UNSW AUSTRALIA

SCHOOL OF MECHANICAL

AND MANUFACTURING ENGINEERING

Solving NP-Hard Problems on an Adiabatic Quantum Computer

by

Dan Padilha

dan.padilha@student.unsw.edu.au

Student ID: z3291677

Thesis submitted as a requirement for the degree of
Bachelor of Engineering (Aerospace Engineering)

Academic Supervisor

Dr. Mark Whitty

Industry Supervisor

Michael Brett
Q^x Branch, LLC.

27th October, 2014

Statement of Originality

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed Danilo Domingues Padilha

Date 27$^{\text{th}}$ October, 2014

Abstract

Quantum computation is heralded as the next big breakthrough in computer technology. Although still in its infancy, quantum computing is expected to eventually tackle the hardest known computational problems, such as cryptography and chemical process simulation. Algorithms with exponentially faster performance already exist in preparation for this quantum future. However, applications of hard computational problems on real quantum computers are still virtually non-existent. This thesis aims to demonstrate the application of a D-Wave Two, the most advanced adiabatic quantum computer in the world, to solving fundamentally hard problems. Simple real-world applications are demonstrated by exploiting the optimisation properties of quantum annealing, and the scaling and performance of these problems is investigated. The results show the wide applicability of adiabatic quantum computation, and provide a first step towards further research of hard problems on future generations of quantum hardware.

Acknowledgements

On the quantum journey, one does not take the road less travelled, but rather all the roads at the same time. Roaming this quantum road for the first time was a long and challenging endeavour. Along the way, several people provided great support in the form of time, patience, and understanding. For this, I am particularly grateful to them:

My supervisor, Dr. Mark Whitty, for foolishly choosing to entangle himself in the madness of this topic. For actually listening to and enjoying my quantum drivel. For making a massive effort to understand everything and suggest extremely useful approaches, as well as for his patience while I sorted out all the red tape.

The Q^xBranch team, including: Michael Brett (for collapsing the wavefunction into a state of giving me access to the D-Wave), Shaun Wilson, Dr. Kingsley Jones (especially for the Fratelli incident), Duncan Fletcher, Tristan Cook, and all the others who assisted in the background.

Dr. Andrea Morello, for the email exchanges with valuable insights into quantum computation, for the encouragement and support for taking on this project, and for putting up with my silly questions.

My friends and family, for occasionally pulling me back into the real (macroscopic) world. For their understanding during my Schrödinger's cat moments ("We don't know if he's dead or alive."), and for putting up with the barely coherent quantum puns.

Table of Contents

List of Figures

List of Referenced Figures

Section 1
Introduction

While quantum computation as a theory has been around for decades, the signs of a real quantum revolution are only being revealed today. It has been heralded as the next major technological step in computing, with wide-ranging applications from cryptography to data processing, artificial intelligence to protein folding, financial modelling to space exploration. Where a traditional computer might iterate through an exponential number of solutions to a problem, a quantum computer is theoretically capable of seeing them all at once – leading to huge potential speed-ups in the processing of information.[26]

Of course, this is the optimistic view, and a quantum computing future, if it ever occurs, is still decades away at best. It was only in 2011 when the world's first fully-programmable quantum computer became commercially available, with a staggering 128 programmable qubits (quantum bits), and a large dose of skepticism to follow. Today, quantum computation is effectively in the same place that computers were in the mid 20^{th} century: massive, room-sized boxes with less computational memory than a floppy disk, or as small-scale prototypes in research laboratories incapable of any useful computation.

While it is proven that quantum algorithms exist problems that would outperform classical computation on specific problems, it is not yet known whether quantum computers will be more effective for all, or even any other, types of problems. Examining how these behave experimentally on actual quantum computers is an important fundamental step towards an understanding of quantum computing's potential.

Figure 1.1 – The D-Wave Two, the world's only existing commercial quantum computer. It is housed in a large magnetically-shielded super-cooled environment.

For the purposes of this thesis, the author has partnered with Q^xBranch, LLC., a US-based quantum computing venture, as part of a collaborative research agreement with Lockheed Martin, owners of a D-Wave Two quantum computer.

The D-Wave Two, as seen in Fig. 1.1, is the world's only commercial quantum computer[1], of which two are publicly known to exist in commercial use, including Lockheed Martin's. It is an advanced 512-qubit machine capable of performing a process known as *adiabatic quantum computation*.

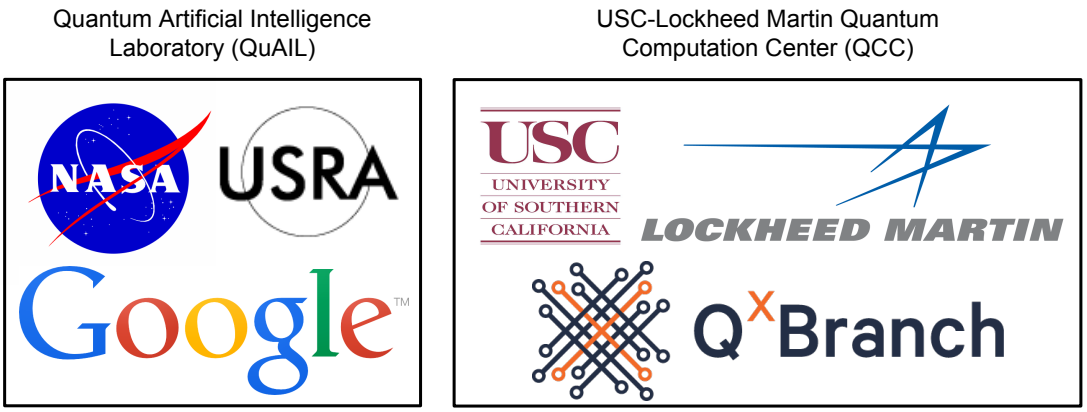


Figure 1.2 – D-Wave Two commercial customers, as of May 2014.

With such limited availability, the research community on large-scale adiabatic quantum computation is extremely small, and is largely composed of small groups at each of the participants listed in Fig. 1.2 (as well as D-Wave Systems themselves). Consequently, this thesis likely contains the very first Australian developments in large-scale adiabatic quantum computing.

1.1 Objectives

The main focus of research around D-Wave's quantum computers has been in determining the quantum properties of the hardware, as well as benchmarking it for randomly-generated problems. Until now, very little published research has experimentally shown actual applications of the D-Wave computers, and the behaviour of **NP**-Hard problems. As quite a new computing platform, there is yet a long way to go to truly grasp the capabilities of adiabatic quantum computing. Such a journey must begin with the fundamentals, by applying and understanding the behaviour of problems that form the building blocks for the rest.

[1]Except for its predecessor, the D-Wave One, which is no longer produced.

This thesis aims to be a part of that beginning, by demonstrating several applications of fundamental **NP**-Hard problems likely for the first time on this computing architecture. In particular, the aims are to:

- Develop algorithms of **NP**-Hard problems for use on an adiabatic quantum computer.
- Demonstrate **NP**-Hard problems being solved successfully on an existing platform, such as the D-Wave Two.
- Evaluate the behaviour of these problems and the early-generation hardware they are solved on.
- Identify potential areas of research to be investigated by the rapidly growing AQC community.

The approach taken in this thesis is to build on the formulations of several problems described by Lucas (2014) [45]. These formulations were chosen for being part of Karp's[40] **NP**-Complete problems, a set of problems that form the basis of most useful computationally hard problems. In essence, this thesis takes one of the first, small steps – from theory to implementation – in the journey towards real-world applications of quantum computers. The problems solved in this thesis, while seemingly simple, add to this growing knowledge, and serve as potential stepping stones for more complex applications to come.

1.2 Thesis Outline

Section 2 introduces the fundamental knowledge required in understanding the results of this thesis. This includes the basics of computational complexity, adiabatic quantum computation, and the specifics of the D-Wave Two hardware. This sets up Section 3, in which the most significant and relevant research to date is discussed. This section identifies the gap in current knowledge of actual algorithms and results for **NP**-Hard problems on adiabatic quantum computing hardware.

Section 4 describes in detail the process of programming a D-Wave Two quantum computer, and the general methodology for testing used throughout the thesis. This is followed by Sections 5 to 10 introduce the algorithms developed for this thesis, as well as discuss how each behaves on the D-Wave Two hardware.

Finally, Section 11 describes the overall findings and conclusions of this thesis, as well as a brief discussion on potential future topics for research.

Section 2
Fundamentals

Due to the somewhat complex and obscure nature of the presented topic, this thesis begins by first introducing some fundamental aspects of quantum computation. All knowledge required to understand the motivations and results of this thesis is presented here – from the basics of complexity (Section 2.1) and quantum computation (Section 2.2), to the Adiabatic Quantum Computation architecture (Section 2.3) and its implementation in the D-Wave processors (Section 2.4).

Concepts are presented here in a simplified manner. While some knowledge of the mathematical underpinnings of quantum computation is important for its applications, quantum physics theory is best left to the physicists, and is not required knowledge for this thesis.

2.1 Complexity of Problems

In order to understand what problems are best suited to quantum computers, one must first turn to *computational complexity theory*. Computational complexity theory allows us to quantify the computational resources (such as processing time or memory) required to solve certain types of problems[7].

In order for a problem to be computationally "feasible", it must be computable in what's known as *polynomial-time*. This refers to the number of computational steps required to find a solution, which must be a polynomial function n^k for a problem of size n (and some constant k).

Any problem that takes more than a polynomial number of steps to compute (for example, an exponential k^n number of steps) is considered infeasible, as the computation time grows too quickly with the problem size, requiring more time than the age of the universe to compute a problem of sizes as small as $n \approx 100$.

Of course, this is not to imply that designing algorithms to solve infeasible problems is a waste of time – rather, the opposite is true: while infeasible problems may require exponential time with size, heuristics (approximations) and clever mathematical trickery help alleviate the scaling of the computational requirements for problems of practically useful sizes. This is the primary basis of modern algorithmic design, and it is likely that quantum computers may bring their own layer of mathematical trickery for further improvements.

2.1.1 Complexity Classes

Problems are grouped into *complexity classes*, based on how feasible they are. The most relevant complexity classes, along with their relationships, are pictured in Fig. 2.1.

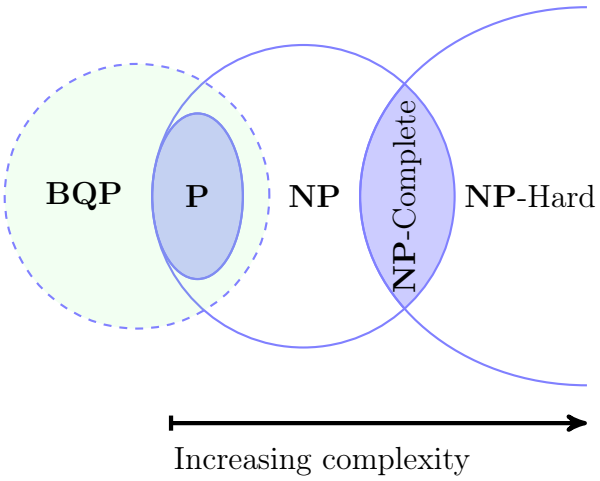


Figure 2.1 – Diagram of some basic complexity classes.[1]

P (Polynomial-time)

Problems in **P** are those that can be **solved efficiently** (in polynomial-time). These problems are considered feasible even with classical computers.

Examples: integer multiplication, sorting, searching through a list

NP (Non-deterministic Polynomial-time)

Problems in **NP** are those that can be **verified efficiently** (a valid solution can be confirmed in polynomial-time).

Assuming that $\mathbf{P} \neq \mathbf{NP}$[2], **NP** problems that are not also in **P** can not possibly be efficiently solved.

Examples: all problems in **P**, integer factorisation

NP-Hard and NP-Complete

NP-Hard problems are those that are at least **as hard as every problem in NP** (equivalent to solving an **NP** problem with some additional overhead). Such problems are **NP**-Complete if they are **also in NP**.

[2]Whether $\mathbf{P} = \mathbf{NP}$ or $\mathbf{P} \neq \mathbf{NP}$ is considered one of the most fundamental unsolved problems in computer science, but general concensus holds that $\mathbf{P} \neq \mathbf{NP}$.[7]

NP-Hard problems can not be solved efficiently. Unfortunately for the real-world, the vast majority of useful problems are provably **NP**-Hard, including fundamental problems that are widely applicable across many disciplines (from cryptography to scheduling to solving Sudokus, and much more).[7]

Fortunately, good enough solutions exist such that small enough **NP**-Hard problems can be solved with acceptable amounts of effort, but there is always room for improved algorithms.

Examples: travelling salesman, halting problem, map colouring

NP-Intermediate

Although not pictured in Fig. 2.1, these are simply **problems in NP that are neither NP-Complete, nor P**. While efficient classical solutions for them may exist, none have ever been found, nor have they been proven to be **NP**-Complete.

Examples: integer factorisation

BQP (Bounded-error Quantum Polynomial-time)

Problems which can be **solved efficiently** (in polynomial-time) using a quantum computer. Such problems are the main focus of quantum algorithms research.

The **BQP** class includes problems from **P** and at least some from **NP**-Intermediate. However, **BQP** does not contain **NP**-Hard/Complete problems.

Examples: integer factorisation (**NP**-Intermediate), simulation of quantum systems

2.1.2 Implications

It is unfortunately believed that neither classical nor quantum computers can ever solve any **NP**-Hard/Complete problem in polynomial-time. If that is the case, then what benefits over classical computing, if any, does quantum computation provide?

For one, it is likely that quantum computers may be able to find more efficient algorithms for certain types of problems. Not all exponential-time algorithms for a problem are equal, as different algorithms scale differently with different coefficients. It is likely that the inherent properties of a quantum computer may lead to improved

scaling coefficients for some problems, even if this does not result in exponential speed-up.

For another, the class of problems efficiently solved by quantum computers (**BQP**) is known to contain at least some problems that are not efficiently solved by classical computers (**P**). A specific example, such as integer factorisation, is known to have an efficient algorithm on quantum computers (Shor's algorithm[70]), but no efficient algorithm has ever been found on classical computers.[3] A much simpler example is Grover's algorithm[34], which provides a sub-linear-time search of an unsorted database, something that is provably impossible with classical algorithms. Quantum physics simulation is yet another problem that is easy for quantum computers but hard for classical, and is not only a fundamental problem for numerous other scientific fields, but one which will likely lead to extremely significant scientific discoveries in the long-term.[3]

Overall, the focus of quantum algorithm development is on identifying problems which are more suited to being solved using quantum mechanical phenomena. Mathematically, this may mean problems like integer factorisation, better performance in **NP**-Hard problems, or general quantum simulation. These cases are where the true significance of quantum computation lies – as a long-term investment in computational efficiency (and being able to extract the full computational power of the physical universe), and, even more importantly, the capability for humans to accurately simulate nature at an atomic level.[2]

[3]This is not to say that an efficient classical algorithm can not exist – rather, integer factorisation is an **NP**-Intermediate problem simply because it has not yet been shown to be either in **NP**-Complete or in **P**. Perhaps a classical polynomial solution is simply too difficult to find.

2.2 Quantum Computing

> *Let a computer smear – with the right kind of quantum randomness – and you create, in effect, a 'parallel' machine with an astronomical number of processors . . . All you have to do is be sure that when you collapse the system, you choose the version that happened to find the needle in the mathematical haystack.*
>
> – **Greg Egan** (1994) [27], *Quarantine*[4]

Quantum computing is a rapidly-emerging technology bringing the power of quantum mechanics into the world of computer science. Conventional computers, when simulating any type of quantum system or solving certain types of problems, require an exponential amount of effort that grows quickly with the problem size (such that computing their solutions is considered to be "infeasible" or "intractable"). For these simulations or problems, quantum computing offers the possibility of "feasible" computation, and is therefore seen as a potentially huge leap in computational efficiency. This section briefly describes some important fundamentals of quantum computation.

2.2.1 Qubits

The quantum bit, or *qubit*, is the fundamental piece of information in quantum computation, similar in function to its classical counterpart the *bit*.

While a bit may only ever be in one of two states (commonly referred to as 0 or 1), a qubit is always in a probabilistic superposition of two states (referred to as $|0\rangle$ or $|1\rangle$, or as up/down *spins*). It always has some probability of being in either of the two states, and it is this property that is behind the power of quantum computation.[5]

Rather than performing operations on each state separately, a quantum computer can instead operate on both simultaneously by indirectly manipulating the probabilities of each state. Much like Schrödinger's cat (Fig. 2.2), as soon as the qubit is measured or observed, it collapses into one of the two basis states ($|0\rangle$ or $|1\rangle$), depending on the probabilities at the time.[26]

[4] As seen in Aaronson (2005) [1].

[5] In reality, a qubit does not have a probability of being in each state per se, but rather *probability amplitudes*, which are complex numbers and can even be negative.

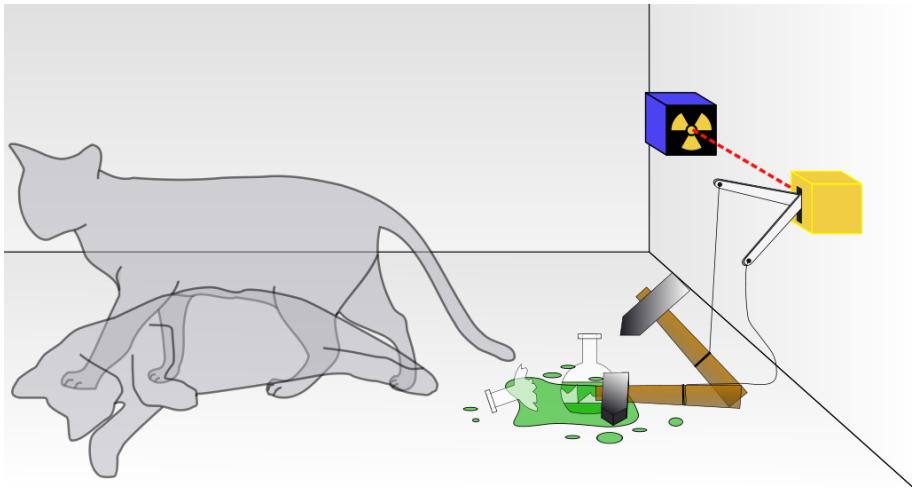


Figure 2.2 – Schrödinger's cat is placed in a sealed box. There is a random chance that an atom in the radioactive source will decay, triggering a flask of poison to be shattered and killing the cat. Quantum mechanics holds that the cat remains in a superposition of being both simultaneously dead and alive, until the box is opened and the true state of the cat is finally observed.

2.2.2 Decoherence and Entanglement

In order for quantum computation to work, a quantum system must be completely sealed against outside interference. Of course, in the real world, this is a virtually impossible property to maintain. As such, quantum computation suffers from a phenomenon known as *decoherence*: the collapse of a quantum system due to interactions with the environment.[5] The larger the system, the more interaction it has with its environment, and the quicker it decoheres. Exposing Schrödinger's cat to an outside observation (by opening the box) is an external interaction, which causes an instantaneous decoherence and collapse of the cat's quantum state into a single basis state of $|dead\rangle$ or $|alive\rangle$. Ensuring long enough coherence in order to exploit the computational power of quantum systems is still one of the biggest roadblocks in developing large-scale quantum computation.

For "true" quantum computation, one other phenomenon is required. That is *quantum entanglement*, in which pairs of qubits (after all, it takes two to tangle[6]) have highly-correlated states. Schrödinger's cat and the radioactive source are, for example, a quantum entangled pair, with only two possible states of $|dead, decayed\rangle + |alive, undecayed\rangle$. As soon as the box decoheres, the quantum system collapses into

[6]Courtesy of [26].

one of the two states, and the cat and radioactive source immediately lose their entanglement. In this sense, one can not have entanglement without having *coherence*.

Quantum entanglement is extremely powerful, as it allows for an exponential number of states to be encoded in a single entanglement. Schrödinger's cat and the radioactive source are two variables, each with two possible states ($|dead\rangle$ or $|alive\rangle$, $|decayed\rangle$ or $|undecayed\rangle$). Their entanglement thus encodes the probabilities of an exponential number of $2^2 = 4$ possible states: $|dead, decayed\rangle$, $|dead, undecayed\rangle$, $|alive, decayed\rangle$, $|alive, undecayed\rangle$. This exponential state compression, where N qubits encode 2^N states, can not be efficiently simulated by classical computation. It is this that makes entanglement the true power behind the exponential speed-ups potentially offered by quantum computation.[26]

2.2.3 Quantum Computing Architectures

The basic requirements for building a quantum computer have now been established:

Qubits – capable of storing a superposition of states.

Coherence – to ensure the probabilities of states remain unaffected by the environment for as long as possible.

Entanglement – to encode an exponential number of states for a given number of qubits.

As it turns out, there are many different ways of creating quantum computers. Fig. 2.3 shows that quantum computers are only one way of physical computation possible (the other three ways being referred to as "classical" computation), and that quantum computation is itself divided into different architectures.

The four quantum architectures seen in Fig. 2.3 are those most commonly found in current literature. Of these, this thesis is primarily concerned with the Adiabatic model, detailed in Section 2.3.

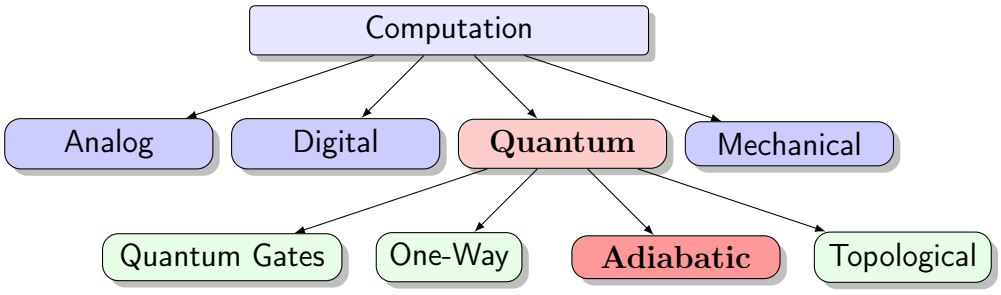


Figure 2.3 – List of physical systems of computation and quantum computing models. The model studied specifically in this thesis is the Adiabatic Quantum Computation model.

2.3 Adiabatic Quantum Computing (AQC) Model

The Adiabatic Quantum Computing (AQC) Model is a quantum computational architecture with similarities to Simulated Annealing optimisation algorithms. In this sense, AQC is primarily concerned with combinatorial optimisation problems. This section describes the theory behind AQC and the basics of programming a quantum computer built on an AQC architecture, as is the case with the D-Wave Two.[21]

2.3.1 Simulated and Quantum Annealing

One can imagine some problems as describing landscapes of mountains and valleys, where the height of the land represents the value or *energy* of that particular location. For optimisation problems, the goal is usually to find the deepest valley (global minimum) in the landscape.

A Simulated Annealing algorithm (as seen in Fig. 2.4) works by slowly scaling up the energy landscape, allowing a point to explore – effectively rolling up and down the mountains and valleys to find the global minimum.

Quantum Annealing works similarly, but due to quantum mechanical effects (such as the superposition of states), it is much easier to find the global minimum as the exploration can effectively "tunnel" through large barriers, such as a tall mountain separating two very deep valleys.

As will be shown shortly, Adiabatic Quantum Computation is effectively a form of quantum annealing.

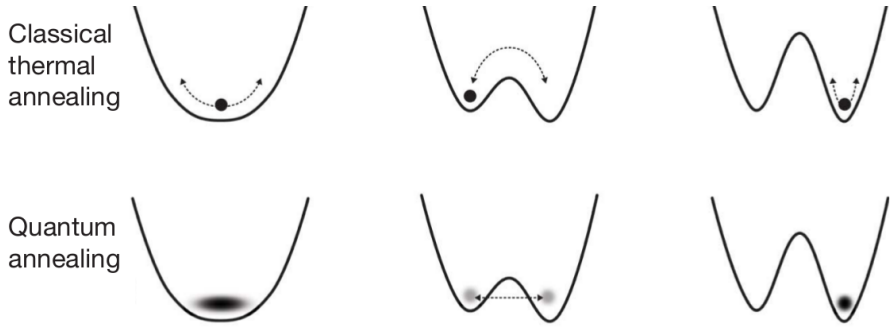


Figure 2.4 – The energy landscape of a problem, as it is traversed by a classical Simulated Annealing algorithm and a quantum annealer.

2.3.2 Hamiltonians

An operator known as the Hamiltonian is used when describing quantum systems, usually defined as H, $\hat{H}$, or $\mathcal{H}$, to represent the energy landscape of a problem.

The Hamiltonian defines all the possible energy states of a system, known as the Hamiltonian's *spectrum*. If one imagines the Hamiltonian of a quantum system to be a matrix H, then its spectrum is simply the set of all its eigenvalues λ for all possible states Ψ the system can be in:

$$H\Psi = \lambda\Psi \tag{2.1}$$

The *ground state* of a Hamiltonian refers to the state with the minimum amount of energy (the global minimum). A Hamiltonian is usually defined such that the ground state has zero energy, denoted as $\lambda_0 = 0$.

The *1st excited state* of a Hamiltonian refers to the next possible state immediately above the ground state, with its corresponding energy value λ_1.

It is important to understand that while an energy landscape may look continuous, a Hamiltonian has a discrete set of states. No state exists, for example, between the ground state and 1st excited state.

2.3.3 Adiabatic Evolution

A quantum system in a given state Ψ and described by a Hamiltonian H evolves over time ∂t according to Schrödinger's equation[28], a fundamental formulation in quantum mechanics playing a similar role to that of Newton's laws in classical mechanics:

$$i\frac{\partial}{\partial t}\Psi = H\Psi \tag{2.2}$$

The *adiabatic theorem* of quantum mechanics states that a quantum system, if evolved slowly enough, will remain in the same eigenstate at all times.[29]

To exploit this theorem, one can begin with a quantum system described by an *initial Hamiltonian* H_B, with a ground state that is easy to find. If the system is evolved to match a *problem Hamiltonian* H_P (describing an energy landscape from which the global minimum is desired), then by the adiabatic theorem (evolving slowly enough), at the end of the evolution the system will still be in the ground state, and thus the global minimum of H_P is found. The whole process acts essentially as a form of quantum annealing.

2.3.4 Evolution Time

The adiabatic theorem holds only when the evolution from the initial Hamiltonian to the problem Hamiltonian is done slowly enough. The total time required for an adiabatic evolution (a single anneal), t_a, is given by the adiabatic theorem as:

$$t_a \gg \frac{1}{g_{min}^2} \tag{2.3}$$

Here, g_{min} is the *minimum energy gap* – the smallest gap between the ground state and the 1st excited state of the Hamiltonian at any given time, as defined by:

$$g_{min} = \min_{0 \leq s \leq 1} (\lambda_1(s) - \lambda_0(s)) \tag{2.4}$$

This inverse relationship implies that the smaller the minimum energy gap is, the longer the evolution needs to be in order to ensure it stays in the ground state. Of course, as with many things quantum, actually calculating the minimum energy gap is generally a difficult process and is effectively equivalent to solving the actual problem itself.

2.3.5 Complexity

Adiabatic Quantum Computation has been shown to be at worst polynomially worse than conventional quantum computation (the quantum gates model), but no better.[4] Fortunately, this implies that the AQC model is capable of universal computation.

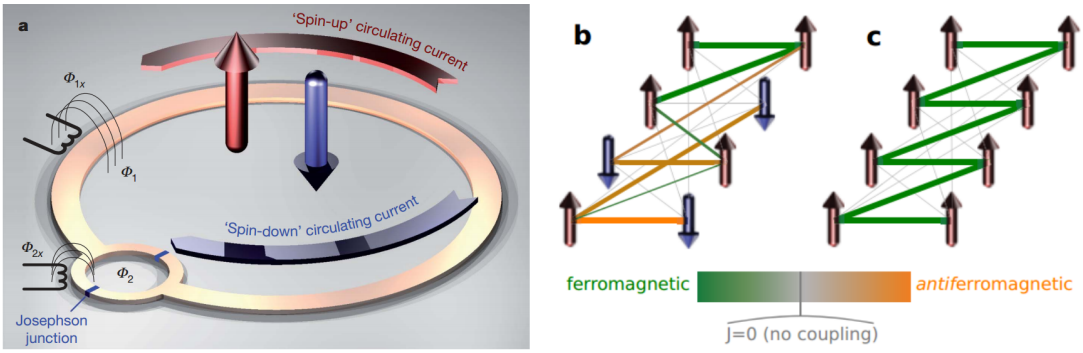
As the Adiabatic model requires only the ability to represent an easy Hamiltonian H_B, a difficult Hamiltonian H_P, and to slowly transition between them, AQC has had much faster technological development than conventional quantum computing. This is what has been exploited in the design of the D-Wave quantum computers.

2.4 D-Wave Two Quantum Computer

The D-Wave Two quantum computer is a device capable of performing adiabatic quantum computation on a set of 512 qubits. The computer works by setting up an initial Hamiltonian H_B with an easy-to-find ground state, then slowly evolving to a problem Hamiltonian described as an Ising spin model (Section 2.4.2) or Quadratic Unconstrained Binary Optimisation (Section 2.4.3).

To ensure quantum coherence, the D-Wave Two system is housed in a controlled environment, being magnetically shielded from external sources, and cooled to temperatures as low as 20 mK to minimise thermal noise (as well as for the operation of the superconducting qubits).

2.4.1 Qubits



(a) Qubit **(b)** Couplings

Figure 2.5 – A qubit and its couplings as represented on a D-Wave Two.

Qubits are represented as superconducting loops (Fig. 2.5a) such that the resulting direction of the electric current in the loops defines the qubit's state or "spin". In essence, during computation the direction of the electric current is unknown (in a probabilistic state), but when the system decoheres (due to observation or noise), the current spins either one way or the other ("up" or "down").

In order to perform useful computations, it is necessary to carefully control the probability distribution of the qubit's spin (in other words, which direction it is most likely to end up spinning). This is achieved by programming the *bias* and *couplings* of each qubit.

The *bias* is a programmable magnetic field applied to the qubit's loop, such that its presence and strength slightly nudges the current spin in a desired direction. Each qubit is also connected to other qubits using programmable ferromagnetic/anti-

ferromagnetic *couplings*, which force qubits to spin in either the same or opposite direction of qubits they are connected to (Fig. 2.5b).

2.4.2 Ising Model

The D-Wave Two solves problem Hamiltonians described as an "Ising spin glass", as in Eq. (2.5):

$$H_P = \sum hs + \sum s' J s \tag{2.5}$$

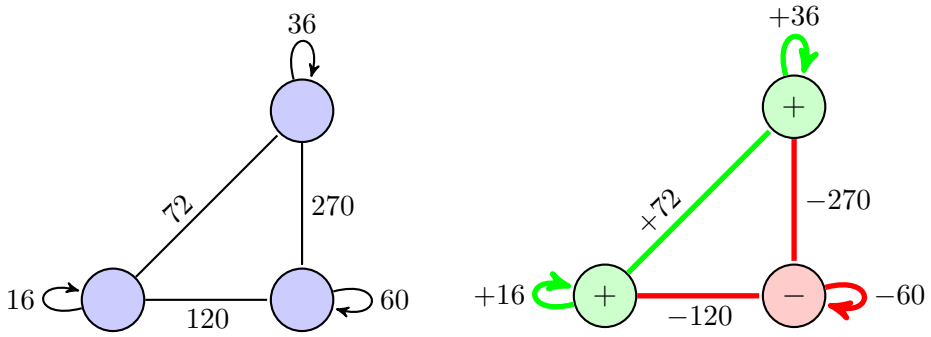
In this model, s and s' are spin variables, each of which corresponds to a single qubit. Specifically, spin variables must take the form of $s = \pm 1$, with qubits having the same direction of current being assigned the same value of spin.

h is an applied *bias* which takes a scalar value that is limited by the hardware in range and resolution (a limitation that can cause issues in precision for solving certain problems). J is a coupling strength between two spin variables s and s', which determines how strongly to couple their spins together or apart (and is similarly limited in range and precision by the hardware).

The precision limitations of the bias and couplings are particularly important for some problems that depend on delicate balances of strengths between spin variables. D-Wave lists the bias and couplings as being a Gaussian distribution centered around the value chosen, with a standard deviation of around $\sigma \approx 0.15$, on a scale of values between $[-1, 1]$.[59] When biases and couplings are normalised to this range, little room is left for distinguishing between values.

Eq. (2.5) describes the problem Hamiltonian H_P, which defines the amount of energy for a given configuration of spin variables. Programming a problem involves setting the values of h and J for each qubit and coupling on the device, then performing an adiabatic evolution. According to the adiabatic theorem, the end result should be the ground (lowest energy) state of the problem Hamiltonian H_P – in practice, of course, this depends on the minimum energy gap, time of evolution, and environmental decoherence effects. The spin of each qubit s is measured at the end of this process, and this set of qubit spins defines the lowest energy state found.

An example of how the couplings and biases of some qubits define the system's energy can be seen in Fig. 2.6.



(a) Problem graph **(b)** Ground (lowest energy) state

Figure 2.6 – Example programmed graph and minimum-energy solution for a problem. In (a), the couplings between qubits and biases of each qubit are programmed. In (b), the lowest-energy configuration of spins is found, with the contribution of each coupling and bias to the energy. By Eq. (2.5), $H_P = 36+72+16-270-60-120 = -326$.

2.4.3 Quadratic Unconstrained Binary Optimisation (QUBO)

The Quadratic Unconstrained Binary Optimisation (QUBO) model is simply another way of defining Ising spin glasses, and is given by Eq. (2.6):

$$H = \sum x' Q x \tag{2.6}$$

Here, x and x' are spin variables with a value of $x \in \{0, 1\}$, and again, each corresponding to a single qubit. Q is a coupling strength between the QUBO spin variables x and x', except for the case when $x = x'$, in which Q is a bias for the spin variable x.

It is sometimes more useful to define problems in terms of either Ising spin variables ($s \in \{+1, -1\}$) or QUBO spin variables ($x \in \{0, 1\}$). Both models are equivalent and easily programmed into the D-Wave Two, and both models can be converted to one another with the following relation:

$$s = 2x - 1 \tag{2.7}$$

2.4.4 Chimera Graph

There are 512 qubits, which are arranged as an array of 8-qubit cells, each cell a bipartite graph $K_{4,4}$[7] of coupled qubits and couplings to adjacent cells. This arrangement is known as the *Chimera graph* (Fig. 2.7) and plays an important role in defining problems for the D-Wave Two.

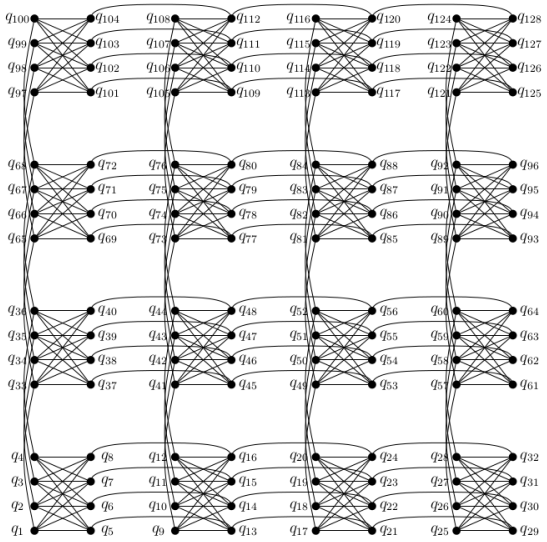


Figure 2.7 – A Chimera graph consisting of 4×4 cells, each a bipartite graph $K_{4,4}$ of qubits.

The most important property of the Chimera graph is that it is *not* a fully-connected graph – each qubit is coupled to at most 6 other qubits, not to every single other qubit. This means that any problem that works on a graph of decision variables[8] that does not exactly match the Chimera structure, must be *embedded* into the Chimera graph.

Embedding refers to transforming a graph to conform to the structure of another graph – that is, transforming the problem graph into the Chimera graph available on the D-Wave Two Hardware. This is done by chaining multiple qubits together such that the chain represents a single decision variable, effectively adding the connections from every qubit along the chain. Much of the theory behind this is found in Choi (2008) [16], and a heuristic to do graph embedding is provided as part of the D-Wave programming interface.[14]

[7] $K_{i,j}$ notation refers to a complete bipartite graph, where i vertices are each connected to j vertices, for a total of ij connections between the two sets of vertices.

[8] Note that the terms "spin" and "decision variable" are used interchangeably. A qubit is simply the physical realisation of a spin.

In the worst-case scenario of embedding a fully-connected graph into the D-Wave Two Chimera graph, the maximum size that can be embedded, assuming a perfect embedding, is 33 variables (requiring 512 qubits to represent). However, use of the D-Wave heuristic to embed graphs often results in worse embeddings, limiting the maximum size of fully-connected graphs that can be embedded.

Section 3
Literature Review

The following literature review expands on the current state of the art in adiabatic quantum computation and its applications.

3.1 Adiabatic Quantum Computation (AQC)

The Adiabatic Quantum Computation (AQC) model was first described by Farhi et al. (2000) [29], who introduced the idea of applying adiabatic evolution of a quantum system as an algorithm for solving certain combinatorial optimisation problems. Farhi et al. argued that for such an algorithm to be useful, the running time to solve must grow at worst polynomially in the number of qubits. They showed that this time was dependent on the minimum spectral gap (g_{min}) between energy states of the Hamiltonians, and that this gap was non-trivial to compute for harder problems.

Further work in Farhi et al. (2001) [30] showed that simulations of AQC were capable of solving harder randomly generated **NP**-Complete problems, and demonstrated small growth of running time in number of qubits, albeit only for small numbers of qubits (due to the space constraints of simulating quantum systems classically). Furthermore, they showed that the time evolution of the Hamiltonians could be approximated using discrete unitary operators, as found in the conventional quantum computing model (in essence proving that AQC could be simulated by conventional QC), but that there was little advantage to doing so and that AQC could instead be an alternative model for the design of real-world quantum computers.

As it was known that AQC could be efficiently simulated by conventional quantum computers, its computational power could not exceed that of the conventional model. It was Aharonov et al. (2004) [4] that first proved that the converse was also true – that the conventional quantum computing model could be efficiently simulated in AQC (with a polynomial complexity overhead, considered small enough to be "efficient"). By extension, this proved both models to be equivalent. Perhaps more important[1] was their additional proof that any quantum computation could be efficiently simulated through AQC using only two Hamiltonians operating on a two-dimensional grid of qubits.

[1] For the purposes of the development of the D-Wave devices.

Also importantly, Childs, Farhi and Preskill (2001) [15] showed that the AQC model had inherent fault tolerances not present in the conventional quantum computing model. As AQC performance depends on the spectral gap g_{min}, environmental decoherence could be minimised by running an AQC device at low temperatures compared to g_{min}. In addition, AQC was also tolerant of errors introduced due to hardware implementation of the problem Hamiltonian, assuming the error was either slowly varying, or rapidly varying (with a frequency large compared to the spectral gap).

It was the above results that led to D-Wave Systems' first major patent in Amin and Steininger (2006) [6] and eventually led to their seminal work in M. Johnson et al. (2011) [38], in which they described the design and validation of the D-Wave One's processor "Rainier". The Rainier processor consisted of an integrated circuit with 128 flux qubits, arranged in 8-qubit cells with bipartite coupling and couplings between cells. Rainier's architecture essentially allowed for a programmable Ising model Hamiltonian, with controllable magnetic fields being used for qubit biases, and the tunable couplers acting as coupling strengths between qubits.

The contribution of M. Johnson et al. (2011) was fairly significant, introducing for the first time an adiabatic quantum computer that could be programmed and provide a solution readout for relatively large problem sizes. Previous adiabatic quantum computers were either unable to be programmed (as they relied simply on the physical configuration of matter), or were incapable of scaling beyond two or three qubits. The experiments described in the paper showed how a single qubit and a single cell of the processor demonstrated quantum annealing properties, a result that was later extended to 108-qubits by an independent team in Boixo et al. (2013) [13].[2] Of course, M. Johnson et al. were quick to note that the sparse connectivity between qubits (the "Chimera graph", as described in Section 2.4.4) meant that the processor was not quite yet a universal quantum computer.

In the time since M. Johnson et al. (2011), multiple independent groups have performed analyses to determine the true worth of D-Wave's computers. McGeoch and Wang (2013) [47] were able to demonstrate significant (orders of magnitude) speed-up of specific problems on the newer D-Wave Two processor as compared to existing classical software solvers. These results were quickly rebutted by others, and a general scientific consensus on the quantum nature of the D-Wave computers

[2] Although a case against this result was later posed by J. A. Smolin and Smith (2013) [75].

has yet to be reached.[3] Indeed, this controversial debate has already lead to significant efficiency gains in classical computation, with highly optimised classical algorithms being shown by Selby (2013a) [66] to be capable of matching or exceeding the performance of the D-Wave machines. Despite this, it has not stopped the likes of NASA, Lockheed Martin, and Google from acquiring their own D-Wave computers for algorithm development, each remaining optimistic about their future applications.[76]

3.2 NP-Complete Problems as Ising Spin Glasses

Attempting to solve difficult computational problems with more efficient algorithms has been a major focus of computer science research for several decades.[18] The inherent capability of quantum computers to represent an exponential number of states has led to quantum algorithms proven to run in faster time than classical algorithms for certain problems (the important examples of this being Shor's integer factorisation and Grover's unsorted database search[34, 70]). While this has quickly shown the power of quantum computation, it is still an open question as to whether quantum computers will prove to be significantly (or perhaps at least marginally) faster than classical computers for solving **NP**-Complete problems.[4]

The D-Wave quantum computers, as described previously, essentially present a programmable model of an Ising spin glass. The problem of spin glasses has been studied extensively in physics literature, and its computational complexity was first described in Barahona (1982) [9]. Barahona showed that finding the ground state of an Ising spin glass is an **NP**-Hard problem. There is an implicit link between **NP**-Complete and **NP**-Hard problems, in which problems of either class can be converted into any other problem with at worst polynomial-time overhead – a process known as polynomial reduction. Barahona's result therefore implies that all **NP**-Complete problems can be polynomially reduced to an Ising spin glass (that is, any **NP**-Complete/Hard problem can be solved using the Ising model).

Indeed, these results make the potential applications of the D-Wave processors and Adiabatic Quantum Computation much more obvious. We know that hard

[3] While the question of whether the D-Wave computers can be considered "true" quantum computers is highly important, it lies far beyond the scope of this thesis. We make the assumption that either the D-Wave machines are true quantum computers, or that they behave sufficiently like one to be capable of running quantum algorithms on real problems.

[4] See Section 2.1 for further discussion on this point.

decision problems can be mapped to an Ising spin glass and that finding the ground state (lowest energy state) of an Ising spin glass is also difficult. However, if we begin with a spin glass with an easy to find ground state, and anneal it over time to the problem we want to solve, then in theory we remain in the ground state throughout the process and we therefore end with the lowest energy solution to the problem. It is thus clear that finding Ising spin glass formulations for **NP**-Complete problems can lead to useful algorithms for Adiabatic Quantum Computers.

Multiple teams of researchers have tried applying random variations of Ising problems to Adiabatic Quantum Computation in an effort to better understand its performance characteristics. Previously-mentioned research in Boixo et al. (2013) [13] and McGeoch and Wang (2013) [47] showed not only quantum-esque behaviour of the D-Wave processors, but also significant potential speed-up of problems. In Crosson et al. (2014) [20], Crosson, Farhi, and Shor[5] simulated randomly-generated hard instances of Ising problems and found that faster annealing times, as well as adding a random intermediate Hamiltonian during the annealing process, may lead to higher success probabilities (for certain problems). These results serve to give us an indication of both the current and future potential of AQC devices, and that current performance evaluations focus on randomly-generated Ising problems, rather than on real-world **NP**-Complete problems.

Perhaps the most practically useful research for real-world Ising formulations can be found in Lucas (2014) [45]. Here, the mathematical Ising formulations of numerous **NP**-Complete and **NP**-Hard problems are described, as are techniques for imposing the constraints of the problem in auxiliary spins, and for reducing the number of spins required. Lucas provides formulations for all of "Karp's 21 **NP**-Complete problems", a set of fundamental problems which Karp proved to be **NP**-Complete[40]. These fundamental problems form the basis upon which more complex problems are proven to be computationally hard – as such, it is likely that these formulations can provide a useful foundation for harder problems to be solved in an AQC context.

[5]Farhi of the seminal work on AQC in Farhi et al. (2000) [29], and Shor of the highly important quantum factorisation algorithm in Shor (1997) [70].

3.3 Applications of AQC

It is obvious to see that, despite being theorised for decades, real-world quantum computation is still very much in its infant years. While there are numerous research groups around the world currently designing and fabricating quantum computers[6], it is safe to assume that the D-Wave processors are currently the only existing commercially-available quantum computers in the world for any significant number of programmable qubits.[26, 50] This means that the capability to develop and test commercial applications for quantum computers currently lies primarily in those few companies with D-Wave access: Google, NASA, and Lockheed Martin.

NASA, as part of the Quantum Artificial Intelligence Laboratory (QuAIL), have focused their research on performance evaluation.[13] Lockheed Martin's research partners at the University of Southern California (USC) also focused on performance and behaviour evaluation – however, they also proposed a method for performing machine learning on a quantum adiabatic processor[61] and other methods, but no published direct implementations on D-Wave hardware. Lockheed Martin have themselves focused on applications to software verification.[33]

In Neven et al. (2009) [53], Google demonstrated an image classification machine-learning algorithm applied on an early prototype D-Wave processor.[7] They showed that "QBoost" (Quantum Boosting algorithm) was capable of providing a strong image classifier with better accuracy than a classical search, despite the small number of available qubits. However, their algorithm did not perform as well as a specialised classical boosting algorithm. They attributed this to the limitations of the D-Wave hardware, which required the problem set to fit on the few qubits available, as well as the loss of qubit interactions due to the sparsity of the Chimera graph structure. The increased number of qubits in the D-Wave One led their work to be expanded in Neven et al. (2012) [52], which found that QBoost produced more generalised classifiers (rather than over-fitting) and required less computational effort for training than the specialised boosting algorithm.

In Smelyanskiy et al. (2012) [73], a joint NASA/D-Wave Systems/USC paper, Smelyanskiy et al. explored various potential applications of adiabatic quantum algorithms for different applications specific to spaceflight. In the same vein as Lucas (2014), they developed several mappings from real problems to Ising formulations.

[6] Including the world-leading Centre for Quantum Computation & Communication Technology at UNSW.

[7] It was this paper and its follow-up work which inspired Pudenz and Lidar (2011) [61].

However, neither paper describes actual implementations or results on D-Wave hardware.

The above research comprised all of the real-world applications of D-Wave hardware known at the beginning of this thesis. For these few applications, little implementation detail was provided, which required developers accessing the D-Wave hardware for the first time to climb a steep learning curve before being capable of implementing even the simplest of applications. With this in mind, this thesis was conceived to demonstrate simple implementations of hard problems on the D-Wave hardware.

3.4 Further Developments

The AQC research community has worked recently at a very rapid pace, with numerous publications having appeared since the commencement of this thesis. Several of these are of particular note, as they further validate the need for knowledge of real-world applications of AQC. This includes McGeoch (2014) [46], who notes that "Many **NP**-Hard optimization and decision problems have yet to be considered. [...] Much more work is needed to understand the full potential of this new approach to problem-solving." Two very recent papers from the NASA group (including authors from Smelyanskiy et al. (2012) [73]) show the steps taken between defining a problem to solving and benchmarking it on a D-Wave Two. These papers are directly relevant in their similar aims to this thesis.

In Rieffel et al. (2014) [62], the authors demonstrate three methods for scheduling-type planning problems on a D-Wave Two, building on one of the applications identified by Smelyanskiy et al. (2012). Rieffel et al. identify several points of importance in successfully implementing a problem, including the definition of the problem and properties of the embedding and annealing profiles. While their work does not result in a competitive solution (relative to classical algorithms), they conclude that improved hardware would contribute to improving their results.

In Perdomo-Ortiz et al. (2014) [56], a fault-detection/diagnosis system is described as a quantum annealing problem, then implemented on a D-Wave Two. The work demonstrates the capability to model problems of sizes useful in real-world applications within the constraints of the current generation hardware, although still much smaller than what is possible by classical solvers. They note that the quantum annealing approach is useful in that it not only returns the best solution found, but also many solutions close to it – this is in contrast to classical tools they compare

against, which return only a single solution. This property proves useful in cases where the most optimal solution may not actually be useful in the real world.

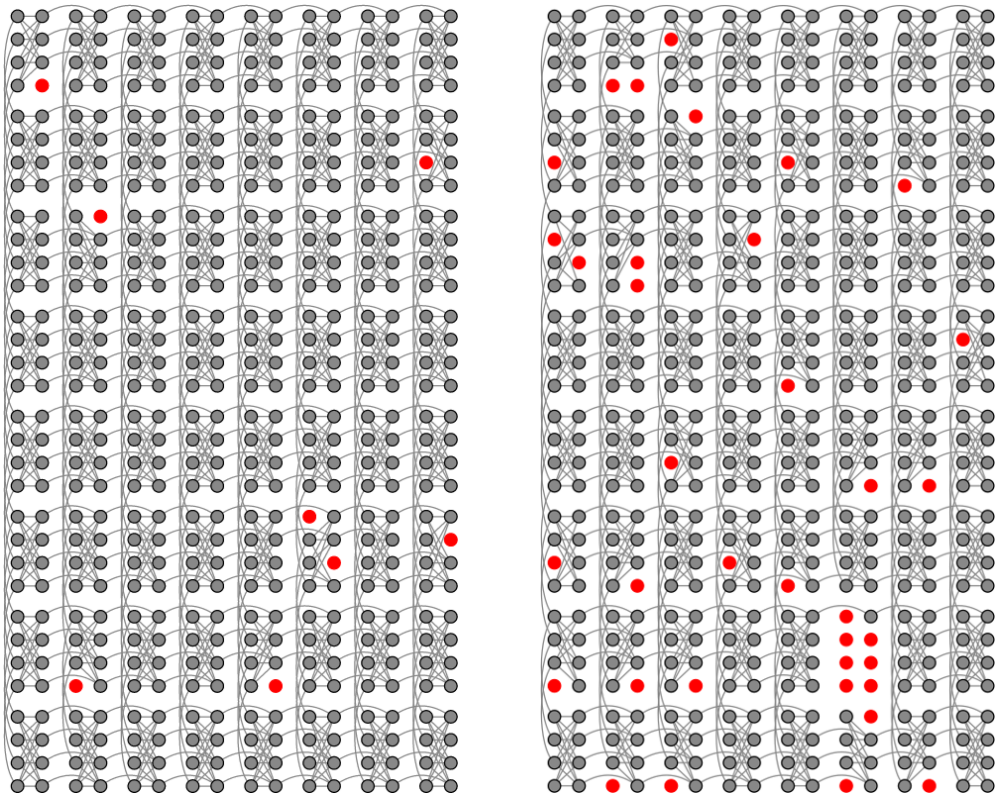
While these two papers demonstrate the application of real-world problems on the D-Wave hardware, they do not demonstrate fundamental problems such as those identified by Karp (1972) [40]. In that sense, the aims of this thesis are subtly different – to investigate applications of hard fundamental problems which may provide the building blocks for more complex algorithms on the D-Wave hardware. There is still a distinct lack of problem implementations, and this in particular applies to the fundamental problems. Investigating the application and behaviour of these types of problems is a small but significant step towards the full understanding of the capabilities of Adiabatic Quantum Computing.

Section 4
Methodology

This section details the overall process of programming the D-Wave Two quantum computers. Access to two machines was provided by Lockheed Martin and D-Wave Systems through the author's involvement with Q^xBranch, LLC.

4.1 Access to D-Wave Twos

Throughout this thesis, access was provided to two D-Wave Two machines: one located at the University of Southern California's campus (provided by Lockheed Martin), and one located in Canada (provided by D-Wave Systems). Both machines presented effectively unique quantum environments, with different calibrations of the adiabatic evolution profile, different equilibrium temperatures, and different qubit characteristics. Information regarding these differences was not readily available. However, the Chimera graph for each machine was easily accessible, and showed that they differed significantly between machines (as per Fig. 4.1).



(a) USC-Lockheed Martin (504 qubits)

(b) Canada (476 qubits)

Figure 4.1 – Comparison of the different Chimera graphs provided by each D-Wave Two machine. Qubits in red are unusable.

Access to the USC-LM machine was limited to short timeslots spread evenly throughout each 24 hour period. As such, experimentation with the machine during the early period of the thesis was restricted, and most experimentation was instead done using software solvers capable of simulating only up to 128 qubits.

Access to the Canadian machine, was provided for 24 hours a day, at the expense of competing with many other researchers for priority. As such, solutions would take significantly longer than when using the USC-LM machine.

A timeline of access to both machines can be seen in Appendix C.

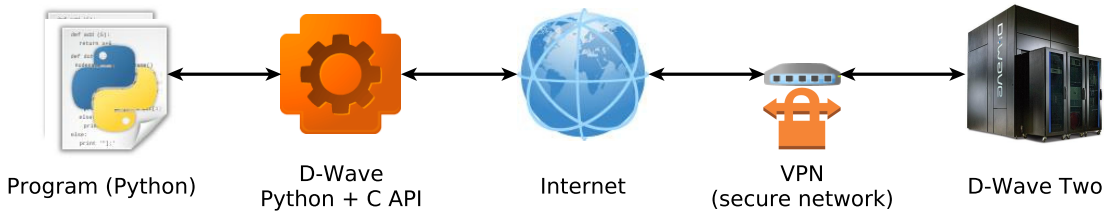


Figure 4.2 – Chain of access to the D-Wave Two machine.

Access to the machines was provided through a web interface and through programming libraries, communicating to each machine through secure VPN connections over the internet. The primary delays in solving problems come from the communication chain (Fig. 4.2), as well as the priority of the user on the D-Wave machine. Problems sent to the machine are placed in a queue before being solved, a process that can take some seconds overall.

4.2 Programming the D-Wave Twos

For this thesis, D-Wave's Python Application Programming Interface (API) was used to generate and set up problems and send them (over a secure internet connection) to one of the machines. There, the problem would be programmed on the quantum annealing device, results would be measured, and sent back to the original program.

Setting up a problem consists of various steps in order to prepare it to be solved on a D-Wave Two. These steps are outlined in the general process seen in Fig. 4.3 on the next page. This process is repeated for each problem described later in this thesis. This section describes in detail each step of the process.

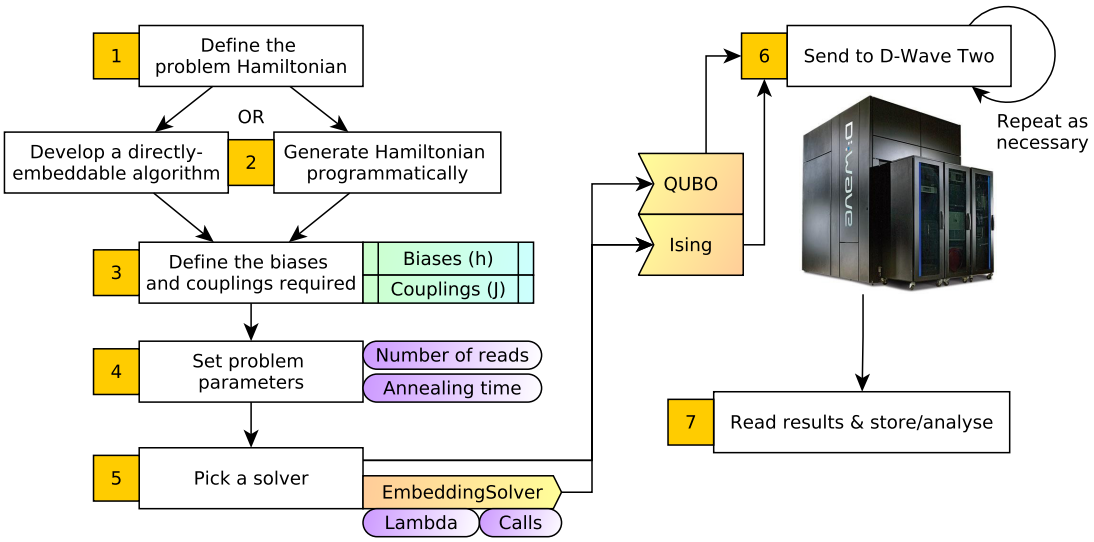
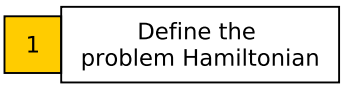


Figure 4.3 – General process for programming the D-Wave Two machine.

4.2.1 Problem Hamiltonian



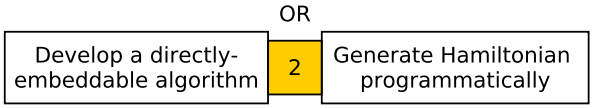
The first step of the process is to define the problem Hamiltonian, H_P. As stated previously in Section 2.3.3, the problem Hamiltonian defines the energy landscape of the problem to be solved.

In essence, the problem Hamiltonian is a function which defines the energy value of any given combination of variables. Given some Ising spin variables s_i or some QUBO spin variables x_i (see Sections 2.4.2 and 2.4.3), H_P defines a value for any given combination of these spin variables – in other words, H_P is defined as a mathematical function of s_i or x_i.

For use in a quantum annealing context, the problem Hamiltonian should generally define the lowest possible energy (the *ground state*, as per Section 2.3.2) only to combinations of spins that match the best solution to the problem. Good design of the Hamiltonian ensures large gaps between successive states, such that it is easy for the annealing to settle in lower energies.

This step is perhaps the most studied in AQC literature. Numerous papers exist which frame problems in the form of a Hamiltonian suitable for use on AQC hardware, including the works of Lucas (2014) [45], Smelyanskiy et al. (2012) [73], and others.

4.2.2 Algorithm Development



Once a problem Hamiltonian has been defined, it must be converted into an algorithm that takes a given problem into a format that can be programmed on to the D-Wave Twos. In this thesis, two methods of doing this were explored: *directly-embeddable* Hamiltonians, and *programmatic* Hamiltonians.

A directly-embeddable Hamiltonian is simply a mathematical function equivalent to that of the problem Hamiltonian, but expressed in such a way as to match either the Ising (Eq. (2.5)) or QUBO (Eq. (2.6)) formats. This is accomplished through analysis of the problem Hamiltonian, re-writing it in such a way as to define the couplings and biases required for each spin variable matching a certain condition (depending on the problem). Examples of this approach include Sections 5 to 7.

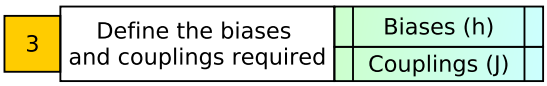
A programmatic Hamiltonian, on the other hand, is built by simply defining the given problem Hamiltonian as a symbolic mathematical expression within the software. When using Python, this can be done using the `sympy` library for symbolic mathematics. Each term of the problem Hamiltonian is defined as a symbol, and these symbols are assembled together to match H_P exactly. The symbolic library is used to then simplify the Hamiltonian until it matches the Ising or QUBO formats. Examples of this approach include Sections 8 and 9. The implementation details for these are omitted for the sake of brevity, but are made available as part of the electronic supplement of this thesis.

While the two approaches are essentially equivalent (simply representing H_P in a different format), they have slight nuances in usefulness.

The directly-embeddable Hamiltonian effectively defines an algorithm: a process that can be followed to assign biases and couplings to any given problem. However, such an algorithm is developed through inspection of the problem Hamiltonian, and thus requires an investment of time to understand the problem Hamiltonian, and may be difficult to do for more complex Hamiltonians.

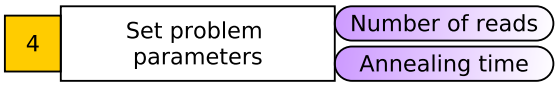
The programmatic Hamiltonian, on the other hand, is easy to program by its very nature. However, there is significant overhead in the operations done by the symbolic library, and the process does not directly result in an algorithm which can easily be ported to another programming language.

4.2.3 Biases and Couplings



Extracting the biases and couplings required for each spin variable is very simple. Once the Hamiltonian is in an Ising or QUBO format (Eqs. (2.5) and (2.6)), the biases and couplings can be extracted directly (being the h and J values).

4.2.4 Problem Parameters

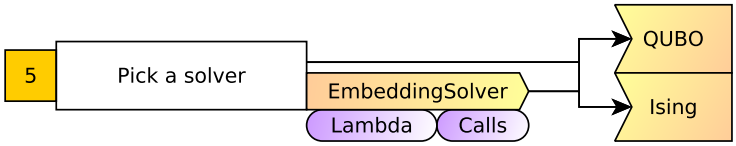


When a problem is sent to the D-Wave Two, it applies and performs an annealing on the problem multiple times, returning a list of solutions to each anneal. The number of anneals performed is a parameter that can be tweaked as necessary.

Similarly, the amount of time it takes for a single anneal to occur can also be tweaked, being limited to a minimum time of 20μs. The annealing time may potentially have an impact on solutions returned, due to the size of the minimum energy gap g_{min} of the Hamiltonian (refer to Section 2.3.4).

The total computational time for a single problem is thus comprised of the number of anneals done multiplied by the time for a single anneal. This total time is limited to $1,000,000\mu$s (1 second). Thus, for a minimum annealing time of 20μs, a maximum of $50,000$ anneals can be performed per single problem sent to the D-Wave.[1]

4.2.5 Solvers



The next step is to select a solver from the D-Wave API. While there are several different solvers serving different purposes, only two types are considered in this thesis: the `EmbeddingSolver`, and the conventional QUBO and Ising solvers.

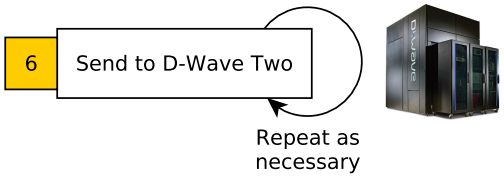
[1] In reality, the maximum number of anneals is lower, as it also depends on a third tweakable parameter known as the thermalisation time. This parameter did not appear to significantly affect results based on limited testing.

The conventional QUBO and Ising solvers are self explanatory: they simply take the biases h and couplings J and return a list of solutions (equal to the number of anneals chosen in the previous step).

The `EmbeddingSolver` is used for abstracting away the details of embedding an arbitrary graph into the D-Wave Two's Chimera graph (refer to Section 2.4.4). An embedding is typically found by D-Wave's heuristic `find_embedding` function, which searches randomly for a valid embedding in the graph.[16] For this reason, a seed is provided to the random search so that the embeddings generated are always the same for all tests performed in this thesis (except where stated otherwise). Any fully-connected graph of size N will have the same embedding, for example, regardless of bias or coupling choices.

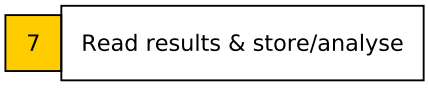
The embedding is then provided to the `EmbeddingSolver`, which becomes a conventional QUBO/Ising solver that takes the biases and couplings of the arbitrary problem graph. The `EmbeddingSolver` also contains several tweakable parameters, however these are mostly left to their default values and are not investigated in detail.

4.2.6 Sending the Problem



The problem is sent to the D-Wave Two by simply calling the solver chosen in the previous step. In the majority of the tests performed, the same problem is sent multiple times to the D-Wave Two to counteract the inherently random effects of noise on the solutions. The average distribution of the results can thus be obtained for the given problem.

4.2.7 Analysing the Results



The response from the D-Wave Two consists of:

- A list of all unique solutions found (or all solutions found, when returning "raw" results).

- A list of the energy values corresponding to each solution, sorted by the lowest energy.
- A list of the number of times each solution was seen (from the total number of anneals performed).
- Depending on the solver used, a table of timing information (including time to program the chip, establish thermal equilibrium, etc.)

The results are normally stored using a Python `pickle` object, so that they can be restored at any time. Analysis typically consists of determining trends in the probability distributions of the results as functions of problem type, size, annealing time, and so on, and determining what issues arise from properties of the quantum hardware. Except where otherwise stated, most problems were run a number of times with the same settings, and the distributions of the solutions averaged out.

This analysis step is the most critical and lacking in the AQC literature, particularly for fundamental **NP**-Hard problems. As such, small advances in the understanding of such problems, as is done in this thesis, are fundamental to the future development of AQC algorithms and hardware.

A given solution occurring num times out of R anneals has a probability of being found in any given anneal of $r = {}^{num}/_{R}$. This can be converted to an expected number k of anneals required to find this solution with p probability confidence given this equation:[62]

$$k = \frac{\log(1-p)}{\log(1-r)} \tag{4.1}$$

Normally, a confidence probability of $p = 0.99$ is used. Multiplying the expected number of anneals by the time taken for a single anneal gives an approximate expected computational time to find that solution:

$$t_{sol} = kt_a = \frac{\log(1-p)}{\log(1-r)} t_a \tag{4.2}$$

For example, for an annealing time of $t_a = 20\mu\,\mathrm{s}$ and probability of being found of $r = 0.5$, the expected computational time would be $t_{sol} = \frac{\log 0.01}{\log 0.5} * 20 = 132.9\mu\,\mathrm{s}$.

4.3 Python Environment

For the sake of completeness, the following is a comprehensive list of the external Python modules that were utilised throughout the thesis:

`dwave_sapi` – D-Wave's proprietary Python API. Significant details of the API are omitted for privacy reasons.

`algolib` – a custom utility library written for this thesis to perform commonly occurring procedures, such as extracting biases and couplings from Hamiltonians, reducing qubit interactions, loading and storing results, and so on.

`sympy` – a symbolic mathematics library, used for programmatic Hamiltonians.

`numpy` – for providing useful mathematical functions, particularly on arrays of data.

`matplotlib` – for plotting all graphs and diagrams.

`scipy` – providing generic scientific functions, such as fitting curves to data.

`networkx` – a graph library used for generating random graph problems, determining properties of graphs, and producing images of graphs.

4.4 Results

The following sections introduce several algorithms for **NP**-Hard problems, demonstrate their implementations on the D-Wave Two, and perform some analysis of their computational performance. The work presented is primarily based on definitions of problem Hamiltonians as per Lucas (2014) [45]. As such, the work presented in this thesis demonstrates steps 2 through to 7 of the general process (Fig. 4.3).

In total, more than 70 hours of total computational time were used for this thesis on the D-Wave Twos available. This involved solving approximately $150,000$ problems, each taking an average of 1.6 s over approximately $35,000$ anneals, for a total of $300 \sim 500$ million anneals.

Section 5
Algorithm: Number Partitioning

The `Number Partitioning Problem` is an **NP**-Complete decision problem that computes whether a given set S of positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of all elements in S_1 equals the sum of the elements in S_2. The difference Δ between the sums of subsets is defined as:

$$\Delta = \left| \sum S_1 - \sum S_2 \right| \tag{5.1}$$

The equivalent **NP**-Hard problem is to find a *perfect partition* of S. Note that when the total sum of the original set S is odd, there will never be a partition where $\Delta = 0$, as the minimum possible difference is $\Delta = 1$. As such, a *perfect partition* of S is defined as any partition in which $\Delta \leq 1$.

An equivalent **NP**-Hard optimisation problem is to find a partition of S with the smallest possible Δ.

The `Number Partitioning Problem` is known as a *weakly* **NP**-Hard problem due to the apparent availability of polynomial-time (efficient) solutions – however, the inherent hardness of the problem arises from the arbitrarily large precision required to represent the numbers in the set S.[31] This leads to the phenomenon of an easy-hard phase transition, which is explored later in Section 5.4.

The `Number Partitioning Problem` has applications in processor allocation and VLSI circuit optimisation, as well as in its reduction into more complicated **NP**-Hard problems. As a simple example, let each number in S represent the time required to perform a task. A minimum Δ partition would split the list of tasks as evenly as possible over two processors. The problem is also closely related to the `Bin Packing` and `Knapsack` problems, with important applications in finance and logistics.[72]

In this section, an algorithm capable of solving all three versions of the `Number Partitioning Problem` is introduced, and its performance on the D-Wave Two is characterised.

5.1 Lucas Formulation

The problem's Ising formulation, as given in Lucas (2014) [45] under Number Partitioning, is first described. Consider a set S containing N positive integers, $S = \{n_1, \ldots, n_N\}$. As this is an Ising model, the variable s_i is taken to represent a quantum spin with value of $s_i = \pm 1$. The problem Hamiltonian is thus described as:

$$H = \left(\sum_{i=1}^{N} n_i s_i \right)^2 \tag{5.2}$$

Each number n_i is assigned its own spin variable s_i, such that the two subsets S_1 and S_2 are essentially represented as the $+1$ subset and -1 subset. Taking the sum of each set multiplied by its spin simply finds the difference between the sums of both sets. Clearly, if this difference is 0, then both sets must have the same sum. Squaring this term ensures that the ground state (minimum energy) of the Hamiltonian is at $H = 0$ (two subsets with equal sum), and all other states have a positive energy $H > 0$. Thus, minimising the Hamiltonian is equivalent to minimising the difference between the sums of each subset.

5.2 Mapping to D-Wave hardware

To convert this Hamiltonian to a problem representable on a D-Wave computer, Eq. (5.2) is first fully expanded:

$$\begin{aligned} H &= (n_1 s_1 + n_2 s_2 + \cdots + n_N s_N)^2 \\ &= (n_1 s_1)^2 + (n_2 s_2)^2 + \cdots + (n_N s_N)^2 + 2 n_1 n_2 (s_1 s_2) + 2 n_1 n_3 (s_1 s_3) + \ldots \\ &= n_1^2 s_1^2 + n_2^2 s_2^2 + \cdots + n_N^2 s_N^2 + 2 * (n_1 n_2 (s_1 s_2) + n_1 n_3 (s_1 s_3) + \ldots) \end{aligned} \tag{5.3}$$

To convert Eq. (5.3) into qubit biases and coupling strengths, some observations must be made:

- An Ising spin s_i, when squared, will always be $+1$. As such, these terms have no direct impact on the spins in the hardware, and can be removed from the Hamiltonian.
- The remaining terms are a sum of every possible combination of spins. Thus, for N integers, there are a total of $\binom{N}{2}$ combinations of spins. This implies that there exists a fully-connected graph of spins.

- For each connection between two spins s_i and s_j, the energy added to the system is $n_i n_j(s_i s_j)$. Thus, a coupling strength of $n_i n_j$ is applied to every combination of spins.
- As there are no terms with a single spin (linear s_i terms), there are no qubit biases required.

The Hamiltonian thus simplifies into H_P, the *problem Hamiltonian* which will be programmed into the D-Wave hardware, as in Eq. (5.5):

$$H = X + H_P \tag{5.4}$$

$$H_P = 2n_1n_2(s_1s_2) + 2n_1n_3(s_1s_3) + \dots \tag{5.5}$$

$$\begin{aligned} X &= n_1^2 + n_2^2 + \cdots + n_N^2 \\ &= N \end{aligned} \tag{5.6}$$

Note that the squared spin terms have been removed from H_P, and introduced in X. As each term becomes $+1$, for N terms, X is simply equal to N. Throughout this thesis, constants similar to X will be known as the *expected energy* of the problem Hamiltonian. This is because a ground state ($H = 0$) solution is expected to have an energy value of $H_P = 0 - X$.

From Eq. (5.5), the qubit biases vector and coupling strengths matrix can be established:

$$h = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad J = \begin{bmatrix} 0 & 2n_1n_2 & \cdots & 2n_1n_N \\ 0 & 0 & & 2n_2n_N \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} \tag{5.7}$$

This assigns N qubits to each of the integers in the set S, giving each qubit numbered i a bias of $h_i = 0$. Furthermore, each qubit i is connected to every other qubit j, with a coupling strength between them of $J_{i,j} = 2n_in_j$ (double the product of the two integers they represent).

The h vector and J matrix can now be sent to the D-Wave hardware (or equivalent software solvers) using the Ising model (Eq. (2.5)). The solutions returned will seek to minimise H_P. Eq. (5.4) is used to calculate the energy value of the Lucas Hamiltonian H, by adding H_P and the expected energy X (from Eq. (5.6)) together.

It is obvious that when $H = 0$ or $H = 1$, the solution must represent a *perfect partition* (one in which $\Delta \leq 1$) – this answers the **NP**-Complete decision problem ("does a perfect partition exist?"). By reading the final spin value of each of the qubits and associating them with their specific integers in S, the two subsets S_1 and S_2 are found – thus solving the **NP**-Hard problem ("find a perfect partition").

Finally, if $H > 1$, then a perfect partition was not found. Either one does not exist, or the quantum annealing process was not able to find it. Without knowing the minimum energy gap g_{min} in order to set a sufficiently long annealing time (see Section 2.3.4), the problem Hamiltonian can not guarantee whether a perfect partition does or does not exist. Regardless, a solution with a positive energy value is the one found with the minimum difference between sums of the two subsets – thus solving the **NP**-Hard optimisation problem.

As H represents the square of the difference between the subset sums, Δ can be redefined as:

$$\Delta = \sqrt{H} \tag{5.8}$$

5.2.1 Removing Degeneracies

The solutions found by the equations proposed above have "degeneracies". These degeneracies refer to the presence of two equal and opposite configurations for every solution of the problem (by swapping the ± 1 spins of the two sets). This implies that a more efficient algorithm must exist that removes these degeneracies – such a modified algorithm is described below.

For example: given the set $S = \{2, 4, 9, 15\}$, each integer is assigned to an Ising spin such that s_1 represents the integer 2, s_2 represents 4, and so on. Clearly, the resulting partitions with minimum energy (both subsets having equal sum) occurs when the spins for s_1, s_2, s_3 are the same (implying they are in one subset), and s_4 is the opposite (and therefore in the other subset). This results in two possible configurations, being $s = [+1, +1, +1, -1]$ and $s = [-1, -1, -1, +1]$.

A simple solution to this problem is suggested by Lucas (2014) [45]: it is not important to know whether a given spin is in the $+1$ or -1 set, only that it matches the spins of the other numbers in its set. Thus, an arbitrary spin can be assumed to be $+1$, which fixes all other qubit spins relative to that one.

This can be done by modifying Eq. (5.5), introducing the constraint that an arbitrary spin is fixed, such as $s_1 = +1$:

$$H_P = 2n_1n_2(s_2) + 2n_1n_3(s_3) + \cdots + 2n_2n_3(s_2s_3) + 2n_2n_4(s_2s_4) + \ldots \tag{5.9}$$

This essentially replaces coupled terms, such as $2n_1n_2s_1s_2$, with a bias on the remaining qubit, such as $2n_1n_2s_2$ (as $s_1 = +1$). This has the effect of effectively removing one qubit from the computation, so that only $N-1$ spins in a fully-connected graph are now required. While this may seem trivial, embedding a fully-connected graph into the D-Wave's Chimera graph is hugely expensive (as seen previously in Section 2.4.4 and Section 4). Saving even a single fully-connected spin means a huge reduction in the number of hardware qubits required.[1]

5.3 Summary of the Algorithm

1. Associate every integer in the set $S = \{n_1, \ldots, n_N\}$ with its own Ising spin variable $\{s_1, \ldots, s_N\}$.
2. Compute the expected energy $X = N$ as per Eq. (5.6), where N is the size of S.
3. Remove one spin variable from the problem. This spin variable will not be represented as any qubits in the fully-connected hardware graph. Let this removed spin be s_1.
4. Build a fully-connected graph of the remaining spin variables (not including s_1).
5. Set the coupling strength between every spin variable (not including s_1) to be $J_{i,j} = 2 * n_i * n_j$. Note that this coupling term applies only once for every coupling (i.e. $J_{j,i}$ does not exist, as the J matrix (Eq. (5.7)) is considered to be upper-triangular).
6. Set the spin biases in the graph to $h_i = 2 * n_1 * n_i$. Note that this is equivalent to including the s_1 variable and all its couplings, and fixing it at a spin of $+1$.
7. Embed the problem into the D-Wave Two's Chimera graph.

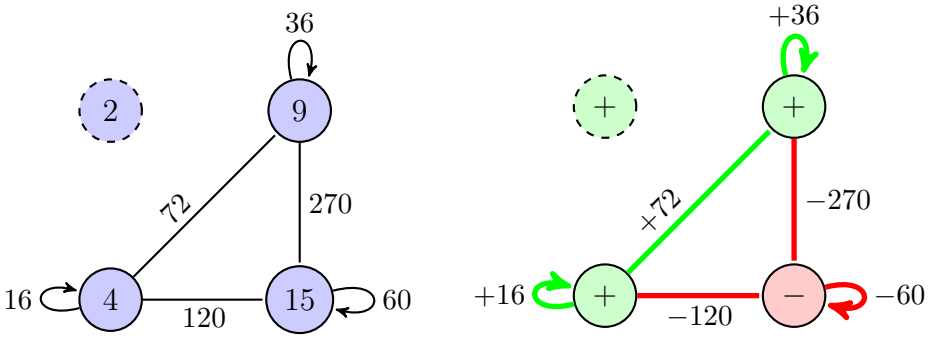
[1] As a simple example, 10 fully-connected spins may require 36 qubits in the D-Wave Two's Chimera graph, but only 26 qubits for 9 fully-connected spins. This difference grows at least with the square of the number of spins.

8. Send the embedded Ising problem on the D-Wave quantum processor. This will return the minimum energy configuration, with energy H_P.
9. Add the expected energy X to the minimum energy found H_P, to calculate the Hamiltonian energy $H = X + H_P$, as per Eq. (5.4).
10. If $\Delta = \sqrt{H} \leq 1$, then this configuration is a perfect partition. Otherwise, this configuration is the minimum difference between sums of the subsets that was found.

5.3.1 Worked Example

Let $S = \{2, 4, 9, 15\}$ be the set to be partitioned. By inspection, it is obvious that $S_1 = \{2, 4, 9\}$ and $S_2 = \{15\}$ is a perfect partition.

Each integer in S is first associated with its own spin variable $\{s_1, \ldots, s_N\}$. In order to remove degeneracies, one of these spins is fixed to $s_i = +1$, and the remaining spins are set up as a fully-connected graph. The biases and couplings between the spins are then computed as shown in Fig. 5.1a.



(a) Spin graph **(b)** Minimum Energy State

Figure 5.1 – Example graph and solution for problem set $S = \{2, 4, 9, 15\}$. Each circle denotes a qubit associated with that specific integer. The numbers looping to a qubit represent its bias, and between qubits their coupling strength. Dashed qubits are not modelled in the hardware, and shown only for their effect on the solution.

By running the adiabatic "annealing" process on the spin graph, the lowest energy solution is computed: $H_P = -326$ (as represented in Fig. 5.1b) whenever the spins on $S_1 = \{2, 4, 9\}$ are the same and $S_2 = \{15\}$ is the opposite. Since the spin on 2 was held constant as $s_1 = +1$, the spins on S_1 must all be $+1$, and S_2 must be -1.

The energy value arises from the Ising spin equation (Eq. (2.5)). Whenever two neighbouring qubits have the same spin, the coupling between them is added,

but when their spins are different, the coupling is subtracted. Their bias is added for any qubits with value $+1$, and subtracted for qubits with -1. Thus, $H_P = (36 + 16 + 72) - (60 + 120 + 270) = -326$.

Next, the expected energy (Eq. (5.6)) is computed to find that $X = 2^2 + 4^2 + 9^2 + 15^2 = 326$. The energy of the Lucas Hamiltonian is found from Eq. (5.4), giving $H = X + H_P = 326 - 326 = 0$. This means that the difference between the sums of the two subsets is $\Delta = \sqrt{H} = 0$, and thus, that it is a perfect partition.

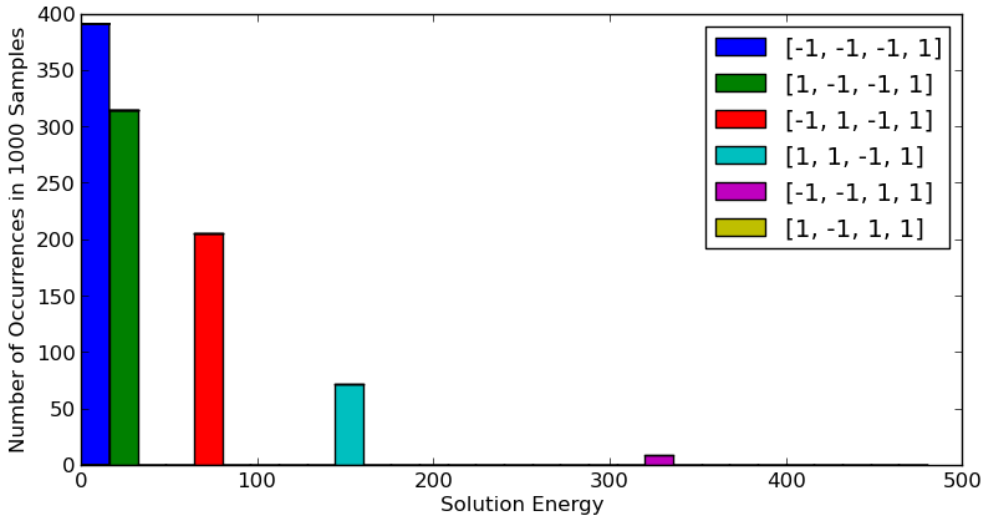


Figure 5.2 – A plot of every minimum solution found by the D-Wave hardware over 1000 runs of the `Number Partitioning Problem` example. Each colour represents a unique configuration of qubit spins. In each case, $s_4 = +1$ is held constant.

When this problem is sent to the D-Wave quantum hardware, the solutions returned are often different due to the various quantum effects described previously, such as due to thermal noise. By posing the same problem to the quantum hardware multiple times, a frequency distribution of possible solutions is returned, an example of which is seen in Fig. 5.2. Notice that the eigenstates of the problem Hamiltonian are clearly visible – there are no solutions found between the ones shown, because they simply cannot exist (for the specific problem given). That a selection of low-energy solutions is returned for this and other problems is considered an advantage of the D-Wave hardware over conventional solvers, as also determined by Perdomo-Ortiz et al. [56] and Rieffel et al. [62].

5.4 Phase Transition

Problem sets for the `Number Partitioning Problem` are characterised by an easy-hard phase transition as described in Gent and Walsh (1998) [31]. Gent and Walsh showed that the number of *perfect partitions* is highly dependent on the total size N of the set S, as well as the range of the numbers in the set.

Consider a problem set S with N numbers drawn uniformly at random from the range $[1, l]$, where l is the maximum possible value of any number in the set. Gent and Walsh defined a constrainedness parameter K, such that lower values imply a constraint due to the size N of the set, and higher values imply a constraint due to the maximum values l of the numbers in the set (or the *precision* required to represent them):

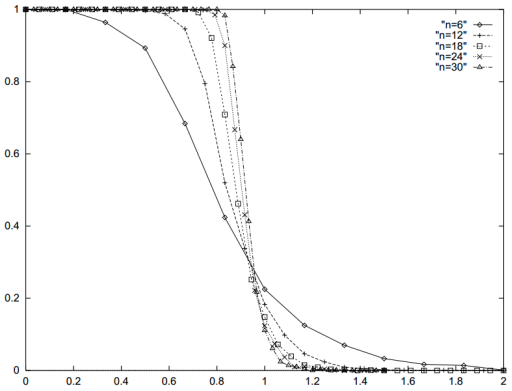
$$K = \frac{\log_2 l}{N} \tag{5.10}$$

Gent and Walsh (1998) showed that the probability of a perfect partition existing in any randomly generated problem of a given size (picking N numbers from the range $[1, l]$) is highly dependent on the constrainedness parameter K, with a sharp drop-off occurring at $K \approx 0.96$, as shown in Fig. 5.3a on the following page for different set sizes N. They determined this to be a *phase transition* in the `Number Partitioning Problem`, in that it divides problems into two distinct types: easy (many perfect partitions likely to exist) and hard (zero, one, or very few perfect partitions).

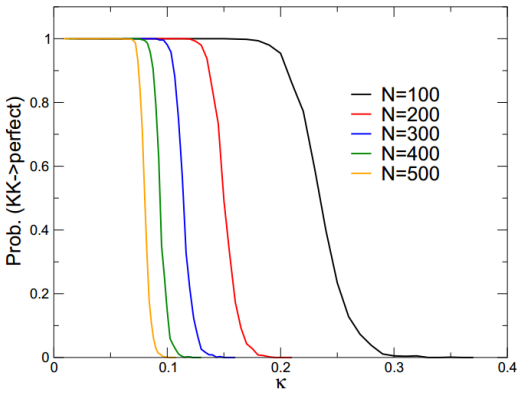
Mertens (2006) [48] later showed that heuristic algorithms used for solving the `Number Partitioning Problem` also contain this phase transition, although they found the point at which the phase transition occurs, K_C, is dependent on the size of the set N being solved, as in Fig. 5.3b.

In Fig. 5.3c, the same behaviour is demonstrated by running the described algorithm on the D-Wave Two. Here, the set size N is varied from 6 to 14, and the parameter K from 0 to 2.[2] Each point represents the mean probability of finding a perfect ($\Delta \leq 1$) partition from a set of 10 randomly-generated sets. The probability itself is calculated as the number of times a perfect partition solution is found out of the total number of times the quantum annealer is run ($35,000$ times with an

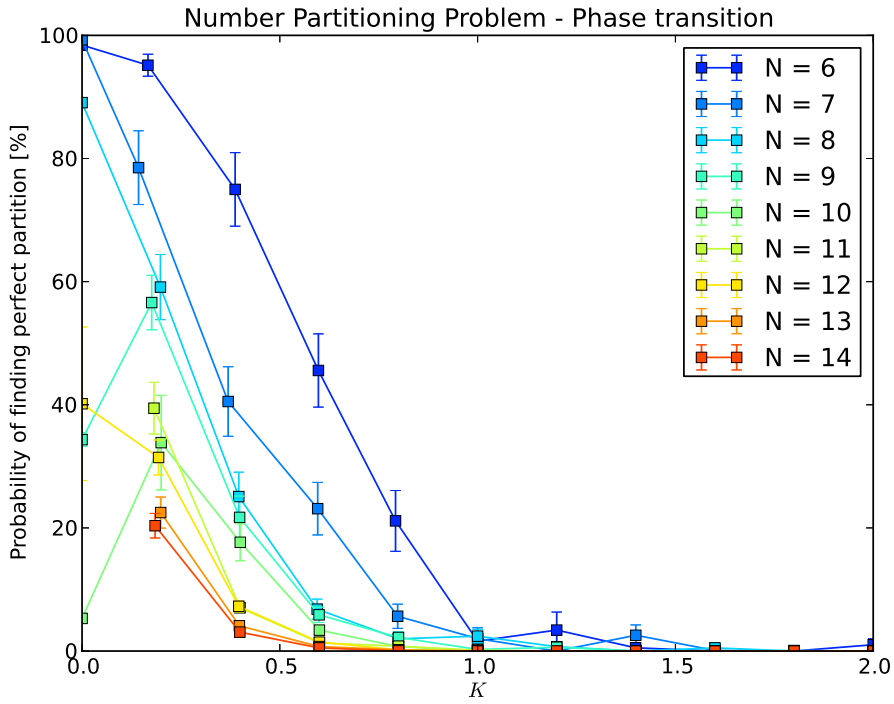
[2] Tests with up to $N = 30$ were possible, but are not shown as they quickly converged to 0%.



(a) Numerical phase transition [31]



(b) KK heuristic phase transition [48]



(c) Experimental phase transition on D-Wave Two

Figure 5.3 – Probability (y-axis) of finding a perfect partition in a randomly chosen `Number Partitioning Problem` set depending on its constrainedness parameter K (x-axis).

annealing time of 20μs). As such, each data point represents a minimum of $350,000$ anneals. The error bars represent the standard error of the observed mean.

On the D-Wave Two hardware, the observed phase transition occurs due to the constrainedness K of the problem, which manifests itself in two ways. Firstly, the maximum size l of numbers in the set directly impact the strengths of the couplings and biases in H_P. As the D-Wave Two hardware is limited in precision

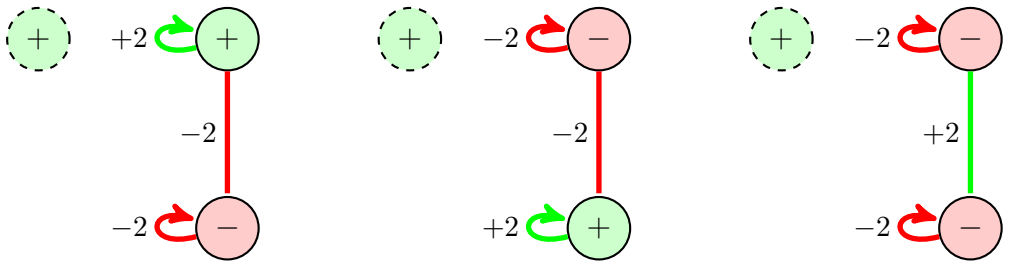
(for representing couplings and biases), higher values of l mean much higher precision requirements. Secondly, the algorithm requires a fully-connected spin graph with $N-1$ spins, but it is expensive to embed such a graph into the D-Wave Two Chimera graph (see Section 2.4.4).

The behaviour of the algorithm as it crosses the phase transition is an important one to understand, as it effectively defines what problems are easy or hard to solve. The scaling of the probability of finding a perfect partition (Fig. 5.3c) is further investigated in Section 5.5.

5.4.1 Undefined States at $K = 0$

It is worth noting the unexpected behaviour of the quantum annealer for the case where $K = 0$. This case represents problem sets constrained to a maximum value of $l = 1$, that is, each set simply contains exactly N ones, and thus must have a perfect partition. For this reason, any solver should be expected to have a 100% chance of finding a perfect partition. This is not the case with the quantum annealer, however, in which the probability is much lower and sometimes lower than the next case. The most likely explanation for this is as follows.[3]

For the case where $N = 3$ and $l = 1$ (and thus $K = 0$), any randomly-generated set will exactly equal $S = \{1, 1, 1\}$. A perfect partition would be $S_1 = \{1, 1\}$ and $S_2 = \{1\}$. By applying the algorithm described previously, a problem graph is generated containing two qubits of bias $h = 2$ each and a coupling of $J = 2$ between them. Such a graph has exactly three cases for the lowest energy configuration, as shown in Fig. 5.4.



(a) $\Delta = 1$ **(b)** $\Delta = 1$ **(c)** $\Delta = 1$

Figure 5.4 – Lowest energy configurations for the `Number Partitioning Problem` when $N = 3$ and $l = 1$ ($K = 0$).

[3]Note that issues with the experimental D-Wave API meant that some values of N could not be programmed into the machines.

While one would expect the D-Wave Two to return all three configurations with approximately the same probability, in reality it occasionally returns what appears to be an undocumented *undefined state*.[4] Including the probability of finding this undefined state (if and when it appears) raises the probabilities at $K = 0$ much higher towards 100%. However, it is not valid to include the undefined state in the results, as it does not actually define what a perfect partition would look like, and thus is relatively useless as a solution for $K = 0$.

5.5 Problem Scalability

As is the case for known existing classical heuristics (Fig. 5.3b), the phase transition of the problem when solved on a quantum annealer is highly dependent on the size N of the set being solved. Determining the exact relationship is difficult as the point of the phase transition is not well-defined in each case. If the phase transition is assumed to occur at the point of 50% probability, and the probability for $K = 0$ is assumed to be 100% (see previous section), then a value K_C representing the phase transition point for a set of size N can be found by interpolating for the point at 50% probability. This is plotted in Fig. 5.5.

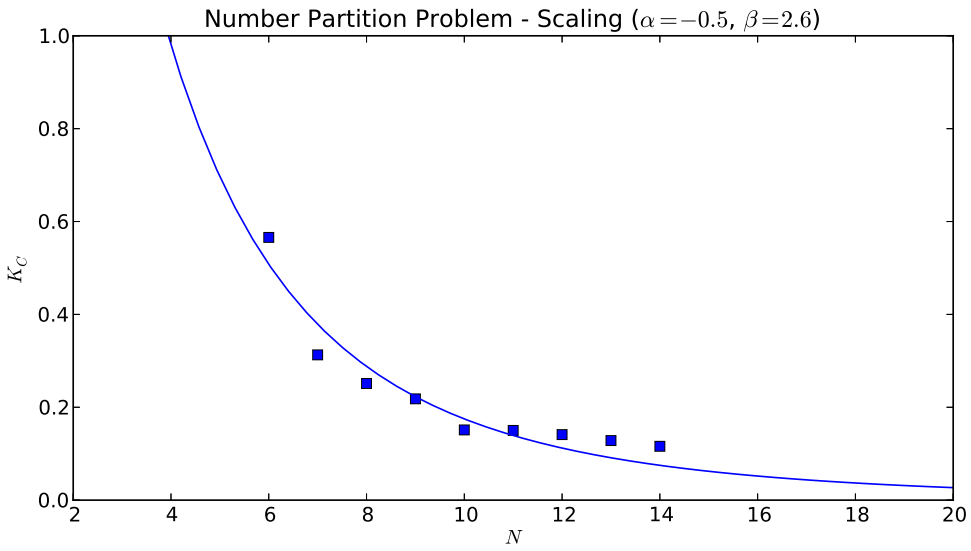


Figure 5.5 – Phase transition value K_C for each set size N. A trend line is plotted as $K_C = \beta N^{\alpha \log N}$.

[4]Such an undefined state appears to occur only when using D-Wave's EmbeddingSolver to embed the problems. It is undocumented in the sense that D-Wave's API does not mention such a feature (or bug). The undefined state is not observed when not using the EmbeddingSolver (see Section 10).

A general trend is observed following $K_C = \beta N^{\alpha \log N}$ for $\alpha = -0.5$ and $\beta = 2.6$, noting again that the interpolation to find K_C makes imperfect assumptions. This appears to be similar to a trend observed in the best performing classical heuristic, of order $O(N^{\alpha \log N})$,[48] although it is unclear whether that refers to similar characteristics. Nevertheless, the observation of an exponential trend such as this may prove useful in further research in properly determining the scaling of similar problems.

Another method for observing the scaling of the problem is as follows. Each randomly generated problem of size N and some constrainedness K is sent to the quantum annealer, and a list of solutions is returned. If the difference between subset sums (Δ) for every optimal (best) solution returned is averaged, an exponential trend is seen, as in Fig. 5.6. Note that the mean optimal Δ is exponential with K, and also dependent on a factor of N.

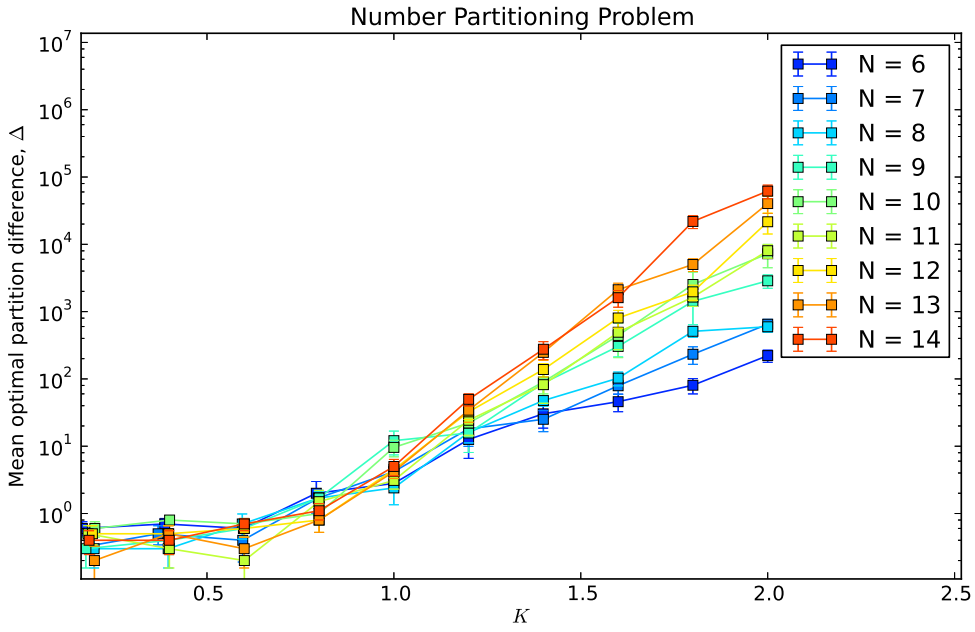


Figure 5.6 – Average (mean) difference between subset sums (Δ) of the best solutions found for all tested values of N and K.

5.6 Performance Comparison

The output of the D-Wave machine can also be compared with those found by the best-performing heuristic algorithm. The Karmarkar-Karp (KK) differencing algorithm, a polynomial-time approximation algorithm, deterministically finds a single solution for any given set. In Fig. 5.7 on the following page, the mean Δ

found by the KK algorithm is shown (for 1,000 randomly generated sets per data point).

By comparison, the *weighted average* Δ_{mean} found by the D-Wave Two (for 10 randomly generated sets, at 35,000 anneals) is also shown. This weighted average is calculated by determining the number of times each solution occurred out of 35,000 anneals, multiplying this by the Δ of that solution, and averaging all the solutions. This results in a Δ that is more representative of what would be found on any single anneal.

For example, if one run of 35,000 anneals results in solutions with $\Delta = \{0, 10, 300\}$ occurring $\{100, 4900, 30000\}$ times respectively, then the weighted average $\Delta_{mean} = (0 * 100 + 10 * 4900 + 300 * 30000)/35000 = 258$, effectively the average Δ one could expect from a single anneal. The Δ_{mean}s of all 10 randomly generated sets are then averaged together to determine one data point in Fig. 5.7.

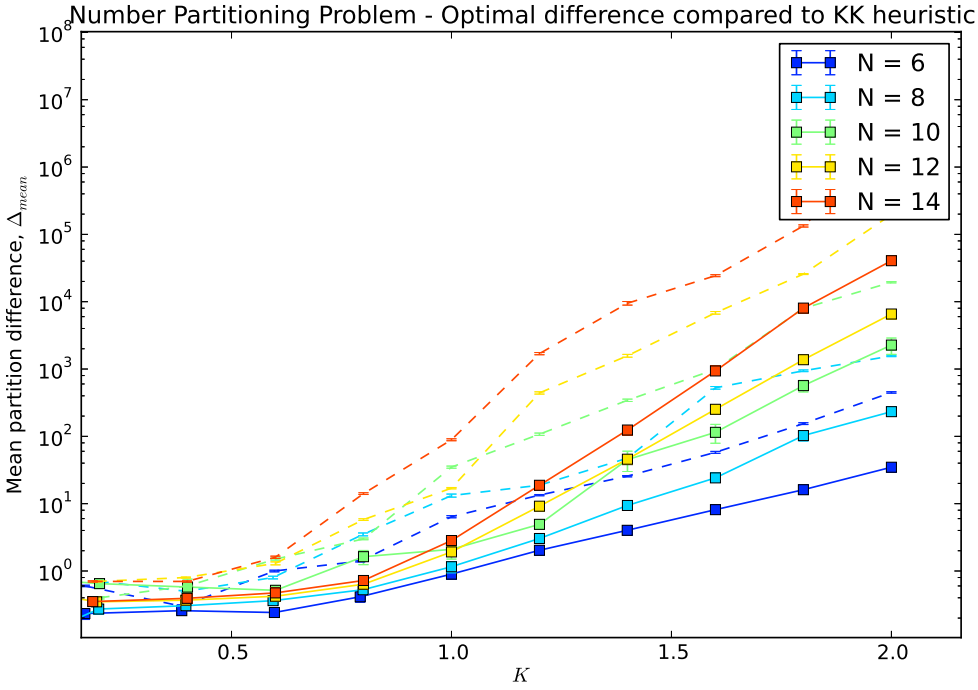


Figure 5.7 – Comparison of the average Δ found by the Karmarkar-Karp differencing algorithm (dashed lines) and weighted average Δ found by the quantum annealer (solid lines).

Perhaps the most interesting revelation is that the quantum annealer consistently outperforms (by finding, on average, partitions an order of magnitude better) the best-known heuristic algorithm in nearly every case. This occurs even when considering the weighted average Δ, which does not represent the average optimal

solutions found by the quantum annealer. This is particularly interesting when compared to the work done by D. S. Johnson et al. (1991) [37], who showed a Simulated Annealing solution to `Number Partitioning Problem` converging to solutions substantially worse than the KK algorithm.

It is also surprising that this is possible from the quantum algorithm being run with a constant $O(1)$ running time (that is, assuming the result is representative of a single anneal of 20μs). This is in contrast to the KK algorithm, which returns a solution taking much longer time ($O(N \log N)$).

However, before jumping to any conclusions, the results must first be considered carefully. While the KK algorithm is considered to be one of the best heuristics for the `Number Partitioning Problem`,[42] it is limited in that it finds only a single partition, and always the same one, for any given set. The reason for using the weighted average Δ for the D-Wave solutions is to try to level the playing field, by effectively considering what the expected Δ is from a single anneal on the D-Wave Two. It is possible that such a comparison is unreasonable.

Furthermore, it is not completely fair to say that the quantum algorithm runs in constant $O(1)$ time. It is almost certain that the annealing time necessary to maintain the performance shown increases significantly with N, though it is not greater than 20μs for the sizes tested. Tests with longer annealing times (of 1000μs and 2000μs) displayed almost exactly the same phase transition scaling with N (matching the trend in Fig. 5.5), and were thus omitted from these results. Properly determining the scaling of the annealing time and its performance against the KK algorithm for much larger test cases (as with future generations of D-Wave hardware) are potential areas for further research.

An extension of the KK algorithm, known as the Complete Karmarkar-Karp (CKK) algorithm,[42] is capable of continuously improving on the KK solution over an allowed time. CKK is thus guaranteed to find the optimal solution of any set, given exponential time in $O(2^N)$. In this sense, it is equivalent to the quantum algorithm presented in this thesis, which is also guaranteed to find the optimal solution if the annealing time is greater than required by the minimum energy gap g_{min}.

It is unfortunately quite difficult to do any sort of meaningful comparison between CKK and the presented quantum algorithm, as doing so would require comparing apples (first-generation quantum annealers) to oranges (a laptop CPU). A useful

comparison would involve a mathematical analysis of the scaling of both algorithms, however that lies beyond the capabilities of this author.

5.7 Conclusions

The quantum algorithm introduced in this thesis successfully solves the `Number Partitioning Problem` on the D-Wave Two quantum hardware. The main features of the algorithm include:

- It serves as a simultaneous solution to three variations of the same problem, being: the decision problem ("does a perfect partition exist?"), the partition problem ("find a perfect partition"), and the optimisation problem ("find the best possible partition").
- The algorithm appears to significantly outperform, at least for small test cases, the best-known heuristic algorithm. This is despite being run on first-generation (and very experimental) AQC hardware.
- It acts both as an optimisation algorithm, as well as an exact algorithm (guaranteed to find the most optimal solution, assuming a sufficiently long annealing time).
- The algorithm always runs in constant-time ($O(1)$). This of course assumes that the annealing time is sufficiently long to produce suitable solutions, a constraint that is likely to depend on the size of the problems (N).
- It demonstrates an application of a real **NP**-Hard problem on first-generation AQC hardware.

However, there are several issues with the approach taken:

- There is unexpected behaviour due to undefined states occurring in the easiest ($K = 0$) problems.
- Guaranteeing that a perfect partition, if it exists, is found, requires an understanding of the scaling of the annealing time of the problem.
- The problem maintains its exponential scaling characteristics, particularly with size of the input set, as well as due to the constrainedness parameter K.

- Problems are constrained both by the precision limitations in the D-Wave Two's couplings and biases, but also due to the requirement of embedding a fully-connected spin graph of $N - 1$ spins.
- Because of the Chimera graph restrictions, only problems of up to size $N \approx 30$ are capable of being solved on this generation of hardware. This is well below what classical algorithms can solve.
- Due to time available on the D-Wave machines, only 10 randomly generated sets were attempted per combination of N and K. This low number of instances may have impacted the perceived scaling of the problem.

At the very least, the results shown in this section prove that in-depth investigation of the algorithm presented is certainly worth pursuing in further research. Such research could focus on the following:

- Determining the scaling of the phase transitions, and comparing this to established algorithms. While an attempt was made to do this, the mathematical complexity lies beyond the scope of this thesis.
- Understanding the behaviour of the D-Wave Two when undefined states occur. It is possible that these states arise due to peculiarities of the D-Wave API, but this is difficult to test.
- A detailed analysis of the scaling properties of the algorithm and its performance when compared to the best-performing heuristics (both the KK and CKK algorithms).
- The arbitrary choice in picking a spin to fix (to remove degeneracies) may affect the overall precision of the problem. Fixing a spin converts couplings to equivalent biases, however couplings and biases have different precision limitations in the hardware, so perhaps making an arbitrary choice is not the best way.

Section 6
Algorithm: Graph Partitioning

The `Graph Partitioning Problem` is an **NP**-Hard problem that asks to find two subgraphs with equal number of vertices of a graph, such that the number of edges connecting the two subgraphs is minimised.

The `Graph Partitioning Problem` has use in divide-and-conquer algorithms as well as in applications such as Finite Element Analysis, in which it is more efficient to split a graph-like structure into parts in order to simplify the computations.[72]

This section describes an algorithm capable of solving the `Graph Partitioning Problem` on the D-Wave Two, as well as characterises its performance.

6.1 Lucas Formulation

The problem's Ising formulation, as given in Lucas (2014) [45] under Graph Partitioning, is first described. Consider a graph with an even number N of vertices and a set of edges E between them. An Ising spin variable $s_i = \pm 1$ is assigned to each vertex in the graph, representing which subgraph that vertex belongs to (the -1 subgraph or the $+1$ subgraph). The problem Hamiltonian is given as:

$$H = A\left(\sum_{i=1}^{N} s_i\right)^2 + B\sum_{(u,v)\in E}\frac{1 - s_u s_v}{2} \tag{6.1}$$

The first term of this Hamiltonian assigns an energy penalty for subgraphs with unequal numbers of vertices. If the sum of all spins is not zero, then there must be a different number of $+1$ spins than -1 spins. Squaring this term ensures that the minimum value of 0 occurs only when both sets have the same number of spins.

The second term in the Hamiltonian assigns an energy penalty for every edge connecting two qubits in different subgraphs. When two qubits have the same spin, their product $s_u s_v = +1$ and therefore this term equals 0. When two qubits have different spins, $s_u s_v = -1$, and so an energy penalty of 1 is applied.

The constants A and B are related in such a way that the penalty for having unequally sized subgraphs always outweighs the cost of extra edges between them. The relation required to maintain this constraint is given as below, where Δ is the maximum degree of the graph:

$$\frac{A}{B} \geq \frac{\min(2\Delta, N)}{8} \tag{6.2}$$

6.2 Mapping to D-Wave hardware

To convert this Hamiltonian to a problem representable on a D-Wave computer, Eq. (6.1) is first fully expanded, assuming a value of $B = 1$:

$$\begin{aligned} H &= A(s_1 + s_2 + \cdots + s_N)^2 + \left(\frac{1 - s_u s_v}{2} + \dots\right) \\ &= A(s_1^2 + s_2^2 + \cdots + s_N^2) + A(2s_1 s_2 + 2s_1 s_3 + \cdots + 2s_1 s_N + \dots) + \left(\frac{1 - s_u s_v}{2} + \dots\right) \end{aligned} \tag{6.3}$$

Note that the first term of Eq. (6.3) consists only of squared spin variables. As the square of an Ising spin variable $(s_i)^2 = +1$, and there are N spin variables, the first term is simply equivalent to $A * N$.

The second term in Eq. (6.3) comprises all edges between all vertices, not just those that are in E. This implies that a fully-connected graph of vertices is required. The second term itself can be simplified to:

$$A\left(2s_1 s_2 + 2s_1 s_3 + \cdots + 2s_1 s_N + \dots\right) = 2A \sum_{(u,v)} s_u s_v$$

The third term of Eq. (6.3) expands to:

$$\left(\frac{1 - s_u s_v}{2} + \dots\right) = \frac{1}{2} + \frac{1}{2} + \cdots - \frac{1}{2}\left(s_u s_v + \dots\right)$$

As this is simply a sum of all edges in the graph (that is, a sum of all $(u, v) \in E$), the $1/2$ terms must occur $|E|$ times, where $|E|$ is the number of edges in the graph. Thus, the third term can be further simplified to:

$$\left(\frac{1 - s_u s_v}{2} + \dots\right) = \frac{|E|}{2} - \frac{1}{2} \sum_{(u,v) \in E} s_u s_v$$

By definition, an edge (u, v) in the problem graph (for vertices u and v) are considered to be $(u, v) \in E$. Let all edges that are in the fully-connected graph of $|E|$ vertices, but not in the problem graph, be defined as being $\notin E$. The second and third terms of Eq. (6.3) can be combined, and the total Hamiltonian simplified to:

$$H = AN + \frac{|E|}{2} + 2A \sum_{(u,v) \notin E} s_u s_v + (2A - 1/2) \sum_{(u,v) \in E} s_u s_v \tag{6.4}$$

Finally, notice that every solution to the problem will have two equal but opposite configurations (either subgraph can be labeled $+1$ or -1, and it does not matter which is which). This is known as a degeneracy, and the general method for solving this was described for the algorithm in Section 5. By choosing an arbitrary spin variable to hold at a fixed value, such as $s_1 = +1$, the degeneracy can be removed by taking this fixed value into account:

$$\begin{aligned} H = N + \frac{|E|}{2} + 2A \sum_{(1,v)\notin E} s_v + (2A - 1/2) \sum_{(1,v)\in E} s_v \\ + 2A \sum_{\substack{(u,v)\notin E \\ u\neq 1 \\ v\neq 1}} s_u s_v + (2A - 1/2) \sum_{\substack{(u,v)\in E \\ u\neq 1 \\ v\neq 1}} s_u s_v \end{aligned} \tag{6.5}$$

The resulting Hamiltonian consists of a fully-connected graph of $|E| - 1$ spins, one for each vertex in the problem graph (excluding one fixed for degeneracies). Edges that exist in the original problem graph (the E set of edges) are given a coupling strength of $2A - 1/2$. The edges that exist in the fully connected graph but not in E are given a coupling strength of $2A$. Each spin (vertex) that was originally connected to the removed s_1 spin are given biases of $2A - 1/2$, and all other spins are given biases of $2A$.

Finally, the Hamiltonian is split into constants that have no effect on the eigenstates of the problem (the *expected energy*, X), and the problem Hamiltonian H_P:

$$H = X + H_P \tag{6.6}$$

$$X = N + \frac{|E|}{2} \tag{6.7}$$

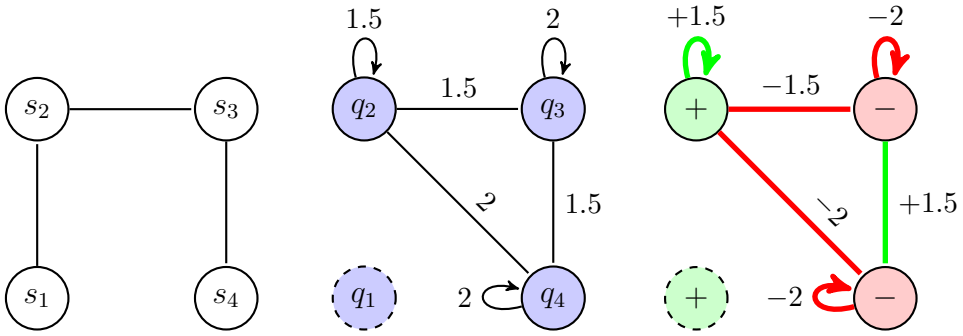
The problem Hamiltonian H_P is given to the hardware for minimisation. If a valid solution is found, the total energy H can be calculated. This energy is expected to be the number of connections between the two subsets, as per the original Hamiltonian in Eq. (6.1).

6.3 Summary of the Algorithm

1. Given a problem graph with even number of N vertices, and a set E of edges.

2. Compute a value for A, based on Eq. (6.2).
3. Associate every vertex in the problem graph with its own Ising spin variable $\{s_1, \ldots, s_N\}$.
4. Remove one spin variable from the problem. This spin variable will not be represented as a qubit in the fully-connected hardware graph. Let this removed spin be s_1.
5. Build a fully-connected graph of the remaining spin variables (i.e. not including s_1).
6. For every edge that existed in the problem graph (in E), set its corresponding coupling strength to 1.5. For all other edges (in the fully-connected graph), set the coupling strength to 2.
7. For all vertices originally connected to the vertex represented by s_1, set their corresponding bias to 1.5. For all other vertices, set the bias to 2.
8. Send the problem to the D-Wave quantum processor. This will return the minimum energy configuration, with energy H_P.
9. Compute the expected energy X using Eq. (6.7).
10. Add the expected energy to the minimum energy found, to get $H = X + H_P$, as per Eq. (6.6).
11. If the returned configuration was valid (equal number of vertices in each subset), then H will be the number of connections between the two subsets.

6.3.1 Worked Example



(a) Problem graph **(b)** Spin graph **(c)** Minimum Energy

Figure 6.1 – Example spin graph and solution for problem graph defined in (a). Note that dashed qubits are not modelled in the hardware, and are shown only for their effect on the spin graph and solution.

Let the problem graph be as shown in Fig. 6.1a on page 53. Clearly, the partition of two equally sized sets with minimum number of edges between them must be $\{s_1, s_2\}$ and $\{s_3, s_4\}$, with 1 edge between them. Note the properties of the graph: $N = 4$ vertices, $|E| = 3$ edges, and $\Delta = 1$. Thus, the constraint parameter $A \geq 2/8$. For simplicity, let $A = 1$.

When running the quantum annealing process on the configuration seen in Fig. 6.1b, the lowest energy level of $H = -4.5$ is obtained whenever the spins are the same for $\{q_1, q_2\}$ and the opposite spins for $\{q_3, q_4\}$. Thus, $H = (1.5+1.5) - (2+2+2+1.5) = -4.5$. A frequency distribution of all minimum solutions found over 1000 runs of this problem can be seen in Fig. 6.2.

The expected energy (Eq. (6.7)) gives $X = 4 + \frac{3}{2} = 5.5$. This in turn gives a total energy of $H = H_P + X = 1$. As this is a valid solution (both subsets are of equal size), this value of H implies that there is 1 connection between the two subsets, and that this was the minimum number of connections found for this problem graph.

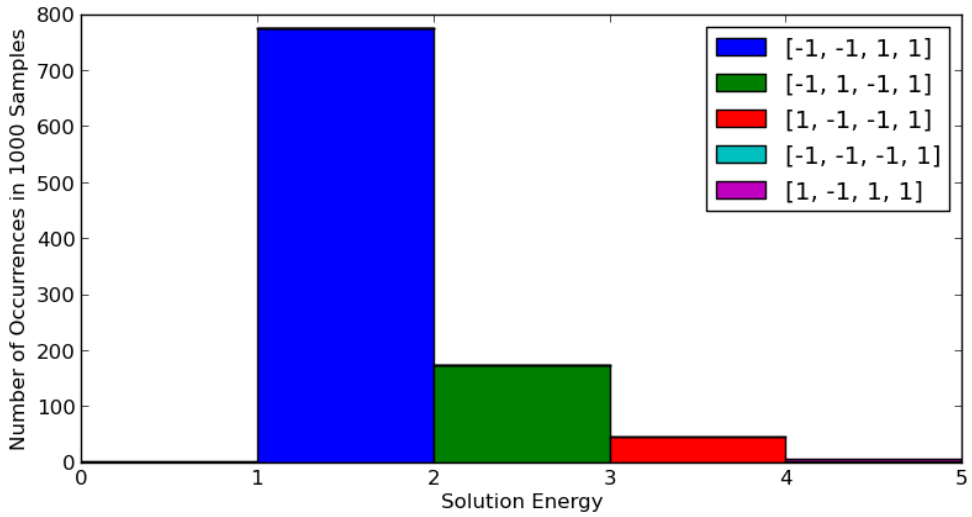


Figure 6.2 – A plot of every minimum solution found by the D-Wave hardware over 1000 runs of the `Graph Partitioning Problem` example. Each colour represents a unique configuration of qubit spins. In each case, $s_4 = +1$ is held constant.

6.4 Problem Scalability

In order to test the algorithm, the Erdős-Rényi model[77] was used to generate random graphs. Graphs were generated by specifying a graph size N and density of edges p, such that a graph $G(N, p)$ had approximately $\binom{N}{2}p$ total edges ($p = 0$ containing no edges, and $p = 1$ being a fully-connected graph).

The optimal graph partition was found by using "METIS", a well-regarded library for graph partitioning.[49, 72] The randomly-generated graphs would be solved both using METIS and the D-Wave Two. An example of a randomly-generated graph and the optimal partitions found by either algorithm, is shown in Fig. 6.3.

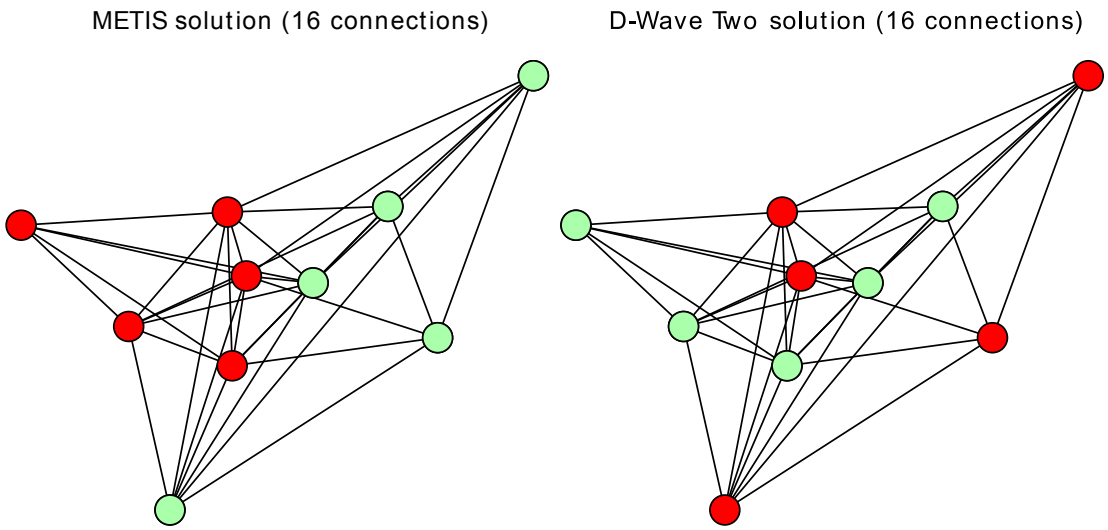


Figure 6.3 – Example of graph partitions found by METIS and the D-Wave Two algorithm.

Note that for this graph, both algorithms found a solution with the same number of connections. However, one advantage that the D-Wave Two offered was the list of solutions returned from the annealer. Whereas METIS would only return a single, deterministic partition, the D-Wave Two would return a series of partitions, ordered by optimality. This included the different graph seen in Fig. 6.4, as well as the solution found by METIS, and others. This capability allows for the most optimal solutions to be directly compared, in case one of the solutions is not applicable to the real-world for some reason. The benefits of this are also discussed by Rieffel et al. (2014) [62].

For testing, a range of values for p and N were selected. Each combination ran 20 randomly-generated graphs with an annealing time of 20μs over $35,000$ anneals. The average probability of finding the optimal partition was graphed in Fig. 6.4, where each data point thus consisted of $700,000$ anneals.

Note that there is a clear trend to solving high-density graphs. The reasoning for this is simple: as $p \to 1$, the graph approaches a fully-connected graph. By Eq. (6.5), this means less edges in the $\notin E$ set, and thus a higher proportion of edges and vertices with the same weightings. This has the direct effect of lessening

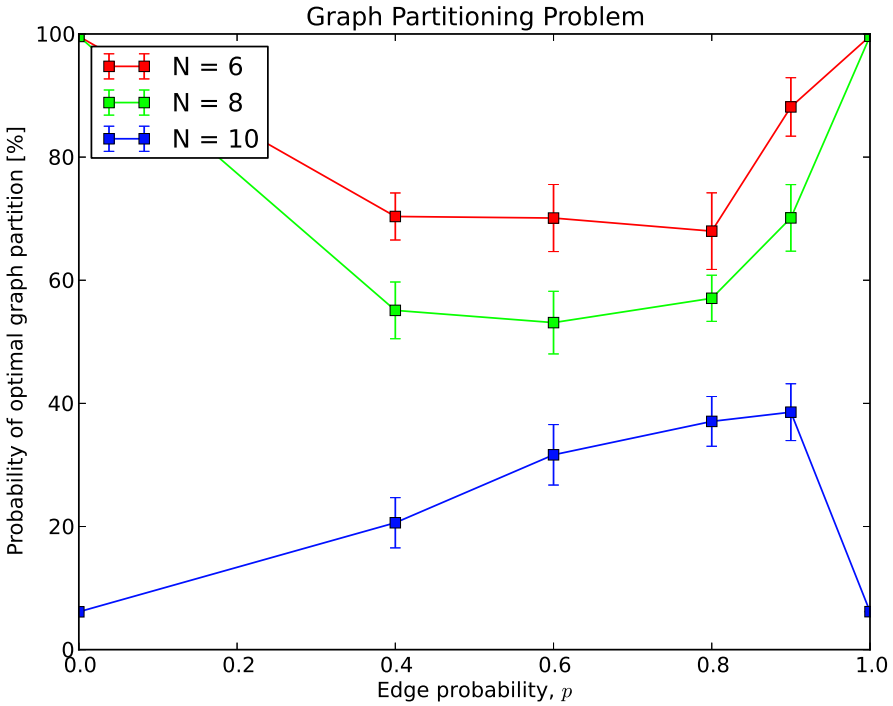


Figure 6.4 – Probability (y-axis) of finding the most optimal partition in randomly-generated graphs, depending on the density p of edges in the graph (x-axis). The error bars represent the standard error of the mean.

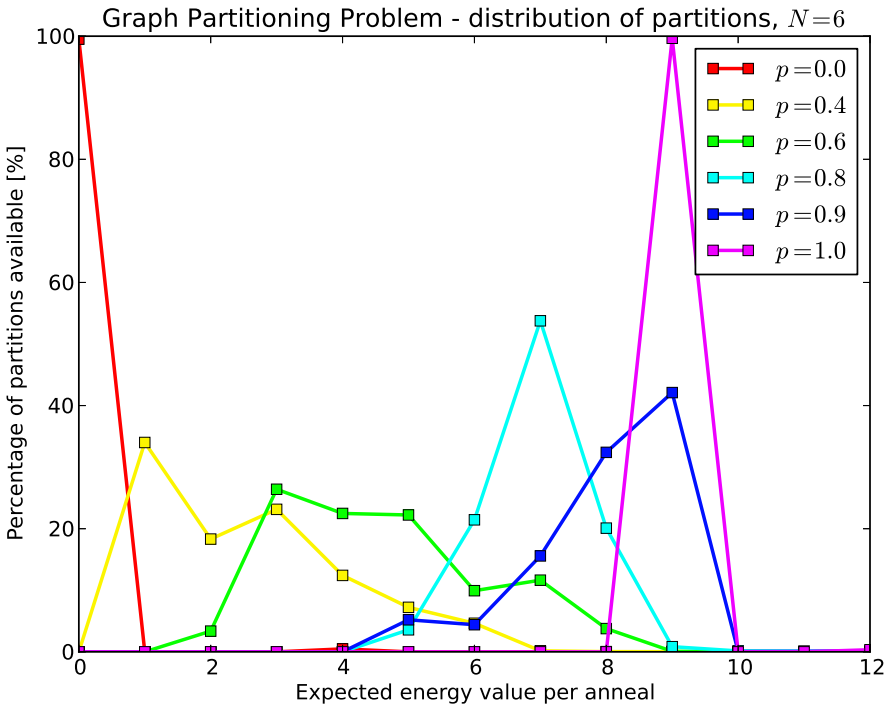


Figure 6.5 – The distribution of partition energy values is shown for multiple randomly-generated graphs of size $N = 6$ for any given edge density p. Note that the distribution narrows as $p \to 0$ or $p \to 1$.

the constraints on each qubit: when all weightings are equal, the qubits do not care which of them take a $+1$ spin and which take a -1 spin.

This effect is more clearly seen in Fig. 6.5. Here, the distributions of all partitions seen over 20 randomly-generate graphs of size $N = 6$ at each edge density p are shown. As $p \to 1$, less constraints are placed on the problem, making it easier to settle on the most optimal solutions. This is seen as a tightening of the distribution, such that when $p = 1$, almost 100% of partitions result in the same value (as it is very easy for them to get to it). The same effect is seen as $p \to 0$, which also removes constraints on the problem. For edge densities around $p = 0.5$, on the other hand, the distribution is much wider, implying that it is difficult to always settle on the most optimal solutions.

Notice that in Fig. 6.4, there is an unexpected decline in the probabilities at $p = 0$ and $p = 1$ for a problem of size $N = 10$. The reason for these is the presence of undefined states, as described previously in Section 5.4.1.

6.5 Performance Comparison

In order to evaluate the effectiveness of the algorithm, the average expected value per anneal is calculated. For each problem instance, a list of solutions is returned, each with its own respective energy value (representing the number of edges connecting the two sub-graphs of the problem). The list of energies is weighted by multiplying it with the number of times that solution appeared, and the sum of all the weighted energies is divided by the number of anneals that occurred. This gives an average energy value expected in a single anneal, which is averaged over all the randomly-generated graphs.

In Fig. 6.6, each data point represents the average expected energy of 20 randomly-generated graphs, as computed by the quantum hardware. The dashed lines represent the average optimal energy value of 100 randomly-generated graphs, as computed by the METIS graph partitioning solver.

The plot shows that the algorithm introduced in this section matches closely the computation of the most optimal partitions. However, this is only the case for small graph sizes N. For graphs of size $N = 10$ and edge density of $p = 0$ and $p = 1$ result in undefined states which highly skew the average expected energy. However, even for $p = 0.4$, the quantum algorithm does not scale similarly to the smaller graph

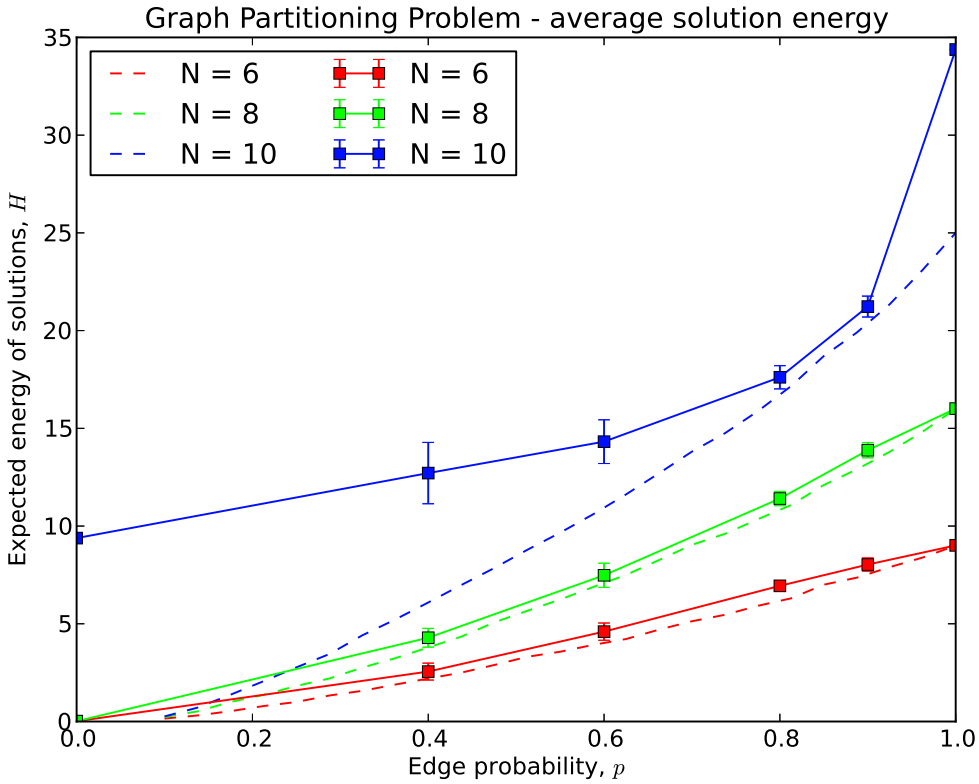


Figure 6.6 – The average expected energy per anneal for the quantum algorithm on the D-Wave Two (solid lines), against the average optimal energy value as computed by METIS (dashed lines).

sizes. It is possible that this is due to the short annealing time chosen (the minimum of 20μs), although this was not tested.

While the results are not particularly promising, they do verify that the algorithm behaves on par (in terms of scaling) with leading classical solvers such as METIS.

6.6 Conclusions

The quantum algorithm introduced in this thesis successfully solves the `Graph Partitioning Problem` on the D-Wave Two quantum hardware. However, the performance of the algorithm shows that it is only, at best, on par with classical solutions, for small problem sizes.

Despite this, the algorithm:

- Successfully solves an **NP**-Hard optimisation problem on the D-Wave Two hardware.

- Returns multiple solutions close to the most optimal, unlike deterministic classical solvers such as METIS.
- Displays good performance at the most extreme cases of edge density (such as when $p = 0$ or $p = 1$).

However, there are numerous issues with this approach, including:

- There is unexpected behaviour due to undefined states occurring for larger problem sizes.
- The algorithm requires a fully-connected spin graph of $N - 1$ spins. This is a similar issue faced by many problems of this type and has been discussed previously.
- The precision required in the couplings and biases depend on the constraint parameter A, which itself depends on the size N and edge density p of the graph. As such, this parameter must be considered carefully when analysing the scaling behaviour of the algorithm. In particular, it may be worth trading off some constrainedness (a lower value of A), which could result in slightly imbalanced partitions (one subgraph is bigger than the other) with potentially less connections between them. This is worthy of further research.

Section 7
Algorithm: Maximum Clique

Finding the `Maximum Clique` in a graph is an **NP**-Hard problem. A clique is defined as a subset of K vertices such that each vertex has a connection to every other vertex – in other words, that the graph formed of only K vertices is fully-connected. The **NP**-Complete problem asks whether a clique of size K exists in a given graph, and the **NP**-Hard optimisation problem is to find the largest clique in the graph.

The `Maximum Clique Problem` is considered one of the very hardest **NP**-Complete problems.[72] Classical solutions tend to exhaustively search through the subsets of vertices in the graph to find cliques, though heuristics utilising a simulated annealing approach are also available. The `Maximum Clique Problem` has applications in social networks, marketing, and cluster identification. An example given by Skiena (2009) [72] is that of identifying fraudulent tax documents by searching for cliques of very similar documents.

This section describes an algorithm for a directly-embeddable version of the **NP**-Complete Decision problem, and compares this to programmatic version of the **NP**-Hard Optimisation problem.

7.1 Lucas Formulation

7.1.1 Decision Version

The Decision version's QUBO formulation is shown below, as given in Lucas (2014) [45] under Cliques. Notice that, unlike the previous problems, the QUBO spin variables $x_i \in \{0, 1\}$ are used (rather than the Ising spin variables $s_i \in \{-1, +1\}$). Once again, every vertex in the problem graph is assigned its own spin variable, and all edges in the problem graph form the set E.

$$H = A\left(K - \sum_v x_v\right)^2 + B\left(\frac{K(K-1)}{2} - \sum_{(i,j)\in E} x_i x_j\right) \tag{7.1}$$

The first term of this Hamiltonian assigns an energy penalty > 0 for a solution with some number other than K vertices (and $= 0$ when the solution is a set of K variables). The second term is only $= 0$ when the number of edges in the solution graph is equal to the number of edges expected in a clique of size K (a clique is a fully-connected graph of size K, and thus has exactly $K(K-1)/2$ edges).

The A and B constants ensure that the first term is always greater than the magnitude of the second term, and thus that $H \geq 0$. Clearly, when both terms are zero (being the ground state of $H = 0$), the solution of all spin variables with value $x_v = 1$ defines a valid clique of size K. If $H > 0$, then no clique of size $\geq K$ exists.

7.1.2 Optimisation Version

The Optimisation version of the algorithm is substantially more complicated, as it requires the introduction of ancillary variables to track the size K of any cliques found. The Optimisation version was implemented through the use of symbolic programming (effectively programming the Hamiltonian directly into the Python code). For the sake of brevity, the details of the Hamiltonian are omitted, but can be found in Lucas (2014). The code for the Optimisation version, as well as for all problems explored in this thesis, is made available as an electronic supplement.

7.2 Mapping to D-Wave hardware

7.2.1 Decision Version

In order to map the problem on to the D-Wave hardware, Eq. (7.1) is expanded to get:

$$\begin{aligned} H &= A(K - x_1 - x_2 - \cdots - x_N)^2 + B\left(\frac{K^2 - K}{2} - \sum_{(i,j)\in E} x_i x_j\right) \\ &= A\left(K^2 + \sum_{i}^{N} x_i^2 + 2\sum_{i,j}^{N} x_i x_j - 2K\sum_{i}^{N} x_i\right) + B\left(\frac{K^2 - K}{2} - \sum_{(i,j)\in E} x_i x_j\right) \end{aligned} \tag{7.2}$$

In Eq. (7.2), $\sum_i^N x_i^2$ is equivalent to $\sum_i^N x_i$, as $x_i = \{0, 1\} = x_i^2$. Note also that $\sum_{i,j}^N x_i x_j$ is the sum of all edges of a fully connected graph on all spin variables, whereas $\sum_{(i,j)\in E} x_i x_j$ is the sum of all edges existing in the original problem graph (that is, all the problem graph edges form the set E). This will become clearer in the worked example.

Eq. (7.2) can thus be further condensed as follows:

$$H = X + A(1-2K)\sum_{u}^{N} x_u + 2A\sum_{u,v}^{N} x_u x_v - B\sum_{(u,v)\in E} x_u x_v \tag{7.3}$$

$$X = AK^2 + B\frac{K^2-K}{2} \tag{7.4}$$

Here, the constant terms are remove and re-named to X (the expected energy). To extract the biases and couplings required in the problem, the final Hamiltonian H is compared with the QUBO model (in Eq. (2.6)). By inspection, this shows that a bias of $A(1-2K)$ is applied to each spin variable, a coupling strength of $2A$ is given to every edge on the fully-connected graph, and actual edges in the problem graph are given a coupling strength of $2A - B$.

The constants A and B constrain the problem such that selecting more qubits than exist in the maximum clique does not cause the second term of Eq. (7.1) to decrease the energy of the Hamiltonian below that of the real solution. Remember, the ground state of $H = 0$ must only occur for a valid clique of size K, and must be $H > 0$ for anything else. The only requirement for A, as given by Lucas, is that:

$$A > KB$$

7.2.2 Optimisation Version

The Optimisation version of the problem is implemented as a symbolic Python expression equivalent to the Hamiltonian described in Lucas (2014) [45]. A series of ancillary spins are introduced to the problem to represent every possible value of K (from 2 to $\Delta + 1$). These spins, along with the spins representing each vertex in the graph, form a large fully-connected graph to be embedded.

The Hamiltonian expression is programmatically simplified through some assumptions, such as that $x_i^2 = x_i$ (the square of a QUBO variable, $x_i \in \{0, 1\}$, is simply equal to itself). The couplings and biases are extracted by fully expanding the expression, and simply keeping track of the coefficients attached to each variable term (that is, a coefficient for a 2-variable term is a coupling, but for a 1-variable term is a bias).

7.3 Summary of the Decision Algorithm

The algorithm consists of two parts: one to find whether a clique of size K exists, and another to find the maximum clique.

To find whether a clique of size K exists:

1. Associate each vertex in the problem graph with its own QUBO spin variable $\{x_1, \ldots, x_N\}$.
2. Build a fully-connected graph of the spin variables.
3. Choose a constant $A > KB$. $B = 1$ and $A = 2K$ can be chosen arbitrarily.
4. Apply a bias of $A(1 - 2K)$ on each spin variable.
5. For every edge in E (edges that exist in the problem graph), set the corresponding coupling strength to $2A - B$. For all other edges, set the coupling strength to $2A$.
6. Send the problem to the D-Wave quantum processor. This will return the minimum energy configuration, with energy $H - X$.
7. Compute the expected energy $X = AK^2 + B^{(K^2 - K)}/2$ (as per Eq. (7.4)).
8. Add the expected energy X to the solution energy returned, to get H.
9. If $H = 0$, then a valid clique of size K was found.

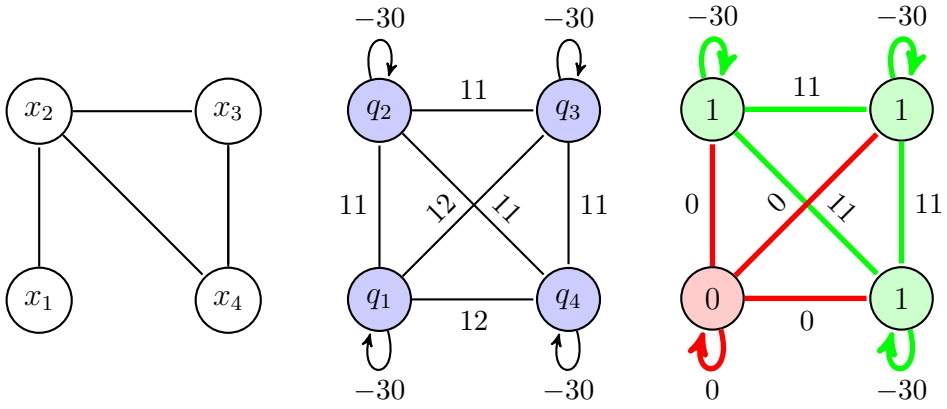
To find the maximum clique:

1. Compute the maximum degree Δ of the problem graph.
2. If the vertex with the maximum degree Δ is part of a clique along with all vertices connected to it, then this subset of vertices would form a clique of size $K + 1$. As such, no clique can be larger than this.
3. Starting at $K = \Delta + 1$, find whether a clique of size K exists in the graph.
4. If no such clique exists, choose a $2 \leq K \leq \Delta$ and find whether a clique of that size exists. For optimal performance, one can perform a binary search on this space (by trying K values halfway between the smallest value for which a clique exists, and the smallest value for which a clique does not exist).

7.3.1 Worked Example

Let the problem graph be as shown in Fig. 7.1a below. Clearly, the maximum clique is one of size $K = 3$, formed by the subset $\{x_1, x_2, x_3\}$. The maximum degree of

the graph is $\Delta = 3$, so the algorithm would begin by searching for a clique of size $K = 4$. As ones does not exist, the algorithm proceeds to search for a clique of size $K = 3$. By letting $A = 2K = 6$, the biases and couplings are set out as per the algorithm described above.



(a) Problem graph **(b)** Spin graph **(c)** Minimum Energy

Figure 7.1 – Example spin graph and solution for the `Maximum Clique Problem` graph defined in (a).

The lowest energy configuration seen in Fig. 7.1b is obtained from the quantum annealing process, with an energy $H - X = (-30 - 30 - 30) + (11 + 11 + 11) = -57$. The expected energy $X = 6 * 3^2 + {}^{3^2 - 3}/_2 = 57$. Thus, the total energy of the solution is $H = -57 + 57 = 0$ – thus, a clique of size $K = 3$ exists and is formed by the qubits of value $x_i = 1$.

7.4 Problem Scalability

The performance of the cliques algorithm introduced in this thesis can be evaluated in a similar manner as to that done for the `Graph Partitioning Problem` in Section 6.4. This begins with the generation of random Erdős-Rényi graphs[77], parametrised by the size of the graph N and edge density p. A set of 10 graphs are randomly-generated for each combination of N and p, running $35,000$ anneals at 20μs each. The probabilities of finding the maximum clique (using the algorithm presented for the Decision version of the problem) are plotted in Fig. 7.2 on the next page.

For the Optimisation version of the problem, the same process is repeated and plotted in Fig. 7.3. The first thing to notice is the significantly lower probabilities when using the Optimisation version. However, this does not necessarily imply performance worse than the Decision version, as will be described shortly.

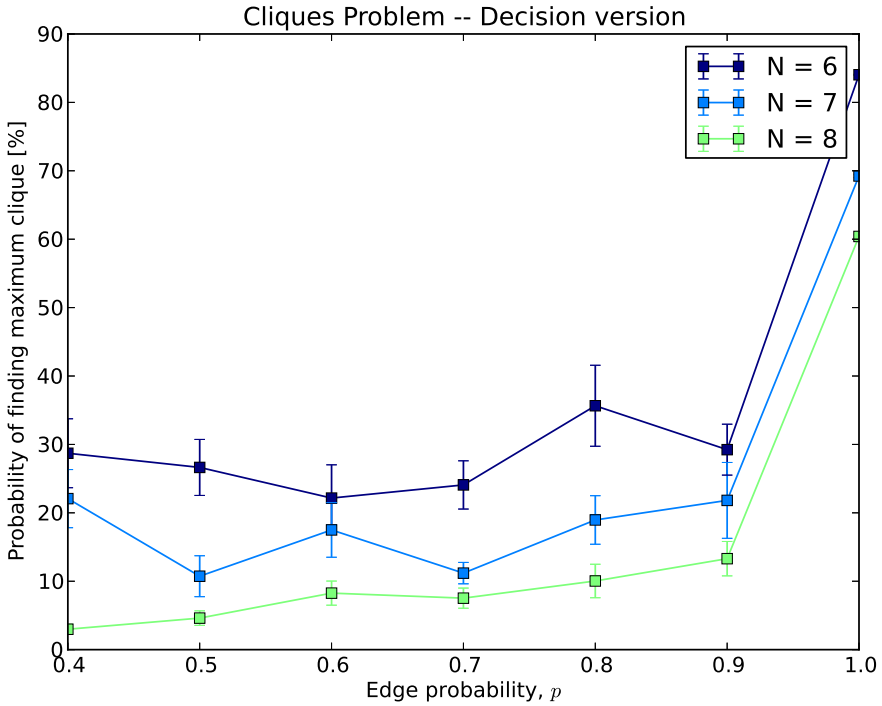


Figure 7.2 – Probability (y-axis) of finding the maximum clique in randomly-generated graphs, depending on the density p of edges in the graph (x-axis), for the Decision version of the problem. The error bars represent the standard error of the mean.

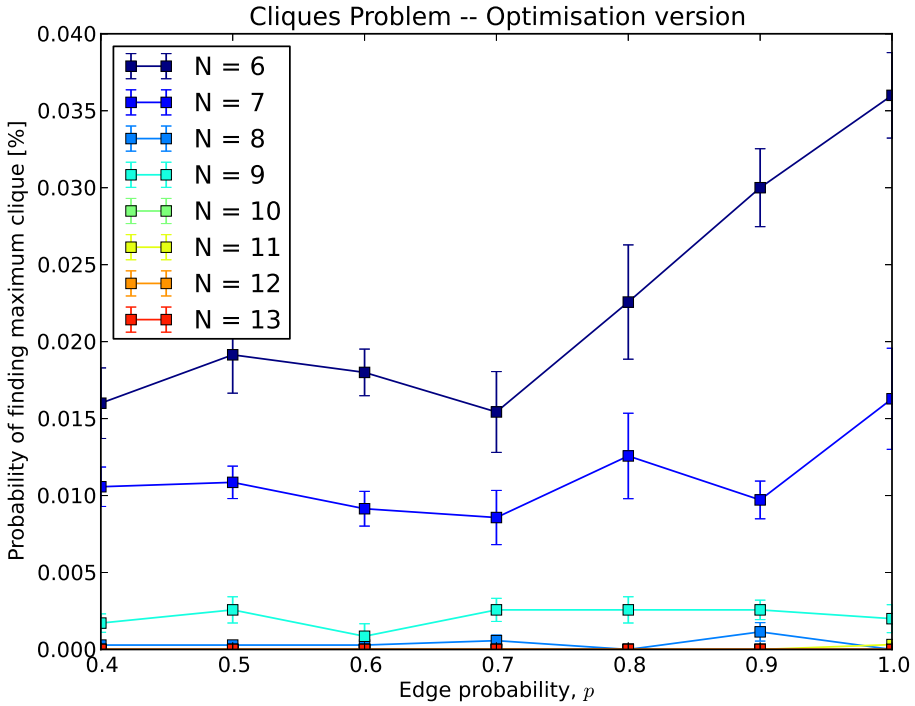


Figure 7.3 – Probability of finding the maximum clique, for the Optimisation version of the problem.

The main reason for this large discrepancy is due to the added ancillary bits in the Optimisation version. These bits, which represent the value of the biggest clique found, are effectively added as vertices to the problem graph, greatly increasing its size. As the graph is fully-connected, it is particularly expensive to embed into the D-Wave hardware. This effect can be seen in Fig. 7.4.

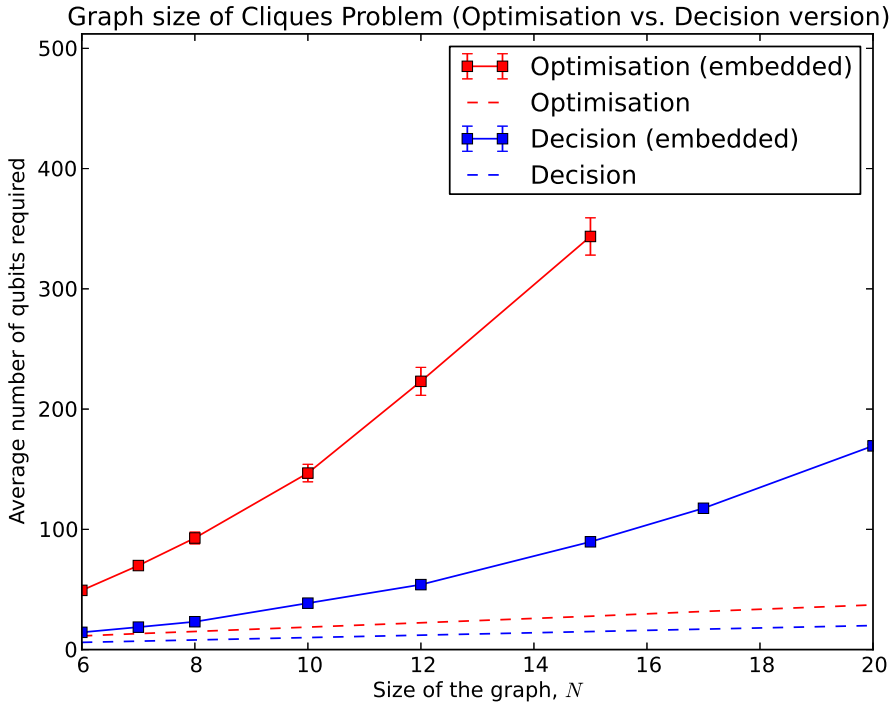


Figure 7.4 – A comparison of the average number of qubits required (solid lines) to embed the Decision and Optimisation versions of the `Maximum Clique Problem`. The number of qubits required for the problem before embedding are also shown (dashed lines).

As this shows, the Optimisation version's embedded graph grows so quickly that it can no longer be embedded at problem sizes above $N = 15$ (except on very rare cases when a particularly good embedding is found).

However, comparing the two versions fairly requires a different approach. The probabilities in Figs. 7.2 and 7.3 can be converted into an expected number of anneals to find the maximum clique. This is done by using Eq. (4.1) from Section 4.2.7, which computes the number of anneals k required at a given probability r, such that there is a 99% certainty of obtaining the optimal result:

$$k = \frac{\log(1 - 0.99)}{\log(1 - r)}$$

Furthermore, as the Decision version of the problem requires running the same problem multiple times with different values of K, the number of tries must be included in the comparison. This is done by assuming that k remains constant for any K test, and thus the total expected number of anneals is simply k multiplied by the number of Ks tried.[1]

The expected number of anneals to find the maximum clique is plotted for both the Decision and Optimisation versions in Fig. 7.5.

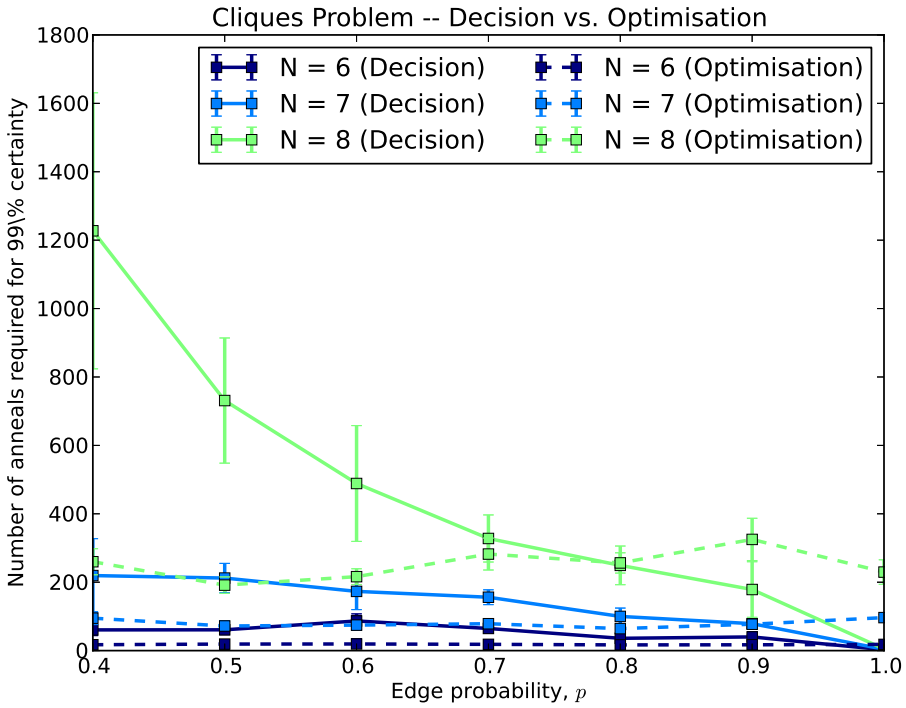


Figure 7.5 – A comparison of the running time (number of anneals to find the maximum clique, on the y-axis) expected for the Optimisation (dashed) and Decision (solid) versions of the problem.

This shows a couple of interesting things. Firstly, that the Decision version is highly efficient at finding the maximum clique of a graph when the graph is a clique itself (when $p = 1$). However, its performance degrades quickly for sparser graphs, as they are more likely to contain smaller cliques, requiring multiple tries of higher K values.

Secondly, that the Optimisation version, despite being inefficient on fully-connected $p = 1$ graphs, is unexpectedly equally capable of finding cliques in graphs of any

[1] To clarify the similarly named variables, the Decision problem asks whether a clique of size K exists, and if it does, it is expected to be found (with 99% certainty) within k number of anneals.

edge density within the same amount of time. This is especially useful as, for larger N and sparser graphs ($p \to 0$), the constant time required far outperforms the Decision version.

That the Optimisation version can directly compete with the Decision version is completely unexpected, particularly due to the significant difference in embedded graph sizes (as per Fig. 7.4). This result proves the value of utilising the quantum annealer for what it does best: solving optimisation problems, rather than solving exact problems.

7.5 Conclusions

This section introduced a quantum algorithm for solving the Decision version of the `Maximum Clique Problem` on the D-Wave Two quantum hardware. The Optimisation version of the problem, while not explicitly described in this section, was also implemented and compared against the performance of the Decision version. The results show that:

- The D-Wave Two hardware is certainly more suited to solving optimisation problems than exact problems, even when the exact problem seems to be easier to implement in the hardware.
- Despite probabilities of finding the maximum clique being well below 1%, the Optimisation version is still able to maintain competitive (against the Decision version) total running times.
- The Decision algorithm tends to be most efficient when it is least constrained by the problem graph at $p = 1$.

Some issues with the approach shown here could lead to potential future research:

- Only small graph sizes were tested, as the performance of both versions degraded quickly with increasing size. The true scaling of the problem should be investigated, particularly with larger problem sizes.
- The effects of the constraint parameter A were not investigated, and A was instead chosen arbitrarily. Both the Decision and Optimisation versions have constraint parameters that can be tweaked.
- The Optimisation version was implemented programmatically. Developing it into a directly-embeddable algorithm, as was done with the Decision version, may be useful in better understanding its properties.

Section 8
Algorithm: Integer Factorisation

`Integer Factorisation` is the problem of finding the factors of a given integer. `Integer Factorisation` is unlikely, although unproven, to be in either **NP**-Complete or **P**, but is proven to be in both **NP** and **BQP**. The implications of this are that no efficient solution exists in classical computers, but do exist on quantum computers, in the form of Shor's algorithm[70].

Due to its complexity and the D-Wave Two's limitations, Shor's algorithm can not easily be converted into an adiabatic quantum algorithm. Instead, adiabatic approaches are described by both Peng et al. (2008) [55] and Xu et al. (2012) [78], who demonstrated the factorisation of the numbers 21 and 143 respectively – the highest integers ever factorised on quantum computers of any kind.[1] For this reason, the algorithm described here is likely an **NP**-Hard algorithm, unlike Shor's **BQP** solution.

`Integer Factorisation` has major applications in cryptography, and is considered one of the most significant future applications of quantum computing.[26] However, it is important to note that factorisation is *not* an optimisation problem, and is therefore likely not suitable for computation on an adiabatic quantum computer, as was seen previously in Section 7.5. It is likely that any algorithm for factorisation would scale poorly with the size of the factors being found.

In this section, the naive adiabatic approach introduced by Peng et al. (2008) is implemented for solving on the D-Wave Two. The approach described here combines the work of Peng et al. and an unrelated paper by Bian et al. (2013) [10] to demonstrate a working implementation on the D-Wave Two hardware.

8.1 Peng et al. Formulation

Let N be an integer with two factors p and q. The factors of N can be found by solving for p and q given $N = pq$. This equation can be converted to an energy function (a Hamiltonian) by re-arranging and squaring the equation, resulting in:

$$H = (N - pq)^2 \tag{8.1}$$

[1] As of July 2014.

Clearly, the minimum energy of $H = 0$ occurs when $N = pq$, and $H > 0$ when $N \neq pq$. Thus, H is a suitable energy function for the problem.

8.1.1 Sizing the Factors p and q

To begin the process, the p and q factors must somehow be represented in qubits on the quantum hardware. The simplest way of doing this is to use the binary representations of p and q,[55] by assigning each bit with its own QUBO spin variable $x_i \in \{0, 1\}$. For a 2-bit p factor, and 3-bit q factor, for example:

$$\begin{aligned} p_2 &= 2^1 x_{p_1} + 1 \\ q_3 &= 2^2 x_{q_2} + 2^1 x_{q_1} + 1 \end{aligned} \tag{8.2}$$

where x_{q_2} is the QUBO spin variable representing the most significant bit of q. Note that both factors are assumed to be odd, which implies that N is also odd. This assumption fixes the lowest two bits to 1, removing two spin variables (and therefore two qubits) from the computation, and is done without loss of generality (if N were even, one could simply halve it until it became odd).[55]

For a factor size of n_p bits for p and n_q bits for q, the maximum factors that can be calculated are $\max(p_{n_p}) = 2^{n_p} - 1$ and $\max(q_{n_q}) = 2^{n_q} - 1$. For the example above, the maximum factors are $\max(p_2) = 3$ and $\max(q_3) = 7$, and therefore the maximum number that can be factorised is $N = 21 = 3 * 7$. Note, however, that the number $N = 19 = 1 * 19$ can not be factorised, as one of its factors is too big to fit in either p_2 or q_3.

From this, the total number of qubits necessary to represent the factors is simply $n_p + n_q - 2$ (removing the two least-significant bits as N is odd).

If one factor of N is assumed to be $p \leq \sqrt{N}$, and the other factor assumed to be $q \leq \dfrac{N}{3}$ (i.e. assuming N is not prime), then Peng et al. (2008) [55] define the number of bits necessary to factor N as:

$$\begin{aligned} n_p &= \text{bits}\left(\left\lfloor \sqrt{N} \right\rfloor_{\text{odd}}\right) \\ n_q &= \text{bits}\left(\left\lfloor \frac{N}{3} \right\rfloor\right) \end{aligned} \tag{8.3}$$

where $\text{bits}(x)$ is the number of bits necessary to represent x, $\lfloor x \rfloor$ is the largest integer less than x, and $\lfloor x \rfloor_{\text{odd}}$ is the largest odd integer less than x.

The energy function H from Eq. (8.1) can now be rewritten with the binary representations of p and q from Eq. (8.2):

$$\begin{aligned} H_{2,3} &= (N - p_2 q_3)^2 \\ &= (N - (2^1 x_{p_1} + 1) * (2^2 x_{q_2} + 2^1 x_{q_1} + 1))^2 \end{aligned} \tag{8.4}$$

The subscript of $H_{2,3}$ is introduced to signify that this is the energy function for solving for a 2-bit p factor and 3-bit q factor.

8.1.2 Computing the Problem Hamiltonian

In order to understand the interactions between qubits, Eq. (8.4) is expanded completely, which results in the unwieldy equation:

$$\begin{aligned} H_{2,3} = &-16Nx_{p_1}x_{q_2} - 8Nx_{p_1}x_{q_1} - 4Nx_{p_1} - 8Nx_{q_2} - 4Nx_{q_1} + 64x_{p_1}^2 x_{q_2}^2 \\ &+ 64x_{p_1}^2 x_{q_2} x_{q_1} + 32x_{p_1}^2 x_{q_2} + 16x_{p_1}^2 x_{q_1}^2 + 16x_{p_1}^2 x_{q_1} + 4x_{p_1}^2 \\ &+ 64x_{p_1} x_{q_2}^2 + 64x_{p_1} x_{q_2} x_{q_1} + 32x_{p_1} x_{q_2} + 16x_{p_1} x_{q_1}^2 + 16x_{p_1} x_{q_1} \\ &+ 4x_{p_1} + 16x_{q_2}^2 + 16x_{q_2} x_{q_1} + 8x_{q_2} + 4x_{q_1}^2 + 4x_{q_1} + (N^2 - 2N + 1) \end{aligned} \tag{8.5}$$

Note that any QUBO spin variable $x_i \in \{0, 1\}$ has the property that $x_i^2 = x_i$. Making this substitution for all spin variables in Eq. (8.5) simplifies the equation considerably:

$$\begin{aligned} H_{2,3} = &\, 128x_{p_1}x_{q_2}x_{q_1} + (192 - 16N)x_{p_1}x_{q_2} + (64 - 8N)x_{p_1}x_{q_1} + 16x_{q_2}x_{q_1} \\ &+ (8 - 4N)x_{p_1} + (8 - 4N)x_{q_1} + (24 - 8N)x_{q_2} + (N^2 - 2N + 1) \end{aligned} \tag{8.6}$$

8.1.3 Reducing Multi-qubit Interactions

The energy function needs to be in the form of a QUBO equation, as in Eq. (2.6), in order to be programmed into the D-Wave computer. This requires all terms to have at most 2 spin variables interacting simultaneously. In Eq. (8.6), for example, the term $128x_{p_1}x_{q_2}x_{q_1}$ requires a 3-spin interaction, which is not possible in the QUBO model. All multi-spin interactions must be reduced until the equation is entirely in QUBO form.

An approach taken in Bian et al. (2013) [10] describes their method of reducing interactions by introducing an ancillary variable $y_{i,j} \in \{0, 1\}$ to represent the interaction between two spins x_i and x_j. In order to do this, an extra "penalty" term must be added to the energy function, to ensure that the spin variable $y_{i,j}$ takes the

correct value of $x_i x_j$. This penalty is given as:

$$P(x_i x_j \to y_{i,j}) = x_i x_j - 2x_i y_{i,j} - 2x_j y_{i,j} + 3y_{i,j} \tag{8.7}$$

It is obvious that $P(x_i x_j \to y_{i,j}) = 0$ when $y_{i,j} = x_i x_j$, and $P(x_i x_j \to y_{i,j}) > 0$ when $y_{i,j} \neq x_i x_j$.

Any multi-qubit interaction such as $Cx_i x_j x_k$ can therefore be reduced by introducing $y_{i,j}$ and replacing the term with $Cy_{i,j}x_k + \mu P(x_i x_j \to y_{i,j})$, where μ is a scaling factor that ensures the penalty is large enough to enforce the $y_{i,j}$ value. Generally, any $\mu > C$ will be sufficient, so $\mu = 2C$ is chosen arbitrarily.

Going back to Eq. (8.6), the $128x_{p_1}x_{q_2}x_{q_1}$ term can be reduced by replacing it with: $128y_{p_1,q_2}x_{q_1} + 256P(x_{p_1}x_{q_2} \to y_{p_1,q_2})$. Expanded out, this finally gives the Hamiltonian in QUBO form:

$$\begin{aligned} H_{2,3} = {} & (128y_{p_1,q_2}x_{q_1} + 256x_{p_1}x_{q_2} - 512y_{p_1,q_2}x_{p_1} - 512y_{p_1,q_2}x_{q_2} + 768y_{p_1,q_2}) \\ & + (192 - 16N)x_{p_1}x_{q_2} + (64 - 8N)x_{p_1}x_{q_1} + 16x_{q_2}x_{q_1} \\ & + (8 - 4N)x_{p_1} + (8 - 4N)x_{q_1} + (24 - 8N)x_{q_2} + (N^2 - 2N + 1) \\ \\ = {} & 128x_{q_1}y_{p_1,q_2} + (64 - 8N)x_{p_1}x_{q_1} + (448 - 16N)x_{p_1}x_{q_2} \\ & - 512x_{p_1}y_{p_1,q_2} + 16x_{q_2}x_{q_1} - 512x_{q_2}y_{p_1,q_2} + (8 - 4N)x_{p_1} \\ & + (8 - 4N)x_{q_1} + (24 - 8N)x_{q_2} + 768y_{p_1,q_2} + (N^2 - 2N + 1) \end{aligned} \tag{8.8}$$

Note that the constant terms can be removed to become the expected energy X, as in previous algorithms:

$$X = N^2 - 2N + 1 \tag{8.9}$$

Separate factorisation Hamiltonians must be generated for different sizes of p and q (i.e. different n_p and n_q). However, once a factorisation Hamiltonian is generated for given factor sizes, it can be stored in its QUBO form for future use. The generation is effectively a once-off process.

Each spin variable x_i and ancillary variable y_i in the factorisation Hamiltonian (Eq. (8.8)) is assigned to its own qubit. The coupling strengths and biases are simply the coefficients of the 2-spin interactions and 1-spin terms, and N is replaced by the number to be factorised.

8.1.4 Summary of the Algorithm

1. Let N be an odd integer to be factorised into p and q.
2. Choose sizes n_p and n_q (number of bits) for the factors. In general (for non-prime number N), factors can be sized as per Eq. (8.3).
3. Compute the problem Hamiltonian by expanding $H_{n_p,n_q} = (N - p_{n_p} q_{n_q})^2$.
4. Reduce all interactions with more than 2 qubits in H_{n_p,n_q} by introducing ancillary variables and adding the necessary penalties.
5. Remove the expected energy constant (Eq. (8.9)), such that $H_P = H_{n_p,n_q} - X$.
6. Set the coupling strengths and biases of each qubit as per the coefficients of terms in the problem Hamiltonian H_P.
7. Send the problem to a D-Wave quantum processor. This will return the minimum energy configuration, H_P.
8. Add the expected energy Eq. (8.9) to the minimum energy found, to get $H_{n_p,n_q} = H_P + X$.
9. If $H_{n_p,n_q} = 0$, then a valid factorisation was found. Energy values greater than zero will represent numbers that multiply close to N.
10. Read the values of the x_p and x_q qubits and convert back to binary integers to determine the resulting factors p and q.

8.1.5 Worked Example

Let $N = 15$ be the number to be factorised. Based on Eq. (8.3), the maximum possible size of factors will be $n_p = 2$ and $n_q = 3$. Thus, the factorisation Hamiltonian $H_{2,3}$ must be computed, as in the process described previously. After expansion and reduction, $H_{2,3}$ is as seen in Eq. (8.8). The biases and couplings are extracted to form the spin graph seen in Fig. 8.1a on the next page.

For $N = 15$, the minimum solution configuration is shown in Fig. 8.1b, with a total energy of $H_P = 768 + 208 - 512 - 512 - 52 - 96 = -196$.

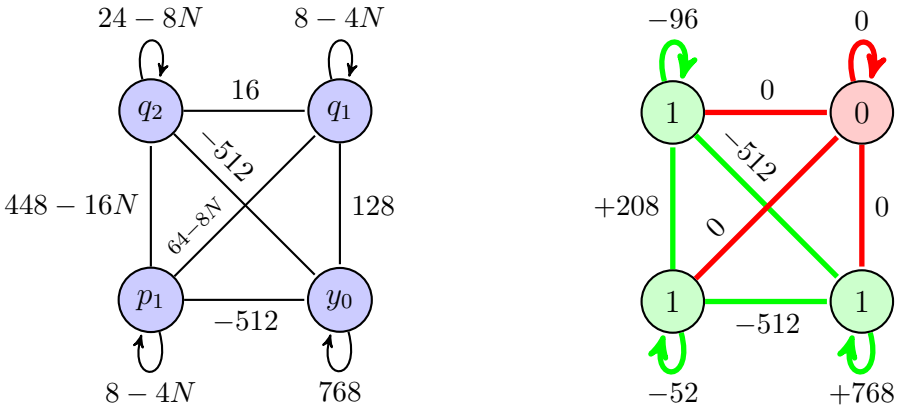
The expected energy is computed using Eq. (8.9): $X = 15^2 - 2 * 15 + 1 = 196$. Adding the expected and solution energies together gives $H_P + X = -196 + 196 = 0$, and therefore the solution satisfies $(N - pq)^2 = 0$ (as per Eq. (8.1)).

The qubit configuration is read off the solutions provided by the D-Wave Two, being $\{x_{p_1}, x_{q_2}, x_{q_1}\} = \{1, 1, 0\}$. Converting these values into their respective binary

representations (using Eq. (8.2)) results in:

$$\begin{aligned} p_2 &= 2^1 * 1 + 1 = 3 \\ q_3 &= 2^2 * 1 + 2^1 * 0 + 1 = 5 \end{aligned} \tag{8.10}$$

As expected, the two factors of $N = 15$ are $p_2 = 3$ and $q_3 = 5$.



(a) Spin graph **(b)** Minimum Energy

Figure 8.1 – (a) Problem graph for solving a generic $H_{2,3}$ integer factorisation problem. Note that $y_0 = y_{p_1,q_2}$.
(b) Minimum solution configuration when $N = 15$.

8.2 Problem Scalability

The `Integer Factorisation` algorithm is limited by a number of parameters. Firstly, by the size of the factorisation Hamiltonian (in number of terms), as shown in Fig. 8.2 on the following page. There is a cubic trend in the Hamiltonian size with increasing factor size, which results in significantly higher numbers of multi-qubit interaction terms, and by extension, more ancillary qubits are required. The factorisation Hamiltonians only need to be generated once, but this trend causes the generation time to rise rapidly, taking several hours to generate a Hamiltonian as small as $H_{10,10}$.

This results in the trend of qubits required to solve a single factorisation problem, as seen in Fig. 8.3. Although the problem graph is not fully-connected as in previous problems, the embedding into the D-Wave's Chimera graph still grows significantly with factor size.

All of these parameters play a part in the performance of the algorithm. The diminishing performance of the algorithm with growing factor size can best be seen

in Fig. 8.4 on the next page, in which all odd non-prime numbers from $N \geq 9$ are factorised. Each probability is an average of factorising the same number 30 times with an annealing time of 200μs.

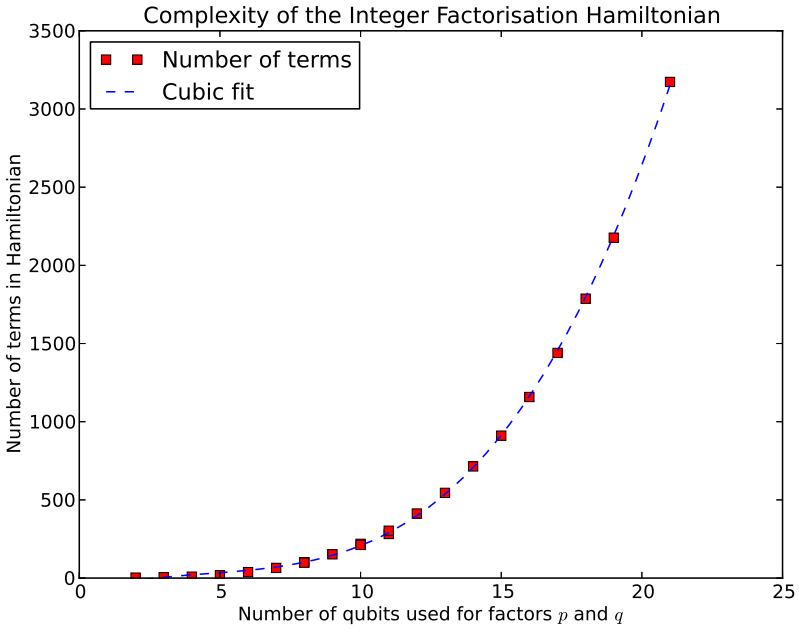


Figure 8.2 – Size of the factorisation Hamiltonian as dependent on the number of bits used for the factors p and q.

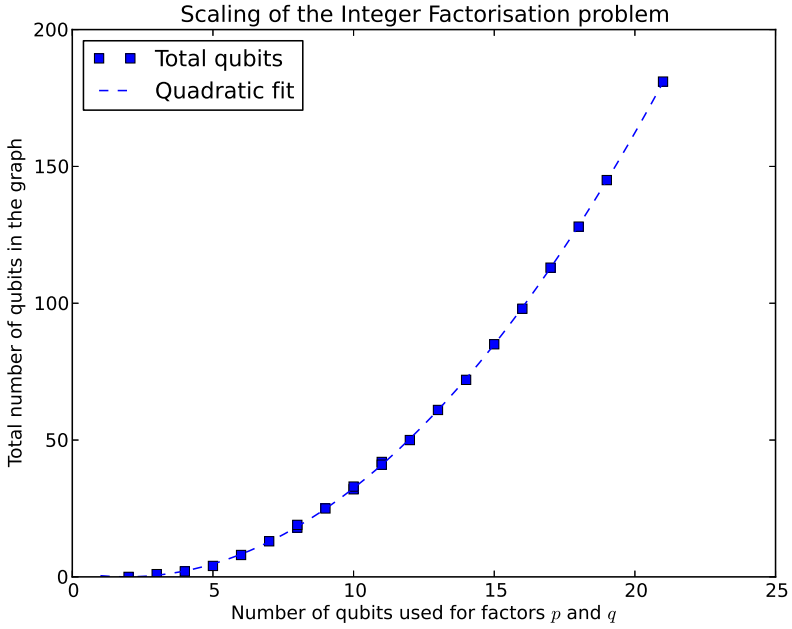


Figure 8.3 – Number of qubits required (including ancillary qubits, but prior to embedding) to factor a number depending on the number of qubits used for the factors p and q.

The sudden drop-off occurring immediately following the factorisation of the number $N = 21$ is due to Eq. (8.3), which determines the factor sizes required to factor a given number. At $N = 25$, the factorisation Hamiltonian switches from $H_{2,3}$ to $H_{3,3}$, resulting in the sudden drop of solution fidelity.

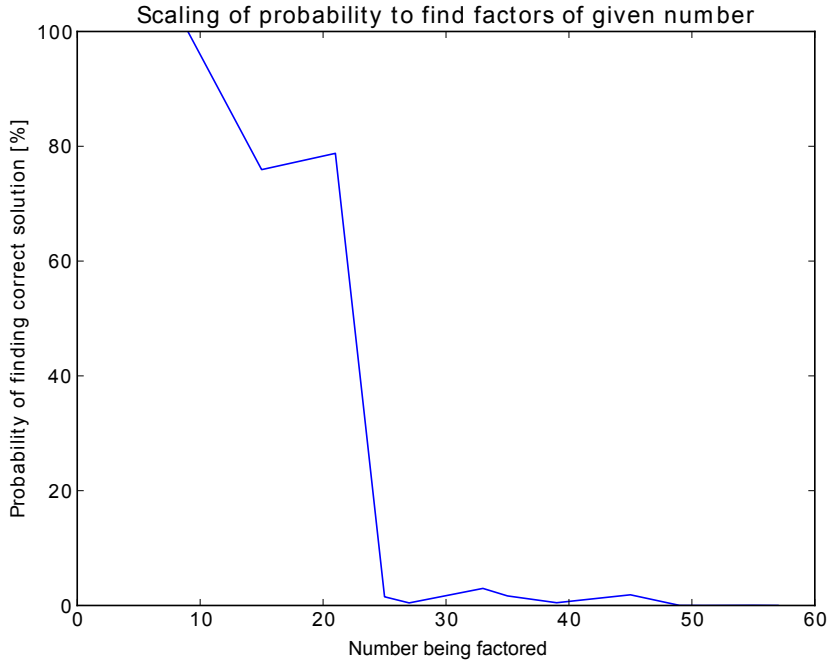


Figure 8.4 – The probability to find factors of a given number.

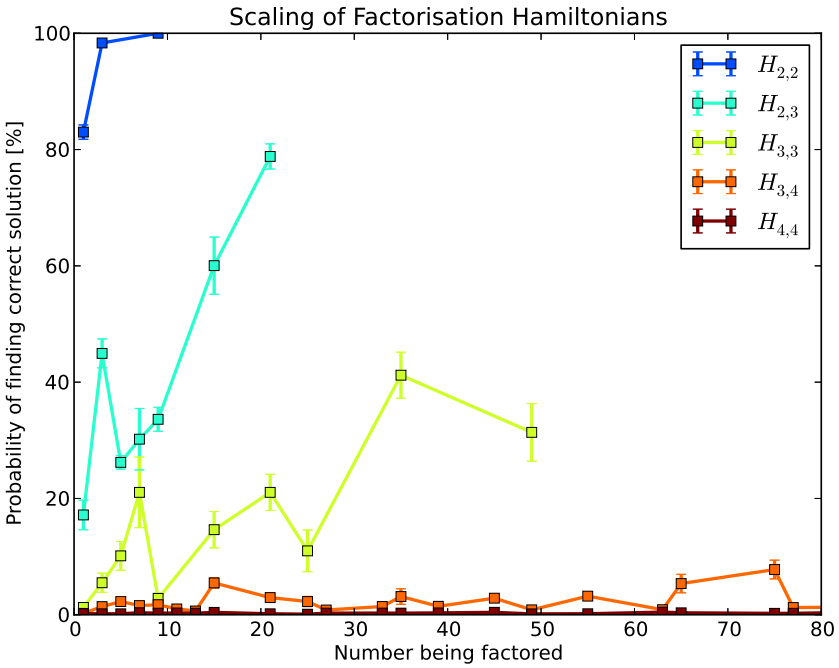


Figure 8.5 – The probability of factorising each number depending on the factorisation Hamiltonian being used.

8.2.1 Effect of Embeddings

In Fig. 8.5 on page 76, each factorisation Hamiltonian is shown, along with the probabilities of factorising numbers using that Hamiltonian. Each data point represents 150 problems, solving with an annealing time of 200μs for approximately 5000 anneals.[2] Despite this large amount of data, there is no clear pattern to the probabilities of each factorisation Hamiltonian.

Part of this, as it turns out, is due to the embeddings of the problems into the Chimera graph. D-Wave's heuristic `find_embedding` function is used to embed the problem graphs into the D-Wave Two's Chimera graph. The heuristic uses a pseudo-random process to find embeddings,[16] which can be controlled by specifying a starting seed for the process. In Fig. 8.6, the factorisation Hamiltonian $H_{3,3}$ shows that different embeddings unexpectedly result in different probability profiles, an effect that is also explored later in Section 9.

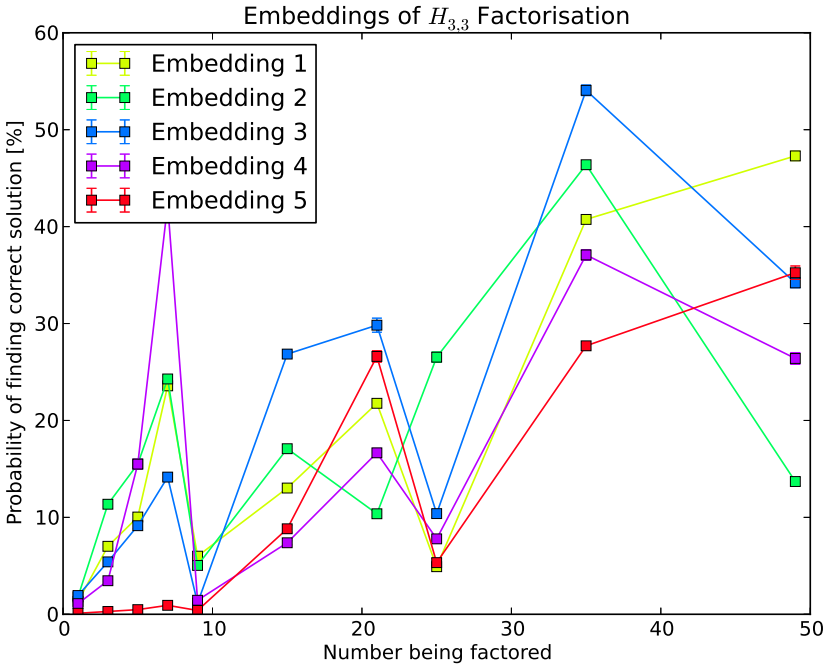


Figure 8.6 – The effect of different embeddings on the probabilities of factorising using $H_{3,3}$.

However, no clear trend remains even when embeddings are kept constant. The reason for this is likely due to the large range of biases and coupling strengths in the factorisation Hamiltonians, as seen in Eq. (8.8). Such a large range can cause issues

[2]Unfortunately, a shorter annealing time was not attempted due to the time constraints in producing this thesis.

with the limited precision available in the hardware, potentially causing certain qubits to be more prone to random flipping.

8.2.2 Largest Number Factorised

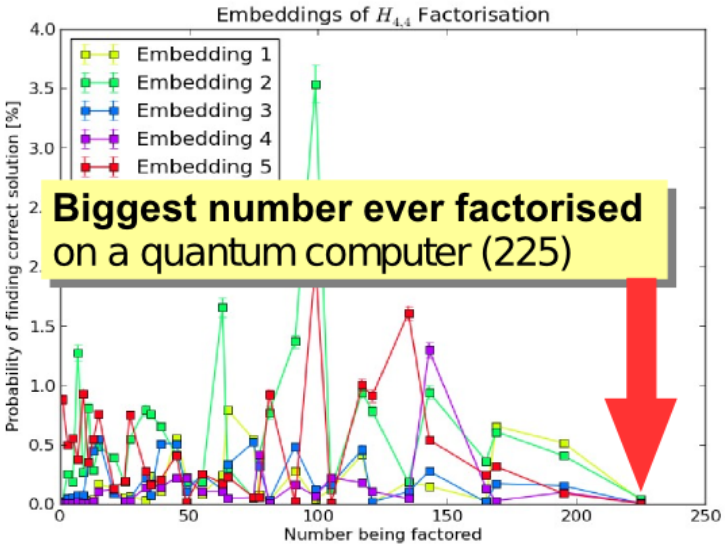


Figure 8.7 – Graph showing the results of the $H_{4,4}$ Hamiltonian, including valid factorisations of the number $N = 225$, likely the biggest number factorised by any quantum computer.

Although the factorisation Hamiltonian presented is somewhat unreliable in computing factors for certain numbers, it was nevertheless capable of finding factors of numbers up to $N = 225$ (using the Hamiltonian $H_{4,4}$). To the best knowledge of the author, this result is the largest number factorised on a quantum computer to date, beating the previous record of $N = 143$ held by Xu et al. (2012) [78]. However, the probability of finding the factors of $N = 225$ is absurdly low, occurring in $< 0.1\%$ of all anneals, or roughly only once every 5000+ anneals.

Theoretically, the D-Wave Two should be capable of factorising up to $N = 8001$ (by using $H_{6,7}$, which can be embedded as a graph of 61 spins into ~ 460 qubits in the Chimera graph). However, based on the scaling seen in the probabilities of finding factors with increased factor size (Fig. 8.5), chances are slim of observing a valid factorisation above $N = 225$ – as such, it is likely the biggest number able to be factorised on the current generation of hardware.

8.3 Conclusions

The algorithm for `Integer Factorisation` by Peng et al. (2008) [55] was adapted to the D-Wave Two hardware, and has been shown to successfully perform factorising of numbers larger than those ever factorised on quantum computers before.

This section has shown the following:

- The largest number ever factorised on a quantum computer ($N = 225$).
- Poor scaling with factor size. This affects the size of the Hamiltonian itself, as well as the number of spins required for the problem (and due to embedding, an even higher number of qubits).
- Potentially an issue with bias and coupling resolutions, which lead to chaotic probabilities of finding a factor, with no easily discernible trend (Fig. 8.5).
- The number of ancillary qubits required to reduce multiple-qubit interactions goes up at least quadratically with size of the factors.
- The total number of connections between qubits goes up sub-exponentially (less than a fully-connected graph). However, it still requires embedding into the D-Wave's Chimera graph, further increasing the number of qubits required.
- The difficulty of computing the $H_{i,j}$ factorisation Hamiltonian itself increases rapidly with the size of i and j. Although these Hamiltonians need only be computed once, they take in the order of hours for factors as small as $i = j = 10$.
- A maximum factorisation Hamiltonian of $H_{6,7}$ can be embedded into the D-Wave Two.
- The actual embedding of the problem greatly affects the probability landscape of the problem, though with no noticeable trends (Fig. 8.6).

Now that `Integer Factorisation` has been successfully demonstrated on the current hardware, future work can focus on:

- Investigating the scaling of computing larger factors, particularly when next-generation hardware is made available.
- Investigating the more advanced `Integer Factorisation` algorithm proposed by Xu et al. (2012).
- Investigating the application of Shor's algorithm[70], likely requiring next-generation hardware to be possible.

Section 9
Algorithm: Exact Cover and Sudokus

The `Exact Cover Problem` is an **NP**-Complete problem similar to set packing problems. Given a list of unique elements, and some sets containing combinations of those elements, the `Exact Cover Problem` asks to find a collection of sets such that each element appears in exactly one set, and all elements are covered in the collection.

`Exact Cover` problems have applications in any problem requiring strict constraints satisfaction, such as in scheduling problems (for example, ensuring all aircraft and crew are assigned depending on each individual crew's constraints).[72]

In this section, an algorithm capable of solving `Exact Cover` problems on the D-Wave Two is introduced, in both a programmatic form, as well as a directly-embeddable Hamiltonian. Furthermore, a direct application of the `Exact Cover Problem` is demonstrated in the form of a quantum Sudoku solver, the first of its kind in the world.

9.1 Lucas Formulation

The problem's Ising formulation, as given in Lucas (2014) [45] under Exact Cover, is first described. Consider a universal set of n unique elements, represented as $U = \{1, \ldots, n\}$. Consider also a list of N subsets of U, being the subsets $V_1, \ldots, V_N$ where each $V_i \subseteq U$. The aim of `Exact Cover` is to find a combination of subsets V_i such that every element in U is represented exactly once within all the sets.

Each set V_i is associated to a QUBO spin variable $x_i \in \{0, 1\}$, as such, N spins are required in total. The Hamiltonian for the problem is thus defined:

$$H = \sum_{\alpha=1}^{n} \left(1 - \sum_{\alpha \in V_i} x_i\right)^2 \tag{9.1}$$

where α is a counter for the elements in U, and i is a counter for the subsets V_i. The total energy is $H = 0$ when every α is represented in exactly one subset V_i. If an α is not in any subset, then the inner sum will equal to zero, and thus a penalty of $(1-0)^2 = 1$ would be applied. If an α is in more than one subsets, then the inner sum will be > 1, and thus $(1 - \sum)^2 > 0$.

9.2 Mapping to D-Wave hardware

9.2.1 Programmatic Hamiltonian

The Lucas Hamiltonian can easily be mapped into the D-Wave as a programmatic Hamiltonian. The following pseudo-code generates the Hamiltonian given the set V of all subsets V_i:

```
# A list of QUBO spin variables (one per subset V_i)
x = symbols(1..N)

# Build the Hamiltonian (see Lucas 2014)
H = 0

# For every element in U
for alpha in range(n):
    sum = 0

    # For every subset V[i]
    for i in range(N):
        if alpha in V[i]:
            sum = sum + x[i]

    H = H + (1 - sum)^2
```

From this point, H can be expanded and simplified using Python's symbolic module (`sympy`). The biases and couplings can then be easily extracted from the coefficients of the terms of H.

9.2.2 Directly-embeddable Hamiltonian

Alternatively, the Hamiltonian can be converted into a directly-embeddable Hamiltonian as follows. The squared term of Eq. (9.1) is first expanded for an arbitrary number of subsets:

$$\begin{aligned} H &= \sum_{\alpha=1}^{n} (1 - x_i - x_j - \cdots - x_k)^2 \\ &= \sum_{\alpha=1}^{n} \left(1^2 + (x_i^2 + x_j^2 + \cdots + x_k^2) - 2(x_i + x_j + \cdots + x_k) + 2(x_i x_j + x_i x_k + \dots)\right) \end{aligned} \tag{9.2}$$

Remembering that a QUBO spin variable is the same as its square ($x_i^2 = x_i$):

$$\begin{aligned} H &= \sum_{\alpha=1}^{n} (1 + (x_i + x_j + \cdots + x_k) - 2(x_i + x_j + \cdots + x_k) + 2(x_i x_j + x_i x_k + \dots)) \\ &= \sum_{\alpha=1}^{n} (1 - (x_i + x_j + \cdots + x_k) + 2(x_i x_j + x_i x_k + \dots)) \end{aligned} \tag{9.3}$$

Expanding the outer sum gives:

$$H = n - (|V_i|x_i + |V_j|x_j + \cdots + |V_k|x_k) + 2(|V_i \cap V_j|x_i x_j + |V_i \cap V_k|x_i x_k + \dots) \tag{9.4}$$

where $|V_i|$ is defined as the number of elements in the set V_i, and $|V_i \cap V_j|$ is defined as the number of elements shared between the two sets V_i and V_j.

9.3 Summary of the Algorithm

1. Let n represent the number of constraints that need to be covered by $V_1, \dots, V_N$, each a subset of $\{1, \dots, n\}$.
2. Assign each subset its own QUBO spin variable, x_i.
3. Apply a negative bias to each subset equal to the number of elements in that subset. That is, the bias for subset V_i is $-|V_i|x_i$.
4. Apply a positive coupling to each pair of subsets equal to double the number of elements both subsets share. That is, the coupling for subsets V_i and V_j is $2|V_i \cap V_j|x_i x_j$.
5. Send the problem to the D-Wave Two quantum processor. Add n to the solution that is returned to get the value of H.
6. If $H = 0$, then an exact cover was found. Otherwise, if $H > 0$, then an exact cover was found, but the solution returned is as close as possible.

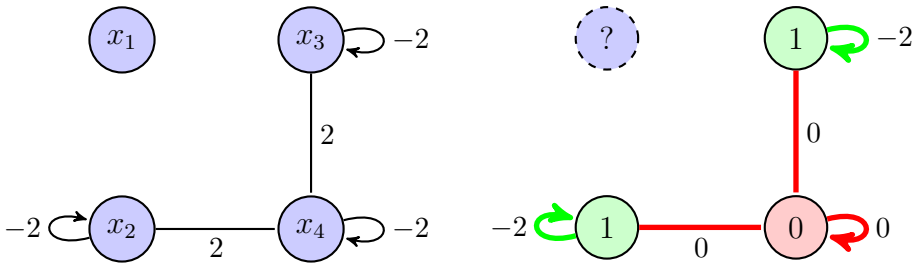
9.3.1 Worked Example

Let $U = \{1, \dots, 4\}$ be the set of constraint elements. Let there also be 4 subsets of U as follows:

- $V_1 = \{\}$
- $V_2 = \{1, 3\}$

- $V_3 = \{2, 4\}$
- $V_4 = \{2, 3\}$

As there are 4 subsets, there will be 4 QUBO spin variables, $x_1 \dots x_4$. The biases for each are subset V_i are set to the negative of the number of elements $|V_i|$ in it. The couplings between subsets V_i and V_j are set to the number of elements shared between them $|V_i \cap V_j|$. Thus, the problem graph is as shown in Fig. 9.1a.



(a) Spin graph **(b)** Minimum Energy State

Figure 9.1 – Example graph and solution for the `Exact Cover` problem described. Each circle denotes a qubit associated with a specific subset. The numbers looping to a qubit represent its bias, and between qubits their coupling strength. Dashed qubits have no effect on the solution.

By running the adiabatic "annealing" process on the spin graph, the lowest energy solution is computed: $H = -4$ (as represented in Fig. 9.1b). The expected energy of the problem is $n = 4$, so this is added to the returned energy to find that $H = -4 + 4 = 0$. Thus, this solution is an exact cover.

The subsets which define the exact cover are those with a final spin of $x_i = 1$. This means that subsets V_2 and V_3 form an exact cover. Inspection shows that $V_2 \cup V_3 = \{1, 2, 3, 4\} = U$ is a valid exact cover solution.

9.4 Solving Sudokus

The `Exact Cover Problem` is a perfect candidate for testing out real-world problems, as unlike most of the problems described so far, `Exact Cover` does not require a fully-connected spin graph. As such, it is much easier to embed larger problem sizes on to the D-Wave Two hardware. Furthermore, Sudoku puzzles are an exact application of `Exact Cover`, and are easy to understand and test.[8]

9.4.1 Sudoku Puzzles

Sudoku puzzles (Fig. 9.2) are composed of a grid of $N * N$ cells containing numbers from 1 to N. The applicability of Sudoku puzzles to `Exact Cover` is that the puzzle is all about constraints satisfaction.[8] To solve a Sudoku puzzle:

- Each cell must contain a digit from 1 to N.
- Each row must contain all of the digits 1 to N.
- Each column must contain all of the digits 1 to N.
- Each $\sqrt{N} * \sqrt{N}$ block of the grid must contain all of the digits 1 to N.

						8	2	
	2		3	7			5	
1			8		4		7	3
	7	6	9			2		4
			7		2			
2		8			1	7	3	
4	1		6		5			7
	5			3	8		9	
	8	3						

(a) Puzzle

7	3	4	1	5	9	8	2	6
8	2	9	3	7	6	4	5	1
1	6	5	8	2	4	9	7	3
5	7	6	9	8	3	2	1	4
3	9	1	7	4	2	5	6	8
2	4	8	5	6	1	7	3	9
4	1	2	6	9	5	3	8	7
6	5	7	4	3	8	1	9	2
9	8	3	2	1	7	6	4	5

(b) Solution

Figure 9.2 – Example of a $9 * 9$ Sudoku puzzle and its corresponding solution.

9.4.2 Defining Constraints

Consider an $N * N$ Sudoku puzzle, in which all cells, rows, columns, and blocks must contain the numbers 1 to N. In order to fit the Sudoku puzzle into an `Exact Cover` solver, a constraints matrix must be defined such that the constraints that must be satisfied form the columns of the matrix, and the ways of satisfying them form the rows of the matrix.

There are four types of constraints to satisfy. There are N rows, and they must each contain N values, as such there are $N * N$ constraints to be satisfied. There are N columns, and they must each contain N values, as such there are $N * N$ constraints to be satisfied. There are N blocks, and they must each contain N values, as such there are $N * N$ constraints. Finally, there are $N * N$ cells, and they

must each contain a single value, as such there are $N * N$ constraints to be satisfied. In total, there are $4N^2$ constraints, or columns, in the constraints matrix.

The ways of satisfying constraints are to place one of N digits into one of $N * N$ cells – there are thus N^3 ways in total. Any combination of $(row, column, value)$ will satisfy four constraints in total: being in a cell, being a value in a row, being a value in a column, and being a value in a block.

The constraints matrix therefore simply consists of $4 * N^2$ columns and N^3 rows. Each row represents a unique combination of $(row, column, value)$ by marking the corresponding columns (one for each of the 4 constraints).

For example, $(4, 1, 7)$ means placing the value 7 in the fourth row and first column (and fourth block) of the puzzle. The constraint row will have four columns marked: one corresponding to $(row4, col1)$, one corresponding to $(row4, value7)$, one corresponding to $(col1, value7)$, and one corresponding to $(block4, value7)$.

9.4.3 Sudoku Exact Cover Algorithm

Once a constraints matrix for an $N * N$ Sudoku has been defined, it is very simple to perform the algorithm:

1. Let each column of the matrix correspond to one value of U. Thus, U contains the values 1 to $4 * N^2$.
2. Let each row of the matrix be a subset V_i of U – as each row contains four values corresponding to columns (values of U).
3. Create a list to store the subsets to be solved for.
4. For every value that exists in the initial Sudoku puzzle, add its corresponding subset to the list. For example, the number 2 in the upper right corner of Fig. 9.2a corresponds to the combination $(1, 8, 2)$, so the subset $V_{(1,8,2)}$ is added to the list.
5. For every empty cell, add all subsets that could match that cell. For example, the empty top left grid cell in Fig. 9.2a would add the subsets $(1, 1, 1)$, $(1, 1, 2)$, $\ldots (1, 1, 9)$ to the list.

The `Exact Cover` algorithm will work by choosing all the sets V_i such that all constraints (columns/values of U) appear in exactly one subset each. For the Sudoku problem, a total of $N * N$ subsets will be selected (one for each cell in the puzzle),

meaning that they will satisfy $4 * N * N$ constraints, which is the total number of constraints in the puzzle.

9.4.4 Sending to the D-Wave Two

Generating the Hamiltonian for the D-Wave Two is now as simple as following the pseudo-code in Section 9.2. A maximum of N^3 subsets can be selected (assuming a puzzle that is completely empty). As the number of subsets is the number of qubits required (plus a small embedding overhead), this means that the D-Wave Two can solve Sudoku puzzles of at most $N = \sqrt[3]{512} = 8$. Unfortunately, this is not a valid Sudoku size. As such, Sudokus of size $4 * 4$ (requiring 64 qubits at worst, plus embedding overheads) are tested in the following sections.

9.4.5 Performance of the Sudoku Solver

Performance of the Sudoku solver can be seen in Fig. 9.5 on page 89. Here, each data point represents the average probability to find the correct solution to the Sudoku puzzle out of 30 runs with the same parameters. The graph shows that, for certain embeddings (more on this shortly), the Sudoku solver could achieve as high as 50% probability to find the ground state reliably. By using Eq. (4.1), this translates to $k \approx 7$ anneals required in order to find the ground state with 99% certainty.

Additionally, when the solver did not find the correct solution to the puzzle, it would return a very low energy solution, effectively solving most of the puzzle save for a missed constraint or two. This again re-iterates the benefits of the annealing approach to problems, as was also discussed by Perdomo-Ortiz et al. [56] and Rieffel et al. [62].

9.5 Problem Scalability

It is easy to qualify the scalability of the `Exact Cover` algorithm thanks to its simplicity. `Exact Cover` scalability is not as dependent on problem size as the other algorithms covered in this thesis, as the `Exact Cover Problem` requires only a small number of connections between qubits, and no ancillary qubits. While extra qubits are always necessary to embed the problem graph into the D-Wave's Chimera graph, substantially less qubits are required than if the problem graph were fully-connected. An example of this is when solving $4 * 4$ Sudokus, which require approximately $40 \sim 60$ qubits but can be easily embedded into the D-Wave Two Chimera graph.

This is in contrast to fully-connected graphs, which can not be embedded into the 512-qubit Chimera graph when they are bigger than $25 \sim 30$ qubits.

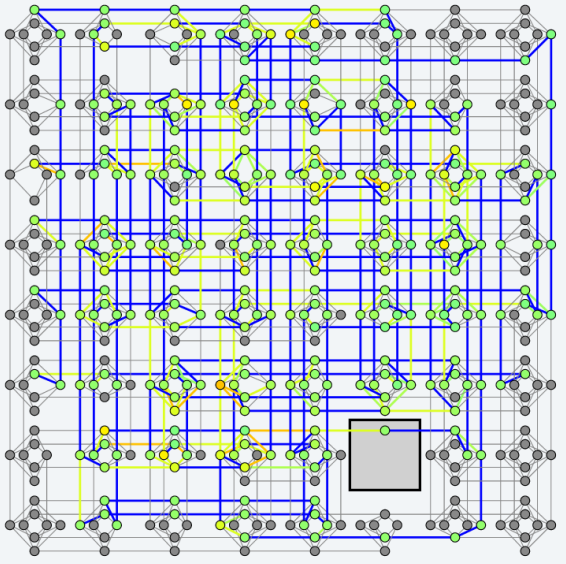


Figure 9.3 – Embedding a $4 * 4$ Sudoku into the D-Wave Two Chimera graph is not a problem. Shown is the Chimera graph for the Canadian D-Wave Two. Qubits and couplings are coloured according to their strengths.

Secondly, there is little issue with scalability due to precision problems, as the biases and couplings depend on the size of the problem, and not on its content. This is in contrast to the `Number Partition Problem` and `Integer Factorisation`, which have huge discrepancies between qubit biases and couplings, resulting in significant issues due to the limited precision of the hardware.

The performance of the `Sudoku Solver` described below shows that the `Exact Cover` algorithm maintains high reliability even when solving large (~ 250 qubits) problems, due to the properties above.

9.5.1 Performance

The algorithm described here successfully solves Sudoku puzzles of size $4 * 4$, by embedding them within ~ 250 hardware qubits. As there are not many tweakable parameters to test in a Sudoku puzzle, an investigation was undertaken of the effects of the annealing time on the solution probability. This is seen in Fig. 9.4 on the following page. Each data point represents the average of 30 runs of the same problem for a given embedding and annealing time.

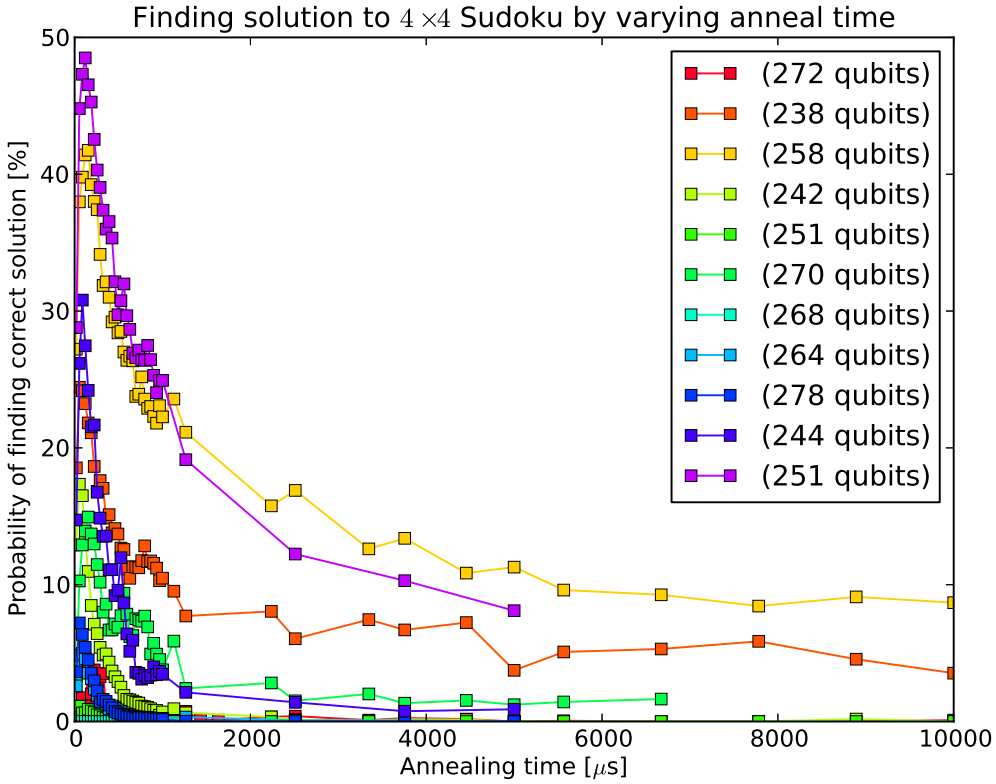


Figure 9.4 – Overview of the effects of annealing time on the probability of finding a solution to a Sudoku puzzle. Each line represents a different graph embedding (and number of embedded qubits).

The first point to note is that there is a significant drop in solution probability as the time to perform an anneal is increased. At first, this appears to be counter-intuitive: according to Section 2.3.4, the adiabatic theorem guarantees that the Hamiltonian will remain in the ground state if the annealing time is long enough. However, this is clearly not the case.

Instead, the reason for the decrease in the solution probability is one of quantum decoherence, as described in Section 2.2.2. The longer annealing times mean that there is a longer period for thermal and electromagnetic noise to destabilise the solution, nudging it out of the ground state. For this reason, longer annealing times are not always better, but rather much worse for most problems at this scale.

The second point to note is that the highest probabilities of finding the correct solution occur close to the very minimum annealing time possible. A closer look into this range is provided in Fig. 9.5 on the next page, which shows that this specific problem had an optimal annealing time of $\sim 100\mu$s. The drop in probability at annealing times lower than that is most likely due to the effects of the minimum energy gap (as in Section 2.3.4): the problem does not spend enough time in the

annealing process to remain in or settle into the ground state. However, despite the lower probability, the overall time to solution (as calculated using Eq. (4.2)) remains lowest at the shortest annealing time of 20μ s, which agrees with the general concensus for programming the D-Wave Two.[64]

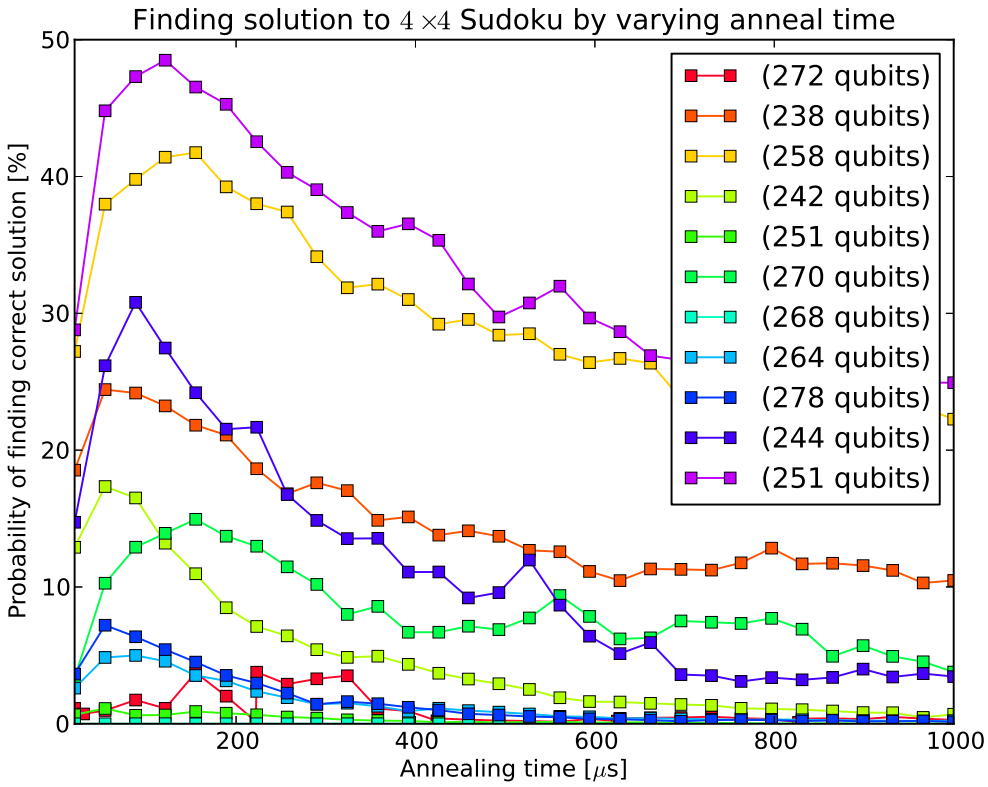


Figure 9.5 – Overview of the effects of annealing time and the embedding used on the probability of finding a solution to a Sudoku puzzle. Each line represents a different graph embedding.

9.5.2 Effect of Embeddings

Finally, a discussion on embeddings is necessary, as they provide the most unexpected source of reliability issues for this problem than anything else.

While this effect was first pointed out in the `Integer Factorisation Problem` (see Section 8.2.1), the effects of embedding the same problem in a different way (using different qubits) makes perhaps the most significant difference of all towards the quality of the solutions.

This is most clearly seen in Fig. 9.5, in which different embeddings clearly cause a substantial drop in solution probability, in some cases almost to 0%. There is no clear link to the actual size of the embeddings, and some of the details of how the embeddings work are hidden away by the D-Wave API. For this reason, a short

investigation was conducted to probe a little deeper into this effect, and is provided in Section 10.

9.6 Conclusions

The `Exact Cover` algorithm introduced in this section provides one of the most perfect cases of a useful algorithm for the D-Wave Two quantum computers. The algorithm displays virtually no issues with scalability due to precision or problem size, and is easily applied to any relevant problem through the directly-embeddable Hamiltonian introduced in this thesis.

Furthermore, a real-world application was built on the `Exact Cover` algorithm and shown to have excellent performance. With the right embeddings, the `Sudoku Solver` was capable of extremely high reliability in finding the correct solution to puzzles, very quickly.

The main outcomes of this section were as follows:

- An algorithm for a directly-embeddable version of the `Exact Cover Problem`, with apparently excellent performance (when compared to the other algorithms developed in this thesis) and scalability.
- An algorithm for solving Sudoku puzzles on the D-Wave Two quantum annealers, demonstrating a real-world application building upon a fundamental algorithm developed for this thesis.
- An investigation into the scalability properties and performance of the `Exact Cover` and Sudoku algorithms.

Further work could involve the following:

- A detailed analysis of the embeddings issue, to understand what exactly is causing it.
- The development of algorithms in the same class as `Exact Cover`, such as for set packing problems. It is possible that these problems may share the same desirable characteristics.
- The development of more complex applications building upon the `Exact Cover` algorithm. This includes applications in constraints satisfaction, scheduling, and so on.

Section 10
Investigation: Effect of Embeddings

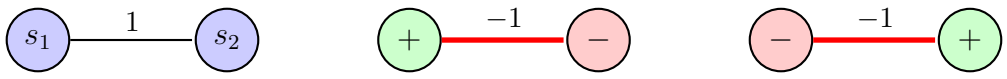
Due to the unexpected results that plagued the `Exact Cover Problem` with reliability issues (Section 9), a short investigation was performed into the effects of graph embeddings.

In this section, a simple test is described for verifying the accuracy of qubits on the D-Wave Two machines, as well as to determine whether there is some effect on solution properties due to the specific embedding used.

10.1 Problem Set-up

In order to effectively evaluate embeddings, a simple method of testing was devised. The tests would consist of selecting edges that exist in the D-Wave Two's Chimera graph, setting up a problem using the single edge, and keeping track of trends in the results.

A simple problem type was devised requiring the use of only two inter-connected qubits. Two qubits connected by a physical edge in the hardware would be set up as in Fig. 10.1a: no biases, and a positive coupling between them. This problem results in two ground state solutions with opposite qubit configurations. Without any sources of bias provided to them, the qubits are forced to choose one configuration or the other. The purpose of this test is to determine whether specific qubits are more prone to choosing one spin over another, whether this be due to noise, calibration issues, or for any other reasons.



(a) Spin graph **(b)** Ground State #1 **(c)** Ground State #2

Figure 10.1 – The problem set-up for the simple embedding test, which consists of forcing the pair of qubits into choosing one of the two possible ground states.

10.2 Testing Methodology

Testing involved trying as many couplings available on the D-Wave Twos as possible. The same problem was run 10 times per qubit pair, with the minimum possible annealing time of 20μs to maintain as little decoherence as possible. Each test ran a total of $30,000$ anneals in succession, and the results were stored for each test.

The results straight from the machine generally returned three configurations: ground state #1 with approximately 50% of the results, ground state #2 with approximately 50%, and an undefined state an insignificant ($< 1\%$) amount of times.

Each qubit was then evaluated individually by considering all couplings they were a part of. For each coupling, a qubit was assigned an expected value of spin, named the "qubit bias" of the qubit (not to be confused with biases applied to setting up problems):

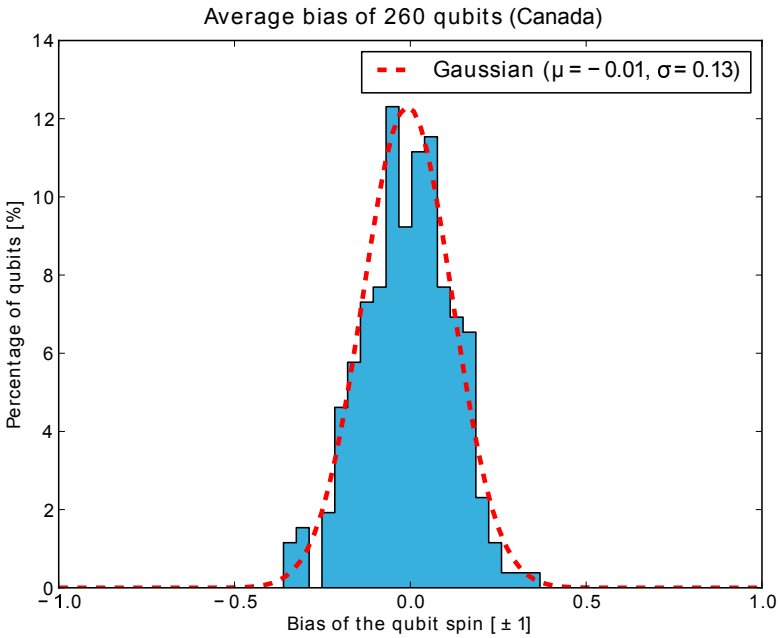
$$\begin{aligned} \textit{qubit bias} = {} & \textit{spin value in ground state } 1 * \textit{number of occurrences of ground state } 1 \\ & + \textit{spin value in ground state } 2 * \textit{number of occurrences of ground state } 2 \end{aligned}$$

The qubit bias was normalised by the number of times the ground states were observed. The bias was then averaged together with all of that qubit's biases in all other couplings. For most qubits, this meant that their resulting bias was the average of $50 \sim 60$ problems.

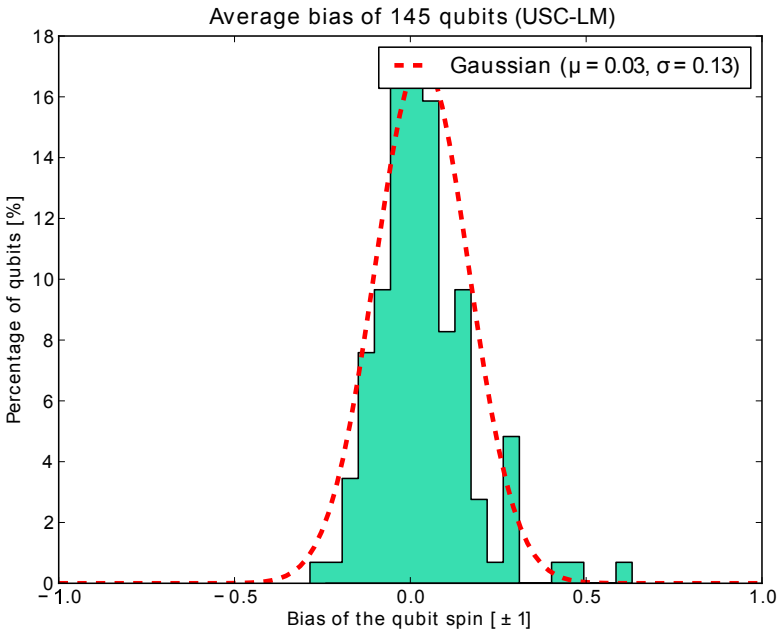
10.3 Results

The results in Fig. 10.2 on the following page show the distributions of qubit biases across both D-Wave Twos. While the vast majority of qubits had neutral biases (did not lean towards any specific spin), the surprising result was that many qubits did lean towards one spin, even if not by much. It is possible that these inherent biases could cause qubits to decide spins prematurely, affecting the rest of the solution. It is possible that these observations may be part of the cause for the embedding issues in the `Integer Factorisation` and `Exact Cover` algorithms presented earlier in this thesis.

It is worth noting that both D-Wave Twos were tested (with different number of qubits due to the limited availability of the machines) and displayed the same effect, implying that this is an issue inherent in the design of the D-Wave quantum computers. The distributions, which looked Gaussian, both had standard deviations of $\sigma = 0.13$, which is eerily similar to the standard deviation quoted for the precision of the qubits (as mentioned in Section 2.4.2). It is possible that biases seen here are simply a result of the natural deviation in the qubits.



(a) Canada



(b) USC-Lockheed Martin

Figure 10.2 – Perceived biases of each tested qubit on the two D-Wave Twos. The biases form a Gaussian distribution, with some qubits being biased more towards one spin than others.

Finally, Fig. 10.3 shows the relative correlation between the same qubits on different machines. The plot clearly shows that there is no correlation between them, and as such that the perceived biases are not due to an inherent design flaw affecting specific qubits.

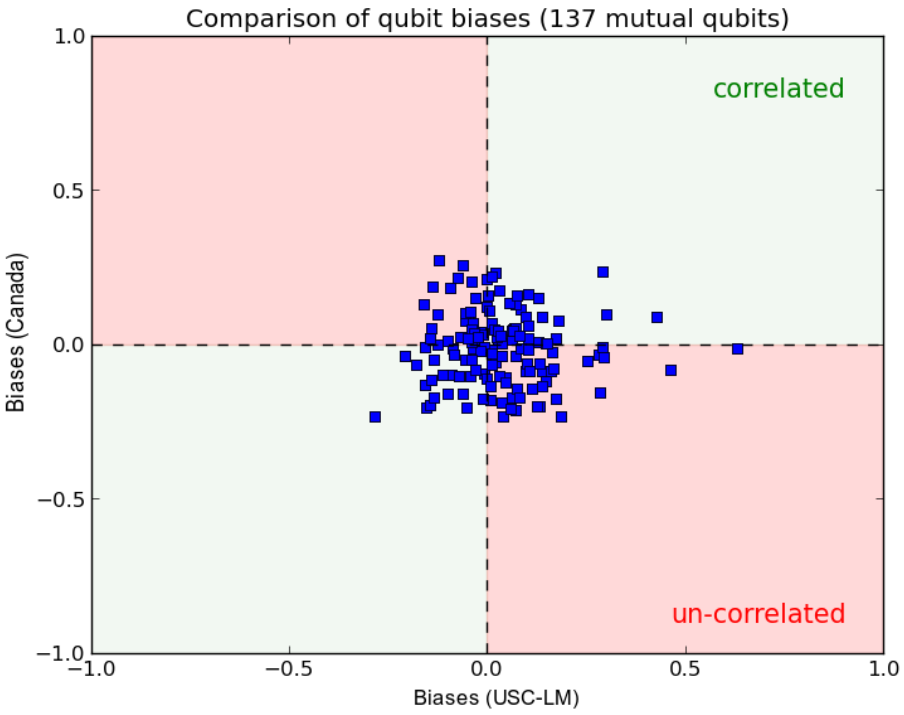


Figure 10.3 – Correlation (or lack thereof) between biased qubits on both D-Wave Two machines.

10.4 Conclusions

This section showed that, at the very least, there is some small but measurable inherent biases in some of the qubits on the D-Wave Twos, and that this is an architectural issue manifested in virtually the same way on both machines. While it is possible that some biased qubits may be affecting the solution landscapes for some problems, the deviation measured was small enough to imply that there is likely another issue causing the problems encountered in earlier Sections.

Section 11
Conclusions & Future Work

11.1 Contributions

The overall objectives of this thesis were to develop and successfully demonstrate algorithms for fundamental **NP**-Hard problems on a D-Wave Two. The work presented in Sections 5 to 9 provide a small step towards the greater understanding of these and harder challenges in the AQC realm.

The specific contributions of this thesis include:

- Directly-embeddable algorithmic implementations of 4 fundametal **NP**-Hard problems (`Number Partitioning`, `Graph Partitioning`, `Maximum Clique`, and `Exact Cover`).
- The demonstration of 2 real-world algorithms (`Integer Factorisation` and `Sudoku Solver`).
- A `Number Partitioning` algorithm that finds, on average in a single anneal, more optimal solutions than the leading classical heuristic.
- An `Integer Factorisation` algorithm demonstrating the largest number ($N = 225$) ever factorised on a quantum computer.
- A `Graph Partitioning` algorithm that performs approximately as well as a leading classical heuristic.
- A `Maximum Cliques` optimisation algorithm that does not depend on graph sparsity to find solutions.
- An `Exact Cover` algorithm that displays excellent overall scalability.
- The identification of the easiest types problems that can be solved by the quantum `Graph Partitioning` algorithm (very sparse or very dense graphs).
- A detailed introduction to and general methodology for solving problems on D-Wave Twos or similar adiabatic quantum computers.
- A demonstration of techniques used in converting Ising formulations into direct algorithms, as shown in the derivation of these algorithms in each Section, including: reducing multi-qubit interactions, expected energies, simplifying Ising and QUBO variables, expansion of sums.

- A simple analysis of the scalability of the problems outlined above, including identification of parameters limiting the reliability of these problems, such as the embeddings used and the effect of annealing time on solutions.
- The identification of undefined behaviour on the D-Wave Two.
- The identification of a slight bias in qubit behaviour, inherent to the design of the D-Wave Two.

The algorithms introduced in this thesis can be further analysed and used for the development of applications, as was done with the `Sudoku Solver`. Understanding the scalability of these problems is not just important for each problem itself – the fundamental nature of the problems chosen means that these scalability properties directly affect the design of more complex algorithms.

This thesis demonstrated that perhaps the most suitable choice of problems for working on the D-Wave Two are those similar to `Exact Cover`, which did not require a fully-connected problem graph (and thus was more easily embedded into the Chimera graph), and which was not significantly affected by precision limitations of the hardware.

The rest of the problems show scalability that was generally limited by connectivity available in the Chimera graph (and thus requiring expensive embeddings) or the precision of the qubit biases and couplings. `Integer Factorisation` and `Number Partitioning`, for example, both required solving problems with wide ranges of values, and thus higher precision requirements. While such limitations are not new revelations to the AQC community, it is only through demonstrating these limitations in real problems that allows hardware developers to truly understand what improvements need to be made.

To summarise, the algorithms presented in this thesis are small but important first experimental steps towards understanding the behaviour of **NP**-Hard problems on adiabatic quantum computers. The thesis itself also serves as a guided tour of current-generation D-Wave quantum computing hardware and the process behind design and testing of quantum algorithms. The author remains hopeful that this document will prove useful to those starting out in the field.

11.2 Future Work

In October 2014 (the publishing date of this thesis) D-Wave Systems announced the next generation of their adiabatic quantum computers sporting 2048 qubits. Having been announced around 2 years after the release of the D-Wave Two, this shows that adiabatic quantum computers are undergoing rapid development akin to that of modern computer technology. It is now, perhaps more than ever, the very beginnings of the quantum computing age.

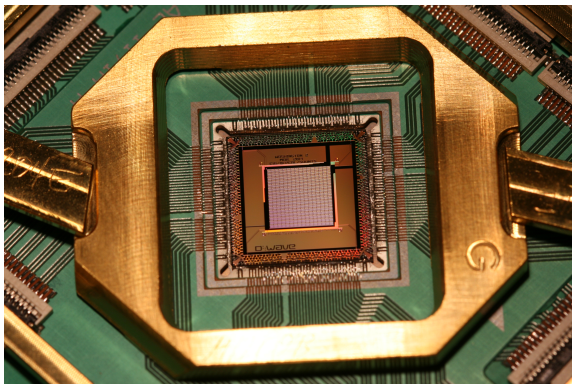


Figure 11.1 – The future of adiabatic quantum computing – D-Wave System's newly announced "Washington" chip.

There remains a lot of room for future research in the area of adiabatic quantum computing. There are still many fundamental **NP**-Hard problems (as defined by Karp, Lucas) that were not investigated in this thesis. There are published Ising formulations for many, many more problems, few of which have ever been attempted on D-Wave hardware. Understanding these problems now, at the dawn of large-scale quantum computing, could have significant ramifications on algorithmic design in years to come.

Performance analysis of quantum algorithms as compared to classical solutions is another important research area, in order to properly quantify the benefits of quantum computing. However, such research is often quite difficult, as the quantum and classical computing realms are different enough that direct comparisons can not easily be made. This is perhaps best demonstrated by the on-going debates surrounding the actual performance of D-Wave's hardware. Despite this, performance analysis of quantum algorithms is likely to yield some unexpected and highly important results.

Finally, a quick discussion on potential real-world applications of quantum computing. While any **NP**-Hard problem is fair game for solving on such hardware, it

is likely that adiabatic quantum computers similar to the D-Wave Two are more apt at solving specific types of problems. In particular, constraints satisfaction problems (such as the fundamental `Exact Cover`) look to be particularly promising, with applications in scheduling, routing, and logistics problems. Other potential applications exist in machine learning and pattern recognition, as in the work being done by NASA's quantum computing group. The inherent randomness of quantum computing could also potentially lead to improved performance in Monte Carlo simulation.

As the development of programmable quantum computers continues into the near future, plenty of unexplored territory will be made available, similar to the early days of computer technology. While quantum computing research today still focuses primarily on physical properties and implementations, the state of programmable quantum computers is rapidly approaching that of usefulness. It is important to understand the fundamentals now so that better algorithms for solving complex problems may be developed for the quantum world of tomorrow.

Appendix A
References

[1] S. Aaronson. 'Guest column: NP-complete problems and physical reality'. *ACM Sigact News* 36.1 (2005), pp. 30–52.

[2] S. Aaronson. 'How Might Quantum Information Transform Our Future?' (July 2014). URL: https://www.bigquestionsonline.com/content/how-might-quantum-information-transform-our-future.

[3] S. Aaronson. 'The limits of quantum computers'. *Scientific American* 298.3 (2008), pp. 62–69.

[4] D. Aharonov, W. van Dam, J. Kempe et al. *Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation*. SIAM Journal of Computing, Vol. 37, Issue 1, p. 166-194 (2007), conference version in Proc. 45th FOCS, p. 42-51 (2004). 2004. DOI: 10.1137/080734479.

[5] V. Allori. 'Decoherence and the classical limit of quantum mechanics'. PhD thesis. Physics Department, University of Genova, 2001.

[6] M. Amin and M. Steininger. *Adiabatic quantum computation with superconducting qubits*. US Patent 7,135,701. Nov. 2006.

[7] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. IT Pro. Cambridge University Press, 2009. ISBN: 9781139477369.

[8] D. Austin. *Puzzling Over Exact Cover Problems*. American Mathematical Society. 2014. URL: http://www.ams.org/samplings/feature-column/fcarc-kanoodle (visited on 12/07/2014).

[9] F. Barahona. 'On the computational complexity of Ising spin glass models'. *Journal of Physics A: Mathematical and General* 15.10 (1982), p. 3241.

[10] Z. Bian, F. Chudak, W. G. Macready, L. Clark and F. Gaitan. 'Experimental determination of Ramsey numbers'. *Physical review letters* 111.13 (2013), p. 130505.

[11] Z. Bian, F. Chudak, W. G. Macready and G. Rose. 'The Ising model: teaching an old problem new tricks'. *D-Wave Systems* (2010).

[12] S. Boixo, T. F. Rønnow, S. V. Isakov et al. 'Evidence for quantum annealing with more than one hundred qubits'. *Nature Physics* 10.3 (2014), pp. 218–224.

[13] S. Boixo, T. F. Rønnow, S. V. Isakov et al. *Quantum annealing with more than one hundred qubits*. 2013. eprint: arXiv:1304.4595.

[14] J. Cai, W. G. Macready and A. Roy. 'A practical heuristic for finding graph minors' (2014). eprint: arXiv:1406.2741.

[15] A. M. Childs, E. Farhi and J. Preskill. 'Robustness of adiabatic quantum computation'. *Phys. Rev. A* 65 (1 Dec. 2001), p. 012322. DOI: 10.1103/PhysRevA.65.012322.

[16] V. Choi. 'Minor-embedding in adiabatic quantum computation: I. The parameter setting problem'. English. *Quantum Information Processing* 7.5 (2008), pp. 193–209. ISSN: 1570-0755. DOI: 10.1007/s11128-008-0082-9.

[17] V. Choi. 'Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design'. English. *Quantum Information Processing* 10.3 (2011), pp. 343–353. ISSN: 1570-0755. DOI: 10.1007/s11128-010-0200-3.

[18] T. Cormen, C. Leiserson, R. Rivest and C. Stein. *Introduction To Algorithms*. MIT Press, 2001. ISBN: 9780262032933.

[19] R. Correll and S. Worden. 'The Applicability of Emerging Quantum Computing Capabilities to Exo-Planet Research'. 223rd American Astronomical Society Meeting, 2014.

[20] E. Crosson, E. Farhi, C. Y.-Y. Lin, H.-H. Lin and P. Shor. 'Different Strategies for Optimization Using the Quantum Adiabatic Algorithm' (2014). eprint: arXiv:1401.7320.

[21] E. D. Dahl. *Programming with D-Wave: Map Coloring Problem*. D-Wave Systems. Nov. 2013. URL: http://www.dwavesys.com/sites/default/files/Map%20Coloring%20WP2.pdf (visited on 16/03/2014).

[22] V. S. Denchev. 'Binary classification with adiabatic quantum optimization'. PhD thesis. Purdue University, 2013.

[23] V. S. Denchev, N. Ding, S. Vishwanathan and H. Neven. 'Robust classification with adiabatic quantum optimization' (2012). eprint: arXiv/1205.1148.

[24] *Developer Guide for C*. Version 1.5.0-beta1. D-Wave Systems Inc. Sept. 2013.

[25] *Developer Guide for Python*. Version 1.5.0-beta1. D-Wave Systems Inc. Sept. 2013.

[26] J. Dowling. *Schrödinger's Killer App: Race to Build the World's First Quantum Computer*. Taylor & Francis, 2013. ISBN: 9781439896730.

[27] G. Egan. *Quarantine*. Harper science fiction. HarperCollins, 1994. ISBN: 9780061054235.

[28] C. Epstein. 'Adiabatic quantum computing: An overview'. *Quantum Complexity Theory* 6 (2012), p. 845.

[29] E. Farhi, J. Goldstone, S. Gutmann and M. Sipser. *Quantum Computation by Adiabatic Evolution*. 2000. eprint: `arXiv:quant-ph/0001106`.

[30] E. Farhi, J. Goldstone, S. Gutmann et al. 'A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem'. *Science* 292.5516 (2001), pp. 472–475. DOI: `10.1126/science.1057726`.

[31] I. P. Gent and T. Walsh. 'Analysis of heuristics for number partitioning'. *Computational Intelligence* 14.3 (1998), pp. 430–451.

[32] M. Grajcar, A. Izmalkov, S. Van Der Ploeg et al. 'Four-qubit device with mixed couplings'. *Physical review letters* 96.4 (2006), p. 047006.

[33] L. Grossman. 'The Quantum Quest for a Revolutionary Computer'. *TIME Magazine* (Feb. 2014).

[34] L. K. Grover. 'A fast quantum mechanical algorithm for database search'. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM. 1996, pp. 212–219.

[35] R. Harris, J. Johansson, A. Berkley et al. 'Experimental demonstration of a robust and scalable flux qubit'. *Physical Review B* 81.13 (2010), p. 134510. eprint: `arXiv:0909.4321`.

[36] B. Hayes. 'The easiest hard problem'. *American Scientist* 90.2 (2002), pp. 113–117.

[37] D. S. Johnson, C. R. Aragon, L. A. McGeoch and C. Schevon. 'Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning'. *Operations research* 39.3 (1991), pp. 378–406.

[38] M. Johnson, M. Amin, S. Gildert et al. 'Quantum annealing with manufactured spins'. *Nature* 473.7346 (2011), pp. 194–198.

[39] K. Karimi, N. G. Dickson, F. Hamze et al. *Investigating the Performance of an Adiabatic Quantum Optimization Processor*. 2010. eprint: arXiv:1006.4147.

[40] R. Karp. 'Reducibility among Combinatorial Problems'. English. *Complexity of Computer Computations*. Ed. by R. Miller, J. Thatcher and J. Bohlinger. The IBM Research Symposia Series. Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2003-6. DOI: 10.1007/978-1-4684-2001-2_9.

[41] H. G. Katzgraber, F. Hamze and R. S. Andrist. *Glassy Chimeras could be blind to quantum speedup: Designing better benchmarks for quantum annealing machines*. 2014. eprint: arXiv:1401.1546.

[42] R. E. Korf. 'A complete anytime algorithm for number partitioning'. *Artificial Intelligence* 106.2 (1998), pp. 181–203.

[43] T. D. Ladd, F. Jelezko, R. Laflamme et al. 'Quantum computers'. *Nature* 464.7285 (2010), pp. 45–53.

[44] T. Lanting, A. Przybysz, A. Y. Smirnov et al. 'Entanglement in a quantum annealing processor'. *Physical Review X* 4.2 (2014), p. 021041.

[45] A. Lucas. *Ising formulations of many NP problems*. Frontiers in Physics 2, 5. 2014. DOI: 10.3389/fphy.2014.00005.

[46] C. C. McGeoch. 'Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice'. *Synthesis Lectures on Quantum Computing* 5.2 (July 2014), pp. 1–93.

[47] C. C. McGeoch and C. Wang. 'Experimental Evaluation of an Adiabatic Quantum System for Combinatorial Optimization'. *Proceedings of the ACM International Conference on Computing Frontiers*. CF '13. Ischia, Italy: ACM, 2013, 23:1–23:11. ISBN: 978-1-4503-2053-5. DOI: 10.1145/2482767.2482797.

[48] S. Mertens. *Number Partitioning*. Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, USA, 2006. ISBN: 9780199760565.

[49] *METIS - Serial Graph Partitioning and Fill-reducing Matrix Ordering*. 2013. URL: http://glaros.dtc.umn.edu/gkhome/metis/metis/overview.

[50] J. Myers. *The Current State and Potential of Quantum Computing*. Rowan University, Department of Computer Science. June 2013. URL: http://jackmyers.info/docs/quantum.pdf (visited on 12/03/2014).

[51] H. Neven, V. S. Denchev, M. Drew-Brook et al. 'NIPS 2009 demonstration: Binary classification using hardware implementation of quantum annealing'. *Quantum* (2009), pp. 1–17.

[52] H. Neven, V. S. Denchev, G. Rose and W. G. Macready. 'QBoost: Large Scale Classifier Training with Adiabatic Quantum Optimization.' *Journal of Machine Learning Research-Proceedings Track* 25 (2012), pp. 333–348.

[53] H. Neven, V. S. Denchev, G. Rose and W. G. Macready. *Training a Large Scale Classifier with the Quantum Adiabatic Algorithm*. 2009. eprint: arXiv:0912.0779.

[54] H. Neven, G. Rose and W. G. Macready. 'Image recognition with an adiabatic quantum computer I. Mapping to quadratic unconstrained binary optimization' (2008). eprint: arXiv/0804.4457.

[55] X. Peng, Z. Liao, N. Xu et al. 'Quantum adiabatic algorithm for factorization and its experimental implementation'. *Physical review letters* 101.22 (2008), p. 220405.

[56] A. Perdomo-Ortiz, J. Fluegemann, S. Narasimhan, R. Biswas and V. N. Smelyanskiy. 'A quantum annealing approach for fault detection and diagnosis of graph-based systems' (Oct. 2014). eprint: arXiv:1406.7601.

[57] S. D. Pinski. 'Adiabatic Quantum Computing' (2011). eprint: arXiv:1108.0560.

[58] J. Pla, K. Tan, J. Dehollain et al. 'A single-atom electron spin qubit in silicon'. *Nature* 489 (2012), pp. 541–545.

[59] *Programming with QUBOs*. Version 1.5.0-beta1. D-Wave Systems Inc. Sept. 2013.

[60] K. L. Pudenz, T. Albash and D. A. Lidar. 'Quantum annealing correction for random Ising problems' (Aug. 2014). eprint: arXiv:1408.4382.

[61] K. L. Pudenz and D. A. Lidar. *Quantum adiabatic machine learning*. 2011. DOI: 10.1007/s11128-012-0506-4. eprint: arXiv:1109.0325.

[62] E. G. Rieffel, D. Venturelli, B. O'Gorman et al. 'A case study in programming a quantum annealer for hard operational planning problems' (July 2014). eprint: arXiv:1407.2887.

[63] R. Rojas. 'AdaBoost and the super bowl of classifiers a tutorial introduction to adaptive boosting'. *Freie University, Berlin, Tech. Rep* (2009).

[64] T. F. Rønnow, Z. Wang, J. Job et al. 'Defining and detecting quantum speedup'. *Science* 345.6195 (July 2014), pp. 420–424. DOI: 10.1126/science.1252319.

[65] G. Schaller and R. Schützhold. 'The role of symmetries in adiabatic quantum algorithms' (2007). eprint: arXiv:0708.1882.

[66] A. Selby. *D-Wave: comment on comparison with classical computers*. June 2013. URL: http://www.archduke.org/stuff/d-wave-comment-on-comparison-with-classical-computers/ (visited on 03/04/2014).

[67] A. Selby. *D-Wave: scaling comparison with classical algorithms*. June 2014. URL: http://www.archduke.org/stuff/d-wave-comment-on-comparison-with-classical-computers/d-wave-scaling-comparison-with-classical-algorithms/ (visited on 03/08/2014).

[68] A. Selby. *Harder QUBO instances on a Chimera graph*. Sept. 2013. URL: http://www.archduke.org/stuff/d-wave-comment-on-comparison-with-classical-computers/harder-qubo-instances-on-a-chimera-graph/ (visited on 03/04/2014).

[69] S. W. Shin, G. Smith, J. A. Smolin and U. Vazirani. 'How" Quantum" is the D-Wave Machine?' (2014). eprint: arXiv:1401.7087.

[70] P. W. Shor. 'Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer'. *SIAM journal on computing* 26.5 (1997), pp. 1484–1509.

[71] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012. ISBN: 9781285401065.

[72] S. Skiena. *The Algorithm Design Manual*. Springer, 2009. ISBN: 9781848000704.

[73] V. N. Smelyanskiy, E. G. Rieffel, S. I. Knysh et al. *A Near-Term Quantum Computing Approach for Hard Computational Problems in Space Exploration*. 2012. eprint: arXiv:1204.2821.

[74] G. Smith and J. Smolin. 'Putting "Quantumness" to the Test'. *Physics* 6 (Sept. 2013), p. 105. DOI: 10.1103/Physics.6.105.

[75] J. A. Smolin and G. Smith. *Classical signature of quantum annealing*. 2013. eprint: arXiv:1305.4904.

[76] *Where do we stand on benchmarking the D-Wave 2?* Google Quantum A.I. Lab. Jan. 2014. URL: https://plus.google.com/+QuantumAILab/posts/DymNo8DzAYi (visited on 22/05/2014).

[77] Wikipedia. *ErdosRenyi model — Wikipedia, The Free Encyclopedia*. 2014. URL: http://en.wikipedia.org/w/index.php?title=Erd%C3%85%C2%91s%C3%A2%C2%80%C2%93R%C3%83%C2%A9nyi_model&oldid=626899391 (visited on 24/08/2014).

[78] N. Xu, J. Zhu, D. Lu et al. 'Quantum factorization of 143 on a dipolar-coupling nuclear magnetic resonance system'. *Physical review letters* 108.13 (2012), p. 130501. DOI: 10.1103/PhysRevLett.108.130501.

Appendix B
Nomenclature

H_P Problem Hamiltonian

API Application Programming Interface

AQC Adiabatic Quantum Computation

CKK Complete Karmarkar-Karp algorithm

CPU Central Processing Unit

KK Karmarkar-Karp algorithm

NP Non-deterministic Polynomial-time

P Polynomial-time

QA Quantum Annealing

QC Quantum Computation

QUBO Quadratic Unconstrained Binary Optimisation

SA Simulated Annealing

USC University of Southern California

USC-LM University of Southern California-Lockheed Martin

VPN Virtual Private Network

Appendix C
Project Timeline

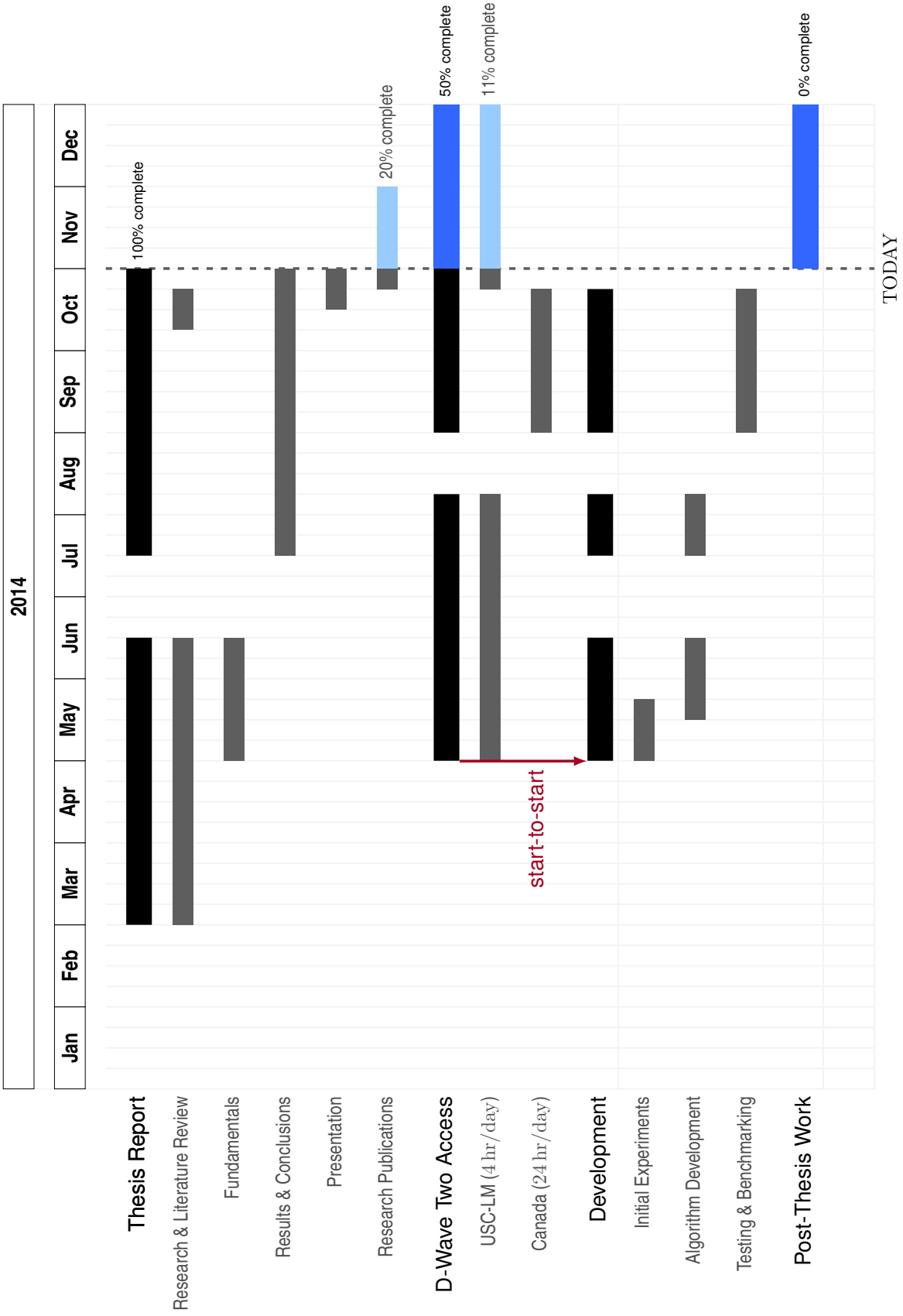


Figure C.1 – Gantt Chart – Approximate project timeline