Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Engineering

Quantum Optimisation for Energy-Efficient Railway Trajectory Design

Cillian Smith

Supervisor: Prof. Biswajit Basu

April 21, 2024

A dissertation submitted in partial fulfilment
of the requirements for the degree of
MAI (Computer and Electronic Engineering)

Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

I consent / do not consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

I agree that this thesis will not be publicly available, but will be available to TCD staff and students in the University's open access institutional repository on the Trinity domain only, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Signed: Cillian Smith Date: 21-04-24

Abstract

Quantum computing and its potential applications have garnered significant attention over the last number of years for its potential to offer advantage in over classical computation. One area of interest for its application is in the field of optimisation. This thesis investigates the feasibility and application of quantum algorithms to optimise railway trajectories for increased energy efficiency. Given the complex and highly nonlinear dynamics of railway systems, formulating these optimisation problems in the framework of quantum computation presents significant challenges.

To address this problem, a novel formulation is proposed, based on the Quadratic Unconstrained Binary Optimisation framework, which is compatible with quantum optimisation techniques. Mathematical models were developed and implemented on two quantum computing platforms: IBM's gate-based quantum computers using the Quantum Approximate Optimisation Algorithm (QAOA), implemented with Qiskit, and D-Wave's quantum annealers using Ocean SDK Quantum annealers are built for solving QUBO problems and therefore, natively support QUBO optimisation.

The results demonstrated that the quantum algorithms could effectively minimise the energy consumption by optimising the railway trajectories. Problems with a low number of binary variables matched the best-known classical solutions. However, as the number of binary variables increases there was some deviation from the optimal value. A comparative analysis was performed for both platforms, with an emphasis on aspects of practical application. A remarkable result obtained in this research is that the quantum annealing based algorithm can provide which is 50% lower in energy solutions as compared to a state-of-the-art classical simulated annealing solution, and that too obtained in a runtime under 3s whereas the classical simulated annealing required over 1030s, confirming a super-polynomial speed up by the quantum algorithm.

The formulation was expanded to optimise a railway network using a travelling salesman optimisation, to find the shortest Hamiltonian cycle through a series of vertices, where the inter-station energies were determined using the newly developed model for energy consumption optimisation. The thesis suggests future work should focus on increasing applicability of the models developed, including external forces such as aerodynamic drag or friction and track gradients, as well as other potential formulations utilising a force-based decision variable.

Lay Abstract

In recent years, quantum computing has emerged as a revolutionary technology with the potential to solve problems faster and more efficiently than traditional computers, especially in areas where calculations are complex and extensive. Quantum computers operate in a fundamentally different way to traditional computers, leveraging quantum mechanics to solve problems which traditional computers cannot in a reasonable amount of time.

In recent years, with the growing switch from fossil fuel to renewable energy sources, there is a growing need for energy efficient sustainable transport. This thesis tackles the problem of creating a model to optimise the energy usage of railway operation, which can be run on a quantum computer. This model was developed for two different quantum computing platforms: IBM's universal quantum computers, and D-Wave's quantum annealers. Each platform has different strengths, and both showed potential for the use of quantum computers to reduce energy consumption. As for simpler problems with less decision variables, the quantum approach matched the best solutions found with classical computing methods. But for more complex ones, the quantum algorithms confirmed a super-polynomial speed up in time.

The problem was also expanded and applied to network wide optimisations, where quantum computing was used to calculate the energy efficient route through all stations.

Acknowledgements

I would like to extend my deepest gratitude to my supervisor, Dr. Biswajit Basu, whose invaluable guidance and expertise were instrumental throughout the course of this project. His insights, patience, and support have been crucial in shaping this thesis. I am also thankful to the team at Alstom for providing the necessary resources and environment to conduct this study. Their collaboration and expert advice have been greatly appreciated.

I am profoundly grateful for my wonderful family, whose unconditional love and support have always encouraged and inspired me. Without them, I would not be where I am today.

A special thanks and love goes to Rebecca and her family, for being supportive and patience, and very welcome distractions throughout the final months of this thesis.

Finally, my sincerest thanks to my friends and those in the Parsons. Their unwavering commitment to games, adventures, and general distractions has been invaluable in maintaining my sanity throughout this year.

Contents

List of Figures

List of Tables

List of Listings

Nomenclature

AQC	**A**diabatic **Q**uantum **C**omputing
QA	**Q**uantum **A**nnealing
QAOA	**Q**uantum **A**pproximate **O**ptimisation **A**lgorithm
QC	**Q**uantum **C**omputing
QPU	**Q**uantum **P**rocessing **U**nit
QUBO	**Q**uadratic **U**nconstrained **B**inary **O**ptimization
TSP	**T**raveling **S**alesman **P**roblem
VQA	**V**ariational **Q**uantum **A**lgorithm
CO	**CO**asting phase in velocity profiles
CR	**CR**using phase in velocity profiles
MA	**M**aximum **A**cceleration phase in velocity profiles
MB	**M**aximum **B**raking phase in velocity profiles
$\mathbf{x}$	Binary decision vector $\{0,1\}^N$
$\mathbf{x}^*$	Optimal Binary Decision Vector

1 Introduction

1.1 Motivation

The growing need for energy efficiency is becoming clearer as we move away from unsustainable energy sources, such as fossil fuels, to renewable sources. As we transition to renewable energy sources like solar and wind, which can be intermittent and variable, it becomes increasingly important to optimise energy usage especially with sustainable transport, where small energy savings can accumulate significantly. In the last two decades, CO2 emissions from diesel rail have increased by an average of 0.6% annually. These emissions need to decrease by about 5% annually to be on track for Net-Zero Emissions by 2050 [3], a target proposed for global energy consumption to achieve net-zero emissions.

This thesis aims to examine the feasibility of using quantum algorithms for optimisation of railway operation for more efficient energy consumption. The optimisation is directly linked to energy consumption and environmental impact. With the pressing need for sustainable transport, improving energy efficiency could prove beneficial, having a positive societal impact through reduced energy consumption and emissions. Quantum algorithms are ideal for tackling complex optimisation problems that are intractable by classical computing techniques. The motivation of this project is to use this quantum advantage to improve energy efficiency. By doing so, we aim to offer insights into the practical use of quantum computing in enhancing energy efficiency in the transport sector, providing a clearer path to sustainability.

1.2 Problem Statement

The problem of energy efficient trajectory planning is an active area of research in the field of railway systems. It involves determining the optimal velocity profiles for trains to minimise energy consumption, while adhering to safety regulations and schedules.

Researchers have historically applied dynamic programming for this problem [4, 5], with more modern research applying machine learning methods or more complex algorithms, such as genetic algorithms [6]. From the literature review, it was found that little to no work has been done on the application of quantum computing for trajectory optimisation. Quantum

computing is an emerging field as the applications are still being explored. It has broad possibility for applications in the both scientific and commercial fields, it is often used in combinatorial optimisation problems. The problem can be considered as casting a integer programming problem to a binary framework which is suitable for quantum optimisation.

1.3 Technical overview

Quantum computing is a nascent field which has garnered significant attention in recent years due to its potential to revolutionise various fields. One field in which it has particular potential is in the field of optimisation problems. As these kinds of problems arise everywhere, improvements over state-of-the-art classical solutions could have a substantial impact. This section offers an a high-level overview of quantum optimisation, with a practical view on implementation.

Quantum computing operates in a fundamentally different way to classical computing, where based on the principles of quantum mechanics the fundamental unit is the qubit. Much like classical a bit, it can exist as either 1 or 0. Unlike a bit, it can also be in both states simultaneously through the principle of superposition. There are several other quantum mechanical effects that can be leveraged to enhance the power of quantum computing, potentially leading to a 'quantum advantage' or speed-up, where a problem can be performed faster on a quantum system than on its classical counterpart. One of the most famous examples would be Shor's algorithm [7], for prime factorisation, theoretically run in polynomial time on a quantum computer, while only a sub-exponential time on classical hardware.

Theoretical advancement in quantum computing includes algorithms as Grover's search, or Quantum Phase Estimation (QPE), which promise speed-up for their respective computational tasks. However, these require fault tolerant hardware due to the resulting circuit size and depth. Algorithms run on current hardware are known as Variational Quantum Algorithms [8]. The mechanisms and instances of these algorithms will be explored in more detail in Sec 2.3, the general mechanism decomposes the optimisation problem into several smaller problems to be solved classically but their expectation evaluated by quantum means.

The value of quantum computing comes from the potential of quantum advantage in computation. Over the last number of years there has been a myriad of claims of quantum advantage by researchers, only to be disproved [9,10]. This is not to say that quantum computing cannot offer any potential speedup, with the evidence for a potential exponential speed-ups even on today's NISQ hardware [11].

Generally, optimisation problems which can be solved on quantum hardware are restricted to certain classes of problems, primarily being discrete optimisation, where we search for a certain configuration of elements which return the lowest cost. These can be applied to

a wide range of field and applications, such as portfolio optimisation [12], credit scoring [13], scheduling [14, 15], Quantum Machine Learning (QML) [16, 17], and other potential applications [13, 18, 19]

1.4 Thesis Objectives

The primary goal of this thesis is to explore the application of quantum computing for energy optimisation in a railway system, focusing on the feasibility of implementing the problem effectively on quantum hardware.

Three objectives were initially proposed at the beginning of the project:

- **Objective One:** To develop a quantum algorithm/formulation which can be executed on quantum hardware to achieve optimal or near-optimal solutions for small scale problems.
- **Objective Two:** Develop and execute the algorithm on multiple platforms, specifically IBM's Qiskit and D-Waves Leap. Perform a comparative analysis of their performance, and understand their respective advantages and limitations within the field of quantum optimisation.
- **Objective Three:** Investigate the role of entanglement in optimisation problems, this objective was to assess how entanglement can be utilised to improve efficiency and effectiveness of optimisation problems.

All of these objectives were accomplished throughout the course of the project, demonstrating the feasibility for the use of quantum computing in the energy efficient railway optimisation problem.

1.5 Organisation of thesis

The rest of the paper is organised as follows:

Chapter 2 is a literature review examining the current methodologies in classical optimisation. It provides an introduction to quantum information, with an overview on quantum optimisation on the current quantum hardware. Following the review, the application of quantum optimisation in different fields are discussed.

Chapter 3 outlines the development of novel mathematical models that adapt railway trajectory optimisation problems for quantum computing. The chapter details the conversion from classical models to quantum-compatible formulations, highlighting the necessary assumptions and constraints.

Chapter 4 describes the implementation of quantum models using IBM's Qiskit and D-Wave's Leap platforms. It discusses practical aspects, including setup, challenges encountered, and specific implementation details.

Chapter 5 analyses the outcomes of the quantum implementations and compares these results with those obtained from classical optimization methods. It evaluates the practicality and efficiency of quantum solutions and discusses their potential impact with a comparison of the two quantum computing platforms.

Chapter 6 applies the quantum models developed to real-world scenarios, focusing on optimizing railway trajectories. It discusses the implications of these models on actual railway network operations.

Chapter 7 concludes the thesis by summarising key findings and challenges encountered in applying quantum computing to railway trajectory optimisation. As well as outlining potential avenues for future work.

2 Literature Review

In this chapter, a structured overview of energy-efficient railway trajectory control is presented. Beginning by examining early research in the field, advancement in the field of research by the application of modern techniques such as deep-reinforcement learning, or techniques which leverage growth in computational power to allow for real-time optimisation is reviewed. Following this, quantum information and computation is explained, leading into the examination of the nascent field of quantum computing for optimisation. Comparing the different quantum computation paradigms available today, the review concludes with the investigation of studies in different fields, to understand how they might aid in the creation of a quantum formulation for the energy efficient railway trajectory control.

2.1 Classical Optimisation

Theory and practical applications of energy efficient optimal train control has been an area of study for many years, since one of the first studies of by Ichiwaka, in 1968 [20] in which he performed the first research into the "economisation of train operation".

Modern research tends to look at the Driver Assistance System (DAS), which serves as an aid to the driver to reduce energy consumption and improve punctuality. While there are other strategies available, some of which are reviewed and compared in detail by Scheepmaker in [4], this thesis focuses on Energy-Efficient Train control (EETC). This method involves identifying and applying the optimal coasting points and cruising speed. The objective is to minimise the total traction energy, while considering a timetable constraint.

Much of the research into energy-efficiency has been focused around EETC [21, 22]. This has involved finding optimal driving strategies to minimise energy consumption. As this research is largely based on optimal control theory, particularly Pontryagin's Maximum principle (PMP) [23], it has lead to optimal driving regimes involving maximum acceleration (MA), cruising (CR), coasting (CO), and maximum braking (MB). A visualisation of the basic scheme of EETC can be seen in Figure 2.1.

These methods developed using PMP have been foundational in optimal control theory. While

the solutions generated have been proved to be mathematically optimal for fixed speed limits and level tracks by Howlett [24], their application may be limited considering more complex scenarios, such as variable gradients and other external forces. The following discussion will explore modern methods of optimising railway control. These methods offer more flexibility in formulation and can handle more complex problems. The progression of optimisation techniques has introduced more computationally advanced and sophisticated methods, which are more dynamic, flexible, and capable of handling multi-objective optimisations.

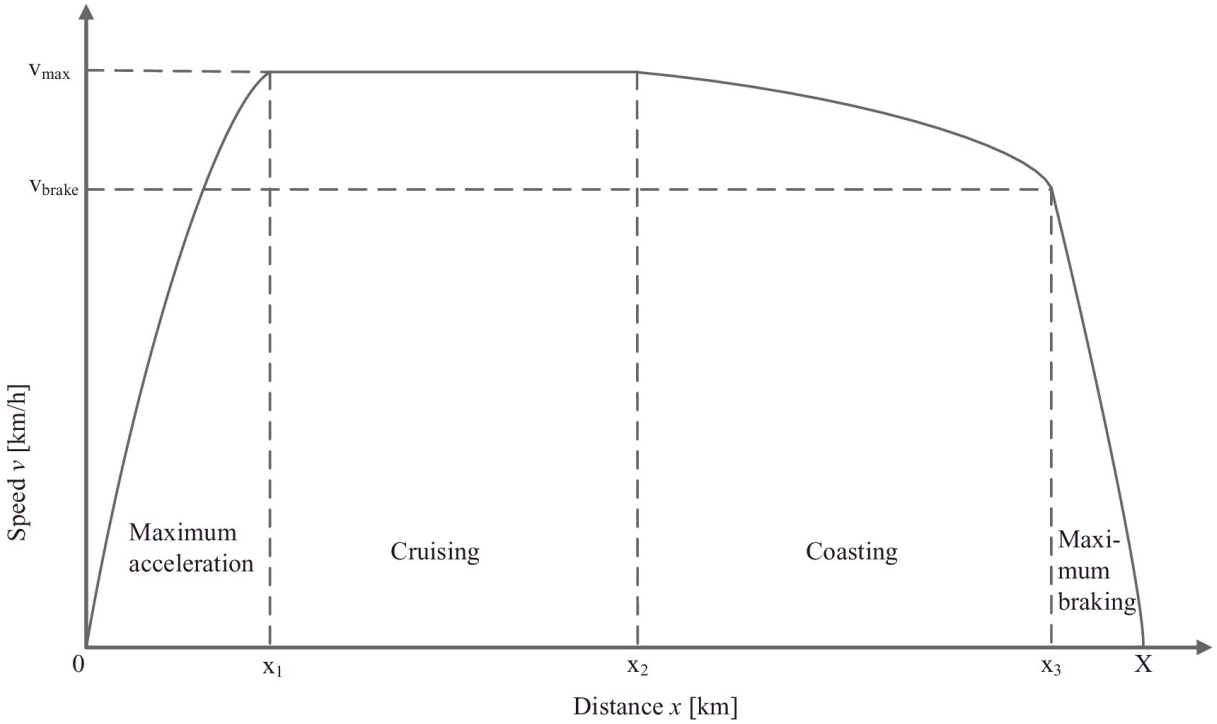


Figure 2.1: EETC profile

2.1.1 Mixed-Integer Linear Programming

Cao et al. [25] aimed to optimise the trajectory of high-speed trains by balancing energy consumption with riding comfort. They extended the work completed by Wang et. al [26] using similar train and MILP models, while additionally considering factors of neutral zones where no catenary power is available, and tunnel resistance. The problem is modelled as a Mixed Integer Linear Programming (MILP) problem, a mathematical approach designed to handle optimisation problems with discrete and continuous variables.

In the mentioned paper, the authors apply MILP to a discrete model of the train's trajectory, considering the train's kinetic energy in place of velocity, as this approach eliminates non-linearities in the model. Additional non-linearities are removed using piece-wise affine (PWA) approximation functions.

The paper presents several key findings from its case studies on the Beijing-Shanghai high-

speed line. Firstl of all, the MILP-based optimisation algorithm achieved a 10.42% reduction in energy consumption compared to standard train operation. When neutral zones were factored into the model, there was a slight increase in energy consumption (about 1.2%). Prioritising riding comfort resulted in a significant increase in energy consumption (ranging from 2.31% to 23.51%). The simulation also showed that tunnel resistance could lead to a 19% increase in energy use. Additionally, optimising the trajectory in multiple segments (rather than a single segment) saved about 1.1% of energy.

This MILP formulation can be solved efficiently on commercial or non-commercial solvers such as CPLEX [27], CBC, or GLPK. Overall, this study demonstrates the effectiveness of MILP, handling both discrete particularly binary and continuous variables. However, MILP can be computationally intensive for large scale problems and can quickly become intractable. Quantum computing could offer potential to solve these intractable problems where classical solutions fail.

2.1.2 Deep Q-Learning

Wang et al. presented the use of Deep Q-learning (DQN) [28], a reinforcement learning technique to enhance energy efficiency for train operation. However, this approach does not depend on pre-existing knowledge of train dynamics or fixed velocity profiles.

The authors formulated the train control problem as a Markov Decision Process (MDP) for the use of DQN. With this method the authors encode the objective function using a reward, granted based on the agent's performance. The agent learns to perform certain actions based on the reward function. It should be noted that the quality of the learnt solution is highly dependant on the reward function. The actions were modelled as discrete 'notches', representing traction forces as values of acceleration (A), breaking (B), and coasting (0), expressed as

$$A = \{-B, ..., 0, ..., P\}.$$

The authors outlined the simulated results, which revealed that a 10% increase in travel time can lead to a 10-15% decrease in energy consumption.

DQN and reinforcement learning is a effective when the dynamics of the system are not known beforehand, making it particularly flexible. However, it requires a significant amount of data for model training and simulation. As well as time demands, a significant amount of computational resources and time for training are required. As mentioned before, the quality of the solution is sensitive to the design of the reward function, making it time consuming to tune.

2.1.3 Simulated Annealing

In their research, Rocha et al. propose the application of Simulated Annealing (SA) for real-time energy optimisation [29]. The algorithm is capable of using optimal driving regimes discussed previously, and addressing multiple constraints such as comfort, speed limits, gradients, and scheduling. Due to its relatively low computational runtime (5-20 seconds depending on problem size), the authors advocate for the application of SA in online systems, highlighting its real-time viability.

Simulated Annealing, a meta-heuristic technique, is designed to approximate a global solution in a large search spaces. The inspiration for SA algorithms lies in natural processes, specifically the metallurgical technique of annealing. This process involves heating a metal to a high temperature and then allowing it to cool gradually, facilitating the formation of a stable crystal structure. This is analogous to how SA searches a solution space for function minima. It is often used in combinatorial optimisation problems.

The algorithm employs a probabilistic solution selection criterion to avoid local optima. Superior solutions are accepted, while inferior ones may be accepted based on the Boltzmann distribution, as outlined in Eq (2.1):

$$\exp - \left[\frac{O_{i+1} - O_i}{T_{i+1}} \right] , \tag{2.1}$$

where O_i is the current solution, O_{i+1} is the proposed new solution; and T_{i+1} is determined by some annealing schedule.

Upon testing this algorithm across four different line sections, the results demonstrated a significant reduction in energy consumption for the generated Optimal Speed Profiles (OSPs). The reductions recorded were 14.67%, 12.06%, 12.55%, and 2.12% for each of the four cases respectively.

Simulated annealing is powerful as it can avoid local minima by it's probabilistic solution criterion. However, there are many parameters to tune to obtain optimal solutions, and the solution quality and runtime are highly dependant on the annealing schedule which controls the rate of temperature reduction. Quantum computing, specifically quantum annealing, which will be discussed in more detail in the next section, could offer a better alternative. As it operates in a quantum physical environment, it can utilise quantum tunnelling to navigate the solution space efficiently.

2.2 Quantum Computing

The following section outlines the fundamentals of quantum information and computing. Qubits are the basic unit of computation for quantum computers. Similar to a classical bit, it represents a two-state system, consisting of $|0\rangle$ and $|1\rangle$. Unlike a classical bit, a qubit can exist in a superposition of its two states $|0\rangle$ and $|1\rangle$. In general, a quantum state $|\psi\rangle$ can be written as a linear combination of its basis states;

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \text{ with } |\alpha|^2 + |\beta|^2 = 1\,, \tag{2.2}$$

where α and β are complex numbers representing the respective probability amplitudes of $|0\rangle$ and $|1\rangle$, and the squared magnitude $|\alpha|^2$ is the probability of $|\psi\rangle$ collapsing to state $|0\rangle$ upon being measured.

Quantum computing leverages unique quantum mechanical phenomena, including superposition, entanglement, and interference, to perform computations in a manner which is fundamentally different from classical computers. This difference in the processing of information leads to potential advantages in solving certain types of problems more efficiently. Superposition refers to a quantum state that can be described as a probabilistic combination of basis states, where the quantum system simultaneously exists in multiple states. The principle of superposition allows quantum computers to compare vast amounts of probabilities simultaneously, creating massive parallelism. Entanglement is a phenomena in which qubits become interconnected in such a way that their states depend on each other, where the state of the entire system cannot be described independently of the other state. A fundamental example of this entangled state is the Bell state or EPR state:

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\,.$$

Interference is the use of constructive and destructive interference patterns, a practical example is Grover's Algorithm, in which quantum interference is used to amplify the probability of correct answers and diminish the probability of wrong ones.

Quantum mechanics drives development in quantum computing, providing the basis for practical quantum algorithms. These include Shor's algorithm for factoring large numbers efficiently [7], and Grover's algorithm for searching unsorted databases faster than classical counterparts [30].

The circuit model is a prevalent framework for quantum computation, similar in structure to classical circuits which are composed of logic gates like AND and OR. In quantum circuits, qubits flow through 'wires' and are manipulated by quantum gates. Key examples include the

Pauli rotational gates (X, Y, Z) that rotate qubits around specific axes, and the Hadamard gate, which puts qubits into a superposition of $|0\rangle$ and $|1\rangle$ [31].

The benefit of quantum computing comes from a problems ability to scale. Generally, the measurement of the difficulty of a problem is handled through complexity theory. It measures the amount of resources to solve problems. The performance difference between classical and quantum computation is asymptotic scaling of complexity [32]. Quantum advantage is achieved through any computation improvement which reduces the complexity of certain problems [33].

2.2.1 Complexity Theory

Complexity theory is crucial in classifying problems and is used extensively in computer science. It classifies problems based on the resources required to solved them. Problems are divided into classes such as P, NP, NP-Hard, and NP-Complete, where they represent sets of problems, illustrated in Fig 2.2.

- **P (Polynomial Time):** This class of problem can be solved by a deterministic machine in polynomial time. It contains problems which are tractable and easily solvable, such as sorting or computing the greatest common divisor.
- **NP (Non-Deterministic Polynomial Time)**: This class includes problem which can be verified in polynomial time, and solved by a non-deterministic machine in polynomial time.
- **NP-Hard:** These problems are at least as hard as the hardest problem in NP, and they cannot be verified in polynomial time. Generally, problems like the control optimisation problem, where the solution space grows exponentially with input, are NP-Hard problems.
- **NP-Complete:** The NP-Complete class is the intersection of NP and NP-Hard.

The promise of quantum computing lies in its potential to solve certain NP-hard and NP-complete problems more efficiently than classical computers. Not all NP-hard problems are equally applicable to quantum solutions, but those that can be expressed in terms suitable for quantum algorithms might see significant computational speedups.

2.3 Quantum Optimisation

The main distinction between quantum computing and classical computing is the method of processing and storing information. Classical computers use bits, 0 or 1, while quantum computing uses qubits, taking advantage of quantum mechanical phenomena to process in-

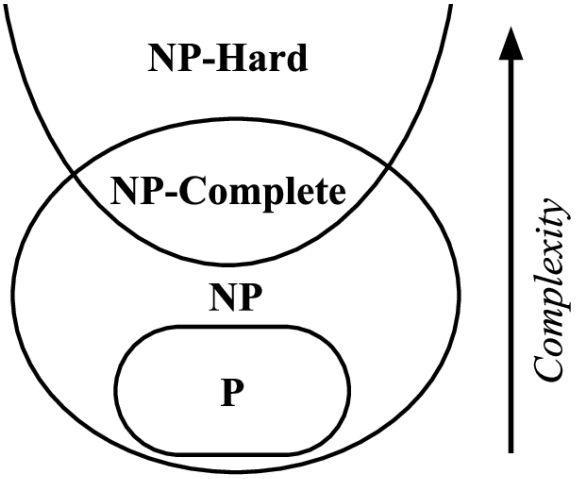


Figure 2.2: Complexity Theory classes

formation more efficiently. However, it requires reformulation of the problem in quantum algebra. [31]

In terms of computation and optimisation, quantum computers have the possibility to process vast amounts of data simultaneously due to these quantum properties. They can explore a large solution space more efficiently, which can lead to finding optimal solutions quicker in certain types of optimisation problems. This capability makes quantum computing particularly promising for complex optimisation challenges, where classical algorithms might struggle or take an impractically long time to find optimal solutions.

2.3.1 Quadratic Unconstrained Binary Optimisation (QUBO)

Before discussing algorithms for optimisation on quantum hardware, Quadratic Unconstrained Optimisation Optimisation problems must be introduced. The QUBO is a key framework for quantum optimisation. The mathematical approach is used to formulate optimisation problems in a way that is compatible with quantum computing algorithms, specifically combinatorial optimisation problems. The QUBO model can be expressed as

$$\textit{Minimising} \quad \mathbf{x}^T Q \mathbf{x} = \sum_{i,j=1}^{n} Q_{ij} x_i x_j \ , \tag{2.3}$$

where $\mathbf{x} \in \mathbb{B}^N$, and Q is a real valued upper-triangular matrix $Q \in \mathbb{R}^{N \times X}$.

Note the need for the formulation to be unconstrained, as well as the need for the decision variables to be binary. As the majority of problems of interest have additional constraints which must be met for feasible solutions, penalties are introduced in place of constraints. The penalties are chosen so that the influence of the original constraints on the solution process can alternatively be achieved natively by the function of the optimiser as it looks for solutions that

avoid incurring the penalties [34]. Tab 2.1 shows a number of common classical constraints with their equivalent penalty terms.

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

Table 2.1: Classical Constraints with their equivalent penalty form, where P is a positive scalar penalty value

Below there is a QUBO example found in [34], which outlines the structure and formulation of the QUBO framework to introduce the concept. The given objective function is defined as:

$$\text{Minimise } y = -5x_1 - 3x_2 - 8x_3 - 6x_4 + 4x_1x_2 + 8x_1x_3 + 2x_2x_3 + 10x_3x_4 \ , \tag{2.4}$$

where variables x_i are binary variables, $x \in \{0, 1\}$. The function to be minimised can be split into two parts. The linear part, with singular binary variables$-5x_1 - 3x_2 - 8x_3 - 6x_4$ and the quadratic part, with quadratic relationships between two binary variables $4x_1x_2 + 8x_1x_3 + 2x_2x_3 + 10x_3x_4$.

The can be re-written as

$$\text{Minimise } y = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix} \begin{bmatrix} -5 & 2 & 4 & 0 \\ 2 & -3 & 1 & 0 \\ 5 & 1 & -8 & 5 \\ 0 & 0 & 5 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \ , \tag{2.5}$$

which can be written succinctly as

$$\text{Minimise } y = x^T Q x \ . \tag{2.6}$$

The solution to the problem Eq (2.4), is

$$y = -11, x_1 = x_4 = 1, x_2 = x_3 = 0 \ .$$

The $N \times N$ QUBO matrix Q can be visualised in a heatmap form, where negative coefficients are coloured cool tones (blue), and positively value coefficients are coloured in warmer tones

(red). An example of this can be seen in Fig 2.3.

The diagonal entries in the matrix represent the linear coefficients associated with each binary variable x_i, in the above example (-5, -3, -8, -6), illustrated by the varying intensities of cool tones in the heatmap. The off diagonal entries represent interactions between different binary variables, which define how variable relate to one another. As an example, we see from the equation that there are no interactions between x_1 and x_4, which is seen in either the top right, or bottom left of the heatmap, and the QUBO is symmetric along the diagonal.

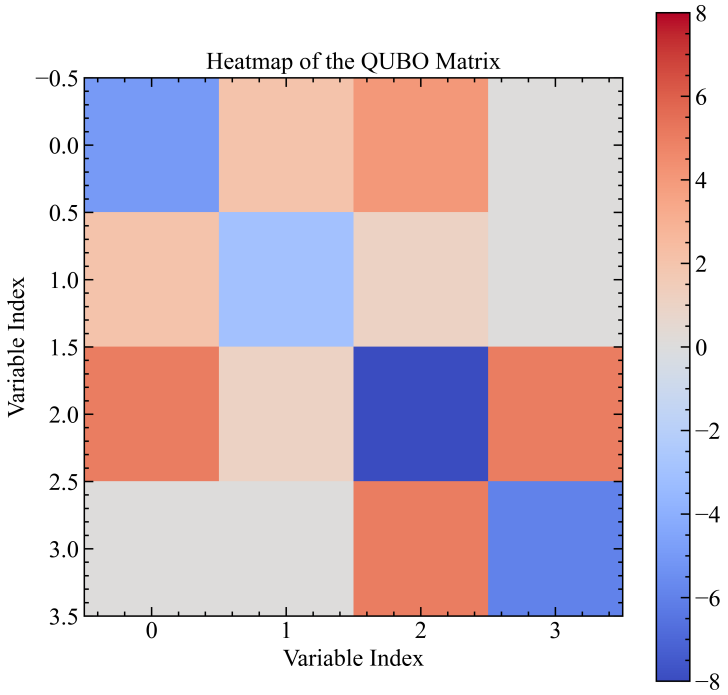


Figure 2.3: Heatmap of QUBO in (2.5)

For an extensive list of problems with their QUBO formulation, see [35].

QUBO and Ising Equivalence

Throughout the thesis, both Ising and QUBO models are frequently mentioned. These models are isomorphic and therefore are equivalent. An Ising model can be transformed to its equivalent QUBO model by applying the transformation $\sigma \to 2x - 1$ to an Ising model. The Ising Hamiltonian is defined as follows:

$$H(\sigma) = \sum_{i,j} J_{ij}\sigma_i\sigma_j + \sum_i h_i\sigma_i \ . \tag{2.7}$$

In Ising formulations, the variable x which represent the binary variable $\{0,1\}^n$ is replaced by spin variables$\{-1,+1\}$.

2.3.2 Quantum Computing Paradigms

Quantum optimisation has focused on Combinatorial Optimisation (CO) problems, which in general is searching for an element in a list of elements which is optimal based on some metric. These problems can be found in supply chain management [36], and manufacturing [14]. The QUBO model is ideal for translating natively binary optimisation problems, such as Max-Cut or Traveling Salesman into a format suitable for quantum processors. This formulation from general problem to QUBO formulation which can be used on a quantum processor is often non-trivial.

Over the past number of years two major paradigms of quantum computing models have emerged, quantum annealing and gate-based quantum computing (also known as universal quantum computing). Currently, noise and error in quantum computing poses a large issue, qubits suffer from a phenomenon called decoherence. As circuits grow deeper we need to research error correcting codes and fault tolerant quantum computers. While fault tolerant quantum machines are still some years away, current hardware is in the era of small, error prone quantum computers, or Noisy Intermediate-Scale Quantum (NISQ) machines as coined by Preskill [37].

2.3.3 Variational Quantum Algorithms (VQA)

Variational Quantum Algorithms are a family of randomised search algorithms which have been at the forefront of research, driven by NISQ era hardware. To be useful for computation, they need to operate on a small number of qubits, and be as shallow as possible to avoid decoherence and error. These are hybrid quantum-classical optimisation technique. The hybrid loop of the of VQA involves a parameterised quantum circuit, and an optimiser which updates the parameters to minimise a cost function based on the outputs of a quantum circuit. VQAs generally have the advantage of being shallow quantum circuits, which makes them ideal to be run on NISQ machines are they are less susceptible to noise [38].

Quantum Approximate Optimisation Algorithm (QAOA)

Quantum Approximate Optimisation Algorithm (QAOA) was first introduced by Farhi et al. [39], as a VQA to find approximate solutions to the Max-Cut problem. In QAOA, the cost function is encoded into a cost Hamiltonian $\hat{H}_c$.

The algorithm operates by alternating between two different Hamiltonians applied to a quantum state. The cost Hamiltonian encodes the problem, and the second is a mixing Hamiltonian that promotes exploration of the solution space to find the optimal bit string $\mathbf{x}$. The interaction between these Hamiltonians under the control of variational parameters, which are optimised through a classical guide function, guides the system towards an approximate

solution leveraging the principles of quantum superposition and entanglement.

The algorithm initialises the system in a superposition state using a set of qubits. Each qubit is prepared in an uniform superposition. This setup represents all possible configurations of the decision variables in the optimisation problem. QAOA is a variational technique, involving a sequence of quantum operations governed by a set of parameters. These operations include:

- **Cost Unitary $U_C(\gamma)$:** Encodes the cost function of the optimisation problem. The parameters γ are variational and are adjusted to optimise the solution.
- **Mixer Unitary $U_m(\beta)$:** Parameterised by β, this unitary is defined by a mixer Hamiltonian facilitates the exploration of the solution space by inducing quantum interference.

These operations are applied iteratively, altering the quantum state in a manner that encodes the probability of various solutions.

The expectation value of the Hamiltonian $\hat{H}_C$ with respect to the ansatz state $|\psi_p(\gamma,\beta)\rangle$, which corresponds to the cost obtained by the quantum algorithm for the underlying problem, is calculated through repeated measurements of the final state in the computational basis:

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

A classical optimisation algorithm is employed to iteratively update the parameters γ and β. The goal of the routine is to find the optimal set of parameters (γ^*,β^*) such that the expectation value $F_p(\gamma,\beta)$ is maximised:

$$(\gamma^*,\beta^*) = \text{arg max}_{\gamma,\beta} F_p(\gamma,\beta) \ .$$

QAOA is an iterative algorithm, with each cycle refining the variational parameters and the resultant quantum state. This iteration continues until the algorithm converges on a near-optimal solution or reaches a set max number of iterations. A visualisation of the QAOA circuit can be seen in Fig. 2.4

IBM Quantum

IBM is at the forefront of gate-base or universal quantum computing. IBM develop an open-source quantum development framework called Qiskit. This enables users to build quantum circuits from primitives and gates, or use provided modules such as circuits or optimisation for fast development. Beyond the software, IBM provides access to quantum processors and simulators through the Quantum cloud platform [40].

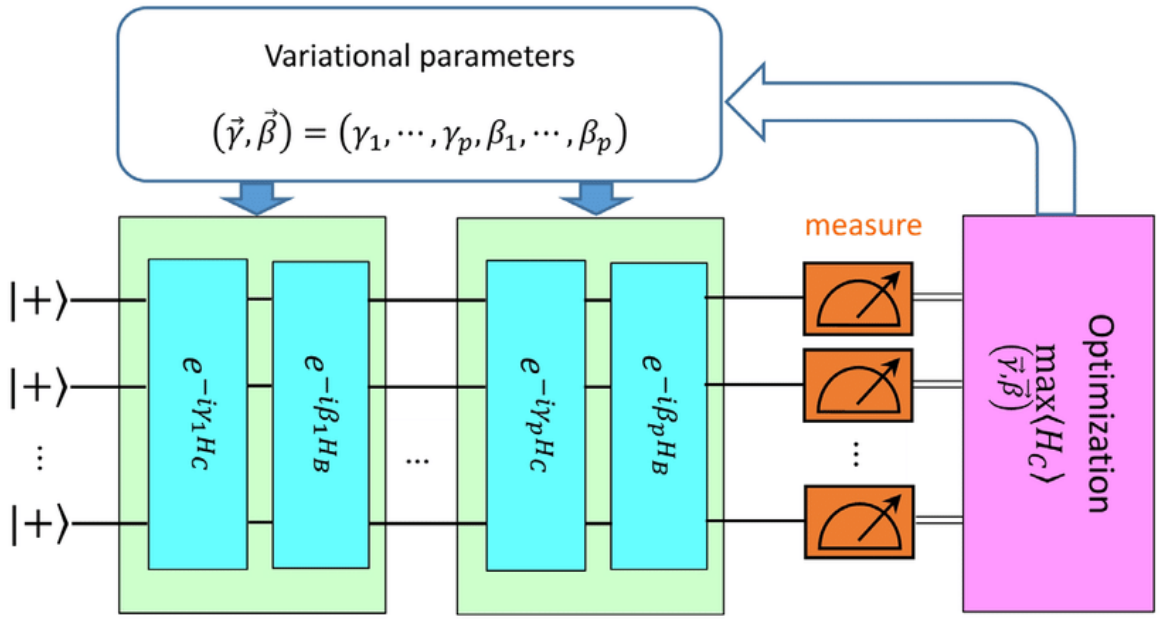


Figure 2.4: Schematic of the hybrid workflow of QAOA with p layers [1]

Draw backs of universal quantum computers and VQA

While gate-based quantum machines and VQA are finding use cases in different fields such as machine learning, and simulation with physics or chemical interactions [41], they are limited by two main barriers; Qubit decoherence and noise, and hardware scale.

Hardware scale has been an issue, limiting the viability of practical application, the number of available qubits has been growing steadily in the last decade. IBM recently announced their newest hardware, Condor, with 1121 qubits. There has been a steady increasing trend, with hardware almost doubling the number of qubits every year, with a 127-qubit chip called Eagle, released in 2021 and a 433-qubit one called Osprey, in 2022 [42].

Qubit decoherence is the other major hurdle in the design of reliable and scalable quantum computers. Extensive research are being done into error correction and higher quality qubits. Notably, IBM also announced a new processor named Heron, with 133 qubits having an qubit error rate 3-times lower than its previous generation of a similar size. Fault tolerant quantum computers (FTQC) are those who can correct errors more quickly than they occur, allowing for reliable computation. As of yet, there are no fault-tolerant quantum computers, and it is unclear when the transition from NISQ to FTQC will occur, or whether it will occur by error correction [43], or by natively fault tolerant hardware. For now, researchers in Harvard and QuEra [44] have demonstrated a quantum computer with 48 logical qubits. A logical qubit is a cluster of physical qubits, with logic qubit error correction applied to them, allowing for more reliable and useful computation.

2.3.4 Adiabatic Quantum Computing (AQC)

AQC operates on the principle of quantum adiabatic evolution, unlike universal computation which is encoded using unitary gates. AQC involves initialising the quantum computer into the ground state of a easy to prepare Hamiltonian H_{init} and evolving to a final Hamiltonian whose ground state encodes the solution to the computational problem:

$$H(t) = (1-t)H_{init} + tH_{final} \quad \text{where } 0 \leq t \leq 1 \ . \tag{2.8}$$

By the adiabatic theorem [45], if we evolve the system slowly enough *i.e.*, adiabatically, it will remain in the ground state of $H(t)$, thus at time t the system will be in the ground state of H_{final} which encodes the solution for the computation [46]. It has been shown that AQC is equivalent to gate-based computation, as any quantum-circuit can be represented as a Hamiltonian with in polynomial time [47].

Quantum Annealing

Quantum Annealing first proposed in 1988 by Apolloni et. al, originally inspired by the simulated annealing algorithm, and can be seen as a subclass of AQC. QA is not universal, and is a relaxation of the AQC model which results in a heuristic variational algorithm. Similarly to AQC, QA is used to find the ground state of a Ising model, which is equivalent to QUBO, and as such QA lends its self well to solving discrete optimisation problems [19].

D-Wave Systems is an industry leader in the development of quantum annealing machines. They released their first commercially available machine in 2011, and have continued to developed largee and more powerful machines with their latest Advantage system having 5000+ qubits [48]. D-Wave have produced several QA machines, capable of solving complex combinatorial optimisation problems.

In these systems, qubits are arranged in a specific graph structure, determining which qubits can directly interact with each other. The most common graph used by D-Wave is the Chimera graph, and more recently, the Pegasus graph, which is found in their Advantage series. Fig 2.5 shows an illustration of these different topologies. Each node in these graphs represents a qubit, and each edge represents a coupler allowing interaction between qubits. The connectivity of a qubit means the number of other qubits it can directly interact with. This connectivity can be expanded with a method called qubits chaining, where multiple qubits are entangled to be in the same state. High-connectivity graphs, such as the Pegasus, provide a denser connection network, allowing more qubits to influence one another during the annealing process. For D-Wave experimental machines, a Zephyr topology contains a connectivity of 20, compared to Pegasus's 16, or previous systems' 6.

Figure 2.5: Graph topologies implemented of D-Wave machines [2]

The Ising model is the preferred formulation for quantum annealer as the problems map directly to the QPU, where the binary variables are mapped to qubits, and the costs are mapped to the couplers between the qubits. On the D-wave quantum annealer, the Hamiltonian is represented as:

$$\mathcal{H}_{ising} = \underbrace{-\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right)}_{\text{Final Hamiltonian}}$$

where $\hat{\sigma}_{x,z}^{(i)}$ are Pauli matrices operating on qubit q_i, and h_i and $J_{i,j}$ are the qubit linear biases and coupling strength respectively. The initial Hamiltonian, also known as the tunnelling Hamiltonian, is a superposition of all qubits of $|0\rangle, |1\rangle$. The final Hamiltonian, encodes the solution to the problem after the annealing process.

Drawback of QA

Quantum annealing is a specialised form of quantum computing, designed for solving discrete optimisation problems. It has some important limitations to consider. QA is a non-universal form of computing. It is unable to run a wide-variety of quantum algorithms such as Shor's algorithm, as it is not a hill-climbing algorithm.

2.4 Applications

There is a lack of research specifically applying quantum optimisation techniques to railway energy consumption problem, and hence insights can be drawn from applications in other fields. For example, the Quantum Technology and Application Consortium (QUTAC) investigates various quantum use cases, providing a broader perspective on potential applications of quantum technologies [18].

Quantum portfolio optimisation

Portfolio optimisation is a commonly seen optimisation problem in finance. It involves, from a pool of assets, selecting the best configuration and their quantities according to some

objective. This could be a multi-objective function considering risk, profit, etc. It commonly focuses on building an efficient portfolio, which is one that maximised expected return for a given amount of risk.

The authors of [12] suggest an method of reducing the typically Quadratic Programming (QP) problems of portfolio optimisation to the more general form of Second Order Cone Programs (SOCP) and the use of a quantum interior point method [49]. This unconventional reduction to SOCP from QUBO may provide insight to solving other problems.

There are other combinatoric problems which are common in banking and finance [13], such as credit scoring and derivative development.

Traffic Flow optimisation

Volkswagen developed an application for real-time traffic flow optimisation [50]. Designed to minimise congestion across a road network using D-Wave's annealer, Q2000 utilising an iterative hybrid workflow, using classical methods to handle data preprocessing and QUBO formulation. Hybrid solving methods were used, to identify solutions aimed at reducing congestion, followed by classical post processing to redistribute the traffic according to the returned solution. This cycle was repeated until a solution with minimal congestion was found. Congestion was determined by the number of vehicles traversing a road segment on the network simultaneously, expressed as:

$$\text{cost}(s) = \left(\sum_i \sum_j \sum_{s \in S_j} q_{ij}\right)^2 , \quad (2.9) \qquad \sum_{s \in S} \text{cost}(s) + \lambda \sum_i \left(\sum_j q_{ij} - 1\right)^2 . \quad (2.10)$$

Equations (2.9) and (2.10) are the cost and objective QUBO formulations respectively for each car i, with proposed routes j with street segments S_j.

This study used a small sub-sample (418 cars) of the available dataset (10,357). Notably, the QUBO formulation for this sample required 1,254 qubits, exceeding the capacity of the available QA which had 1135 functional qubits. This restriction facilitated the use of a hybrid/quantum tool called `qbsolv` [51].

`qbsolv` is a software tool developed by D-Wave, designed for solving larger QUBOs with higher connectivity than is achievable through directed embedding onto the QPU. The algorithm partitions the input problem into smaller independent queries to the QPU, rendering large intractable problems solvable for quantum solvers. While `qbsolv` has since been deprecated in favour of D-Wave's hybrid solvers, it offers insight into addressing problems with large QUBO formulations.

In terms of runtime performance, the algorithm succeeded in minimising congestion across the network, however, due to the use of shared cloud-based resources, there were significant variations on runtime. The minimum of the recorded run-times was noted, 22 seconds, as

the authors felt it was most representative of the algorithm performance. This result suggests potential applications for real-time use. However, it should be noted that the study was limited in its scope. It excluded other traffic participants, no communication with traffic infrastructure, and congestion minimisation was the only optimisation target.

Quantum Shuttle

The Quantum Shuttle project [52], developed by Volkswagen is an extension of the work done in [50], where they address a number of the limitations from that study, most notably being traffic-aware solutions.

The project was devised to manage the influx of people into Lisbon for the 2019 Web Summit. It comprised of two phases, the first being a comprehensive data analysis prior to the event to establish the required capacity, new routes and scheduling. The second was the development of a custom android app for real-time optimisation of bus routes, using D-Waves QP, for driver navigation. The traffic optimisation utilised the HERE API [53] to generate traffic aware routes to be used in the optimisation.

The optimisation strategy used mirrored the approach used in [50]. The authors outline three approaches which were direct embedding, custom hybrid workflow and D-Waves available hybrid solvers. The hybrid solvers are the successors of the previously discussed `qbsolv` algorithm. As the authors note, it allowed for faster run-times than the custom hybrid approach, as well as having more effective scaling.

3 Quantum Model Formulation for Energy Optimisation

This chapter outlines the mathematical formulations for the proposed new model for railway energy efficiency control developed over the course of the thesis. The research in this project employs a progressive modelling strategy, beginning with a simplified mathematical model designed to provide a baseline reference and facilitate validation of the subsequent models. It then outlines the approach for creating a model in a suitable QUBO framework for implementation in a quantum system, and the steps for integrating different energy models, efficiency considerations, and for introducing breaking mechanisms.

3.1 Classical Formulation

With no existing simple classical formulation of the problem under consideration, a simple system formulation is required to test the quantum formulation and the results.

Consider a railway trajectory where the motion can be described in three distinct phases within the interval $[0, T]$. Let phase I be described by constant acceleration a, let phase II be described by motion at a constant velocity v_{max}, and finally let phase III be described by the the constant braking at $-a'$. Mathematically the velocity can be defined as (see Figure 3.1):

$$v(t) = \begin{cases} at & \text{if } 0 \leq t < t_1 \quad \text{(MA)}, \\ v_{max} & \text{if } t_1 \leq t < t_2 \quad \text{(CR/CO)}, \\ v_{max} - a'(t - t_2) & \text{if } t_2 \leq t \leq T \quad \text{(MB)}. \end{cases} \tag{3.1}$$

The following assumptions are made for the system:

- **Negligible Frictional and Drag Forces:** Drag or friction forces are discarded, as it is assumed that there is no energy consumption in when the acceleration is zero, therefore there is no energy consumption in phase II.

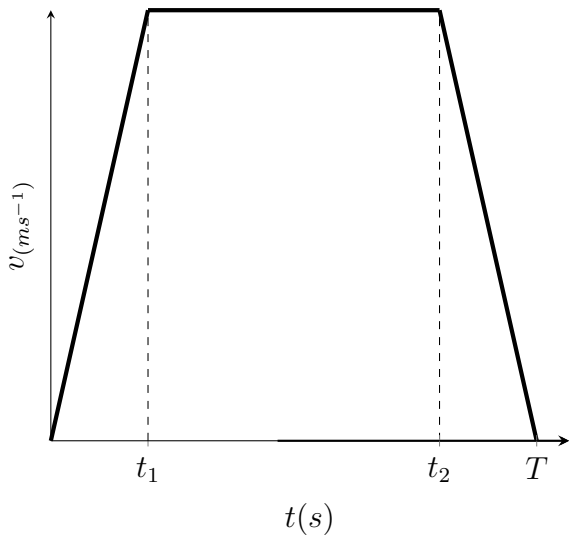


Figure 3.1: Velocity versus Time

- **No regenerative braking or energy expenditure:** Only energy dissipation occurs in phase III, it is assumed there is no energy recovery from regenerative braking, and that no expenditure of energy is required for braking. Therefore, there is no energy consumption in phase III.

The total energy consumption (E) of the system can be calculated as

$$E = E_1 + E_2 + E_3 \tag{3.2}$$

where E_i is the energy of phase i. As state previously, the energy consumed in phase II and phase III is negligible, therefore $E_2 = 0$ and $E_3 = 0$. E_1 only contributes to the energy consumption, which is expressed as:

$$E_1 = \int_0^{t_1} P(t)\, dt \; = \int_0^{t_1} F(t) \cdot v(t)\, dt \;, \tag{3.3}$$

where $P(t)$ and $F(t)$ are the power and force respectively at time t. This model can be solved analytically, by minimising

$$E_1 = \int_0^{t_1} \eta(v) \cdot \frac{at}{t_1} \cdot v_{max}\, dt \quad , \tag{3.4}$$

since mass is constant.

Assuming a constant efficiency $\eta \; = \eta_0$ and defining $a = \frac{v_{max}}{t_1}$, we have

$$E_1 = \eta_0 \frac{(v_{max})^2}{t_1^2} \int_0^{t_1} t\, dt \;,$$

leading to

$$E_1 = \eta_0 \frac{v_{max}}{t_1^2} \frac{t_1^2}{2} = \eta_0 \frac{v_{max}}{2} \,. \tag{3.5}$$

This result matches the result which we expect to see [54],(in Fig 3.2, considering the modes of MA, CR, and MB as described in Sec 2.1).

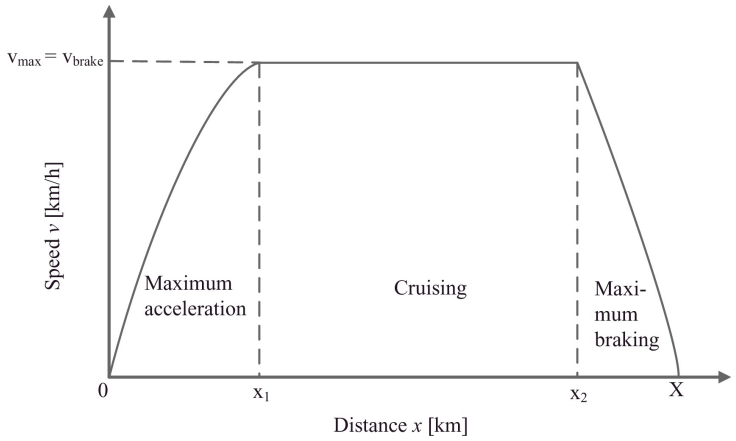


Figure 3.2: Speed profile of an energy-efficient driving strategy without coasting and with switching points between driving regimes at x1 and x2.

3.2 Quantum Formulation

3.2.1 Modeling the decision variables

As discussed previously in Sec 2.3.1, the problem must be cast into a form which is suitable for quantum computing. In this case , the problem is reformulated as a QUBO, using binary decision variables $\{0, 1\}$. In the conversion of the classical form to a quantum form, the time-domain $[0, T]$ must be dicretised to finite intervals which can be used within the QUBO framework. The time-domain is broken into N distinct segments, each represented by a binary variable, denoted as $\mathbf{x} = \{0, 1\}^N$. These variables determine the state of the system in their respective interval. The continuous functions which determine the state of the system must also be discretised; continuous functions $F(t)$, $a(t)$, $v(t)$, and $s(t)$ are mapped to the their discrete counterpart $F[i]$, $a[i]$, $v[i]$, and $s[i]$ respectively, for all $i = 1, \ldots, N$.

A critical decision was to select the most suitable variable to use as the primary decision variable represented in a binary form, as there are a number of options available including a force based decision variable, or similar to [25], where they used kinetic energy. Velocity was chosen as the decision variable, as it has a direct effect on the energy consumed and the dynamics of the system. Velocity can be quantified and manipulated within the binary QUBO framework. The velocity update rule in (3.6), outlines the relationship between the binary decision variables and the velocity, in which the decision variable determines the velocity

increases in the i^{th} time-step, given by

$$\mathbf{x} \in \{0,1\}^N \rightarrow \begin{cases} v_{i+1} = v_i + \Delta v & \text{if } x_i = 1 \quad \text{(MA)} \\ v_{i+1} = v_i & \text{if } x_i = 0 \quad \text{(CO)} \end{cases}, \tag{3.6}$$

where Δv signifies the discrete change in velocity associated with the binary decision x_i. This rule can be written succinctly as $v_{i+1} = v_i + \Delta v \cdot x_i$, and the velocity i^{th} time-step is given by the cumulative sum of the decision variables scaled by the discrete change in velocity Δv; $v_i = \Delta v \sum_{j=0}^{i} x_j$ where v_0 is assumed to be zero, as the train will start its journey from rest.

3.2.2 Energy Models

The energy consumption model defines the cost associated with changing velocity. To optimise the energy efficiency of the model we have to define an energy model that can be used within the QUBO framework and can accurately depict the energy consideration of the system. This is a non-trivial task, as accurately portraying the energy dynamics of this highly non-linear system while adhering to the constraints of the QUBO framework is challenging. This means avoiding higher than quadratic dimensionality, as the QUBO framework only allows up to quadratic relationships between variables *i.e.*, relationships considering two binary variables $x_i x_j$.

Constant energy model

The constant energy model assumes that the energy cost associated with velocity change is constant regardless of the current velocity or state at the i^{th} time-step. This is a simplification which avoid higher order terms and is based on the fact that energy changes are proportional to the changes in kinetic energy, formulated as:

$$E_{i+1} \propto \frac{1}{2}(v_{i+1}^2 - v_i^2) = (v_{i+1} - v_i)(\frac{v_{i+1} + v_i}{2}) = \Delta v_i \cdot v_{avg_i} \ . \tag{3.7}$$

In Eq. (3.7), v_{avg_i} is the average velocity defined as the arithmetic mean of the initial and final velocity over the interval. If efficiency η is proportional to the ratio of the average velocity, v_{avg_i}, to Δv_i upto some constant C, i.e. $\eta = C\frac{v_{avg_i}}{\Delta v_i}$, then the energy expression is refined to

$$E_{i+1} = \frac{\Delta v_i}{C \cdot v_{avg_i}} \Delta v_i \cdot v_{avg_i} = \frac{1}{C} {\Delta v_i}^2 \ . \tag{3.8}$$

Therefore, the minimisation of energy consumption for the entire system, and the objective

function can be defined as:

$$\text{Minimise} \quad E(\mathbf{x}) = \sum_{i=0}^{N} (x_i \cdot \Delta v)^2 \ . \tag{3.9}$$

This effectively means that the only contributing terms to the energy of the system are the linear terms and there are no quadratic interactions between qubits which contribute the energy. This simple models allows for easy formulation of the problem but simplifies the model by ignoring the fact energy contribution changes with velocity.

Quadratic energy model

To develop a more accurate energy model within the QUBO framework, we need to consider not only the incremental changes on velocity but also the average velocity with the time-interval. This approach expands upon the constant energy model by incorporating the dynamics of average velocity between successive time intervals $[i, i+1]$. The energy change between two consecutive states is derived from the kinetic energy difference, calculated as follows:

$$\begin{aligned} E_i &\propto v_{i+1}^2 - v_i^2 \\ &= (v_i + \Delta v \cdot x_i)^2 - v_i^2 \\ &= (v_i^2 + 2\Delta v \cdot x_i v_i + (\Delta v)^2 x_i^2) - v_i^2 \ , \end{aligned} \tag{3.10}$$

where v_i is the velocity in the interval i. This form captures the quadratic relationship between energy consumption and velocity changes. Simplifying the expression, and considering $x_i^2 = x_i$ for binary variables, the energy for a single time-step simplifies to:

$$E_i = 2\Delta v \cdot x_i v_i + (\Delta v)^2 x_i \ . \tag{3.11}$$

The total energy of the system, can be expressed as a function of binary decision variables aligned with the QUBO upper triangular matrix form:

$$\text{Minimise} \quad E = 2\Delta v^2 \sum_{i=1}^{N} \sum_{j=0}^{i-1} x_i x_j + \Delta v^2 \sum_{i=1}^{N} x_i \ .$$

This formulation integrates into quantum optimisation frameworks. However, incorporating more complex system dynamics through higher order terms can be difficult. Although it's

possible to reformulate these using auxiliary variables, this increases complexity of the system.

3.2.3 Constraints

Generally any practical optimisation problems are subject to constraints, and are required for the above model to return any non-trivial solution. As mentioned previously in Sec 2.3.1, QUBOs are inherently unconstrained, however, constraints can be applied through penalty terms. The penalties assign a higher energy cost to the solutions which violate the constraints. In this model there are two critical constraints that need to be considered: a distance constraint and a speed limit constraint.

Distance Constraint: The total distance D that must be travelled between two stations is a constraint. To model the constraint within the QUBO framework, consider d which represents the distance travelled per time-step, assuming the time-intervals are constant. It can be represented as

$$\sum_{i=0}^{N} x_i \cdot [(N-i) \cdot d] = D \ . \tag{3.12}$$

Speed Limit Constraint: These are defined to ensure the model does not violate safety standards. It can be defined as

$$\sum_{i=0}^{N} x_i \cdot \Delta v \leq v_{max} \ . \tag{3.13}$$

This assumes the speed limit is some multiple of Δv. It ensures the sum of the velocity changes must not exceed the maximum allowed velocity v_{max}.

The constraints require some additional assumptions in that ΔT, the time-interval is uniform. For simplicity it is assumed as 1. With this, d is defined as Δv. Also given the absence of braking mechanism encoded within the QUBO, it is assumed there is a period of maximum deceleration from v_{max} at the end of the journey. This both matches the models seen in the literature of MB, and allows to simplify the model by only considering increasing velocity. To incorporate this, the speed limit constraint is modified to

$$\sum_{i=0}^{N} x_i \cdot \Delta v = v_{max} \ ,$$

which both satisfies the speed limit constraint and the need to reach v_{max} prior to the end of the journey.

3.3 Efficiency function

Efficiency is an important consideration when modelling energy consumption in the domain of the railway problem. As mentioned in previous sections, efficiency is affected by a number of different factors considered, such as velocity, heat produced, etc.

Figure 3.3 provided by the railway transport company Alstom shows that the energy efficiency varies with both force applied and velocity. It can be shown by theoretical analysis that efficiency varies from 0.9 to 0.78, with the value maximised at approximately 120km/h applying a force of 10kN.

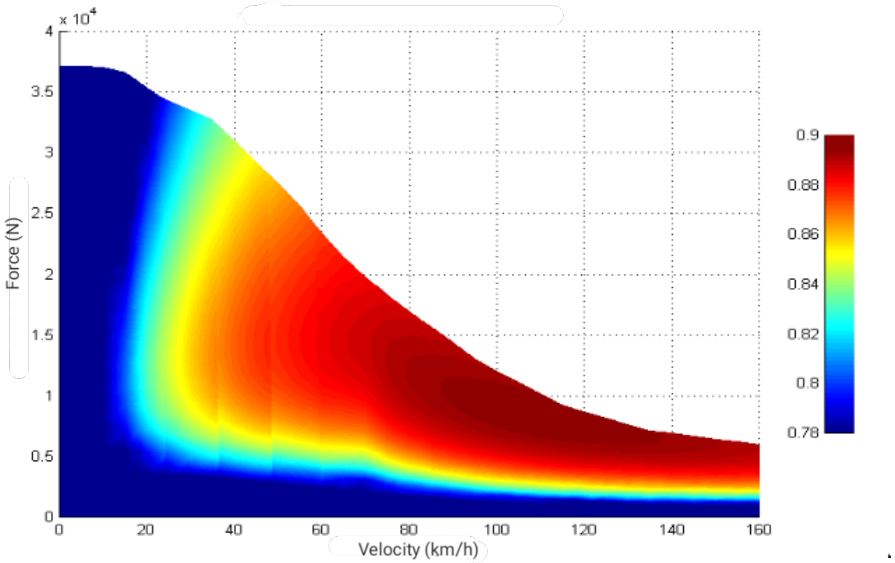


Figure 3.3: Energy Efficiency Curve

To capture these interactions, an efficiency function is defined within the model. This acts as an extension to the energy model. We start with a linear approximation of the efficiency function.

Linear Efficiency Function

We know from the previous formulations

$$E \propto \frac{1}{2}(v_{i+1}^2 - v_i^2) = E_{i+1} \ ,$$

i.e.,

$$E_{i+1} = (v_{i+1} - v_i)(\frac{v_{i+1} + v_i}{2}) = \Delta v_i \cdot v_{avg_i} .$$

The efficiency function η is introduced. Previously the efficiency term was modelled as $\frac{1}{\eta}$ (based on conventional norm), where η approaches the maximum value η_{max} at the optimal velocity. We now model η as a function which approaches the minimum value as v_i approaches

v_{opt} (to suit the quantum formulation). We model η as a step-wise approximate function which is defined as the cumulative sum of the binary decision variables up to i^{th} step,

$$\eta_i = \eta_0 + \Delta\eta \sum_{j=0}^{i} x_j \ ,$$

where η_0 is the baseline efficiency, and $\Delta\eta$ represents efficiency change per velocity increment. As a simplification, which removes higher order terms from the equation, we assume $v_i \approx v_{avg_i}$. Therefore the energy consumption for a single time-step is given as

$$E_i = \Delta v \cdot v_i \cdot \eta_i \ .$$

The step-wise formulation of η is similar to the method used for velocity, expressing it as the cumulative sum of the previous decision variables.

$$\eta_i = \eta_0 + \Delta\eta \sum_{j=0}^{i} x_j \ , \tag{3.14}$$

where η_0 is the baseline efficiency, and in this case we cannot assume η_0 to be 0. Re-expressing the energy over a time-step in terms of η and v, we get

$$E_i = \Delta v \cdot \left(\Delta v \sum_{j=0}^{i} x_j \right) \cdot \left(\eta_0 + \Delta\eta \sum_{j=0}^{i} x_j \right) \ .$$

The total energy of the system is defined by summing the energy consumption across all time-steps N, leading to the expression

$$E_{tot} = \sum_{i=1}^{N} \left[\eta_0 \cdot \Delta v^2 \left(\sum_{j=0}^{i} x_j \right) + \Delta\eta \cdot \Delta v^2 \left(\sum_{j=0}^{i} \sum_{k=0}^{i} x_j x_k \right) \right] \ .$$

As both, η and v are influenced by the same binary decisions, we can simplify the above expression by replacing the double summation with a single summation, as the expression will be valued when $j = k$, yielding

$$E_{tot} = \sum_{j}^{i} \sum_{k}^{i} x_j x_k = \left(\sum_{j}^{i} x_j \right)^2 = \sum_{j}^{i} x_j \ ,$$

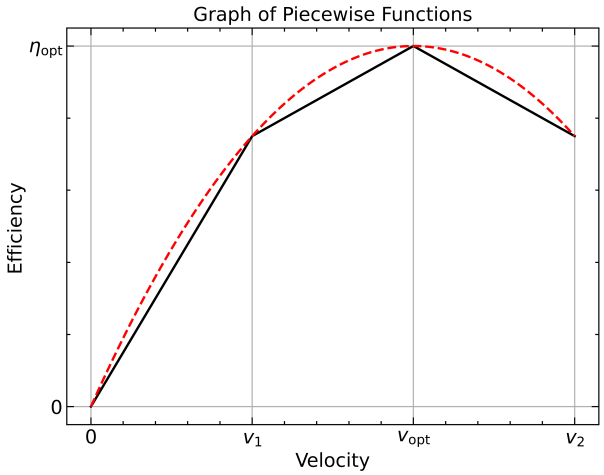


Figure 3.4: Piecewise approximation of η

resulting in the expression for total energy consumption

$$E_{tot} = \sum_{i=1}^{N} \left[\eta_0 \cdot \Delta v^2 \left(\sum_{j=0}^{i} x_j \right) + \Delta\eta \cdot \Delta v^2 \left(\sum_{j=0}^{i} x_j \right) \right].$$

General Efficiency Functions

The efficiency function as described above is simplistic assuming a linear relationship with increasing velocity. However, the data in Fig 3.3 suggests that the relationship between efficiency, velocity, and force is much more complicated. Ideally, it can be modelled in a more realistic way while not overly complicating the model and relying on higher-order relationships. A more realistic model which can be used is a quadratic curve which increases to optimal efficiency η_{opt}, at v_{opt}, and starts decreasing again as it becomes less efficient above the optimal velocity.

Lookup table: It was first considered to utilise a lookup table to dynamically adjust the $\Delta\eta$ efficiency value based on the current velocity (v_i). This implementation serves to illustrate both the benefits and draw backs of operating in a quantum setting. The binary variables cannot be evaluated while in a superposition, as it would destroy the superposition and the state would collapse. This illustrates that traditional methods are not applicable in a quantum setting.

Problem sectioning: Another option to obtain varying efficiency values dependant on the value of v, was to separate the problem into multiple sections, and use piecewise linear approximations of the efficiency function, illustrated in Fig 3.4. This method would allow the implementation of simpler linear models within each segment while closely approximating a more complex, non-linear efficiency curve.

3.4 Acceleration-Deceleration Model

While the above formulations consider different energy models and allow for the modelling of the basic mechanics, it is missing any mechanism to allow the model to arbitrarily vary its velocity.

To incorporate arbitrarily varying velocity, the problem is needed to be reformulated to model more than a single binary state. Following a method used in [55], the binary decision variables are refined as $\{0,1\}^2 N$ and for each time step a two-bit string decision variable is defined. These allow for the representation of 4 possible states $00, 01, 10, 11$. The following generalised velocity update rule is defined based on a two-bit string decision variable,

$$\mathbf{x} \in \{00,01,10,11\}^N \rightarrow \begin{cases} v_{i+1} = v_i & \text{if } x_i = 00 \quad \text{(CO)} \\ v_{i+1} = v_i + \Delta v & \text{if } x_i = 01 \quad \text{(MA)} \\ v_{i+1} = v_i - \Delta v & \text{if } x_i = 10 \quad \text{(MB)} \\ v_{i+1} = v_i & \text{if } x_i = 11 \quad - \end{cases} \tag{3.15}$$

The model maps the bit-strings 00, 01, 10 to cruising (CR), braking (MB), and acceleration (MA) respectively. It does not allow for the state 11, and considers that it to be invalid.

Formally, we can consider a system with N discrete time-steps, with the i^{th} time-step having a corresponding two-bit decision variable $x_{i,a}, x_{i,b}$.

Energy Models

The energy model is defined in a similar manner to the above linear efficiency function, with a constant energy cost per unit increase in velocity, and a reduction based on binary decisions up to the point i. With the addition of a braking mechanism, the energy of regenerative braking can also be considered. For simplicity, it is assumed for every braking decision will return a small portion of the energy α of the energy required for acceleration, leading the problem:

$$\textit{Minimise} \quad \sum_{i=1}^{N} (\Delta v^2 \cdot x_{i,a} - \alpha \Delta v^2 \cdot x_{i,b}) \; . \tag{3.16}$$

Additionally, a penalty term is defined to discourage simultaneous braking and acceleration $x_{i,a} x_{i,b} = 11$ with a Lagrange multiplier λ_1, expressed as

$$\sum_{i=1}^{N} \lambda_1 \cdot x_{i,a} x_{i,b} \; . \tag{3.17}$$

Constraints

The constraints are similar to those applied to the previous model, with the notable difference that a new constraint of net-zero velocity is applied. The constraints are defined as:

Distance constraint: The total distance D that must be travelled between two stations, where braking is considered as

$$\sum_{i=1}^{N}((N-i)\Delta v \cdot x_{i,a} - (N-i)\Delta v \cdot x_{i,b}) = D \ , \tag{3.18}$$

where D is distance requirement between two stations.

Net-Zero Velocity Constraint: This constraint is to ensure the sum of acceleration and braking decisions is zero. This effective ensures that, assuming the train starts from rest, the train will also be at rest by the end of the solution, and the constraint is

$$\sum_{i=1}^{N}(x_{i,a} - x_{i,b}) = 0 \ . \tag{3.19}$$

Final unconstrained penalty model

The final model can be defined as a single objective function, with the addition of the penalty terms

$$\begin{aligned} \text{Minimise} \quad & \sum_{i=1}^{N}\left(\Delta v^2 x_{i,a} + P \cdot x_{i,a} \cdot x_{i,b}\right) \\ & + \lambda_1 \left(\sum_{i=1}^{N}((N-i)\Delta v \cdot x_{i,a} - (N-i)\Delta v \cdot x_{i,b}) + D\right)^2 \\ & + \lambda_2 \left(\sum_{i=1}^{N}(x_{i,a} - x_{i,b})\right)^2 \end{aligned} \tag{3.20}$$

where λ_1 & λ_2 are appropriately sized Lagrange multiplier parameters.

A step-by-step implementation of the developed models, simple to more complex is accomplished using quantum algorithms in the following chapters, with the results discussed.

4 Implementation of Quantum Algorithm

4.1 Qiskit Optimisation

The implementation on the IBM cloud platform was facilitated by the Qiskit library, an open-source SDK for developing circuits aimed at quantum hardware. It provides methods for executing circuits on hardware, which available through IBM Cloud. This include quantum simulators, and currently 127-qubit machines.

Qiskit Optimisation is an extension of the Qiskit framework, which allows for the high-level modelling of problems and automatic problem conversion, i.e. Quadratic Program to QUBO. It leverages the Qiskit Algorithms [56], to solve a number of different problems such as Grover's Adaptive search, or in this case the use of VQA, with an implementation of QAOA. This enables problems formulated using Qiskit Optimisation to be run on both quantum hardware, simulators, and classical optimisers such as CPLEX or Gurobi, which are provided as part of the module.

It is important to note that the majority of results using quantum algorithms within the Qiskit framework were obtained using the quantum simulator, Aer. Aer is a high-performance simulator offered with qiskit and allowed for the testing and debugging of circuits. Jobs are submitted to IBM hardware are place in a queue, which can take from minutes to hours to complete. Algorithms such as QAOA need multiple submissions to quantum hardware to get a solution, this is infeasible to run all results without dedicated time on hardware. As such, the QAOA circuits below were executed on a quantum simulator, with even small circuits with 8 qubits taking up to 5 minutes to run, and larger circuits being restricted by memory requirements.

The Qiskit Optimisation abstracts the creation of circuits, as it makes use of the available qiskit algorithms. By using an abstract API for the solver, we avoid having to define the Ising Hamiltonians to run on quantum hardware, though these steps are outlined in Sec 4.1.3. This streamlines the process of submitting the problem, allowing for focusing on problem

representation. As the problem definitions are compatible with classical optimisers, it allows for ease of benchmarking and testing.

Classical Solvers

Classical solvers (CPLEX and Gurobi) were used throughout the project for testing and validation. IBM ILOG CPLEX Optimisation Studio (CPLEX) is a classical optimisation suite, developed by IBM. CPLEX is a comprehensive suite which can be used to solve linear programming (LP), mixed-integer programming (MIP), and quadratic programming (QP) problems. Given a problem, the CPLEX suite determines the optimal algorithm to solve it based on the problem structure. For example, a Mixed-Integer problem would be solved with a branch-and-cut algorithm. These details are abstracted away from the user. Gurobi is another classical solver for optimisation problems, also part of the Qiskit module, allowing for its use in conjunction with CPLEX to validate the results obtained from the quantum circuit.

4.1.1 Constant and Quadratic energy models

The qiskit models were implemented using a `QuadraticProgram`, which is a class to model quadratically constrained quadratic programs of the form

$$\begin{aligned} &\text{minimise} && x^\top Q_0 x + c^\top x \\ &\text{subject to} && Ax \leq b \\ & && x^\top Q_i x + a_i^\top x \leq r_i, \quad 1, \ldots, i, \ldots, q \\ & && l_i \leq x_i \leq u_i, \quad 1, \ldots, i, \ldots, n, \end{aligned}$$

where,

- x represents the vector of decision variables, which can be binary, integer, or continuous,
- Q is a matrix defining the quadratic coefficients of the objective function and the quadratic constraints respectively.
- c and a_i are vectors representing the linear coefficients in the objective function and the linear constraints respectively,
- A is the matrix representing the linear coefficients for the inequality constraints,
- b and r_i are vectors that define the right-hand side limits of the linear and quadratic inequality constraints respectively.
- l_i and u_i specify the lower and upper bounds for each variable x_i.

The implementation of the railway control problem will be restricted to binary variables. To get a better understanding of how it is constructed from the objective function and constraints,

the problem will be built up step by step. Using the constant energy model as an example, implementing the objective function, as expected there are energy interactions on the diagonal of the heatmap, in Fig 4.1.

The objective function is defined via a pair of dictionaries, defining the linear and quadratic interactions between variables. Allowing for the definition

$$E = \alpha \underbrace{\sum_{i} x_i}_{\text{Linear}} + \beta_{i,j} \underbrace{\sum_{i=1}^{N} \sum_{j=0}^{i-1} x_i x_j}_{\text{Quadratic}}$$

where α_i is the linear bias, and $\beta_{i,j}$ is the quadratic interaction bias.

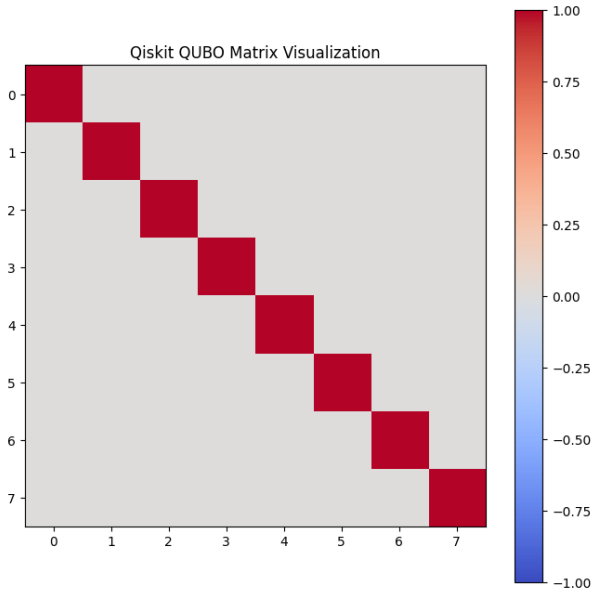
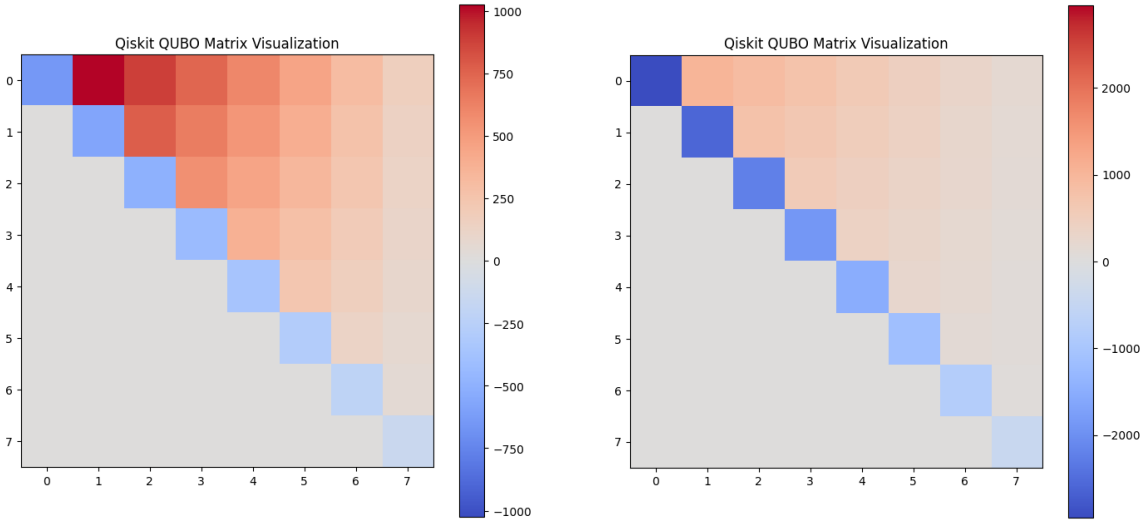


Figure 4.1: Objective function with no constraints

Implementing the two necessary constraints, the **distance constraint**, and the **speed limit constraint** the result can be seen in Fig 4.2

The inclusion of linear constraints leads to the inversion of the energies along the diagonal, illustrated in Fig 4.2. Qiskit Optimisation does not allow for modification of these penalties or the Lagrangian modifiers. However, as D increases there is a noticeable change to the magnitude of the diagonal terms, especially the earlier linear terms. It has a greater influence on the model, encouraging the optimiser to find solutions in the which the model accelerates earlier in solution, as it leads to a greater energy reduction.



(a) D=4 (b) D=20

Figure 4.2: QUBO with constraints, varying the value of D

4.1.2 Efficiency Models

The efficiency model was implemented in a similar way to the constant and quadratic models above, with the inclusion of an efficiency term which reduces the energy consumption based on previous interactions. The efficiency is modelled as some small energy recovery by term α, therefore the energy consumption in the i^{th} timestep is given by:

$$E_i = x_i \cdot \Delta v^2 - \alpha \sum_{j=0}^{i-1} x_j \cdot \Delta v^2 \ ,$$

where α is a small negative coefficient. This requires careful consideration of the α coefficient dependant on the number of time-step in the problem considered.

Generalised Efficiency Functions

As mentioned in the previous section, the proposed method was through problem sectioning, where the Quadratic program is partitioned into three sections based on the piece-wise linear approximation of the efficiency function. As the model has no encoded mechanism for braking and only accounts for velocity increase, it is separated into three values of $\Delta\eta_1, \Delta\eta_2, \Delta\eta_3$, corresponding to the three linear approximation values.

As the model, has no encoding for braking the value of the optimal solution results in the minimum number of steps required to satisfy the distance constraint.

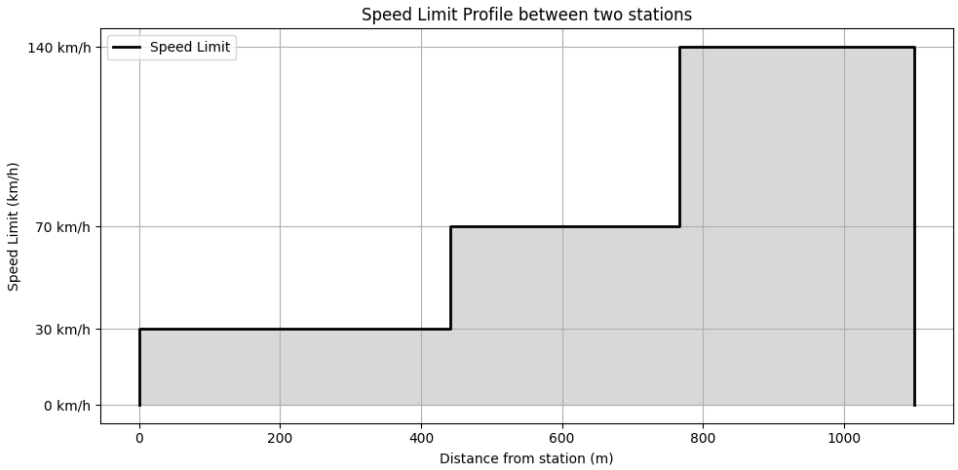


Figure 4.3: Speed limit profile between two stations

Speed-limits

Partitioning the model into sections allows for the modelling of local speed limit constraints. An example of the local speed-limits for different sections of track can be seen in Fig 4.3. Using the problem sectioning method, the track can be split into any number of sections, with each section being constrained by the local speed limit for that section.

For example, the sections of track in Fig 4.3, can be split into three sections, with local limits 30km/h, 70km/h, and 140km/h for each section respectively.

4.1.3 Ising Hamiltonians

As mentioned previously, the QUBO model is equivalent to the Ising model. To solve a QUBO using a quantum solver, the problem must first be converted to an Ising model. To convert a quadratic program to an Ising model, it is converted to a QUBO to create an unconstrained model, which can then be transformed into equivalent Ising model.

For clarity, the speed limit constraint has been omitted from the problem instance used in the following example. This is due to inequality constraints requiring the introduction of additional slack variables in the QUBO/Ising model. The problem is defined with N=4 and D=13. Fig 4.4, shows the solutions to the given example. The corresponding circuit used to solve this Ising Model is depicted in Fig 4.5. The circuit implements the QAOA algorithm to find the solution to the Ising model. Each qubit begins with a Hadamard gate to create a superposition, followed by multi-qubit gates that apply complex phase shifts defined by the Ising model parameters $\gamma[0]$ and $\beta[0]$. Measurements at the end of the circuit determine the state of each qubit.

Additionally, Figure 4.6 shows the quasi-probability distribution obtained from the results of the Minimum EigenOptimiser (MEO) used in this computation. The optimal solution within

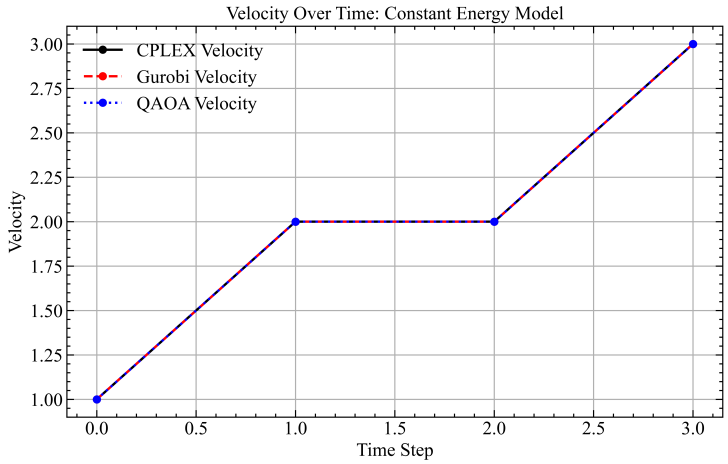


Figure 4.4: Solution to the example Ising Hamiltonian

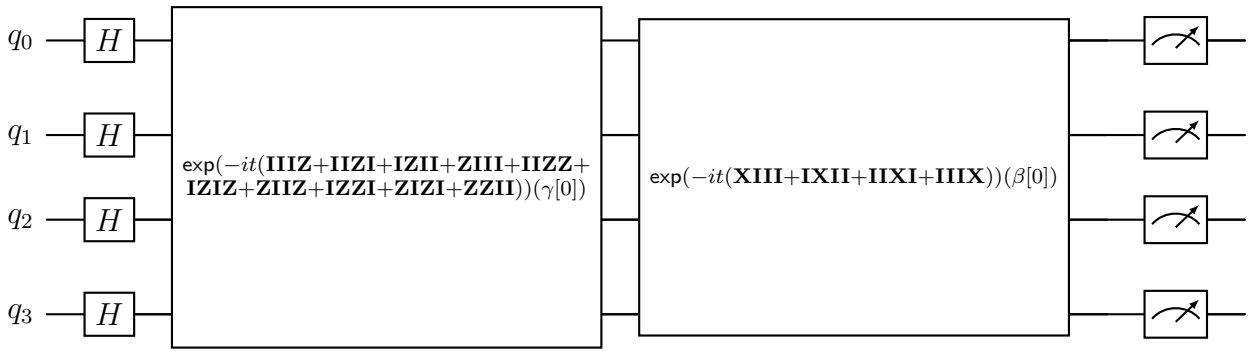


Figure 4.5: Optimal Circuit ($\beta[0] = 0.29468$ and $\gamma[0] = -0.00916$)

this distribution is highlighted in red.

4.2 D-Wave Implementation

D-Wave Systems provide a platform called Leap, which includes access to their quantum annealing-based quantum computers, software development kits (SDKs), and tools designed to help developers program quantum computers. The Ocean SDK is a key component of this ecosystem, offering a suite of Python libraries for designing, implementing, and running optimisation problems on quantum annealers. One of the core models within the Ocean SDK is the Binary Quadratic Model (BQM) for formulating optimisation problems.

The BQM is a mathematical representation used to describe optimisation problems in a form suitable for submission to a QPU. It is defined over binary variables and consists of linear and quadratic terms. The purpose of a BQM is to model optimisation problems that can be naturally expressed through linear and quadratic relationships among binary variables. This includes, but is not limited to, Ising models and QUBO problems [48]. The general form of a

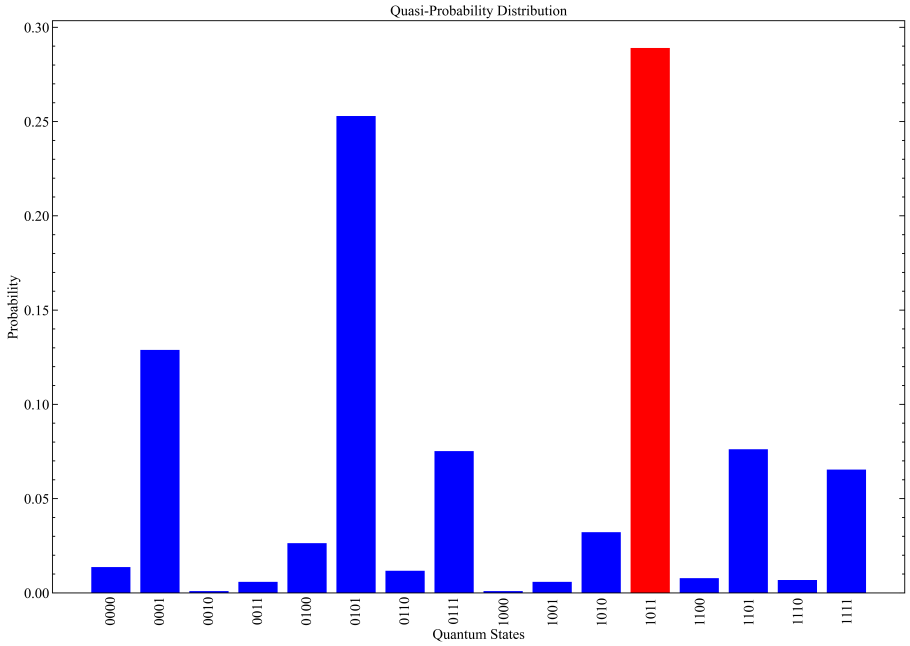


Figure 4.6: Quasi-Probability Distribution;

BQM can be expressed as:

$$E(\mathbf{x}) = \sum_i a_i x_i + \sum_{i,j} b_{ij} x_i x_j + c \tag{4.1}$$

where:

$\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ are binary decision variables; a_i are linear coefficients; b_{ij} are quadratic coefficients between variables x_i and x_j ; and c is a constant offset.

4.2.1 Sampling the problem

As the QPU is a probabilistic system, there is no guarantee that on any read in the annealing cycle the optimal solution will be the output. As such, generally multiple reads are taken when sampling a given problem. This is relatively inexpensive as the read time for a single cycle is on the scale of microseconds, therefore, with a larger number of reads, a more diverse number of solutions is returned with a higher probability of containing an optimal solution [57].

There are two main methods of accessing the QPU within D-Wave. The D-Wave sampler is a low-level sampler which allows for control over QPU parameters, such as annealing schedule, annealing time, etc. To use this sampler the problem must first be embedded onto the architecture of the QPU being used, this entails specifying how the binary variables are encoded onto the qubit architecture. This is called direct-embedding. Thankfully, D-Wave provides a software tool, which given a QUBO problem, will calculate the most effective way to embed the problem onto the topology.

As this problem does not map perfectly onto the graph topology, in this case the sampler makes use of chaining. This is the use of multiple qubits to represent a single variable, the qubits are constrained to have the same value. Allowing the problem to be mapped onto the topology, comes with the disadvantage of increasing the number of qubits required to represent a problem. For an example problem with 20 binary variables, mapping onto the QUBO topology required 53 physical qubits. The direct embedding sampler only accepts BQM models. This embedding is visualised in Figure 4.7, showing the embedding of the 20 binary variable problem onto a Pegasus graph.

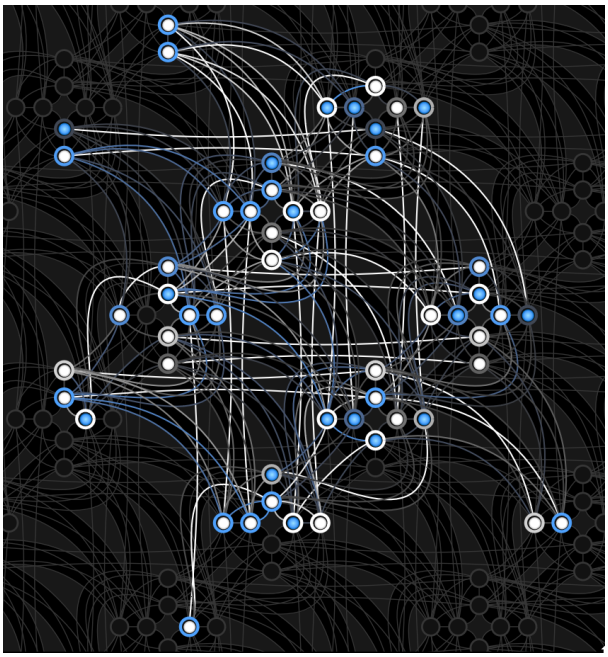


Figure 4.7: Example of direct embedding with 20 variable problem

The other method of accessing the QPU is the Leap-Hybrid sampler. This solver abstracts the details of direct embedding and handles the preprocessing and submission to the QPU separately. This allows for larger problems with more binary variables than available on a system to be submitted by similar means to that discussed in Sec 2.4 by qbsolv, through QUBO decomposition. The Leap-Hybrid solver can also accept other forms of problems, such as Constrained Quadratic Models CQM. These models can be built using Integer, or continuous variables and are not constrained to binary models.

4.2.2 Classically sampling larger problems

Ocean SDK offers a number of classical samplers that can be run locally on one's own CPU for debugging or testing. As the problem size increases it becomes more difficult to solve the problem classically. D-Wave offers simulated annealing sample, a probabilistic heuristic for optimisation and approximate Boltzmann sampling, well suited to finding good solutions of large problems.

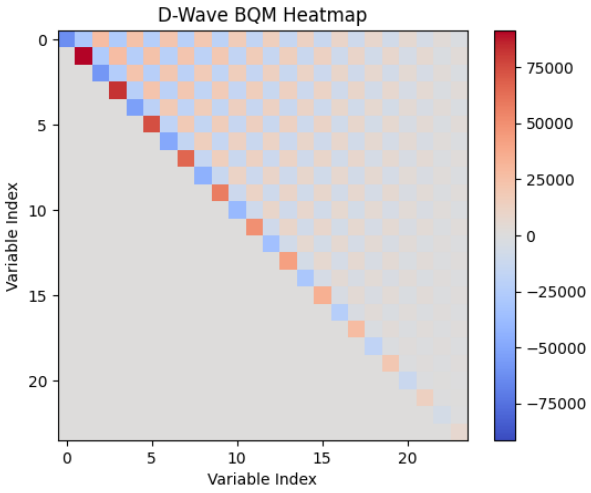


Figure 4.8: Heatmap of two-bit model

4.2.3 Acceleration-Deceleration implementation

The D-Wave implementation has a similar formulation to the Qiskit formulation, which was created using a Quadratic programming environment. Using the D-Wave SDK, we can create a BinaryQuadraticModel, which encodes the QUBO directly. As mentioned previously, since QUBO formulations are unconstrained, the constraints must be formulated as penalty terms, creating less favourable outcomes where constraints are violated. The two-bit implementation results in the QUBO structure seen in Fig 4.8, with alternating positive and negative terms, representing the acceleration and deceleration decision variables.

Constraints and Lagrange parameter choice

The constraints for the BQM are imposed as linear equality/inequality constraints, acting as a wrapper for the penalty models of the penalty model equivalent forms which were outlined in Tab 2.1

The Lagrange multiplier controls the weight or penalty strength. The linear constraint is multiplied by this value when added to the binary quadratic model [58]. The choice of this value effects a model's ability to converge a feasible solution, balancing between satisfying constraints and minimising the objective function. There is no set method for choosing the parameters; this involves trial and error using feedback from both the solutions of both classical and leapHybrid solvers.

5 Evaluation of the Results

This chapter aims to assess the performance of the implemented models, comparing the effectiveness for solving the railway trajectory problem for both the Qiskit and D-Wave implementations.

5.1 Evaluation Metrics

- Solution Quality: Comparing the quantum solution to its classical equivalent and the optimal solution.
- Computation Time: Time to retrieve solutions. Computation time for Qiskit can be measured during simulation or when submitting to quantum hardware through the IBM Cloud. Classical solvers, which are run locally, are evaluated based on the time taken to solve and obtain a feasible solution.
- Scalability: It depends on how well the solution technique can handle larger problem instances.
- Robustness: Consistency of the solutions across different runs, mostly applicable for the heuristic solvers such as simulated annealing or the quantum annealer.

5.2 Qiskit Evaluation

The problem instance is defined with the following parameters:

- N = 8, $\Delta v = 1$, $D = 27$
- N = 12, $\Delta v = 1$, $D = 39$
- N = 16, $\Delta v = 1$, $D = 101$

The three problem instances were formulated and run for the three developed energy models (Constant, Quadratic, and Efficiency) using the Aer simulator. The simulations and optimisations were run on an intel i7-10750H, with 32GB of RAM. Tables 5.1,5.2 and 5.3 show the

solutions, energy, and computation times for each solver and energy model. Fig 5.1, 5.2 and 5.3 show their respective graphical solutions.

Energy Model	Algorithm	Duration (s)	Solution Vector	Energy
Constant Energy Model	CPLEX	0.111s	[1 1 1 1 0 0 0 1]	5.0
	GUROBI	0.016s	[1 1 0 1 1 1 0 0]	5.0
	QAOA	1.535s	[1 1 1 1 0 0 0 1]	5.0
Quadratic Energy Model	CPLEX	0.058s	[1 1 1 1 0 0 0 1]	25.0
	GUROBI	0.007s	[1 1 1 1 0 0 0 1]	25.0
	QAOA	1.652s	[1 1 1 0 1 0 1 0]	25.0
Efficiency Model	CPLEX	0.0453s	[1 1 1 1 0 0 0 1]	4.85
	GUROBI	0.005s	[1 1 0 1 1 1 0 0]	4.85
	QAOA	1.683s	[1 1 1 0 1 0 1 0]	4.85

Table 5.1: Comparison of Optimisation Algorithms for Problem Instance (N=8, D=27)

Energy Model	Algorithm	Duration	Solution Vector	Energy
Constant Energy Model	CPLEX	0.050174s	[1 0 1 1 1 0 0 0 0 0 0 0]	4.0
	GUROBI	0.015271s	[1 0 1 1 1 0 0 0 0 0 0 0]	4.0
	QAOA	3.701301s	[1 1 1 0 0 0 1 0 0 0 0 0]	4.0
Quadratic Energy Model	CPLEX	0.102950s	[1 0 1 1 1 0 0 0 0 0 0 0]	16.0
	GUROBI	0.010810s	[1 1 1 0 0 0 1 0 0 0 0 0]	16.0
	QAOA	3.648816s	[1 1 0 1 0 1 0 0 0 0 0 0]	16.0
Efficiency Model	CPLEX	0.017139s	[1 1 1 0 0 0 1 0 0 0 0 0]	3.9
	GUROBI	0.008471s	[1 1 0 1 0 1 0 0 0 0 0 0]	3.9
	QAOA	3.355220s	[1 0 1 1 1 0 0 0 0 0 0 0]	3.9

Table 5.2: Comparison of Optimisation Algorithms for Problem Instance (N=12, D=36)

Energy Model	Algorithm	Duration (s)	Solution Vector	Energy
Constant Model	CPLEX	0.012255s	[1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1]	9.0
	GUROBI	0.003179s	[1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0]	9.0
	QAOA	9.036605s	[1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1]	9.0
Quadratic Model	CPLEX	0.015933s	[1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0]	81.0
	GUROBI	0.030111s	[1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0]	81.0
	QAOA	8.034787s	[1 1 1 1 1 1 0 0 1 1 0 1 0 0 0 0]	81.0
Efficiency Model	CPLEX	0.018490s	[1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1]	8.55
	GUROBI	0.026078s	[1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0]	8.55
	QAOA	8.338014s	[1 1 1 1 1 0 0 1 1 1 0 1 0 0 1 0]	9.45

Table 5.3: Comparison of Optimisation Algorithms for Problem Instance (N=16, D=101)

5.2.1 Discussion

QAOA has been evaluated through application of problem instances with increasing number of binary variables - 8, 12, 16. In case of the instances with 8 and 12, QAOA consistently matched the optimal solution found by the classical solvers, showing the effectiveness of QAOA for smaller instances.

There was a deviation in optimal performance in the problem instance with 16 binary variables; QAOA returned a non-optimal solution. As the number of qubits increases, so does the size of the energy landscape. QAOA is a heuristic solver and as such there is no-guarantee to find the global minimum. In this case, it is possible that the problem returned some local minima. Due to the heuristic nature of the quantum solver, ideally the problem would be required to be run multiple times and some majority voting used to determine the final solution.

The three problems are defined with 8, 12, 16 binary variables. It is important to note that for these input sizes, the solution space grows exponentially: $2^8\ (256), 2^{12}\ (4,096), 2^{16}\ (65,526)$ possible solutions respectively. In terms of scalability, the limitation of QAOA simulation limits the number of qubits that are feasible to be emulated. For physical hardware, the scaling is limited by the number of available qubits on hardware.

The computation time for QAOA is significantly longer compared to the classical solvers, this is due to the differences in operation. Simulating the quantum system requires additional computation, and QAOA itself is a complex algorithm requiring the state preparation, gate operation, measurement, and classical optimisation to run. While ideally these computations would be run on quantum hardware, practically there are long wait times. There is also the issue of noise with multiple runs required for confident solutions.

The quantum simulation above was run on ideal simulators; however, it is also possible to simulate the noise models of different quantum back-ends (using `ibm_kyoto`) at an increased computational cost. These simulated back-ends employ a realistic noise model as part of the simulation, while the classical simulators are deterministic in their results, QAOA can vary from run to run (and can be made deterministic by providing a seed value).

5.3 D-Wave Evaluation

The D-Wave implementation reformulated the problem to use two bits per time-step. Problem instance one (N=8, D=27) was remodelled to account for the braking period that was assumed in the Qiskit implementation. This instance was run using the available solvers (Exact, Simulated Annealing, Leap-Hybrid, Direct-Embedding). Table 5.4 presents the outcomes of running the reformulated problem using these solvers.

All the solvers returned the optimal solution found by the Exact solver; Fig 5.4 shows the

Algorithm	Energy	Time (s)
Exact	3.6	36.12
Simulated annealing	3.6	24.34
Leap-Hybrid	3.6	2.997
Direct-Embedding	3.6	0.4930

Table 5.4: Solutions of problem instance one with multiple solvers

solution found. The exact solver offers a benchmark for small problems; however, problems quickly become intractable for this solver, requiring the use of simulated annealing, a heuristic solver. The computation times shown for Leap-Hybrid and Direct-Embedding are taken from the values of QPU and solver access time respectively.

Problem scaling

One of the benefits of using the D-Wave platform is the capacity to process a significantly larger number of qubits compared to gate-based quantum computers. The hardware utilised is the Advantage 4.1 which supports 5000 qubits arranged in a Pegasus graph topology with a connectivity of 16.

As the size of the problem increases, the solution space grows exponentially. A problem is defined with parameters $N = 100$ and $D = 1100m$; modeling a scenario with 100 time steps, which translates into 200 binary variables due to the inclusion of both speed and braking variables at each step.

As quantum annealing, being a heuristic approach, does not guarantee an optimal solution within a finite number of samples, the problem was sampled 12 times using different solvers, and the results were averaged to mitigate the probabilistic results. This approach provides a more reliable estimate of the typical performance one can expect in practical scenarios. Table 5.5 illustrates a comparison of the two samplers.

Algorithm	Energy	Time (s)	Time to retrieve (s)
Leap-Hybrid	978.0834	2.997	20.9183
Simulated annealing	2029.5	1030.1101	-

Table 5.5: Comparison of classical heuristic optimiser to quantum annealing

The two solutions for the lowest energy results are illustrated in Fig 5.6.

The Leap-Hybrid solver demonstrates better performance in terms of energy minimisation and computation time. Returning an energy of 978.0834 with a total computation time of under 3 seconds and a retrieval time of 20.9183 seconds, where retrieval time accounts for the overhead of accessing and retrieving the solution from the cloud-based solver.

Simulated Annealing resulted in a much higher energy of 2029.5 and required over 1030 seconds to return the solution. This quality of the solution could be increased by increasing the number of reads, at the cost of a longer computation time.

The energy of both of these solutions are higher than would be expected for both solvers; this is due to constraint violations imposing penalties increasing the energy of the solutions. The Lagrange parameters used in the solutions could be further tuned to improve the convergence to valid the energy solutions.

A key advantage of using the D-Wave system is its capability to handle increasingly larger sets of qubits. However, due to the extensive connectivity related to this problem it becomes difficult to directly map the QUBO problem onto a Quantum Processing Unit (QPU). As previously highlighted, even a modest-scale example with just 20 binary variables required the use of 53 physical qubits. This is primarily due to the dense connectivity which required qubit chaining to effectively represent the problem. This scaling challenge is visualized in Fig 5.5, which shows the increase of binary interactions with the number of binary variables.

5.4 Comparison of the IBM and D-Wave from a practical standpoint

In the nascent field of quantum computing, different platforms fulfil different computational needs, especially in optimisation problems. IBM's Qiskit and D-Wave's Leap platform represent two distinct approaches to quantum computing, general-purpose universal quantum computing and specialised quantum annealing, respectively. Each platform has unique strengths, especially for solving optimisation problems.

The Qiskit platform uses gate-based quantum computing, meaning it can perform a wide range of computations, not just those specifically designed for quantum computers. The hardware is limited by the qubit scale. However, IBM's development of qubit technology has rapidly advanced, from a 17-qubit processor in 2017 to the 127-qubit Eagle processor in 2021, and further to the 433-qubit Osprey in 2022. By the end of 2023, IBM announced its 1121-qubit Condor processor, with a road-map to reach a 100,000-qubit system by 2033. They have seen similar growth in the quality of the qubits by improving error rates. This growth highlights rapid development of universal quantum computing, with increasing applicability of gate-based computing to practical problems.

D-Wave specialises in quantum annealing, a method particularly suited for optimisation problems. D-Wave's systems are designed to solve these problems natively, offering a direct approach to finding minimum-energy solutions for optimisation tasks. Starting from an 8-qubit system in 2011, D-Wave has aggressively scaled its technology. The 2000Q model,

introduced in 2015, has 2,000 qubits, and the latest Advantage system has over 5,000 qubits with enhanced connectivity ranging from 6 to 15.

From a practical standpoint, the choice between IBM's and D-Wave's platforms depends on specific project requirements: IBM's gate-based systems offer greater flexibility. However, for optimisation problems, particularly discrete optimisation, D-Wave provides an efficient but probabilistic solution.

IBM's advancements in qubit count are a step close to practical computation but must be viewed through the lens of the error correction and gate fidelity necessary for practical applications. D-Wave's quantum annealer, although not universal, scales effectively in solving large optimisation problems where high connectivity and qubit count directly translate to performance gains.

When it comes to the speed of accessing solutions, D-Wave is much faster. Solution time is on the scale of milliseconds for thousands of annealing cycles, with low waiting time for access to the QPU making it practically feasible to use quantum hardware in the project. With 10 minutes of access time per month and solutions taking only seconds, it is feasible to run many problems. IBM, on the other hand, also offers 10 minutes of access time to quantum hardware. However, running just one QAOA circuit on this hardware takes approximately 5 minutes. This means it is infeasible to run more than two solutions per month. They do offer more time on larger cloud based quantum simulators, but these simulators also suffer from long wait times.

The choice between IBM's Qiskit and D-Wave's Leap platforms should be guided by the specific requirements of the quantum applications in question. D-Wave excels in quick, specialised optimisation scenarios, particularly where the problem can be mapped directly onto a QUBO framework. IBM, while currently more suited for a broad range of experimental and theoretical quantum computing tasks, that cannot be run on a quantum annealer.

5.4.1 The Effect of Entanglement

Entanglement is a unique quantum mechanical phenomenon which allows qubits to be correlated in ways that are not possible in traditional computing. Both QAOA and QA make use of entanglement throughout their computation.

Entanglement is utilised in QAOA through the use of mixer Hamiltonian's, in our implementation of QAOA the TwoLocal ansatz was used with linear entanglement. The TwoLocal circuit is a parameterized circuit consisting of alternating rotation layers and entanglement layers. Qubits are entangled in a linear manner with qubit i entangled with qubits $i+1$ as so on, the circuit for which can be seen in Fig 5.7.

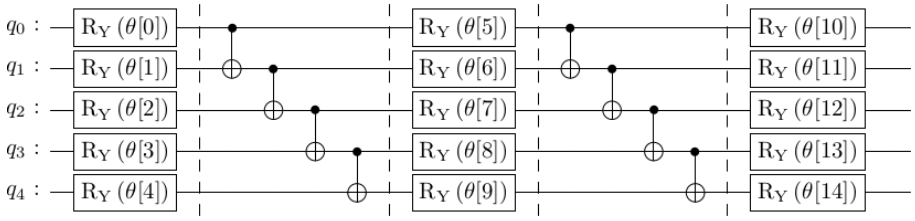
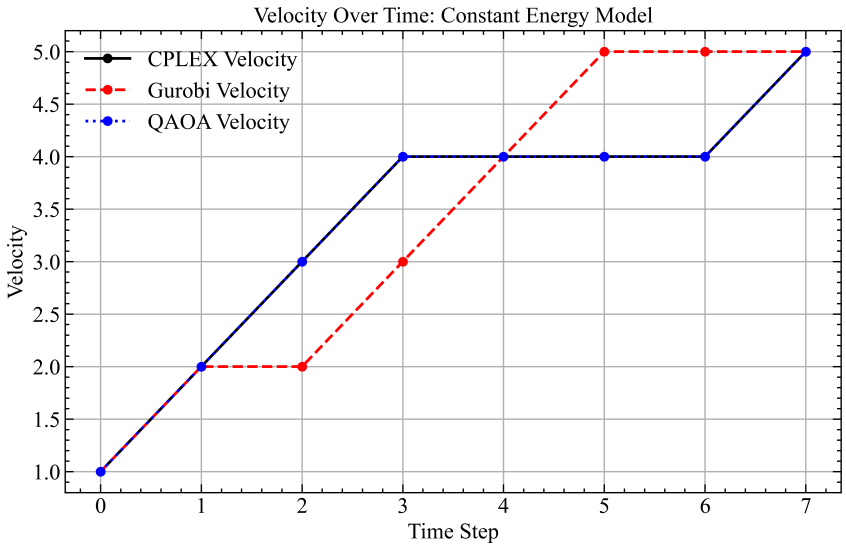
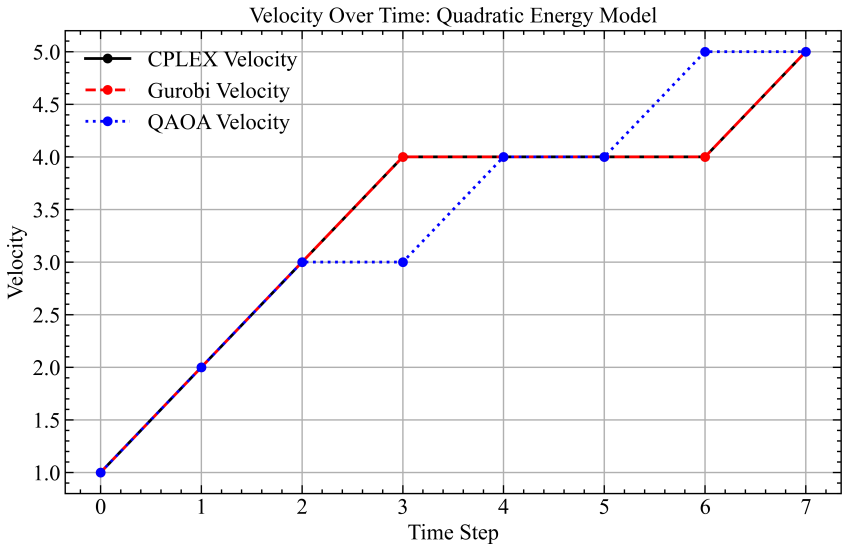


Figure 5.7: 5-qubit TwoLocal circuit with linear entanglement

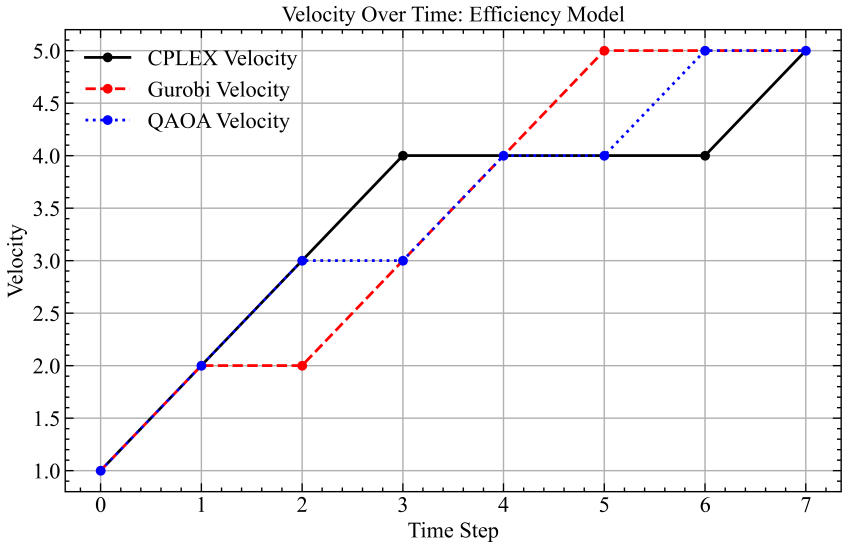
Entanglement is also utilised within the D-Wave system, though less explicitly. In 2014 by D-Wave, with Lanting *et al*. [59] showing that the qubits become entangled during the quantum annealing process, and this entanglement persists even as these systems reach equilibrium with a thermal environment.



(a) Constant energy model

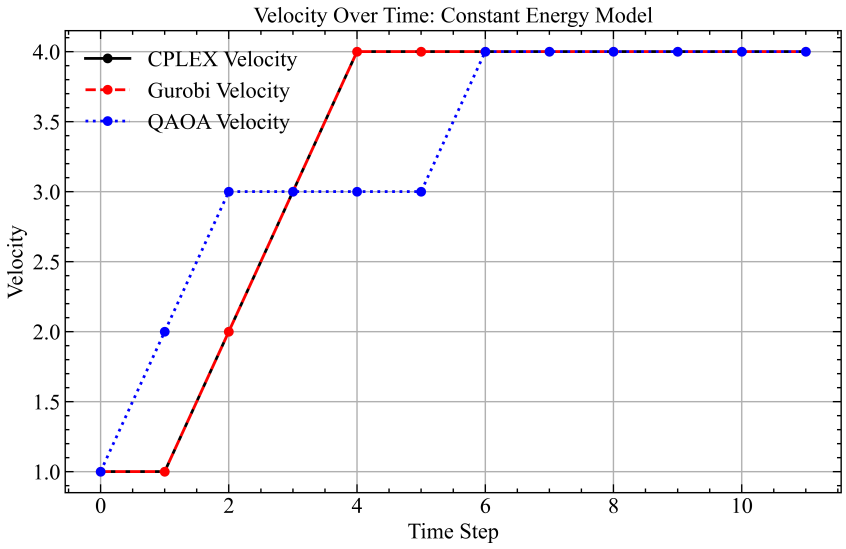


(b) Quadratic Energy Model

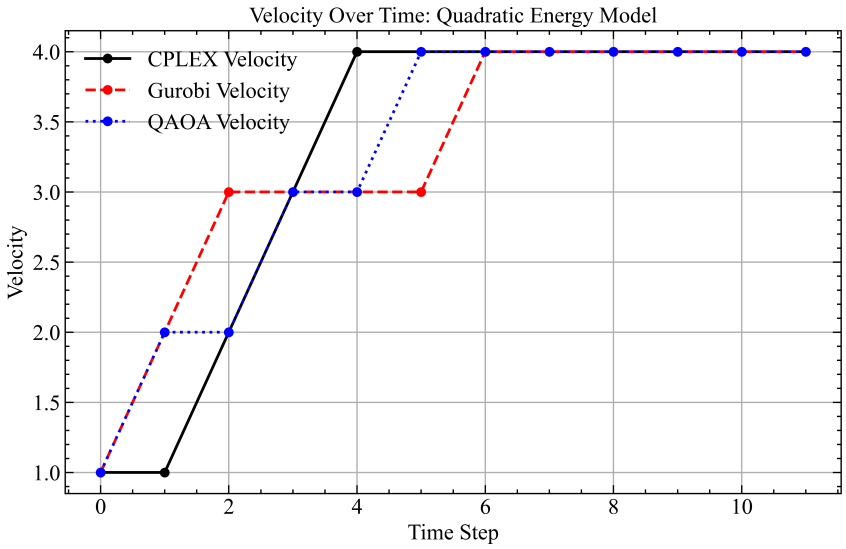


(c) Efficiency Energy Model

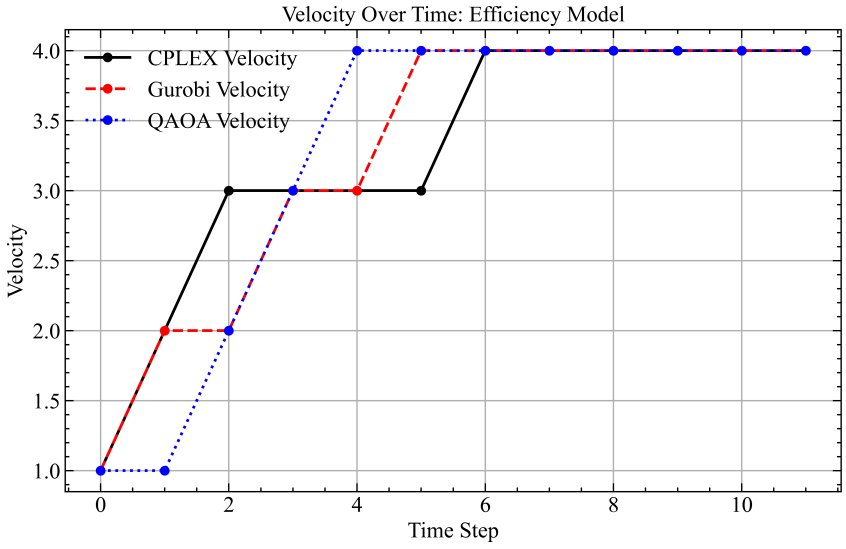
Figure 5.1: Energy model results for problem instance one (N=8)



(a) Constant energy model

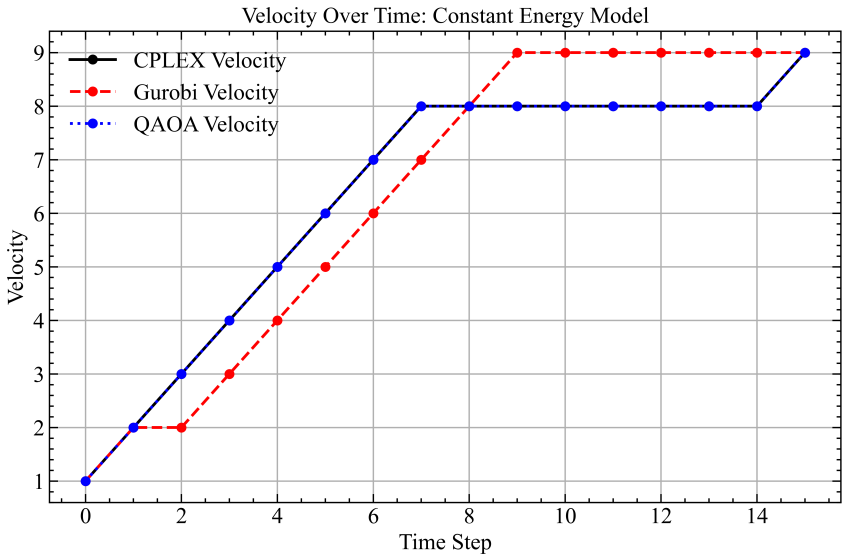


(b) Quadratic Energy Model

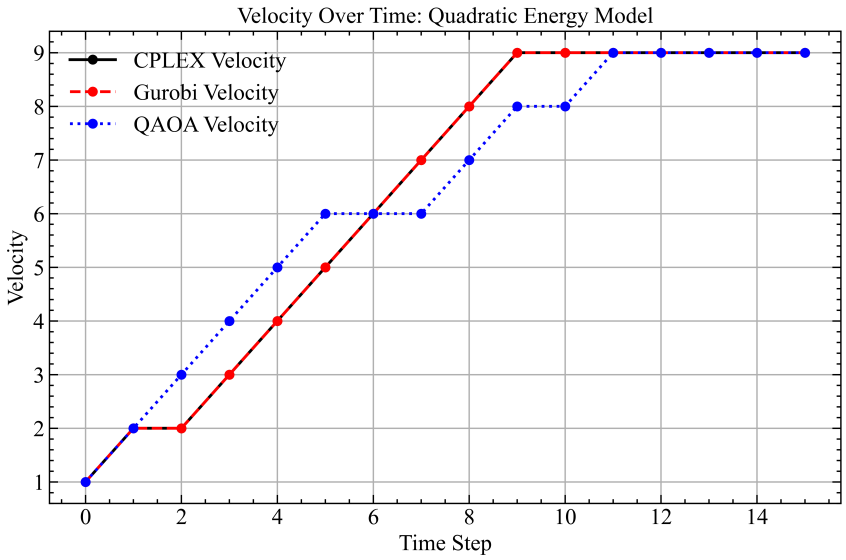


(c) Efficiency Energy Model

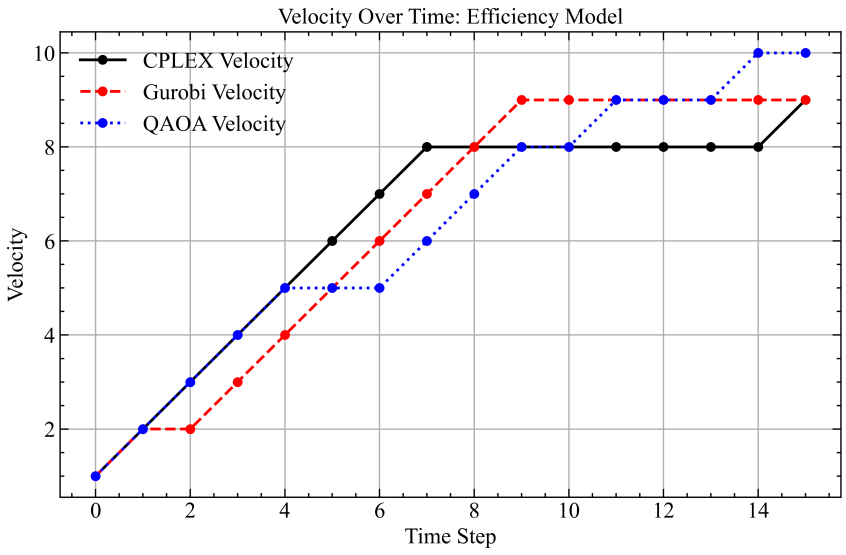
Figure 5.2: Energy model results for problem instance two (N=12)



(a) Constant energy model



(b) Quadratic Energy Model



(c) Efficiency Energy Model

Figure 5.3: Energy model results for problem instance three (N=16)

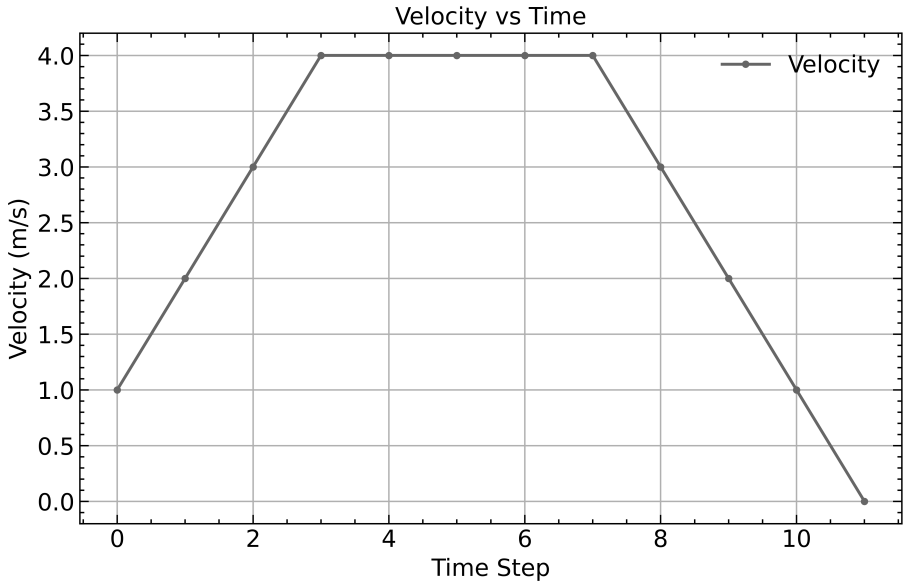


Figure 5.4: Solution found by all solvers on D-Wave

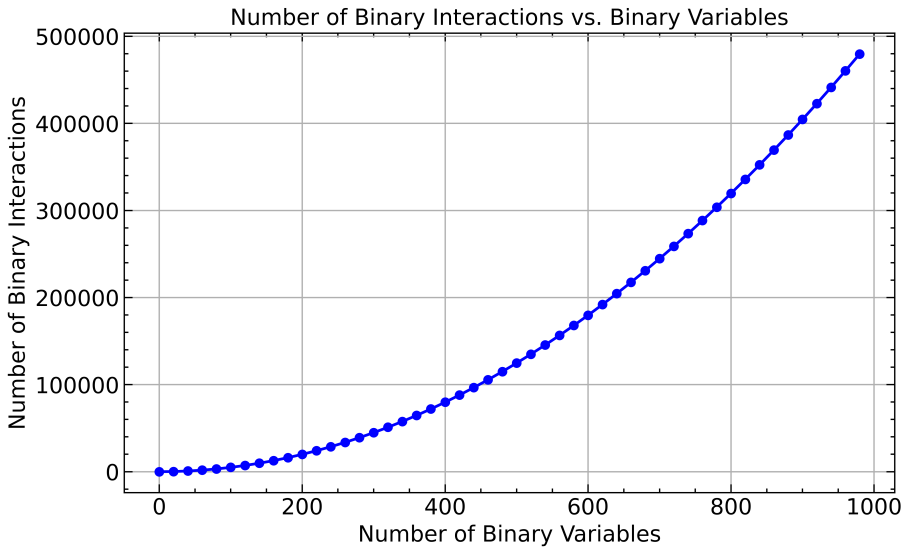
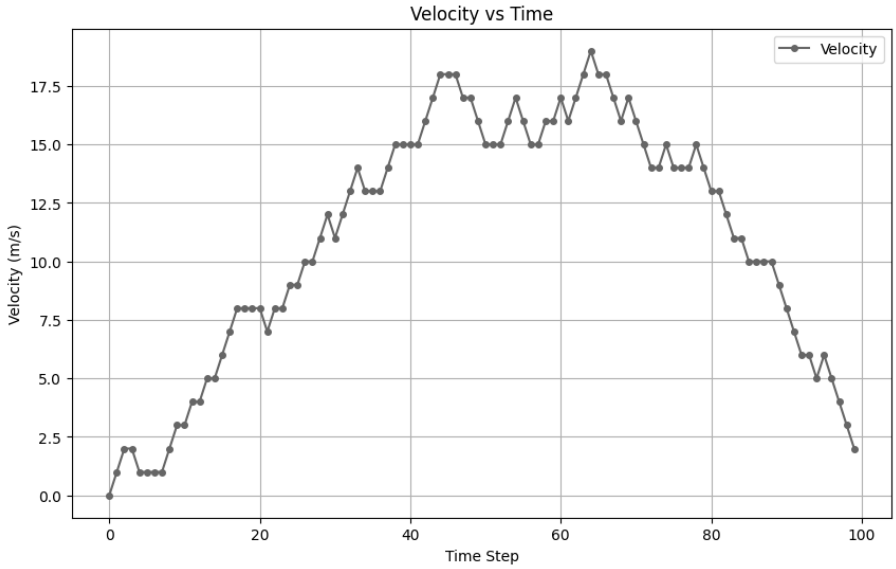
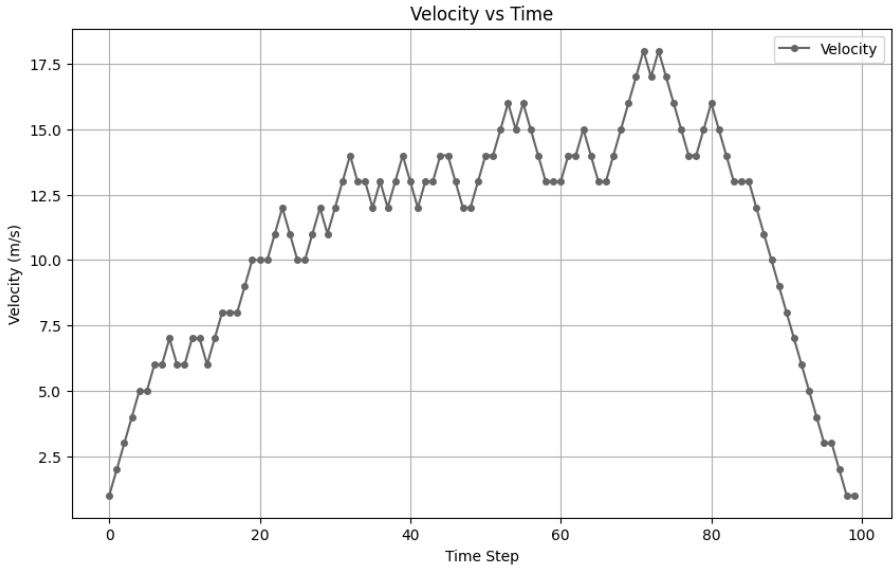


Figure 5.5: Scaling of No. Binary variables versus Interactions



(a) Best SA result; Energy: 1445



(b) Best QA result; Energy: 846

Figure 5.6: Lowest energy results for the two samplers

6 Railway Network Model

The optimisation of individual railway trajectories provides improvements of energy efficiency in terms of energy usage on a point-to-point basis. The problem of railway trajectory optimisation can be extended to a network of rails.

The travelling salesman problem (TSP) is a famous NP-complete problem. It is described as follows: a travelling salesman goes from city to city to sell products. The objective is to find a shortest path to visit all the cities and return to the city he started from. Doing this, he maximizes his potential sales in the least amount of time. Mathematically, it can be represented as a graph problem, specifically finding the shortest closed path that traverses every vertex in the graph once at a minimum cost; this is known as a Hamiltonian cycle. There is no known algorithm to solve this problem in polynomial time.

Given the NP-complete nature of the TSP, exact solutions for large networks are computationally infeasible with classical algorithms. However, heuristic and approximation algorithms, such as Genetic Algorithms, Simulated Annealing, or even Quantum Annealing and QAOA offer viable alternatives. These methods can provide near-optimal solutions within reasonable time frames.

Consider a rail network modeled as a weighted graph $G = (V, E)$, where V represents the set of stations and E corresponds to the direct railway connections between them. The weight w_{ij} of each edge (i, j) in this graph is defined by the minimum energy required to travel from station i to station j, derived from the trajectory optimisation results. The goal is to find a Hamiltonian cycle in G that minimises the total energy expenditure.

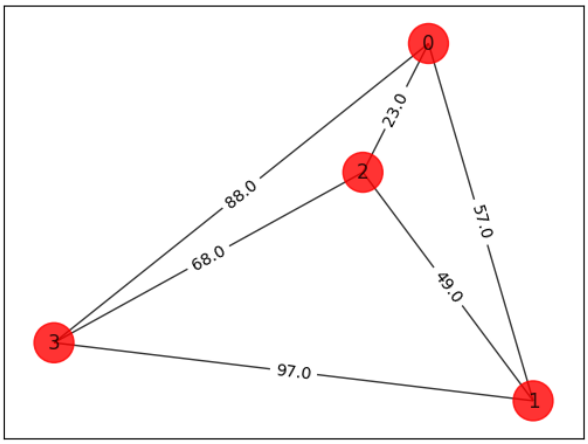


Figure 6.1: Graph of TSP nodes

A Hamiltonian cycle is described by N^2 binary variables $x_{i,p}$ where i is the i^{th} node, and p is the order in in the cycle. $x_{i,p}$ is 1 if solution contains the i^{th} node in order p. Each node must only appear once in the cycle. The two constraints are defined as:

$$\sum_{i} x_{i,p} = 1 \forall p \ , \tag{6.1}$$

$$\sum_{p} x_{i,p} = 1 \forall i \ . \tag{6.2}$$

A penalty is applied if both $x_{i,p}$ and $x_j, p+1$ are 1 and (i,j) are not connected, which can be expressed as:

$$\sum_{(i,j) \notin E} \sum_{p} x_{i,p} x_{j,p+1} > 0 \ , \tag{6.3}$$

assuming the boundary condition $(p = 0) \equiv (p = N)$. The cost function, or energy to be minimised, is given by

$$C(\mathbf{x}) = \sum_{i,j} w_{i,j} \sum_{p} x_{i,p} x_{j,p+1} \ . \tag{6.4}$$

6.1 Mapping trajectory optimisation to TSP

A graph is defined as $G = (V, E)$, where each vertex V represents a station in the rail network, and each edge is assigned a weight w_{ij}, which is derived from the trajectory optimisation method proposed in the previous sections. The network is illustrated in Fig 6.1.

Qiskit Formulation

This TSP problem can be solved in Qiskit by converting the QUBO representation to an Ising model in a similar manner as described in Sec 4.1.3. The number of required qubits for this formulation grows rapidly, n^2 qubits are required, where n is the number of vertices. It can be submitted to a quantum solver, such as QAOA.

From this result, they quasi-probability distribution seen in Fig 6.2 is extracted, showing only the 10 highest probability solutions.

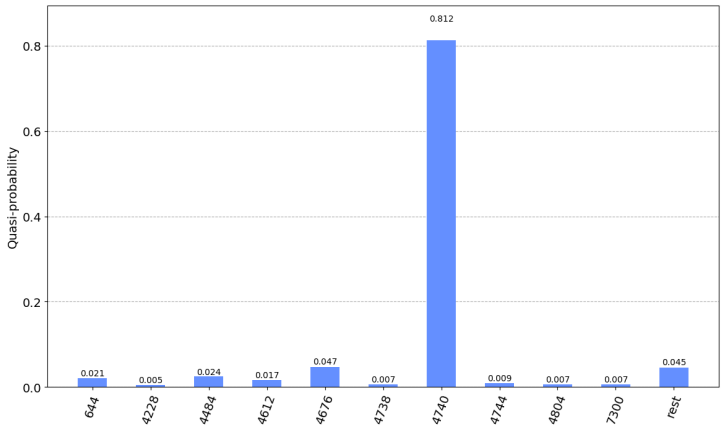


Figure 6.2: Quasi-Probability distribution

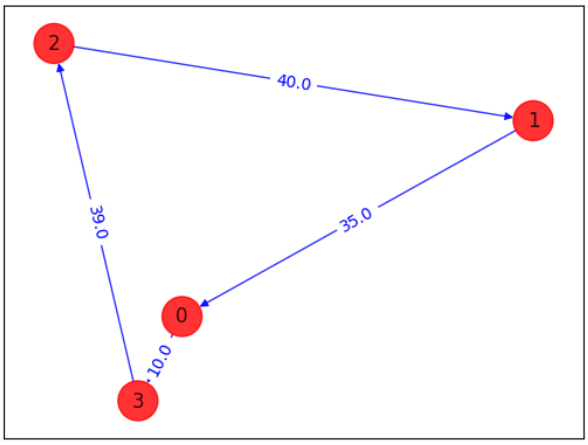
6.1.1 D-Wave formulation

This problem can also be solved using D-Wave's quantum annealers as the problem takes the form of a QUBO. This problem formulation can be directly submitted to the QPU via direct embedding.

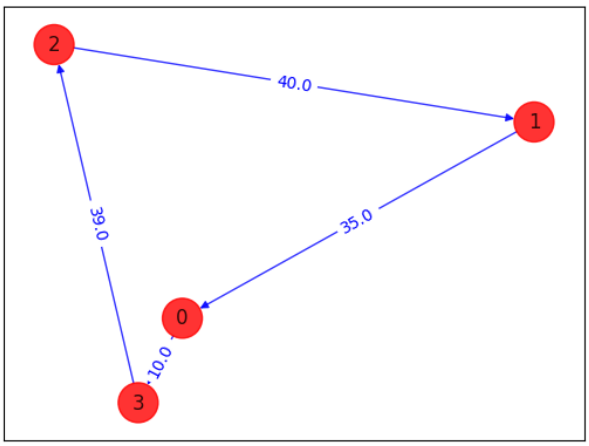
6.1.2 Results

Both QAOA, QA and the exact classical solver return the exact solution. The solutions are illustrated in Fig 6.3.

Both quantum results match the exact classical solution, showing the application of QC in the context of network wide optimisation. As mentioned, this problem scales rapidly as n^2 qubits are required. This in fact quickly outscales the number of qubits that are available on hardwares of today, such as the 127 qubits machines available with IBM. Table 6.1 shows a runtime comparison for graphs with 4 and 5 vertices, illustrating how the NP-hard nature of the problem is. We see while the runtime for quantum annealing remains low, it provides no guarantee of returning an optimal, or even feasible solution.



(a) Equivalent Qiskit and D-Wave result [1,0,3,2] with energy 124.0



(b) Classical result [2,1,0,3] with energy 124.0

Figure 6.3: Solutions to TSP problem

Graph size	Solver	Time
4	Classical Exact Solver	0.4s
4	QAOA Simulation	16.4s
4	QA	0.136 s
5	Classical Exact Solver	402s
5	QAOA	-
5	QA	0.179s

Table 6.1: Comparison of computation time for different solvers

7 Conclusion

7.1 General Overview

This thesis explored both the feasibility and potential of the application of quantum algorithms in energy efficient railway control. The primary goal was to develop mathematical formulations to model railway trajectories, which can be run on quantum hardware. Over the course of the thesis, a novel approach was developed for formulating railway trajectory optimisation, along with different models for energy consumption.

The research successfully demonstrated that quantum algorithms can be formulated and implemented to solve railway trajectory optimization problems. Different energy consumption models were developed and tested on platforms such as IBM's Qiskit and D-Wave's Leap, showing the potential for application.

A result worth highlighting from this research is the possibility of super-polynomial speed up of the optimal solution of the proposed formulation achieved by the quantum annealing algorithm. The quantum algorithm obtained the solution in a runtime under 3 sec, whereas in comparison the classical simulated annealing required over 1030 sec. Further, the quantum algorithm could reduce the optimal energy consumption by over 50% when compared with the optimal solution obtained from a classical approach.

The following are the main conclusions drawn from this study:

Analytical and classical results

The results from the quantum models matched both the analytical solutions derived in Sec 3.1, as well as the results found in the literature. The quantum solutions aligned with the solutions obtained from the classical solvers, with some minor deviation, which may be attributed to the probabilistic nature of quantum computation and the heuristic algorithms used with QAOA and QA.

Performance of QAOA and QA

The implementation of QAOA on using Qiskit showed potential, with results matching the classical solvers. This implementation made use of ideal quantum simulators, and could expect better performance in terms of scaling and run-time with better access to quantum hardware.

D-Wave's quantum annealers are more specialised for the optimisation with QUBO/Ising models, with a native ability to minimise energy state. It allowed for faster convergence and high quality solutions. Practically, it allowed almost instantaneous access to quantum hardware, with high qubit count compared to IBM. However, it is worth noting that with more binary variables the number of interactions scale quickly, requiring more physical qubits to represent the binary variables.

A comparative analysis was preformed between the two platforms, which found that D-Wave's systems generally performed better for QUBO/Ising based problem used in this thesis, both in terms of solutions and providing greater access to hardware. D-Wave's architecture is designed specifically for the types of optimisation problems addressed in this research, allowing for more direct and efficient problem-solving capabilities.

Quantum mechanical effects

The research illustrated how quantum mechanical effects, such as entanglement and superposition, are utilised within quantum computation. These allow for greater parallelism, and more efficient searching compared to classical methods.

7.2 Challenges and limitations

7.2.1 Hardware access

Access of advanced hardware remains one of the most significant barriers for practical quantum computation. Although recent advancements have significantly increased the number of qubits available, practical quantum applications such as the optimization tasks discussed in this thesis require a large number of qubits.

Practical scaling of complexity

Modelling and capturing the complexity of efficiency functions within the constraints of the QUBO framework is challenging. Binary variables simplify the modeling process but can limit the granularity with which efficiencies and other continuous variables are represented.

7.3 Network Problem

The scope of the thesis extended to a broader network application, examining the potential to optimise an interconnected system of railway stations through the application of QC applied to the travelling salesman problem (TSP). The quantum solutions matched the exact classical result showing the applicability for quantum computing in the context of railway network optimisation.

7.4 Future research

With the promising results of the proposed energy efficient railway optimisation, there are a number of areas of improvements to be explored before this can be used practically.

Realistic External Force modelling

A primary area for future improvement will be the inclusion of realistic external forces, such as aerodynamic drag, friction and track gradients. These factors significantly influence the energy consumption of railway systems but were simplified for feasibility and their incorporation would facilitate more accurate optimisation solutions, with great applicability to real-world problems.

Improved Efficiency Modelling

Another area is the improvement of efficiency modelling as efficiency plays a considerable role in energy consumption. Promising possible formulations for this improvement may include force-based decision variable model, allowing for more realistic and flexible modelling of energy efficiency. The use of a one-hot encoding scheme in conjunction with the force-based decision variable would allow for the quantisation of N force values, where N is the number of binary variables per timestep, resulting in more granularity in control.

References

[1] Y. Ruan, S. Marsh, X. Xue, Z. Liu, and J. Wang, "The quantum approximate algorithm for solving traveling salesman problem," *Computers, Materials and Continua*, vol. 63, pp. 1237–1247, 01 2020.

[2] G. Jaumà Gómez, J. García-Ripoll, and M. Pino, "Exploring quantum annealing architectures: A spin glass perspective," *Advanced Quantum Technologies*, vol. 7, 03 2024.

[3] IEA, "Global energy and climate model," https://www.iea.org/reports/global-energy-and-climate-model, 2023, licence: CC BY 4.0.

[4] G. M. Scheepmaker, H. Y. Willeboordse, J. H. Hoogenraad, R. S. Luijt, and R. M. Goverde, "Comparing train driving strategies on multiple key performance indicators," *Journal of Rail Transport Planning & Management*, vol. 13, p. 100163, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210970619300101

[5] P. Howlett, P. Pudney, and X. Vu, "Local energy minimization in optimal train control," *Automatica*, vol. 45, pp. 2692–2698, 11 2009.

[6] K. K. W. * and T. K. Ho, "Dynamic coast control of train movement with genetic algorithm," *International Journal of Systems Science*, vol. 35, no. 13-14, pp. 835–846, 2004. [Online]. Available: https://doi.org/10.1080/00207720412331203633

[7] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, p. 1484–1509, Oct. 1997. [Online]. Available: http://dx.doi.org/10.1137/S0097539795293172

[8] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, p. 625–644, Aug. 2021. [Online]. Available: http://dx.doi.org/10.1038/s42254-021-00348-9

[9] G. Kalai, Y. Rinott, and T. Shoham, "Google's quantum supremacy claim: Data, documentation, and discussion," 2023.

[10] B. Basu, S. Gurusamy, and F. Gaitan, "A quantum algorithm for computing dispersal of submarine volcanic tephra," *Physics of Fluids*, vol. 36, no. 3, p. 036607, 03 2024. [Online]. Available: https://doi.org/10.1063/5.0189674

[11] J. Golden, A. Bärtschi, D. O'Malley, and S. Eidenbenz, "Numerical evidence for exponential speed-up of qaoa over unstructured search for approximate constrained optimization," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2023, pp. 496–505.

[12] I. Kerenidis, A. Prakash, and D. Szilágyi, "Quantum algorithms for portfolio optimization," *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, Oct 2019. [Online]. Available: http://dx.doi.org/10.1145/3318041.3355465

[13] F. Bova, A. Goldfarb, and R. G. Melko, "Commercial applications of quantum computing," *EPJ Quantum Technology*, vol. 8, no. 1, p. 2, Jan 2021. [Online]. Available: https://doi.org/10.1140/epjqt/s40507-021-00091-1

[14] C. Carugno, M. Ferrari Dacrema, and P. Cremonesi, "Evaluating the job shop scheduling problem on a d-wave quantum annealer," *Scientific Reports*, vol. 12, no. 1, p. 6539, Apr 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-10169-0

[15] K. Kurowski, T. Pecyna, M. Slysz, R. Różycki, G. Waligóra, and J. Weglarz, "Application of quantum approximate optimization algorithm to job shop scheduling problem," *European Journal of Operational Research*, vol. 310, no. 2, pp. 518–528, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221723002072

[16] M. Das, A. Naskar, P. Mitra, and B. Basu, "Shallow quantum neural networks (sqnns) with application to crack identification," *Applied Intelligence*, vol. 54, no. 2, p. 1247–1262, jan 2024. [Online]. Available: https://doi.org/10.1007/s10489-023-05192-1

[17] E. Mitchell, B. BASU, and P. Mitra, "PERFORMANCE ANALYSIS OF a QUANTUM-CLASSICAL HYBRID REINFORCEMENT LEARNING APPROACH," in *The Second Tiny Papers Track at ICLR 2024*, 2024. [Online]. Available: https://openreview.net/forum?id=1POy0o0Z8j

[18] A. Bayerstadler, G. Becquin, J. Binder, T. Botter, H. Ehm, T. Ehmer, M. Erdmann, N. Gaus, P. Harbach, M. Hess, J. Klepsch, M. Leib, S. Luber, A. Luckow, M. Mansky, W. Mauerer, F. Neukart, C. Niedermeier, L. Palackal, R. Pfeiffer, C. Polenz, J. Sepulveda, T. Sievers, B. Standen, M. Streif, T. Strohm, C. Utschig-Utschig, D. Volz, H. Weiss, F. Winter, Q. Technology, and A. C. QUTAC, "Industry quantum computing applications," *EPJ Quantum Technology*, vol. 8, no. 1, p. 25, Nov 2021. [Online]. Available: https://doi.org/10.1140/epjqt/s40507-021-00114-x

[19] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, "Quantum annealing for industry applications: introduction and review," *Reports on Progress in Physics*, vol. 85, no. 10, p. 104001, Sep. 2022. [Online]. Available: http://dx.doi.org/10.1088/1361-6633/ac8c54

[20] K. Ichikawa, "Application of optimization theory for bounded state variable problems to the operation of train," *Bulletin of JSME*, vol. 11, no. 47, pp. 857–865, 1968. [Online]. Available: https://www.jstage.jst.go.jp/article/jsme1958/11/47/11_47_857/_article

[21] P. Howlett, "The optimal control of a train," *Annals of Operations Research*, vol. 98, no. 1, pp. 65–87, Dec 2000. [Online]. Available: https://doi.org/10.1023/A:1019235819716

[22] E. Khmelnitsky, "On an optimal control problem of train operation," *IEEE Transactions on Automatic Control*, vol. 45, no. 7, pp. 1257–1266, 2000.

[23] L. Bittner, "L. s. pontryagin, v. g. boltyanskii, r. v. gamkrelidze, e. f. mishechenko, the mathematical theory of optimal processes. viii + 360 s. new york/london 1962. john wiley & sons. preis 90/–," *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 43, no. 10-11, pp. 514–515, 1963. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19630431023

[24] P. Howlett, "An optimal strategy for the control of a train," *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 1990.

[25] Y. Cao, Z. Zhang, F. Cheng, and S. Su, "Trajectory optimization for high-speed trains via a mixed integer linear programming approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 666–17 676, Oct 2022.

[26] Y. Wang, B. De Schutter, B. Ning, N. Groot, and T. J. J. van den Boom, "Optimal trajectory planning for trains using mixed integer linear programming," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct 2011, pp. 1598–1604.

[27] I. I. Cplex, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[28] C. Wang, W. Liu, Q. Tian, S. Su, and M. Zhang, "An energy-efficient train control approach based on deep q-network methodology," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, Sep. 2020, pp. 1–6.

[29] A. Rocha, A. Araújo, A. Carvalho, and J. Sepulveda, "A new approach for real time train energy efficiency optimization," *Energies*, vol. 11, no. 10, 2018. [Online]. Available: https://www.mdpi.com/1996-1073/11/10/2660

[30] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219. [Online]. Available: https://doi.org/10.1145/237814.237866

[31] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*. Oxford: Oxford University Press, 2007.

[32] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, no. 1, p. 15023, Jan 2016. [Online]. Available: https://doi.org/10.1038/npjqi.2015.23

[33] A. Abbas, A. Ambainis, B. Augustino, A. Baertschi, H. Buhrman, C. J. Coffrin, G. Cortiana, V. Dunjko, D. J. Egger, B. G. Elmegreen, N. Franco, F. Fratini, B. Fuller, J. Gacon, C. Gonciulea, S. Gribling, S. Gupta, S. Hadfield, R. Heese, G. Kircher, T. Kleinert, T. Koch, G. Korpas, S. Lenk, J. Marecek, V. Markov, G. Mazzola, S. Mensa, N. Mohseni, G. Nannicini, C. O'Meara, E. Peña Tapia, S. Pokutta, M. Proissl, P. Rebentrost, E. Sahin, B. C. B. Symons, S. Tornow, V. Valls, S. Woerner, M. L. Wolf-Bauwens, J. Yard, S. Yarkoni, D. Zechiel, S. Zhuk, and C. Zoufal, "Quantum optimization: Potential, challenges, and the path forward," 12 2023. [Online]. Available: https://www.osti.gov/biblio/2229681

[34] 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. [Online]. Available: https://doi.org/10.1007/s10479-022-04634-2

[35] XA0, "List of qubo formulations," 2023, accessed: 19-11-2023. [Online]. Available: https://blog.xa0.de/post/List-of-QUBO-formulations/

[36] P. Toth and D. Vigo, Eds., *The vehicle routing problem*. USA: Society for Industrial and Applied Mathematics, 2001.

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

[38] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, "A review on quantum approximate optimization algorithm and its variants," 06 2023.

[39] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 11 2014.

[40] IBM, "Ibm quantum," 2023. [Online]. Available: https://www.ibm.com/quantum

[41] B. A. Cordier, N. P. D. Sawaya, G. G. Guerreschi, and S. K. McWeeney, "Biology and medicine in the landscape of quantum advantages," *Journal of*

The Royal Society Interface, vol. 19, no. 196, Nov 2022. [Online]. Available: http://dx.doi.org/10.1098/rsif.2022.0541

[42] D. Castelvecchi, "Ibm releases first-ever 1,000-qubit quantum chip," *Nature*, vol. 624, p. 238, 2023. [Online]. Available: https://doi.org/10.1038/d41586-023-03854-1

[43] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error rate," *SIAM Journal on Computing*, vol. 38, 07 1999.

[44] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. Bonilla Ataides, N. Maskara, I. Cong, X. Gao, P. Sales Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin, "Logical quantum processor based on reconfigurable atom arrays," *Nature*, vol. 626, no. 7997, pp. 58–65, Feb 2024. [Online]. Available: https://doi.org/10.1038/s41586-023-06927-3

[45] M. Born and V. Fock, "Beweis des adiabatensatzes," *Zeitschrift für Physik*, vol. 51, no. 3, pp. 165–180, Mar 1928. [Online]. Available: https://doi.org/10.1007/BF01343193

[46] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, Jan. 2018. [Online]. Available: http://dx.doi.org/10.1103/RevModPhys.90.015002

[47] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, "Adiabatic quantum computation is equivalent to standard quantum computation," *SIAM Review*, vol. 50, no. 4, pp. 755–787, 2008. [Online]. Available: https://doi.org/10.1137/080734479

[48] D-Wave Systems Inc., "D-wave quantum computing," 2023, accessed: 2023-11-20. [Online]. Available: https://www.dwavesys.com/

[49] I. Kerenidis, A. Prakash, and D. Szilágyi, "Quantum algorithms for second-order cone programming and support vector machines," *Quantum*, vol. 5, p. 427, Apr. 2021. [Online]. Available: http://dx.doi.org/10.22331/Q-2021-04-08-427

[50] F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney, "Traffic flow optimization using a quantum annealer," *Frontiers in ICT*, vol. 4, Dec 2017. [Online]. Available: http://dx.doi.org/10.3389/fict.2017.00029

[51] dwavesystems, "qbolv," https://github.com/dwavesystems/qbsolv, 2021.

[52] S. Yarkoni, F. Neukart, E. M. G. Tagle, N. Magiera, B. Mehta, K. Hire, S. Narkhede, and M. Hofmann, "Quantum shuttle: traffic navigation with quantum computing," *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures*

and Paradigms for Engineering Quantum Software, Nov 2020. [Online]. Available: http://dx.doi.org/10.1145/3412451.3428500

[53] HERE Technologies, "Here - location based services and mapping platform," 2023, accessed: 2023-11-19. [Online]. Available: https://www.here.com/

[54] G. M. Scheepmaker, R. M. Goverde, and L. G. Kroon, "Review of energy-efficient train control and timetabling," *European Journal of Operational Research*, vol. 257, no. 2, pp. 355–376, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221716307962

[55] F. Gaitan and L. Clark, "Graph isomorphism and adiabatic quantum computing," *Physical Review A*, vol. 89, no. 2, Feb. 2014. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.89.022342

[56] Qiskit contributors, "Qiskit: An open-source framework for quantum computing," 2023.

[57] D-Wave Systems, "Programming the d-wave qpu: Parameters for beginners," https://www.dwavesys.com/media/qvbjrzgg/guide-2.pdf, 09 2020, white paper.

[58] D.-W. S. Inc., "dimod — ocean documentation," https://docs.ocean.dwavesys.com, 2024, accessed on April 16, 2024. [Online]. Available: https://docs.ocean.dwavesys.com/en/latest/index.html

[59] T. Lanting, A. Przybysz, A. Smirnov, F. Spedalieri, M. Amin, A. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, N. Dickson, C. Enderud, J. Hilton, E. Hoskinson, M. Johnson, E. Ladizinsky, N. Ladizinsky, R. Neufeld, T. Oh, I. Perminov, C. Rich, M. Thom, E. Tolkacheva, S. Uchaikin, A. Wilson, and G. Rose, "Entanglement in a quantum annealing processor," *Physical Review X*, vol. 4, no. 2, May 2014. [Online]. Available: http://dx.doi.org/10.1103/PhysRevX.4.021041

A1 Code Examples

A1.1 Qiskit Implementation

A1.1.1 Constant Energy Model

```
def constant_energy_model(
    N: int,  # Number of time steps
    distance_constraint: int,  # Total distance to be covered
    v_maximum: int,  # Maximum allowed velocity
    delta_V: int,  # Velocity increment per time step
    delta_t: int  # Length of a single time step
) -> QuadraticProgram:  # A QuadraticProgram object representing the QUBO problem.
    '''
    Builds a Quadratic Unconstrained Binary Optimization (QUBO) problem
    using Qiskit's QuadraticProgram for train energy optimization.
    '''
    # Create a new QuadraticProgram
    qp = QuadraticProgram()

    # Add binary variables to qp for each time step
    for i in range(N):
        qp.binary_var(name=f"x{i}")

    # Define the objective function
    qp.minimize(linear=[delta_V**2] * N)

    # Constraint 1: Total distance must be covered
    qp.linear_constraint(
        linear=[(N - i) * delta_V * delta_t for i in range(N)],
        sense="==",
        rhs=distance_constraint,
        name="distance_c",
    )

    # V_max constraint
```

```
    qp.linear_constraint(
        linear=[delta_V] * N,
        sense="<=",
        rhs=v_maximum,
        name="Speed_limit_constraint"
    )

    return qp
```

A1.1.2 Quadratic Energy Model

```
# Create a new QuadraticProgram
qp = QuadraticProgram()

# Add binary variables to qp for each time step
for i in range(N):
    qp.binary_var(name=f'x{i}')

linear = {}
quadratic = {}

# Add the linear terms
for i in range(N):
    linear[f'x{i}'] = delta_V ** 2

# Add the quadratic terms
for i in range(1, N):
    for j in range(i):
        quadratic[(f'x{j}', f'x{i}')] = 2 * delta_V ** 2

# Define the objective function
qp.minimize(linear=linear, quadratic=quadratic)

# Constraint 1: Total distance must be covered
qp.linear_constraint(linear=[(N-i) * delta_V * delta_t for i in range(N
    )],
                     sense='==',
                     rhs=total_distance,
                     name='distance_c')

# V_max constraint
qp.linear_constraint(linear=[delta_V] * N,
                     sense='<=',
                     rhs=V_max,
                     name='V_max_c')
return qp
```

A1.1.3 Efficiency Energy Model

```
def energy_program(N, total_distance, V_max, delta_v, delta_t, eta_0, delta_eta, offset=0):
    # Create a new QuadraticProgram
    qp = QuadraticProgram(name='Energy Minimization')

    # Add binary variables to qp for each time ste
    for i in range(N):
        qp.binary_var(name=f'x_{i}')

    linear = {}
    quadratic = {}

    for i in range(0, N):
        linear_key = f'x_{i}'
        linear[linear_key] = linear.get(linear_key, 0) +  eta_0 * (delta_v ** 2)
        for j in range(0, i + 1):
            quadratic_key = (f'x_{i}',f'x_{j}')
            quadratic[quadratic_key] = quadratic.get(quadratic_key, 0) +  delta_eta * (delta_v ** 2)

    qp.minimize(quadratic=quadratic, linear=linear)

    # V_max constraint
    v_max_constraint = {f'x_{i}': delta_v for i in range(1, N)}  # Create a dictionary of coefficients
    qp.linear_constraint(linear=v_max_constraint, sense='<=', rhs=V_max, name='sum_x_equals_v_max')

    # distance travelled constraint
    distance_constaint = {f'x_{i}': delta_v * ((N+offset)-i) * delta_t for i in range(0, N)}
    qp.linear_constraint(linear=distance_constaint, sense='==', rhs=total_distance, name='distance_constaint')

    return qp
```

A1.2 D-Wave implementation

```
class two_bit_alstom:
    def __init__(self, N: int, distance_requirement: int, efficiency: bool=False) -> None:
        self.bqm = BinaryQuadraticModel("BINARY")
        self.N = N
```

```
        self.dv = 1
        self.v_max = 14
        self.distance_requirement = distance_requirement

        # define binary variables
        self._define_binary_variables()
        # define constrains
        self._define_constraints()
        if efficiency:
            print("Applying efficiency")
            self._efficiency_function()

    def _define_binary_variables(self) -> None:
        """
        Define the variables for the the system.

        Modifies:
            self.bqm : adds binary variables x_ia, and x_ib for each time step
        """
        for i in range(self.N):
            self.bqm.add_variable(f"x_{i:03d}a", self.dv**2)
            self.bqm.add_variable(f"x_{i:03d}b", -0.1 * (self.dv**2))

    def _efficiency_function(self, delta_eta:float=-0.01):
        '''Defines the efficiency, i.e. energy redcution for quadratic interaction

        Modifies:
            self.bqm: Defines a linear efficiency function based on previous acceleration
        '''
        for i in range(self.N, 1):
            for j in range(i):
                self.bqm.add_interaction(f'x_{i:03d}a', f'x_{j:03d}a', delta_eta * self.dv**2)

    def _define_constraints(self) -> None:
        """
        Defines the constraints for the system

        Modifies:
            self.bqm: adds constraints
        """
        self._one_state_at_a_time_constraint()
        self._distance_constraint(self.N, self.dv)
        self._speed_limit_constraint(self.N, self.v_max)
```

```
48          self._zero_start_finish_constraint(self.N)
49
50      def _one_state_at_a_time_constraint(self) -> None:
51          """
52          Only one state at a time. i.e. for a time-step onlt one x_ia or
    x_ib can be 1.
53
54          Modifies:
55              self.bqm : adds quadratic interations
56          """
57          for i in range(self.N):
58              self.bqm.add_interaction(f"x_{i:03d}a", f"x_{i:03d}b", 100)
59
60      def _distance_constraint(self, N, delta_v) -> None:
61          """
62          Adds distance constraint to the system
63
64          Modifies:
65              self.bqm : adds distance constraint
66          """
67          distance_constraint = [
68              (f"x_{i:03d}a", (N - i) * delta_v) for i in range(self.N)
69          ] + [(f"x_{i:03d}b", -(N - i) * delta_v) for i in range(self.N)
    ]
70          self.bqm.add_linear_equality_constraint(
71              distance_constraint,
72              constant=-self.distance_requirement,
73              lagrange_multiplier=100,
74          )
75
76
77      def _speed_limit_constraint(self, N, v_max) -> None:
78          """
79          Ensures the velocity at any time step does not exceed v_max *
    dv.
80
81          Modifies:
82              self.bqm: Adds speed limit constraint
83          """
84          max_acc_count = v_max
85
86          speed_limit_constraints = []
87
88          for i in range(1, N + 1):
89              acc_variables = [(f"x_{j:03d}a", 1) for j in range(i)]
90              dec_variables = [(f"x_{j:03d}b", -1) for j in range(i)]
91
```

```
            # Combine the variables for the constraint at this time step
            speed_limit_constraint = acc_variables + dec_variables

            self.bqm.add_linear_inequality_constraint(
                speed_limit_constraint,
                ub=max_acc_count,  # Upper bound is the max_acc_count
                lagrange_multiplier=1000,
                label=f'velocity_limit_at_step_{i:03d}'
            )

    def _zero_start_finish_constraint(self, N) -> None:
        """
        The system must start and end at zero.

        Modifies self.bqm: adds constraint
        """
        speed_zero_constraint = [(f"x_{i:03d}a", 1) for i in range(N)] + [
            (f"x_{i:03d}b", -1) for i in range(N)
        ]
        self.bqm.add_linear_equality_constraint(
            speed_zero_constraint,
            constant=0,
            lagrange_multiplier=100,
        )

    def solve(self, plot: bool = False) -> None:
        sampler = SimulatedAnnealingSampler()

        # Solve the BQM
        sample_set = sampler.sample(self.bqm, num_reads=8192, num_sweeps=8192)

        # Extract and analyze the best solution
        self.best_sample = sample_set.first.sample
        self.best_energy = sample_set.first.energy
        self.solution_vector = list(self.best_sample.values())

        print("Best Sample (Solution):", self.best_sample)
        print("Energy of the best sample:", self.best_energy)
        print("=" * 50)

        if plot:
            self._plot_velocity_changes()

    def solve_leap(self, plot: bool = False) -> None:
```

```
        sampler = LeapHybridSampler()
        sample_set = sampler.sample(self.bqm)

        # Extract and analyze the best solution
        self.best_sample = sample_set.first.sample
        self.best_energy = sample_set.first.energy
        self.solution_vector = list(self.best_sample.values())

        print("Best Sample (Solution):", self.best_sample)
        print("Energy of the best sample:", self.best_energy)
        print("=" * 50)

        if plot:
            self._plot_velocity_changes()

    def solve_direct(self, plot: bool = False) -> None:
        sampler = EmbeddingComposite(DWaveSampler())
        solution = sampler.sample(self.bqm, label='Demo', num_reads
=4096)

        best_sample = solution.first.sample
        best_energy = solution.first.energy
        self.solution_vector = list(best_sample.values())

        print("Best Sample (Solution):", best_sample)
        print("Energy of the best sample:", best_energy)
        print("=" * 50)
        if plot:
            self._plot_velocity_changes()
            show(solution)


    def _plot_velocity_changes(self) -> None:
        """
        Plots the velocity changes over time steps for multiple
solutions on the same graph, interpreting
        each pair of binary values as acceleration or braking actions.

        Parameters:
        - solution_data_list: A list of tuples, each containing (list[
int], str) where
        the list is the binary solution vector and the str is the
solver's name.
        - delta_v: The velocity change unit per acceleration or braking
 step.
        """
        plt.figure(figsize=(10, 6))
```

```
178
179         # Time steps are defined by the length of the solution vector
    divided by 2 (since 2 bits per time step)
180         time_steps = len(self.solution_vector) // 2
181         velocity = [0] * time_steps
182         for i in range(time_steps):
183             # Extract acceleration (a) and braking (b) states for the
    current time step
184             a = self.solution_vector[2 * i]
185             b = self.solution_vector[2 * i + 1]
186             # Update the velocity: increase for acceleration (10),
    decrease for braking (01)
187             velocity_change = (a - b) * self.dv
188             if i > 0:
189                 velocity[i] = velocity[i - 1] + velocity_change
190             else:
191                 velocity[i] = velocity_change
192
193         plt.plot(range(time_steps), velocity, label=f"Velocity", marker
    ="o", color='#666666', markersize=4)
194
195         plt.title("Velocity vs Time")
196         plt.xlabel("Time Step")
197         plt.ylabel("Velocity (m/s)")
198         plt.legend()
199         plt.grid(True)
200         plt.show()
201
202
203     def __str__(self) -> str:
204         return str(self.bqm)
```