COMPILING PROGRAMS FOR AN ADIABATIC QUANTUM COMPUTER

by

Elliot Snow-Kropla

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science

at

Dalhousie University
Halifax, Nova Scotia
August 2014

To Kim

Table of Contents

List of Tables

List of Figures

Abstract

A method for compiling programs for adiabatic quantum computers is described, and the resulting programs evaluated on a physically implemented adiabatic quantum machine. These results are used to characterize the performance of the machine. The AQC machine is found to perform well on some problems with linear scaling with problem size for some instances, but it performs poorly on others and is unable to solve a 6-variable satisfiability problem at all.

List of Abbreviations and Symbols Used

α	The ratio M/N of clauses over variables in a SAT problem. For a given k-SAT there is a single α which produces the most difficult problem instances.
σ_i^z	The operator for spin along the z axis of the ith particle.
D-Wave Two	The D-Wave Two is an machine designed to be an adiabatic quantum computer created by D-Wave Systems
`QCC`	The program collection which compiles `QSM` programs into embeddable Hamiltonians.
`QSM`	A high level language for describing Hamiltonians.
AQC	Adiabatic Quantum Computing
Clone Coupling	The coupling value that the embedding process uses to ensure different qubits share the same logical value.
SAT	Boolean Satisfiability is an NP-Complete problem, the first such discovered.

Acknowledgements

I would like to acknowledge the help of numerous people whose assistance was invaluable in assembling this document, including my supervisor Jordan Kyriakidis, Todd Belote, Aaron Whiteway, Micah McCurdy and Jeff Egger.

Chapter 1

Introduction

Quantum computing is the study of computers which operate according to the laws of quantum mechanics rather than the laws of classical physics. Quantum computers are theorized to be more efficient than classical computers because quantum algorithms for integer factorization[30] and global search[13] have been found that are asymptotically more efficient than the best known classical algorithms, although it is not proven that there are no equivalent classical algorithms.

Unlike classical computers, which operate on bits, quantum computers operate on quantum bits or qubits. While classical information must take on a definite value (i.e. 0 or 1), quantum information can take on superpositions (the familiar quantum mechanics: $\frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]$), as well as become entangled. The laws of quantum mechanics, specifically the way in which Hilbert spaces behave for multiple particles, require that in order to fully describe a quantum system consisting of n two-level particles we must record 2^n complex numbers. Classical systems require only $O(n)$ numbers to be recorded to fully describe a system of n particles. This means that quantum systems require exponentially more computational resources than classical systems do, or alternatively that the computational power of the actual, quantum, universe is much larger than that of the apparent classical universe.

1.1 Computational Complexity

Computational Complexity theory is the branch of computer science dealing with the study of computational difficulty, and of classifying different computational problems according to their difficulty. The problem of distinguishing the performance of a particular method of computation from the process of computation in the abstract is done via Turing machines.[33] A Turing machine is a hypothetical computing device that is extremely simple to reason about; it consists only of a tape containing a grid of symbols, and a read head able to read and write symbols to the tape. The machine's

behaviour is controlled entirely by the symbol under the read head; based on the symbol the machine may write a new symbol in place, or move somewhere else on the tape. The usefulness of Turing machines for understanding computation is encoded in the Church-Turing thesis:

A function is efficiently calculable only if it is computable by a Turing machine

Informally, this assertion means that we only need concern ourselves with Turing machines, since other modes of computation are equivalent.

The computational resources a problem takes to solve is generally described using Big-O notation.[25] For a given problem, we say that it is of order $O(g)$ if the number of steps a Turing machine must take to solve as a function of the input size is bounded from above by g. For example, if a function f takes $3T^2 - T$ steps for T input then f is $O(T^2)$ (dropping constant factors and trailing terms). Equivalently we may say a Turing machine takes an amount of time proportional to T^2 to solve the problem by assuming each instruction carried out by the Turing machine takes a constant amount of time.

Computational problems are divided into *complexity classes*. Problems considered easy to solve are those which fall into class P (for Polynomial): those problems which are $O(n^d)$ for input sized n and some constant d; that is, problems which require a polynomial number of steps. Problems are considered hard to solve if they require an exponential number of steps, e.g. $O(2^n)$.

A special group of problems are those that fall into the *NP* class: these are decision problems (that is, problems whose output is either yes or no) for which verifying whether or not a candidate solution is correct is in P. Clearly, problems in P must also be in NP; if we can construct a solution in a polynomial amount of time then the time to verify is constant, which is in P. It is one of the greatest open questions of the modern era whether P = NP, or if there exist problems for which verifying the solution is easy (i.e. in P) but solving is hard (not in P).

It was shown in 1971 by Cook[7] that there are problems in NP which any other problem in NP can be reduced to with only at most a polynomial overhead in difficulty. As a result if such a problem were efficiently solvable then all NP problems would be (and thus P would equal NP). These are known as NP-Complete problems. Twenty-one such were famously demonstrated by Karp in 1972.[19]

Many problems of practical interest are known to be NP or NP-Complete. As such, methods to solve them quickly are of much use. The initial idea of using quantum mechanics to carry out computation is usually attributed to Feynman.[11] He noticed the difficulty in simulating quantum mechanics on classical computers due to the exponential resources required, and proposed using quantum mechanics to build a computer that could simulate physics. This lead to the development of quantum computing.

1.2 Quantum Computing

Quantum computing is usually described using the so-called gate model.[21] Some initial register of qubits is prepared, and then acted on sequentially by unitary operators known as quantum gates. This sequence of quantum gates results in the input register of qubits being put into some sort of quantum state, generally a mixture of superposition and entanglement. Finally the qubits are measured, resulting in a classical vector of information as output depending on which wave-function component is picked out by the measurement. The number of gates required to go from the specified input to the desired output maps analogously to the number of classical gates required for a classical digital computation, and in the same way that counting classical gates allows us to form big-O asymptotic bounds on computation time we can asymptotically bound the running time of a quantum algorithm by counting the number of unitary gates it requires.

Interest in quantum computers expanded with the discovery of an algorithm that was provably asymptotically faster than any classical algorithm: the Deutsch-Josza algorithm.[8] The Deutsch-Jozsa algorithm solves a highly contrived problem: given n input bits that are guaranteed to be one of

1) equally partitioned between $n/2$ zeros and $n/2$ ones (balanced)
2) contain all zeros (constant)
3) contain all ones (constant)

determine whether the input is balanced or constant. The fastest classical algorithm is to simply inspect half of the elements plus one, $n/2+1$ operations; meanwhile, the Deutsch-Josza algorithm can determine whether the input is balanced or constant in

only a constant number of operations. This problem is not very interesting on its own, but what *is* of interest is that a computer that can take advantage of quantum operations can solve it faster than a purely classical computer.

The discovery of a quantum algorithm asymptotically faster than the best classical one led to a surge of research, resulting in the discoveries of the Quantum Fourier Transform (QFT)[25], Shor's Algorithm[30] and Grover's Algorithm[13].

Building a quantum computer is a difficult task;[21][25] in order for the computer to remain well described by quantum mechanics, interaction with the outside world must be minimized otherwise the computational elements will become thermally mixed and no longer behave quantum mechanically. However, a computer must interact with the outside world to receive input and return output. This means that the computational section of a quantum computer cannot be too isolated. There exist quantum circuit elements that do *quantum error-correction*,[29] which can counteract the deleterious effect of thermal noise or other sources of decoherence. Unfortunately the larger a quantum circuit grows the more vulnerable to decoherence it becomes, and quantum error correcting schemes add qubits to the circuits which they error correct. These extra qubits bring in more decoherence; at present, more decoherence than the error-correcting codes fix.[25]

1.3 Outline

One proposed method for solving this runaway decoherence problem is *adiabatic quantum computing* (AQC).[10] AQC is essentially a form of analog computing, where a physical system is set up in such a way that as it evolves according to the laws of physics it naturally computes the solution to a problem. This dissertation will cover the background of AQC as well as explore the process of computing on an actual adiabatic quantum computer, the "D-Wave Two", including the requirements of compiling problems into a specific machine solvable form. We will also look at the experimental results of solving some actual problems on the D-Wave Two.

Chapter 2

Adiabatic Quantum Computing

In this chapter we will explore the development of AQC as well as the physics of the adiabatic theorem. We then discuss how adiabatic quantum computing works, and conduct a numerical simulation of an adiabatic quantum computation.

2.1 A Brief History of AQC

The idea of expanding simulated annealing to include quantum effects to solve problems was initially introduced by Kadowaki and Nishimori.[18] The linking of quantum annealing to quantum computation was done slightly later by Farhi et. al in 2000.[10] The structure of AQC appears here entirely intact: evolving from easily prepared initial Hamiltonians into problem Hamiltonians via the adiabatic theorem. Farhi et. al.[10] lacks any experimental or theoretical arguments for the speed of AQC, or comparison to gate model quantum computing. Farhi et. al. followed up with another paper[9] with some preliminary simulations of AQC carried out by numerically integrating Schrödinger's Equation. These results seemed to indicate exponential speedup over classical algorithms for an NP-Complete problem, but due to the difficulty of simulating quantum mechanics on a classical computer the maximum problem size they could experiment with was quite small so they could not draw very broad conclusions.

After this encouraging initial result it was shown by van Dam et. al.[34] that, at least in principle, AQC is equivalent to gate model quantum computing. Specifically they showed that adiabatic evolution taking time T may be approximated by $O(T^2)$ unitary transformations of n qubits. However, van Dam et. al. also designed a class of minimization problems for which AQC has an exponential lower bound. This was one of the earliest results in a controversy which continues to this day: how powerful are adiabatic quantum computers? Proposed answers have included: being *more* powerful than gate-model quantum computers, being able to solve NP-Complete problems efficiently;[28][24] *as* powerful as gate-model quantum computers;[34] or no

more powerful than classical computers.[27] Care must be taken to distinguish the questions of how powerful adiabatic quantum *computing* is, and how powerful a *given* adiabatic quantum computer is.

More papers on this topic followed, including several from A. P. Young with others.[16][37][36] Using Quantum Monte Carlo techniques, they were able to increase the size of analyzable Hamiltonians from 24 spins to 256 spins.[36] Their results seemed to indicate an exponential slowdown when solving a particular NP-Complete problem (Locked 1 in 3-SAT).[37] This would suggest that an adiabatic quantum computer would *not* be more powerful than a classical computer.

Concurrent with this effort to characterize adiabatic quantum computers theoretically was a commercial effort by D-Wave Systems to build one. Beginning with a single qubit[14], scaling to eight[15] and then 128[4] and 512.[26] While entanglement has been demonstrated to occur inside of this machine[22], the evidence for a quantum speedup is mixed,[26][3][31] with a very recent paper by Rønnow et. al. suggesting that the D-Wave Two (the 512 spin machine) is not asymptotically faster than a classical computer.[27]

In this work we carried out a variety of benchmarking on the D-Wave Two machine both to hopefully shed some light on the question of the speedup particular to this individual machine, and to validate our approach to problem compilation for AQC.

2.2 The Adiabatic Theorem

The central idea of Adiabatic Quantum Computation is solving problems by finding the ground states of specially constructed Hamiltonians.[10] Ground state relaxation is a form of analog computing where we rely on physics to do the calculating for us: we set up a physical system so that the ground state is the solution to our problem and let the system evolve. Generally speaking, we expect that classical or quantum systems will still take exponentially long to reach this desired ground state for NP-hard problems.[1] By starting the system in an easy to prepare state (for example, if your system was composed of spins then an easy to prepare state would be all spins ferromagnetically aligned) and then evolving adiabatically into our prepared problem state AQC uses the *adiabatic theorem* to (hopefully) avoid this exponential delay.

The adiabatic theorem states that if a system is initially prepared in the ith energy

level $|\psi_i\rangle$, and the Hamiltonian is evolved according to the adiabatic condition, the system will remain in the ith state after the evolution. The adiabatic condition is:

$$\left|\hbar\frac{\partial}{\partial t}\langle\psi_f|\hat{V}(t)|\psi_i\rangle\right| << \Delta^2 \tag{2.1}$$

where the energy gap between states $|\psi_i\rangle$ and $|\psi_f\rangle$ is $\Delta = |E_f - E_i|$ and $\hat{V}(t)$ is the time dependant part of the Hamiltonian. For a derivation of the adiabatic theorem see Appendix A. When the adiabatic condition is satisfied the probability of transition from state i to state f is zero, that is

$$|\langle\psi_f|\psi_i\rangle|^2 \to 1 \tag{2.2}$$

So if the speed at which the Hamiltonian changes (the derivative term) is slow enough, and the gap Δ between the initial state and the other states is large enough, then the system won't transition into a new state.

2.3 Finding a Problem Hamiltonian

While in principle there are an unlimited number of ways to construct a Hamiltonian whose ground state encodes the solution to a computation, our method is to use N 2-spin particles with a Hamiltonian:

$$\mathcal{H}_f = \sum_i h_i\sigma_i^Z + \sum_{i<j} J_{ij}\sigma_i^Z\sigma_j^Z \tag{2.3}$$

where σ_i^Z is the z-pauli matrix of the ith particle and h_i and J_{ij} are the parameters of the Hamiltonian; using spin as our computational basis we choose logical 1 (`true`) to be represented by spin up $|\uparrow\rangle$ (or $s = +1$) and logical 0 (`false`) by spin down $|\downarrow\rangle$ (or $s = -1$). We call the h_i parameters fields and the J_{ij}'s couplings. The parameters h_i control the tendency of the ith particle to point up or down, while the J_{ij}'s control the tendency of the ith and jth particles to align or anti-align.

We can find the energy of a state by multiplying each h_i by the corresponding spin $s_i = \pm 1$ and each J by the corresponding pair of spins s_i and s_j. Choosing a problem Hamiltonian is then the problem of finding the hs and Js so that the ground state corresponds to the problem we are trying to solve. For example, we can encode

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Fields	Couplings
$h_A = -1$	$J_{AB} = 1$
$h_B = -1$	$J_{AC} = 2$
$h_C = -2$	$J_{BC} = 2$

A	B	C	Energy
↓	↓	↓	9
↓	↓	↑	-3
↓	↑	↓	1
↓	↑	↑	-3
↑	↓	↓	1
↑	↓	↑	-3
↑	↑	↓	-3
↑	↑	↑	1

Table 2.1: From left to right the truth table of the logical `NAND`; the fields and couplings that result in a Hamiltonian satisfying that truth table; the complete energy structure of that Hamiltonian. Each state of the spins (↑ / ↓) whose logical equivalent appears in the truth table is part of the degenerate ground state of the final system.

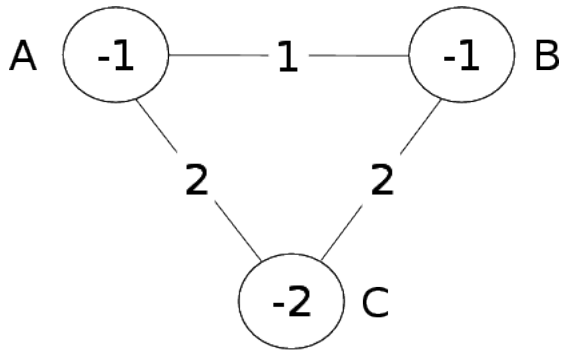

Figure 2.1: Graphical representation of the Hamiltonian implementing the logic of a `NAND` gate. Each of the vertices is a spin and the number inside is the field value on that spin; each edge is the coupling value between the respective vertices. Each ground state of this Hamiltonian enforces the logic $C = \neg(A \wedge B)$.

the logic of a digital `NAND` gate using the collection of h_i's and J_{ij}'s shown in Table 2.1 (the procedure for generating this Hamiltonian is described in Section 4.2). There are four degenerate ground states of this Hamiltonian, each corresponding to one of the entries in the logic table for a `NAND` gate.

Most of the time we don't want to calculate the whole truth table of a circuit, merely a subsection of it with some variables fixed. To do this we fix the variables we don't wish to solve for and adjust each of their neighbour's fields by the amount contributed by the couplings shared with the fixed variables. For example, if we had a Hamiltonian

Clamping Hamiltonian			
Fields		Couplings	
A	1.0	A, B	-1.0
B	1.0		

and we wanted to clamp B to -1, we would remove B from the Hamiltonian and the final result would be a field on A with magnitude 2.

2.4 Adiabatic Evolution

Once we have a Hamiltonian whose ground state encodes the problem we would like to solve, we need to set up the evolution. The Adiabatic Theorem lets us transition from the ground state of one Hamiltonian to another, so we need an initial Hamiltonian. In general almost any Hamiltonian of which we know the ground state will work. For simplicity we use the following initial Hamiltonian:

$$\mathcal{H}_i = \sum_i^N B\sigma_i^x \tag{2.4}$$

where B is some large constant. This Hamiltonian has as it's ground state all the spins pointed along the (negative) x-axis (or equivalently, an equal superposition of $|\pm z\rangle$). This is easy to prepare by applying a large magnetic field along the negative x direction. Once we've assembled both our initial and final Hamiltonians, we can conduct the evolution. We call the total Hamiltonian H_{tot}, and it is simply the sum of the initial and final Hamiltonian:

$$\mathcal{H}_{tot} = A(t)\mathcal{H}_i + B(t)\mathcal{H}_f \tag{2.5}$$

where $A(t)$ and $B(t)$ are dimensionless functions of time such that $A(0) = B(T) = 1$ and $A(T) = B(0) = 0$ where T is the final annealing time; this ensures that at $t = 0$ the Hamiltonian is equal to H_i and at $t = T$ the Hamiltonian is equal to H_f. Then the functions A and B describe the *evolution path*. Combining the evolution path with the speed at which the Hamiltonian is changing (or just the speed), that is $\frac{\partial A}{\partial t}$ and $\frac{\partial B}{\partial t}$, gives us the evolution trajectory.

Given a ground state $|psi_{gs}\rangle$ of $\mathcal{H}_{tot}$, the *fidelity* ($\mathcal{F}$) is the probability of measuring (after an anneal time T) ψ_{gs}, or $|\langle\psi(T)|\psi_{gs}\rangle|^2$. If there are N correct answer to our computation, and thus $|\psi_N\rangle$ degenerate ground states, the fidelity is defined as

$$\mathcal{F} = \sum_{i=0}^{N} |\langle\psi(T)|\psi_i\rangle|^2 \tag{2.6}$$

For a perfectly annealed system, the fidelity should go to one as the annealing time increases. Since any real annealing will not be in the infinite anneal time limit, in practice our fidelity values are restricted to be ≤ 1. Hence AQC is fundamentally a probabilistic process. The closer the fidelity is to 1, the more likely our computation is to succeed. We will use the fidelity later in the experimental results section to evaluate how close we get to ideal AQC.

For a purely quantum mechanical system described by the Hamiltonian $\mathcal{H}_{tot}$, the evolution trajectory completely determines the fidelity. The simplest and most obvious trajectory is $A(t) = 1 - t/T$ and $B(t) = t/T$ at a constant speed, or a straight line path through Hamiltonian space. There is no particular reason that a straight line from $(A = 1, B = 0)$ to $(A = 0, B = 1)$ should maximize the fidelity; indeed, because moving away from the origin in Hamiltonian space scales the Hamiltonian and its derived quantities up (including the energy gap), and the adiabatic condition is stated in terms of the energy gap, we expect that some more complicated curve through Hamiltonian space, potentially including changing the evolution speed, will give the best results.

However, in this work we restrict ourselves to evaluating straight line paths through Hamiltonian space due to the D-Wave Two being limited in this manner (see Chapter 3).

2.5 Simulating AQC

For small Hamiltonian we can simulate directly the process of quantum annealing by integrating the Schrödinger equation. For time dependent Hamiltonians such as ours, the general form of the Schrödinger equation is

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \mathcal{H}(t)|\psi(t)\rangle \tag{2.7}$$

this is a system of linear ordinary differential equations which can be solved by numerical methods. Using a fourth order Runge-Kutta method[12] we can find the overlap between the ground state $|\psi_f\rangle$ and the current state $|\psi(t)\rangle$ and plot this as a function of the annealing parameter. We anneal an eight spin Hamiltonian ("k44_and", see Table 6.2) with linear control parameters $A(t) = 1 - t/T$ and $B(t) = t/T$. The state vector $|\psi(t)\rangle$ is written as a superposition over all the possible states

$$|\psi(t)\rangle = \sum_{i=0}^{2^8-1} c_i(t)\,|i\rangle \tag{2.8}$$

where $|i\rangle$ is a shorthand for

$$\begin{aligned} |0\rangle &= |\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\rangle \\ |1\rangle &= |\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\downarrow\rangle \\ &\vdots \\ |255\rangle &= |\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\rangle \end{aligned} \tag{2.9}$$

where each $c_i(0) = 1/16$ (that is, an equal superposition) and we solve for all of the $c_i(t)$s using Equation 2.7 to generate each time step $c_i(t + \Delta t)$

Figure 2.2 shows the results of this simulation. The features correspond to what we expect from the adiabatic theorem: the fidelity starts low and climbs as the system evolves. It does not reach one as we are not in the adiabatic limit because we are annealing for only a finite time. We expect to see similar results to the one shown here from experimental data; deviations will show us how far our experiments depart from the assumptions of AQC.

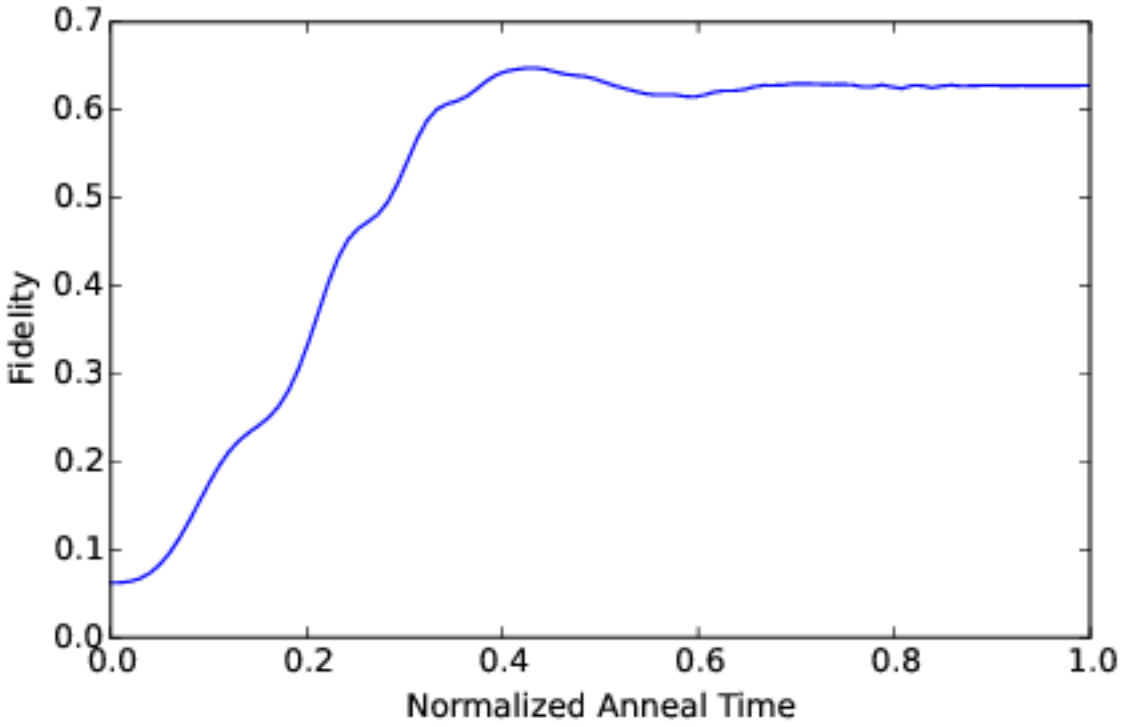

Figure 2.2: Numerical simulation of adiabatic evolution by integrating the Schrödinger equation.

Chapter 3

The D-Wave Two Machine

3.1 Machine Description

The D-Wave Two machine on which our experiments were conducted is composed of 512 rf-SQUID superconducting qubits of the type described in [14] [17]. It is the same physical machine as the larger of the two machines described in [26].

Each qubit consists of a super-conducting Josephson junction, with left and right circulating currents serving the roles of $\uparrow/\downarrow$ or logical 1/0. A super-conducting flux qubit is essentially a LC oscillator circuit (the flux Φ and the charge Q have the commutator $[\Phi, Q] = i\hbar$ leading to harmonic oscillator behaviour) with an additional thin insulating layer inserted into the circuit; cooper pairs can tunnel through the insulating layer. In general, Josephson-junction qubit circuits have contributions from inductance, capacitance and current, with the following potential:[21]

$$\mathcal{U}(\Phi) = E_J \left[1 - \cos\left(2\pi \frac{\Phi_{ex} - \Phi}{\Phi_0}\right)\right] + \frac{\Phi^2}{2L} \tag{3.1}$$

where Φ_0 is the flux quantum ($h/2e$), E_J is the Josephson energy, L is the inductance of the loop, and Φ_{ex} is the bias flux. In our case the D-Wave Two machine uses flux qubits, where the bias flux is roughly equal to the flux quantum: $\Phi_{ex} \approx \Phi_0/2$. This causes two wells of equal energy, as seen in Figure 3.1.

Figure 3.1 shows both a diagram of a qubit and a corresponding coupler, and also the energy structure of the Josephson junction. Each well of the double well potential corresponds to one of left or right circulating current. This can be modelled as a two-level system, although in reality there are multiple other energy levels both in the wells and above them and above the wells themselves. Qubits occupying higher energy levels within the wells is generally not a problem, because the machine measures magnetization at the end and measuring a state higher in a well will produce the same net result.[14] Qubits occupying even higher energy states (delocalized across

both wells) will result in incorrect results, and the method for dealing with this is to run the machine as cold as possible to reduce the likelihood of thermal excitation to those states. Because AQC is statistical even in the absence of machine failures of this kind (because we never reach the adiabatic limit), these higher excited states are not a major weakness.

3.2 Programming the D-Wave Two

Each qubit and coupler is attached to another SQUID called a Φ-DAC which allows them to be programmed. The Φ-DACs are controlled by classical circuitry and according to the binary input they are given induce a flux quantum in their superconducting loops. A Φ-DAC attached to a qubit controls the field term h_i on the qubit by modifying the bias flux Φ_{ex}, while a Φ-DAC attached to a coupler controls the coupling between two qubits by scaling the strength of the qubit-coupler-qubit interaction according to the number of flux quanta in the Φ-DAC. The Φ-DACs on the D-Wave Two are capable of containing between zero and seven flux quanta of either polarization, giving 4-bit precision for both the fields and couplings. For more detail about the design of the programming hardware see Bunyk et. al.[5]

3.3 Resolution

As discussed in Section 3.2, the D-Wave Two machine cannot implement arbitrary physical Hamiltonians. The machine can only implement fields and couplings as one of 15 distinct values: $-7/7, -6/7 \ldots 5/7, 6/7, 7/7$ corresponding to the number of flux quanta inside each Φ-DAC. Call the largest coupling present in the problem Hamiltonian C_{max}, and the smallest *magnitude* coupling C_{min}. As our Hamiltonians don't generally have a C_{max} to C_{min} distance of exactly 7, the machine must manipulate them to fit in this range. First all the field and coupling values are normalized into the range $[-1..1]$, and then each coupling is coerced to the nearest machine implemented value. Unfortunately we don't know exactly how the coercion is done, because D-Wave has not released much information about the classical software side of the D-Wave Two . It seems reasonable that each normalized coupling would be rounded

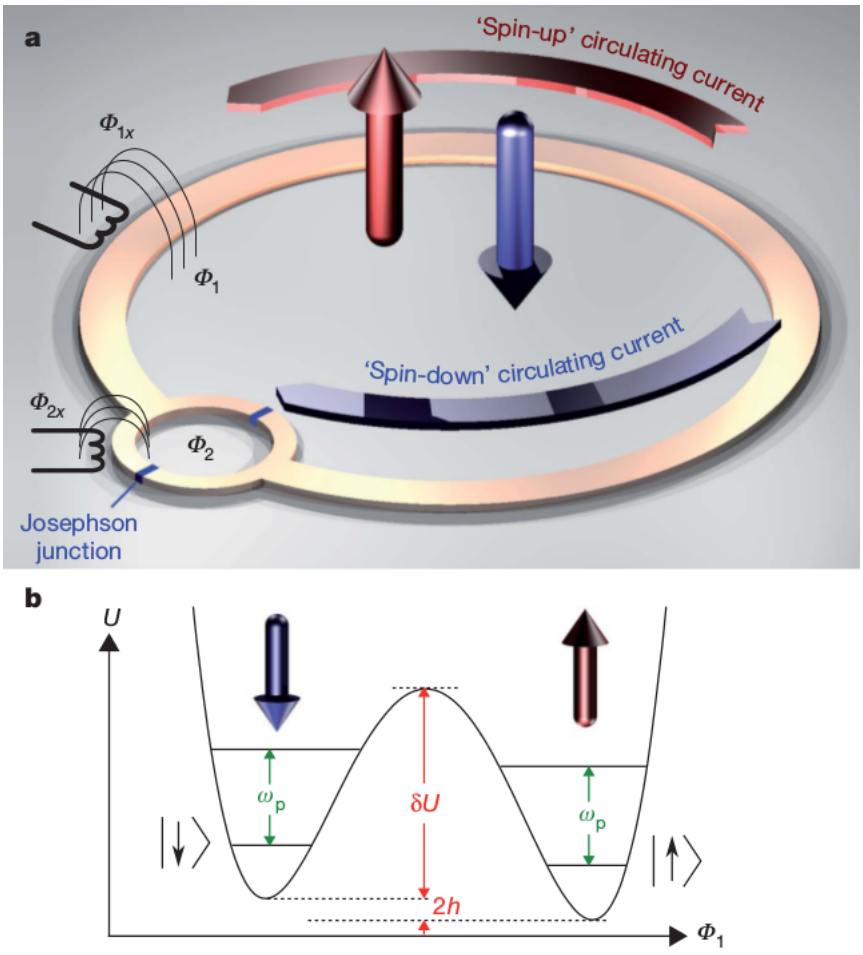

Figure 3.1: Inset a) shows the structure of a single qubit Josephson Junction along with an inductive coupler. Inset b) shows the energy of one of these qubits as a function of the internal flux ϕ_1; each of the double wells is a computational basis state. Figure taken from [14]

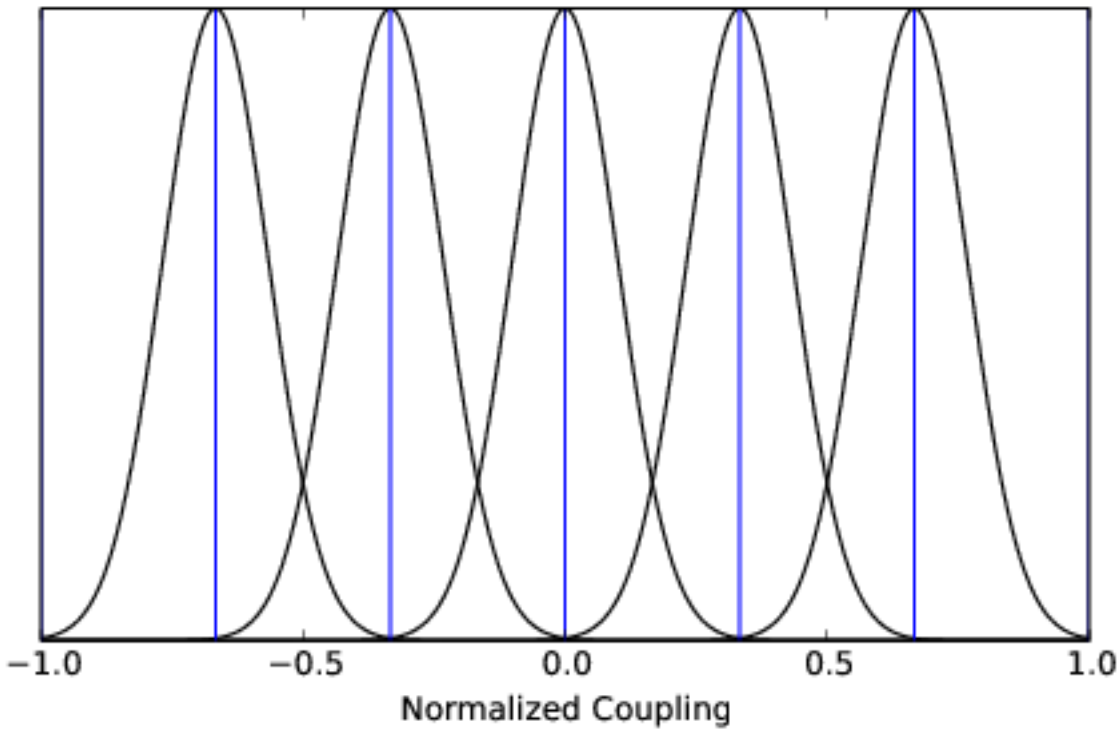

Figure 3.2: This figure illustrates the difference between ideal and physically implemented couplings; the blue δ-function like peaks are what ideal couplings would behave as, and the Gaussians what a real machine must implement. The amount of overlap between the tails of the real couplings is dependent on the magnitude of the programming noise in the machine; the actual amount of overlap present in the D-Wave Two machine is unknown.

to the nearest machine implemented coupling, but other schemes such as rounding toward or away from zero are possible. The linear spacing of the machine implemented couplings implies that Hamiltonians with couplings that linearly fill the range from $-C_{max}$ to C_{max} will be least affected by the coercing issue, while those Hamiltonians which have gaps between coupling values or other spacings will be most impacted by the effects of finite machine resolution.

3.4 Annealing Schedule

As previously stated in Chapter 2 the form of the Hamiltonian we use for AQC is

$$\mathcal{H}_{tot} = A(t)\mathcal{H}_i + B(t)\mathcal{H}_f \tag{3.2}$$

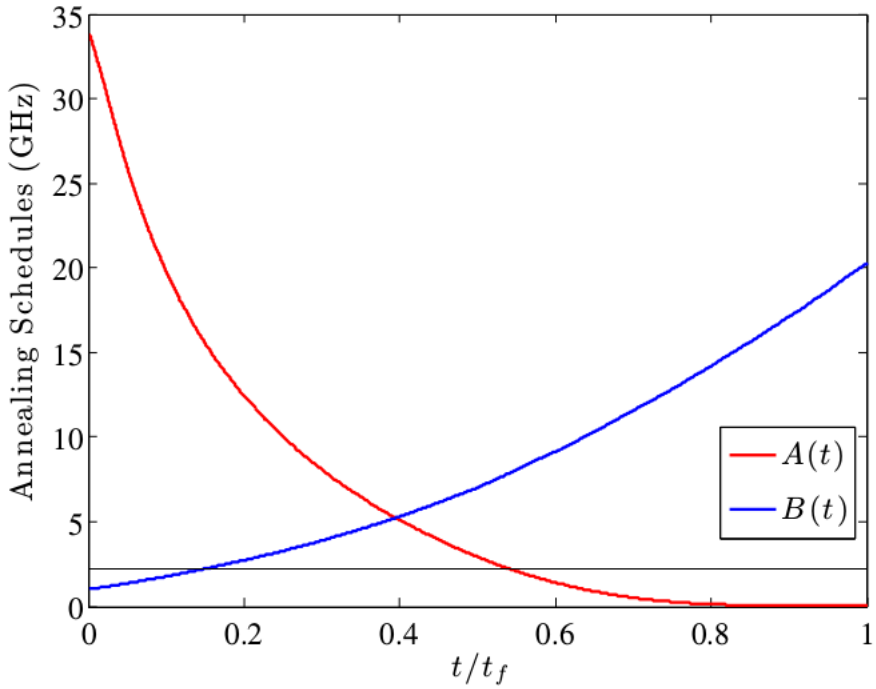

Figure 3.3: Plot of the two control parameters $A(t)$ and $B(t)$ for the D-Wave Two machine as a function of the annealing time, as well as the temperature energy scale. Figure from [26]

and in our case the experiments were carried out with a transverse field of $A(0) \approx 33.8$ GHz and a maximum normalized coupling value of $B(T) \approx 20.5$ GHz, and an annealing temperature of 17 mK. These correspond to energies of 140 μeV and 85 μeV for the Hamiltonian components respectively and a thermal energy of 1.5 μeV. The particular values of A and B as a function of time are shown in Figure 3.3.

3.5 Programming Noise

The resolution issue is exacerbated by the fact that there is some error in the programming of each coupling (or field). Ideally one could imagine the physical coupling values implemented by the machine as a series of δ-functions from $-C_{max}$ to C_{max}; however, because the couplings are non-ideal, the actual structure of the couplings looks more like Figure 3.2. These errors arise because each of the SQUID elements making up the machine are subject to physical flaws; the qubits, the couplers and the programming elements (Φ-DACs). In addition there are errors that occur each time the machine is programmed, coming from stray inductance or thermal noise or

and other things. The magnitude of the resulting programming error is uncertain; efforts are underway to quantify how much error is consistent with results from the machine [35]. If the overlap between adjacent programmable coupling values is large enough, then the Hamiltonian which is actually implemented during a particular run may have a different ground state than what we wished to program in; the larger the overlap, the more likely to get incorrect ground states. The preliminary results in Chapter 6.1 suggest that the programming noise is in the neighbourhood of the machine coupling spacing, or large enough that it is possible for the programming noise to cause incorrect ground states.

3.6 Inoperable Qubits

Not every qubit on the D-Wave Two machine is functional all of the time. As the machine is cooled to super-conducting temperatures, lines of magnetic flux can become trapped inside. As some parts cool past the critical temperature before others, the Meissner effect forces magnetic flux out of the super-conducting regions into the regions which are still above the critical temperature. If the regions of trapped flux are surrounded by super-conducting regions, there is no way for the trapped flux to escape and so it remains inside the machine, hindering the operation of whichever qubits it intersects. As in the normal course of operation the machine is brought above super-conducting temperatures only for maintenance, the pattern of broken qubits can remain stable for long periods of time. Figure 3.4 shows the layout of the D-Wave Two machine, with the qubits that were inoperable for the duration of this study marked in red.

The presence of such inoperable qubits lowers the size of the Hamiltonians we can embed more than is at first obvious. Even though the number of inoperable qubits is fairly small, the embedding algorithm (see Chapter 4) requires many qubits to embed the connections in a given Hamiltonian. As a result, in the case where there are 9 disabled qubits the largest embeddable complete graph shrinks from 32 spins to 29, a 10% decrease in size for losing only 2% of the spins. This demonstrates how reliant on the connectivity of the hardware graph the embedding process is.

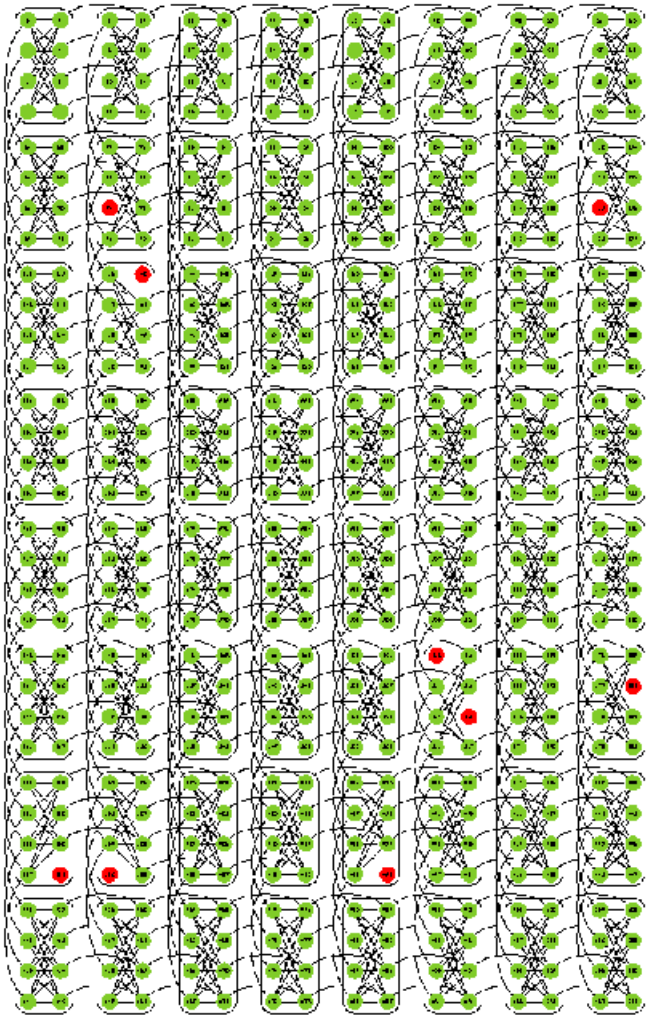

Figure 3.4: Diagram of the connectivity of the D-Wave Two machine showing the Chimera graph structure, with the qubits which were inoperable during the duration of this study marked in red.

Chapter 4

Embedding

In this chapter we will cover the various processes by which we go from general Hamiltonians like the ones from Chapter 2 to Hamiltonians which will fit on the D-Wave Two as described in Chapter 3.

4.1 Creating Problem Hamiltonians

Embedding is the name we give to the general process of translating a computational problem into something which is ready for an adiabatic quantum computer to evaluate. As mentioned in Section 2.3, the first step is to generate a problem Hamiltonian. We employ two methods for finding suitable problem Hamiltonians. The first, using linear programming, is suitable for small Hamiltonians (generally only a few variables in size). The second is to use the software `QCC` which takes advantage of the gluing theorem to stitch together a problem Hamiltonian out of smaller sub-Hamiltonians. The second step in the embedding process is to translate our problem Hamiltonians, which generally may have arbitrary topologies, coupling values, number of variables etc. into Hamiltonians which the D-Wave Two is capable of physically implementing.

These are not the only methods of generating problem Hamiltonians. The programmatic style described above seems natural for solving actual computational problems, but e.g. if one wanted to go a more academic computer science route one could reduce one's problems to known NP-Complete problems and evaluate those. Lucas [23] handily provides a compilation of ising formulations of a variety of different NP-Complete suitable for such an approach.

4.2 Linear Programming

The requirements we have for a problem Hamiltonian can be described using a system of constraints, and solved using the techniques of linear programming (LP). Briefly,

LP is a method to optimize some objective function subject to linear equality and linear inequality constraints. There are three standard parts that describe a linear programming problem:

- A linear function to be maximized or minimized

  $f(x_0, x_1) = c_0 x_0 + c_1 x_1$

- Constraint Matrix A:

  $A\vec{x} \leq \vec{b}$

- Non-negative variables

  $x_i \geq 0 \ \forall i$

(thus the name *linear* programming). There are many software packages that implement LP solving, both Free Software and proprietary. This makes it an easily accessible technique.

To use linear programming to create a problem Hamiltonian with N spins we first assign a variable x_i for each field and each coupling that could be in the end result; this is $N(N-1)/2 + N$ or $N(N+1)/2$ (N choose $2 + N$ variables). Then we add a constraint for each possible assignment of the spins, making 2^N constraints. In these constraints we have every sign combination of the variables that correspond to fields, e.g.

$$\begin{aligned} c_0 &= x_0 + x_1 + x_2 \\ c_1 &= x_0 + x_1 - x_2 \\ c_2 &= x_0 - x_1 + x_2 \\ c_3 &= x_0 - x_2 - x_2 \\ &\text{etc.} \end{aligned} \tag{4.1}$$

In each constraint we also add the variables representing couplings; the sign of each coupling variable should be the product of the two variables that represent the two spins the coupling connects. Now for each state which we want to be part of the ground state we set the value of the constraint equal to k; for each state that is

not part of the ground state, we enforce that the corresponding constraint is strictly greater than k.

For example, if we know the truth table for a `NAND` gate and want to find a set of fields and couplings that create a matching Hamiltonian, the constraints would be

$$\begin{aligned}
c_0 &= -x_0 - x_1 - x_2 + x_3 + x_4 + x_5 + k && \geq 1 \\
c_1 &= -x_0 - x_1 + x_2 + x_3 - x_4 - x_5 + k && = 0 \\
c_2 &= -x_0 + x_1 - x_2 - x_3 + x_4 - x_5 + k && \geq 1 \\
c_3 &= -x_0 + x_1 + x_2 - x_3 - x_4 + x_5 + k && = 0 \\
c_4 &= x_0 - x_1 - x_2 - x_3 - x_4 + x_5 + k && \geq 1 \\
c_5 &= x_0 - x_1 + x_2 - x_3 + x_4 - x_5 + k && = 0 \\
c_6 &= x_0 + x_1 - x_2 + x_3 - x_4 - x_5 + k && = 0 \\
c_7 &= x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + k && \geq 1
\end{aligned} \tag{4.2}$$

For all three spin gates the constraints would be identical except for which states are $k = 0$ and which are $k \geq 1$. This particular set of constraints will produce a `NAND` gate; each of the variables x_0, x_1, x_2 in the constraint equal to zero matches the truth table (see Table 2.1) for `AND` with input variables x_0, x_1 and output variable x_2.

The combined effect of all of these constraints is that when we solve the system for the set of variables x_i which minimize the objective function, states which encode the desired logic will have an energy of $-k$ and states which do not encode the desired logic will have an energy greater than $-k$. This is a slightly unusual use for LP; normally finding the set of x_i's that optimize the objective function is the goal, but for us finding the x_i's which satisfy the constraints if the central objective. Minimizing the sum of the coupling values simply helps to keep the number of different coupling values small and the spread between them down. This will be important because physical adiabatic quantum computers can only implement a finite number of different coupling values in finite ranges.

Unfortunately while this method allows us to find a set of couplings to encode the logic of any circuit, we require a constraint for each possible state of each of the variables. For a binary circuit with n variables, there are 2^n states. The above process for creating problem Hamiltonians is thus clearly exponentially hard, and becomes

extremely computationally inefficient for larger circuit sizes. That said, this method retains utility for finding small sub-Hamiltonians to use as building blocks for larger problem Hamiltonians using `QCC` as described in the next section.

4.3 `QCC`

`QCC` is a collection of programs and libraries designed to produce problem Hamiltonians. The core of `QCC` are two pieces: the `QSM` language and interpreter, and the embedding algorithm. The `QSM` language is a programming language designed to simplify the connecting of sub-Hamiltonians. It is both human writable and serves as a compilation target for more complicated or high level languages. One can write a `QSM` file describing a circuit composed of multiple sub-Hamiltonians, and the `QSM` interpreter will construct the appropriate problem Hamiltonian, including any auxiliary spins that the sub-Hamiltonians need. Once the problem Hamiltonian has been generated, a Hamiltonian with equivalent ground states that is isomorphic to a sub-graph of a particular hardware implementation must be created in order for the problem Hamiltonian to be solved; this is the job of the embedding algorithm.

4.4 `QSM` Language

The `QSM` language is a simple language designed to be readable and writable by both humans and machines. A `QSM` file consists of one or more `QSM` statements; each `QSM` statement is enclosed by parentheses and consists of a whitespace separated list of the statement components. If a component has multiple sub-parts then they are also enclosed in parentheses. There are two kinds of `QSM` statements, gate statements and clamp statements.

A typical `QSM` gate statement consists of three components: first the name of the gate or sub-Hamiltonian to be included, then a parenthesis-enclosed list of the input spins for the gate, and third a parenthesis-enclosed list of the output spins for the gate. An example `QSM` gate statement is:

```
(and (a b) (c))
```

This `QSM` statement encodes the logic $A \wedge B = C$. A variable name may be prefixed by a $\sim$, in which case the logic of that variable is negated. The other type of `QSM`

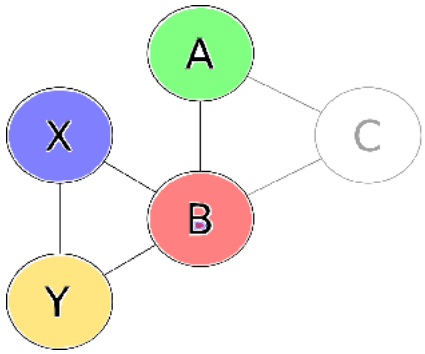

Figure 4.1: The shape of an example Hamiltonian as would be created by `QCC`. This graph will not fit on the D-Wave Two , so in the next section we will demonstrate the embedding algorithm on it.

statement besides a gate statement is the clamp statement; this statement consists of two components. First the keyword `clamp`, followed by a list of variables and the values that they are to be clamped to. A clamp statement looks like

```
(clamp ((a +1) (b -1)))
```

The above clamp statement sets the value of the variables `a` and `b` to +1 and -1, respectively.

A complete `QSM` file might then look like

```
(and (A ~B) (C))
(or (X Y) (B))
(clamp ((C +1)))
```

which would compile to a Hamiltonian which encodes the logic $A \wedge \neg B = C, X \vee Y = B, C =$ TRUE and the resulting ground states will be those states for which the logic is satisfied.

Figure 4.1 shows the shape of the resulting graph. The node for C is greyed out to represent the fact that it is clamped. Each of the input sub-Hamiltonians (`and` and `or`) connect each of the spins they're given, and the resulting shape has node B connected to every other node, X and Y connected to each other, and A only connected to B. This Hamiltonian graph does not have the same shape as the D-Wave Two graph, and so will have to be embedded in the following section.

4.5 Embedding algorithm

Once we have our problem Hamiltonians we need to convert them into Hamiltonians which have the same shape as the graph that the D-Wave Two machine has. This is done by adding *clone spins* into the graph: each clone spin has the same value in the ground state as one of the logical spins in the input graph. This allows us to decrease the connectivity of the problem Hamiltonian until it is sparse enough to be the same shape as the physical machine. The following algorithm is essentially equivalent to one presented in [6]. We expect that translating complete graphs onto graphs with constant connectivity to require roughly a quadratic size penalty in the number of clone spins (as the number of connections in a complete graph is $\sim n^2$). Ideally, to ensure that each clone spin's ground state is the same as it's parent we would have $J_{ij} << J_{clone} \forall i, j$. Because of physical limits of the machine (especially the resolution mentioned in Section 3.3) large values of J_{clone} may not maximize fidelity: for a machine with finite resolution, having a large J_{clone} may compress other coupling values together such that the ground state starts to overlap with higher energy state. On the other hand a too small value of J_{clone} may result in ground states in which the clones are not aligned, which will almost certainly also produce incorrect ground states. Thus we have to find an empirical value of J_{clone} that maximizes the fidelity while preserving the ground states.

Figure 4.2 shows a demonstration of the embedding algorithm. The main idea is to assign nodes from the Hamiltonian you are trying to embed along the diagonal of the D-Wave Two graph. Then link up connected nodes by cloning right and up from the nodes that need to be connected.

The embedding algorithm is as follows:

Definitions:

Designate the input Hamiltonian graph V and the output graph G. Label the spins in V and G as V_i and G_i respectively. We define the *clone map* C_i as the set of spins $[i, j \ldots]$ which have the same logical value as their parent spin: so for example the clone map of spin 5, C_5, might be $[2, 3, 13]$ which would mean that spins 2, 3, 5 and 13 all share the same logical value. We define a mapping M between logical spins in graph V and computation spins in graph G. For example, spin V_{12} representing the 12th variable in a problem might be mapped to G_{187}, the 187th machine spin.

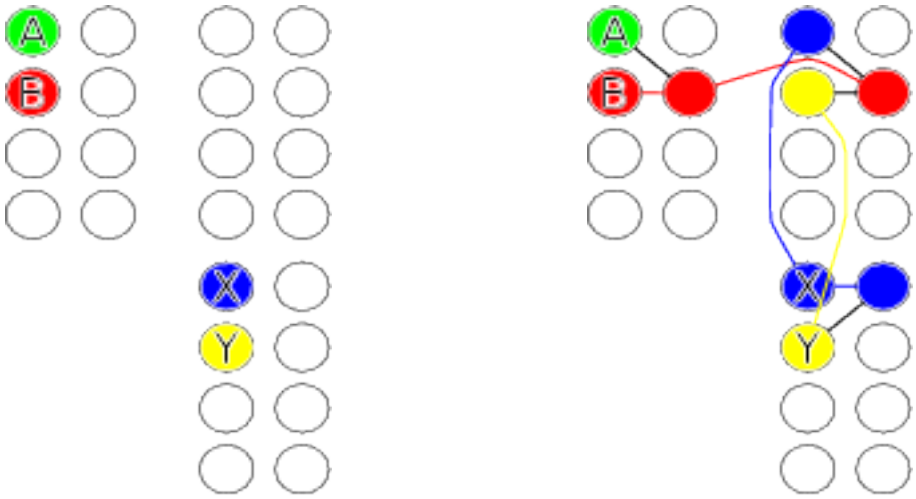

Figure 4.2: This figure shows two steps in the embedding process. On the left is the first step, assigning elements of the input graph (Figure 4.1) along the diagonal of the D-Wave Two graph. On the right is the connection step; nodes with the same colour share logical values. Coloured lines represent the *clone couplings*, while black lines represent the computational couplings.

We also define a *clone coupling* value which is as large as possible and ferromagnetic; this attempts to ensure that all the members of a clone map are aligned in the ground state. In the final G, each member of a clone map should have a clone coupling connecting it to at least one other member of the clone map.

Procedure:

1. Populate M by mapping each V_i to one of the spins G_j on the left side of a unit cell that lies along the diagonal of G

2. For each field term in V, add a field to G on the corresponding spin

3. For each J_{ij} in V, conduct the following operation:

    - Scan through both of C_i and C_j to find the pair which are nearest to each other in G; call these x and y
    - Get a list of each spin that lies along the path between x and y
    - Assign half of these spins to C_i and half to C_j; add the appropriate clone coupling into G for each spin in the path to ensure that the clone map is properly connected

- Finally, at the interface between the two new clone map members, add a coupling into G with the same value as J_{ij}

Each term in V, both fields and couplings, should now have a corresponding term in G. G should also contain many coupling terms that group the necessary clone maps.

This algorithm is tailored to AQC machines whose qubits are arranged in a square shape (specifically the shape of the D-Wave Two), however, the principle should be generalizable: as mentioned above any reasonably compact graph with constant connectivity should be able to embed a complete graph with an n^2 overhead by allocating the input graph spins along the diagonal.

Chapter 5

Boolean Satisfiability

In this chapter we describe the NP-Complete problem *boolean satisfiability* (SAT) and the techniques needed to use it as a problem for D-Wave Two benchmarking. We then build on the previous chapters and take a SAT problem from conception to embedded Hamiltonian ready for evaluation on the D-Wave Two .

5.1 Satisfiability Problems

Boolean satisfiability is the problem of determining if there exists an assignment of boolean variables to satisfy a given boolean formula. A boolean formula is *satisfiable* if there is some assignment of the variables within it that causes the whole expression to evaluate to `true`, and is *unsatisfiable* if there is no such assignment. A boolean expression in conjunctive normal form consists of the conjunction of a set of clauses, each clause consisting of the disjunction of some number of variables (or their negations). When each clause has k variables, then the problem of finding a satisfying assignment is called k-SAT. An example 3-SAT expression might look like the following

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \tag{5.1}$$

which has five satisfying assignments:

	Assignments				
x_1	T	T	T	F	F
x_2	F	F	T	T	F
x_3	F	T	F	T	F

k-SAT is NP-Complete[7] for $k > 2$, and is interesting because it was the first and is one of the most general NP-Complete problems; it is generally very easy to

find reductions from other problems into boolean satisfiability. For this reason we use boolean satisfiability (specifically 3-SAT) for the testing of the D-Wave Two machine.

5.2 Randomly Generating 3-SAT Problems

We can build 3-SAT instances with N variables and M clauses by randomly selecting three elements from the set of N variables and their negations M times. Care must be taken when choosing N and M however; while 3-SAT is NP-Complete in general, the difficulty of any particular instance depends strongly on the ratio $\alpha = M/N$[20]. In the small α regime there are many satisfying assignments, and any algorithm to find a solution will likely stumble on one by chance. Meanwhile in the large α regime there are likely to be *no* satisfying assignments.

As discussed in Kirkpatrick and Selman[20], there is a phase transition between the many solution and zero solution regimes, which occurs around the value of α for which there is likely to be one solution. This is the region where finding a satisfying assignment (or proving there is none) is the hardest, and helps to keep us from falsely believing our algorithm is better than it is due to too easy SATs. This transition is sharper for larger values of N or M, and broader for smaller values. As the value of N and M that will fit onto the D-Wave Two machine are very small, we don't expect to see a very sharp difference for different values of α. Kirkpatrick and Selman report that for 3-SAT, the transition occurs at $\alpha \sim 4.17$. Therefore we shall attempt to generate SATS which lie as close to this transition as possible; in practice this usually means around 4.25 ± 0.25 due to the fact that N and M must be integers and the size of our hardware graph restricts us to small values of N and M.

5.3 Embedding 3-SAT

Embedding 3-SAT problems requires two different uses for the qubits: each variable in the SAT requires a qubit, and each clause also requires a qubit. This is because we embed each clause as a `3OR` gate, a 4 spin gate where the fourth spin's logical value is the disjunction of the three input spins. This gate requires an auxiliary spin beyond the computational spins to encode its logic. Because we know that for a satisfying statement each clause must be satisfied, clamping each clause's output

Example SAT Hamiltonian			
Fields		Couplings	
x1	-1.0	x1,x2	-1.0
x2	1.0	x1,aux	-2.0
x3	-3.0	x2,aux	2.0
aux	-1.0	x3,aux	1.0

Table 5.1: The fields and couplings for the example Hamiltonian we are creating to solve the SAT problem in Equation 5.2.

variable completes the 3-SAT `QSM` construction. As we have roughly four times as many clauses as variables, each clause shares its variables with many other clauses. This means that the resulting graph is reasonably close to being a complete graph, so we embed them as if they were and tolerate some inefficiency.

Since the largest complete graph the hardware graph can embed is 32 spins, the largest SAT problem we could embed at an α close to the phase transition is 6 variables and 25 clauses for an alpha of 4.17; however, as discussed in Section 3.6, the capability of the D-Wave Two machine during our experiments was actually 29 spins, which is too small for 6 SAT variables plus 25 clauses. The actual largest SAT embedded was thus 6 SAT variables and 21 clauses.

5.4 Example SAT embedding

Take a simple single clause 3-SAT problem:

$$(x_1 \wedge \neg x_2 \wedge x_3) \tag{5.2}$$

for which we want to find a satisfying assignment. We can translate this SAT problem into a `QCC` program:

```
(3and (x1 ~x2 x3) (output))
(clamp ((output +1)))
```

compiling this program with `QCC` yields the Hamiltonian described in Table 5.1. This Hamiltonian requires an extra qubit which we label 'aux' beyond the three SAT variables.

Example SAT Hamiltonian			
Fields		Couplings	
0	-1.0	0,4	-3.0
1	-1.0	1,4	-2.0
2	1.0	1,5	-3.0
3	-3.0	2,4	2.0
4	0.0	2,5	-1.0
5	0.0	2,6	-3.0
6	0.0	3,4	1.0
7	0.0	3,7	-3.0

Table 5.2: The fields and couplings for the embedded version of the example Hamiltonian 5.1. Where aux is mapped to 0, x1 $\rightarrow$ 1, x2 $\rightarrow$ 2, x3 $\rightarrow$ 3 and the variables 4..7 are clones.

The shape of this Hamiltonian is the same as the one shown in Figure 4.1, so it will not fit on the machine without being embedded. We call the embedding routine in `QCC` and get the resulting Hamiltonian shown in Table 5.2. The Hamiltonian now takes up 8 qubits due to the embedding process trading connectivity for space.

If we upload this Hamiltonian into the D-Wave Two , we would find that there is a single ground state

$$\begin{array}{ll} 0 & \uparrow \\ 1 & \uparrow \\ 2 & \downarrow \\ 3 & \uparrow \\ 4 & \uparrow \\ 5 & \uparrow \\ 6 & \downarrow \\ 7 & \uparrow \end{array}$$

which we translate into the variables $x_1 =$ `true`, $x_2 =$ `false`, $x_3 =$ `true`, which satisfies the original SAT problem.

Chapter 6

Results and Discussion

6.1 Running procedure

The D-Wave Two features a classical software interface which takes text files containing the information of a Hamiltonian like Table 5.2. After this is uploaded the machine anneals the Hamiltonian as described in Chapter 3 and returns the values of the spins it found after annealing.

Each time the machine loads a Hamiltonian it programs in the fields and couplings as described in Section 3.2, it can evaluate the Hamiltonian many times. Each time the D-Wave Two loads in a Hamiltonian we call a *run*, while each repetition of the same Hamiltonian without loading we call a *read*. Any programming error (see 3.5) in the Hamiltonian should be the same between reads and differ between runs.

The state read out after each read is either the ground state, or some other higher energy (and incorrect) state. The successes and failures thus follow a binomial distribution; there is some probability p of getting the ground state, and probability $1 - p$ of getting a different state. Our best estimate of the fidelity from a single run is thus the fraction of successes, or $p = \frac{gs}{n}$ for gs reads of the ground state and n total reads. We can also estimate the error in our estimate of the fidelity, because the variance of a binomial distribution is $\sigma^2 = np(1 - p)$.

6.2 Single Qubyte Hamiltonian

The first Hamiltonian evaluated was very similar to the Hamiltonian described in Table 5.2 to encode a single clause SAT problem. The Hamiltonian "k44" is shown in table 6.1, consisting of a single qubyte of spins and encoding the logic of a single SAT clause was evaluated in with runs at anneal times from 20 μs to 500 μs with 1000 reads per run. One thousand repetitions at a given anneal time will give us good statistics to estimate the fidelity.

k44 Hamiltonian			
Fields		Couplings	
A	1.0	A,B	1.0
B	1.0	A,D	-2.0
C	-1.0	B,D	-2.0
D	-3.0	C,D	1.0

Table 6.1: The fields and couplings making up the Hamiltonian "k44".

k44_and Hamiltonian			
Fields		Couplings	
A	1.0	A,B	1.0
B	-1.0	A,D	-2.0
C	3.0	B,D	-2.0
D	-1.0	C,D	-1.0

Table 6.2: The fields and couplings that make up the Hamiltonian "k44_and". This single qubyte Hamiltonian has a single non-degenerate ground state: $|\uparrow\uparrow\downarrow\uparrow\rangle$

Figure 6.1 shows the results of two such sweeps. The fidelity was estimated as described in Section 6.1. First, contrary to our expectations, the fidelity *decreases* as the anneal time increases. We would expect that longer anneal times bring us closer to adiabatic evolution and thus remaining in the ground state. This result suggests to us that adiabaticity (or lack thereof) is not dominating the fidelity for long annealing times. Second, there is significant change in the fidelity over very small changes in anneal time for less than ~ 300 μs, well outside what we would statistically expect based on the number of reads conducted if the cause is simple statistical noise. This suggests that there is something going on besides the annealing time that determines the fidelity when the anneal time is short. This factor disappears for longer anneal times, when whatever is damping the fidelity also seems to dampen the short time oscillations.

6.3 Non-degenerate Single Qubyte Hamiltonian

It was thought initially that the reason for the variability of the fidelity at short anneal times was due to the degeneracy of the "k44" Hamiltonian ground states. The Hamiltonian "k44_and" was created to be similar in form (same number of qubits)

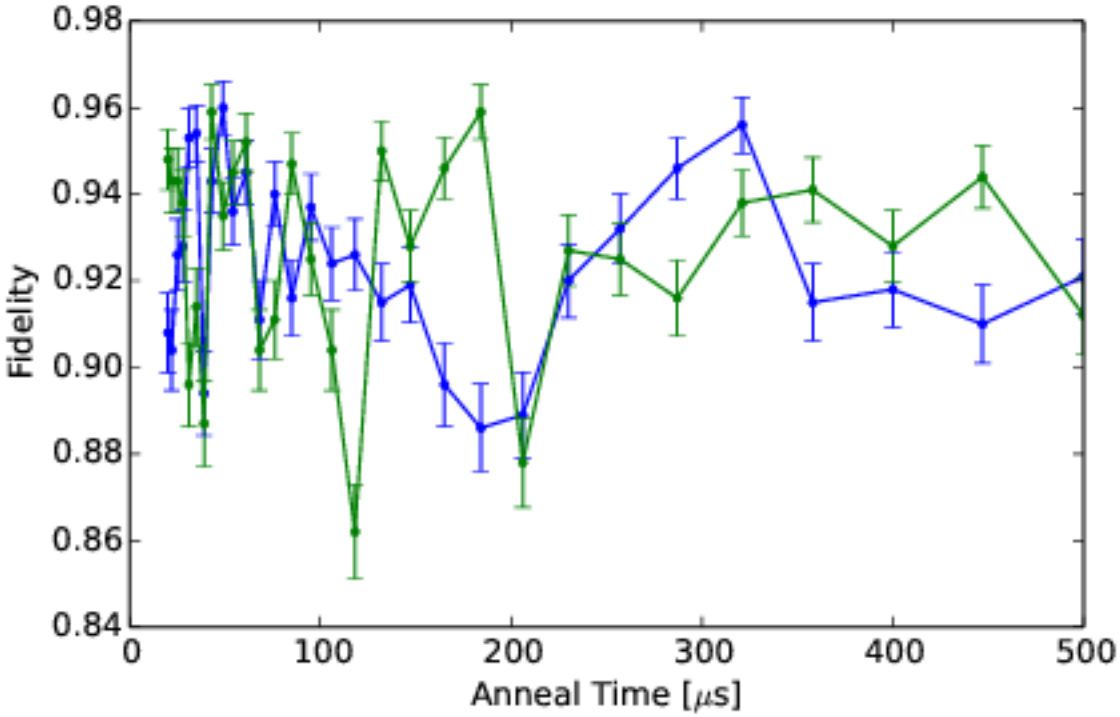

Figure 6.1: Fidelity as a function of anneal time for the Hamiltonian "k44" which consists of a single SAT clause embedded into a single qubyte. The error bars are estimated from the standard deviation of a binomial distribution of the reads.

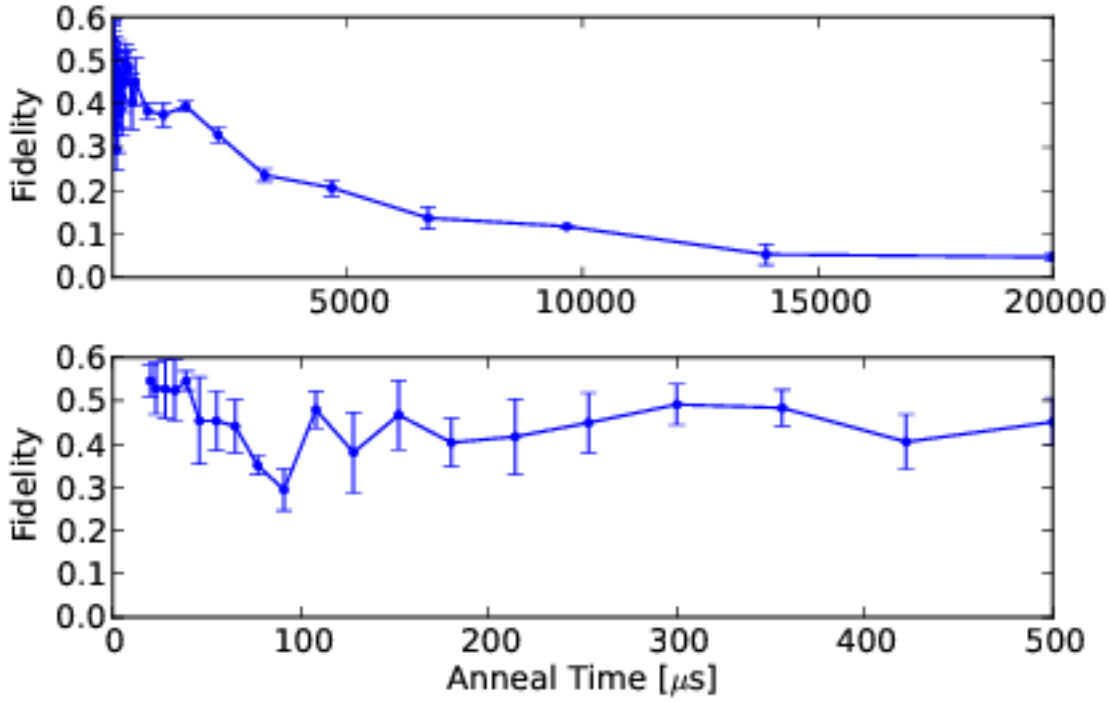

Figure 6.2: Fidelity vs anneal time with each data point averaged over four different runs for the single qubyte "k44_and" with a clone coupling value of -9. Each run was used to estimate the fidelity independently, with the final estimate the weighted average of each run. The errorbars reflect the standard deviation of the distribution of estimates of the runs.

but having only a single ground state. This Hamiltonian was evaluated for times ranging from 20 μs to 20 ms, with 4 runs taken at each time point and 100 reads per run. Figure 6.2 shows these results. Each run produced as estimated fidelity as in Section 6.1, and the results shown here were synthesized by making a weighted average of each of the individual run's estimated fidelities. The error bars shown here are the standard deviations of the means of the weighted averages, so small when the four runs produced nearly the same result and large when there was a large spread. The weighting for each run is $w_i = \frac{1}{\sigma^2}$ where σ is the standard deviation of each run. The short time oscillations are not entirely removed, although our estimates of the standard deviation matches up to the observed data much better; the trend of the data points falls almost entirely within the error bars.

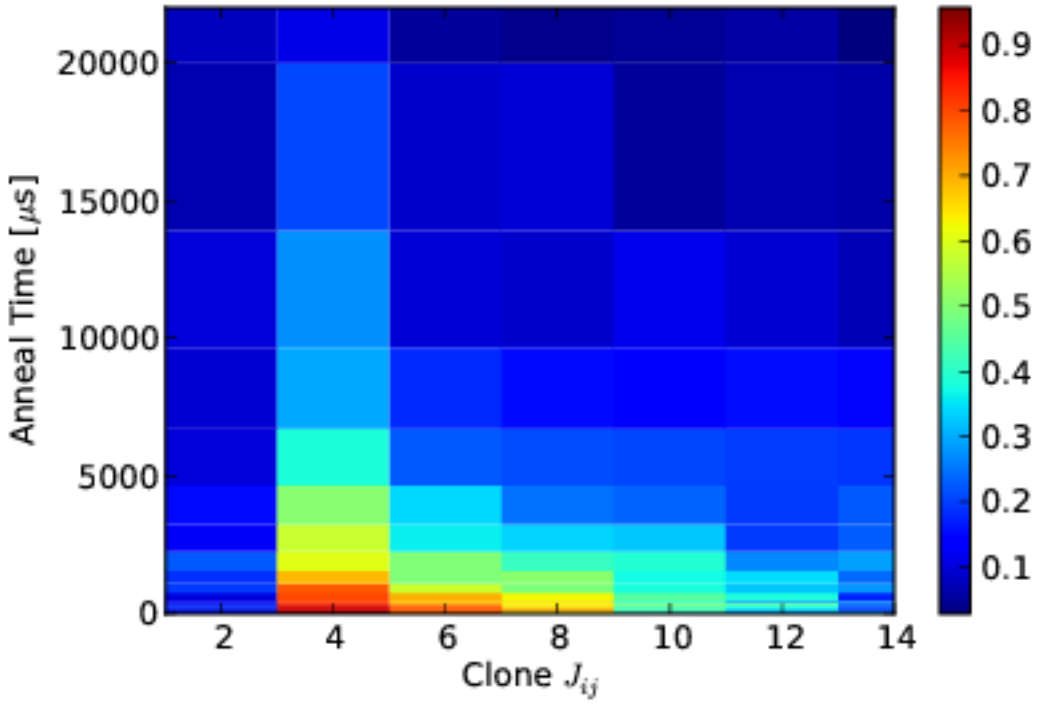

Figure 6.3: Heat map of the fidelity as a function of anneal time and clone coupling value for the single qubyte "k44_and". The measured fidelity drops of rapidly with increasing clone coupling, but low clone coupling values do not preserve the ground state intact. The sudden drop in fidelity at the smallest clone coupling value is caused by incorrect states with the same energy as the ground state.

6.4 Clone coupling value

As discussed in Sections 4.5 and 3.3, while for an ideal machine maximizing the clone coupling value would ensure that adding clones did not alter the ground states of our problem Hamiltonians, the D-Wave Two machine can only implement 15 different clone coupling values. In addition, these values must be evenly spaced. If we were to use a clone coupling value of -14 in a problem Hamiltonian which natively had coupling values of $-2, -1, 1, 2$, this would result in compressing each of the positive and negative couplings into one machine value. The machine only implements couplings of the form

$$\pm \frac{x}{7}, 1 \leq x \leq 7 \tag{6.1}$$

so the above example would result in a physically implemented Hamiltonian of -1/7, -1/7, 1/7, 1/7 in addition to the clone coupling of -1. Most of the time this will

result in a malformed Hamiltonian with incorrect ground states.

In addition to incorrect ground states, it is possible that the programming error in the machine is large enough to make the couplings 1/7 and 2/7 close enough to be likely to collide, whereas 1/7 and 3/7 could be far enough apart to be programmed correctly every time. This could mean that even in scenarios where D-Wave Two has enough range to encode the problem Hamiltonian couplings correctly with a large clone coupling value we still don't want to do so.

This means that when selecting a clone coupling value we must balance these two competing factors, the desire to make the clone coupling as large as possible from a Hamiltonian correctness standpoint, and the desire to make it as small as possible from a machine resolution viewpoint. The Hamiltonian correctness issue is correctness and knowable at the time the Hamiltonian is constructed; however in general to know whether a given clone coupling is large enough or not requires diagonalizing the Hamiltonian, and if this were possible then that would enable us to solve the initial computational problem directly. The machine resolution problem is not necessarily the same from run to run; it could be that the compression of the problem Hamiltonian caused by the clone coupling value only makes it more likely for the machine to end up in an excited state.

To empirically examine the impact of the value of the clone coupling, the "k44_and" Hamiltonian was embedded multiple different times with a clone coupling value ranging from -1 to -14 (from the smallest coupling in the problem Hamiltonian to twice the native resolution of the machine). Figure 6.3 shows a heat map of the results of annealing these Hamiltonians from 20 μs to 20 ms. Not only does fidelity increase as the clone coupling value is decreased, the short-time oscillations that we saw in the earlier results of "k44" in Figure 6.1 are removed. This is shown very clearly in Figure 6.4, which displays the standard deviation of the estimated fidelity as a function of anneal time and clone coupling value.

At longer anneal times the difference in fidelity is reduced, and the same long-time drop in fidelities that was observed in other Hamiltonians continues. This means that the mechanism for the long-time fidelity drop is not strongly impacted by the clone coupling value.

The fidelity drops off strongly from its peak at a clone coupling value of -3. Each

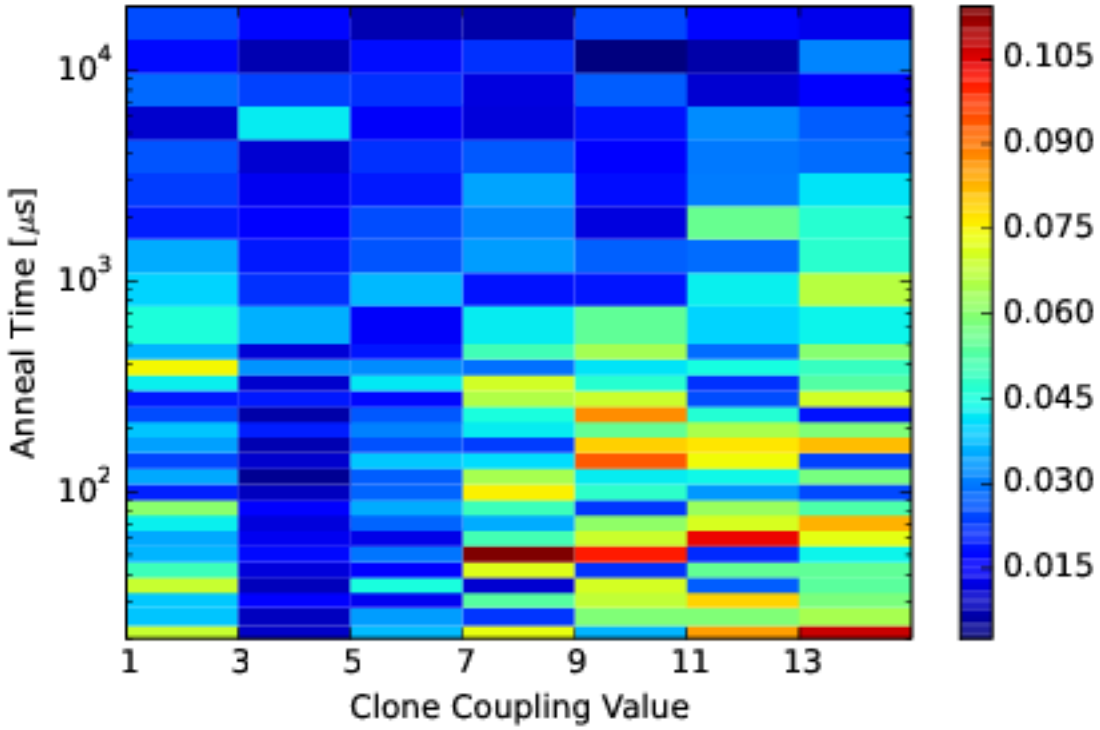

Figure 6.4: Heat map of the standard deviation of the estimated fidelity as a function of clone coupling value and annealing time for the single qubyte "k44_and". After $\sim$ 1000 μs the error drops down as we enter the long-time regime and the short-time oscillations die off. The spread of the fidelity is also much lower for a clone coupling value of -3, which corresponds to the higher fidelities.

of the higher clone coupling values corresponds to more and more compression of the computational couplings into fewer physical couplings. This pattern does not continue to the -1 coupling, because that clone coupling value results in a malformed Hamiltonian where a domain wall forms in the ferromagnetic couplings, leading to what should be an excited state having the same energy as the ground state. The "k44_and" Hamiltonian's largest coupling value is |3.0|, which suggests that a clone coupling value of -3 performs well because it does not compress any couplings together.

6.5 Size scaling

The size scaling behaviour of "k44_and" was also studied. This was done by copy and pasting the fields and couplings over to new unit cells. The resulting Hamiltonian still encodes the same SAT problem as the original "k44_and", and still only has one ground state, but is able to be scaled from 1 to 55 qubytes, a range of 8 to 440 spins. Figure 6.5 shows the fidelity plotted against anneal time and number of spins, for the clone coupling value which maximizes fidelity (-3). The same plot for other clone couplings looks similar but with lower fidelity values across the board. Larger Hamiltonians show similar behaviour to the original one qubyte "k44_and", but fall off even faster with increased anneal time.

Figure 6.6 shows the fidelity for annealing for 20 μs with a clone coupling value of -3 (that is, the values which maximize fidelity) as a function of number of spins. The result is very close to a straight line; a linear regression is shown against the data. This result would seem to indicate that for this particular set of parameters (anneal time of 20 μs, clone coupling value of -3) the D-Wave Two machine scales linearly. This is what we would expect if there was a some success probability for a given single "k44_and" unit cell, where the total success probability being the product of each one succeeding individually.

6.6 Boolean SAT Hamiltonian

The last investigation was conducted on a six variable, 21 clause 3-SAT Hamiltonian with two satisfying assignments, `(ttftft,ttttft)`. When embedded this Hamiltonian filled essentially the entire hardware graph. Figure 6.7 shows the estimated

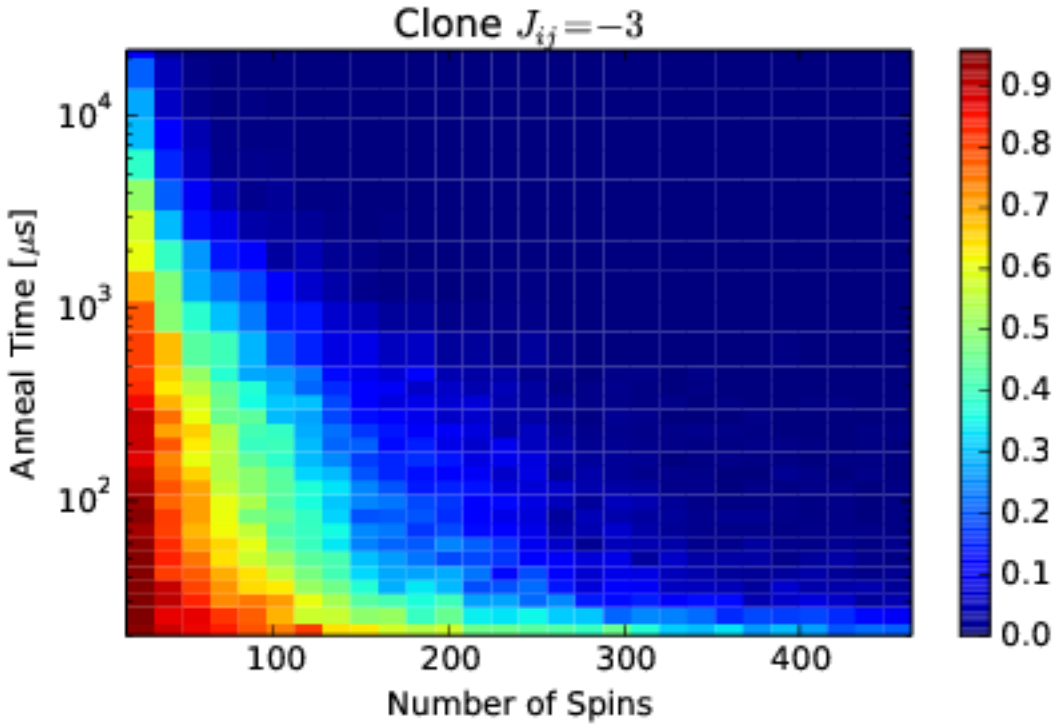

Figure 6.5: Heat map of the estimated fidelity as a function of annealing time and number of spins in the Hamiltonian, with a clone coupling value of -3.

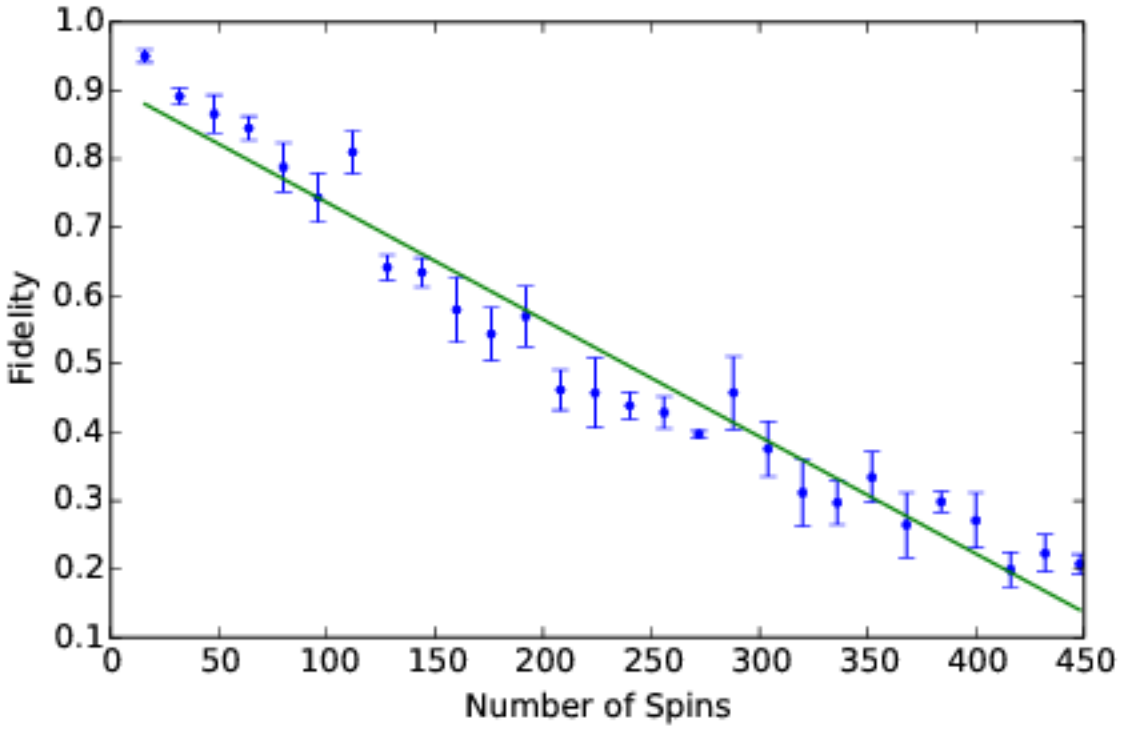

Figure 6.6: Fidelity as a function of number of qubits for an anneal time of 20 μs and a clone coupling value of -3 as well as line of best fit.

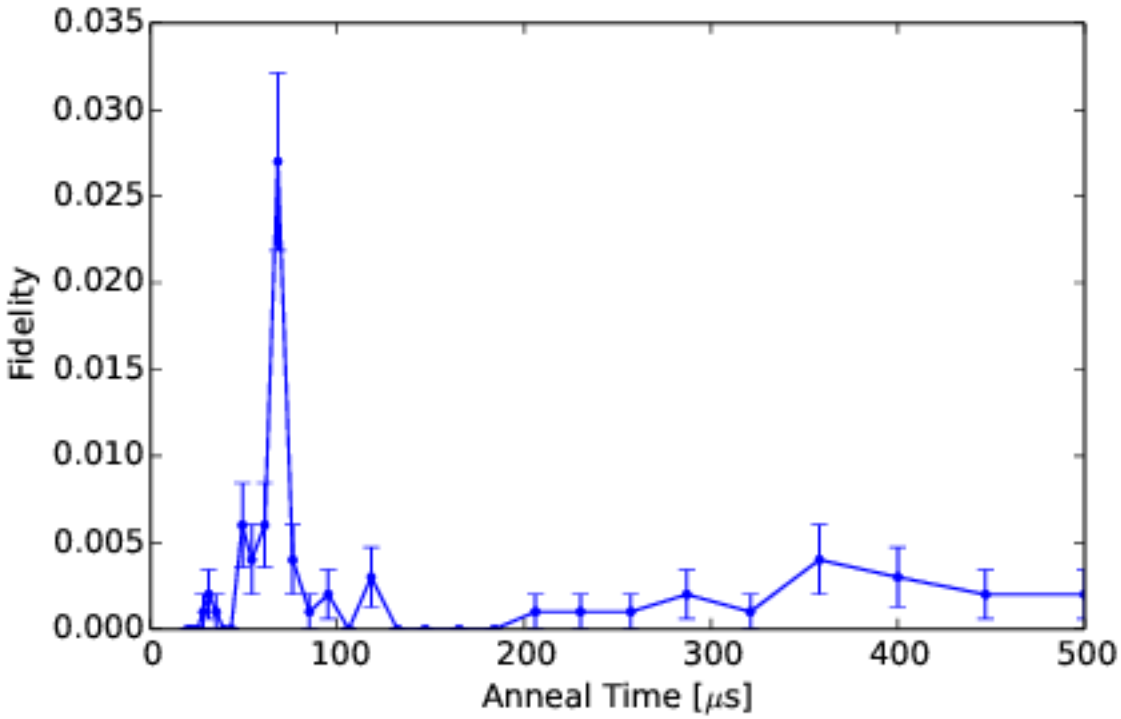

Figure 6.7: This figure shows the estimated fidelity as a function of annealing time for a six variable, 21 clause 3-SAT instance. The fidelity is essentially zero throughout the time range.

fidelity from anneal times ranging from 20 μs to 500 μs, estimated from 1000 reads and single runs at each anneal time. The fidelity is close to zero throughout the time range. In contrast to our previously observed Hamiltonians, the D-Wave Two machine seems unable to successfully solve this particular problem instance.

Figure 6.8 is a pair of histograms showing all 1000 results of annealing at 20 μs for both the "6_018" Hamiltonian and the 3-SAT Hamiltonian. The peak of the distribution for "6_018" is near the ground state, while the peak of the distribution for the 3-SAT Hamiltonian is in the neighbourhood of the 10th excited state. Presumably this is due to the energy levels of the 3-SAT Hamiltonian closing together where the "6_018" levels do not. The distribution of the results for the 3-SAT Hamiltonian is also broader, for reasons which are not clear. Both of these Hamiltonians are much too large to be simulated classically, so we cannot predict how the energy levels change over the annealing to explain the distribution differences directly.

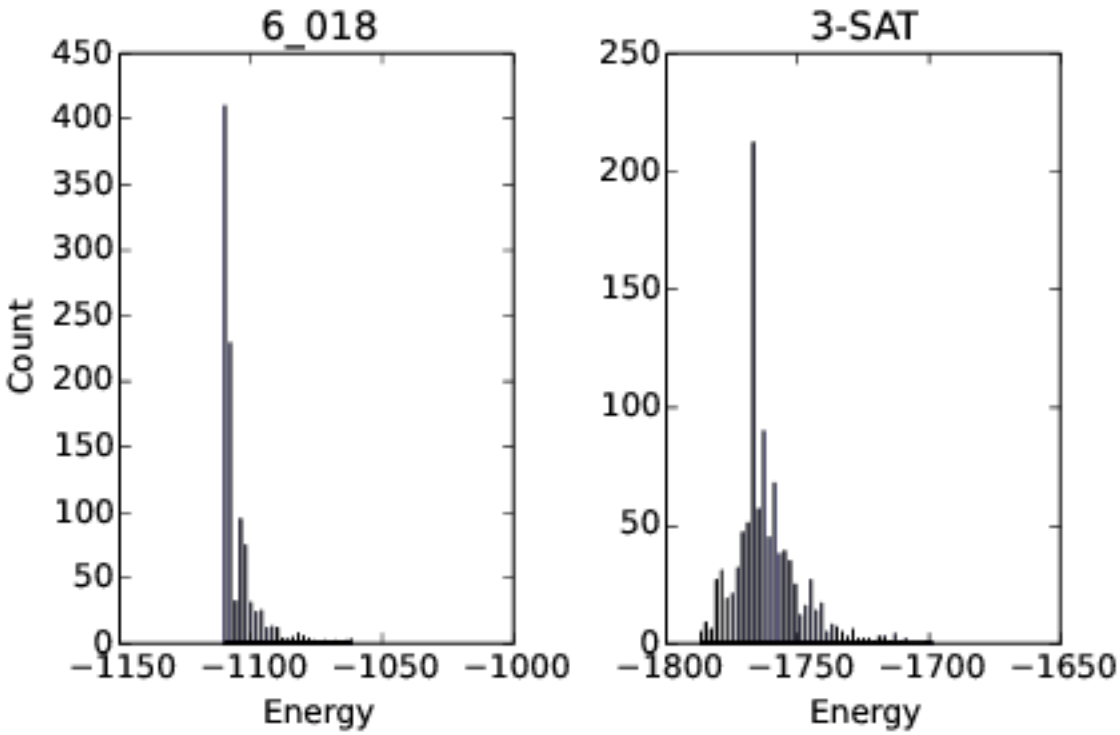

Figure 6.8: Histograms showing all 1000 states found after 20 μs of annealing for both "6_018" on the left and the 6 variable 3-SAT on the right. The fidelity is clearly much higher on the left. While the distribution of the "6_018" results peaks near the ground state and decays exponentially, the 3-SAT results are peaked in the vicinity of the 10th excited state.

Chapter 7

Conclusion

A variety of Hamiltonians were constructed and evaluated on the D-Wave Two hardware. We have seen several general methods for embedding problems into a form suitable for evaluation on an adiabatic quantum computer at a modest overhead, although even that overhead is a stiff price to pay on the current generation of adiabatic hardware. While the machine is able to solve single clause SAT problems and other small (8 qubit) problems, a real (still small) problem of 6 SAT variables was too difficult. The machine *was* able to solve a similar number of qubits when the SAT was simple (the "k44_and" case), so the difficulty of the 6 variable SAT must not be purely based on the number of qubits.

In the process of solving these problems we discovered several factors important for using the D-Wave family of adiabatic machines. Among them the fact that contrary to expectations shorter anneal times give better results, and that the limited number of built in machine couplings requires careful thought when embedding to ensure that coupling collisions do not occur.

Bibliography

[1] Scott Aaronson. Np-complete problems and physical reality. *http://www.scottaaronson.com/papers/npcomplete.pdf*, 2014.

[2] M. H. S. Amin. Consistency of the adiabatic theorem. *Phys. Rev. Lett.*, 102:220401, Jun 2009.

[3] Sergio Boixo, Tameem Albash, Federico M. Spedalieri, Nicholas Chancellor, and Daniel A. Lidar. Experimental signature of programmable quantum annealing. *Nat. Commun.*, 4, June 2013.

[4] Sergio Boixo, Troels F. Rønnow, Sergei V. Isakov, Zhihui Wang, David Wecker, Daniel A. Lidar, John M Martinis, and Matthias Troyer. Quantum annealing with more than one hundred qubits. *Nat. Phys.*, 10:218 – 224, February 2014.

[5] PI Bunyk, E Hoskinson, MW Johnson, E Tolkacheva, F Altomare, AJ Berkley, R Harris, JP Hilton, T Lanting, and J Whittaker. Architectural considerations in the design of a superconducting quantum annealing processor. *arXiv preprint arXiv:1401.5504*, 2014.

[6] Vicky Choi. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing*, 7(5):193–209, 2008.

[7] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[8] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.

[9] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001.

[10] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[11] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.

[12] Nicholas J Giordano and Hisao Nakanishi. *Computational physics*. Pearson Education India, 2006.

[13] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[14] R. Harris, J. Johansson, A. J. Berkley, M. W. Johnson, T. Lanting, Siyuan Han, P. Bunyk, E. Ladizinsky, T. Oh, I. Perminov, E. Tolkacheva, S. Uchaikin, E. M. Chapple, C. Enderud, C. Rich, M. Thom, J. Wang, B. Wilson, and G. Rose. Experimental demonstration of a robust and scalable flux qubit. *Phys. Rev. B*, 81:134510, Apr 2010.

[15] R. Harris, M. W. Johnson, T. Lanting, A. J. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, F. Cioata, I. Perminov, P. Spear, C. Enderud, C. Rich, S. Uchaikin, M. C. Thom, E. M. Chapple, J. Wang, B. Wilson, M. H. S. Amin, N. Dickson, K. Karimi, B. Macready, C. J. S. Truncik, and G. Rose. Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor. *Phys. Rev. B*, 82:024511, Jul 2010.

[16] Itay Hen and A. P. Young. Exponential complexity of the quantum adiabatic algorithm for certain satisfiability problems. *Phys. Rev. E*, 84:061152, Dec 2011.

[17] MW Johnson, MHS Amin, S Gildert, T Lanting, F Hamze, N Dickson, R Harris, AJ Berkley, J Johansson, P Bunyk, et al. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 2011.

[18] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

[19] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[20] Scott Kirkpatrick and Bart Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264(5163):1297–1301, 1994.

[21] Thaddeus D Ladd, Fedor Jelezko, Raymond Laflamme, Yasunobu Nakamura, Christopher Monroe, and Jeremy L OBrien. Quantum computers. *Nature*, 464(7285):45–53, 2010.

[22] T Lanting, AJ Przybysz, A Yu Smirnov, FM Spedalieri, MH Amin, AJ Berkley, R Harris, F Altomare, S Boixo, P Bunyk, et al. Entanglement in a quantum annealing processor. *Physical Review X*, 4(2):021041, 2014.

[23] Andrew Lucas. Ising formulations of many np problems. *arXiv preprint arXiv:1302.5843*, 2013.

[24] Hartmut Neven. Launching the quantum artificial intelligence lab. *googleresearch.blogspot.ca/2013/05/launching-quantum-artificial.html*, 2013.

[25] M. A. Nielson and I. L. Chuang. *Quantum Computation and Quantum Information Information*. Cambridge University Press, 2000.

[26] Kristen L. Pudenz, Tameem Albash, and Daniel A. Lidar. Error corrected quantum annealing with hundreds of qubits. *Nat. Commun.*, 5, February 2014.

[27] Troels F. Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V. Isakov, David Wecker, John M. Martinis, Daniel A. Lidar, and Matthias Troyer. Defining and detecting quantum speedup. *Science*, 2014.

[28] Geordie Rose. The google / nasa quantum artificial intelligence lab. *http://dwave.wordpress.com/2013/05/16/the-quantum-artificial-intelligence-lab/*, 2013.

[29] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, Oct 1995.

[30] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on computing*, 26(5):1484–1509, 1997.

[31] John A Smolin and Graeme Smith. Classical signature of quantum annealing. *arXiv preprint arXiv:1305.4904*, 2013.

[32] John S Townsend. *A modern approach to quantum mechanics*. University Science Books, 2000.

[33] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58:345–363, 1936.

[34] W. van Dam, M. Mosca, and U. Vazirani. How powerful is adiabatic quantum computation? In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 279–287, Oct 2001.

[35] A. J. Whiteway. Effects of errors in the hamiltonian programming process of adiabatic quantum computing. 2014.

[36] A. P. Young, S. Knysh, and V. N. Smelyanskiy. Size dependence of the minimum excitation gap in the quantum adiabatic algorithm. *Phys. Rev. Lett.*, 101:170503, Oct 2008.

[37] A. P. Young, S. Knysh, and V. N. Smelyanskiy. First-order phase transition in the quantum adiabatic algorithm. *Phys. Rev. Lett.*, 104:020502, Jan 2010.

[38] Nouredine Zettili. *Quantum Mechanics: Concepts and Applications*. John Wiley and Sons, second edition, 2009.

Appendix A

The Adiabatic Theorem

We derive a perturbative version of the adiabatic theorem here following Zettili[38]. For more on the validity of the adiabatic theorem including its use outside of the perturbative regime see Amin[2].

Using time dependent perturbation theory, we find the transition probability $P_{if}(t)$ from state $|\psi_i\rangle$ to state $|\psi_f\rangle$ is (to first order)[32]

$$P_{if}(t) = \left| -\frac{i}{\hbar} \int_0^t \langle \psi_f | \hat{V}(t') | \psi_f \rangle \, e^{i\omega_{fi} t'} \mathrm{d}t' \right|^2 \tag{A.1}$$

where $\hat{V}(t)$ is the time dependent perturbative part of the Hamiltonian, $\omega_{fi} = \frac{E_f - E_i}{\hbar}$. We can recast this into a form with explicit dependence on the time derivative of $\hat{V}(t)$ using the fact that $e^{i\omega_{fi}t} = (1/i\omega_{fi})\partial e^{i\omega_{fi}t}/\partial t$, and then integrating by parts to get

$$\begin{aligned} P_{if}(t) &= -\frac{1}{\hbar\omega fi} \langle \psi_f | \hat{V}(t) | \psi_i \rangle \, e^{i\omega_{fi}t} \Big|_{t=0}^{t} + \frac{1}{\hbar\omega_{fi}} \int_0^t e^{i\omega_{fi}t'} \left(\frac{\partial}{\partial t'} \langle \psi_f | \hat{V}(t') | \psi_i \rangle \right) \mathrm{d}t' \\ &= \frac{1}{\hbar\omega_{fi}} \int_0^t e^{i\omega_{fi}t'} \left(\frac{\partial}{\partial t'} \langle \psi_f | \hat{V}(t') | \psi_i \rangle \right) \mathrm{d}t' \end{aligned} \tag{A.2}$$

where we assume $\hat{V}$ is turned on and then off again and thus vanishes at the limits.

If we assume that the speed at which $\hat{V}$ is changing is very small, then the derivative term $\partial \langle \psi_f | \hat{V}(t') | \psi_i \rangle / \partial \mathrm{d}t'$ will be almost constant over the integral, and we can pull it out giving a transition probability of

$$\begin{aligned} P_{if}(t) &\approx \frac{1}{\hbar^2\omega_{fi}^2} \left| \frac{\partial}{\partial t} \langle \psi_f | \hat{V}(t) | psi_i \rangle \right|^2 \left| \int_0^t e^{i\omega_{fi}t'} \mathrm{d}t' \right|^2 \\ &\approx \frac{4}{\hbar^2\omega_{fi}^4} \left| \frac{\partial}{\partial t} \langle \psi_f | \hat{V}(t) | \psi_i \rangle \right|^2 \sin^2\left(\frac{\omega_{fi}t}{2} \right) \end{aligned} \tag{A.3}$$

In the adiabatic limit this transition probability goes to zero (or alternatively, the adiabatic limit is defined as when this transition probability goes to zero). In that case we have $\sin^2(x) << 1$ and we arrive at the adiabatic condition:

$$\left|\hbar\frac{\partial}{\partial t}\langle\psi_f|\hat{V}(t)|\psi_i\rangle\right| << \Delta^2 \tag{A.4}$$

where Δ, the energy gap, is defined as $\Delta = |E_f - E_i|$.