Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Optimisation of Exam Timetabling through Quantum Annealing

Author:
Dhivyarupini Ravindran

Supervisor:
Prof. Herbert Wiklicky

Second Marker:
Prof. Alessio Lomuscio

February 3, 2023

Abstract

Despite large variations in the types of quantum computers currently being studied and developed, practical computational metrics surrounding these studies have predominantly been confined to manufactured, toy problems.

In the last few decades, D-Wave's Quantum Annealers have placed claim to commercial success in applying their new adiabatic quantum computing algorithm to solve well-known practical optimisation problems. The implications of which, are significant improvements in terms of computational time and memory over classical algorithms. However, due to D-Wave's implementation of quantum behaviours in their Quantum Processing Unit (QPU), they faced a significant amount of skepticism concerning the validity of the Quantum Annealer.

In this project, we present a proof-of-concept application that optimises exam timetabling problems using this Quantum Annealer. Here, we describe our mapping of the exam timetabling problem into a Binary Quadratic Model, which can be embedded to the QPU. We also investigate the relationship between the quality of the embedding and the overall performance of the Quantum Annealing process.

Additionally, we investigate the capabilities of a quantum-classical optimisation approach. Our hybrid solver successfully produces valid solutions for large, well-known exam timetabling problems from the University of Toronto Benchmark Dataset. The performance of the final solutions produced by our solvers were comparable to that of the industry-standard benchmarks.

Acknowledgements

I would like to thank my supervisor, Prof. Herbert Wiklicky, for his constant support and advice throughout my project. I greatly appreciated the knowledge and guidance he had to offer during our meetings, as well as the freedom and encouragement I was given to take this project in directions that interests me.

I would also like to thank my friends, without whom, these last four years at Imperial would have been without a great deal of laughter. Your friendship has made all the difference.

Finally, I would like to thank my family for all their sacrifices, unwavering support, and constant enthusiasm about my degree, especially on the days where I had none. I hope that through reading this thesis, you will finally have an idea what I have been studying for the last few years.

Contents

1 Introduction **4**
- 1.1 Motivation . . . 4
- 1.2 Objectives . . . 4
- 1.3 Contributions . . . 5
- 1.4 Thesis Structure . . . 5

2 Ethics Considerations **6**

3 Background **7**
- 3.1 Optimisation of Exam Timetabling . . . 7
  - 3.1.1 Classical Optimisation methods . . . 7
  - 3.1.2 Examination Timetabling . . . 8
  - 3.1.3 Benchmark Datasets . . . 9
- 3.2 Quantum Annealing . . . 10
  - 3.2.1 From Simulated Annealing to Quantum Annealing . . . 10
  - 3.2.2 The D-Wave Quantum Annealer . . . 11
  - 3.2.3 Formulating problems for D-Wave's QPU . . . 12
  - 3.2.4 Quantum Physics underlying the Annealing Process . . . 15
  - 3.2.5 Comparing Quantum Annealing and Gate Quantum Computers . . . 17

4 Optimising Exam Timetabling through Quantum Annealing: Implementation **19**
- 4.1 Exam Timetabling Problem Description . . . 19
  - 4.1.1 Getting the problem's QUBO formulation . . . 20
- 4.2 High-level Workflow . . . 22
- 4.3 Preparation of Exam Timetabling Problems . . . 22
- 4.4 Problem Formulation . . . 23
  - 4.4.1 Preprocessor: Representing the Problem Graphically . . . 23
  - 4.4.2 BqmBuilder: Encoding the QUBO onto a Binary Quadratic Model (BQM) 25
- 4.5 Sampling . . . 29
  - 4.5.1 Minor Embedding . . . 29
  - 4.5.2 D-Wave's Samplers . . . 30
  - 4.5.3 Access to D-Wave's Quantum Samplers . . . 31
- 4.6 Validation and Visualisation of Results . . . 31

5 Optimising Exam Timetabling through Quantum Annealing: Evaluation **33**
- 5.1 Aims . . . 33
- 5.2 Comparing initial results to Simulated Annealing (SA) . . . 33
  - 5.2.1 Analysing QA solutions . . . 35

5.3 Improving QA Performance . . . 36
5.3.1 Optimising the Annealing Process . . . 36
5.3.2 Improving the Problem Embedding . . . 36
5.3.3 Other improvements considered . . . 40
5.4 Final Results and Summary . . . 42

6 Optimising Exam Timetabling through Hybrid Quantum Annealing: Implementation & Evaluation 44
6.1 Aims . . . 44
6.2 Preparing the University of Toronto Benchmark Dataset . . . 44
6.3 Extending Quantum Annealing with Classical Postprocessing . . . 45
6.4 Hybrid Solvers . . . 46
6.4.1 dwave-hybrid . . . 46
6.4.2 Leap's Hybrid Quantum Sampler . . . 48
6.5 Final Results and Summary . . . 48

7 Conclusion 51
7.1 Conclusions . . . 51
7.2 Future Work . . . 52

Chapter 1

Introduction

1.1 Motivation

It has been long noted that quantum computers could provide significant improvements in terms of computational time and memory over a classical computer [1, 2]. These improvements have particularly beneficial implications for computational tasks involving a large number of parameters and potential configurations. As such, many have speculated about the potential benefits to be gained by the field of combinatorial optimisation. This research into optimisation heuristics has been a popular focus in fields of Mathematics, Computer Science and Economics for centuries, and is the focal point of this project.

Current classical methods of optimisation have three main downfalls: (1) the exponential increase in time complexity as problem size increases, (2) the difficulty in exploring a massive (or potentially infinite) configuration space and (3) the inability to move from a configuration to a better one due to the local minima trap. A new approach to quantum computing called Quantum Annealing claims to have seen some success in tackling these issues through the implementation of quantum behaviours such as quantum tunneling, superposition and qubit entanglement. Due to the nature of these claims, this adiabatic quantum computing approach has faced a significant amount of debate and skepticism concerning the validity of its quantum behaviours since it has been made commercially available by D-Wave Systems in the last few decades.

1.2 Objectives

The objective of this project is to investigate the applicability of Quantum Annealing to a specified optimisation problem, which we have chosen to be the exam timetabling problem. This will be achieved through:

- Building a **proof of concept** application to solve the exam timetabling optimisation problem using D-Wave System's Quantum Annealer.
- Investigating the extent of the capacity of D-Wave's Quantum Processing Unit (QPU) in embedding an optimisation problem, as well as the feasibility of finding a problem's optimal embedding.
- Analysing the quality of the solution produced by the Quantum Annealer against Classical and Hybrid methods.
- Exploring the capabilities and performance of a Quantum-Classical Hybrid Solver.

1.3 Contributions

This thesis has the following contributions:

1. An Exam Timetabling application that produces optimised schedules using a new Quantum Annealing approach by formulating the problem as a Binary Quadratic Model and embedding it to D-Wave's Quantum Processing Unit (QPU) which outputs low-energy samples.

2. Analysis of how embedding of Binary Quadratic Model to the Quantum Processing Unit affects the performance of the Quantum Annealer.

3. Evaluation of the maximum size of exam timetabling optimisation problems that can be embedded and solved on both the D-Wave 2000Q computer and the Advantage System.

4. Utilisation of D-Wave Leap's Quantum-Classical Hybrid Solvers to find solutions to the well-known University of Toronto Benchmark Dataset of exam timetabling problem. Our hybrid method managed to find valid solutions for large, complex exam timetable problems which performed in a manner comparable to industry-standard benchmarks.

1.4 Thesis Structure

In Chapter 2, we discuss the ethical implications of this project.

In Chapter 3, we outline the relevant work that provides the context for this project and explain the concepts needed to form background knowledge for the later chapters.

In Chapter 4, we describe the implementation of optimising exam timetables using D-Wave's Quantum Annealer.

In Chapter 5, we evaluate the results of the quantum annealer against classical algorithms. We then analyse the results and investigate how the embedding of the problem to the QPU affects performance. We use this investigation to make improvements to our Quantum Annealing process and finally, evaluate the results of this improved method against the initial approach.

In Chapter 6, we discuss the implementation of multiple Quantum-Classical Hybrid methods in optimising exam timetabling problems, particularly in larger, more complex problems from the University of Toronto Benchmark Dataset.

In Chapter 7, we discuss the successes and limitations of the methods we investigated and draw the project to a conclusion.

Chapter 2

Ethics Considerations

This project has little ethical, societal, or legal issues. No personal information was needed or used at any point in the process as all datasets were either publicly available or artificially generated. Therefore, there was no need for collection, retention or processing of any individual data.

Chapter 3

Background

3.1 Optimisation of Exam Timetabling

In this section, we aim to give context to the optimisation problem chosen to study in this project, the exam timetabling problem. The following sections investigate classical optimisation methods (3.1.1), history of the exam timetabling problem and techniques developed to tackle it (3.1.2) as well as some notable benchmark datasets in the industry (3.1.3).

3.1.1 Classical Optimisation methods

From the theory of optimal allocation of resources by Kantorovich and Koopmans [3] to the landmark development of the simplex tool by Dantzig [4], the roots of combinatorial optimisation have numerous claims. This is due to the nature of optimisation problems, which present themselves in the practice of even the most monotonous tasks. For instance, the Travelling Salesman Problem[1], arguably the most famous combinatorial optimisation problem, can be seen cropping up when planning a shopping trip, or holiday activities.

Methods to solve such problems entail finding extremum (maximum or minimum) values of a cost function across an (often large) configuration space. The aim of these methods are to find the set of variables that minimises the cost function and produces the optimal solution. In most cases, explicitly solving the cost function is impossible and manually searching the entire configuration space with a *brute-force* method is highly unfeasible due to time and resource constraints. *Greedy Optimisation* algorithms have also been explored. These algorithms involves consistently following a specific problem-solving heuristic, and consequently making the optimal choice at each iteration. In problems where there is only 1 local minimum, these algorithms are good options as they are guaranteed to find the optimal solution and terminate in a reasonable amount of time. However, this is often not the case as most cost functions are complex and will contain multiple local minimums. Greedy algorithms have no way of confirming that the minimum it has found is the global minimum.

In this case, *Stochastic Optimisation* techniques are more effective. They employ a method of introducing randomness to the search of the configuration space instead of always picking the most optimal choice. Introduced by Kirkpatrick et al. in 1983 [5], *Simulated Annealing* is a stochastic optimisation technique for finding the global minimum of a given objective function. It is modelled after the physical process of heating a metal followed by controlled cooling to minimise the overall energy of the system. As opposed to always evolving in the direction to reduce the system's energy, the notion at the core of simulated annealing is a non-zero

[1] https://en.wikipedia.org/wiki/Travelling_salesman_problem

probability at any time step to search the configuration space in the direction that increases the energy. This non-zero probability represents thermal fluctuations in the overall system and is characterised as an artificial temperature parameter T. The significance of T is to allow the system to escape from local minima by climbing over an energy barrier (Figure 3.1) and consequently, the system explores a larger configuration space. This parameter is initialised to a high value but gradually reduced throughout the annealing process. Therefore, at the end of the process when $T = 0$, we expect to find the configuration that gives the global minimum energy state. This method avoids the traps of local minima that other optimisation algorithms famously fall into as they prioritise minimisation of the objective function at each iteration.

In *ergodic*[2] systems where it is possible visit all states, simulated annealing is effective in finding the global minimum of an optimisation problem. However, this is not always the case. For example, consider a cost function that has energy barriers that take $O(N)$ time steps to overcome. When $N \to \infty$, at any finite value of T, the system will take infinite time to find the global minimum. Such NP-hard[3] problems do not have classical algorithms to find existing solutions. This issue is among the motivations for using quantum annealing to find solutions to optimisation problems.

3.1.2 Examination Timetabling

Timetabling problems are a staple combinatorial optimisation problem, having been investigated in multiple forms over the years (e.g. nurse scheduling, job shop scheduling, transportation timetabling). Among them, academic timetabling problems have occupied a significant proportion of focus in this area of research. This is as it is one of the the most important tasks with a guarantee of cropping up periodically in all academic institutions. Another motivation is that it also greatly impacts a large number of stakeholders including students, invigilators, administrators and so on.

Within academic timetabling itself, there exists many variants including class-lecturer scheduling, course timetabling and exam timetabling. This study found many similarities between course and exam timetabling [7], though significant differences can exist, as each problem is defined slightly differently in each paper. This project will focus on examination timetabling.

Graph Based Techniques for Examination Timetabling

Though there are a significant number of techniques researched for examination timetabling, this 1967 study by Welsh and Powell was the first to draw the link between graph colouring problems and timetabling [8]. This can be done by mapping examination timetabling problems to a graph where each node represents an exam, and an edge between nodes represent if two exams are clashing (i.e. two exams that have shared resources, such as students). As the aim of graph colouring is to colour each node so no two adjacent nodes have the same colour, when solving this graph for the graph colouring problem, we can set each different colour to a different time slot. This will produce a schedule with no clashing exams sharing the same time slot. Though this is a trivial version of an examination timetabling problem, which usually has to be optimised for other additional soft constraints, it is a good starting point in understanding our optimisation problem and moving forward with new techniques.

[2]Idea of ergodicity in classical spin glasses as explained in Quantum Annealing and Related Optimization Methods [6].

[3]https://en.wikipedia.org/wiki/NP-hardness

3.1.3 Benchmark Datasets

Due to the significant amount of interest in examination timetabling research, multiple benchmark datasets have been established. This provides the opportunity for meaningful comparisons to be made between different research methods and techniques into solving the same problem.

University of Toronto Benchmark Data

This 1996 study introduced 13 exam timetabling problems from a collection of Canadian, American, British and Saudi Arabian high schools and universities [9]. While this dataset has been widely studied, there has been confusion as multiple versions of a few of the problems in the datasets have been circulating and studied in different capacities.

Given this confusion, and that this paper investigates new approaches on this dataset in Chapter 6, we have detailed which versions of each problem we had considered and used to test the approaches in table 3.1 below.

Dataset	Version	Number of Exams	Number of Days	Number of Students	Conflict Density
UTE92	1	184	10	2750	0.08
YOR83	3	180	21	919	0.3
UTA92	2	638	35	21329	0.12
TRE92	1	261	23	4360	0.18
STA83	3	138	35	549	0.19
RYE92	1	482	23	11483	0.07
PUR93	1	2419	42	30029	0.03
LSE91	1	381	18	2726	0.06
KFU93	1	461	20	5349	0.06
HEC92	2	80	18	1108	0.42
EAR83	3	189	24	1108	0.27
CAR92	1	543	32	18419	0.14
CAR91	1	682	35	16925	0.13

Table 3.1: Description of Toronto Benchmark Dataset

A notable characteristic of problems in the Toronto Benchmark Dataset detailed in table 3.1 is *Conflict Density*. The conflict density is calculated from the *Conflict Matrix*, which is a square matrix where each term $C_{i,j} = 1$ if exams i and j have shared students, and $C_{i,j} = 0$ otherwise. Conflict Density is the ratio between the number of pairs of conflicting exams and the overall number of terms in the conflict matrix. It is a good measure of complexity as it directly correlates to the constraints of the exam timetabling problems.

Other Benchmark Data

Other benchmark datasets for exam timetabling include:

- This 1994 exam timetabling dataset from the **University of Nottingham** that was introduced by Burke, Newall and Weare [10].
- Exam timetabling datasets from the **University of Melbourne** introduced during the Practice and Theory on Automated Timetabling (PATAT) Conference in 2002 [11].

These problems were considered but we decided not to use them for this project as the complexity and size of the problems were too large. Due to the hardware limitations of the Quantum Computer, it is not currently possible to solve problems of this size through quantum annealing.

3.2 Quantum Annealing

In this section, we discuss the quantum annealing technique that aims to build upon and improve methods of tackling combinatorial optimisation problems. This includes an introduction to the concept (3.2.1), the D-Wave quantum annealer and its implementation (3.2.2), formulating problems for the QPU (3.2.3), the underlying quantum physics of quantum annealing (3.2.4) and how it compares to the generic gate quantum computer model (3.2.5).

3.2.1 From Simulated Annealing to Quantum Annealing

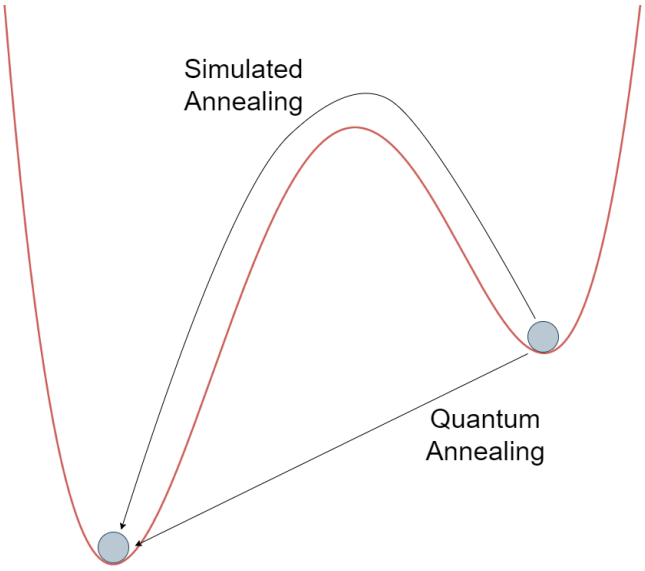


Figure 3.1: How particles move from local to global minima in Simulated vs Quantum Annealing

The present day form of Quantum Annealing was formulated in the late 1990s [12] by T. Kadowaki and H. Nishimori. Through their work, quantum fluctuations were introduced to the preexisting simulated annealing process, with the aim of refining the optimisation process for unconstrained and bound-constrained problems. Following the concept of physics that everything is constantly in search of its minimum energy state, quantum annealing uses concepts such as quantum tunnelling to find the minimum energy state of an optimisation problem. This results in an optimal (or near-optimal) solution. Similarly to thermal fluctuations in the classical simulated annealing process, quantum fluctuations also help the system escape from local minima. However, instead of a particle having to move over an energy barrier, *quantum tunneling* allows it to penetrate across the barrier. This difference is visualised in Figure 3.1, showing the difference and possible advantage quantum annealing has over classical simulated annealing.

3.2.2 The D-Wave Quantum Annealer

Table 3.2: Timeline of D-Wave System Products

Feb 2007	D-Wave demonstrates the Orion system, the first alleged quantum computer, and its applications on finding drug compounds given a molecule database, computing a compatible seating arrangement, and solving a Sudoku puzzle.
Dec 2009	D-Wave's processor is used by a research team at Google to train a binary image classifier. [13]
May 2011	D-Wave Systems announces D-Wave One [14], the first commercial quantum computer system with a 128-qubit processor. Its processor, known as "Rainier", was used to perform a discrete optimisation operation.
	D-Wave Systems and Lockheed Martin sign multi-year contract [15] entailing the purchase of the D-Wave One quantum computer along with professional services from D-Wave.
May 2012	D-Wave Systems announces D-Wave Two, a commercial 512-qubit quantum computer named "Vesuvius".
Aug 2012	Researchers at Harvard University use D-Wave One quantum computer to solve lattice protein folding model [16].
May 2013	Catherine McGeoch shows significant improvement in performance of the D-Wave computer with 439 qubits over CPLEX algorithms on classical machines. [17]
Feb 2014	A study by Matthias Troyer and Daniel Lidar shows no significant speed increase of the D-Wave One to classical computers when solving the same discrete optimisation problem. [18]
Aug 2015	D-Wave announces the D-Wave 2X, running on 1000 qubits. It was paired with a report [19] comparing speeds of the quantum computer to that of high-end single threaded classical computers. This processor was based off a 2048-qubit chip that disabled half of its qubits. The consequent D-Wave 2000Q processor activated these qubits [20].
Feb 2019	D-Wave announces a "Pegasus" commercial quantum processing chip with over 5000 qubits and reduced noise. [21]

D-Wave System is the world's leading company in quantum annealing, having been the only company offering a commercial quantum annealer for the better part of the last 2 decades. Table 3.2 shows the evolution of D-Wave system products in the market, along with some of its ground-breaking applications and the skepticism its products faced. This skepticism is largely due to D-Wave implementing a quantum annealer over the generic Gate Quantum Computer.

D-Wave's implementation of Quantum Annealing in their Quantum Processing Unit (QPU) can be understood through the example shown in Figure 3.2. This example shows how quantum annealing is carried out on 1 *qubit*. A qubit, or quantum bit, is the lowest quantum energy state that forms the D-Wave QPU. It can occupy a classical state such as being in the 0 state or 1 state and also be in a quantum state. That is a superposition state of both 0 and 1 at the same time. Qubits also have circulating current around them and an associated magnetic field. This

magnetic field can be manipulated by applying an external magnetic field. The qubit shown in part (a) of Figure 3.2 is in a superposition state of both 0 and 1 at the same time.

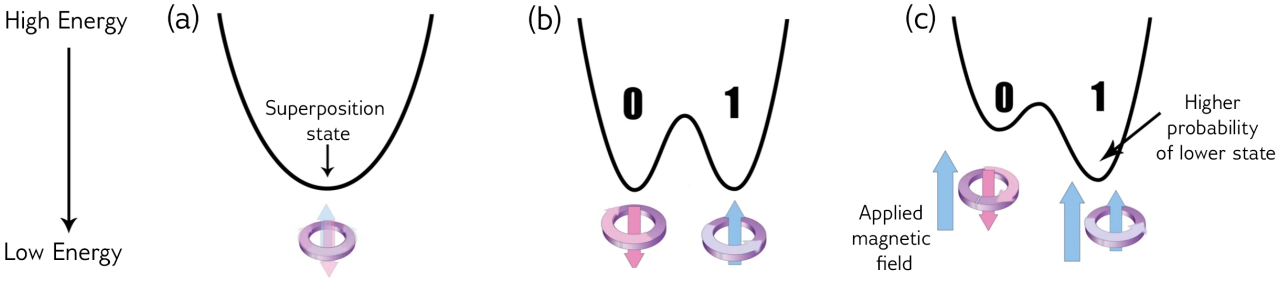


Figure 3.2: Taken from D-Wave System Documentation [22]

Figure 3.2 shows the energy landscape throughout the quantum annealing process. This energy landscape is a representation of the cost function from the optimisation problem that we are trying to solve. To begin with, the qubit is in its superposition state in part (a). As the system carries out the anneal, we see the barrier being raised in part (b). Here, the energy landscape turns into what is known as double-well potential, where the probabilities of the qubit collapsing into either valley are equal. However, as our cost functions would generally favour one configuration over the other, we want to manipulate the energy landscape to better represent this difference in configuration. This is done by using a programmable quantity called a *bias* to apply a magnetic field to the qubit. This process can be seen in Figure 3.2 (from (b) to (c)) as the double-well potential tilts in favour of the 1 state. By tilting it in this direction, there is a higher probability the qubit will collapse into the state with the lowest energy. At the end of the anneal process, the qubit has collapsed into a classical state. This represents the configuration of the optimal value of this optimisation problem.

As described in the quantum annealing process above, we use the bias term to describe the cost function of the optimisation problem. However, the example above only considers a small scale quantum annealing operation. On a larger scale, the bias term alone is not sufficient to influence the energy landscape. This is done by linking qubits together with *couplers*. Couplers control the influence linked qubits have on each other, and this influence can be controlled by a programmable coupler strength. Couplers represent quantum entanglement in the D-Wave system. 2 entangled qubits represent one object that has 4 states: (0,0), (0,1), (1,1), and (1,0). The bias term of each qubit along with the coupler strengths define the energy for each of these states. This combination of biases and coupler strengths is what allows programmers to define the energy landscape of the optimisation problem when using the D-Wave quantum computer. The annealer then tries to find the minimum energy of the energy landscape and returns the configuration for this point, which will be the optimal solution.

3.2.3 Formulating problems for D-Wave's QPU

We have just seen how the quantum annealing process works for one qubit through the definition of biases and coupler strengths. In this section, we will expand this idea to describe how we can specify biases and coupler strengths for a larger, more practical problem. This process is shown on a high level in figure 3.3 and explored below.

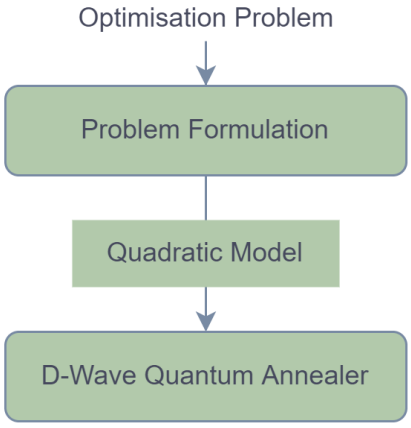


Figure 3.3: Mapping an optimisation problem to the QPU

D-Wave's supported Quadratic Models

To use D-Wave's Quantum or Classical-Quantum Hybrid Annealers, we first need to formulate our optimisation problem's objective function as one of three quadratic models. These are:

- **Binary Quadratic Models (BQMs):** BQMs are used to define unconstrained problems that have binary variables (decisions where variables could have values of either True or False). They represent the constraints of a problem through a penalty model. Binary Quadratic Models can encode the optimisation problems as both the **Ising** and **QUBO** (Quadratic Unconstrained Binary Optimisation) models.

  The **Ising model** represents an objective function where variables correlate to physical Ising spins. Therefore, for each variable v_i, $v_i \in \{-1, +1\}$, linear coefficients h_i represent the biases and quadratic coefficients $J_{i,j}$ the interactions between spins (coupler strengths).

$$Ising: \qquad E(v) = \sum_{i=1} h_i v_i + \sum_{i<j} J_{i,j} v_i v_j \qquad v_i \in \{-1, +1\}$$

  Similarly, the **QUBO model** represents an objective function of binary variables: $v_i \in \{0, 1\}$. Upper-diagonal matrix Q represents the coefficients of each variable to each other. Its diagonal terms are linear coefficients of each variable to itself and its off-diagonal terms represent the quadratic coefficients between different variables.

$$QUBO: \qquad E(v) = \sum_{i \leq j} v_i Q_{i,j} v_j \qquad v_i \in \{0, 1\}$$

  As we can see, the BQM equation:

$$E(v) = \sum_{i=1} a_i v_i + \sum_{i<j} b_{i,j} v_i v_j + c \qquad v_i \in \{-1, +1\} \textbf{ or } \{0, 1\}$$

  can represent both. Therefore, we can specify problems in either form for a BQM, as conversion is trivial between them [23].

- **Constrained Quadratic Models (CQMs):** CQMs are used for constrained problems that have real integer or binary variables, and one or more constraints. CQMs are the abstractions of BQMs, Therefore, there are two main advantages of CQMs over BQMs. These are that: (1) BQM variables can only have 2 states, while CQMs can accept variables that are binary, integer or real with max values of up to $\pm 2^{53}$, (2) While constraints are represented using a penalty model above in BQMs, these can be stated natively in CQMs.

- **Discrete Quadratic Models (DQMs):** DQMs are used to optimise over various discrete variables. Each variable can be defined by up to 65,000 possible values. Constraints are specified by adding penalties to a defined objective function.

Due to the QPU limitations, Binary Quadratic Models are currently the only quadratic model being supported by the Quantum Annealers and therefore, this is the form that we deal primarily with in this project. These models, when passed to the QPU, will represent the biases and coupler strengths that defines the energy landscape of the problem. This is done by mapping the linear coefficients of the model to the biases and the quadratic coefficients to the coupler strengths.

While the Quantum Samplers only accept BQMs, D-Wave's Hybrid Solvers also accept problems formulated in CQMs and DQMs.

D-Wave QPU Topologies

Understanding the architecture of the D-Wave QPU is critical when formulating the objective function of a problem. This is as the quadratic model passed to the Quantum Annealer, like in figure 3.3, will need to be embedded to D-Wave's QPU. It should be noted that D-Wave does provide software that automates this embedding. It maps the quadratic model's linear and quadratic coefficients to qubit biases and coupler strengths on the QPU. However, understanding the different topologies is key to comprehending the performance of the QPU on different types of problems, as the structure of the QPU has implications for (1) problem-graph size and (2) quality of a solution. The QPU topologies we will the Chimera topology of the D-Wave 2000Q computer, and the Pegasus topology of the Advantage System.

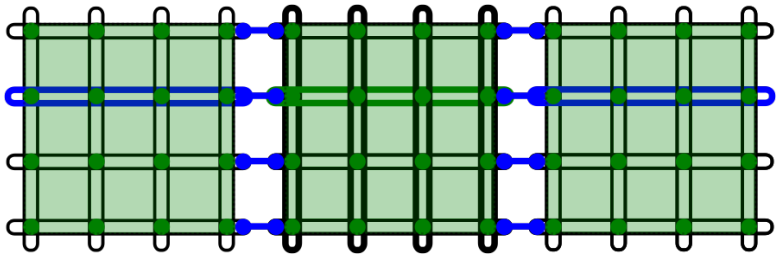


Figure 3.4: Chimera Topology showing external couplers (blue) and internal couplers (green). This image was taken from D-Wave System Documentation [24]

The **Chimera topology** represents horizontal and vertical loops of qubits as we can see in Figure 3.4. This figure shows a row of 12 vertical qubits and three columns of 12 horizontal qubits. Its internal couplers (coloured in green), connect qubits to orthogonal qubits, while external couplers (coloured in green) connect qubits to qubits in the same row. The overall capacity of the Chimera structure QPU in D-WAve 200Q is 2048 qubits and 6016 couplers.

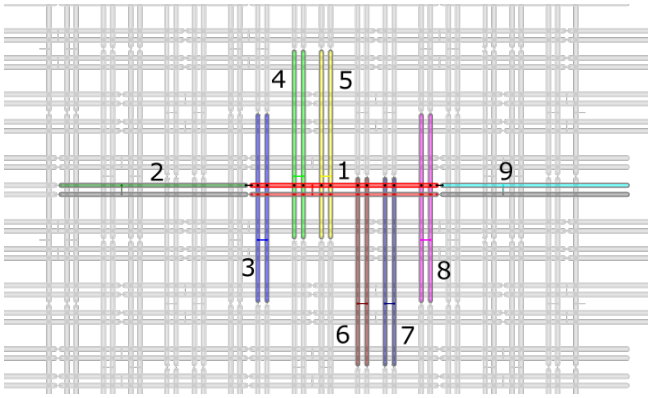


Figure 3.5: Pegasus Topology showing external couplers (between 1 2 and 1 9), internal couplers (1 with pairs 3 through 8) and odd couplers (shown in a differnt colour between each pair). This image was taken from D-Wave System Documentation [24]

Meanwhile, the **Pegasus topology** represents horizontal and vertical loops of qubits as we can see in Figure 3.4. This figure shows internal couplers connect qubits to orthogonal qubits, external couplers connecting qubits to qubits in the same row and a new odd coupler that connects similarly aligned pairs of qubits to each other. The overall capacity of the Pegaus structure in the Advantage system is 5760 qubits.

3.2.4 Quantum Physics underlying the Annealing Process

Having embedded the problem to the QPU by setting bias and coupling values, we can begin the anneal process. This section gives a brief overview of the quantum physics underlying D-Wave's Quantum Annealer.

Initially, the optimisation problem is framed as an energy minimisation problem. This is done by representing the problem's cost function as a BQM, which is a classical *Hamiltonian*[4]. This is also known as the *problem Hamiltonian* defined of the form (3.1). Here, h_i and $J_{i,j}$ represent the bias terms of the qubits and the coupler strengths between them.

$$H_{prob} = \frac{B(s)}{2}\left(\sum_i h_i \sigma_z^{(i)} + \sum_{i>j} J_{i,j}\sigma_z^{(i)}\sigma_z^{(j)}\right) \tag{3.1}$$

We then choose a well-suited *tunnelling Hamiltonian* of the form (3.2), of which the lowest-energy state occurs when all qubits are in superposition state. This Hamiltonian defines the quantum fluctuations that allow the system to escape the local minima traps. The expected effect of the quantum tunelling term is to make the barriers appear transparent to the system so that eventually, it will be possible for the system to visit any configuration with finite probability [6].

$$H_{tunnel} = -\frac{A(s)}{2}\left(\sum_i \sigma_x^{(i)}\right) \tag{3.2}$$

Adding the two terms (3.1) and (3.2) gives us the Hamiltonian (3.3) of our quantum system.

$$H = H_{tunnel} + H_{prob} \tag{3.3}$$

[4]A Hamiltonian maps eigenstates to energies in a quantum system.[22] The energy is only well-defined if the system is in one of the eigenstates.

The quantum annealing process, which happens between time $t = 0$ and $t = t_f$ can be parametrised with s. s represents the normalised quantum annealing fraction that takes a value between 0 and 1. t_f is also known as the annealing time and can be specified as a parameter in D-Wave's system. This process starts by initialising the system to the lowest-energy *eigenstate* of the tunneling Hamiltonian, which is when all qubits are in superposition state. At this point ($t = 0$, $s = 0$), $A(s) >> B(s)$ as bias terms and couplings have not been applied to the qubits. Then, the problem Hamiltonian, which contains information on the cost function that is represented as biases and coupler strengths, is then gradually introduced. This slowly anneals the system by simultenously decreasing the influence of the tunneling Hamiltonian $A(s)$, while it increases $B(s)$. In the default annealing schedule, this process continues gradually until time t_f where $A(s) << B(s)$. This means that the tunneling Hamiltonian tends to zero ($H_{tunnel} \rightarrow 0$) to return the problem Hamiltonian H_{prob}. When the anneal process ends, the system will be in one of the eigenstates of the problem Hamiltonian. This is ideally the global minimum state, which is the optimal configuration of variables to the optimisation problem.

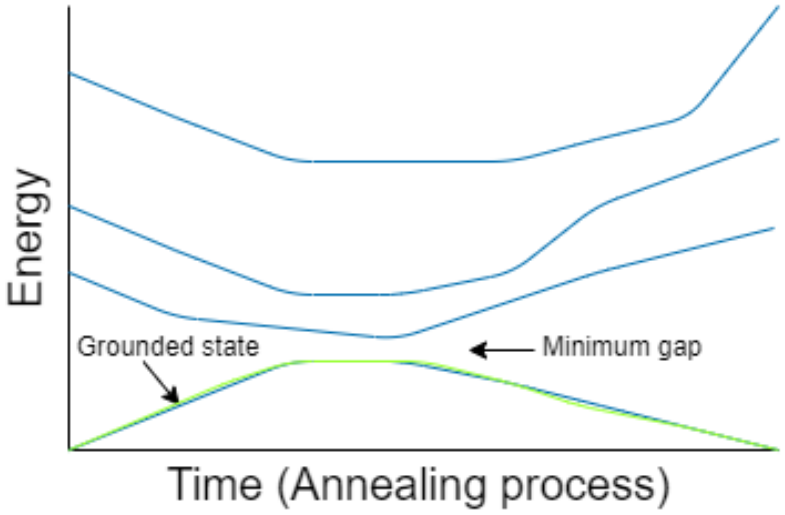


Figure 3.6: Ideal energy of system throughout anneal

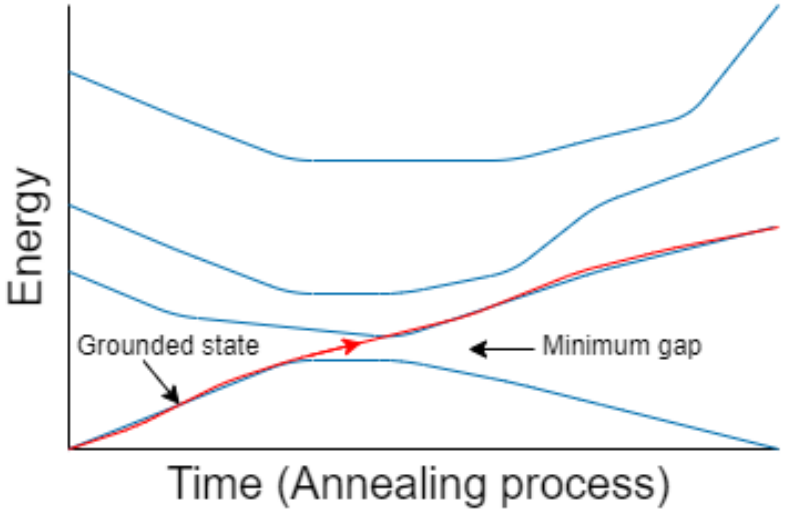


Figure 3.7: System jumping from ground to excited state

Throughout the annealing process, the ideal case would be for the system to remain in the lowest energy state at all times as shown in Figure 3.6. This is called the ground state and is visualised in figures 3.6 and 3.7 as the lowest energy line. The system begins in the lowest energy state due to the tunneling Hamiltonian. Throughout the anneal, as the system is introduced to the problem Hamiltonian, other eigenstates of the problem Hamiltonian may have energy levels that are closer to the ground state. When this occurs, it gets easier for the system to jump from the ground state into one of the excited states, as shown in Figure 3.7. The smallest distance between the ground start and the lowest (energy) excited state is called the minimum gap. The minimum gap represents the part of the annealing process with the highest probability for the system to jump to an excited state. In terms of optimisation problems, the smaller the minimum gap, the more difficult the problem.

This jump could be caused by a number of factors such as thermal fluctuations of the physical computational system or an accelerated rate at which the annealing process is carried out. The latter can be combated by running the annealing process at a slower and more controlled rate. By doing this, and carrying out the annealing process in a completely isolated environment with no external interference, we have an *adiabatic process*[5]. It should be noted that this is a theoretical idea as no physical computation can achieve this, and thus the quantum annealing process usually returns a low-energy state instead of the global minimum.

3.2.5 Comparing Quantum Annealing and Gate Quantum Computers

A Universal Gate Quantum Computer uses reliable qubits that can put together any sequence of quantum circuit[6] operations. Owing to the flexibility of building these quantum circuits, it can be used to run all sorts of algorithms, including increasingly complex ones. This is in comparison to a smaller subset of optimisation problems that can be solved with quantum annealing. An example of this is Shor's algorithm[7], an integer factorisation algorithm used to break modern cryptography which can be run in polynomial type on a gate quantum computer but not on a quantum annealer. This is as the algorithm requires state changes that are complex, which a quantum annealer cannot handle. This shows that gate quantum computers can run a much larger set of problems compared to quantum annealers. However, gate quantum computers do not come without its drawbacks. Research and development for gate quantum computers are still trying to taking the problem of coherence (lifespan of stored information) and the stability of qubits. These problems are significant and only escalate as the number of qubits increase - which is detrimental to running large-scale optimisation operations. Due to this, gate quantum computers are confined to labs unlike D-Wave's quantum annealer whose applications are sold commercially. As a consequence, these quantum computers are yet to have practical applications.

While quantum annealers can only solve a subset of the problems that can be solved by gate quantum computers, they are still effective in optimising solutions to NP-hard problems by searching over large configuration spaces to find the optimal configuration. The quantum annealers work best on problems with many potential solutions, and ones where finding a near-optimal (local minima) solution is good enough. It is also less affected by noise than the gate quantum model which allows for usage of more qubits and thus, large configuration spaces for the optimisation problems. Its main advantage and current practical applications in the industry are predominantly on optimisation problems that stump classical optimisation

[5] Adiabatic quantum computing got its name from this.
[6] https://en.wikipedia.org/wiki/Quantum_logic_gate
[7] https://en.wikipedia.org/wiki/Shor%27s_algorithm

methods at a larger scale. A report by Google claims that quantum annealing is up to 10^8 times faster than its classical (simulated) annealing counterpart on a single core [25]. However, debate over if there is a significant speed up in performance is ongoing, and often contradicting in literature [18, 26, 19].

Chapter 4

Optimising Exam Timetabling through Quantum Annealing: Implementation

Through the use of D-Wave's Quantum Annealers, we were able to use to obtain low-energy state solutions to exam timetabling optimisation problems. This resulted in a more resource efficient method to solving these problems. This implementation involves encoding exam timetabling problems into Binary Quadratic Models (BQMs) which are then embedded onto D-Wave's Quantum Samplers.

We begin by defining the Exam Timetabling optimisation problem used throughout this paper.

4.1 Exam Timetabling Problem Description

Exam timetabling is a combinatorial optimisation problem in which the goal is to find the optimal assignment of a set of N exams $E = \{e_1, e_2, ..., e_N\}$ to a number of M days (time slots) denoted as $D = \{d_1, d_2, ..., d_M\}$ within the resource capacity of each time slot $R = \{r_1, r_2, ..., r_M\}$.

While exam timetabling problems usually vary, the common objective function of the problem is to minimise scheduling conflicting exams (exams that involve shared students) to the same time slot, i.e. denoting $d_i \in D$ as the day that exam i is assigned to and $S_{i,j}$ as the number of students sitting for both exams i and j, this can be written as:

$$i \neq j \wedge S_{i,j} > 0 \rightarrow d_i \neq d_j \forall i, j \in D$$

Real world versions of the Exam Timetabling problem also contain a variation of additional constraints, some of which can be contradictory, that is specific to the institution. Examples of these are usually to do the with capacity constraints that each time slot has (classroom or invigilator availability, classroom seating capacity, etc.). These additional constraints introduce a higher level of complexity and represent the challenge that exam timetabling problems pose.

In this paper, we present new methods for solving exam timetabling problems using D-Wave's Quantum and Hybrid solvers. To try and solve more practical variants of this problem, we made the assumption that the maximum time slots is known and stated as a parameter to the optimisation problem. This assumption generalises the problem from an arbitrary graph colouring problem and allows us optimise the allocation of exams by spreading out conflicting exams as much as possible (constraint 3 in table 4.1). The motivation behind this constraint is to leave as much time between conflicting exams as possible so that each student's individual

exam schedule will be more spread out. For example, a student having to take 3 exams over 3 days is a less desirable solution compared to taking 3 exams over 5 days. Additionally, in a real-world setting, this information (number of days available in an academic term to allocate exams) is usually known to institutions and readily available.

	Constraints
1.	Each exam has to be allocated to a time slot exactly once.
2.	No clashing exams (i.e. exams with shared students) assigned simultaneously.
3.	Spread conflicting exams out evenly as much as possible (i.e. avoid conflicting exams in x consecutive days).
4.	Exam allocation has to fall within the capacity of the time slot (i.e. size/number of exams need to be within room capacity/number of rooms).

Table 4.1: Constraints considered in exam timetabling solvers

We present and solved for 2 variants of the exam timetabling problem. In this chapter, both these variants are mapped to and solved entirely by D-Wave's Quantum Processing Unit (QPU). The first version is a basic exam timetabling problem that solves for constraints 1 to 3 in Table 4.1. The second version is a more practical approach where problems build upon the first version and additionally considers constraint 4. Below, we will see these optimisation problems formulated into Quadratic Unconstrained Binary Optimisation (QUBO) models which is sent to the D-Wave's Quantum Samplers to be solved.

4.1.1 Getting the problem's QUBO formulation

QUBO 1: Constraints 1 to 3

To map constraints 1 to 3 into a QUBO formulation, we started with a basic QUBO formulation of a graph colouring problem as defined in [27]. The link between these two problems are defined in section 3.1.2. We use the exam timetabling problem modelled as a graph $G = (V, E)$ with each exam as a vertex in vertex set V and each pair of conflicting exams as an edge in set E. The binary function defining constraint 1, each exam must be allocated exactly once, is:

$$C_1 = \sum_{i=1}^{N} (1 - \sum_{v=1}^{M} x_{i,v})^2 \tag{4.1}$$

where binary variable $x_{i,v} = 1$ denotes that exam i is allocated to day v. In the optimal case, C_1 will become 0 if satisfied as each $\sum_{v=1}^{M} x_{i,v} = 1$ since each exam is allocated only once.

To define the QUBO for the other constraints, we start with a basic exam timetabling problem covering constraints 2 and 3 that is based on a similar formulation in [28]. We can define this optimisation problem as:

$$minimise \quad \sum_{i} \frac{w_i}{S}, \quad i = \{0, 1, 2, 3, 4\} \tag{4.2}$$

where S is the number of students, w_i is approximate cost of all students having to sit two exams i tdays apart. These costs could be specified as a parameter to the application based on the varying aims of each problem. The default costs that we used, with the aim of spacing out clashing exams is:

$$minimise \quad \sum_i \frac{w_i}{S}, \quad i = \{0, 1, 2, 3, 4\} \tag{4.3}$$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 16 \\ 8 \\ 4 \\ 2 \\ 1 \end{bmatrix} \tag{4.4}$$

where S is the number of students, w_i is approximate cost of all students having to sit two exams i days apart.

The second term, representing constraint 2 from table 4.1, is defined as:

$$C_2 = \sum_{i,j \in E} \sum_{v=1}^{M} x_{i,v} x_{j,v} \tag{4.5}$$

C_2 totals the number of clashing exams allocated on the same day. If a ground state exists for our problem ($H = 0$), this term will also equate to 0 as no clashing exams i and j will be set on the same day, satisfying this constraint. We can modify 4.5 to take into consideration the weights specified in 4.4, giving:

$$C_3 = \sum_{i,j \in E} \sum_{v=1}^{M} \sum_{u=0}^{4} w_u x_{i,v} x_{j,v+u}, \quad v + u < M \tag{4.6}$$

where w_u is the cost function defined in 4.4. In this formulation, C_3 does not only penalise conflicting exams on the same day but also conflicting exams assigned to u days apart, successfully representing both constraints 2 and 3 from table 4.1.

Hence, we formulate our first exam timetabling QUBO that represents constraints 1-3 from table 4.1 as:

$$QUBO_1 = A_1 \cdot C_1 + A_2 \cdot C_3 \tag{4.7}$$

Here, variables (A_1, A_2) represent penalty terms that determine how strongly we want to represent constraints 4.1 and 4.6 respectively, by specifying an energy penalty when each term is violated. The larger the values of terms (A_1, A_2), the stronger the constraint becomes. This is called the penalty model and is explored in further detail below **??**.

QUBO 2: Constraints 1 to 4

The fourth constraint in table 4.1 can be represented as:

$$C_4 = \sum_{v=1}^{M} (r_v - \sum_i c_i x_{i,v})^2 \tag{4.8}$$

where r_v is the resource capacity of day v and c_i is the resources needed by exam i. Therefore the second exam timetabling problem considered, covering constraints 1 to 4 can be represented by:

$$QUBO_2 = QUBO_1 + A_3 \cdot C_4 \tag{4.9}$$

4.2 High-level Workflow

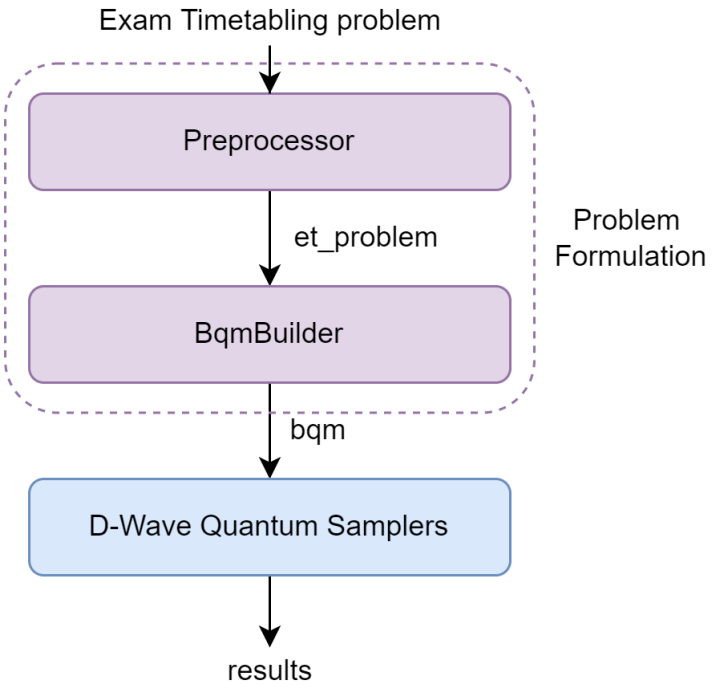


Figure 4.1: High-level workflow of D-Wave Quantum Annealer

Figure 4.1 is a brief overview of the application developed in this project to problem solve exam timetabling problems using D-Wave's Quantum Computer. This process can be divided largely into two steps: (1) Problem formulation and (2) Sampling. The following sections explore each step and its implementation.

4.3 Preparation of Exam Timetabling Problems

Given the hardware limitations of current quantum computers, we are unable to embed problems of large complexity onto them. Due to this, we conducted experiments and built our proof-of-concept application upon datasets we created based on realistic exam timetabling problems. We artificially generated these exam timetabling problems, keeping the **conflict density** of the datasets in a similar range to benchmark exam timetabling datasets [28]. For the data we generated, we kept the conflict density within the range of $[0.05 - 0.40]$, which is similar to that of the Toronto Benchmark Dataset. Table 4.2 describes these problems.

We wanted a variety of problems, so we made a handful of datasets that needed to solve for an additional constraint on the resource capacity (constraint 4 in table 4.1) on top of the core constraints (constraints 1 to 3). These problems can be identified by the term "cc" in the dataset name.

We also generated some disjoint problems, *small-03-cc-dis* and *med-02-dis*, to model exam timetables realistically. This is as most university departments will have a separate sets of courses offered to students in different years of their degrees with little overlap between them. Therefore, these problems will have disjoint groups of exams with clashes between them but not with exams in other groups.

All datasets were generated randomly and can be found in the folder "timetabling-data/test-data" in the software archive.

Dataset	Number of Exams	Number of Days	Number of Students	Constraints	Conflict Density
small-01	5	2	109	C1, C2	0.24
small-01-cc	5	2	103	C1, C2, C4	0.24
small-02	7	3	142	C1, C2	0.327
small-02-cc	7	3	145	C1, C2, C4	0.245
small-03	10	4	203	C1, C2	0.32
small-03-cc	10	4	207	C1, C2, C4	0.32
small-03-dis	10	4	202	C1, C2	0.08
small-03-cc-dis	10	4	202	C1, C2, C4	0.08
med-01	15	5	306	C1, C2	0.204
med-01-cc	15	5	306	C1, C2, C4	0.204
med-02	20	7	400	C1, C2	0.135
med-02-cc	20	7	400	C1, C2, C4	0.135
med-02-dis	20	7	403	C1, C2	0.045
med-03	30	10	605	C1, C2	0.197

Table 4.2: Dataset description

4.4 Problem Formulation

In the process of finding a solution to an exam timetabling problem, we want to optimise an objective function representing the energy of this system where low-energy states represent good solutions to the problem. D-Wave's quantum computers take this objective functions in the form of quadratic models whose lowest energy values represent the best solutions to the problems.

As we recall from section 3.2.3, the D-Wave QPU only accepts formulations of objective functions in either the Ising model or QUBO, both of which are Binary Qudratic Models (BQMs). This is as the BQM equation,

$$E(v) = \sum_{i=1} a_i v_i + \sum_{i<j} b_{i,j} v_i v_j + c \qquad v_i \in \{-1, +1\} \textbf{or} \{0, 1\}$$

represents both Ising and QUBO models. We chose to formulate our objective function for exam timetabling problems as QUBOs (detailed above in 4.1.1, using some of D-Wave Ocean SDK's in-built shared API *dimod* [29] to represent some of the constraints. As BQMs are unconstrained models, we represent constraints for our problem using a penalty model. In the following subsections, we will walk through the formation of an exam timetabling problem, and how it is encoded into a BQM that can be passed to the QPU.

4.4.1 Preprocessor: Representing the Problem Graphically

As we have seen, the exam timetabling problem has significant ties to the well-known graph colouring problem[1] and graph heuristics have been heavily researched in relation to timetabling

[1] https://en.wikipedia.org/wiki/Graph_coloring

problems [8]. Both problems are NP-complete [30], and have exact classical algorithms that can successfully find ground states in relatively smaller instances (ref to background section on this).

To produce the intermediate representation that the `BqmBuilder` uses to produce the BQM of the problem, we first parse the dataset and model this data graphically. This is done by setting each exam to a vertex in the graph and edges between vertices to represent the clashes. We also set attributes of "size" to each vertex to represent the number of students sitting for the exam, and "weight" to each edge as the number of students who share two exams. This is shown in figure 4.2 as it models a dataset of 10 exams sat by a total of 203 students. By solving this problem as a graph colouring problem, no adjacent vertices will have the same colour. These colours corresponds to the allocated time slot and therefore, no clashing exams will be in the same time slot.

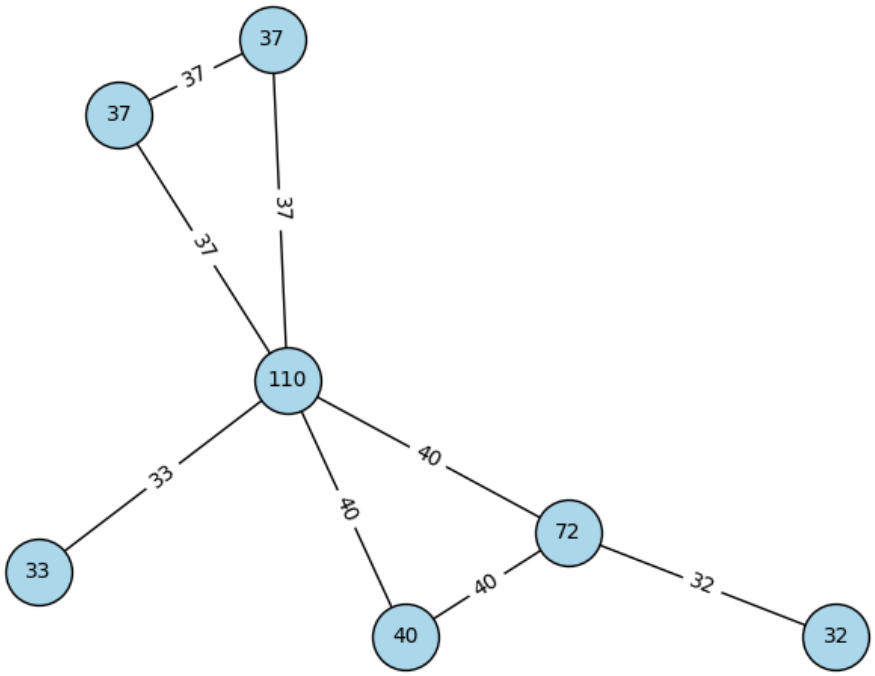


Figure 4.2: Graphical representation of dataset "small-03"

When parsing the dataset to create the intermediate representation, we must also extract other key pieces of information which are passed as parameters to the application, and used when solving the problem. Among these parameters of note are the number of days for which to build the schedule, the number of students sitting for the exams and for the second QUBO, the specified amount of capacity for each time slot.

```
1 classrooms 30 40 60
```

Listing 4.4.1 is an example of a line generated in our example problems and it specifies that students sitting for exams allocated to a specific day should be able to fit into classrooms the of size 30, 40 and 60 respectively.

4.4.2 BqmBuilder: Encoding the QUBO onto a Binary Quadratic Model (BQM)

Having received the intermediate representation of the problem, `et_problem` in figure 4.1, from the preprocessor, we now need to encode the problem's QUBO (specified in section 4.1.1) into a BQM. This is as BQMs are the only type of quadratic models accepted by the QPU.

We start with a simple example with 7 exams to be allocated over 3 days. The clashes between exams is described in the heatmap in figure 4.3.

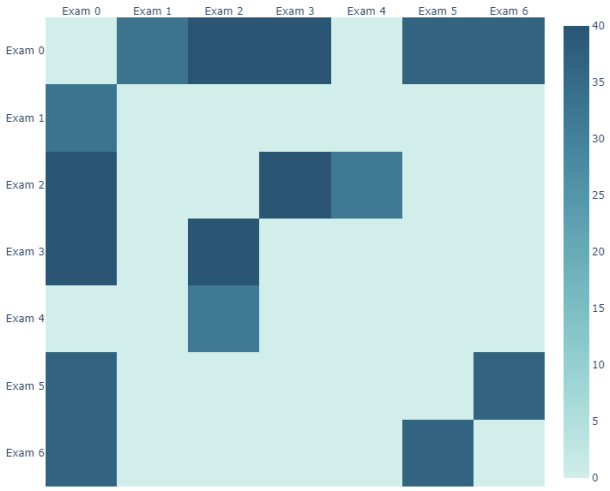


Figure 4.3: Heatmap of clashes between exams in dataset "small-02"

We then build the BQM by considering each constraint individually.

Constraint 1: Each exam has to allocated to a time slot exactly once.

This constraint, described in equation 4.1, is fundamentally a one-hot encoding on each set of binary variables for each exam over the available number of days. We want to ensure that each exam is allocated to only one day.

For example, Exam 1 can only be allocated to day 1 or day 2 or day 3. To build the BQM for this constraint, we use `dimod`'s in built function `combinations` which give the lowest energy solutions to the combinations which satisfy this one-hot constraint.

```
bqm = dimod.generators.combinations(['v_1,1', 'v_1,2', 'v_1,3'], 1)
```

Constraints 2 & 3: Avoid clashing exams in x consecutive days.

To represent these constraints, we can calculate the QUBO function 4.6 directly and use `dimod` to convert it into a BQM:

```
J = defaultdict(int)

for (i, j) in graph.edges:
    for day in range(num_days):
        for ind, weight in enumerate(weights):

            new_day = day + ind
            if new_day >= num_days:
                break

            J[f'v_{i},{day}', f'v_{j},{new_day}'] += (weight * graph[i][j]["weight"]) / num_students
            J[f'v_{j},{day}', f'v_{i},{new_day}'] += (weight * graph[i][j]["weight"]) / num_students

bqm = dimod.BQM(J, vartype='BINARY')
```

Figure 4.4 below shows the result of this code on the above dataset. We can see that while there is a harsh penalty when clashing exams are set on the same day, there also exists a penalty, albeit to a lower degree, when they are set on the consecutive days. This encourages the model to find solutions with clashing exams spaced out as much as possible as these would have the lowest energy.

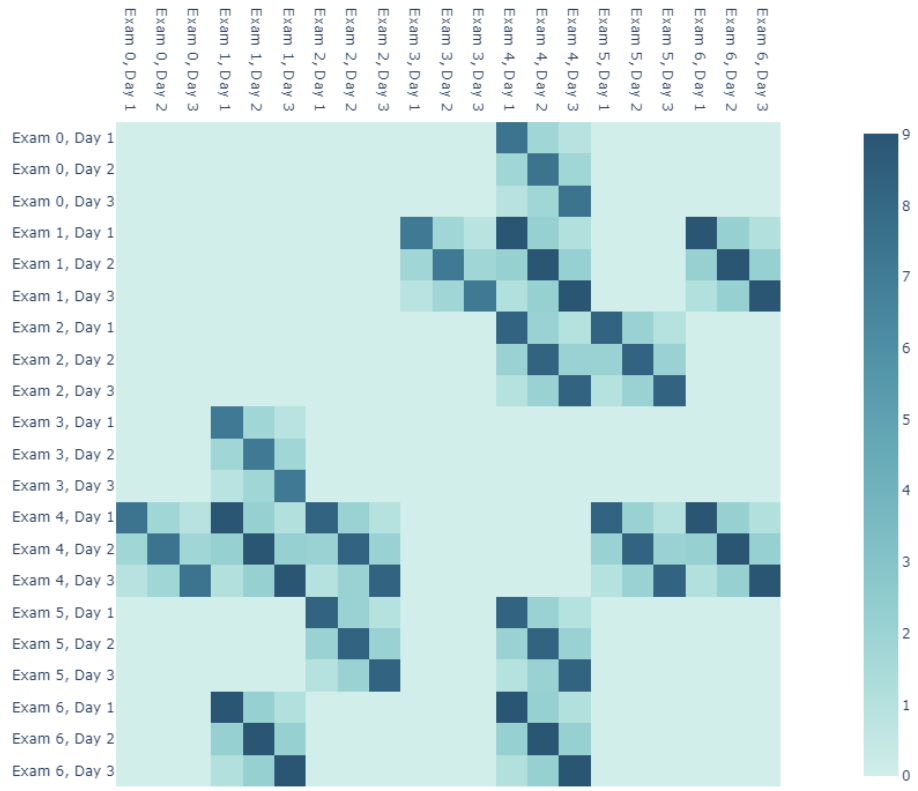


Figure 4.4: QUBO function of constraints 2 & 3 on dataset small-02

Constraint 4: Exam allocation has to fall within capacity of time slot

For this constraint, we could not define 4.8 directly as we did above. This is as a BQM only takes linear and quadratic biases. Therefore, we can only directly define the energy relationship between a variable and itself, or between two variables. Due to this, we could not simply add a penalty to combinations of exams that exceeds the classroom capacity if the length of the exam combination is more than 2.

Therefore, to define this constraint, we used D-Wave's `dwavebinarycsp` library to help construct the BQM from a Constraint Satisfaction Problem (CSP). We did this by calculating which combinations of exams would exceed the classroom capacities and penalising that particular configuration on the given day.

```
csp = dwavebinarycsp.ConstraintSatisfactionProblem(dwavebinarycsp.
    BINARY)

combs = list(combinations([f'v_{exam},{day}' for exam in graph.nodes
    ], len(resources['classrooms'])))
for comb in combs:

    if not satisfies_classroom_constraints(list(comb)):
        valid_configurations = set(product([0, 1], repeat=3)) - {(1,
    1, 1)}
        csp.add_constraint(valid_configurations, list(comb))

bqm = dwavebinarycsp.stitch(csp)
```

Penalty Model

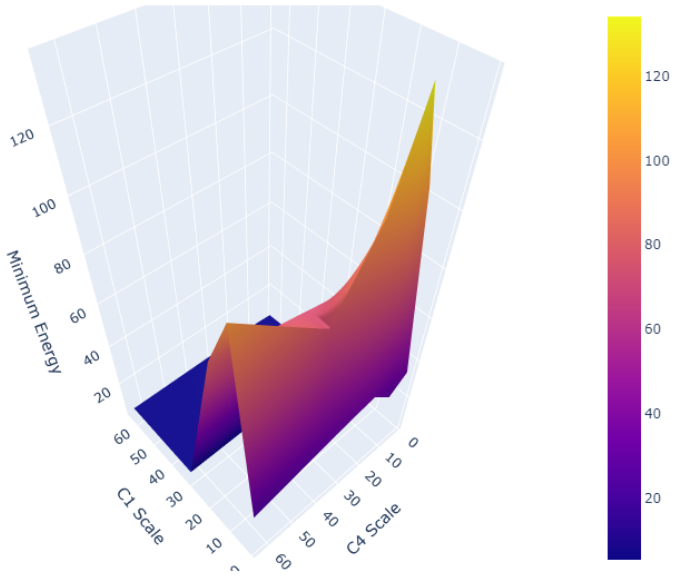


Figure 4.5: Minimum energy when applying penalty models of constraints differently

These individual BQMs can be added together using the `dimod.QuadraticModel.update()` function. We can also scale the individual BQMs representing each constraint using the `dimod.QuadraticModel.scale()` function. This acts similarly to terms A_1 and A_2 in equation 4.7, and scales the amount of penalty a solution receives if it breaks that specific constraint. By changing these values we can control how strongly we want to emphasise each constraint. However, a balance has to be struck as placing a high scale on one constraint and a lower one on another can result in finding solutions that satisfy some but not all constraints. This happens as the sampler will prioritise solutions that obey the constraints with the higher penalty over other constraints as it costs more to break it.

C1 Scale	C4 Scale	C1	C2	C4
1	1	No	Yes	Yes
1	8	No	Yes	Yes
1	16	No	Yes	Yes
1	32	No	Yes	Yes
1	64	No	Yes	Yes
8	1	Yes	Yes	No
8	8	Yes	Yes	No
8	16	Yes	Yes	Yes
8	32	Yes	No	No
8	64	No	Yes	Yes
16	1	Yes	Yes	Yes
16	8	Yes	Yes	Yes
16	16	Yes	Yes	Yes
16	32	Yes	Yes	Yes
16	64	Yes	Yes	No
32	1	Yes	Yes	Yes
32	8	Yes	Yes	No
32	16	Yes	Yes	Yes
32	32	Yes	Yes	Yes
32	64	Yes	No	No
64	1	Yes	Yes	No
64	8	Yes	Yes	No
64	16	Yes	Yes	Yes
64	32	Yes	Yes	No
64	64	Yes	Yes	No

Figure 4.6: Constraints fulfilled for each configuration of penalties for dataset med-01-cc

To find the optimal configuration of penalty terms (C1 Scale and C4 Scale), we ran multiple experiments on dataset small-02-cc. We fixed the penalty term of the BQM representing constraints 2 and 3 to 1, and varied the penalty terms of BQMs representing constraints 1 and 4. The ranges explored for each penalty term were from 1 to 64. This range was chosen as we wanted constraints 1 and 4 to be represented to a stronger level as constraints 2 and 3. Therefore, we chose to investigate a range of numbers around the maximum weight in w (equation 4.4).

Figure 4.5 shows the minimum energy of the optimal solution for each configuration and the table in Figure 4.6 shows which constraints were fulfilled for each configuration. Based on these results, I chose to set the penalty terms of the BQMs representing constraint 1 and constraint 4 to be 16 each.

Initially, we defined the optimisation problem in 4.3 without normalising the weights by the number of students. This caused a significant problem when defining the penalty model, as the numbers in each QUBO defining constraints 2 & 3 were largely influenced by the number of students in each dataset. This meant that the penalty terms for each constraint would have to be tuned for individual datasets in order to find terms that properly represent the problem. This problem was solved by adding the normalisation factor to the QUBO in question.

4.5 Sampling

The second part to finding a solution is sampling. The BQM formulated in the previous section is sent to one of D-Wave's samplers anneal over the energy landscape specified by our BQM and retrieve samples from the low-energy states. The initial part of this sampling process is to embed the BQM to the qubits in the quantum computer, which is carried out by D-Wave Samplers using an in-built tool `minorminer`. The next section discusses this process, which is called minor embedding.

4.5.1 Minor Embedding

In order for the QPU to sample from the objective function represented by our BQM, we first must map it to the QPU. This is done by mapping linear coefficients (weight of each variable to itself) of the BQM to qubit biases and quadratic coefficients (weight of each variable to another variable) to coupler strengths. This mapping provides the energy landscape that the QPU anneals over to find the global energy minimum, which then corresponds to the optimal solution of our problem.

While the D-Wave QPUs are a lattice of interconnected qubits, not all qubits have couplers connecting them to all other qubits. As explored in the background section 3.2.3, D-Wave QPUs follow topologies known as Chimera and Pegasus. As neither are fully connected graphs, it is unlikely that the BQM graph can map directly to our QPU's topology. When this occurs, we minor embed variables to qubits using *chains*.

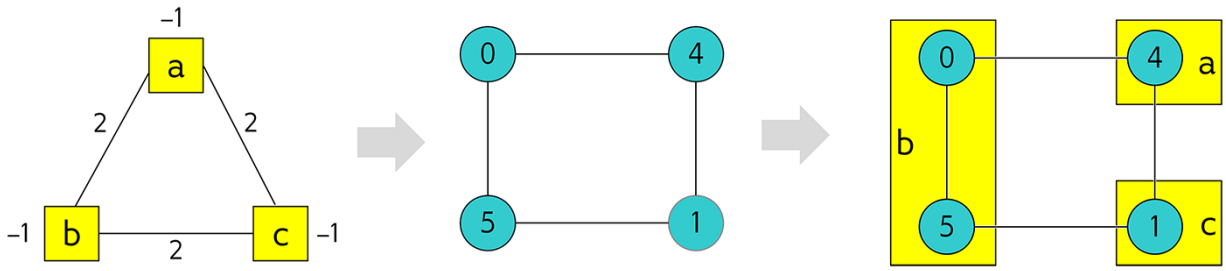


Figure 4.7: Taken from D-Wave System Documentation [31]

The idea of this is to chain more than one qubit together to make a logical qubit that represent one variable with the same connections to other variables. Figure 4.7 shows the embedding of a three-node graph onto a four-node structure. It shows that qubit 0 and 5 are chained together to represent variable b. This is done by setting a negative coupler strength,

known as the *chain strength*, to the chained qubits in order to correlate their states. At the end of the quantum annealing process, the chained qubits should represent the same classical state as they are acting as a single variable. This is not always the case, and *chain breaking* can occur. This is explored in further detail in section 5.3.

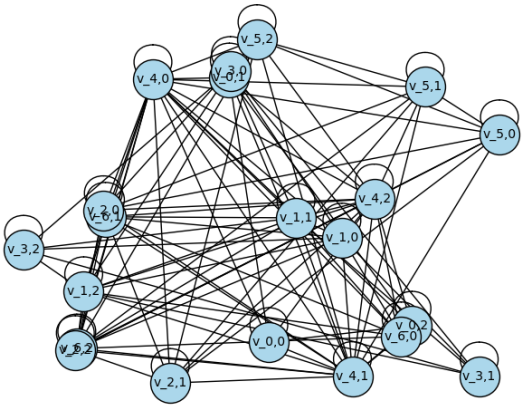


Figure 4.8: Graph of BQM

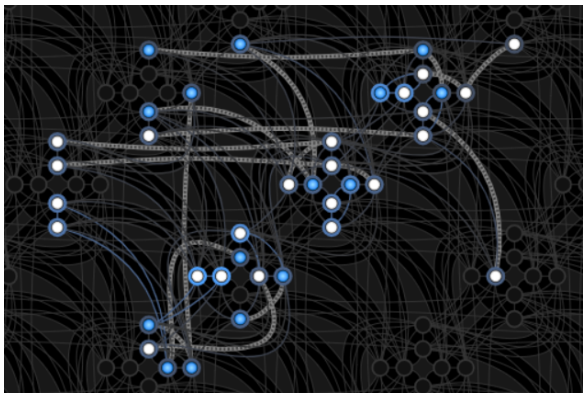


Figure 4.9: Minor Embedding to QPU

Figure 4.8 shows the graphical representation of the BQM generated for dataset small-02 which has 7 exams to be allocated over 3 days. The edges of each node to itself represents the linear coefficients that are mapped to qubit biases and the edges between nodes represent the quadratic coefficients that are mapped to coupler strengths. This BQM is then minor embedded to D-Wave's QPU and this embedding is shown in figure 4.9. Initially, we embedded problem's to the QPU with the Chimera topology, and later explored the Pegasus topology (detailed in section 5.3. The highlighted (light grey) edges in this figure represent the chains. We can see that all qubits in each chain have the same classical value (blue for 1, white for 0).

We can now send our BQM to the D-Wave Sampler. We do this using D-Wave's `EmbeddingComposite` function which implicitly automates the minor embedding of the BQM using tools in D-Wave Ocean's software kit such as `minorminer`. It does this by running a heuristic algorithm to find the optimal embedding, which is one with least/shortest chains. We discuss difficulties finding optimal embeddings and how suboptimal embeddings can lead to poorer final solutions in section 5.3. This minor embedding can also be manually defined; however, this is a tedious process, especially as the problems become more complex and number of variables increase. As this is not very feasible, we did rely on D-Wave's automated embedding functionality, but worked to improve it by tuning parameters later on.

4.5.2 D-Wave's Samplers

Once our BQM is embedding to the quantum computer, it anneals over the energy landscape and retrieves samples of low-energy states. D-Wave provides a multitude of samplers that run remotely in D-Wave's Leap environment or locally. These are:

- **Quantum Solvers**: Advantage and D-Wave 2000Q quantum computers. These are the Quantum Samplers we investigated and used for our final results in Chapter 5.
- **Classical Solvers:** Exact solvers that run greedy classical algorithms locally, and a Simulated Annealing sampler developed using D-Wave Ocean's `dwave-neal` tool. We used these samplers to evaluate our results against in Chapter 5.

- **Quantum-Classical Hybrid Solvers:** Cloud-based Leap Hybrid Solvers and customisable hybrid solvers developed using D-Wave Ocean's `dwave-hybrid` tool. We investigated and ran these samplers on our dataset as well as larger problems, detailed in Chapter 6.

4.5.3 Access to D-Wave's Quantum Samplers

For the entirety of this project, all results obtained and displayed were from examples that were run directly on D-Wave's QPU (**not** on simulators) for Chapters 4 and 5 and Hybrid Solvers for Chapter 6. This was facilitated through D-Wave Leap's [32] Developer Plan which offered flexible access to QPUs and hybrid solvers through contributions of the software developed to the open-source community.

Besides access to the samplers, the plan also allowed access to D-Wave Ocean Software, D-Wave Leap IDE, interactive demos of featured applications using Quantum Annealing and other online resources.

4.6 Validation and Visualisation of Results

The results output from the D-Wave Samplers come in form of a list of records, each representing a sample from one anneal cycle on the D-Wave QPU. This list can be passed to the `Timetable` class in our application that runs checks to validate the best result (state if and which constraints are broken in the final solution) and also helps to visualise the results.

These are seen in figure 4.10 which shows the initial timetable at iteration 1, and the optimised timetable in figure 4.11. The colour of each block describes the number of clashes seen by each exam. We can see that there are many clashes initially, and multiple exams have been allocated more than once. Both of these problem are fixed in our final timetable, as the quantum annealer succeeds in finding a valid solution.

This visualiser was built with Plotly [33] and binded to a user interface using a framework built on top of Plotly called Dash [34].

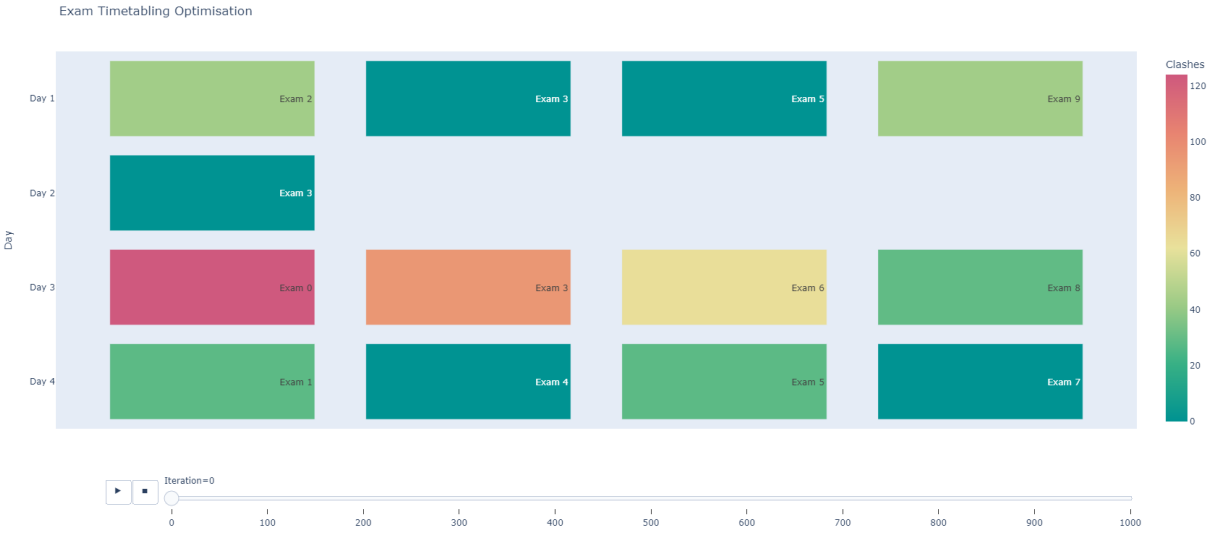


Figure 4.10: Initial timetable at Iteration 1

Figure 4.11: Optimal timetable at Iteration 1000

Chapter 5

Optimising Exam Timetabling through Quantum Annealing: Evaluation

5.1 Aims

This project aims to use quantum annealing to investigate exam timetabling optimisation problems. In this project, we focused on *building a proof of concept* for quantum annealing to solve optimisation problems, which we achieved through embedding problems on the QPU using D-Wave's Ocean Software. Building on this, we also wanted to *investigate the extent to which quantum annealing can problem solve* in terms of complexity of the problem, problem structure, resources, etc.

We analysed the success of this method by considering *how quantum annealing compares to classical methods* (i.e. Simulated Annealing) in terms of quality of solution, computational time, etc. and the *feasibility of finding the optimal embedding* for a particular problem.

5.2 Comparing initial results to Simulated Annealing (SA)

We start by comparing our initial results to solutions produced through simulated annealing. Figure 5.2 shows the spread of minimum energy retained from each method for 10 sweeps each. As both SA and QA are probabilistic sampling algorithm, we fixed the number of samples to 1000 for each sweep, to be able to fairly compare the results. We can see that SA is fairly reliable for all datasets, achieving a low energy solution without much variation in the smaller datasets and only slight variation in the larger dataset.

While QA shows significant probability of producing good solutions for the smaller problems, we can see that it struggles to produce a satisfying solution when the problem complexity increases. We can see that this is consistent with the data visualised in figure 5.1 as the quantum annealer manages to find solutions with equal/slightly higher energy as simulated annealing in the smaller datasets but produced results with significantly higher energies as complexity increases.

On top of this, QA is also less reliable at producing the same solutions for larger problems, as we can see a bigger spread of results in figure 5.2. Both methods struggle with this issue as they are both probabilistic methods and larger problems represent larger energy landscapes that need to be explored and sampled over to find the optimal solution.

As explored in the background, the main motivation for QA is that it significantly outperforms SA in terms of computational time. Figure 5.3 shows that while SA produces its

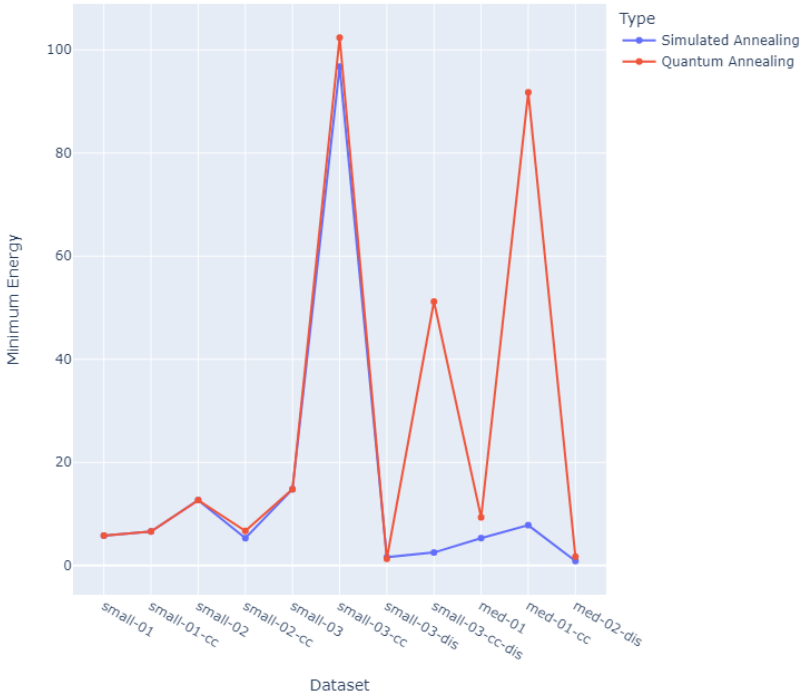


Figure 5.1: Minimum Energy for each dataset for QA and SA (The datasets omitted from these graphs were problems that were unable to embed to the QPU)

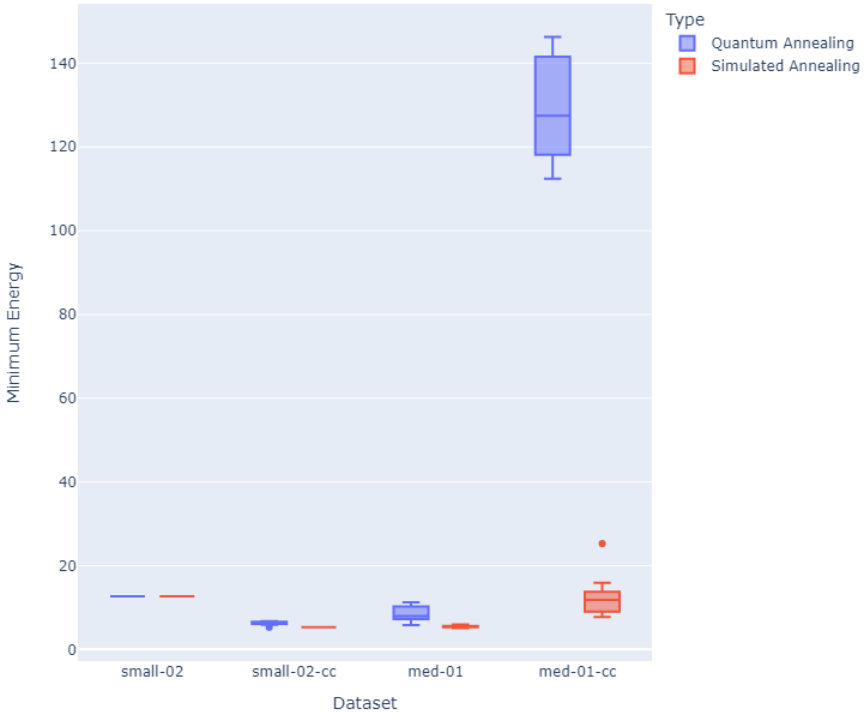


Figure 5.2: Minimum Energy from 1000 samples for 10 sweeps for QA and SA

solution in relatively reasonable times for smaller problems, this increases significantly as the complexity of the problem increases.

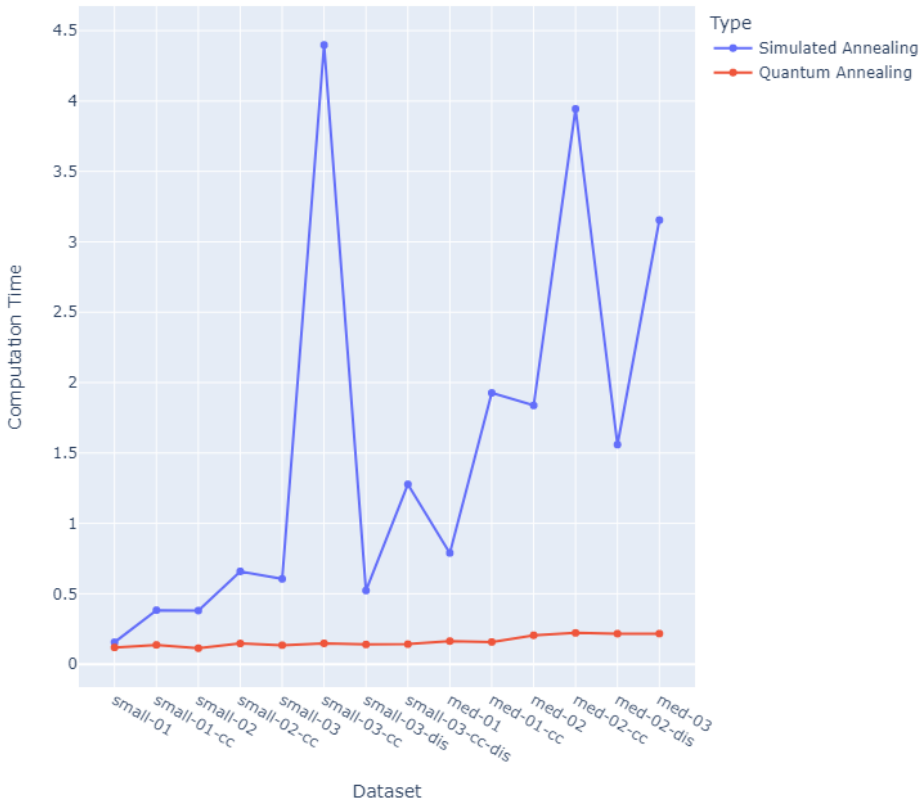


Figure 5.3: Computation Time for each dataset for QA and SA

5.2.1 Analysing QA solutions

From figures 5.2 and 5.1, we have noted that QA struggles to find satisfying solutions to larger problems with higher complexity. Given that each problem and its subsequent BQM formulation needs to be embedded to the QPU, as described in Section 4, we see that as the complexity of the problems increase, so does the complexity of the BQM graph that needs to be embedded. Finding the optimal embedding of these graphs is the main differentiator in obtaining good solutions from the Quantum Sampler. Therefore, the lackluster performance of QA for the larger datasets can be explained by poorer/sub-optimal embeddings of the BQMs to the QPUs. While, for the most part, this cannot be helped as the complexity of BQM graphs severely increase with regards to number of variables and addition of constraints, there are some steps that we can take to find a better embedding to the QPU, and subsequently improve QA performance on the larger datasets. Besides the embedding, we can also achieve improvements in performance by small optimisations to the annealing process.

5.3 Improving QA Performance

5.3.1 Optimising the Annealing Process

Number of Reads

The simplest change we can make to the annealing process is to increase the number of reads taken during the anneal. As the QA process works through collecting samples on each anneal, a higher number of reads correlates to a higher number of samples taken. This gives a higher probability of finding better solutions with lower energy states, which is illustrated in figure 5.4.

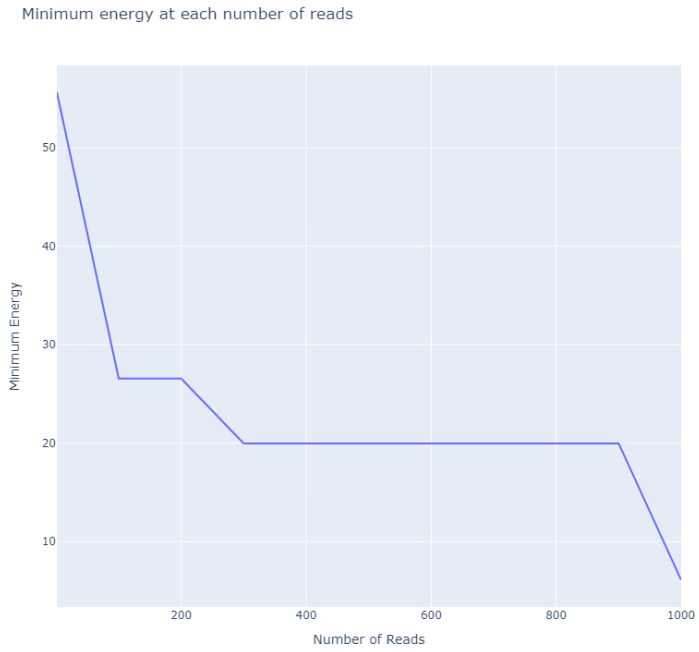


Figure 5.4: Minimum energy for each number of reads for dataset small-02-cc

5.3.2 Improving the Problem Embedding

Topology

Our initial implementation embeds all problem BQMs to the D-Wave 2000Q QPU which supports a Chimera topology (see section 3.2.3). We want to investigate if there is a significant change in performance when embedding to Advantage's Pegasus structure instead.

Chimera and Pegasus topologies differ in two main categories: (1) maximum graph size supported and (2) graph connectivity.

The Chimera architecture supports up to a C16 Chimera graph which consists of 2048 qubits and 6016 couplers. In contrast to this, the Pegasus architecture supports a larger graph of 5760 qubits. We can see this evidently in figure 5.5 as the BQM of the same dataset med-01 is embedded to a much higher percentage of D-Wave 200Q's QPU (Chimera) compared to Advantage (Pegasus). Additionally, we also found that Chimera could not embed and therefore find solutions for some of our larger datasets including med-02, med-02-cc, and med-03 - all

of which were successfully embedded to Pegasus. The largest dataset we could embed onto Chimera and Pegasus were med-01-cc and med-03 respectively.

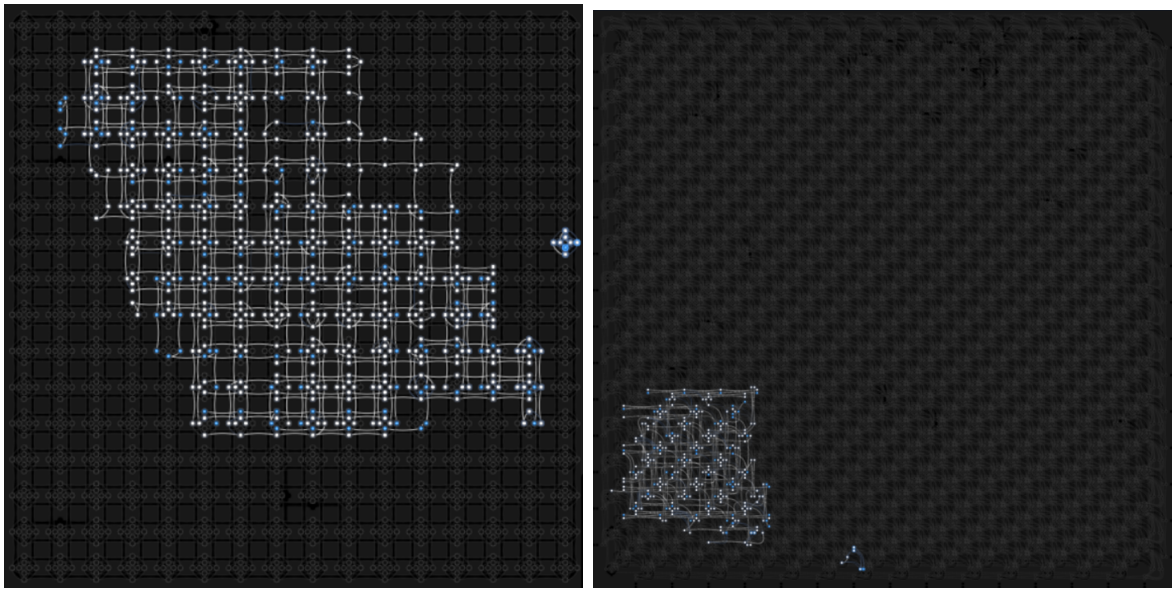


Figure 5.5: Chimera (left) & Pegasus (right) embedding for dataset med-01

The second significant different between the two topologies is the connectivity of the graph structures. As visualised in figures 3.4 and 3.5 in section 3.2.3, the Chimera architecture connects each qubit to 4 other orthogonal qubits internally and 6 qubits externally. This is notably less than Pegasus which connects qubits to 12 other qubits internally and 15 qubits externally. The connectivity of the graph structure is important as we need to map our BQM graph to the QPU. As the QPU's structure is not fully connected, we need to sometimes represent BQM variables using more than one qubit chained together. The concept of chaining is explained in section 4.5.1. It follows that the higher the level of connectivity of the QPU graph structure, the simpler the embedding of the BQM, and therefore, the fewer/shorter the length of chains.

The **maximum chain length**, which is the length of the longest chain in the embedding of a problem, is a significant factor in ability of the QPU to sample good solutions during the annealing process. This is as throughout the annealing process, qubits in longer chains then to *freeze out* earlier in the process. Frozen qubits stop changing and evolving in an appreciable manner as the problem Hamiltonian evolves (see section 3.2.4). These frozen qubits ultimately lead to poorer solutions as the variables these qubits represent will be unable to find its lowest energy state.

To properly illustrate this improvement in embedding, figure 5.6 shows a difference in the maximum chain length for a small dataset that embeds to 8 logical qubits on both graph structures. We can see that the maximum chain length is 3 for Chimera and 1 for Pegasus. This follows as Pegasus has a higher level of connectivity. While Chimera performs well on these small datasets, it is evident how this improvement in embedding will become more substantial as the size of problem grows and BQM graphs become more complex. Therefore, we found that reducing the maximum chain length by switching to D-Wave's Advantage computer's Pegasus topology, significantly improves solutions in the larger datasets. For this reason, Final QA values in section 5.4 are all generated by embedding problems to the Advantage computer.

Besides switching graph topologies to improve chain lengths, we also investigated D-Wave's `minorminer` library, which calls function `find_embedding` to run a heuristic algorithm mapping

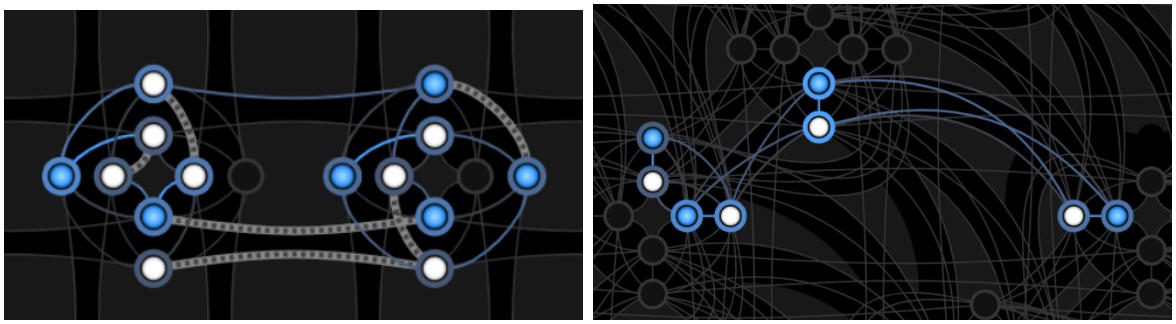


Figure 5.6: Chains in Chimera (left) & Pegasus (right) embedding for dataset small-01

source graph s to a target graph t. We modified this function's `chainlength_patience` parameter to see if we could find a better embedding for each problem. This parameter defines the number of failed iterations the algorithm was run to find a mapping from s to t. By increasing this number, it was more likely the algorithm would find an embedding with a smaller maximum chain length. An issue that came up when trying this optimisation was that running this algorithm many times - especially on larger problems - caused a notable increase in computational time (and resources, as it was running locally). Due to this, we ran this function with a value of 100 for `chainlength_patience` to embed each problem to Pegasus and saved these embeddings. We then reused these fixed problem embeddings when calling the Quantum Samplers throughout the rest of the experimentation.

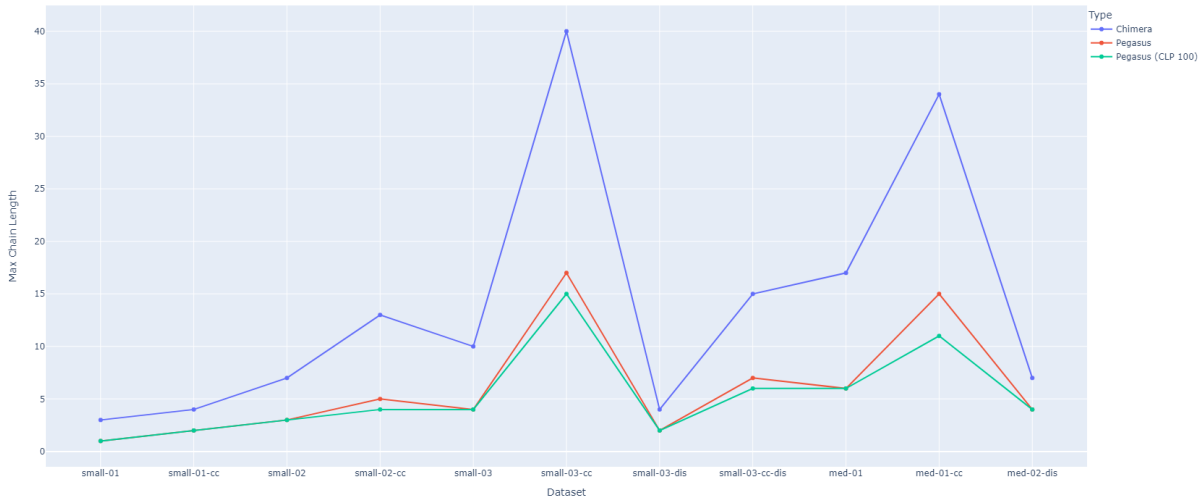


Figure 5.7: Max Chain Lengths for Chimera, Pegasus & Pegasus (`chainlength_patience`=100)

Figure 5.7 shows the difference in maximum chain lengths for each dataset (omitting the datasets Chimera could not embed) for an embedding to the Chimera architecture, Pegasus architecture and Pegasus architecture when `chainlength_patience` is set to 100. As we can see, increasing `chainlength_patience` did reduce the maximum chain lengths, albeit not by much. The embeddings calculated here were saved and used to generate the final QA results shown in section 5.4.

Chain Strength

Logical qubits (chain of qubits representing one variable) in the QPU need to be in a classical state at the end of the annealing process, 0 or 1. This means that every qubit in the chain is constrained to have the same value in order to properly represent a solution. This is done by weighing each chain with a `chain\_strength`. In the case that the chain strength is too low, qubits in a chain have differing values at the end of the annealing cycle, which is known as a *chain break*.

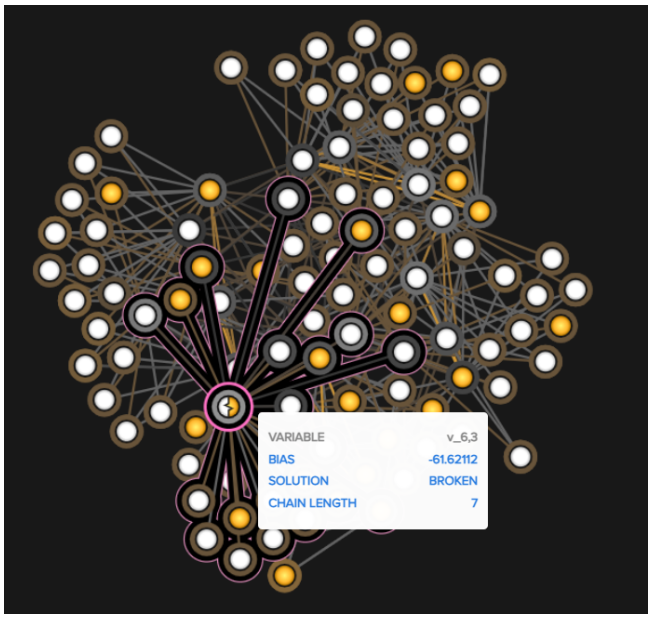


Figure 5.8: Broken chain on dataset small-03-cc-dis shown using D-Wave's Problem Inspector tool

We can see a broken chain visualised in figure 5.8, which was investigated with the use of D-Wave's Problem Inspector tool. When this occurs, a chain break fixing method is used to assign a classical state to the binary variable. The chain break method we have used is `majority_vote`, where a postprocessing software takes the classical state of the majority of qubits in the chain and applies it to the variable. This is not ideal, as the QPU will return solutions that are unlikely to be optimal due to the misrepresentation of qubit values in the broken chain.

Chain Strength	Minimum Energy	C1	C2	C4
30	15.019			
40	26.514			
50	11.188			
60	6.152			
65	3.673			
70	13.283			
75	45.374			

Table 5.1: Minimum energy for different chain strengths for dataset small-03-cc-dis and which constraint is broken

However, we also need to be careful not to raise the chain strength too high, as chain strengths are considered alongside qubit biases and coupler strengths which represent our prob-

lem. As an auto scaling feature scales all weights in a QUBO, having a chain strength that is too large will cause our QUBO terms which will diminish. As these terms represent the minimisation objective and problem constraints, the effect of this is that our original problem is no longer effectively represented on the QPU.

To find the appropriate chain strength to problems, we first tried using the `dimod.embedding.chain_strength.scaled()` function, which scaled the chain strength up to the upper bound of the bias values. However, in the larger problems, this function calculated an extremely high chain strength due to the additional variables and constraints. As a result, the inflated chain strengths significantly changed the meaning of the problem and we started getting solutions that completely ignored our constraints. Due to this, we did not use this method in our final model.

We then tuned different chain strengths on different problems through experimentation. We tried values within a range of the highest bias provided in our problem QUBO. Table 5.1 shows a case of this for dataset small-03-cc-dis. We can see that close to all constraints were broken when chain strength was set higher than its optimal level which for this dataset, is 65. Given that we found there to be significant improvement when tuning the chain strength, we tuned the chain strengths for all datasets and these were the values, shown in table 5.2, were used in the final QA solutions in section 5.4.

Dataset	Chain Strength
small-01	11.7
small-01-cc	40.7
small-02	16.6
small-02-cc	59.3
small-03	20.0
small-03-cc	59.3
small-03-dis	19.6
small-03-cc-dis	65.0
med-01	16.3
med-01-cc	65.5
med-02	33.6
med-02-cc	58.7
med-02-dis	27.7
med-03	68.3

Table 5.2: Final tuned chain strength for each problem

5.3.3 Other improvements considered

The D-Wave Ocean software provides other parameters that can be tuned to control the annealing process on different problems. Some of which we investigated were:

Annealing Time

This parameter controls the time, in microseconds, between which the quantum annealing process takes place. A higher annealing time would increase the probability of finding good low energy states. However, having tried this method on some of the larger datasets, we did not

observe any significant improvement in results. Additionally, an increase in annealing time was also considerably expensive in terms of computation time on the QPU, which we had limited access to. Due to these reasons, we decided not to investigate this control further.

Anneal Offsets

Anneal Offsets is a parameter that can be adjusted to control the annealing path of each individual qubit. The motivation behind this control is for offsets to be added to logical qubits with longer chains so that they start the annealing process later and can effectively avoid the freezing out problem. However, this parameter needs to be tuned for each individual logical qubit and therefore is highly unfeasible at the size and complexity of our problems.

Anneal Schedules: Pause, Quench & Reverse Annealing

The D-Wave QPUs allow for customisation of the global anneal schedule by specifying a set of points to define the annealing pattern. These schedule points come in the form of a list of tuples (t, s) where t is some time-μs during the annealing process and s is the anneal fraction (rate of annealing). We can use these points to manipulate the annealing trajectories. Some common trajectories are:

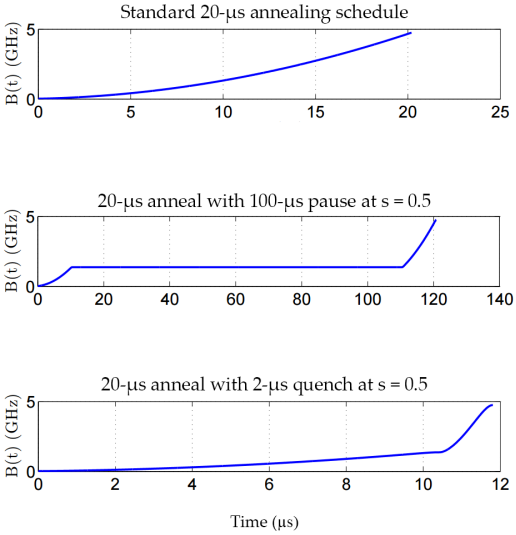
- **Pause:** where the anneal is stopped halfway through for a specified period of time at a particular anneal fraction before beginning again.
- **Quench:** where the default anneal is interrupted midway at a particular anneal fraction, then ended abruptly
- **Reverse Annealing:** where the anneal begins at a specified classical state and the rate of annealing is reversed from 1 towards 0 then back to 1.

Description	Points
Standard Trajectory	(0.0,0.0), (20.0,1.0)
Pause	(0.0,0.0), (10.0,0.5), (110.0,0.5), (120.0,1.0)
Quench	(0.0,0.0), (10.0,0.5), (12.0,1.0)
Reverse Annealing	(0.0,1.0), (2.5,0.5), (27.5,0.5), (30.0,1.0)

Table 5.3: Example of Schedule Points, illustrated in Figure 5.10

Research has found that there exists a critical region where even relatively short pauses can produce significantly better solutions. Additionally, the anneal cycles produced will have a higher probability of finding solutions with low energy states [35, 36]. It also states that later quenches are more likely to produce solutions at lower lying energy states with a higher success rate. We investigated these theories on dataset med-01.

Figure 5.11 shows that we found values for pause and quench starts that gave us minimal energy solutions for a range of values. However, it should be noted that neither of these values were lower than the minimum energy solution found using the standard annealing trajectory. As the experiment carried out were computationally expensive and did not yield significantly better results, we did not tune every single dataset to find the critical points for pause and quench starts. This was also due to its impractical nature, especially in larger problems. For this reason, Final QA values in section 5.4 do not take into account pause or quench schedules.



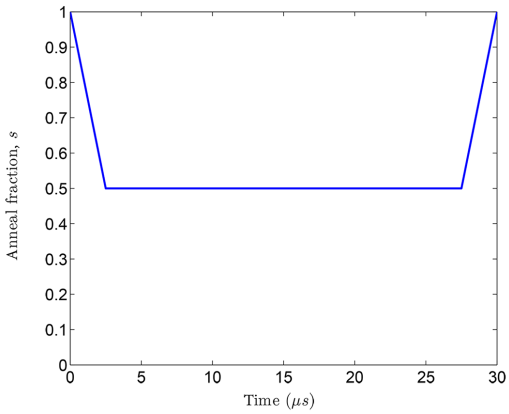


Figure 5.9: Reverse Annealing

Figure 5.10: Taken from D-Wave System Docs [35]

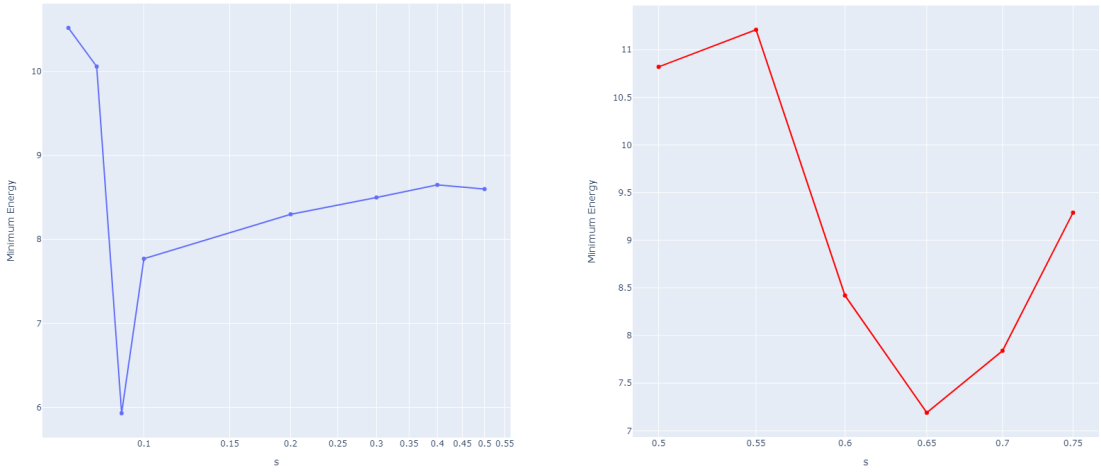


Figure 5.11: Minimum Energy for different Pause Starts (left) and Quench Starts (right) **s** on med-01

5.4 Final Results and Summary

Table 5.4 show the minimum energy of samples produced through Simulated Annealing and Quantum Annealing (before and after making the improvements discussed in this chapter). This is also visualised in figure 5.12. We can see that the Quantum Annealer performs well on the smaller datasets, keeping up with quality of solution despite running much quicker than its classical counterpart. However, as the size of the problems get larger and its complexity grows, the QA struggles to find good solutions to the problems that are still fairly easily calculated by the Simulated Annealer.

Through the improvements made in this chapter, we have learned that improvements can be made to the embedding of the problems and the annealing process in order to increase performance. We achieved this successfully in datasets such as small-03-cc-dis and med-01 among others. However, improvements to the embedding of problems can only help to an extent. When problems get too complex and the maximum chain lengths get too high, it is

Dataset	Simulated Annealing	Quantum Annealing (Initial)	Quantum Annealing (Final)
small-01	5.798	5.798	5.798
small-01-cc	6.601	6.601	6.601
small-02	12.676	12.676	12.676
small-02-cc	5.296	6.703	5.403
small-03	14.788	14.788	14.788
small-03-cc	96.760	102.384	89.855
small-03-dis	1.603	1.297	1.297
small-03-cc-dis	3.554	49.821	3.673
med-01	4.823	6.833	5.208
med-01-cc	6.738	46.516	26.294
med-02	2.557	-	2.650
med-02-cc	37.794	-	85.346
med-02-dis	0.863	1.559	0.987
med-03	5.763	-	90.453

Table 5.4: Final Results showing energy of best solution by SA and QA (before and after making improvements)

difficult to produce decent embeddings of the problem BQM to the QPU. When this happens, the Quantum Annealer will struggle to output low-energy solutions.

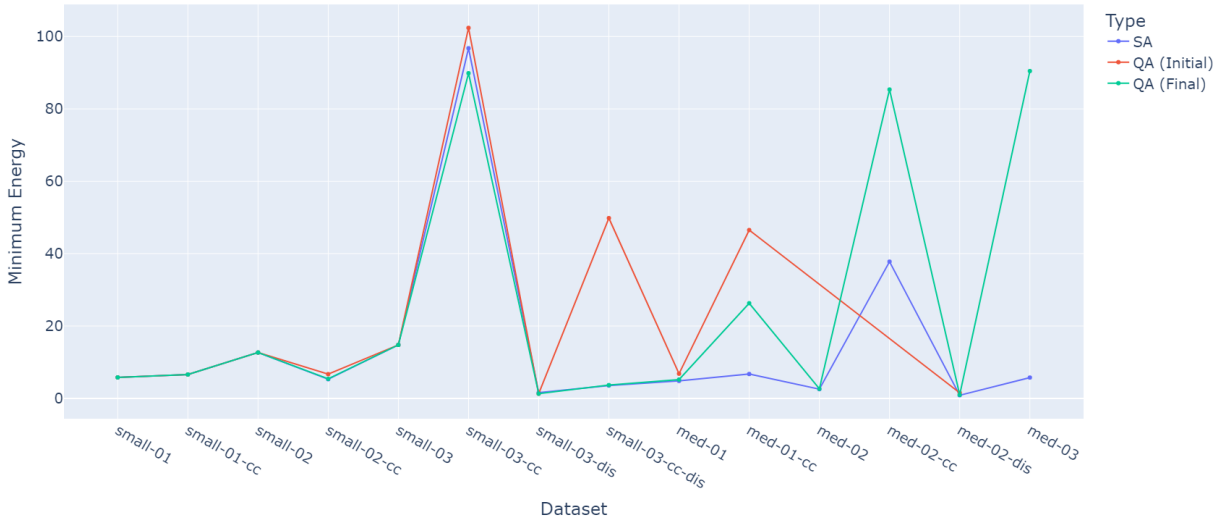


Figure 5.12: Minimum Energy Solutions for Simulated Annealing, Quantum Annealing (initial configuration) & Quantum Annealing (final configuration). Note that QA (Initial) is missing values for datasets med-02, med-02-cc and med-03 as it was not able to embed the problems.

Chapter 6

Optimising Exam Timetabling through Hybrid Quantum Annealing: Implementation & Evaluation

While the potential of Quantum Annealing has been investigated and seen to some success in previous chapters, we have learned that it struggles to embed larger problems, therefore producing poor solutions to them. Additionally, most problems of interest cannot be embedded onto current capacities of the QPUs. This is as practical problems are often much larger, complex and require a significant amount of computing power. In this chapter, we investigate simple hybrid optimisation methods on larger datasets that work by exploiting the strengths of quantum and classical computational methods.

6.1 Aims

This chapter aims to build and test a hybrid solver on a large real-world problem. We will measure the success of this solver on:

- If it returns a **valid solution** for a specified problem: One that does not break hard constraints of (1) all exams are only allocated to one time slot and (2) No clashing exams allocated to the same time slot.
- The quality of the solution as compared to the benchmark penalty value for a specified problem.

6.2 Preparing the University of Toronto Benchmark Dataset

For the following hybrid methods, our aim was to build a solver that could find solutions to more practical problems. Therefore, we wanted to test our methods on the University of Toronto Benchmark Dataset, detailed in section 3.1.3.

The optimisation problem solved for problems in this dataset follow exactly that of $QUBO_1$ in equation 4.7. We compared our results to the highly regarded benchmark results published by Pilar Moreno Diza (Alfonso X el Sabio University) [28]. This compilation of benchmarks also provided a helpful executable program which we used to calculate the penalty of a solution. The penalty calculated here followed exactly equations 4.3 and 4.4.

6.3 Extending Quantum Annealing with Classical Postprocessing

Classical postprocessing can be applied to the solutions produced by the QPU to find local improvements without much overhead of classical processing time. While the D-Wave 2000Q systems support optimisation and sampling postprocessing algorithms, the Advantage system is limited to the use of classical algorithms in D-Wave's `dwave-greedy` library to compute energies of the returned samples. We have chosen to implement the latter on top of our Quantum Annealing model as we have received better results on the Advantage system.

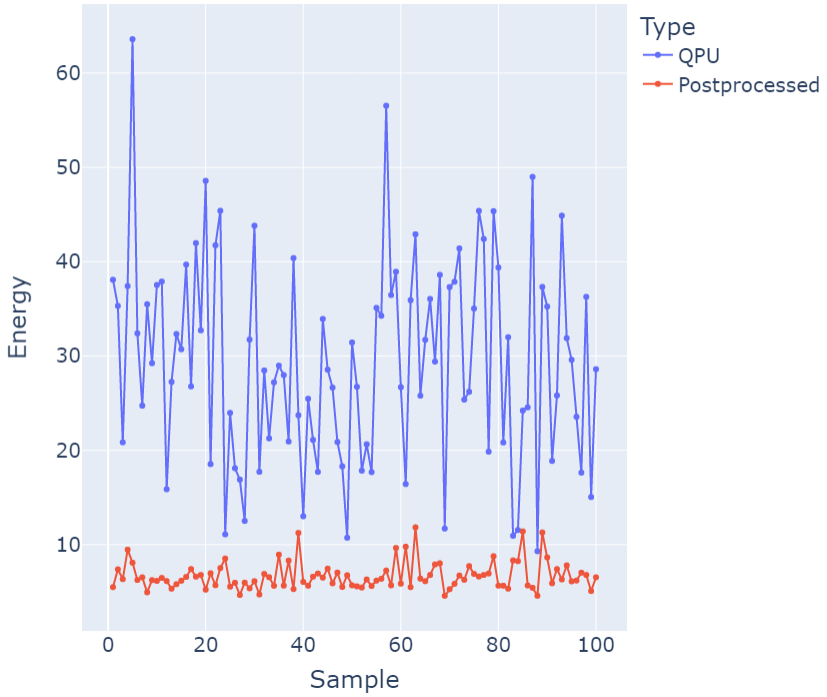


Figure 6.1: Energy of each sample from QPU and after steepest descent algorithm has been run for 100 samples on med-01

To implement postprocessing, we made use of `dwave-greedy`'s `SteepestDescentSampler` for BQMs, which performs a local search for minima within the sample's neighbourhood. After retrieving samples from the QPU, we pass these samples in as the initial states to the `SteepestDescentSampler`. This sampler carries out postprocessing by running the steepest-descent algorithm on each sample to find the nearest minima. The lowest state solution is then returned. Figure 6.1 shows the nearest minima found by the steepest descent postprocessing algorithm for each sample on dataset med-01. We can see that postprocessing with a classical greedy algorithm can improve local solutions quite significantly.

It should be noted that as this method relies on running the full problem on the QPU first, we could only obtain results on our own dataset and not the larger problems in the University of Toronto dataset.

6.4 Hybrid Solvers

D-Wave offers two types of Hybrid Solvers to approach tackling optimisation problems using quantum-classical hybrid computing. These are (1) D-Wave Ocean's `dwave-hybrid` tool which allows for customising flexible hybrid workflows and (2) Leap Hybrid Solvers which are a cloud-based service that accept arbitrary quadratic models.

In this section, we investigate both methods and developed one further to obtain results and solve problems from the University of Toronto dataset as well as our dataset.

6.4.1 dwave-hybrid

D-Wave Ocean's `dwave-hybrid` software is a Python framework for the customisation of flexible hybrid workflows that makes use of both quantum and classical heuristics. The aim of this framework is to streamline development of experimental prototypes and allow developers to obtain insight into how a productised versions of a solver will perform.

`dwave-hybrid` solvers are built by arranging 'building blocks' which are components of the workflow in a specific order. By using this framework, large problems can be decomposed down into smaller sub-problems and run on separate samplers in parallel. Examples of components available are:

- **Samplers**: Quantum Samplers such as `QPUSubproblemAutoEmbeddingSampler()` and Classical Samplers such as `InterruptableTabuSampler()` and `InterruptableSimulatedAnnealingSampler()` which runs a classical local search tabu algorithm and simulated annealing respectively.
- **Decomposers**: Decompose large problems into smaller sub-problems that can be embedded onto the QPU. Examples of these are `EnergyImpactDecomposer()` which selects the sub-problem based on variables that are contributing to the system's energy the most, and `RandomConstraintDecomposer()` which selects random variables that are related to each other through specific constraints. There are many more examples of decomposer components offered by this framework but we mainly focused on the `EnergyImpactDecomposer()`.
- **Composers**: Merge together samples based on a pre-defined heuristic. The main example we investigated was `SplatComposer()` which merges and overwrites the current samples in the solver with the best samples from sub-problems.

The components can be created to form a branch in the form `{ decomposer | sampler | composer }` which decomposes the problem into sub-problems, sends the sub-problems to a sampler, then composes results from the sampler back together. Multiple branches can be created and these branches can be run in parallel using `dwave-hybrid`'s `RacingBranches` class.

Figure 6.2 shows an example of the workflow we built. We chose to split up the problem into two branches, one using a Tabu Sampler, which has found some success in exam timetabling optimisation problems [37], and the other using D-Wave's Quantum Sampler. We also experimented with the `size` and `rolling_history` parameters of `EnergyImpactDecomposer()` to modify the workflow to suit problems with larger number of variables. These parameters specified the size of sub-problems and the maximum percentage of variables each sub-problem would cover.

Results and Evaluation

This hybrid solver performs extremely well on our dataset of smaller problems (results in table 6.1), finding the lowest energy solutions out of all the solvers we have investigated. Despite this,

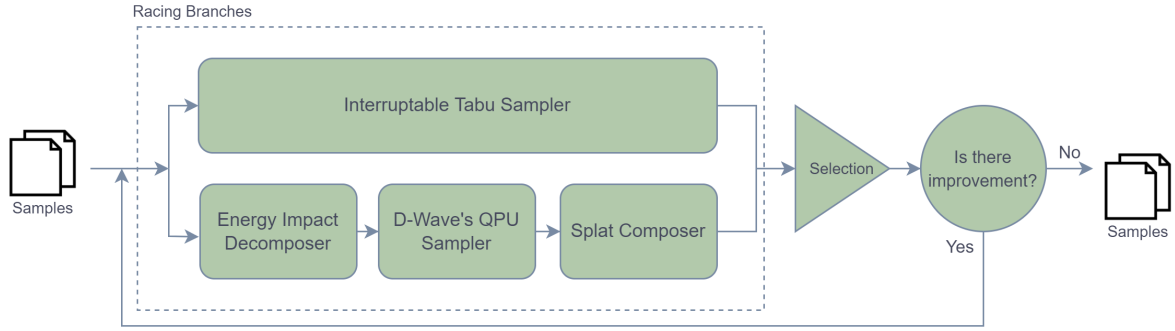


Figure 6.2: Example of the workflow investigated, built using the `dwave-hybrid` tool

it struggled on the larger problems in the University of Toronto Benchmark dataset. It did not manage to find a valid solution (with no direct clashes of exams) on any of the problems. Even with the modifications detailed above, there was not a significant increase in performance.

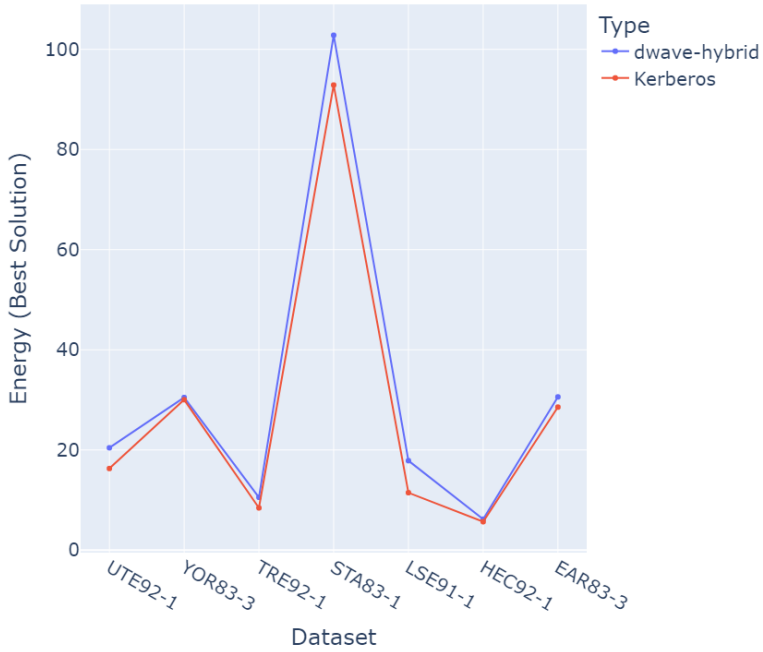


Figure 6.3: **Energy** of best solution for custom `dwave-hybrid` workflow (blue) and reference sampler Kerberos (red). The datasets omitted from this graph could not be embedded onto the solvers due to size. *Note that this is the energy of the best solution, not penalty scores of valid solutions.*

We also compared the results of our custom hybrid solver in figure 6.2 to `dwave-hybrid`'s reference hybrid sampler, **Kerberos** and we observed that neither solver could find a valid solution for any of the Toronto Benchmark Dataset problems. Figure 6.3 shows that the energy of the best solution from each solver does not differ by significant amounts.

Despite great results with the smaller datasets, we decided not to develop this solver any further as it was very experimental with a lot of moving parts, which made tuning of parameters a tedious process with little gain in performance. On top of that, poor performance from its reference sampler Kerberos also indicated that it would be difficult to tune the workflow to a

configuration that would work well on the larger datasets at all.

6.4.2 Leap's Hybrid Quantum Sampler

D-Wave Leap offers multiple hybrid solvers that use quantum-classical methods to solve problems formulated in arbitrary quadratic models (BQM, CQM and DQM). Unlike the `dwave-hybrid` solvers, Leap is a cloud-based service that handles all interactions with the QPU. Therefore, a high level of customisability, as offered in `dwave-hybrid` workflows is not available here. However, we chose to build and investigate Leap's hybrid solvers as they were built to solve larger optimisation problems using the strengths of both quantum and classical methods.

Formulating a Constrained Quadratic Model

We initially tested out Leap's Hybrid Solvers by formulating our problems as BQMs using the `BqmBuilder` detailed in chapter 4. While this solver still gave excellent results for the smaller datasets, we did notice it struggle to find valid solutions for the larger datasets.

This whitepaper published by D-Wave [38] investigates the performance difference between CQMs and BQMs. It found that due to CQM's feature of allowing native representation of constraints, it is more efficient than its counterparts at finding good low-energy solutions to constrained problems. This is in contrast to BQMs that have a computational advantage in an unconstrained problem. Due to the constrained nature of our optimisation problem, we made the decision to re-formulate the problem as a CQM instead. The reformulation of BQM to CQM itself was quite simple as CQM is an abstraction of the former. To do this, we had to specify constraints explicitly as opposed to in the QUBO form.

Results and Evaluation

We formulated each problem from the Toronto Dataset as a CQM. An issue we ran into was a large increase in computational time. We realised that this bottleneck was not due to a significant increase in computation time from the solver but rather the construction of the CQMs, as there were a large number of constraints to be created. We overcame this problem by saving these CQMs and loading these models locally when they needed to be used.

The Leap Hybrid Solver produced excellent results for the smaller datasets and valid solutions for *some* of the problems in the Toronto Benchmark Dataset. The solver was unable to produce results for the other datasets as Leap's service has a ceiling on the number of constraints allowed, which was 100,000. Table 6.2 below shows that we were unable to encode some of the problems in this dataset as they had constraints of more than 100,000.

6.5 Final Results and Summary

Table 6.1 shows the energy values of the best solutions produced by all the solvers we have investigated. We can see that while classical postprocessing methods can improve the solutions from a Quantum Annealer, the methods that yielded the best results were ones that reaped the benefits of both classical and quantum algorithms simultaneously.

In Table 6.2, we see that the problems that were unable to be mapped onto Leap's Hybrid CQM Solver had more than 100,000 constraints, which is the limit on the number of constraints that the service provides. However, for the problems that managed to be solved, we managed to obtain solutions with penalties were very close to the benchmark solution in the industry.

Dataset	Simulated Annealing	Quantum Annealing	QA with Postprocessing	dwave-hybrid	Leap Hybrid Solvers
small-01	5.798	5.798	5.798	5.798	5.798
small-01-cc	6.601	6.601	6.601	6.601	6.601
small-02	12.676	12.676	12.676	11.070	11.070
small-02-cc	5.296	5.403	5.296	5.296	5.296
small-03	14.788	14.788	14.433	14.433	14.433
small-03-cc	96.76	89.855	73.649	16.840	42.099
small-03-dis	1.603	1.297	1.297	1.297	1.297
small-03-cc-dis	3.554	3.673	3.561	1.297	1.297
med-01	4.823	5.208	3.908	3.846	4.509
med-01-cc	6.738	26.294	17.065	4.924	5.300
med-02	2.557	2.65	0.892	0.615	2.062
med-02-cc	37.794	85.346	80.025	2.369	4.452
med-02-dis	0.863	0.987	0.364	0.089	0.535
med-03	5.763	90.453	82.394	1.325	4.652

Table 6.1: **Energy** values of best solutions from each solver

Considering the Leap Hybrid Solver is still in early stages of development and constantly being upgraded with new features and optimisations, the results produced by Leap's Hybrid Solvers on the Toronto Benchmark problems are certainly promising.

Dataset	Version	Benchmark Penalty	Leap Solution Penalty	Number of Constraints
UTE92	1	24.769	27.648	14484
YOR83	3	34.708	-	103563
UTA92	2	3.044	-	865698
TRE92	1	7.717	-	141274
STA83	1	157.032	159.417	18092
PUR93	1	-	-	-
LSE91	1	9.818	14.734	81939
KFU93	1	12.901	-	118321
HEC92	1	10.050	12.577	24615
EAR83	3	32.626	-	115222
CAR92	1	3.707	-	650303
CAR91	1	4.394	-	1044172

Table 6.2: **Penalty** of best solution from Leap's Hybrid Sampler on Toronto Benchmark Dataset Problems

Dataset	Version	Benchmark Penalty	Leap Solution Penalty	Number of Constraints
UTE92	1	**24.769**	**27.648**	14484
YOR83	3	34.708	-	103563
UTA92	2	3.044	-	865698
TRE92	1	7.717	-	141274
STA83	1	**157.032**	**159.417**	18092
PUR93	1	-	-	-
LSE91	1	**9.818**	**14.734**	81939
KFU93	1	12.901	-	118321
HEC92	1	**10.050**	**12.577**	24615
EAR83	3	32.626	-	115222
CAR92	1	3.707	-	650303
CAR91	1	4.394	-	1044172

Chapter 7

Conclusion

7.1 Conclusions

Our first objective of this project was to **build a proof of concept** application using D-Wave's Quantum Annealer to solve exam timetabling optimisation problems. We managed to achieve this by first formulating our optimisation problem in the form of a Binary Quadratic model which was then embedded to D-Wave's QPU and annealed over to produce low-energy state samples. We found that we managed to successfully embed these problems, and for the most part, find a valid solution with a low-energy state. Therefore, we achieved our first objective.

The next aim we had was to **investigate the extent of the capacity of D-Wave's Quantum Processing Unit in embedding optimisation problems**. To achieve this goal, we wanted to find the largest optimisation problem whose BQM had a valid embedding onto the D-Wave QPU. This was to determine the size of optimisation problems that Quantum Annealers could solve by themselves (without help from classical algorithms). In section 5.3.2, we found that the largest dataset we could embed onto the D-Wave 2000Q computer was med-01-cc which was an optimisation problem of 15 exams and 5 days. Similarly, the largest dataset we could embed onto the Advantage system was med-03 of 30 exams and 10 days.

Additionally, we also wanted to investigate the **feasibility of finding a problem's optimal embedding**. In Chapter 5, we discussed the embedding of problems in detail. This is as the quality of problem embedding to the QPU is a critical factor in the performance of the Quantum Sampler when producing solutions for that problem. Some of the factors we learned that greatly affected the embedding were the connectivity of the topology of the QPU and the chain strength between qubits. While we did manage to improve performance of the quantum annealer through tuning certain parameters, we found that a lot of these parameters were specific to each problem and tuning each in a real-world context would be unfeasible.

Our third aim was to **analyse the quality of the solutions** produced by the Quantum Annealer. In Chapters 5 and 6, we learned that:

- Quantum Annealing could produce good solutions for the smaller problems, but struggled to produce near-optimal solutions when the complexity increased
- Quantum Annealing methods was significantly faster than the classical Simulated Annealing methods, even when the complexity of the problems increased
- With improvements to the embedding and annealing process, the Quantum Annealer did show an increase in performance. However, when the problem's BQM got too convoluted, chain lengths became too long and it proved quite difficult to get a decent solution from the Quantum Sampler.

- Quantum Annealing could benefit from applying classical algorithms in postprocessing to find local minimas of the samples from the Quantum Sampler.
- The best performance we observed from all solvers were from the solvers that combined quantum and classical methods simultaneously to solve the optimisation problems.

Finally, our last aim was to **explore the capabilities of a Hybrid Solver**. To achieve this objective, we used D-Wave Leap's Hybrid Solver and formulated problems in CQM form. We observed that the solver could find low-energy solutions to all the smaller datasets, most of which were better solutions than achieved by the Simulated Annealing sampler. We also managed to successfully embed benchmark problems from the University of Toronto dataset, and produce a valid solutions whose penalty scores were comparable to that of the industry-standard benchmarks. Therefore, to a greater extent, this objective was achieved.

7.2 Future Work

Through each of our objectives, we learned and achieved outcomes that were meaningful and indicative of the potential of Quantum Annealing in the field of optimisation. Following this project, there are many opportunities to build upon these results for further investigation and these are discussed below.

Experimenting with workflow structures using `dwave-hybrid`: As we have shown that customisable workflows built using `dwave-hybrid` are capable of producing extremely low-energy state solutions for smaller problems, it would be worth experimenting with a larger set of the workflow structures available through the framework. This raises an opportunity to evaluate the different algorithmic components and strategies offered by the framework on our own application.

Optimisation of embedding algorithms: We have learnt that the embedding of a problem's BQM to the QPU is critical in achieving lower-energy state solutions. D-Wave currently uses in-built function `minorminer` which runs a heuristic algorithm to find the optimal embedding of optimisation problems to the QPU topology. Optimising this algorithm gives us the opportunity to increase the quality of the solutions from the Quantum Annealer without needing expensive upgrades in the QPU. Research into this area has already begun, with this study proposing two new algorithms that are variations on `minorminer` [39].

Embedding Optimisation Problems to new Zephyr Topology: The size of optimisation problems that can be solved through our application is currently limited primarily by the qubit-capacity of the QPU. D-Wave is soon to be releasing a new Quantum Computer with a Zephyr Topology [40], which is to have an increased qubit-capacity and overall connectivity in qubits. Besides being able to embed larger problems, the increased connectivity would also boost the performance of our Quantum Annealer.

Bibliography

[1] Feynman RP. Simulating physics with computers. 1982;21:467–488. Available from: https://doi.org/10.1007/BF02650179.

[2] Deutsch D. Quantum theory, the Church–Turing principle and the universal quantum computer. 1985 Jul. Available from: http://doi.org/10.1098/rspa.1985.0070.

[3] Gardner R. L. V. Kantorovich: The Price Implications of Optimal Planning. Journal of Economic Literature. 1990;28(2):638-48. Available from: http://www.jstor.org/stable/2727266.

[4] Dantzig GB. Linear Programming and Extensions. Santa Monica, CA: RAND Corporation; 1963.

[5] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by Simulated Annealing. Science. 1983;220(4598):671-80. Available from: https://www.science.org/doi/abs/10.1126/science.220.4598.671.

[6] Das A, Chakrabarti B. Quantum Annealing and Related Optimization Methods. Quantum Annealing and Related Optimization Methods, Edited by A Das and BK Chakrabarti 2005 XIV, 378 p 124 illus Also available online ISBN 3-540-27987-3 Berlin: Springer, 2005. 2005 01.

[7] Simonis H. The CHIP system and its applications. In: Montanari U, Rossi F, editors. Principles and Practice of Constraint Programming — CP '95. Berlin, Heidelberg: Springer Berlin Heidelberg; 1995. p. 643-6.

[8] White GM, Xie BS, Zonjic S. Using tabu search with longer-term memory and relaxation to create examination timetables. European Journal of Operational Research. 2004;153(1):80-91. Timetabling and Rostering. Available from: https://www.sciencedirect.com/science/article/pii/S0377221703001000.

[9] Carter MW, Laporte G, Lee SY. Examination Timetabling: Algorithmic Strategies and Applications. The Journal of the Operational Research Society. 1996;47(3):373-83. Available from: http://www.jstor.org/stable/3010580.

[10] Burke EK, Newall JP, Weare RF. A memetic algorithm for university exam timetabling. In: Burke E, Ross P, editors. Practice and Theory of Automated Timetabling. Berlin, Heidelberg: Springer Berlin Heidelberg; 1996. p. 241-50.

[11] Miles R. Computer Timetabling: A Bibliography. British Journal of Educational Technology. 2006 10;6:15 18.

[12] Kadowaki T, Nishimori H. Quantum annealing in the transverse Ising model. Physical Review E. 1998;58(5):5355-63.

[13] Aron J. Google and NASA team up to use quantum computer;. Available from: https://institutions.newscientist.com/article/dn23554-google-and-nasa-team-up-to-use-quantum-computer/.

[14] Anthony S. First Ever Commercial Quantum Computer Now Available for $10 Million;. Available from: https://www.extremetech.com/computing/84228-first-ever-commercial-quantum-computer-now-available-for-10-million.

[15] Group HH. Harris & Harris Group Notes Sale of Quantum Computing System by D-Wave Systems to Lockheed Martin Corporation;. Available from: http://www.nanotech-now.com/news.cgi?story_id=42543.

[16] Perdomo-Ortiz DNDBMea A. Finding low-energy conformations of lattice protein models by quantum annealing. Scientific Report. 2012;2:571. Available from: https://www.nature.com/articles/srep00571.

[17] McGeoch CC, Wang C. Experimental Evaluation of an Adiabatic Quantum System for Combinatorial Optimization. 2013. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.592.5580&rep=rep1&type=pdf.

[18] Boixo S, Rønnow TF, Isakov SV, Wang Z, Wecker D, Lidar DA, et al. Evidence for quantum annealing with more than one hundred qubits. Nature Physics. 2014 Feb;10(3):218–224. Available from: http://dx.doi.org/10.1038/nphys2900.

[19] King J, Yarkoni S, Nevisi MM, Hilton JP, McGeoch CC. Benchmarking a quantum annealing processor with the time-to-target metric; 2015.

[20] Wang B. Dwave Systems shows off quantum chip with 2048 physical qubits;. Available from: https://web.archive.org/web/20150513044844/http://nextbigfuture.com/2014/10/dwave-systems-shows-off-quantum-chip.html.

[21] Team TDW. D-Wave Previews Next-Generation Quantum Computing Platform;. Available from: https://www.dwavesys.com/company/newsroom/press-release/d-wave-previews-next-generation-quantum-computing-platform/.

[22] Team TDW. What is quantum annealing?;. Available from: https://docs.dwavesys.com/docs/latest/c_gs_2.html.

[23] Appendix: Next Learning Steps;. Available from: https://docs.dwavesys.com/docs/latest/c_gs_9.html#getting-started-advanced.

[24] D-Wave QPU Architecture: Topologies;. Available from: https://docs.dwavesys.com/docs/latest/c_gs_4.html.

[25] Denchev VS, Boixo S, Isakov SV, Ding N, Babbush R, Smelyanskiy V, et al. What is the Computational Value of Finite-Range Tunneling? Physical Review X. 2016 Aug;6(3). Available from: http://dx.doi.org/10.1103/PhysRevX.6.031015.

[26] Rønnow TF, Wang Z, Job J, Boixo S, Isakov SV, Wecker D, et al. Defining and detecting quantum speedup. Science. 2014 Jul;345(6195):420–424. Available from: http://dx.doi.org/10.1126/science.1252319.

[27] Lucas A. Ising formulations of many NP problems. Frontiers in Physics. 2014;2. Available from: https://www.frontiersin.org/article/10.3389/fphy.2014.00005.

[28] Benchmark Data Sets in Exam Timetabling;. Available from: http://www.cs.nott.ac.uk/~pszrq/data.htm.

[29] Ocean DW. dimod;. Available from: https://docs.ocean.dwavesys.com/en/stable/docs_dimod/.

[30] Cooper TB, Kingston JH. The complexity of timetable construction problems. 1996:281-95.

[31] Systems DW. Constraints Example: Minor-Embedding;. Available from: https://docs.dwavesys.com/docs/latest/c_gs_7.html#getting-started-embedding.

[32] Leap DW. The Only Real-Time Quantum Cloud Service Built for Business;. Available from: https://www.dwavesys.com/solutions-and-products/cloud-platform/.

[33] Inc PT. Collaborative data science. Montreal, QC: Plotly Technologies Inc.; 2015. Available from: https://plot.ly.

[34] Inc PT. Dash Python User Guide;. Available from: https://dash.plotly.com.

[35] Systems DW. Annealing Implementation and Controls;. Available from: https://docs.dwavesys.com/docs/latest/c_qpu_annealing.html.

[36] Marshall J, Venturelli D, Hen I, Rieffel EG. Power of Pausing: Advancing Understanding of Thermalization in Experimental Quantum Annealers. Physical Review Applied. 2019 apr;11(4). Available from: https://doi.org/10.1103%2Fphysrevapplied.11.044083.

[37] Petrovic S, Bykov Y. A Multiobjective Optimisation Technique for Exam Timetabling Based on Trajectories. In: Burke E, De Causmaecker P, editors. Practice and Theory of Automated Timetabling IV. Berlin, Heidelberg: Springer Berlin Heidelberg; 2003. p. 181-94.

[38] Hybrid Solver for Constrained Quadratic Models;. Available from: https://www.dwavesys.com/media/rldh2ghw/14-1055a-a_hybrid_solver_for_constrained_quadratic_models.pdf.

[39] Zbinden S, Bärtschi A, Djidjev H, Eidenbenz S. Embedding Algorithms for Quantum Annealers with Chimera and Pegasus Connection Topologies. In: Sadayappan P, Chamberlain BL, Juckeland G, Ltaief H, editors. High Performance Computing. Cham: Springer International Publishing; 2020. p. 187-206.

[40] Systems DW. Zephyr Topology of D-Wave Quantum Processors;. Available from: https://www.dwavesys.com/media/2uznec4s/14-1056a-a_zephyr_topology_of_d-wave_quantum_processors.pdf.