j} x_i x_j \tag{2.1}$$

is the cost function to be minimized/maximized, $\mathbf{x} = (x_1, \ldots, x_n)^T$ is the vector of unknown decision variables and the optimal solution $\mathbf{x^*}$ can be expressed as:

$$\mathbf{x^*} = \arg\min_{\mathbf{x} \in \{0,1\}^n} \mathbf{C(x)} = \arg\max_{\mathbf{x} \in \{0,1\}^n} -\mathbf{C(x)} \tag{2.2}$$

It is important to note that QUBO problems are unconstrained, namely there are no constraints on the variables $\mathbf{x}$, although constraints can be enforced as penalty terms in the cost function.

2.1.1 Mapping of QUBOs to Ising models

QUBO instances can also be mapped into Ising models. Ising models replace the QUBO binary variables $\mathbf{x} \in \{0,1\}^n$ with Ising or spin variables $\mathbf{s} \in \{-1,1\}^n$, such that $s_i = 2x_i - 1$ for $i = 1, \ldots, n$. The final Ising Hamiltonian, which depends on $\mathbf{s}$ is equivalent

to Eq. (2.1) up to a constant irrelevant to the optimization [67], [68].

To find solutions to the QUBO problem using traditional quantum approaches, the cost function is first mapped to an Ising Hamiltonian:

$$\hat{H}_{\text{ising}} = \frac{1}{4} \sum_{i,j=1}^{n} Q_{i,j}(1 - \hat{\sigma}_z^{(i)})(1 - \hat{\sigma}_z^{(j)}) \tag{2.3}$$

where $\hat{\sigma}_z^{(i)}$ is the Pauli Z matrix, with eigenvalues -1 and 1, acting on qubit i and $Q_{i,j}$ are the elements of the QUBO matrix $\mathbf{Q}$. The ground state of $\hat{H}_{\text{ising}}$ is a basis state $|x_i\rangle$ that corresponds to an exact solution $\vec{x_i}$ of the QUBO problem defined by $\mathbf{Q}$.

Due to its equivalence with Ising models, QUBO represents a class of problems suitable to be solved using adiabatic quantum computers such as quantum annealers [69].

2.1.2 Overview of some Classical Algorithms for Solving QUBOs

QUBOs, by their very nature, involve the optimization of quadratic functions over a binary domain. While this might seem straightforward, the combinatorial nature of these problems makes them inherently challenging [80].

In Table (2.1), we provide a summary of the evolution of classical algorithms used to solve optimization problems in general. While these algorithms were initially developed for a broad spectrum of optimization challenges, they have found applicability in the domain of QUBO problems as well [81]–[90].

It is important to note that Gurobi [91] and CPLEX by IBM [92], two of the state of the art commercial solvers, are using a combination of Branch and Bound, Dynamic Programming and Cutting Planes algorithms.

Type	Method (Year)	Description
Exact	Branch and Bound (1966)	A systematic method that enumerates candidate solutions via relaxation problems. Complexity is exponential with the problem size [70].
	Dynamic Programming (1966)	Leverages underlying problem structure for efficient solving [71].
Heuristic	Simulated Annealing (1993)	Explores solution space by occasionally accepting non-improvements, guided by a temperature parameter [72].
	Tabu Search (1998)	Uses memory structures to explore the solution space beyond local optima [73].
	Genetic Algorithms (1992)	Operates on a population of solutions, evolving better solutions over iterations [74].
Hybrid	Hybrid Genetic Algorithms (2006)	Blends genetic algorithms with other optimization techniques [75].
	GRASP (1995)	A multi-start metaheuristic combining solution construction with local search [76].
Decomposition	Lagrangian Relaxation (2001)	Uses Lagrange multipliers to relax certain constraints, making the optimization problem more tractable [77].
Iterative	Gradient Descent (1993)	An optimization algorithm that iteratively adjusts variables to minimize a given function, commonly used for continuous functions [78].
Cutting Plane	Cutting Plane Methods (1960)	Refines a problem's relaxation by adding linear inequalities [79].

Table 2.1: Classical optimization algorithms.

2.2 Quantum Adiabatic Optimization

Quantum Annealing (QA) and the Quantum Adiabatic Algorithm (QAA) are closely related quantum computational techniques that leverage the principles of quantum mechanics to solve optimization problems. At their core, both approaches are grounded in the quantum adiabatic theorem, which ensures that a quantum system remains in its ground state during a slow, continuous evolution of its Hamiltonian, provided there is no crossing (degeneracy) between its eigenstates.

Quantum annealing, primarily utilized for solving combinatorial optimization problems, begins with a quantum system in a superposition of all possible candidate solutions. It employs a time-dependent Hamiltonian $\hat{H}(t)$ that transitions from an initial simple Hamiltonian $\hat{H}_{\text{init}}$ to a final Hamiltonian $\hat{H}_{\text{final}}$ encoding the optimization problem. This gradual evolution allows the system to exploit quantum tunneling, a phenomenon enabling it to escape local minima and potentially find the global minimum more efficiently than classical algorithms. Quantum annealing has shown promise in various fields, including optimization, machine learning, finance, and materials science [93]–[96] typically in machines developed by DWave Inc. [97].

The Quantum Adiabatic Algorithm extends the principle of adiabatic evolution to solve optimization problems on a quantum computer, forming a subset of Adiabatic Quantum Computing (AQC). QAA involves evolving from an easily preparable ground state of an initial Hamiltonian $\hat{H}_{\text{M}}$ to the ground state of a final Hamiltonian $\hat{H}_{\text{C}}$, which represents the solution to the problem. The transition follows a predefined path described by a transitional Hamiltonian $\hat{H}(t) = f(t)\hat{H}_C + g(t)\hat{H}_M$, enabling the system to remain in its instantaneous ground state throughout the evolution.

Both QA and QAA can be implemented on different types of quantum computing platforms, including specialized annealers and gate-based quantum computers. Gate-based implementations can discretize the continuous evolution of QAA using the Trotter-

Suzuki formula, allowing for a stepwise approximation of the adiabatic process. This flexibility underscores the adaptability and potential of adiabatic quantum computation in addressing complex optimization challenges.

In a study conducted by Crosson et al. [98], numerical simulations of the QAA were carried out for over 200,000 instances of MAX2-SAT on 20 qubits, each with a unique optimal solution. For instances where the success probability at $T = 100$ was less than 10^{-4}, three strategies were proposed to enhance the success probability. The first strategy involved running the adiabatic algorithm more rapidly, leading to increased success probabilities at shorter times for all instances. The second strategy focused on initializing the system in a random first excited state of the problem Hamiltonian, resulting in an average success probability close to the upper bound for most challenging instances. Lastly, the third strategy introduced a random local Hamiltonian into the middle of the adiabatic path, often boosting the success probability. These strategies were also tested on the QAA version of the Grover search algorithm, but they did not yield improvements in the success probability. The authors concluded that these strategies might be beneficial primarily for exceptionally difficult instances and suggested testing them on a quantum computer with a higher number of qubits.

In conclusion, Quantum Annealing and the Quantum Adiabatic Algorithm represent significant advancements in quantum computing's ability to tackle complex optimization problems. Their development and implementation continue to be a vibrant area of research within the quantum computing community, promising to unlock new possibilities across various scientific and industrial domains.

2.3 Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA) is a quantum algorithm introduced by Farhi et al. [19] for tackling optimization problems. It represents a novel approach combining quantum computing and classical optimization techniques. The algorithm operates in a hybrid quantum-classical fashion, where the quantum computer is used to prepare states that encode possible solutions, and a classical computer is used for optimization.
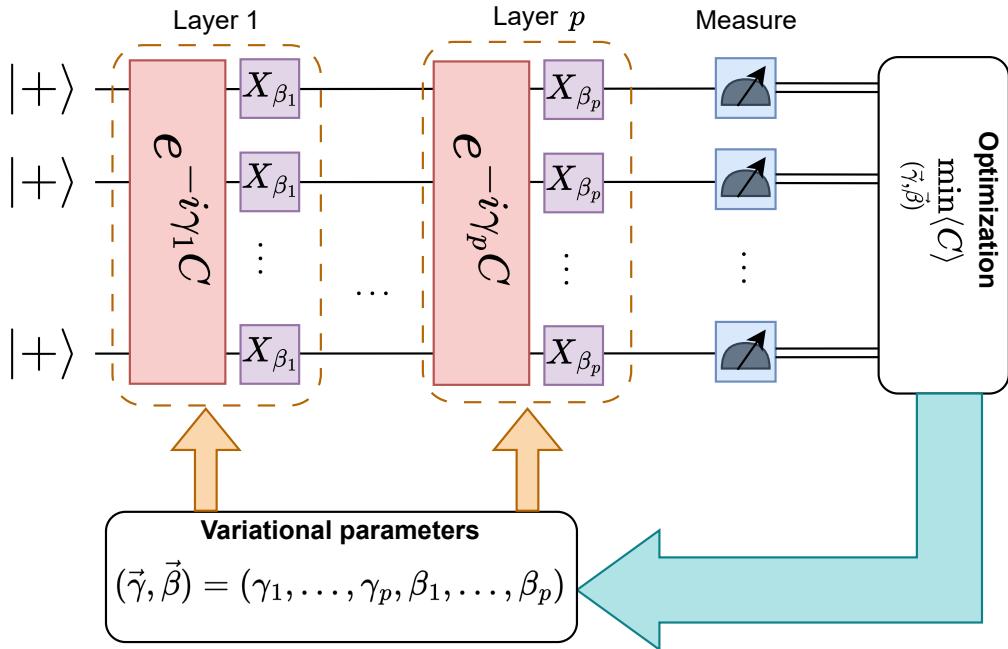
Figure 2.1: Optimization procedure for the QAOA. All qubits start from a superposition after applying Hadamard gates to all. Operators $U(C, \gamma)$ (red) and $U(B, \beta)$ (purple) are being applied for each layer of QAOA. At the end we measure all the qubits in the computational basis, calculate the mean value of the cost operator and optimize this quantity over all variational parameters using a classical optimization algorithm.

The core of QAOA is a parameterized quantum circuit with $p \geq 1$ layers as seen in figure (2.1). This circuit consists of two types of unitary operators applied repeatedly in

alternation: The **problem unitary** $U(C, \gamma)$, which encodes the problem's cost function C. The cost function is a function of the angle γ and is defined as $U(C, \gamma) = e^{-i\gamma C}$. The **mixing unitary** $U(B, \beta)$, which provides quantum exploration of the solution space. It depends on the angle β and is typically defined as a series of X-rotations $U(B, \beta) = e^{-i\beta B}$, where B is a sum of Pauli-X operators.

The performance of QAOA depends on the depth of the circuit (number of repetitions of the unitaries), with higher depths generally providing better approximations to the optimal solution. Various modifications and extensions to the standard QAOA have been proposed to enhance its performance and applicability.

In the original QAOA paper, the authors discuss the relation to QAA described in section (2.2) with QAOA. Especially they mention that as QAA is proven to converge if given enough time, QAOA can approximate this scenario by applying the two operators $U(C, \gamma)$ and $U(B, \beta)$ where the sum of the angles is the total runtime using a large number of layers, perhaps exponentially large in the system size. Although, there are also some key differences that the authors argue, like the fact that in QAOA as p grows the approximation gets better while this is not always the case for the QAA (we kindly refer the reader to citations within the original QAOA paper for a more detailed discussion on this).

QAOA has been applied to a range of combinatorial problems as described in this very helpful survey [99]. Its hybrid quantum-classical nature makes it particularly suitable for near-term quantum computers. QAOA stands as a significant development in quantum algorithms, showing potential for solving complex optimization problems that are challenging for classical computers. Its ongoing development and refinement continue to be an active area of research in quantum computing.

2.4 Variational Quantum Eigensolver

The Variational Quantum Eigensolver (VQE) is often recognized as the algorithm that initiated the exploration of Variational Quantum Algorithms (VQAs) [100], [101]. The VQE represents a groundbreaking quantum algorithm that combines the capabilities of both quantum and classical computing to address the challenge of determining eigenvalues for a given Hamiltonian operator initially introduced by Peruzzo et al. [100]. It showcased its prowess in tackling quantum chemistry problems by utilizing a hybrid system that integrates a photonic quantum processor with a classical computer. Subsequent advancements by McClean et al. [102] further refined and solidified its theoretical underpinnings. The VQE allows for the probabilistic measurement of observables over certain classes of parameterized approximate wavefunctions, which can neither be sampled from nor have their properties computed efficiently (e.g. in polynomial time) on conventional devices as the system size gets large [103].

The VQE algorithm relies on a parametrized quantum circuit, often referred to as an ansatz, defined by a set of parameters denoted as $\boldsymbol{\theta}$. The optimization of these ansatz parameters is a systematic process, aiming to find the best possible solution through the application of the variational principle on a quantum computer. This entire variational process can be symbolized by a unitary gate $U(\boldsymbol{\theta})$. The ansatz operates on the initial state of an N-qubit quantum circuit, commonly chosen as the ground state $|0\rangle$, resulting in an output state $U(\boldsymbol{\theta})|\psi_0\rangle = |\psi(\boldsymbol{\theta})\rangle$.

From this setup, it becomes evident that $U(\boldsymbol{\theta})|\psi_0\rangle$ represents a normalized wave function, enabling the formulation of the optimization problem as follows:

$$\lambda = \min_{\boldsymbol{\theta}} \langle\psi_0| U^{\dagger}(\boldsymbol{\theta})\hat{H}U(\boldsymbol{\theta}) |\psi_0\rangle \tag{2.4}$$

This state undergoes iterative optimization by adjusting the parameters $\boldsymbol{\theta}$ to find the optimal parameter set, $\boldsymbol{\theta}$*, ultimately determining the optimized expectation value:

21

$$\lambda_{\min} = E_0 \approx \langle\psi(\boldsymbol{\theta})|\,\hat{H}\,|\psi(\boldsymbol{\theta})\rangle \tag{2.5}$$

VQAs draw inspiration from the variational principle of quantum mechanics. This principle seeks to determine the lowest expectation value of a specific observable, often the ground state energy, using a trial wave function. The uniqueness of the trial wave function lies in its parametrization, which can adapt to fit a general wave function to the system and identify the least expectation value. With a Hamiltonian $\hat{H}$ and a trial wave function $|\psi\rangle$, the ground state energy E_0 is constrained by:

$$E_0 \leq \frac{\langle\psi|\,\hat{H}\,|\psi\rangle}{\langle\psi|\psi\rangle} \tag{2.6}$$

Within this context, VQAs aim to tailor the parametrization of $|\psi\rangle$ to minimize the Hamiltonian's expectation value. The process entails refining an ansatz by successively enhancing its initial trial wave function and parameter set. Usually, the parameter initialization occurs randomly within a reasonable scope, considering the quantum system.

Determining the right ansatz is challenging, with multiple strategies based on the system's Hamiltonian and the nature of the problem. For instance, in quantum chemistry, discerning a helium atom's ground state requires an ansatz formed from two hydrogen atom wave functions, which is further refined. Another popular method is the hardware-efficient ansatz shown in figure (2.2)[1], built using one and two-qubit gates native to the hardware. Its merits lie in its reduced circuit depth and adaptability, making it suitable for diverse problems. This flexibility in crafting an ansatz for a variational task facilitates the adoption of quantum computers. They can represent Hamiltonians as unitary operations, and the amalgamation of these operations helps craft an easily adjustable ansatz wave function using the Bloch sphere rotation angles.

The concept of combining quantum and classical computers to handle different aspects

---

[1]Throughout this thesis we are using the hardware efficient ansatz as a parametrized quantum circuit for all problems solved.
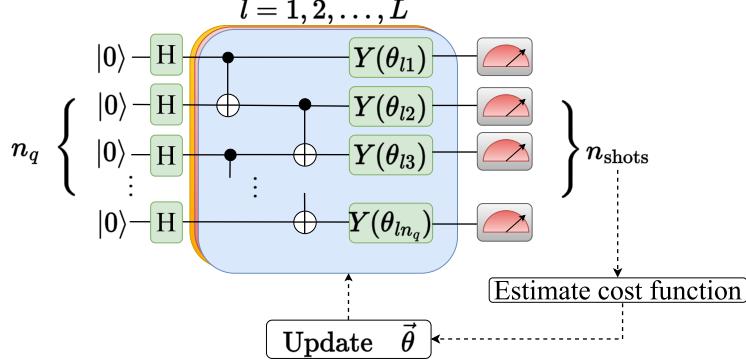
Figure 2.2: Hardware efficient variational ansatz. Each circuit begins with a layer of Hadamard gates followed by L repetitions of CNOT gates and single qubit $Y(\theta_i)$ rotations. The circuit is measured in the computational basis to estimate the respective cost functions, and a classical optimizer is used to update the variational parameters $\vec{\theta}$.

of a complex problem is known as hybrid quantum-classical computing and forms the core of any VQA [102].

Furthermore, this hybrid approach proves particularly valuable in numerous use cases, especially with NISQ devices, as quantum computers are less efficient and fault-tolerant than their classical counterparts for certain tasks, such as parameter optimization. Consequently, the computational load is distributed between the two systems to maximize overall performance. In this hybrid regime, the task of fine-tuning the parameters is outsourced to classical computers, which subsequently relay the optimized values to the quantum computer for eigenvalue calculations. The classical computer computes the new parameters through optimization techniques, typically based on gradient descent, which calculates a hyperplane of error or deviation from the ideal solution to locate a minimum point that signifies the highest model accuracy.

It's important to note, however, that hybrid computing is not always the most efficient solution, as some algorithms exhibit greater power when exclusively relying on quantum hardware, as demonstrated by Magann et al. [104]. Kandala et al. [105] utilized a medium-sized quantum computer to optimize Hamiltonian problems involving

up to six qubits and over one hundred Pauli terms. They successfully determined the ground-state energy for molecules like BEH_2 . Their approach involved a VQE, the efficient preparation of trial states tailored to the available quantum processor interactions, and a robust stochastic optimization algorithm. Their results shed light on the requirements for scaling the method to larger systems and bridging the gap between high-performance computing problems and their implementation on quantum hardware. Recent variations of VQE have also been proposed to effectively address combinatorial optimization problems [29], [106].

2.4.1 Barren Plateaus

When dealing with parametrized quantum circuits like VQE and QAOA during training, a significant obstacle emerges concerning the landscape of the cost function. Specifically, for certain sets of parametrized quantum circuits, the cost function landscape can appear quite flat. This flatness implies that the gradients associated with the adjustable parameters become exceedingly small as the number of qubits increases, leading to a stagnation of the optimization process. This phenomenon is commonly referred to as the *barren plateaus* problem [107], wherein the gradient of the cost function with respect to any parameter exponentially diminishes as the number of qubits grows. In more formal terms, a cost function $C(\boldsymbol{\theta})$ exhibits barren plateaus if, for all trainable parameters $\theta_i \in \boldsymbol{\theta}$, the variance of the partial derivative of the cost function decreases exponentially with respect to the number n of qubits:

$$\text{Var}_{\boldsymbol{\theta}}[\partial_i C(\boldsymbol{\theta})] \leq F(n) \tag{2.7}$$

where $F(n) \in O(b^{-n})$ for a constant $b > 1$. Equation (2.7) indicates that, on average, the gradient of the cost function becomes exponentially small. According to Chebyshev's inequality, the likelihood of the partial derivative $\partial_i C(\boldsymbol{\theta})$ deviating from its average

value (which is zero) by more than a specified constant c, with $c > 0$, is bounded by $\text{Var}_{\boldsymbol{\theta}}[\partial_i C(\boldsymbol{\theta})]$, expressed as:

$$Pr[|\partial_i C(\boldsymbol{\theta})| \geq c] \leq \frac{1}{c^2} \text{Var}_{\boldsymbol{\theta}}[\partial_i C(\boldsymbol{\theta})] \tag{2.8}$$

2.4.2 Parameter Shift Rule for Quantum Circuits

The parameter shift rule [108] is a crucial technique for calculating gradients in quantum circuits, especially in the context of variational quantum algorithms. This rule addresses the challenge of computing derivatives of quantum expectation values with respect to gate parameters.

Consider a quantum circuit with a parameterized gate $U(\theta) = e^{-i\theta H}$, where θ is the parameter and H is a Hermitian operator. The expectation value of an observable O in the state prepared by this circuit is given by:

$$\langle O \rangle(\theta) = \langle \psi | U^{\dagger}(\theta) O U(\theta) | \psi \rangle, \tag{2.9}$$

where $|\psi\rangle$ is the initial state of the system.

The gradient of $\langle O \rangle(\theta)$ with respect to θ can be estimated using the parameter shift rule. This rule states that for a gate of the form $U(\theta) = e^{-i\theta H}$, where $H^2 = I$ (the identity operator), the derivative of the expectation value can be expressed as:

$$\frac{d\langle O \rangle}{d\theta} = \frac{\langle O \rangle(\theta + \pi/2) - \langle O \rangle(\theta - \pi/2)}{2}. \tag{2.10}$$

The parameter shift rule allows for the estimation of gradients without the need for finite difference methods, which can be less accurate and more resource-intensive. This is particularly beneficial in quantum computing, where resources (such as qubits and gate operations) are limited and costly. By applying this rule, gradients can be calculated efficiently, enabling more effective optimization in variational quantum algorithms.

While the parameter shift rule is powerful, it has limitations. It assumes that $H^2 = I$, which is not true for all Hermitian operators. Generalizations of this rule exist for more complex cases, involving shifts by different angles and potentially more terms in the gradient estimation formula.

3 Qubit Efficient Route Optimization

This chapter summarizes the material written in the paper :

- I. D. Leonidas, A. Dukakis, B. Tan, and D. G. Angelakis, **"Qubit efficient quantum algorithms for the vehicle routing problem on quantum computers of the NISQ era"**, 2023. arXiv: 2306.08507 [quant-ph].

- **Accepted to be presented at the American Physical Society (APS) March Meeting 2024**, Session: K49.00004 : Qubit efficient quantum algorithms for the vehicle routing problem on NISQ processors, Authors : I. D. Leonidas, A. Dukakis, B. Tan, and D. G. Angelakis.

- The results of this work are accepted for publication at Advanced Quantum Technologies journal, published by Wiley.

3.1 Introduction

We start off by formulating the VRPTW using a greedy heuristic introduced in [2]. We acknowledge discussion with Dimitar Trenev and Stuart Harwood from [2] who kindly agreed on explaining their paper to us and helped us formulate the VRPTW instances covered in this thesis. After formulating the problem, we show how we can map it to a QUBO using the traditional encoding scheme of mapping one qubit to one classical variable, and how we map it using the minimal encoding. Then we compare these two encoding schemes in small problem instances of $n_c = 8, 16$ classical variables using real quantum hardware. To further advance the robustness of the encoding scheme, we employ it to larger problem instances of $n_c = 128, 3964$ classical variables using only up to 13 qubits and we compare results with the classical commercial solver Gurobi.

Lastly, we introduce a reinforcement learning algorithm that can be used after the qubit efficient algorithm to further enhance the quality of solutions obtained allowing us to solve even bigger problem instance of up to 26000 classical variables using up to 16 qubits and a single laptop GPU.

3.2 Qubit Efficient Quantum Algorithms for the Vehicle Routing Problem

3.2.1 Vehicle Routing Problems with Time Windows

Vehicle Routing Problems (VRP) are a well explored and widely applicable family of problems within logistics and operations and are a generalized form of the well-known Travelling Salesman Problem (TSP) [109]. The overarching goal of these problems is to manage and dispatch a fleet of vehicles to complete a set of deliveries or service customers while trying to minimize a total cost. Many variations of this problem exist, oftentimes seeking to maximize profits or minimize travel costs [12], [110].

The Vehicle Routing Problem with Time Windows (VRPTW) is a VRP variant that enforces time windows within which individual deliveries must be made [111]–[113]. Reference [2] provides 3 different methods of mapping the VRPTW to a QUBO, of which we will follow the denser, route-based formulation, requiring fewer binary variables for the same number of destinations as the other formulations.

A problem instance is characterized by a network of nodes $\mathcal{N}$. This set consists of N nodes representing different customers and an additional 0th node, d to serve as the "depot" node, which must be the initial departure point and final destination for all vehicles $v \in \mathcal{V}$. Nodes are connected by directed arcs, $(i, j) \in \mathcal{E}$, each of which has an associated cost c_{ij}. These costs are frequently functions of the travel distance or travel time between two nodes, but other measures can also be used. The problem may be specified further by assigning each non-depot node a demand level. For the sake of this work, all vehicles are assumed to be homogeneous both in speed and capacity.

Each node i has an associated time window $[a_i, b_i]$ and can only be serviced after time a_i and before time b_i (although we may permit vehicles to arrive earlier and wait until a_i). The depot can be treated as having a time window of $[0, +\infty]$ and the effective arrival time at node i_{p+1} is given by $T_{i_{p+1}} = \max\{a_{i_{p+1}}, T_{i_p} + t_{i_p, i_{p+1}}\}$.

According to this route-based formulation, valid solutions to the problem instance will inform whether or not a route should be travelled. We define a route r as an ordered sequence of P nodes $(i_1, i_2, ..., i_P)$ for a vehicle to travel to. To satisfy the requirement that vehicles must start and end at the depot, we require that $i_1 = i_P = d$. In order for a route to be considered valid, it must be composed entirely of valid segments: $(i_p, i_{p+1}) \in \mathcal{E} \ \forall \ 1 \leq p \leq P - 1$.

We must also ensure that the arrival time T_{i_p} for any given node p along route i, must not exceed the upper limit of that node's time window, i.e. $T_{i_p} \leq b_{i_p} \forall p$. This set $\mathcal{R}$ containing all valid routes defines the size of the problem. For a route $r \in \mathcal{R}$ with sequence $(i_1, i_2, ..., i_P)$, we can calculate a route cost $c_r = \sum_{p=1}^{P-1} c_{i_p, i_{p+1}}$. We also define

(a) Vehicle routing problem

(b) Mapping to Binary variables

(c) Complete encoding
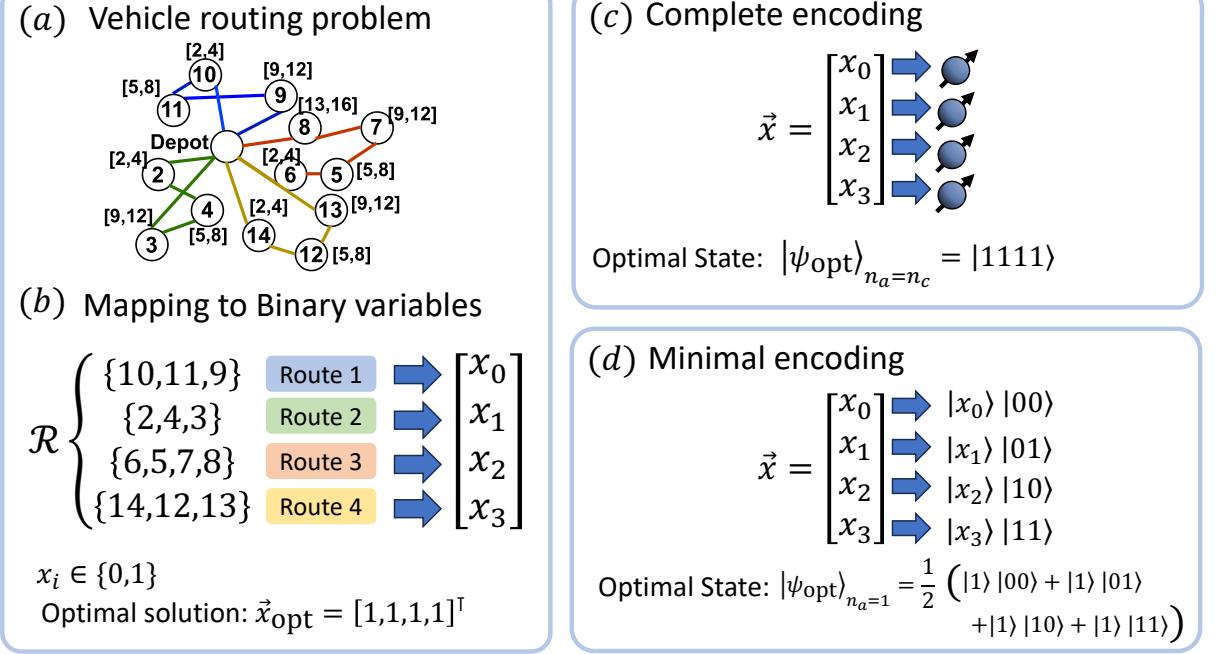
(d) Minimal encoding

Figure 3.1: Mapping of a simple 4 route VRPTW optimization problem to a quantum device using complete (or full) and minimal encoding. a) Graph of the original optimization problem. Each node in the graph is associated with a location id and a time window. b) After obtaining the set of feasible routes [2], we generate the optimization problem by assigning a binary variable to every route. c) Visual representation of the full encoding, showing a mapping of a binary variable to a single qubit on the quantum device. The optimal quantum state using this encoding is a single basis state, which is the ground state of the Ising Hamiltonian representing the problem. d) Visual representation of the minimal encoding using 3 qubits to represent this 4-route VRPTW problem by mapping the binary variables to basis states spanned by the qubits. The optimal quantum state using this encoding scheme is now a superposition of the register qubits followed by the appropriate value of the ancilla qubit, as shown for the optimal classical solution in (b).

a value $\delta_{i,r}$ which is equal to 1 if node i lies in route r and 0 otherwise. This allows us to express the objective of our VRPTW in terms of a vector $\vec{x}$, containing decision variables x_r corresponding to each route r:

$$\min_{\vec{x}} \sum_r^R c_r x_r, \qquad x_r \in \{0,1\} \ \forall \ r \in \mathcal{R} \tag{3.1}$$

$$\text{s.t.} \sum_r^R \delta_{ir} x_r = 1, \qquad \forall \ i \in \mathcal{N} \tag{3.2}$$

where x_r is a binary variable that has value 1 if route r is to be travelled, and 0 if not. The linear equality constraint (3.2) ensures that all nodes in the network are visited exactly once within our optimal solution. As an additional optional constraint, it may be desirable to set the number of used vehicles to some fixed value V. This is achieved with the additional condition:

$$\sum_r^R \delta_{0r} x_r = V$$

where we have used the nomenclature of $\delta_{0,r}$ referring to the inclusion of the depot node in route r. This is included purely for illustrative purposes, as all valid routes must include the depot node and therefore $\delta_{0,r} = 1 \ \forall \ r \in \mathcal{R}$.

A toy example of the VRPTW problem with 14 destinations without constraints on the number of vehicles is shown in figure (3.1)a. In this example, we show the destinations being grouped into 4 routes, with each route starting and ending at the depot. This can be done using a classical heuristic or otherwise, to remove routes that do not respect the time windows or unallowed routes due to other constraints (e.g. untraversable paths, long travel time between nodes, etc). Figure (3.1)b shows how we set up the VRPTW by assigning a binary variable to each route and figures (3.1)c & d shows the full (standard encoding used to map binary variables to qubits) and minimal encoding respectively.

For an instance of VRPTW specified by valid routes $\mathcal{R}$, the optimal route for fully

Number of routes	Number of locations	Number of classical variables	Number of qubits using minimal encoding
11	13	11	5
16	14	16	5
128	23	128	8
687	33	687	11
1285	39	1285	12
3964	63	3964	13
6081	80	6081	14
9644	110	9644	15
14095	148	14095	15
20565	200	20565	16
25382	242	25832	16

Table 3.1: Information for all the VRPTW instances covered. These instances were created using the heuristic proposed in [2]. The number of locations includes demand, supply and dummy nodes.

connected problems with N nodes rapidly becomes computationally infeasible, with potential routes scaling as $R_{max} = \sum_{i=1}^{N} \frac{N!}{(N-i)!}$. In real-world scenarios, full connectivity may not always be present, and $R \ll R_{max}$.

To convert the route set into a QUBO problem, the classical optimization cost function is expanded with penalty terms to represent constraints, resulting in a new unconstrained cost function:

$$\min_{\vec{x}} \sum_{r \in R} c_r x_r + \rho \sum_{i \in N} \left(\sum_{r \in R} \delta_{ir} x_r - 1 \right)^2 \tag{3.3}$$

where ρ is a positive penalty coefficient, ensuring violations are penalized. The above equation can be written in matrix form as follows:

$$\min_{\vec{x}} \vec{c}^{\top} \vec{x} + \rho \vec{x}^{\top} \delta^{\top} \delta \vec{x} - 2\rho (\vec{1}^{\top} \delta) \vec{x} + \rho N \tag{3.4}$$

By setting matrix $Q = \text{diag}(\vec{c}) + \rho \delta^{\top} \delta - \text{diag}(2\rho \vec{1}^{\top} \delta)$, where $\text{diag}(\vec{c})$ denotes a matrix with $\vec{c}$ in its diagonal, we arrive at the definition of a QUBO problem in (2.1).

3.2.2 Systematic Encoding of Binary Variables

The encoding scheme in [55] starts off by dividing the n_c classical variables within the QUBO problem into subgroups of n_a variables. n_a qubits are then used to represent the values of the binary variables within those subgroups, and the basis states of $n_r = \log_2(\frac{n_c}{n_a})$ register qubits are used as addresses to denote which subgroup should take on the values that the ancilla qubits represent. The smallest possible subgroup allowed consists of only one classical variable per subgroup where $n_a = 1$. We refer to this grouping as the *minimal encoding*. The largest possible subgroup allowed is when the subgroup consists of all the classical variables, i.e. $n_a = n_c$, in which case only one possible subgroup exists and no register qubits are needed i.e. $n_q = n_a = n_c$. We refer to this as the *full encoding* and is the one-to-one qubit to variable mapping typically used in the standard approaches described above.

Minimal Encoding

The parametrized quantum state can be expressed as

$$\left|\psi_{\mathrm{me}}(\vec{\theta})\right\rangle = \sum_{k=1}^{n_c} \beta_k(\vec{\theta})[a_k(\vec{\theta})\left|0\right\rangle_a + b_k(\vec{\theta})\left|1\right\rangle_a] \otimes \left|\phi_k\right\rangle_r \tag{3.5}$$

where $\{\left|0\right\rangle_a, \left|1\right\rangle_a\}$ refer to the quantum state of the $n_a = 1$ qubit and $\{\left|\phi_k\right\rangle_r\}$ refer to the quantum state of the register qubits. To derive the cost function for the minimal encoding, we start from a QUBO cost function that also takes into the account the probability of sampling a bitstring $\vec{x_i}$:

$$C_x = \sum_{i=1}^{2^{n_c}} \vec{x_i}^\top \mathbf{Q} \vec{x_i} P(\vec{x_i}) \tag{3.6}$$

However, the probability $P(\vec{x_i})$ of obtaining a particular bitstring $\vec{x_i}$ is now calculated differently using the state given in (3.5).

From this state, we can construct a probability distribution using the coefficients to obtain the probability of sampling a bitstring $\vec{x}$ from the state. This can be done by letting the probability of the kth binary variable in $\vec{x}$ being $x_k = 0$ and $x_k = 1$ be $P(x_k = 0) = |a_k(\vec{\theta})|^2$ and $P(x_k = 1) = |b_k(\vec{\theta})|^2$ respectively. This gives the total probability $P(\vec{x_i})$ of obtaining a particular bitstring $\vec{x_i}$ as

$$P(\vec{x_i})_{\vec{\theta}} = \prod_{k=1}^{n_c} |b_k(\vec{\theta})|^2 \tag{3.7}$$

$$= \prod_{k=1}^{n_c} P(x_k)_{\vec{\theta}} \tag{3.8}$$

for a given set of $\vec{\theta}$. To obtain a classical bitstring post optimization, coefficients of the optimal quantum state $\left|\psi_{\mathrm{me}}(\vec{\theta}_{\mathrm{opt}})\right\rangle$ are estimated using n_{shots} number of shots to obtain the $b_i(\vec{\theta}_{\mathrm{opt}})$ coefficients in eq.(3.5), from which we can construct the probability distribution above. Multiple classical solutions $\vec{x}$ can then be sampled from this probability distribution by assigning the kth bit in each classical solution to be $x_k = 1$ with a probability of $P(x_k = 1) = |b_k|^2$ and $x_k = 0$ with probability $P(x_k = 1) = 1 - |b_k|^2 = |a_k|^2$.

The minimal encoding allows us to reduce the number of qubits required for a problem with n_c classical variables to $n_q = 1 + \log_2(n_c)$ qubits, the largest reduction possible using the encoding scheme.

To estimate the cost function for optimization using the encoding scheme, it is possible to use the variational cost function in Eq. (3.6) by measuring the output state of the quantum circuit, constructing the probability distribution in Eq. (3.8) as described above, and then sampling multiple classical solutions. $P(\vec{x_i})$ in Eq.(3.6) is estimated by calculating the fraction of each unique sample $\vec{x_i}$ obtained, over all samples. However, this results in a slight classical overhead at each optimization step as the quantum state is used to construct a probability distribution to sample classical solutions to estimate eq. (3.6).

By substituting the probability distribution $\prod_{k=1}^{n_c} P(x_k)$ into $P(\vec{x}_i)$ in Eq.(3.6), we can obtain a cost function that directly depends on the coefficients in our minimal encoding quantum state (3.5). When expressed in the form of projectors, this becomes:

$$C_{\mathrm{me}}(\vec{\theta}) = \sum_{k \neq l}^{n_c} A_{kl} \frac{\langle \hat{P}_k^1 \rangle_{\vec{\theta}} \langle \hat{P}_l^1 \rangle_{\vec{\theta}}}{\langle \hat{P}_k \rangle_{\vec{\theta}} \langle \hat{P}_l \rangle_{\vec{\theta}}} + \sum_{k=1}^{n_c} A_{kk} \frac{\langle \hat{P}_k^1 \rangle_{\vec{\theta}}}{\langle \hat{P}_k \rangle_{\vec{\theta}}} \tag{3.9}$$

$$= \sum_{k \neq l}^{n_c} A_{kl} |b_k|_{\vec{\theta}}^2 |b_l|_{\vec{\theta}}^2 + \sum_{k=1}^{n_c} A_{kk} |b_k|_{\vec{\theta}}^2 \tag{3.10}$$

which is the minimal encoding cost function as described in [55]. The projectors $\hat{P}_k = |\phi_k\rangle \langle \phi_k|_r$ are projectors over the register basis states $|\phi_k\rangle_r$ and $\hat{P}_k^1 = |1\rangle \langle 1|_a \otimes \hat{P}_k$ are projectors over the states where the ancilla is in the $|1\rangle_a$ state. This removes the need to reconstruct classical bitstrings $\vec{x}_i$ to evaluate multiple values of $\vec{x}_i^\top \mathcal{A} \vec{x}_i$ at each optimization step.

This minimal encoding cost function is a sum over n_c^2 number of terms, compared to regular encoding approaches that minimize $\langle \hat{H}_{\mathrm{Ising}} \rangle$. Evaluating $\langle \hat{H}_{\mathrm{Ising}} \rangle$ requires summing over a maximum of 2^{n_c} terms in the limit of $n_{\mathrm{shots}} \to \infty$, or at most n_{shots} number of terms if $n_{\mathrm{shots}} < 2^{n_c}$ shots are used.

For small number of shots used to estimate (3.9), some of the register states may not be measured, leading to $\langle \hat{P}_k \rangle_{\vec{\theta}} = 0$ in the denominator. Intuitively, this can be interpreted as not having any information on whether to assign the kth bit to be $x_k = 0$ or $x_k = 1$ (with their respective probabilities). In such cases, we manually set $\frac{\langle \hat{P}_k^1 \rangle_{\vec{\theta}}}{\langle \hat{P}_k \rangle_{\vec{\theta}}} = 0.5$. This is interpreted as randomly guessing the value of the kth bit to be 0 or 1 with a 50% probability.

The minimal encoding, while being able to exponentially reduce the number of qubits required to find solutions to a QUBO problem, comes at a cost of being able to capture classical correlations between the binary variables during sampling. This can be seen from Eq. (3.8) which, for fixed $\vec{\theta}$ describes a probability distribution of multiple indepen-

dent variables where the outcome of sampling the kth bit is independent of the outcome of sampling another bit. During optimization however, these marginal probabilities depend on the same set of variational parameters, $\vec{\theta}$, when using the ansatz in figure (2.2). Adjusting $\vec{\theta}$ to change $P(x_k = 1)$ for the kth bit may affect the value $P(x_l = 1)$ of the lth bit. Regardless, unless the optimal state is able to produce $P(x_k = 1) = 0$ or $P(x_k = 1) = 1$ exactly, it is difficult for the minimal encoding state to produce a sharply peaked distribution around the optimal distribution.

Another trade-off of the minimal encoding is the number of shots needed to properly characterize a classical solution. In traditional mappings of 1 qubit to a binary variable, each measurement of the quantum circuit yields a single solution $\vec{x}$. For the minimal encoding, multiple shots are needed to properly characterize the $|b_k|^2$ coefficients in the optimal quantum state. For large problem sizes, this can result in large number of shots, especially if the corresponding register probabilities, $|\beta_k|^2$ are very small. Failure to measure these probabilities results in the issues mentioned above, where the values of certain bits have to be guessed, resulting in poorer solutions.

Full Encoding

Problems solved using the full encoding scheme will have each classical variable mapped to its own qubit. The full encoded quantum state is given by

$$\left| \psi_c(\vec{\theta}) \right\rangle = \hat{U}_c(\vec{\theta}) \left| \psi_0 \right\rangle = \sum_{i=1}^{2^{n_c}} \alpha_i(\vec{\theta}) \left| x_i \right\rangle, \tag{3.11}$$

where $\hat{U}_c(\vec{\theta})$ is the unitary evolution implemented on the quantum computer and the probability of sampling the ith solution is given by the $|\alpha_i(\vec{\theta})|^2$. The cost function is calculated according to (3.6). Each measurement of the circuit results in a basis state that represents a specific bitstring x_i, whose cost is equal to $C_{\text{fe}} = \vec{x}_i^T \mathbf{Q} \vec{x}_i$.

3.3 Implementation of real quantum computers on the cloud

Cloud quantum computing merges quantum computing and cloud technology, offering remote access to quantum processors via the internet, thus eliminating the need for personal, expensive quantum hardware. This union transforms the accessibility model of quantum computing, making it more feasible for a wider array of users, thereby promoting a more diverse innovation and scalability in the field. However, it also faces challenges such as limited quantum resources, data security concerns, and the technical complexity of quantum algorithms. Despite these obstacles, cloud quantum computing is vital for progress in quantum technologies, with significant contributions from major platforms like IBM Quantum Experience [114], Amazon Braket [115], Google Quantum AI [116], Microsoft Azure Quantum [117], and IonQ [118], each providing various tools and access to different quantum systems. These platforms are pivotal in advancing research, developing quantum applications, and preparing for the future of computation, making cloud quantum computing a cornerstone of the impending technological revolution. We acknowledge support from CQT at NUS for accessing IBM and AWS quantum devices and performing experiments.

IBM and Rigetti use superconducting qubits [119], which operate by creating and manipulating superconducting circuits at extremely low temperatures to maintain quantum coherence. IonQ on the other hand, employs trapped ion technology, using individual ions as qubits and manipulating them with lasers [120]. The performance of these qubits can be characterized by parameters such as coherence times (T_1 and T_2), and fidelities [1] for single and two-qubit operations.

- T_1 - The energy relaxation time, or the time it takes for a qubit to decay from its excited state to its ground state.

- T_2 - The dephasing time, or the time over which a qubit loses its phase information.

---

[1]Qubit fidelity is a measure of accuracy and reliability of a qubit or a quantum operation

- Qubit number - The total number of qubits in the system.

Quantum Device	Technology	Qubits	T_1 (µs)	T_2 (µs)	Fidelity % Single Qubit	Fidelity % Two Qubit
IBM Cairo	Superconducting	27	91.91	102.32	99.9924	99.998
IBM Guadalupe	Superconducting	16	N/A	N/A	N/A	N/A
IonQ Aria-1	Trapped Ion	21	10^7	10^6	99.95	99.6
IonQ Harmony	Trapped Ion	11	10^7	10^6	99.6	97.3
Rigetti AspenM2	Superconducting	80	31	18	99.8	91.3

Table 3.2: Specifications of various quantum devices including technology, qubit number, coherence times, and average fidelity for single and two qubit operations.

Table [2] (3.2) shows specifics about the quantum computers we used via cloud in order to solve VRPTW instances with sizes shown in (3.1). We observe that Rigetti AspenM2 has the highest number of qubits and the lowest T_1 and T_2 which means that running deep circuits with many layers is prohibited. Although, IBM Cairo which uses the same superconducting technology has higher T_1 and T_2 times compared with AspenM2, still, is order of magnitudes behind compared with the ion technology. Since ions are trapped atoms it is expected to have longer decoherence times as long as they don't fall out of the trap. Ion technology also offers all to all qubit connectivity in contrast with the superconducting technology which has a fixed architecture. We must also note that IBM has the highest average fidelity which is very useful since we want our qubit operations to have as less errors as possible. In figure (3.2) we show an example of a 5 qubit ion trap and the architecture of the superconducting IBM Cairo quantum machine. Using lasers in the ion technology, one can manipulate pairs of qubits independently, while in

---

[2]As of 2024 IBM Guadalupe has been deprecated from the IBM quantum-ecosystem and its details are not available. For the IBM Cairo quantum machine a more detailed practical analysis can be found in the appendix (7.1).
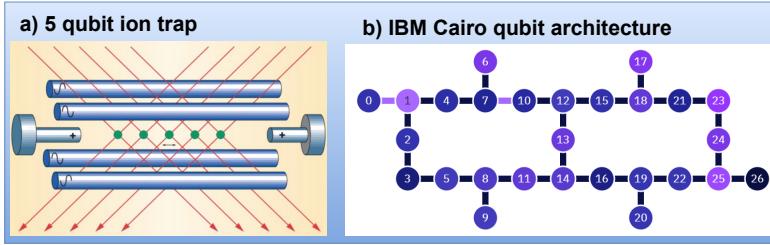
Figure 3.2: Different state of the art quantum computer architectures. **a)** Diagram of a 5 qubit ion trap. Ions are trapped by the electrodes and laser beams are used to perform quantum transormations on the ions. **b)** Qubit architecture of the IBM Cairo machine. Nodes are actual physical qubits in the machine and edges represent a two qubit operation between qubits. Purple color shows higher T_1 times for the qubits and higher controlled-not gate errors for the edges.

the IBM Cairo architecture we can perform two qubit operations only between qubits that are connected with an edge.

Overall, quantum machines are still in the NISQ era and the current fidelity of qubit operations does not allow their scalability. Even if we only consider the single and two qubit fidelity shown here, having a quantum computer with 1000 qubits would result in a machine that only produces noise. Nevertheless, the technology behind quantum computers is an active area of research and progress is being made everyday.

3.3.1 Solving VRPTW Using Different Cloud Quantum Computers

Each problem is set up with a basic variational ansatz with $L = 4$ layers of the gates exemplified in figure (2.2). We used ADAM [121], a gradient based classical optimizer, to find the optimal parameters for our circuit. During each iteration, the parameter shift rule [122] was used to calculate the gradient of each parameter. The final parameters were used to run optimal circuits using a noise-free simulator in Pennylane [123] and, depending on the problem sizes, on the cairo or guadalupe quantum backends from IBMQ [114], the Aspen-M2 quantum backend from Rigetti via the AWS Braket [115], and on the Harmony and Aria-1 quantum backends from IonQ [118]. In all problem

instances for both the minimal and full encoding case, 20 randomly seeded starting points, $\vec{\theta}_{\text{init}}$, were used. 10 classical solutions were sampled from the post-optimization quantum state for a total of 200 classical solutions per problem instance, and compared according to a cumulative distribution of their normalized cost function value:

$$C_{\text{norm}} = \frac{C(\vec{\theta}) - C_{\text{min}}}{C_{\text{max}} - C_{\text{min}}} \qquad (3.12)$$

where C_{min} and C_{max} are the minimum and maximum cost function values associated with the problem instances, found using Gurobi [91]. Classical solutions were drawn from the minimal encoding quantum state as described in section (3.2.2). In the case of full encoded quantum states, to obtain the probability distribution of the solutions we simply measure the quantum circuit in the computational basis. By doing so, we are able to compare these two encoding techniques based on the classical solutions obtained from each method and their associated costs for each VRPTW instance.

Comparison of Minimal Encoding with Full Encoding Scheme

The optimization process and the derived approximate solutions for the smaller problem sizes with $n_c = 11, 16$ are displayed in figures (3.3) and (3.4) respectively. The optimization utilized varying numbers of shots, n_{shots}, based on the problem size and were evaluated against results obtained with full access to the statevector, essentially $n_{shots} \to \infty$. For these smaller problem sizes, the number of shots needed to achieve a solution comparable to statevector simulation is within the reach of current quantum technology. ADAM, with an adequate number of shots, achieves comparable solution quality in both minimal and full encoding scenarios, even though minimal encoding uses fewer qubits.

When considering classical solutions derived from quantum states from IBMQ quantum backends, they consistently performed better than those from IonQ and Rigetti

backends in both minimal and full encoding scenarios. Furthermore, figures (3.3) and (3.4) show that IonQ devices yield higher quality solutions for the fully encoded cases as opposed to the minimally encoded ones. However, due to scheduling constraints, instances with fewer qubits, like those in minimal encoding, were run on Harmony instead of Aria-1 qpu, and additional analysis is needed to understand these performance differences for IonQ backends.

Figure (3.5) compares the distribution of classical solutions from minimal and full encodings for both the 11R and 16R instances, using IBMQ quantum devices, given their similarity to simulated outcomes.

In both 11R and 16R scenarios, a significant number of classical solutions from both encoding methods resulted in similar quality solutions across all optimization attempts. However, the minimal encoding state showed a more concentrated clustering of sampled solutions compared to the full encoding. In specific problems like the VRPTW instances here, this is attributed to the minimal encoding state converging to a state that has a high probability of yielding a single, low-cost solution. Sampling classical solutions from this final state as per (3.8) often results in this favorable solution. Nevertheless, the inability to capture classical correlations in the minimal encoding means that solutions with poor cost function values can still be sampled with a nonzero probability.

In contrast, in the full encoding scenario, a good cost function value indicates the maximization of multiple viable solutions with low costs, ideally centering on the optimal solution. Therefore, sampling from such a post-optimized state in full encoding is more likely to yield other solutions with close cost function values.
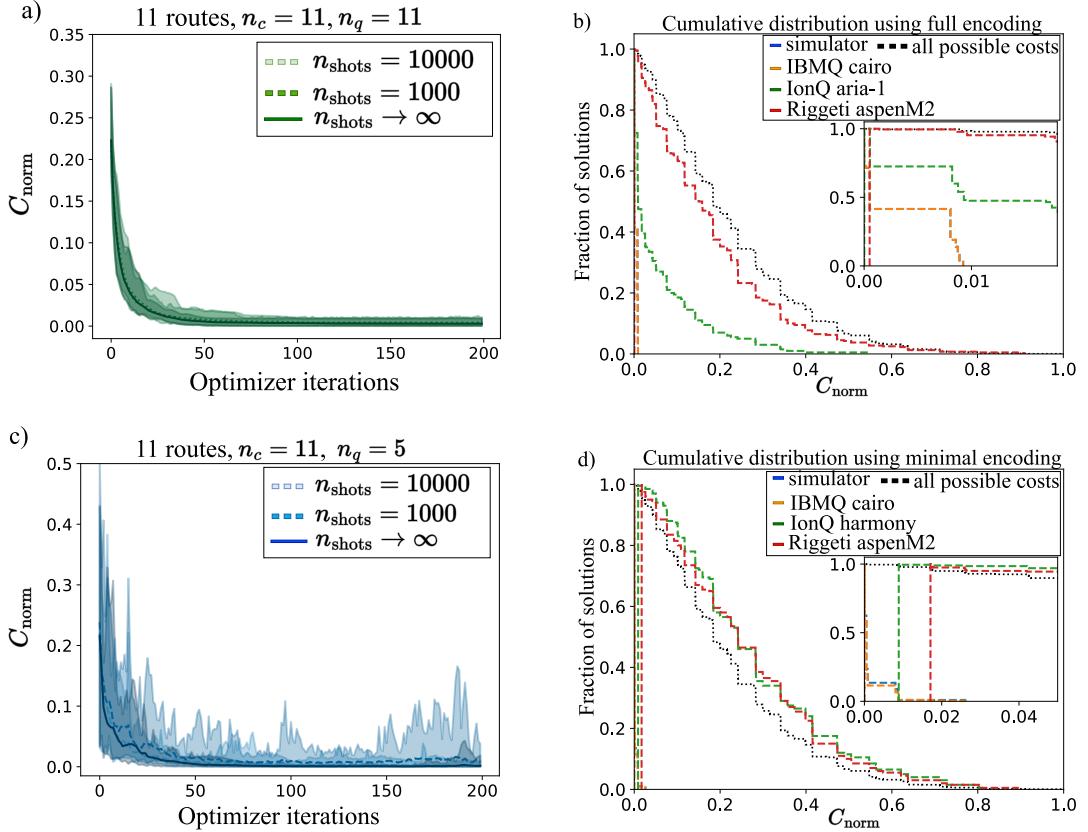
Figure 3.3: Comparison between a $n_c = 11$ route optimization instance, using full and minimal encoding. a, c) Convergence graphs for the full encoding and minimal encoding schemes with $n_{\text{shots}} = 1000, 10000$, and $n_{\text{shots}} \to \infty$. Shaded regions show the deviation of the normalized cost function values per optimizer iteration. b, d) Cumulative distribution of classical solutions obtained post optimization using full and minimal encoding respectively. Black curves show the distribution of C_{norm} values of all 2^{n_c} classical solutions. Blue curves show classical bitstrings obtained using $n_{\text{shots}} = 10000$ shots from simulator. Orange, green, and red curves show the classical bitstrings obtained using the optimal parameters from optimization, on IBMQ, IonQ, and on Riggeti quantum backends. Insets show the zoomed version close to $C_{\text{norm}} = 0$.
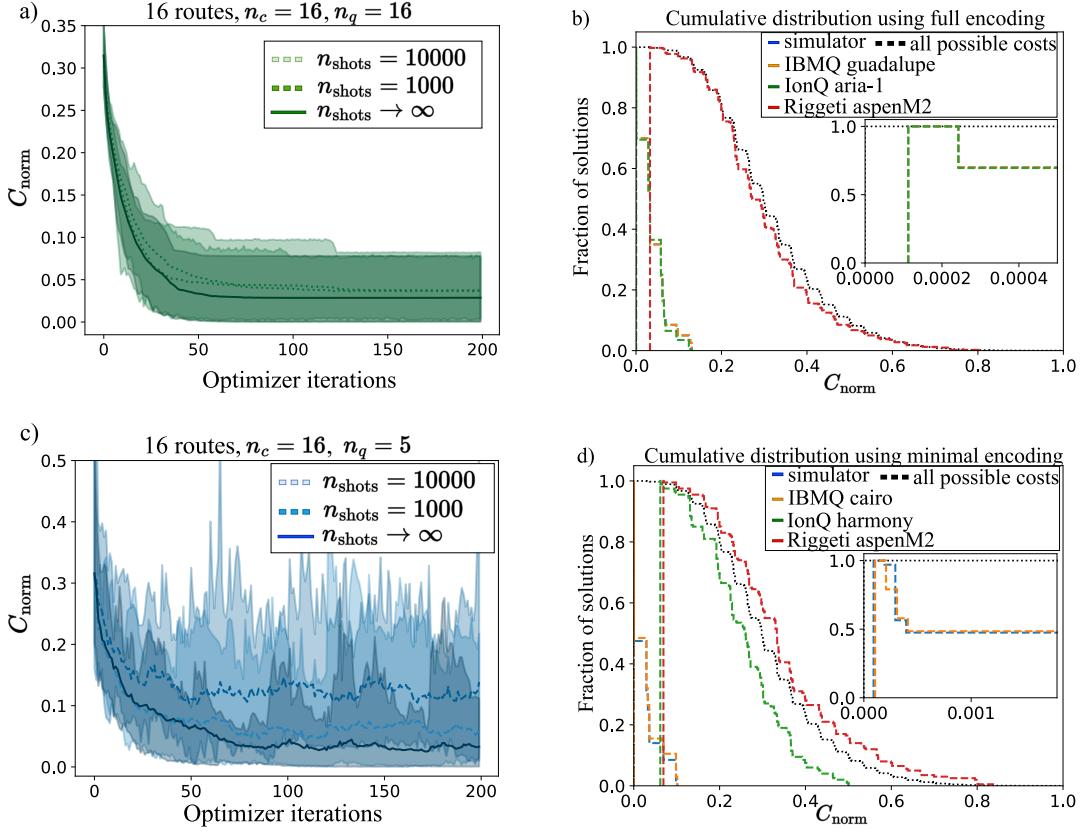
Figure 3.4: Comparison between the full encoding and minimal encoding for a $n_c = 16$ route problem, using $n_q = 16$ qubits for the full encoding and $n_q = 5$ qubits for the minimal encoding. a, c) Optimization runs over 20 starting points. Shaded region shows minimum and maximum value of the 20 runs obtained at each optimization iteration. b, d) Cumulative distribution of bitstrings obtained using $n_{shots} = 10000$ shots on a simulator (blue) , IBMQ (orange), IonQ (green), Riggeti (red) quantum backends compared with the distribution of all possible cost function values (black). Inset shows zoomed in versions of the cumulative distributions close to $C_{norm} = 0$.
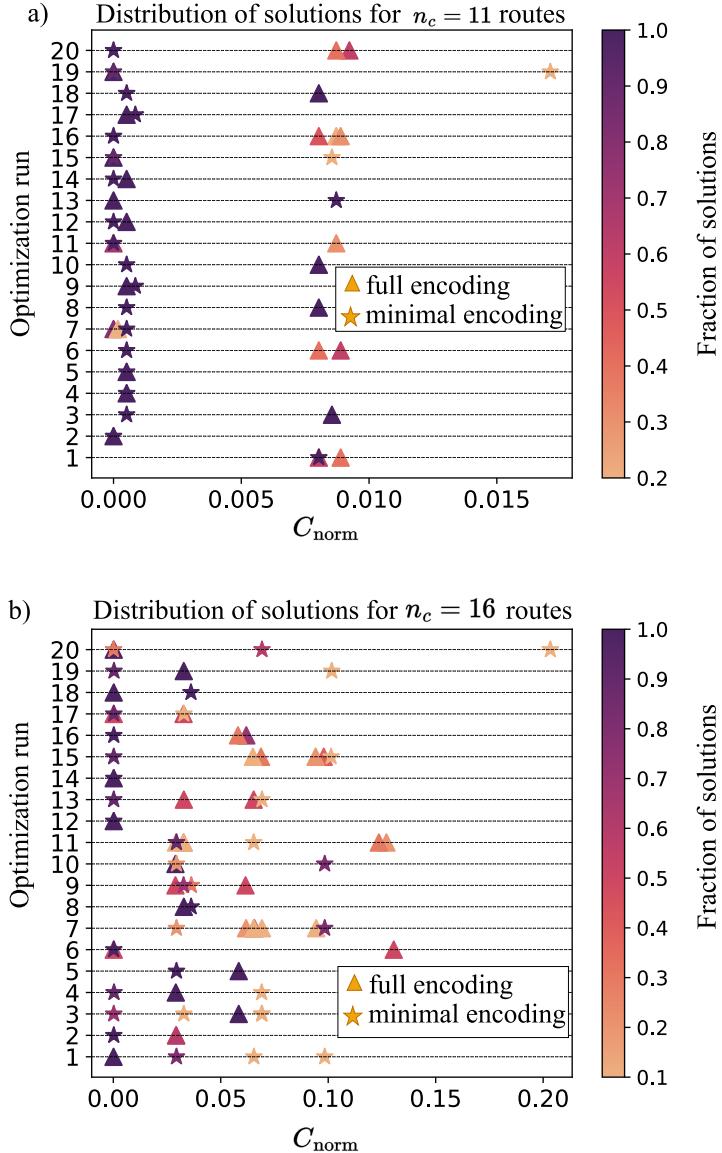
Figure 3.5: Distribution of solutions obtained using the minimal (stars) and full encoding (triangles) for a) $n_c = 11$ and b) $n_c = 16$ for 20 different optimization runs from the IBMQ backends. For each optimization run, the final quantum state is prepared on the IBMQ backends according to the optimized parameters, and 10 classical solutions are sampled for the full encoding and for the minimal encoding. In both the 11R and 16R instances, the minimal encoding is able to produce classical solutions of similar quality while maintaining a tighter spread in terms of cost function values.

Finding Solutions to VRPTW Problems using Minimal Encoding

To advance the capability of quantum devices in solving larger industry-relevant problems, we implemented the minimal encoding method to approximate solutions for VRPTW problems with $n_c = 128$ and $n_c = 3964$ classical variables. This required $n_q = \log(128) + 1 = 8$ qubits for the former and $n_q = \lceil \log(3964) \rceil + 1 = 13$ qubits for the latter, where $\lceil \cdot \rceil$ represents the ceiling function. The instance with 128 routes, shown in Fig. (3.6) (a-c), achieved reasonable solutions with a shot count similar to the smaller 11 and 16 route instances. However, for the 3964 route instance, as depicted in Fig. (3.6)d, the necessary resources for adequate solutions increased significantly, with the number of shots limiting solution quality.

The solutions from the IBMQ quantum backends for these larger instances, shown in Figures (3.6)b,e, align with those from noise-free simulations. Nonetheless, as indicated in the inset axes of these figures, neither of the larger problem instances reached the exact optimal solution.

Looking at the outcomes for smaller problems in Fig. (3.3) and Fig. (3.4), the solutions from the IBMQ backends were markedly better than those from the IonQ devices using minimal encoding.

For the 3964-route instance, the minimal encoding state has $2^{12} = 4096$ register states, but only 3964 are allocated to individual routes. Given only $n_{\text{shots}} = 1000$, many register states remain unmeasured, necessitating manual setting of $\frac{\langle \hat{P}_k^1 \rangle_{\vec{\theta}}}{\langle \hat{P}_k \rangle_{\vec{\theta}}} = 0.5$ for several terms in (3.9). Consequently, many register states went unmeasured, forcing assumptions about the inclusion of various routes in the solution, leading to overall lower quality solutions. Increasing the shot count to $n_{\text{shots}} = 10,000$ showed a reduction in the cost function value due to more register states being measured. Yet, with the quantum state comprising $2^{13} = 8192$ basis vectors in the computational basis, 10,000 shots proved insufficient for accurately characterizing all associated coefficients. It's presumed that larger instances like this would require more shots, a hypothesis not investigated in this
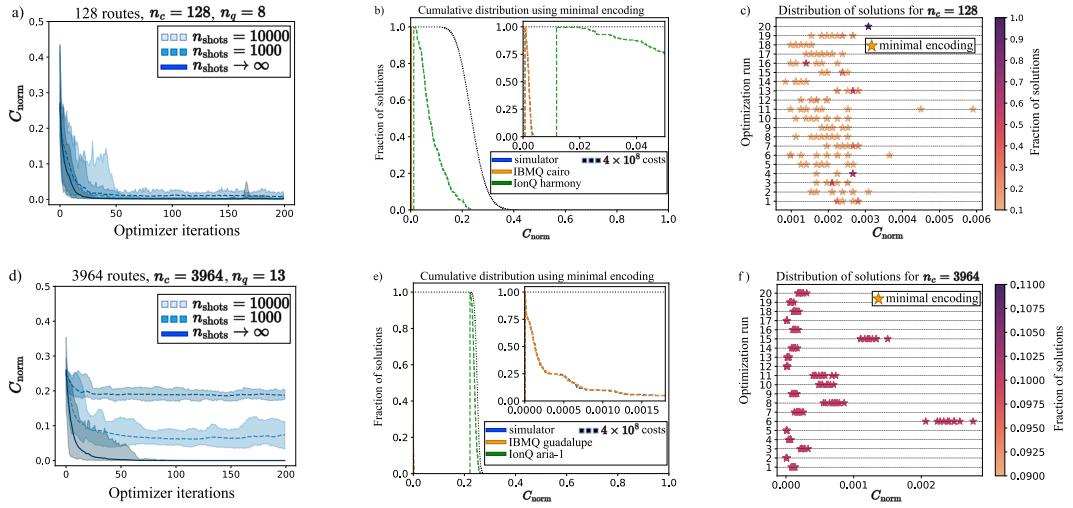
Figure 3.6: Optimization runs, cumulative distribution and distribution of solutions using minimal encoding for the $n_c = 128$ instance using $n_q = 8$ qubits and $n_c = 3964$ instance using $n_q = 13$ qubits. a,d) Optimization runs over 20 starting points. b,e) Cumulative distribution of solutions obtained using minimal encoding. Black dotted curves represents the cumulative distribution of 4×10^8 randomly generated solutions. Blue, orange, and green curves display the cumulative distribution of solutions obtained using minimal encoding with $n_{\text{meas}} = 10000$ shots on a noise free simulation and on IBMQ, IonQ quantum backends respectively. c,f) Distribution of solutions obtained from the minimal encoding state used to find approximate solutions to the $n_c = 128$R and $n_c = 3964$R problem instances. The final parameters from each of the 20 starting points are used to produce the final quantum state using the devices offered by IBMQ, and 10 classical solutions were sampled from each state according to (3.8).

study due to budgetary constraints.

In figure (3.6)c, for the 128R instance, we notice that the classical solutions derived from the optimized minimal encoding state are more dispersed, indicating less than optimal convergence. Despite this, the final state still yields bitstrings with low cost function values. For the 3964R instance, as shown in figure (3.6)f, there's a more concentrated clustering of solutions. However, several initial starting points did not converge satisfactorily.

3.4 Reinforcement Learning Based Enhancement of the Quality of Solutions

We present here a novel approach to enhance the quality of solutions generated by the qubit efficient algorithm presented earlier, exploiting RL to improve the quality of generated bitstrings. The latter is particularly important in large problems of industrial value. The applicability of this algorithm to large QUBO matrices stems from the inclusion of a GPU to handle time-consuming computations, namely large matrix multiplications, as shown in figure (3.7). [3]
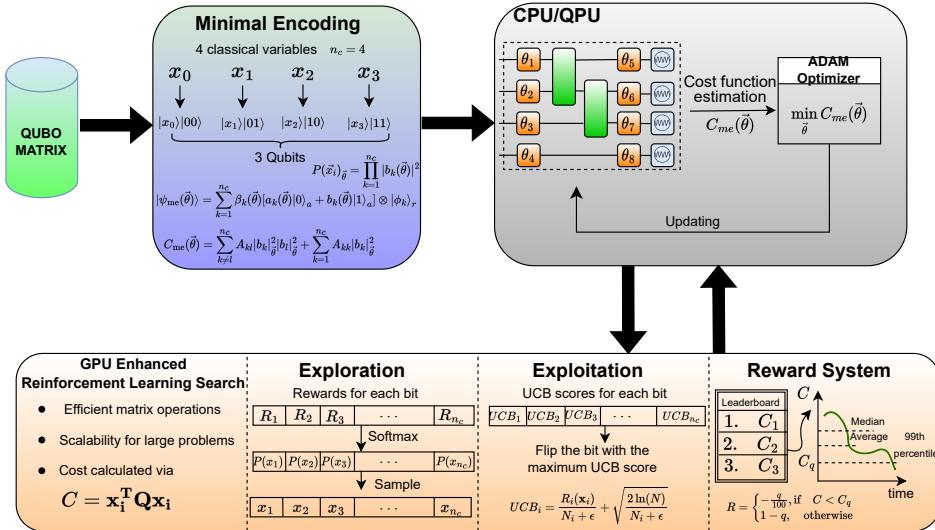
Figure 3.7: Pipeline of the reinforcement learning enhanced algorithm. Minimal encoding is used to transform large QUBO problems into a NISQ compatible format in order to use ADAM optimizer to obtain promising candidate solutions. The GPU enhanced reinforcement learning algorithm further enhances the quality of solution obtained by the qubit efficient algorithm.

---

[3]Unfortunately, at the time of writing this thesis, the classical simulation of the qubit efficient algorithm is done in a CPU rather than a GPU. This is because many GPU compatible quantum simulators require Linux distributions rather than Windows as they are still developing. Although, this issue can be addressed by sandboxing the algorithm's codebase into a Linux environment.

The first phase (top row in figure (3.7)) of the algorithm utilizes a VQA with a qubit efficient encoding proposed in [55]. This approach significantly reduces the number of required qubits, achieving a logarithmic reduction relative to the number of classical variables. The qubit efficient algorithm is optimized using an iterative process, typically employing the ADAM optimizer [121], to find high-quality initial solutions as described in section (3.3.1). These solutions serve as the starting point for the subsequent reinforcement learning algorithm.

Following the quantum optimization, the algorithm transitions into the second phase (bottom row in figure (3.7)), which involves the reinforcement learning algorithm. The RL search is divided into two segments, exploration and exploitation, each allocated some portion of the total RL algorithm time. Since initial solutions are given by an optimized quantum circuit, we choose to benefit exploitation more than exploration.

As this algorithm falls under the category of heuristic approximate algorithms, we have parameterize the two main phases by their execution time. This allows the algorithm to be more versatile in terms of applicability to various problem sizes since the user can choose the balance between performance and time-to-solution. In appendix section (7.1), we present an extrapolation on the resources that we need to solve QUBO problems of sizes relevant to industry using the qubit efficient algorithm on the IBM Cairo 27-qubit machine. There we have calculated the time it take for a single ADAM optimization step of the quantum circuit, based on information from the IBM's library qiskit [124], without accounting for the time to connect to the quantum computer through the API. Also, using the errors for the IBM Cairo quantum computer summarized in section (3.3), one could simulate how this algorithm work using this quantum machine by introducing a noise model to the classical simulator and adjusting the optimization time to be the same as optimizing in the actual device.

The reinforcement learning method is designed for enhancing the quality of solutions obtained from the qubit efficient algorithm. It intricately merges softmax-based exploration with Upper Confidence Bound (UCB) based exploitation, offering a robust solution-seeking mechanism. In this work we are using the terms exploration and exploitation to address the fact that in the exploration phase the agent is allowed to flip multiple bits of the bitstring solution per iteration, while in the exploitation phase the agent always change one bit per iteration.

3.4.1 Reward system

The rescaled rank reward system [65] used in the RL algorithm operates through a structured process designed to enhance solution quality over time. This reward system is designed to encourage the agent to flip bits that previously lead to worst solutions while also not flipping bits that have lead to good solutions. The rescaled rank reward for a single bit x_i from the bitstring solution is :

$$R_i(\mathbf{x}_i) = \begin{cases} -\frac{q}{100} & \text{if } \mathbf{x_i^T Q x_i} < C_q \\ 1 - \frac{q}{100} & \text{otherwise} \end{cases}$$

where C_q is the q-th percentile of previous solution qualities and $\mathbf{x_i}$ is a bitstring where the i-th bit has been flipped. The reward for finding a solution with lower cost than the q-th percentile is negative and the reward for worse solutions is positive. Although this is counter intuitive [4], our goal is to transform these rewards into probabilities that we will use to flip these bits, so we want bits that previously led to worse solutions to be flipped more often.

Initially, the system evaluates the quality of the current solution using equation (2.1). Then, the algorithm moves to calculate a rescaled rank reward for the current solution. This step is critical as it involves comparing the current solution's quality against a backdrop of historical solution qualities. Specifically, the algorithm uses the quality at

---

[4]Because all the optimization problems here are minimization problems.

the q-th percentile from these historical data as a benchmark. This percentile-based comparison helps in normalizing rewards across different scales of solution qualities, ensuring that the reward reflects how the current solution stands relative to past attempts rather than just its absolute quality. Lastly, the algorithm updates the rewards for the actions that led to the current solution, such as flipping a bit in a binary optimization problem. This updated reward is not arbitrary but is directly influenced by the rescaled rank reward calculated in the previous step allowing the agent to effectively guide the search process.

3.4.2 Softmax-Based Exploration

The softmax function with a given input vector of rewards $\mathbf{r}$ and a temperature parameter T, serves to convert the rescaled rank reward values into a probability distribution that will be used to sample a new bitstring. This process involves transforming each reward element $R_i(\mathbf{x}_i)$ into a probability p_i. The transformation formula is such that p_i is calculated by exponentiating the difference between $R_i(\mathbf{x}_i)$ and the maximum reward value in $\mathbf{r}$, divided by T, and then normalizing this value across all elements to ensure that the sum of probabilities p_i equals 1 as shown below :

$$p_i = \frac{e^{(R_i(\mathbf{x}_i) - \max(\mathbf{r}))/T}}{\sum_{j=1}^{n} e^{(R_i(\mathbf{x}_j) - \max(\mathbf{r}))/T}}$$

The role of $\max \mathbf{r}$, the maximum reward value, is pivotal in this calculation as it serves as a reference point for scaling the rewards, ensuring that the exponentiation does not lead to disproportionately large values for higher rewards. The temperature parameter T acts as a crucial control lever in this process. It moderates the distribution's entropy, where a higher T leads to a more evenly spread probability distribution, encouraging exploration of a broader range of solutions. Conversely, a lower T results in a distribution that is more concentrated around the highest rewards.

After calculating this probability distribution, the algorithm employs it to sample a bitstring for the subsequent iteration. It does so by iterating through the current bitstring and use the softmax probabilities to flip each bit. As a consequence, high rewards are transformed through the softmax function to high probabilities of flipping each bit. The use of the softmax function in this manner is instrumental in navigating the solution space efficiently, guiding the search process towards better solutions by dynamically adjusting the probability of flipping a bit.

3.4.3 UCB-Based Exploitation

Following the phase of softmax exploration, the algorithm progresses to a UCB (Upper Confidence Bound)-based method for enhanced exploitation. The UCB score for bit i, denoted as UCB_i, is calculated as:

$$UCB_i = \frac{R_i(\mathbf{x}_i)}{N_i + \epsilon} + \sqrt{\frac{2\ln(N)}{N_i + \epsilon}}$$

In this calculation, the term $\frac{R_i(\mathbf{x}_i)}{N_i+\epsilon}$ determines the mean reward per flip for bit i, assessing its average performance. Concurrently, $\sqrt{\frac{2\ln(N)}{N_i+\epsilon}}$ measures the uncertainty or the latent potential for enhancement by flipping bit i. This component incentivizes the exploration of bits that have been less engaged, while still acknowledging and leveraging the historical rewards, thereby striking a balance between the exploitation of well-performing bits and the exploration of underutilized ones. This nuanced approach ensures the algorithm dynamically explores and exploits the solution space, optimizing for better outcomes through informed decision-making.

At each iteration this algorithm will flip the bit with the highest UCB score in order to obtain a new bitstring solution. This means that the algorithm will change one bit per iteration and try to construct the optimal solution this way, resulting in more iterations as the problem size grows.

3.4.4 Comparison of Minimal Encoding with and without RL Enhancement

Below we compare the solution quality obtained using the minimal encoding scheme with and without the reinforcement learning enhancement algorithm. We extract the minimal encoding solutions as described in section (3.3) using a classical simulator to perform 20 optimization runs per problem instance, with different starting angles of the variational quantum circuit. Then we input the best angles found and sample 10 solutions per problem instance using the IBM Cairo and IBM Guadalupe quantum devices for the 128R and 3964R VRPTW instances respectively. For each optimization run, we feed the best solution found from the qubit efficient algorithm from the two quantum computers to the reinforcement learning algorithm to enhance its quality.
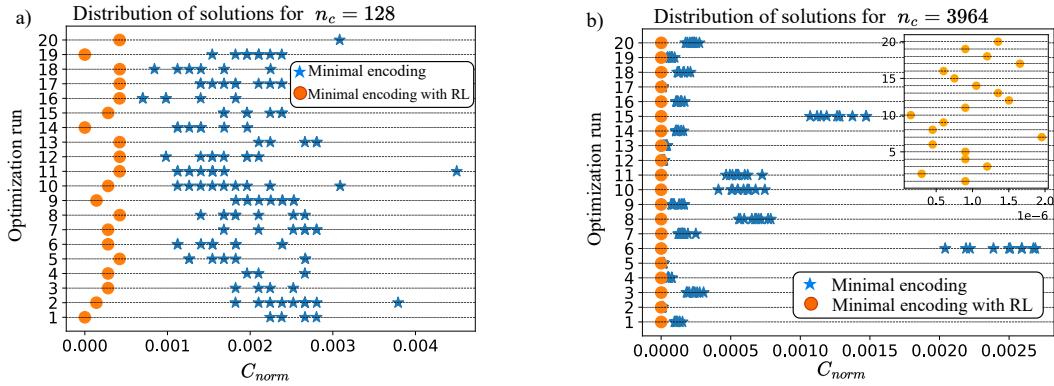
Figure 3.8: Comparison of the normalized cost function obtained using minimal encoding with and without the reinforcement learning enhancement algorithm. 20 optimization runs were performed on classical simulators and the best angles found were used on quantum devices in order to sample 10 solutions per optimization run (blue stars). The reinforcement learning algorithm is then used to enhance the quality of the best solution from each optimization run (orange circles). **a)** Results for the 128R VRPTW instance, **b)** Results for the 3964R VRPTW instance. In both figures we observe that the reinforcement learning algorithm enhances the quality of solutions and also that different starting solutions from the minimal encoding scheme leads to different improved solutions. The inset axis in the 3964R figure is used to show the difference in scale of using the reinforcement learning enhancement algorithm.

In figure (3.8) we observe that better starting solutions for the reinforcement learning enhancer don't necessarily yield better improvements. For example in figure (3.8)a, optimization run 16 has a better starting solution compared to optimization run 15, but the enhanced solution in optimization run 15 is better than the one in optimization run 16. The reinforcement learning algorithm is designed such that, it won't flip bits of the bitstring solution that have lead to a better solution quality. With this in mind, different starting solutions will lead to different paths of flipping bits and therefore different enhanced solutions obtained. Looking at the inset axis of figure (3.8)b, we found that enhanced solutions for the 3964R problem instance fall inside the $1e-6$ scale of the normalized cost function value while minimal encoding solution are in the order of $1e-4$.

Overall, we found that random sampling results in a normalized cost function value of approximately 0.3 (see dashed black line in figures (3.6) b & e). The qubit efficient algorithm is able to produce better than random solutions and closer to the optimal solution up to a relative gap. The RL enhancement is able to further shorten the gap between the proposed solution and the optimal one.

3.4.5 Solving Large Scale VRTPW Problems using the RL Enhanced Optimizer

Here we present results of the reinforcement learning enhanced optimizer for all problem sizes shown in table (3.1). We slightly change the metric from previous figures, where now we plot the approximation ratio defined by

$$\alpha = \frac{C_{max} - C_x}{C_{max} - C_{min}} = 1 - C_{\text{norm}} \tag{3.13}$$

and it's highest value is equal to 1 when $C_x = C_{min}$. In order to find C_{min} for the approximation ratio we have used the student version of the Gurobi commercial solver

Classical variables	Qubits using minimal encoding	Time for classical simulator of the qubit efficient algorithm (s)	Time for RL enhancement (s)	Avg approximation ratio (%)
687	11	60	1	99.99
1285	12	80	1.2	99.99
3964	13	120	6	99.99
6081	14	200	11	99.99
9644	15	360	25	99.99
14095	15	600	60	99.99
20565	16	1000	300	99.99
25382	16	2000	500	99.99

Table 3.3: Details on the usage of the reinforcement learning enhanced optimizer. The number of classical variables of each VRPTW instance is the same as the number of routes. The time displayed here is in seconds and it is divided to classical simulator time (CPU) and reinforcement learning enhancement time (GPU). The overall time to solution is calculated by adding these two times and the translation of time to iterations can be seen in figure (3.10).

with a time limit of 10 hours for all problem instances.

Table (3.3) shows the runtime of the overall reinforcement learning enhanced algorithm to achieve an approximation ratio $\alpha \geq 0.9999$. The time for quantum simulation also includes QUBO matrix handling and processing, like estimating the cost function (3.10) and estimating the probability distribution from (3.8). The maximum number of qubits used for these instances is 16, which results in a quantum state of $2^{16} = 65536$ basis states. In order for the qubit efficient algorithm to estimate the cost function value we have used 10000 shots. As mentioned in [55], this number of shots limits the accuracy of the minimal encoding scheme in estimating the cost function in equation (3.5) and one should expect even more errors when moving this encoding scheme to real noisy quantum hardware.

In figure (3.9) we show collective results for 20 optimization runs per problem instance of the VRPTW. For each problem size we plot the average approximation ratio found as

well as its minimum and maximum values. We obtain near optimal solution quality for every problem instance while the deviation from the mean is in the order of the solution quality indicating that the reinforcement learning enhanced algorithm converged in all optimization runs. More, we see that as the problem size grows the approximation ratio becomes better which is counterintuitive. That is because different iterations were used during the optimization of VRPTW instances with different size in order to keep the approximation ratio consistent.
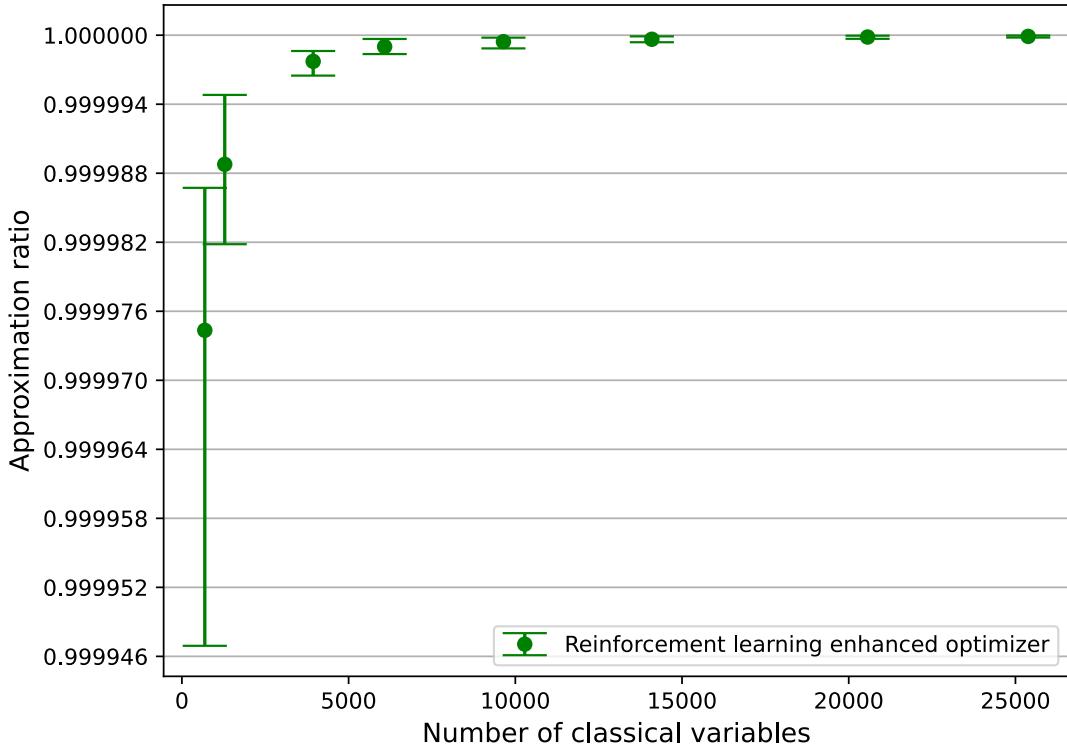
Figure 3.9: Performance of the reinforcement learning enhanced optimizer for various problem sizes of the VRPTW. For each problem instance 20 optimization runs were performed. Error bars denote the minimum and maximum approximation ratios obtained during the optimization runs, while the circle in each line shows the median approximation ratio obtained.

Figure (3.10) shows the progress of the approximation ratio during an optimization run for each problem instance considered here. The inset axis shows the optimization

done using the minimal encoding scheme, where we observe that minimal encoding alone produced approximation ratios bigger than 0.9 for smaller problem instances (1285 & 3964 routes). Optimization using the minimal encoding becomes more difficult as the problem size grows, with the optimization of the biggest 25382 routes problem getting stuck at approximation ratio equal to 0.8. As the problem size grows, the reinforcement learning algorithm requires more iterations in order to converge to a solution with $\alpha \geq$ 0.9999. We also plot the average approximation ratio of random sampling among all VRPTW instances considered, where we sample 4×10^6 random solutions for VRPTW instances with less than 10000 classical variables and 4×10^8 random solutions for larger VRPTW instances, and then take the average.

The trend shown in this figure shows a sub-linear complexity of the number of iterations with the problem size in order for the algorithm to converge. Although it is evident that the search of the reinforcement learning algorithm depends on the scale of the problem and also in the cost function landscape. Because the algorithm is more focused on changing one bit per iteration in order to find better solutions, it progresses by doing small adjustments in the current solution and obtains a better solution by a small fraction.
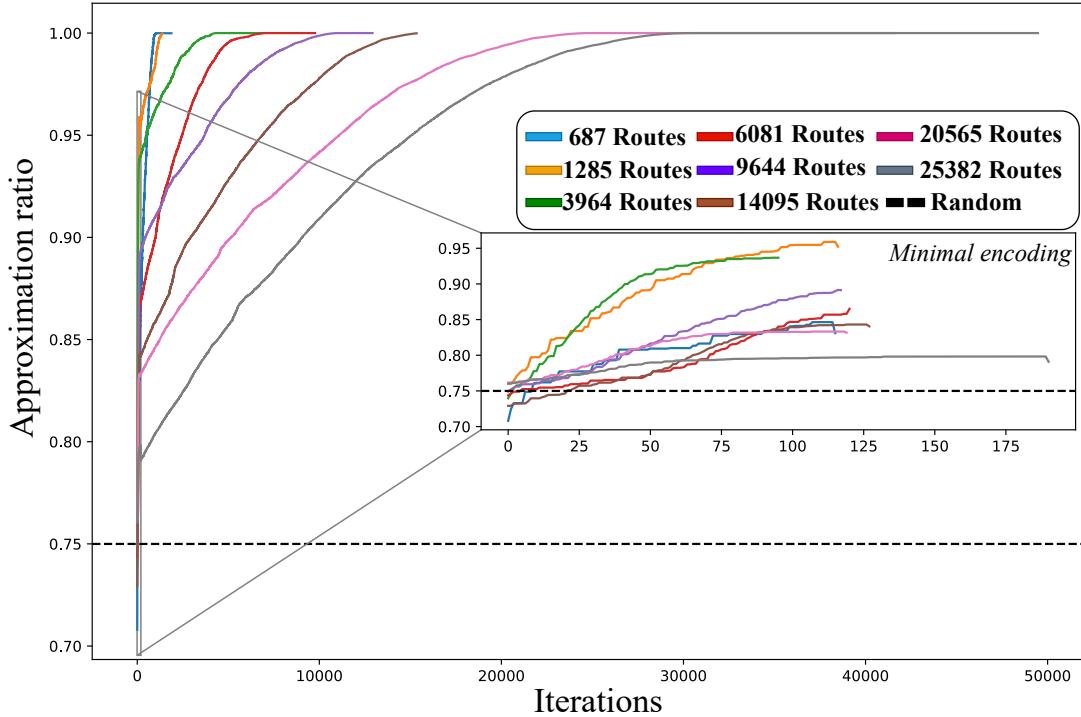
Figure 3.10: Progress of the approximation ratio for every VRPTW instance during optimization of the reinforcement learning enhanced optimizer. Optimization runs for different VRPTW problem sizes are showed with different color. Inset axis shows the optimization part of the algorithm that uses minimal encoding while the main plot captures iterations from the reinforcement learning part of the main algorithm. The black dotted line shows the average approximation ratio obtained by random sampling amongst all VRPTW instances.

4 Qubit Efficient Optimization for the Tail Assignment Problem

4.1 Introduction

In this chapter, we delve into the intricate world of the Tail Assignment Problem (TAP), an essential optimization challenge in the airline industry. The formulation of TAP as a QUBO problem has been expertly proposed by Vikstål et al. [1]. This innovative approach allows for the application of the qubit efficient encoding technique, as detailed by Tan et al. [55]. Using this method, we demonstrate the ability to tackle TAP instances involving up to 26,000 classical variables, utilizing as few as 16 qubits and a single laptop GPU. We further explore the creation of synthetic instances, demonstrating the process of translating these instances into a QUBO framework. The chapter also discusses the performance of a reinforcement learning enhanced optimizer. We focus on the average approximation ratio across various problem sizes and offer insights into the specific contributions of the qubit efficient encoding and the reinforcement learning component of the algorithm.

4.2 Tail Assignment Problem

Airlines face a range of complex, large-scale planning challenges every day that involve coordinating various resources including passengers, crew, aircraft, maintenance, and ground staff. The standard planning procedure for airlines is carried out in stages, beginning with the development of a flight schedule, followed by various steps for planning aircraft and crew assignments. These tasks all involve large-scale optimization with distinct goals, yet they share the common objectives of maximizing profits, ensuring safety, enhancing crew satisfaction, and reducing the risk of operational disruptions. Additionally, airlines must comply with a myriad of intricate regulatory, operational, and quality standards [125].

One specific aspect of fleet planning is the tail assignment problem, which entails deciding which particular aircraft, identified by its tail number, will fly each scheduled flight. A route is defined as a series of flights consecutively performed by the same aircraft. To be viable, a route must meet various criteria. For instance, there must be sufficient buffer time between an aircraft's arrival and the subsequent departure of its next flight—this is known as the minimum turn time. This turn time varies based on factors such as the flight's nature (domestic or international), the airport, the time of day, and even the particular characteristics of the aircraft. Additionally, certain aircraft may be subject to destination restrictions that prevent them from flying to particular airports, often due to limitations such as insufficient engine power for shorter runways. There are also curfew restrictions, which typically prevent loud aircraft from flying during night hours, particularly at airports located near city centers. Lastly, routes must comply with maintenance needs, both immediate and long-term, ensuring aircraft are scheduled for maintenance stops at equipped airports with enough time allocated for necessary work to be carried out.

The formulation of the Tail Assignment Problem (TAP) comes from [1], where the

authors have successfully applied the QAOA to solve TAP instances of 8,15 and 25 routes using 8, 15 and 25 qubits respectively. In particular they reduced the TAP to a feasibility version with a unique solution and showed that as the layers of the QAOA circuit increases, QAOA is able to find the unique solution with close to unit probability. Here we use the same feasibility formulation with a unique solution and apply the minimal encoding scheme with the reinforcement learning enhanced algorithm to tackle problem sizes of up to 26000 classical variables using 16 qubits and a single laptop GPU.

Let us denote by $\mathcal{F}$ the collection of all flights, by $\mathcal{T}$ the collection of tails (aircraft identifiers), and by $\mathcal{R}$ the ensemble of permissible routes. Let the cost associated with a route r from the set $\mathcal{R}$ be represented by c_r, and let C_f stand for the cost incurred from not assigning flight f. The cost of a route might reflect its resilience to operational disruptions, the expenditure on fuel, or an amalgam of multiple factors [1]. Define the binary indicator a_{fr} as 1 if flight f is included in route r, and 0 otherwise. Similarly, let b_{tr} be 1 if tail t is deployed for route r, and 0 otherwise. The decision variable x_r takes the value 1 if route r is selected for use in the plan, and 0 if not. The variables u_f and v_t are assigned 1 if flight f is omitted or tail t remains idle, respectively, and 0 otherwise.

The tail assignment problem can thus be expressed as the following optimization problem [1]:

$$\min_{r \in \mathcal{R}} \quad \sum_{r \in \mathcal{R}} c_r x_r + \sum_{f \in \mathcal{F}} C_f u_f \tag{4.1a}$$

$$\text{subject to} \quad \sum_{r \in \mathcal{R}} a_{fr} x_r + u_f = 1, \quad \forall f \in \mathcal{F}, \tag{4.1b}$$

$$\sum_{r \in \mathcal{R}} b_{tr} x_r + v_t = 1, \quad \forall t \in \mathcal{T}, \tag{4.1c}$$

$$x_r, u_f, v_t \in \{0, 1\}. \tag{4.1d}$$

The aim specified in (4.1a) is to minimize the aggregate expenses of the employed routes, with the requirement (4.1b) guaranteeing that every flight is allocated to one and only one route, and condition (4.1c) ascertaining that each tail is allocated no more than once. Leaving a flight unassigned carries a cost denoted by C_f, which is generally significant relative to the costs of assigning routes. There are no penalties for tails that are not in use.

Our focus is shifted to the decision variant of the tail-assignment task [1], which aims to identify any solution that fulfills the necessary conditions without considering the costs c_r. This allows us to exclude the u_f variables from our model. Additionally, we assign a unique initial flight to each aircraft, thus eliminating the need for constraints in Eq. (4.1c). This variant, known as the exact-cover problem, is acknowledged as NP-complete and can be depicted as the subsequent optimization problem:

$$\text{minimize } 0 \tag{4.2a}$$

$$\text{subject to} \quad \sum_{r \in \mathcal{R}} a_{fr} x_r = 1, \quad \forall f \in \mathcal{F}, \tag{4.2b}$$

$$x_r \in \{0, 1\}, \tag{4.2c}$$

where the minimization over 0 serves as a placeholder to indicate that this framework originates from the tail-assignment problem, wherein we disregard the costs in Eq. (4.1a). Despite these reductions, the exact-cover problem remains highly pertinent for the analysis of tail-assignment since numerous airlines, such as Air France, view the tail-assignment issue predominantly as a problem of feasibility [126].

To derive the QUBO matrix for the feasibility version of the TAP let's consider the cost function for the tail-assignment problem first. We form the cost function by converting the unique constraint in Eq. (4.2b) to a penalty term that is added to the cost function. Using the cost function $C(\mathbf{x})$ below, we observe that, when the cost is equal to 0, the

4 Qubit Efficient Optimization for the Tail Assignment Problem

constraint in Eq. (4.2b) is satisfied.

$$C(\mathbf{x}) = \sum_{f=1}^{|F|} \left(\sum_{r=1}^{|R|} a_{fr} x_r - 1 \right)^2 .$$

Expanding the square, we get:

$$C(\mathbf{x}) = \sum_{f=1}^{|F|} \left(\left(\sum_{r=1}^{|R|} \alpha_{fr} x_r \right)^2 - 2 \sum_{r=1}^{|R|} \alpha_{fr} x_r + 1 \right) .$$

Since $x_r^2 = x_r$ and $\alpha_{fr}^2 = \alpha_{fr}$ for binary values,

$$C(\mathbf{x}) = \sum_{f=1}^{|F|} \left(\sum_{r=1}^{|R|} \alpha_{fr} x_r + \sum_{r=1}^{|R|} \sum_{r' \neq r}^{|R|} 2\alpha_{fr}\alpha_{fr'} x_r x_{r'} - \sum_{r=1}^{|R|} 2\alpha_{fr} x_r + 1 \right) =$$

$$\sum_{r=1}^{|R|} \sum_{r' \neq r}^{|R|} \sum_{f=1}^{|F|} 2\alpha_{fr}\alpha_{fr'} x_r x_{r'} - \sum_{r=1}^{|R|} \sum_{f=1}^{|F|} \alpha_{fr} x_r + |F|$$

The complete QUBO matrix is then:

$$Q = \begin{bmatrix} Q_{11} & Q_{12} & \cdots & Q_{1|R|} \\ Q_{21} & Q_{22} & \cdots & Q_{2|R|} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{|R|1} & Q_{|R|2} & \cdots & Q_{|R||R|} \end{bmatrix}$$

Where the diagonal terms Q_{rr} and the off-diagonal terms $Q_{rr'}$ are given by:

$$Q_{rr} = -\sum_{f=1}^{|F|} \alpha_{fr},$$

$$Q_{rr'} = 2\sum_{f=1}^{|F|} a_{fr}a_{fr'}, \quad \text{for } r \neq r'.$$

4.3 Instance Generation

In the TAP, we consider a predefined number of routes, n_r, and flights, n_f. Each route and flight is uniquely identified, with routes potentially consisting of different sets of flights. The synthetic data generation for TAP involves dividing the total number of routes into sets, each containing an adjustable fraction of the total flights representing the variability that would occur in a real world scenario.

The aim is to create a scenario where there exists a unique optimal solution for the problem. For this reason we first generate the solution routes by filling a predefined number of routes until all flights are served. This unique solution is represented by a binary string with a specific number of ones. The remaining routes are filled with randomly selected flights. It's important to note that randomly filling the rest of the routes can result in near optimal solution with high overlap with the optimal one, thus making the optimization procedure hard.

4.4 Using the RL Enhanced Optimizer to Tackle Large TAP

The reduced TAP considered here is purely an assignment problem and can be seen as an Exact Cover problem [1]. This means that the optimal solution will include routes that have no overlapping flights and each overlapping flight will raise the value of the cost function as seen in Eq. (4.2). This results in a cost function landscape with many local minima.

After creating the QUBO matrix Q discussed in section (4.2), we use the parametrized quantum state from eq. (3.5) represented by a hardware efficient ansatz (2.2) with 4 layers and 10000 measurement shots. This allows us to estimate the cost function using eq. (3.10) by substituting matrix A with the current QUBO matrix Q and thus take advantage of the qubit efficient encoding scheme from [55] in order to fit big TAP instances into quantum circuits with a small number of qubits.

Routes	Flights	Classical variables	Qubits using minimal encoding	Time for classical simulator of the qubit efficient algorithm (s)	Time for RL enhancement (s)	Avg. approximation ratio (%)
1000	2000	1000	11	60	20	99.97
6000	12000	6000	14	200	40	99.91
11000	22000	11000	15	500	120	99.7
16000	32000	16000	15	600	300	99.7
21000	42000	21000	16	2000	600	99.5
26000	52000	26000	16	2500	900	99.05

Table 4.1: Detailed information for all the TAP instances considered. The number of classical variables is the same as the number of routes and the number of qubits using minimal encoding follows a logarithmic reduction when compared with the number of classical variables, as mentioned in section (3.2.2). The quantum simulator time refer to the CPU time it took for the optimization part with minimal encoding to produce good initial solutions and the reinforcement learning enhancement time refers to the GPU time it took the algorithm to produce near optimal solutions. The average approximation ratio per problem instance is calculated using data from 20 optimization runs per problem instance.

The resources needed to obtain average approximation ratios $\alpha \geq 0.99$ are shown in table (4.1). We must note that the time for the classical simulator using a CPU also includes matrix operations for equations (3.8) and (3.10) of the minimal encoding part. This was also the most time consuming part of the minimal encoding algorithm, since running the classical simulator for less than 20 qubits is very fast in state of the art simulators. We used the hardware efficient ansatz (2.2) with 10000 shots for all problem instances. For problem instances requiring more than 14 qubits 10000 shots are not enough to accurately estimate the cost function and as we discuss in the previous section some bits of the sampled solution will be assigned randomly. Our goal is to use the RL enhanced optimizer to overcome this problem.
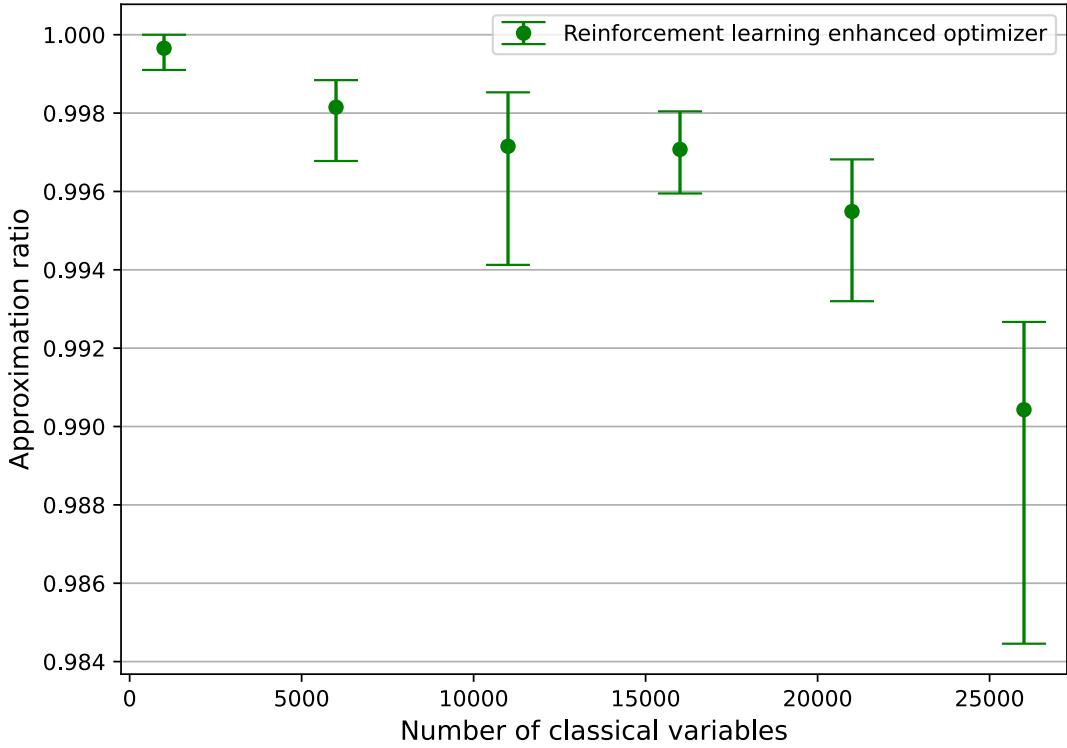
Figure 4.1: Performance of the reinforcement learning enhanced optimizer for various problem sizes of the TAP. For each problem instance 20 optimization runs were performed. Error bars denote the minimum and maximum approximation ratios obtained during the optimization runs, while the circle in each line shows the median approximation ratio obtained.

To showcase the effectiveness of the RL enhanced algorithm we perform 20 optimization runs per problem instance and plot the average approximation ratio as well as the minimum and maximum values obtained through these optimization runs in figure (4.1). For each optimization run, we first use the qubit efficient algorithm to find some initial solutions, we sort them by their cost function values, and then we use the RL enhanced algorithm to refine the solution with the minimum cost function value. We observe that if we give the algorithm enough computation time we can obtain an average approximation ratio $a \geq 0.99$ for all TAP instances.

For the smaller TAP instance considered in this work, namely the one with 1000 classical variables the maximum approximation ratio obtained from the 20 optimization runs is equal to one, indicating that the RL enhanced optimizer found the exact same solution as the Gurobi commercial solver. As the problem size grows the deviation of the mean approximation ratio from the extreme values also raises. This is due to the fact that this formulation of the TAP results in a single optimal solution. Other near optimal solutions have high overlap with the optimal solution due to eq. (4.2) of the cost function making the optimization procedure harder due to the resulting landscape.

Figure (4.2) below shows the optimization process during a single optimization run for all TAP instances. The inset axis shows the performance of the minimal encoding scheme in finding good initial solutions and compares them with the approximation ratio obtained from random sampling which is approximate equal to 0.72. To calculate the average random approximation ratio we are using 4×10^6 random solutions for problem sizes with less than 10000 classical variables and 4×10^8 random solutions for problem sizes with more than 10000 classical variables.

We observe that minimal encoding alone can produce better than random solutions for all the TAP instances, but as the problem size grows the optimization progress becomes flat indicating that the qubit efficient algorithm got stuck at local minima. The progress of the RL enhancement algorithm shown in the main plot, starts off from the solutions found by the qubit efficient algorithm, and then refines these solutions in order to achieve a better cost function value. The whole optimization process is smooth and does not have many fluctuations, indicating the effectiveness of the UCB exploitation part of the algorithm to obtain better solutions changing one bit of the whole solution per iteration.
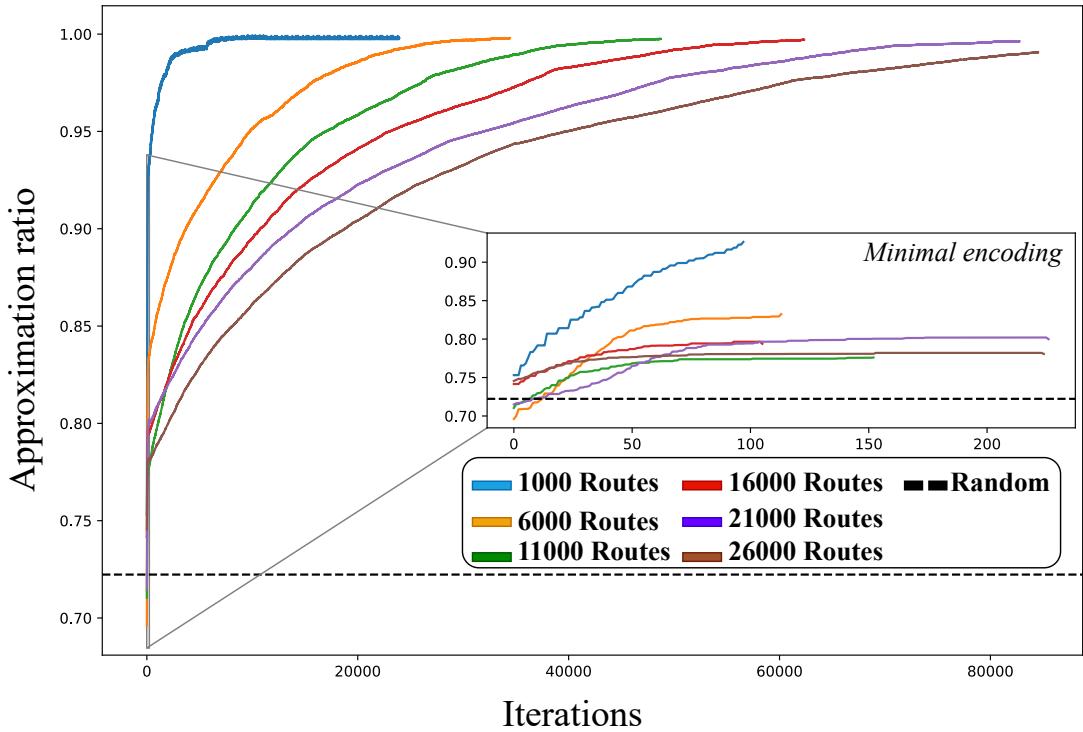
Figure 4.2: Optimization procedure for each TAP instance. The inset axis shows the progression of the minimal encoding algorithm, the black dashed line denotes the average random approximation ratio for the TAP instances. The main axis focuses on the progression of the approximation ratio during the RL enhancement of the minimal encoding solution. Each color denotes separate problem instances.

5 Qubit Efficient Optimization for the Flight Gate Assignment Problem

5.1 Introduction

Following the exploration of the Tail Assignment Problem, this chapter shifts focus to another critical optimization challenge in the aviation sector: the Flight Gate Assignment Problem (FGA). Stollenwerk et al. [66] have adeptly formulated the FGA as a QUBO, paving the way for innovative problem-solving strategies. This formulation permits the use of the qubit efficient encoding technique, as outlined by Tan et al. [55], enabling us to address FGA instances with up to 21,000 classical variables using a mere 16 qubits and a single laptop GPU. This chapter offers an insightful look into the generation of synthetic FGA instances, aligning closely with the mathematical structure of the problem and detailing the conversion process to a QUBO format. A critical part of this exploration is the assessment of a reinforcement learning enhanced optimizer. We evaluate its effectiveness by measuring the average approximation ratio across different scales of the problem.

5.2 Flight Gate Assignment Problem

The flight gate assignment (FGA) problem is a well-studied problem in the field of airport operations and optimization [66], [127]. It involves assigning a set of flights to available gates, taking into consideration factors such as transfer passengers, flight times, and buffer times between flights. The goal is to find an assignment of flights to gates in the airport such that the total transit time of all passengers in the airport is minimized.

In [66] the authors used a quantum annealing machine to solve the FGA problem for a medium size airport company, although due to the increased connectivity of the problem, the embedding to the quantum annealing machine is restricted. This allowed the authors to solve instances of up to 84 classical variables. Authors of [127] used a VQE variant to solve the FGA problem. Their biggest problem size consists of 18 classical variables using 18 qubits. In this work we consider the FGA problem sizes shown in table (5.2), with the biggest FGA instance consisting of 20961 classical variables using 16 qubits for the minimal encoding.

The goal of the FGA is to determine the best way to allocate flights to gates at an airport. While there are various ways to measure the success of such an allocation, our primary concern is to reduce the overall time passengers spend transiting within the airport. We categorize passengers into three groups: those arriving on a flight and exiting the airport from their arrival gate, those entering the airport through security and walking to their departure gate, and those in transit who arrive at one gate and need to walk to another for their connecting flight.

The mathematical formulation of the FGA problem is essential for understanding the problem's structure and constraints. By representing the problem mathematically, we can apply optimization techniques to find the best solution. The formulation provided below is a mixed-integer linear programming (MILP) model[1], which is a powerful tool

---

[1]We kindly refer the reader to [128], where the authors combine MILP and QUBO formulations for solving the Job Shop Scheduling problem in a quantum annealing machine.

for solving combinatorial optimization problems. However, solving MILP problems can be computationally expensive, especially for large instances.

Symbol	Description
F	Set of flights.
G	Set of gates.
$n_{\mathrm{dep}}(f)$	Number of departing passengers for flight f.
$n_{\mathrm{arr}}(f)$	Number of arriving passengers for flight f.
$n(f_1, f_2)$	Number of transfer passengers between flights f_1 and f_2.
$t_{\mathrm{arr}}(g)$	Arrival time at gate g.
$t_{\mathrm{dep}}(g)$	Departure time from gate g.
$t(g_1, g_2)$	Transfer time between gates g_1 and g_2.
$t_{\mathrm{in}}(f)$, $t_{\mathrm{out}}(f)$	Arrival and departure times of flight f, respectively.
t_{buf}	Buffer time between two flights at the same gate.

Table 5.1: Symbols and their descriptions for the flight gate assignment problem.

To formulate the FGA problem, we follow the formulation outlined in [66]. We begin by defining our decision variables, constraints, and objective function. The decision variables represent the choices we need to make, the constraints ensure that our choices are valid, and the objective function quantifies the quality of our choices.

The core decision in the FGA problem is the assignment of flights to gates. To represent this, we introduce a binary decision variable $x(f, g)$ for each flight f and gate g. The value of $x(f, g)$ will be 1 if flight f is assigned to gate g and 0 otherwise.

$$x(f, g) = \begin{cases} 1 & \text{if flight } f \text{ is assigned to gate } g \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

The primary goal is to minimize the total transit time of passengers. This includes the time taken by arriving passengers to exit the airport, the time taken by departing passengers to board their flights, and the transfer time for transit passengers moving between gates. The objective function aggregates these times, weighted by the number of passengers affected.

In this formulation two constraints are present :

1. **Assignment Constraint:** Every flight must be assigned to exactly one gate. This ensures that all flights are catered for in the solution.

2. **Occupancy Constraint:** Two flights that overlap in time cannot be assigned to the same gate. This ensures that there are no scheduling conflicts at any gate.

With these components, we can now express the FGA problem as an MILP:

Minimize:

$$\text{Cost} = \sum_{f \in F} \sum_{g \in G} n_{\text{arr}}(f) \cdot t_{\text{arr}}(g) \cdot x(f,g) + \sum_{f \in F} \sum_{g \in G} n_{\text{dep}}(f) \cdot t_{\text{dep}}(g) \cdot x(f,g) \qquad (5.2)$$

$$+ \sum_{f_1 \in F} \sum_{f_2 \in F} \sum_{g_1 \in G} \sum_{g_2 \in G} n(f_1, f_2) \cdot t(g_1, g_2) \cdot x(f_1, g_1) \cdot x(f_2, g_2) \qquad (5.3)$$

Subject to:

$$\text{Assignment Constraint:} \quad \sum_{g \in G} x(f,g) = 1 \quad \forall f \in F \qquad (5.4)$$

$$\text{Occupancy Constraint:} \quad x(f_1, g) + x(f_2, g) \leq 1 \quad \forall f_1, f_2 \in F, g \in G \qquad (5.5)$$

$$\text{if } f_1 \text{ and } f_2 \text{ overlap}$$

This MILP formulation provides a comprehensive representation of the FGA problem, capturing the complexities and nuances of flight gate assignments in a structured manner suitable for optimization.

Converting our MILP model to a QUBO formulation allows us to leverage quantum computing to potentially find solutions more efficiently. The process of conversion involves representing the objective function and constraints as a quadratic function of binary variables.

The cost function provided in the MILP model is already quadratic due to the terms $x(f_1, g_1) \cdot x(f_2, g_2)$. This can be directly incorporated into the QUBO matrix Q. The QUBO model can be represented as shown in eq. (2.1) where x is a binary vector representing our decision variables, and Q is a matrix encoding the objective function and constraints.

To ensure that our constraints from the MILP model are satisfied in the QUBO formulation, we introduce penalty terms to the objective function.

1. **Assignment Constraint:** The constraint ensures that each flight is assigned to exactly one gate. In the QUBO formulation, we can penalize violations of this constraint by adding a large penalty term P to the objective function for each violation:

$$P \sum_{f \in F} \left(1 - \sum_{g \in G} x(f, g) \right)^2 \tag{5.6}$$

2. **Occupancy Constraint:** This constraint ensures that two flights that overlap in time do not get assigned to the same gate. We penalize these violations by adding a penalty term to the objective function:

$$P \sum_{f_1, f_2 \in F, g \in G} x(f_1, g) \cdot x(f_2, g) \tag{5.7}$$

Where f_1 and f_2 overlap.

Combining the objective function and the penalty terms for the constraints, we obtain the final QUBO formulation:

$$C(x) = x^T Q x + P \sum_{f \in F} \left(1 - \sum_{g \in G} x(f, g) \right)^2 + P \sum_{f_1, f_2 \in F, g \in G} x(f_1, g) \cdot x(f_2, g) \tag{5.8}$$

Choosing an appropriate value for the penalty term P is the bottleneck of using such an approach to formulating QUBO problems. The penalty term alone should separate

clearly solutions that don't satisfy the constraints from those solutions that satisfy the constraints. For this reason, P is typically chosen to be larger than the maximum value of the cost function C_{max}. The matrix Q can be constructed by identifying the coefficients of the quadratic terms in the objective function.

Constructing the Q matrix for the FGA problem involves capturing the coefficients of the quadratic and linear terms in the objective function. In practice, constructing the Q matrix involves iterating over all pairs of binary variables, determining the coefficient of their product in the objective function, and setting the corresponding entry in the Q matrix.

1. **Objective Function:** From the MILP model, the quadratic terms in the objective function are:

$$n(f_1, f_2) \cdot t(g_1, g_2) \cdot x(f_1, g_1) \cdot x(f_2, g_2)$$

   For each pair of flights f_1, f_2 and gates g_1, g_2, the coefficient of $x(f_1, g_1) \cdot x(f_2, g_2)$ is $n(f_1, f_2) \cdot t(g_1, g_2)$. This coefficient will be an entry in the Q matrix.

2. **Penalty Terms:** For the assignment constraint, the quadratic penalty term is:

$$P \sum_{f \in F} \left(1 - \sum_{g \in G} x(f, g) \right)^2$$

   Expanding the square, we get terms like $x(f, g_1) \cdot x(f, g_2)$ with a coefficient of P when $g_1 \neq g_2$ and a coefficient of $-2P$ for the term $x(f, g)^2$ (but since $x(f, g)^2 = x(f, g)$ for binary variables, it's effectively a linear term).

   For the occupancy constraint, the quadratic penalty term is:

$$P \sum_{f_1, f_2 \in F, g \in G} x(f_1, g) \cdot x(f_2, g)$$

   Here, the coefficient of $x(f_1, g) \cdot x(f_2, g)$ is P.

3. **Constructing Q:** The Q matrix is constructed such that Q_{ij} (the entry in the ith row and j^{th} column) is the coefficient of the product of the corresponding binary variables in the objective function.

   For example, if the binary variables are ordered as $x(f_1, g_1), x(f_1, g_2), \ldots, x(f_2, g_1), \ldots$, then the entry Q_{ij} is the coefficient of $x(f_i, g_j) \cdot x(f_k, g_l)$ in the objective function.

   Given the above terms, the Q matrix will have entries from the original objective function and the penalty terms. Specifically:

   - Q_{ij} will have the value $n(f_1, f_2) \cdot t(g_1, g_2)$ for the terms from the original objective function.

   - Q_{ij} will have the value P or $-2P$ for the terms from the penalty.

5.3 Instance generation

For the flight gate assignment problem our goal is to minimize the total transit time of passengers moving to/from gates, while also assigning all flights to gates so that flights do not overlap in the arrival/departure time window. The creation of the QUBO matrix for solving the flight gate assignment problem involves generating synthetic data that simulates real-world scenarios of flight schedules and gate assignments. This process is crucial for formulating the optimization problem in a manner that can be addressed using quantum computing techniques.

A set of flights and gates are considered, represented by F and G respectively. The number of flights, n_f, and the number of gates, n_g, are predefined parameters. Each flight and gate is assigned a unique identifier. For each flight, the number of arriving and departing passengers is randomly generated to simulate the variability in passenger load. This is crucial for computing the cost associated with passenger transfers and gate assignments. A key aspect of the problem is the handling of transfer passengers. A transfer probability, P_{tr}, is defined to model the likelihood that passengers need to trans-
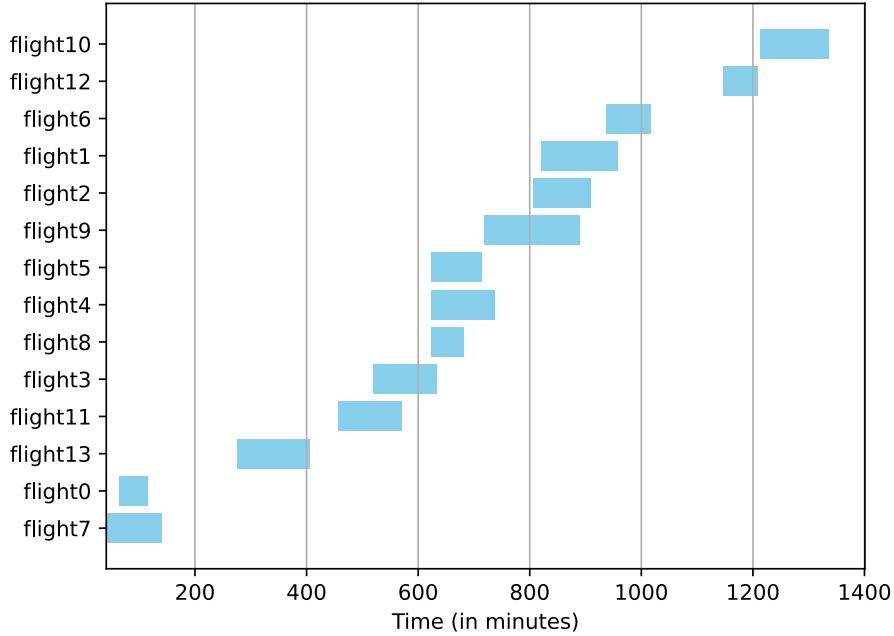
Figure 5.1: Duration of the time that each flight needs to spend to the airport after arriving and before departing from the airport, sorted by ascending arrival time. Here we consider a small example of 14 flights spending some time in the airport (60 minutes) during a period of 1 day (1440 minutes). The start of each bar represents the flights' arrival time at the airport and the end of each bar shows the departure time of the marked flight. Flights that have overlapping duration windows should not be assigned to the same gate.

fer between any two flights. This probability is used to randomly generate the number of transfer passengers between each pair of flights, introducing a layer of complexity to the gate assignment problem.

Each flight is assigned a random arrival and departure time within a 24-hour period, ensuring that the flight schedules are distributed throughout the day. Additionally, a buffer time is included to account for the time required to prepare a gate for the next flight. Along this buffer time, each aircraft need to spend some time in the airport for maintenance or waiting the passengers to board and more. Figure (5.1) shows the time that the aircraft operating the assigned flight spends in the airport regardless the gate that it will be assigned. Each bar denotes the total time that the indexed flight is

spending in the airport while the start of that bar can be viewed as the arrival time (the time that the flight arrives in the airport) and the end of the bar as the departure time (the time that the flight leaves the airport to perform the actual flight).
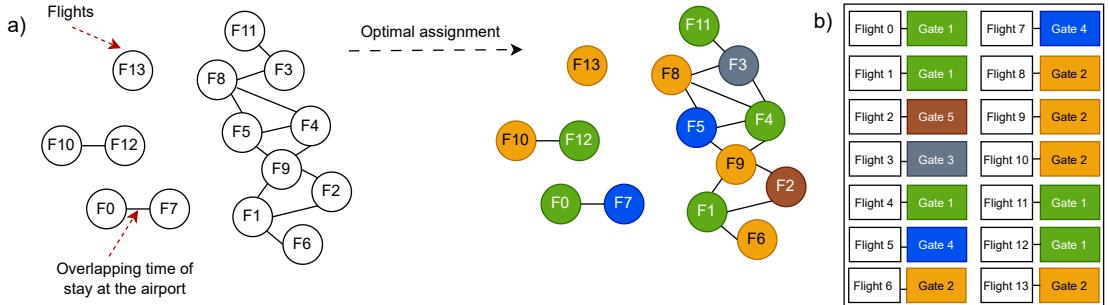

Figure 5.2: **a)** Overlapping graph for an example flight gate assignment problem of 14 flights and 5 gates. Nodes represents flights and an edge exists between two flights that have overlapping time windows. Since we want two overlapping flights not to be assigned in the same gate, we can color the nodes such that two connecting nodes have different colors, and obtain an optimal assignment of flight to gates with not overlapping time windows. **b)** Assignment schedule of flight to gates for this small example.

The QUBO matrix is then formulated based on the synthetic data. Each element of the matrix corresponds to a binary variable representing the assignment of a particular flight to a specific gate.

In order to visualize the problem we create a flight overlapping graph as shown in figure (5.2) where we connect flights that have overlapping time window of arrival/departure plus a buffer time for staying in the gate as seen in equation (5.8). Then, the optimal solution would be to assign gates (colors) to flights (nodes) such that no connecting nodes will have the same color and also minimize the total transit time of passengers.

5.4 RL Enhanced Optimization for Large FGA Problems

In this section, we present results for solving FGA instances shown in table (5.2). The pipeline of the RL enhanced algorithm is shown in figure (3.7) where we first use the minimal encoding scheme to obtain some good initial solutions and then use the RL enhanced algorithm to refine upon those solutions. Times for the classical simulator and reinforcement learning enhancement are chosen empirically by testing its performance and choose the ones that lead to an average approximation ratio $\alpha \geq 0.999$. Since the data for the FGA instances are synthetically created and the solution to the problem is unknown a-priori, we are using the commercial solver Gurobi [91] with an academic license and a time limit of 10 hours, after which we use the solution obtained to calculate C_{min} in the eq. (3.13) for the approximation ratio.

Flights	Gates	Classical variables	Qubits using minimal encoding	Time for classical simulator of the qubit efficient algorithm (s)	Time for RL enhancement (s)	Avg. approximation ratio (%)
55	14	770	11	60	2	99.98
88	22	1936	12	80	4	99.99
165	33	5445	14	120	8	99.98
201	36	7236	14	200	15	99.98
331	38	12578	15	1000	60	99.98
411	51	20961	16	2000	500	99.99

Table 5.2: Detailed information for all the FGA instances considered. The number of classical variables is the product of the number of flights and the number of gates. Time for classical simulator (seconds) refers to the time that we run the optimization using the minimal encoding scheme to produce initial solutions in a CPU machine. Time for reinforcement learning enhancement refers to the given GPU runtime of this algorithm to refine those initial solutions. The average approximation ratio obtained using data from 20 optimization runs using minimal encoding sequentially with the RL enhancement algorithm.

After obtaining the QUBO matrix Q for the FGA problem, one can set up the qubit efficient optimization algorithm using the minimal encoding scheme by preparing the quantum state in (3.5) using a hardware efficient ansatz shown in figure (2.2) with 10000 measurement shots. Sampling from this quantum state at each optimization step allows us to estimate and minimize the cost function in (3.10) using ADAM optimizer. After the optimization is done we can use coefficients $b_i(\vec{\theta}_{opt})$ in order to sample multiple solutions of the original FGA problem.

Using our qubit efficient encoding scheme we were able to fit FGA instances of 20961 classical variables to classical simulators using 16 qubits. Although, since we want to estimate the probability distribution in eq. (3.8) over $2^{16} = 65536$ probability amplitudes, 10000 shots is surely not enough. Amplitudes that are not measured will result in division with zero in eq. (3.9), therefore we address this issue by randomly assigning bits whose position registers is not measured sacrificing the overall accuracy of our algorithm. Consequently, in the limit of the qubit count going to infinity this technique could approximate a totally randomize sampling algorithm, evident also from eq. (3.8) which described a probability distribution of statistically independent variables. To address this issue we apply the RL enhancement algorithm after the quantum optimization to further enhance the quality of solutions obtained. This classical post-processing step is designed to run on a single laptop GPU[2] for faster computations.

Overall, for all FGA instances considered here, we were able to obtain solutions with quality comparable with this obtained from the commercial solver Gurobi. Combining the qubit efficient algorithm with the RL enhancement algorithm could offer a new way to address large scale QUBO problems in the NISQ era.

---

[2]Although this can be extended to multiple GPUs by leveraging the fact that using the best variational angles from the qubit efficient algorithm we can sample multiple solutions and use the RL enhancement algorithm to refine these solutions in parallel GPUs.
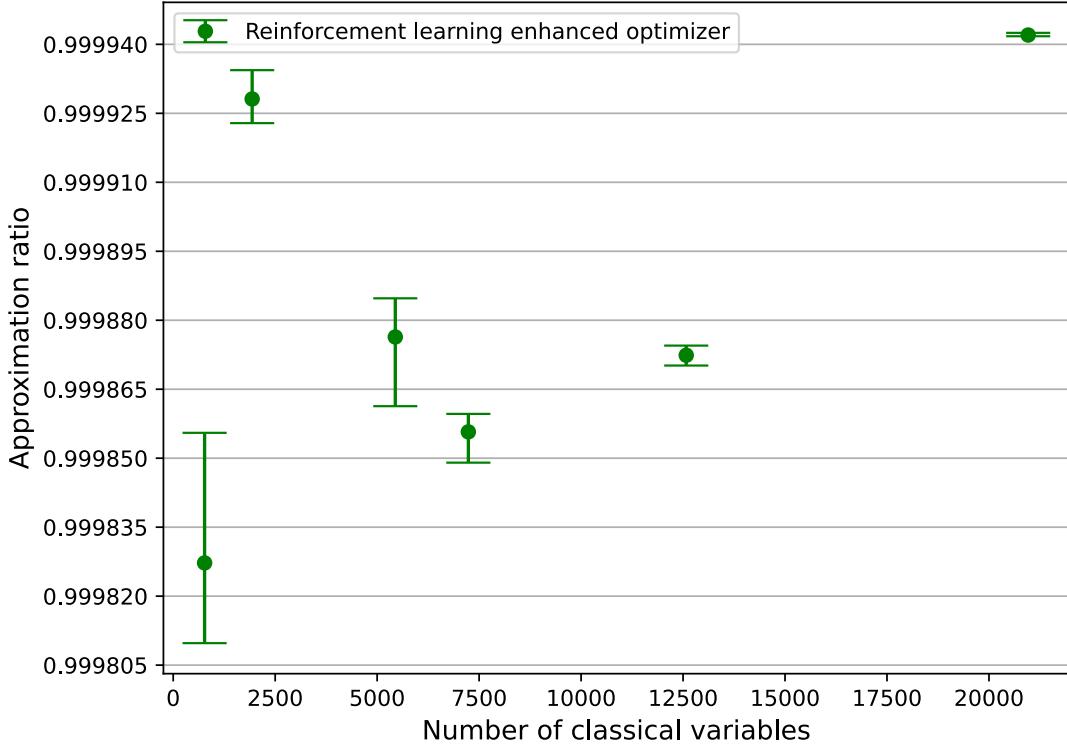
Figure 5.3: Approximation ratio for flight gate assignment problems of up to 20961 classical variables using the reinforcement learning enhanced optimizer and minimal encoding with 16 qubits. 20 optimization runs were performed and we plot the median approximation ratio (circle) as well as the minimum and maximum values found as error bars.

Figure (5.3) shows the average approximation ratio obtained using the RL enhanced optimizer shown in fig. (3.7) for all FGA instances considered here. For each FGA instance we perform 20 different optimization runs using different initial variational angles for the qubit efficient algorithm. For each optimization run we first use the minimal encoding scheme to approximate the cost function in eq. (3.10) and then we use the ADAM optimizer in order to find the variational angles that minimize this cost function. Then we use the best variational angles to run the classical simulator and use the probability distribution from (3.8) to sample 10 solutions per optimization run.

For each optimization run, we sort these 10 solutions based on their cost function values using equation (2.1) and then choose the one with the lowest cost function value to be given as input to the reinforcement learning enhancement algorithm. After running the RL algorithm for a predefined time (seconds) shown in table (5.2), we obtain one final post-processed solution for each optimization run[3], resulting in 20 final solutions per problem instance and plot the average approximation ratio.

The average approximation ratio for all FGA instances display small deviation from the mean, indicating that for each problem instance and for each of the 20 optimization runs the RL enhanced optimizer converged satisfactorily. Solutions obtained from the RL enhanced optimizer have a high overlap with those obtained from Gurobi, although none of them is the same as the approximation ratio is less than one. For the two bigger FGA instances, namely the ones with 12578 and 20961 classical variables the deviation from the mean appears to be very small. This indicates that for all the optimization runs the RL enhanced optimizer converged to solutions nearly identical in terms of cost function value, starting from 20 different solutions[4].

Results for a single optimization run are shown in figure (5.4) where we plot the qubit efficient optimization part as well as the reinforcement learning part. The main plot shows the evolution of the approximation ratio using the RL enhanced algorithm. The inset axis shows the optimization procedure using the minimal encoding scheme and compares with random sampling. For the random sampling we are using 4×10^6 random solutions for problem sizes with less than 10000 classical variables and 4×10^8 random solutions for problem sizes with more than 10000 classical variables. We observe that as the problem size grows, minimal encoding produces better than random solutions but is susceptible to barren plateaus as the approximation ratio produced during optimization becomes flat.

---

[3]Appendix section (7.2) shows the pseudocodes for the qubit efficient and the RL enhanced algorithms.
[4]Since in the quantum optimization part we have fixed the number of measurement shots to 10000 we are bound to randomly assigning bits in the proposed solution.
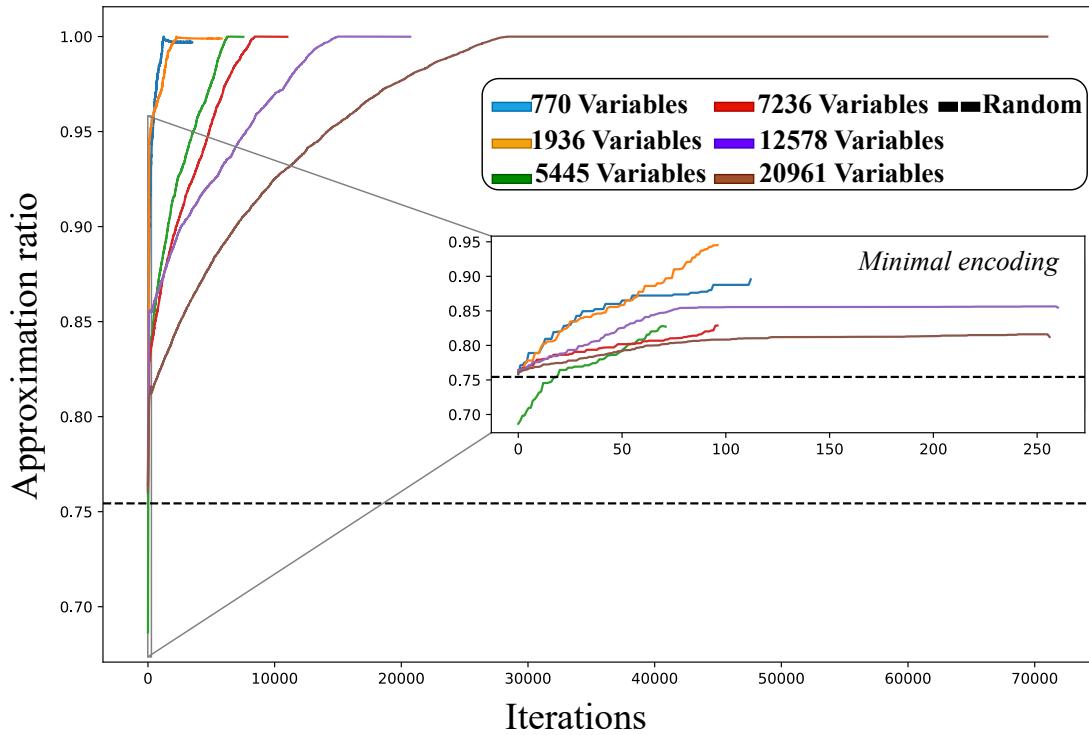
Figure 5.4: Optimization progress for all FGA problem instances. The main plot focuses on iterations performed by the reinforcement learning algorithm in order to enhance the quality of solutions from the minimal encoding scheme. The inset axis shows the performance of the minimal encoding scheme during optimization and is compared with an average approximation ratio obtained through random sampling.

6 Conclusion and Future Work

This thesis has presented a quantum-classical approach for addressing large-scale Quadratic Unconstrained Binary Optimization (QUBO) problems. The proposed reinforcement learning enhanced algorithm integrates qubit efficient encoding with advanced computational strategies, offering a novel pathway in the realm of quantum computing. This duality of quantum and classical techniques has demonstrated significant promise in dealing with complex optimization challenges, as evidenced by its application to real-world problems in routing and scheduling.

We applied a qubit efficient encoding scheme from [55] to drastically reduce the number of qubits needed, in order to tackle large QUBO problems. The downside of this technique is that the number of shots needed to accurately characterize the quantum state and thus the approximated cost function scales exponentially with the number of qubits. To address this issue we perform an additional RL-based post processing step in order to enhance the quality of solutions obtained by this scheme. This RL algorithm is designed to run on GPUs for better performance and it is based on an exploration-exploitation process.

We demonstrate the applicability of the RL enhanced optimizer in 3 optimization problems (vehicle routing problem with time windows, tail assignment problem and flight gate assignment problem) from industry which can be mapped to QUBO format. For certain problem sizes of the routing problem, we apply the qubit efficient encoding scheme to obtain solutions from real quantum computers through the cloud and enhance

upon these solution using the RL algorithm. Furthermore, we increase the problem size for all optimization problems to nearly 25,000 classical variables and use quantum simulators alongside a single laptop GPU to approximately solve them and compare the quality of solutions obtained with those obtained from a commercial solver. Comparing the approximation ratios for the 3 problems we find that the tail assignment problem has the lowest. This is because the tail assignment problem inherently has a unique optimal solution and due to eq. (4.2) other non optimal solutions may have a large overlap with the optimal solution. [1]

Future work may involve leveraging high-performance computing clusters to scale the algorithm for tackling even larger and more complex problems relevant to industry. The use of high-performance clusters is expected to significantly enhance computational capabilities, allowing for more extensive and intricate computations that are currently beyond reach. This progression will enable the exploration of new frontiers in optimization problems, particularly those with substantial industrial relevance. For example, instead of refining one solution, one could use the RL enhancement algorithm in parallel to enhance the full cluster of solutions sampled using the minimal encoding scheme.

Another interesting avenue, will be to study the relevant gap between the proposed solutions of the minimal encoding and the optimal solution. Connecting this gap with the probability to sample a solution could provide upper bounds in the accuracy of the overall algorithm. It is important to note that this probability is parameterized by the angles of the quantum circuit at hand and encodes independent classical variables. One has to use another encoding also proposed by Tan et al. [55] to encode correlations between classical variables in this probability but using more qubits than minimal encoding.

---

[1] If the optimal solution contains some routes and we delete one route, the resulting solution will have a large overlap with the optimal solution and a little difference in the cost function.

7 Appendix

7.1 Practical Analysis of IBM Cairo Device Used In Our Implementation

Cloud quantum computing represents a significant shift in computational power, utilizing quantum computing to enhance information processing. By integrating quantum computing with cloud technology, it becomes widely accessible, allowing the provision of quantum computing services via the internet. Users can remotely access quantum processors and conduct computations without the need for direct ownership or maintenance of costly hardware. Cloud quantum computing is pivotal for advancing quantum technologies, enabling practical experimentation with quantum algorithms, application development, and contributing to the quantum computing field. Companies, including IBM Quantum Experience [114], Amazon Braket [115], Google Quantum AI [116], Microsoft Azure Quantum [117], and IonQ [118], have developed advanced cloud quantum computing platforms, each offering unique resources and tools.

This section presents an analysis of the performance of a hardware efficient quantum circuit (2.2) when executed on the 27-qubit IBM Cairo quantum device we used for our results in chapter (3.2.1). The analysis is based on three key metrics :

1. Execution time of the hardware efficient ansatz including the gradient estimation time using the parameter shift rule described at section (2.4.2)

2. The number of gates of the hardware efficient ansatz and the resulting number of gates after translating the quantum circuit to the native quantum hardware

3. The memory it takes to classically store QUBO matrices as well as quantum states

In order to gather the data for this analysis we used the official Qiskit libary provided by IBM [129] and namely the Properties package which holds the properties of each gate. For the quantum circuit at hand we are using a hardware efficient ansatz as shown in Fig. (2.2) with 4 layers.

7.1.1 Run Time Analysis

Figure (7.1) shows the total run times for hardware efficient quantum circuits with varying numbers of qubits, ranging from 11 to 27 qubits. The total run time is given by

$$t_r = t_{tr} + \text{num}_{\text{shots}} \times t_e + \text{num}_{\text{shots}} \times t_{gc} \tag{7.1}$$

where t_{tr} is the time it takes for the transpilation of the quantum circuit, which refers to the time it takes to transform the given quantum circuit to native quantum gates that respect the architecture of the IBM Cairo quantum device. $\text{num}_{\text{shots}} \times t_e$ refers to the time it takes to run the transpiled quantum circuit $\text{num}_{\text{shots}}$ times where $\text{num}_{\text{shots}} = 10000$, and t_e is the single execution time of the transpiled quantum circuit. $\text{num}_{\text{shots}} \times t_{gc}$ refers to the time it takes to estimate the gradients using the parameter shift rule as shown in section (2.4.2). Essentially, since we have to execute the quantum circuit 2 more times to estimate the gradients, one with the angles shifted by $\frac{\pi}{2}$ and one with the angles shifted by $-\frac{\pi}{2}$, the gradient execution time $t_{gc} = 2t_e$.

Overall in Fig. (7.1), the total run time can be seen as the height of the bar for each qubit count, where t_{tr} can be seen with light-blue color, $\text{num}_{\text{shots}} \times t_e$ with dark-blue color and $\text{num}_{\text{shots}} \times t_{gc}$ with blue color. The summation of these times can give us an
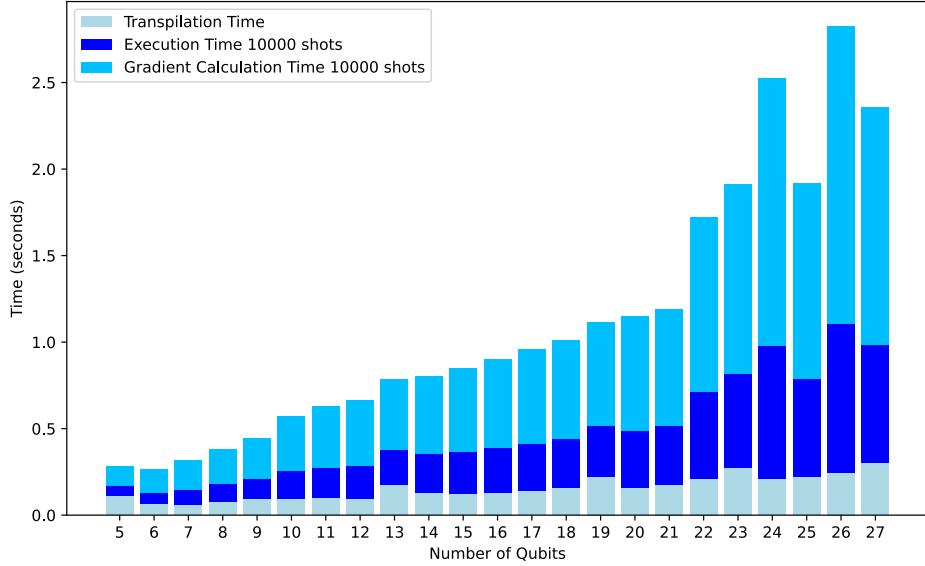
Figure 7.1: Total run time of hardware-efficient quantum circuits on the IBM Cairo device including the gradient estimation time using the parameter shift rule. The transpilation time refers to the time it takes to convert the hardware efficient ansatz into native gates with respect to the IBM Cairo device. The execution time and gradient calculation time, using the parameter shift rule to estimate the gradients, are scaled up to reflect realistic experiments where we want to measure the quantum circuit using 10000 shots.

essence of how much time it would take to run a single iteration of the qubit efficient algorithm for each number of qubits [1].

7.1.2 Gate Count Before and After Transpilation

Quantum circuits designed for execution on the IBM Cairo quantum device undergo a process known as transpilation. This process adapts the circuit to the specific architecture and native gate set of the quantum computer. The IBM Cairo device, which utilizes a superconducting qubit architecture, supports a specific set of native gates.

---

[1]Note here that we have excluded the time to estimate the cost function in eq. (3.10) since this can be done in a CPU/GPU machine rather than a QPU.
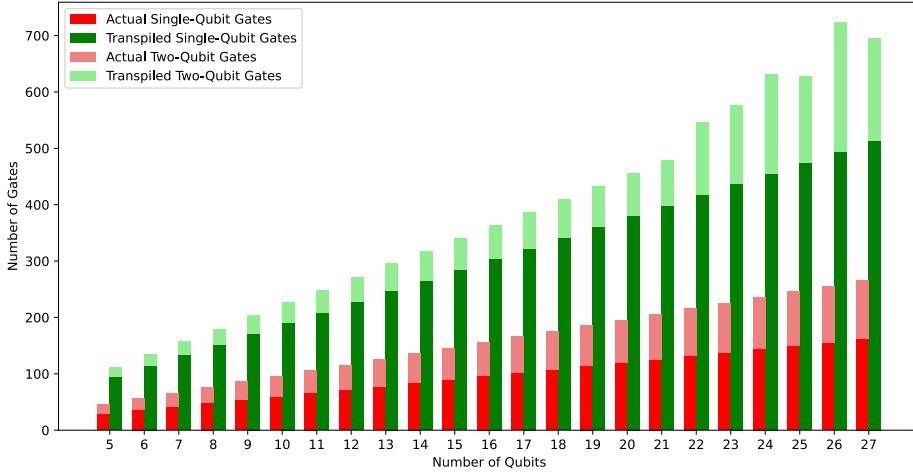
Figure 7.2: Comparison of gate counts before and after transpilation for a hardware efficient ansatz comprising of 4 layers on the Cairo device.

The transpilation process decomposes the high-level gates of a hardware-efficient ansatz into these native gates. The primary gate types used after transpilation include:

- **U1, U2, U3 Gates:** These represent general single-qubit rotation gates. The U1 gate is a rotation around the Z-axis, U2 is a general single-qubit rotation with one parameter, and U3 is the most general form of a single-qubit rotation with three parameters.

- **SX Gate (or $\sqrt{X}$ Gate):** This gate performs a $\pi/2$ rotation around the X-axis on the Bloch sphere. It is also known as the square root of the Pauli-X gate.

- **X Gate:** Also known as the Pauli-X gate, this gate performs a π rotation around the X-axis. It functions as a NOT gate in quantum computing.

- **CNOT Gate (Controlled-NOT Gate):** Essential for creating entanglement, this gate flips the state of the target qubit if the control qubit is in the state $|1\rangle$.

- **Idle Gates or Delays:** These gates are used for synchronization or for waiting.

The transpiler in Qiskit, IBM's quantum computing framework, optimizes the circuit for the Cairo device's connectivity and gate fidelities. It aims to minimize the gate count and circuit depth to reduce the impact of noise and decoherence. The transpilation process involves merging, commuting, or canceling gates where possible to reduce the complexity of the final circuit and ensure adaptability to the hardware's limitations and characteristics. Figure (7.2) illustrates the number of gates in the quantum circuits both before and after transpilation.

7.1.3 Memory Usage Analysis

In the context of implementing the qubit efficient algorithm, two critical factors contribute significantly to memory usage: the representation of the quantum state and the QUBO matrix. The quantum state memory grows exponentially with the number of qubits due to the need to represent the state vector of a quantum system, while the memory required to store the QUBO matrix, representing the problem to be solved, grows with the square of the number of classical variables encoded by the qubits.

The memory required to store a quantum state, where each coefficient is a complex number (requiring two floats), is given by:

$$\text{Quantum State Memory (bytes)} = 2^n \times 16$$

where n is the number of qubits and we assume 64-bit (8-byte) precision.

For 18 qubits, the quantum state and QUBO matrix memory requirements are calculated as follows:

The quantum state memory for 18 qubits is given by:

$$\text{Quantum State Memory} = 2^{18} \times 16 \text{ bytes} = 262144 \times 16 \text{ bytes} = 4194304 \text{ bytes}$$

Converting this to gigabytes:

$$\text{Quantum State Memory} = \frac{4194304}{1024^3} \text{ GB} \approx 4.00 \text{ MB}$$

To calculate the memory requirements for holding a QUBO matrix of size N, we consider using the minimal encoding from [55] in order to calculate the number of classical variables N.

The memory required for a QUBO matrix of size $N \times N$, where each element is a single float, is given by:

$$\text{QUBO Matrix Memory (bytes)} = N^2 \times 8$$

Table (7.1) summarizes the memory requirements for quantum states and QUBO matrices for various numbers of qubits. Storing such matrices in their entirety is not feasible with current conventional computing resources. For instance, a QUBO matrix for a system with 18 qubits, serving 130 thousan variables, requires approximately 128 GB of memory, and the requirement grows exponentially with more qubits.

A significant observation is that QUBO matrices for many practical problems are sparse, meaning most of their elements are zero. This sparsity arises because not all variables in a QUBO problem interact with each other. Leveraging this sparsity can drastically reduce the memory and computational requirements. There are several techniques to exploit the sparsity of QUBO matrices:

- **Sparse Matrix Representations:** Instead of storing every element of the matrix, sparse matrix formats store only the non-zero elements and their indices. This approach can lead to significant reductions in memory usage.

- **Distributed Computing:** Large QUBO matrices can be partitioned and distributed across multiple computational nodes.

Number of Qubits using Minimal Encoding	Number of Classical Variables	Quantum State Memory	QUBO Matrix Memory
11	1024	0.03 MB	8 MB
12	2048	0.06 MB	32 MB
13	4096	0.12 MB	128 MB
14	8192	0.25 MB	512 MB
15	16384	0.50 MB	2 GB
16	32768	1.00 MB	8 GB
17	65536	2.00 MB	32 GB
18	131072	4.00 MB	128 GB
19	262144	8.00 MB	512 GB
20	524288	16.00 MB	2 TB
21	1048576	32.00 MB	8 TB
22	2097152	64.00 MB	32 TB
23	4194304	128.00 MB	128 TB
24	8388608	256.00 MB	512 TB
25	16777216	512.00 MB	2048 TB

Table 7.1: Memory requirements for quantum states and QUBO matrices. The first column shows the number of qubits of the quantum system while the seconds column shows the number of classical variables that we can handle using the minimal encoding scheme. As mentioned in [55] the relationship of number of qubits and number of classical variables using the minimal encoding scheme is given by Number of Classical Variables $= 2^{\text{Number of qubits}-1}$

GPUs and other specialized hardware architectures offer parallel processing capabilities that are particularly effective for handling large, sparse matrices. By performing multiple operations simultaneously, GPUs can significantly accelerate the computation of QUBO matrices and related algorithms.

7.2 Pseudocode for Main Algorithms

Algorithm (1) shows the optimization step used in variational quantum circuits. Given an optimizer [2] and a set of variational angles [3], the algorithm performs an optimization step to minimize the cost function associated with the quantum circuit. The updated

---

[2]Throughout this thesis we are using the ADAM optimizer.

[3]We choose to start with random initial angles.

variational angles and the new cost are returned, which are then used in further iterations of the optimization process. The cost function is calculated using eq. (3.10) as we are using minimal encoding throughout this thesis [55].

Algorithm 1 Optimizer Step

Require: Optimizer opt, Variational angles $\vec{\theta}$
Ensure: New variational angles $\vec{\theta}$, New cost opt_cost
1: $\vec{\theta}, \mathrm{opt\_cost} \leftarrow opt.\mathrm{step\_and\_cost}(cost\_function, \vec{\theta})$
2: **return** $\vec{\theta}, \mathrm{opt\_cost}$

Algorithm (2) outlines the full pipeline of the reinforcement learning enhanced optimizer shown in (3.7) for solving QUBO problems. It initializes the quantum circuit and performs optimization using the minimal encoding scheme as describes in section (3.2.2). After the optimization, it samples bitstrings from the VQA and applies the reinforcement learning enhancement algorithm to each bitstring, aiming to enhance the quality of the solutions. The best bitstring and cost found over the iterations are returned.

Algorithm 2 Reinforcement Learning Enhanced Optimizer

Require: QUBO matrix, QUBO Constant, Number of layers , Number of bitstrings to sample, Optimizer time, RL algorithm time, Number of cycles, Initial temperature, Verbosity
 1: Initialize quantum circuit parameters: nqq, n_layers
 2: Check and set computing device (CPU or GPU)
 3: Configure quantum device or simulator
 4: Initialize optimizer and variational angles $\vec{\theta}$
 5: **for** Number of cycles **do**
 6:     Optimize VQA for Optimizer time
 7:     Sample candidate solutions from VQA
 8:     Apply RL enhancement on each solution for RL algorithm time / number of solutions
 9:     Record best cost and best solution
10: **end for**
11: **return** best solution, best cost

Algorithm (3) describes the reinforcement learning enhancement algorithm. It is applied to enhance the quality of solutions obtained from the quantum optimization part.

The algorithm iteratively explores and exploits the solution space using softmax for exploration and UCB for exploitation, adjusting the state based on rewards and counts, as explained in section (3.4). The best state and cost found within the given time limit are returned.

Algorithm 3 Reinforcement Learning Enhancement Algorithm

Require: Initial solution, QUBO matrix, QUBO constant, time limit, temperature
  Initialize rewards, counts, and best solution
  while within time limit **do**
    if in exploration phase **then**
      Calculate flip probabilities using softmax
      Flip multiple bits based on probabilities
    else
      Calculate UCB scores for each bit
      Flip the bit with highest UCB score
    end if
    Update current solution, rewards, and counts
    Adjust temperature
  end while
  return best solution, best cost

Bibliography

[1] 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. 034 009, 2020.

[2] S. Harwood, C. Gambella, D. Trenev, A. Simonetto, D. Bernal, and D. Greenberg, "Formulating and solving routing problems on quantum computers," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–17, 2021. DOI: 10.1109/TQE.2021.3049230.

[3] G. Kochenberger, J. Hao, F. Glover, *et al.*, "The unconstrained binary quadratic programming problem: A survey," *Journal of Combinatorial Optimization*, vol. 28, no. 1, pp. 58–81, 2014.

[4] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using qubo models," *arXiv preprint arXiv:1811.11538*, 2018.

[5] Ö. Salehi, A. Glos, and J. A. Miszczak, "Unconstrained binary models of the travelling salesman problem variants for quantum optimization," *Quantum Information Processing*, vol. 21, no. 2, p. 67, 2022.

[6] F. Dominguez, J. Unger, M. Traube, B. Mant, C. Ertler, and W. Lechner, "Encoding-independent optimization problem formulation for quantum computing," *arXiv preprint arXiv:2302.03711*, 2023.

[7] M. Mattesi, L. Asproni, C. Mattia, *et al.*, *Financial portfolio optimization: A qubo formulation for sharpe ratio maximization*, 2023. arXiv: 2302.12291 [quant-ph].

[8] F. Glover, G. Kochenberger, R. Hennig, and Y. Du, "Quantum bridge analytics i: A tutorial on formulating and using qubo models," *Annals of Operations Research*, vol. 314, no. 1, pp. 141–183, Jul. 2022, ISSN: 1572-9338. DOI: 10.1007/s10479-022-04634-2. [Online]. Available: https://doi.org/10.1007/s10479-022-04634-2.

[9] M. Streif, S. Yarkoni, A. Skolik, F. Neukart, and M. Leib, "Beating classical heuristics for the binary paint shop problem with the quantum approximate optimization algorithm," *Phys. Rev. A*, vol. 104, p. 012 403, 1 Jul. 2021. DOI: 10.1103/PhysRevA.104.012403. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.104.012403.

[10] D. Venturelli, D. J. Marchand, and G. Rojo, "Quantum annealing implementation of job-shop scheduling," *arXiv preprint arXiv:1506.08479*, 2015.

[11] A. P. Adelomou, E. G. Ribe, and X. V. Cardona, *Using the parameterized quantum circuit combined with variational-quantum-eigensolver (vqe) to create an intelligent social workers' schedule problem solver*, 2020. arXiv: 2010.05863 [quant-ph].

[12] C. Lin, K. L. Choy, G. T. Ho, S. H. Chung, and H. Lam, "Survey of green vehicle routing problem: Past and future trends," *Expert systems with applications*, vol. 41, no. 4, pp. 1118–1138, 2014.

[13] R. Moghdani, K. Salimifard, E. Demir, and A. Benyettou, "The green vehicle routing problem: A systematic literature review," *Journal of Cleaner Production*, vol. 279, p. 123 691, 2021.

[14] I. Kara, B. Y. Kara, and M. K. Yetis, "Energy minimizing vehicle routing problem," in *Combinatorial Optimization and Applications: First International Con-*

ference, COCOA 2007, Xi'an, China, August 14-16, 2007. Proceedings 1, Springer, 2007, pp. 62–71.

[15] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, *et al.*, "Noisy intermediate-scale quantum algorithms," *Rev. Mod. Phys.*, vol. 94, p. 015 004, 1 Feb. 2022. DOI: `10.1103/RevModPhys.94.015004`. [Online]. Available: `https://link.aps.org/doi/10.1103/RevModPhys.94.015004`.

[16] N. P. Sawaya, A. T. Schmitz, and S. Hadfield, "Encoding trade-offs and design toolkits in quantum algorithms for discrete optimization: Coloring, routing, scheduling, and other problems," *arXiv preprint arXiv:2203.14432*, 2022.

[17] A. Finnila, M. Gomez, C. Sebenik, C. Stenson, and J. Doll, "Quantum annealing: A new method for minimizing multidimensional functions," *Chemical Physics Letters*, vol. 219, no. 5, pp. 343–348, 1994, ISSN: 0009-2614. DOI: `https://doi.org/10.1016/0009-2614(94)00117-0`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/0009261494001170`.

[18] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Phys. Rev. E*, vol. 58, pp. 5355–5363, 5 Nov. 1998. DOI: `10.1103/PhysRevE.58.5355`. [Online]. Available: `https://link.aps.org/doi/10.1103/PhysRevE.58.5355`.

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

[20] ——, "A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem," *arXiv preprint arXiv:1412.6062*, 2014.

[21] E. Farhi, J. Goldstone, S. Gutmann, and L. Zhou, "The quantum approximate optimization algorithm and the sherrington-kirkpatrick model at infinite size," *arXiv preprint arXiv:1910.08187*, 2019.

[22] K. Blekos, D. Brand, A. Ceschini, *et al.*, *A review on quantum approximate optimization algorithm and its variants*, 2023. arXiv: 2306.09198 [quant-ph].

[23] S. Hadfield, Z. Wang, B. O'gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, 2019.

[24] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *arXiv preprint arXiv:1812.01041*, 2018.

[25] H. Mohammadbagherpoor, P. Dreher, M. Ibrahim, *et al.*, *Exploring airline gate-scheduling optimization using quantum computers*, 2021. arXiv: 2111.09472 [quant-ph].

[26] T. Chatterjee, S. I. Mohtashim, and A. Kundu, *On the variational perspectives to the graph isomorphism problem*, 2021. arXiv: 2111.09821 [quant-ph].

[27] P. Díez-Valle, D. Porras, and J. J. García-Ripoll, "Quantum variational optimization: The role of entanglement and problem hardness," *Phys. Rev. A*, vol. 104, p. 062 426, 6 Dec. 2021. DOI: 10.1103/PhysRevA.104.062426. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.104.062426.

[28] D. J. Egger, C. Gambella, J. Marecek, *et al.*, "Quantum computing for finance: State-of-the-art and future prospects," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–24, 2020. DOI: 10.1109/TQE.2020.3030314.

[29] D. Amaro, C. Modica, M. Rosenkranz, M. Fiorentini, M. Benedetti, and M. Lubasch, "Filtering variational quantum algorithms for combinatorial optimization," *Quantum Science and Technology*, vol. 7, no. 1, p. 015 021, 2022.

[30] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018, ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. [Online]. Available: https://doi.org/10.22331/q-2018-08-06-79.

[31] F. Leymann and J. Barzen, "The bitter truth about gate-based quantum algorithms in the nisq era," *Quantum Science and Technology*, vol. 5, no. 4, p. 044 007, 2020.

[32] J. W. Z. Lau, K. H. Lim, H. Shrotriya, and L. C. Kwek, "Nisq computing: Where are we and where do we go?" *AAPPS Bulletin*, vol. 32, no. 1, p. 27, 2022.

[33] F. Glover, M. Lewis, and G. Kochenberger, "Logical and inequality implications for reducing the size and difficulty of quadratic unconstrained binary optimization problems," *European Journal of Operational Research*, vol. 265, no. 3, pp. 829–842, 2018, ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2017.08.025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221717307567.

[34] E. Boros, P. Hammer, and G. Tavares, "Preprocessing of unconstrained quadratic binary optimization," *RUTCOR Research Report*, Jan. 2006. [Online]. Available: http://users.cecs.anu.edu.au/~pcarr/qpbo/BorosRRR102006.pdf.

[35] J.-H. Lange, B. Andres, and P. Swoboda, "Combinatorial persistency criteria for multicut and max-cut," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6086–6095. DOI: 10.1109/CVPR.2019.00625. [Online]. Available: https://doi.org/10.1109/CVPR.2019.00625.

[36] D. Ferizovic, D. Hespe, S. Lamm, M. Mnich, C. Schulz, and D. Strash, "Engineering kernelization for maximum cut," in *2020 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*. 2020 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX), 2020, pp. 27–41. DOI: 10.1137/1.9781611976007.3. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976007.3. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611976007.3.

[37] P. L. Hammer and S. Rudeanu, "Minimization of nonlinear pseudo-boolean functions," in *Boolean Methods in Operations Research and Related Areas*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1968, pp. 113–134, ISBN: 978-3-642-85823-9. DOI: 10.1007/978-3-642-85823-9_7. [Online]. Available: https://doi.org/10.1007/978-3-642-85823-9_7.

[38] R. Shaydulin and M. Pistoia, *QAOA with $N \cdot p \geq 200$*, 2023. arXiv: 2303.02064 [quant-ph].

[39] E. Pelofske, A. Bärtschi, and S. Eidenbenz, "Quantum annealing vs. QAOA: 127 qubit higher-order Ising problems on NISQ computers," in *Lecture Notes in Computer Science*, Springer Nature Switzerland, 2023, pp. 240–258. DOI: 10.1007/978-3-031-32041-5_13. [Online]. Available: https://doi.org/10.1007%2F978-3-031-32041-5_13.

[40] A. Lowe, M. Medvidović, A. Hayes, *et al.*, "Fast quantum circuit cutting with randomized measurements," *Quantum*, vol. 7, p. 934, Mar. 2023, ISSN: 2521-327X. DOI: 10.22331/q-2023-03-02-934. [Online]. Available: https://doi.org/10.22331/q-2023-03-02-934.

[41] Z. H. Saleem, T. Tomesh, M. A. Perlin, P. Gokhale, and M. Suchara, *Divide and conquer for combinatorial optimization and distributed quantum computation*, 2022. arXiv: 2107.07532 [quant-ph].

[42] M. Bechtold, J. Barzen, F. Leymann, *et al.*, *Investigating the effect of circuit cutting in QAOA for the Maxcut problem on NISQ devices*, 2023. arXiv: 2302.01792 [quant-ph].

[43] T. Peng, A. W. Harrow, M. Ozols, and X. Wu, "Simulating large quantum circuits on a small quantum computer," *Physical Review Letters*, vol. 125, no. 15, Oct. 2020. DOI: 10.1103/physrevlett.125.150504. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.125.150504.

[44] K. Fujii, K. Mizuta, H. Ueda, K. Mitarai, W. Mizukami, and Y. O. Nakagawa, "Deep variational quantum eigensolver: A divide-and-conquer method for solving a larger problem with smaller size quantum computers," *PRX Quantum*, vol. 3, p. 010 346, 1 Mar. 2022. DOI: `10.1103/PRXQuantum.3.010346`. [Online]. Available: `https://link.aps.org/doi/10.1103/PRXQuantum.3.010346`.

[45] K. Mitarai and K. Fujii, "Constructing a virtual two-qubit gate by sampling single-qubit operations," *New Journal of Physics*, vol. 23, no. 2, p. 023 021, Feb. 2021. DOI: `10.1088/1367-2630/abd7bc`. [Online]. Available: `https://dx.doi.org/10.1088/1367-2630/abd7bc`.

[46] S. Bravyi, G. Smith, and J. A. Smolin, "Trading classical and quantum computational resources," *Physical Review X*, vol. 6, no. 2, Jun. 2016. DOI: `10.1103/physrevx.6.021043`. [Online]. Available: `https://doi.org/10.1103%2Fphysrevx.6.021043`.

[47] Y. Zhang, L. Cincio, C. F. A. Negre, *et al.*, "Variational quantum eigensolver with reduced circuit complexity," *npj Quantum Information*, vol. 8, no. 1, p. 96, Aug. 2022, ISSN: 2056-6387. DOI: `10.1038/s41534-022-00599-z`. [Online]. Available: `https://doi.org/10.1038/s41534-022-00599-z`.

[48] C.-Y. Liu and H.-S. Goan, *Hybrid gate-based and annealing quantum computing for large-size Ising problems*, 2022. arXiv: `2208.03283 [quant-ph]`.

[49] M. Boost, S. Reinhardt, and A. Roy, "Partitioning optimization problems for hybrid classical/quantum execution," Technical Report, http://www. dwavesys. com, Tech. Rep., 2017.

[50] W. Tang and M. Martonosi, *Scaleqc: A scalable framework for hybrid computation on quantum and classical processors*, 2022. arXiv: `2207.00933 [cs.ET]`.

[51] C. Piveteau and D. Sutter, *Circuit knitting with classical communication*, 2023. arXiv: `2205.00016 [quant-ph]`.

[52] A. Glos, A. Krawiec, and Z. Zimborás, "Space-efficient binary optimization for variational quantum computing," *npj Quantum Information*, vol. 8, no. 1, p. 39, Apr. 2022, ISSN: 2056-6387. DOI: 10.1038/s41534-022-00546-y. [Online]. Available: https://doi.org/10.1038/s41534-022-00546-y.

[53] J. I. Latorre, *Image compression and entanglement*, 2005. arXiv: quant-ph/0510031 [quant-ph].

[54] J. Plewa, J. Sieńko, and K. Rycerz, "Variational algorithms for workflow scheduling problem in gate-based quantum devices," *COMPUTING AND INFORMATICS*, vol. 40, no. 4, pp. 897–929, Dec. 2021. DOI: 10.31577/cai_2021_4_897. [Online]. Available: https://www.cai.sk/ojs/index.php/cai/article/view/2021_4_897.

[55] B. Tan, M.-A. Lemonde, S. Thanasilp, J. Tangpanitanon, and D. G. Angelakis, "Qubit-efficient encoding schemes for binary optimisation problems," *Quantum*, vol. 5, p. 454, May 2021. DOI: 10.22331/q-2021-05-04-454. [Online]. Available: https://doi.org/10.22331/q-2021-05-04-454.

[56] B. Fuller, C. Hadfield, J. R. Glick, *et al.*, *Approximate solutions of combinatorial problems via quantum relaxations*, 2021. arXiv: 2111.03167 [quant-ph].

[57] K. Teramoto, R. Raymond, E. Wakakuwa, and H. Imai, *Quantum-relaxation based optimization algorithms: Theoretical extensions*, 2023. arXiv: 2302.09481 [quant-ph].

[58] M. J. Ran či ć, "Noisy intermediate-scale quantum computing algorithm for solving an n-vertex Maxcut problem with $\log(n)$ qubits," *Phys. Rev. Res.*, vol. 5, p. L012021, 1 Feb. 2023. DOI: 10.1103/PhysRevResearch.5.L012021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.5.L012021.

[59]   D. Winderl, N. Franco, and J. M. Lorenz, *A comparative study on solving optimization problems with exponentially fewer qubits*, 2022. arXiv: 2210.11823 [quant-ph].

[60]   J.-G. Liu, Y.-H. Zhang, Y. Wan, and L. Wang, "Variational quantum eigensolver with fewer qubits," *Physical Review Research*, vol. 1, no. 2, Sep. 2019. DOI: 10.1103/physrevresearch.1.023025. [Online]. Available: https://doi.org/10.1103%2Fphysrevresearch.1.023025.

[61]   M. DeCross, E. Chertkov, M. Kohagen, and M. Foss-Feig, *Qubit-reuse compilation with mid-circuit measurement and reset*, 2022. arXiv: 2210.08039 [quant-ph].

[62]   F. Hua, Y. Jin, Y. Chen, *et al.*, *Exploiting qubit reuse through mid-circuit measurement and reset*, 2023. arXiv: 2211.01925 [quant-ph].

[63]   M. Dupont, B. Evert, M. J. Hodson, *et al.*, "Quantum enhanced greedy solver for optimization problems," *arXiv preprint arXiv:2303.05509*, 2023.

[64]   C.-Y. Liu and H.-S. Goan, "Reinforcement learning quantum local search," *arXiv preprint arXiv:2304.06473*, 2023.

[65]   D. Beloborodov, A. E. Ulanov, J. N. Foerster, S. Whiteson, and A. Lvovsky, "Reinforcement learning enhanced quantum-inspired algorithm for combinatorial optimization," *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 025 009, 2020.

[66]   T. Stollenwerk, E. Lobe, and M. Jung, "Flight gate assignment with a quantum annealer," in *International Workshop on Quantum Technology and Optimization Problems*, Springer, 2019, pp. 99–110.

[67]   B. Lodewijks, "Mapping np-hard and np-complete optimisation problems to quadratic unconstrained binary optimisation problems," *arXiv preprint arXiv:1911.08043*, 2019.

[68] F. Glover, G. Kochenberger, R. Hennig, and Y. Du, "Quantum bridge analytics i: A tutorial on formulating and using qubo models," *Annals of Operations Research*, vol. 314, no. 1, pp. 141–183, 2022.

[69] P. Date, R. Patton, C. Schuman, and T. Potok, "Efficiently embedding qubo problems on adiabatic quantum computers," *Quantum Information Processing*, vol. 18, pp. 1–31, 2019.

[70] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.

[71] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[72] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical science*, vol. 8, no. 1, pp. 10–15, 1993.

[73] F. Glover and M. Laguna, *Tabu search*. Springer, 1998.

[74] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[75] T. A. El-Mihoub, A. A. Hopgood, L. Nolle, and A. Battersby, "Hybrid genetic algorithms: A review.," *Eng. Lett.*, vol. 13, no. 2, pp. 124–137, 2006.

[76] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of global optimization*, vol. 6, pp. 109–133, 1995.

[77] C. Lemaréchal, "Lagrangian relaxation," *Computational combinatorial optimization: optimal or provably near-optimal solutions*, pp. 112–156, 2001.

[78] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[79] J. E. Kelley Jr, "The cutting-plane method for solving convex programs," *Journal of the society for Industrial and Applied Mathematics*, vol. 8, no. 4, pp. 703–712, 1960.

[80] H. Yasuoka, "Computational complexity of quadratic unconstrained binary optimization," *arXiv preprint arXiv:2109.10048*, 2021.

[81] S. Dash, "A note on qubo instances defined on chimera graphs," *arXiv preprint arXiv:1306.1202*, 2013.

[82] H. Xu, H. Ushijima-Mwesigwa, and I. Ghosh, "Scaling vehicle routing problem solvers with qubo-based specialized hardware," in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, IEEE, 2022, pp. 381–386.

[83] D. Pastorello and E. Blanzieri, "Quantum annealing learning search for solving qubo problems," *Quantum Information Processing*, vol. 18, pp. 1–17, 2019.

[84] H. Kagawa, Y. Ito, K. Nakano, *et al.*, "Fully-pipelined architecture for simulated annealing-based qubo solver on the fpga," in *2020 Eighth International Symposium on Computing and Networking (CANDAR)*, IEEE, 2020, pp. 39–48.

[85] R. N. Liang, E. A. Anacleto, and C. N. Meneses, "Data structures for speeding up tabu search when solving sparse quadratic unconstrained binary optimization problems," *Journal of Heuristics*, vol. 28, no. 4, pp. 433–479, 2022.

[86] K. Nakano, D. Takafuji, Y. Ito, *et al.*, "Diverse adaptive bulk search: A framework for solving qubo problems on multiple gpus," in *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, 2023, pp. 314–325.

[87] J. Supasil, P. Pathumsoot, and S. Suwanna, "Simulation of implementable quantum-assisted genetic algorithm," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1719, 2021, p. 012102.

[88] E. Gabbassov, G. Rosenberg, and A. Scherer, "Quantum optimization: Lagrangian dual versus qubo in solving constrained problems," *arXiv preprint arXiv:2310.04542*, 2023.

[89] T. Zaborniak, J. Giraldo, H. Müller, H. Jabbari, and U. Stege, "A qubo model of the rna folding problem optimized by variational hybrid quantum annealing," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, IEEE, 2022, pp. 174–185.

[90] D. Rehfeldt, T. Koch, and Y. Shinano, "Faster exact solution of sparse maxcut and qubo problems," *Mathematical Programming Computation*, pp. 1–26, 2023.

[91] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2023. [Online]. Available: https://www.gurobi.com.

[92] C. Bliek1ú, P. Bonami, and A. Lodi, "Solving mixed-integer quadratic programming problems with ibm-cplex: A progress report," in *Proceedings of the twenty-sixth RAMP symposium*, 2014, pp. 16–17.

[93] B. Tasseff, T. Albash, Z. Morrell, *et al.*, "On the emerging potential of quantum annealing hardware for combinatorial optimization," *arXiv preprint arXiv:2210.04291*, 2022.

[94] H. N. Djidjev, G. Chapuis, G. Hahn, and G. Rizk, "Efficient combinatorial optimization using quantum annealing," *arXiv preprint arXiv:1801.08653*, 2018.

[95] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, "Perspectives of quantum annealing: Methods and implementations," *Reports on Progress in Physics*, vol. 83, no. 5, p. 054 401, 2020.

[96] Y.-T. Kao, J.-L. Liao, and H.-C. Hsu, "Solving combinatorial optimization problems on fujitsu digital annealer," *arXiv preprint arXiv:2311.05196*, 2023.

[97]   D. Willsch, M. Willsch, C. D. Gonzalez Calaza, *et al.*, "Benchmarking advantage and d-wave 2000q quantum annealers with exact cover problems," *Quantum Information Processing*, vol. 21, no. 4, p. 141, 2022.

[98]   E. Crosson, E. Farhi, C. Y.-Y. Lin, H.-H. Lin, and P. Shor, "Different strategies for optimization using the quantum adiabatic algorithm," *arXiv preprint arXiv:1401.7320*, 2014.

[99]   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)*, IEEE, 2019, pp. 138–142.

[100]   A. Peruzzo, J. McClean, P. Shadbolt, *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, p. 4213, 2014.

[101]   M. Cerezo, A. Arrasmith, R. Babbush, *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[102]   J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023 023, 2016.

[103]   J. Tilly, H. Chen, S. Cao, *et al.*, "The variational quantum eigensolver: A review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022.

[104]   A. B. Magann, K. M. Rudinger, M. D. Grace, and M. Sarovar, "Feedback-based quantum optimization," *Physical Review Letters*, vol. 129, no. 25, p. 250 502, 2022.

[105]   A. Kandala, A. Mezzacapo, K. Temme, *et al.*, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *nature*, vol. 549, no. 7671, pp. 242–246, 2017.

[106] J. S. Otterbach, R. Manenti, N. Alidoust, *et al.*, "Unsupervised machine learning on a hybrid quantum computer," *arXiv preprint arXiv:1712.05771*, 2017.

[107] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, p. 4812, 2018.

[108] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, "General parameter-shift rules for quantum gradients," *Quantum*, vol. 6, p. 677, 2022.

[109] J. K. Lenstra and A. R. Kan, "Some simple applications of the travelling salesman problem," *Journal of the Operational Research Society*, vol. 26, no. 4, pp. 717–733, 1975.

[110] B. Gavish and S. C. Graves, *The travelling salesman problem and related problems*, 1978.

[111] M. M. Solomon and J. Desrosiers, "Survey paper—time window constrained routing and scheduling problems," *Transportation science*, vol. 22, no. 1, pp. 1–13, 1988.

[112] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations research*, vol. 35, no. 2, pp. 254–265, 1987.

[113] O. Bräysy and M. Gendreau, "Vehicle routing problem with time windows, part i: Route construction and local search algorithms," *Transportation science*, vol. 39, no. 1, pp. 104–118, 2005.

[114] I. Q. team. Retrieved from https://quantum-computing.ibm.com, *Ibmq*, 2020.

[115] Amazon Web Services, *Amazon Braket*, 2020. [Online]. Available: https://aws.amazon.com/braket/.

[116] Google, *Google Cirq*, 2023. [Online]. Available: 10.5281/zenodo.4062499.

[117]   Microsoft Azure Quantum, *Microsoft Azure Quantum*, 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/quantum/.

[118]   I. T. I. Q. C. https://ionq.com/.

[119]   S. Bravyi, O. Dial, J. M. Gambetta, D. Gil, and Z. Nazario, "The future of quantum computing with superconducting qubits," *Journal of Applied Physics*, vol. 132, no. 16, 2022.

[120]   S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Demonstration of a small programmable quantum computer with atomic qubits," *Nature*, vol. 536, no. 7614, pp. 63–66, 2016.

[121]   Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, Ieee, 2018, pp. 1–2.

[122]   G. E. Crooks, "Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition," *arXiv preprint arXiv:1905.13311*, 2019.

[123]   V. Bergholm, J. Izaac, M. Schuld, *et al.*, *Pennylane: Automatic differentiation of hybrid quantum-classical computations*, 2022. arXiv: 1811.04968 [quant-ph].

[124]   Qiskit contributors, *Qiskit: An open-source framework for quantum computing*, 2023. DOI: 10.5281/zenodo.2573505.

[125]   O. Khaled, M. Minoux, V. Mousseau, S. Michel, and X. Ceugniet, "A compact optimization model for the tail assignment problem," *European Journal of Operational Research*, vol. 264, no. 2, pp. 548–557, 2018.

[126]   A. Parmentier and F. Meunier, "Aircraft routing and crew pairing: Updated algorithms at air france," *Omega*, vol. 93, p. 102 073, 2020.

[127] Y. Chai, L. Funcke, T. Hartung, *et al.*, "Towards finding an optimal flight gate assignment on a digital quantum computer," *arXiv preprint arXiv:2302.11595*, 2023.

[128] A. Ajagekar, K. Al Hamoud, and F. You, "Hybrid classical-quantum optimization techniques for solving mixed-integer programming problems in production scheduling," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–16, 2022.

[129] L. Burgholzer, R. Raymond, and R. Wille, "Verifying results of the ibm qiskit quantum circuit compilation flow," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, IEEE, 2020, pp. 356–365.