4-8-2019

Exploring and improving the uses of near-term quantum annealers

Nic Ezzell
Mississippi State University

Exploring and improving the uses of near-term quantum annealers

Nic Ezzell

Department of Physics and Astronomy

Mississippi State University

Supervisor

Mark Novotny

In partial fulfillment of the requirements for the degree of

Bachelors of Science in Physics and in Mathematics

April 8, 2019

Acknowledgements

There are countless people I'd like to thank for helping me get to this point in my academic career. But to keep it short, I will restrict myself to those who were immediately involved with my thesis.

First, I'd like to thank my primary mentor and supervisor for this thesis, Dr. Mark Novotny. He's done a lot help me on my journey into quantum computing–including, of course, letting me join his research group.

I'd also like to thank Dr. Travis Humble and Paul Kairys of Oak Ridge National Laboratory for working with me in Summer of 2018 where I completed about half of the work I am writing on.

Last but not least, I'd like to thank Dr. Yaroslav Koshka and Dr. Seth Oppenheimer for agreeing to take time out of their busy schedules to be apart of my thesis committee, and of course, Mrs. Katelyn Wright for helping us all schedule this in the first place!

General Audience Abstract

In this project, we are interested in exploring and improving the usage of modern day quantum annealing (QA). In short, QA is a quantum process that is designed to solve optimization problems. Optimization problems try to find the best answer from a choice of many. For example, a pizza delivery driver might be interested in finding the most gas efficient route to save money. It is believed that QA, by utilizing resources like quantum tunneling, can outperform modern computational methods to find answers to such questions. To aide us in our exploration, we have developed a program that makes it easier to use the state-of-the-art quantum annealing devices designed by D-Wave Systems as well as run our own numerical tests. Using it, we are able to show some of the pitfalls of current quantum annealing devices as well as some promising future directions. For example, we show a method we created could reduce errors on modern QA devices using numeric simulations.

Abstract

Quantum annealing (QA) is a global search-heuristic designed to solve NP-hard optimization problems. Currently, D-Wave Systems is the only commercial QA company, boasting a line of chips that have over 2000 working qubits. Unfortunately, their API lacks many useful features for exploring the actual physics and efficacy of QA as an optimization tool. Further, the typical QA routine is prone to errors for a variety of reasons: noise, problem mis-specification, minor-embedding errors, and using incorrect annealing parameters. This work attempts to alleviate both of these issues with a single Python package: dwaveutils. With it, we explore two case studies: simulations of transverse-field Ising Hamiltonians and forward-reverse error-mitigation annealing. The first shows the utility that dwaveutils brings when it comes to submitting and post-processing problems on D-Wave. The second shows the power of having a numeric annealing solver that allows for exploration of modified annealing routines. With it, we've uncovered several potentially fruitful investigations that could reduce errors on modern devices with no cost in ancilla qubits and allow for a new feature: graph introspection. Overall, our package allows for easy exploration and improvements of modern quantum annealing.

Contents

List of Figures

Chapter 1

Introduction

Thesis Overview

Throughout this thesis, we intend to give just enough details for a student familiar with under-gradate level quantum mechanics to follow (though we do at least include the postulates of quantum mechanics for reference in Appendix A). In this way, we do not intend for this to be "self-contained." In fact, we will often refer readers to external sources for additional information.

We begin with standard and expected topics in the Introduction: history and motivation and some prerequisite theory. In particular, we start with a brief discussion of basic quantum mechanics relevant for quantum annealing. We emphasize solution regimes to Schrödinger's equation under a time-dependent Hamiltonian–something often glossed over in an undergraduate physics curriculum. We also include a section on the Ising model, as this is the basis for the quantum annealing procedures used on D-Wave Systems devices and in our simulations. Finally, we discuss how the Ising model is related to solving difficult optimization problems, the theoretical basis for quantum annealing.

Next, we discuss the basic theory and practicalities of D-Wave System quan-

tum annealing devices. We emphasize the basic theoretical ideas and the process of actually submitting a problem. Physical implementation is discussed, but not in theoretical detail. From here, this naturally spills into a discussion of our simulation package. Our package accomplishes two things. First, it makes submitting problems on D-Wave and doing the subsequent data wrangling much easier. Second, it allows one to submit numerical simulations of the quantum annealing that emulates D-Wave devices.

Finally, we present two case studies that illustrate the usefulness of our package: *Simulations of Ising Model Quantum Phase Transitions* and *Forward Reverse Error Mitigation Annealing.*

1.1 History and Motivation

Quantum computing (QC) is a relatively new field with it's initial roots sprouting around the 1960's with Stephen Wiesner's work [2]. In 1981, Richard Feynman gave a talk at the *First Conference on the Physics of Computation* at MIT where he discussed the impossibility of simulating large quantum systems with classical computers and proposed a primitive model of quantum computation to perform quantum chemistry [3]. For many, this was the birth of quantum computing as it's own field. Today, it attracts the attention of the world. With over 100 start-up companies, Google, Microsoft, IBM, Intel, Volkswagen, and universities and governments across the world all pouring resources into the field at once, it's no surprise that it's blowing up [4], but why?

Richard Feynman's proposal lies at the heart of people's interest in QC: the possibility for a *quantum advantage* and ultimately for *quantum supremacy.* Put simply, if a problem can be solved faster on a quantum computer than on a classical computer, then we say that there is a quantum advantage for that problem. If

that difference is large enough to make a problem go from practically unsolvable (intractable) on modern classical hardware to tractable on quantum hardware, then we've found a demonstration of so called quantum supremacy. At this point, it's clear that not all problems even provide a quantum advantage; and yet, the interest is still there [5]. This is for a variety of reasons, but perhaps the biggest is Peter Shore's prime-factorization algorithm [6]. Currently, this algorithm has a demonstrable exponential speed-up over any known classical algorithm. Strictly speaking, a lower bound on classical prime-factorization algorithms has never been proven, but the computer science world is fairly confident that a better algorithm will not be found. In fact, that prime-factorization is "hard"–ie intractable–via classical algorithms is the entire basis of Random Secure Access (RSA) encryption that helps keeps computers safe. In other words, the problems that have been shown to potentially have a quantum advantage are very important–from RSA encryption via prime factorization to simulations of quantum systems. Again, buy why?

In general, the why is non-trivial, but there is a nice way to get an intuitive picture by investigating Feynman's conjecture more closely. In particular, his idea is predicated on the idea that it is difficult to simulate quantum systems with a classical computer. A classical computer relies on classical rules that we are comfortable with. If I tell you a table is 1 meter in length, I am describing a property of the table, but I'm also implicitly saying that if you go and measure the table that (up to some small measurement uncertainty dependent on your device), you will also get 1 meter. In the classical world, we take for granted that there is a direct correspondence between an object's properties and the results of measurements. It turns out that this correspondence doesn't hold in the quantum world–so this is not just fluffy philosophizing.

As described in Appendix A, the state of a quantum system and measurements

of that system are no longer in direct correspondence. In general, measurements are probabilistic, and worse, the very act of measuring a quantum system alters its state. Hence, the state of a quantum system can only be sussed out by preparing an ensemble of identically prepared quantum systems and making the same measurement on all of them. It's only in the statistical limit on a infinite set of measurements that absolute certain conclusions regarding the state can be made.

Though this is certainly not obvious from our discussion so far, it turns out that keeping track of all the strange properties of quantum systems on a classical computer takes an enormous amount of memory and time. In fact, the memory cost grows exponentially with the system size. For example, simulating 500 interacting qubits, the simplest quantum mechanical system imaginable, requires, at minimum, storing 2^{500} floating bit numbers in memory. For context, this number is larger than the number of atoms in the Universe [7]! But in nature, quantum systems with many more qubits than 500 interact all time; evidently, Nature is capable of seamlessly performing enormous calculations not conceivable on any classical computer. Quantum computation is a field that tries to harness this unimaginable computational power.

While the world has good reason to be enthralled by the potential of quantum computers, no breakthroughs have yet lead to an experimentally verified quantum advantage. In truth, most well developed theory in the field relies on a *fault-tolerant* quantum computer with hundreds if not thousands of qubits, the basic logical unit of QC that we'll discuss in more depth in the next section on theory. Today's devices are Noisy Intermediate-Scale Quantum (NISQ) computers, plagued by things like experimental noise and lack of error-correction [8]. There are two ways to fix this: create a fault-tolerant device or re-evaluate current algorithms and usage to make NISQ era devices useful now. While the wider community works in both areas, our work focuses on the latter goal.

1.2 Prerequisite Theory

If you are totally unfamiliar with quantum mechanics, I would recommend reading about the Stern-Gerlach experiment. Pedagogically, McIntyre's treatment in the first chapter of Quantum Mechanics: A Paradigms Approach is quite good [9], but if you are interested in the source, you can read the original paper [10]. I've included a concise statement of the postulates as taken from Nielsen and Chaung [7] in Appendix A that I will refer to throughout my discussion, so it might be worthwhile to give them a quick look if you aren't comfortable enough with undergraduate level QM to list them off the top of your head.

1.2.1 A Primer on Quantum Mechanics

Associated with every quantum system is a complex vector space with an inner product, known as a Hilbert space, $\mathcal{H}$. One of the most important properties of Hilbert spaces is that they are complete, so once $\mathcal{H}$ is specified for a system, all possible states of that system can be realized by constructing normalized linear combinations of basis vectors in $\mathcal{H}$. In the notation of quantum mechanics, these normalized state vectors are represented with "kets", $|\psi\rangle$.

The simplest quantum system is a *qubit*, or quantum bit, and it is the basic unit of quantum information used in quantum computing. Qubits reside in $\mathcal{H}_2$, a two-dimensional Hilbert space. Traditionally, the orthonormal basis for $\mathcal{H}_2$ is represented by the state-vectors $|0\rangle$ and $|1\rangle$. By completeness, any arbitrary state vector in $\mathcal{H}_2$ can be written as

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle\,, \tag{1.1}$$

where a and b are complex numbers. The normalization condition requires that the inner product of $|\psi\rangle$ with itself be one, or compactly, $\langle\psi|\psi\rangle = 1$, which implies

that $|a|^2 + |b|^2 = 1$. This is important for the statistical interpretation of quantum mechanics encoded in the measurement postulate[1]. For a qubit, the statistical interpretation means that measurements on $|\psi\rangle$ give exclusive outcomes of $|0\rangle$ with probability $|a|^2$ or outcomes of $|1\rangle$ with probability $|b|^2$.

The time evolution of the state of a closed quantum system is described by the *Schrödinger equation*,

$$i\hbar \frac{\mathrm{d}\,|\psi\rangle}{\mathrm{d}t} = H\,|\psi\rangle\,, \tag{1.2}$$

where i is the imaginary number, $\hbar$ is Plank's constant h divided by 2π with numeric value $\hbar = 1.05457 \times 10^{-34} J \cdot s$, $|\psi\rangle$ is the quantum state vector, and H is a Hermitian operator known as the *Hamiltonian*. If H is independent of time, then solving the Schrödinger equation is equivalent to finding the eigenvalues and eigenvectors of H. In particular, since H is Hermitian, it is guaranteed to have a spectral decomposition

$$H = \sum_E E\,|E\rangle\,\langle E|\,, \tag{1.3}$$

with eigenvalues E and corresponding normalized eigenvectors $|E\rangle$. In terms of physics, E is the energy of the eigenstate $|E\rangle$. These states are also known as stationary states because their time dependence under the Schrödinger equation is only a phase,

$$|E\rangle = exp(-iEt/\hbar)\,|E\rangle\,. \tag{1.4}$$

Note that physical Hamiltonians are bounded below. That is, there always exists a lowest energy state which we call the *ground-state*. Any state with a higher energy is called an *excited state*, ordered by increasing energy. In addition, some systems have multiple distinct states that give rise to the same energy. In this case, we say that the system is *degenerate*. In particular, if a system's first excited state has four distinct physical states that lead to that energy, we would say that its first

[1] See Appendix A, Postulate 3.

excited state is four-fold degenerate.

1.2.2 Time-Dependent Hamiltonians

If $\frac{\mathrm{d}H}{\mathrm{d}t} \neq 0$, then the solution, in general, is not nearly as straightforward. We will summarize three important results regarding the solution of Eq. 1.2 for time-dependent Hamiltonians in the following three sub-sections devoted to the sudden approximation, the adiabatic theorem, and Landau-Zener transitions. In all three cases, we will be interested in a time-dependent Hamiltonian of the form [11]

$$H(t) = \begin{cases} H_1 & t \leq t_1 \\ H_i(t) & t_1 < t < t_2 \\ H_2 & t \geq t_2. \end{cases}$$

In words, this is a Hamiltonian that changes from H_1 to H_2 over some time interval $T = t_2 - t_1$. In between, it can, in general, take on any functional form of a valid Hamiltonian, H_i. Assuming the initial state was an eigenstate of H_1, we'd like to predict the state of the system at $t = t_2$ wherein the system then evolves according to H_2. In our following discussion, it will be useful to consider the size of T with respect to the minimum Bohr frequency of a system,

$$\omega_{21} = \frac{E_2 - E_1}{\hbar}, \tag{1.5}$$

where E_2 and E_1 and the first-excited state and ground-state of the system, respectively. Furthermore, it is useful to consider the point during the evolution from H_1 to H_2 for which E_2 and E_1 are closest in value. This point is known as

the "level-crossing", and the spectral gap at this point can be written as

$$\Delta_{H_1 \to H_2} = \min(E_2(t) - E_1(t))_{H_1 \to H_2}, \tag{1.6}$$

where the subscript $H_1 \to H_2$ will be understood and left off from now on.

The Sudden Approximation [11]

As $T \to 0$, the Hamiltonian is said to change *suddenly*, and intuitively, this means that the state does not have time to react to the change. Mathematically, if the initial state is $|\psi(t_1)\rangle$, then the final state satisfies

$$\begin{aligned} |\psi(t_2)\rangle &= |\psi(t_1)\rangle \\ |\langle\psi(t_2)|\psi(t_2)\rangle|^2 &= |\langle\psi(t_1)|\psi(t_1)\rangle|^2, \end{aligned} \tag{1.7}$$

so it truly is the same state–just written in H_2's basis, presumably different from H_1's.

In reality, $T = 0$ is not possible, but the sudden approximation is expected to be valid as long as T is much shorter than the time-scale governing the dynamics of states in H_1. Put mathematically, we need T to be much smaller than the time-scale given by the minimum Bohr frequency

$$T \ll \frac{2\pi}{\omega_{21}}. \tag{1.8}$$

The Adiabatic Theorem [11]

As $T \to \infty$, the Hamiltonian is said to change adiabatically. Intuitively, if a Hamiltonian is changed slowly, the state always has time to adjust to the time-dependent Hamiltonian, so it simply tracks the *instantaneous eigenstates.*

Under this approximation, one can diagonalize the Hamiltonian at each t,

$$H(t)\left|i\right\rangle_t = E_i(t)\left|i\right\rangle_t . \tag{1.9}$$

The eigenstates are called the *instantaneous eigenstates* since they diagonalize the Hamiltonian at each instance. The adiabatic theorem states that,

$$\left|i(t)\right\rangle = e^{-i\int_{t_1}^{t} dt' E_i(t')} e^{i\phi(t)}\left|i\right\rangle_t \tag{1.10}$$

saying that a state-ket's time-evolution tracks the corresponding instantaneous eigenstate. The first and second exponential are known as the *dynamical* and *geometrical* phases, respectively, and while they are important, they will play no role in our future discussions. Evidently, the probability distribution function necessarily changes as well, so

$$\left|\langle\psi(t_2)|\psi(t_2)\rangle\right|^2 \neq \left|\langle\psi(t_1)|\psi(t_1)\rangle\right|^2 .$$

Since $T \to \infty$ is not a realistic (or at least useful) bound, there are numerous *adiabatic theorems* defined by different constraints on $H(t)$ that give rise to different minimum upper bounds on T [12]. For our case, we will employ a highly simplified condition by flipping the inequality in Eq. 1.8 and realizing the tightest-bound is at the level-crossing,

$$T \gg \frac{2\pi\hbar}{\Delta}, \tag{1.11}$$

where the subscript t_{min} refers to the instantaneous time for which this is the tightest bound–referring to the time during which the energy gap $E_2 - E_1$ is the smallest. Note that most rigorous minimum upper-bounds on T for the adiabatic theorem are stricter than Eq. 1.11.

Landau-Zener Transitions [13]

If the time scale of the change T neither satisfies the sudden nor the adiabatic conditions, Eq. 1.8 and Eq. 1.11, then non-trivial energy-level transitions can occur during the evolution of $H(t)$. In general, that is all that can be said. Nevertheless, a qualitative (and sometimes even quantitative) idea of what is going on can be derived with a few simplifications to the most general case.

The Landau-Zener formula is one such analytic solution for a 2-level, non-degenerate quantum mechanical system where $H(t)$ varies such that the separation between E_2 and E_1 is a linear function of time, $\Delta E(t) = E_2(t) - E_1(t) = \alpha t$. With these simplifications, one can derive the probability of a non-adiabatic transition,

$$P_D = e^{-2\pi\Gamma} \text{ with} \qquad (1.12)$$
$$\Gamma = \frac{\Delta^2}{4\hbar|\alpha|}.$$

1.2.3 The Ising Model of Magnetism

This Ising model is used to understand the physics of phase transitions in magnetic materials [14]. In particular, it tries to explain how short-range interactions between things like molecules in a crystal give rise to long-range, correlated behavior and how this gives rise to magnetic phase-transitions.

Graphically, the Ising model can be represented by a d-dimensional lattice. A lattice is simply a set of connected, regularly spaced points. For example, a 1-dimensional lattice with 3 lattice sites (nodes in graph theory) is shown in Fig. 1.1, and a 2-dimensional square lattice with 2 lattice sites is shown in Fig. 1.2. Each line that connects lattice sites is known as a bond (edge in graph theory), and if two lattice sites are connected by a bond, they are considered "nearest-neighbors".

In the Ising model, each lattice site can take on a *spin* value of $s_i = \pm 1$. Geometrically, we can think of this as bar magnets pointing "up" or "down". These spins can interact with external magnetic fields and with themselves. The

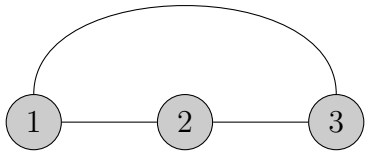

Figure 1.1: A 1-d lattice with 3 lattice sites (nodes). The curved line that connects nodes 1 and 3 represents a periodic-boundary condition that connects the ends of a 1-d lattice chain. This is commonly used to avoid difficult boundary-value problems.

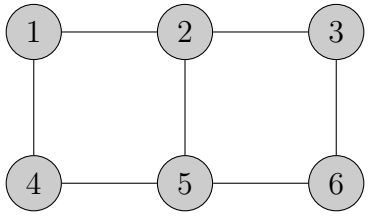

Figure 1.2: A 2-d square lattice with 6 lattice sites.

total energy of a particular configuration can be summarized classically,

$$E(\vec{s};\{\vec{h},J\}) = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j, \tag{1.13}$$

where $\vec{s} = \{s_1, s_2, \ldots, s_N\}$, $\vec{h} = \{h_1, h_2, \ldots, h_N\}$, and J is a symmetric matrix. The h_i parameters encode the energy associated with a spin aligning parallel to ($h_i s_i < 0$) or anti-parallel to ($h_i s_i > 0$) an externally applied magnetic field. The J_{ij} parameters capture the tendency for nearest-neighbor spins to want to align parallel ($J_{ij} < 0$) or anti-parallel ($J_{ij} > 0$) with respect to each other. We often refer to this as ferromagnetic or anti-ferromagnetic to make a connection to the magnetic systems we are trying to model. The particular notation here is used to emphasize that E has a direct dependence on $\vec{s}$ and a parametric dependence on $\vec{h}$ and J. In other words, the energy depends on the spin-configuration of the lattice for a given set of interaction parameters.

In the quantum picture, each of these spins can be represented as a qubit, and

we are more interested in the Hamiltonian operator than the energy function for a given configuration. This Hamiltonian can be written as,

$$H = \Gamma \sum_i \sigma_i^x + \left(\sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z \right). \tag{1.14}$$

Here, $\sigma_i^x(\sigma_i^z)$ is the $x(z)$ Pauli-spin operator acting on the i^{th} qubit [15]. By convention, we refer to the magnetic field pointing in the z-direction as a "longitudinal field" and that in the x-direction a "transverse field." While there is not a "preferred" spin-direction axis, quantum computers almost always make measurements in the z basis, so this is what we mean by longitudinal and transverse. With this in mind, the effect of the transverse magnetic field comes from the commutator relation,

$$[\sigma^x, \sigma^z] = -2i\sigma_y \neq 0. \tag{1.15}$$

At risk of oversimplifying, this has an effect of "scrambling" the spins analogous to temperature. The stronger the transverse field strength, the closer the qubit spin-state measurement statistics approach a random 50/50 split of spin up and down.

In many computations, it is useful to recast the Hamiltonian into a matrix. Doing so requires knowledge of the tensor product nature of the Hilbert space of composite quantum systems [1]. The basic idea is that an N-qubit problem must have a Hamiltonian of N-qubit operators. Thus, if any operator acts on a subset of qubits (as nearest neighbor interactions do), then we must extend the operation to all other qubits by applying the identity operator on everything else. As it turns out, the tensor product, denoted $\otimes$, is just the right operation to translate this intuitive idea into one of mathematical rigour. If we write our operators as matrices, then the Kronecker product produces the matrix that respects the

[1] See Postulate 4 in Appendix A: The Postulates of Quantum Mechanics

same mathematical rules. For example, for an $N = 3$ qubit Ising system with $\Gamma = h_2 = h_3 = J_{12} = J_{13} = 0$, we get

$$\begin{aligned} H_{\text{3qubits}} &= h_1\sigma_1^z + J_{23}\sigma_2^z\sigma_3^z \\ &= h_1(\sigma^z \otimes I \otimes I) + J_{23}(I \otimes \sigma^z \otimes \sigma^z), \end{aligned}$$

which produces a diagonal $2^N \times 2^N = 8 \times 8$ matrix where each diagonal entry is an eigen-energy of the system.

1.2.4 Quantum Phase Transitions

A quantum phase transition (QPT) is a change in the bulk-properties of a material at absolute zero temperature ($0K$) [16]. Concretely, if a material is at $0K$, then it is necessarily in its ground-state (lowest energy state). A QPT results from continuously changing some non-temperature parameters (like magnetic field strength or pressure), causing the ground-state to change abruptly. Since absolute zero is not physically realizable, we must consider a more realistic limit,

$$\hbar\omega_{21} \gg k_B T, \tag{1.16}$$

which says that the energy scale associated with quantum effects must be much greater than that due to thermal effects. Here, k_B is Boltzmann's constant. In our case, we are interested in observing changes in the magnetic properties of an Ising system through continuous changes in Γ, the $h_i's$, and the $J_{ij}'s$ as defined in Eq. 1.14.

As discussed, the Ising model was originally invented as a means to study phase transitions of magnetic systems. One common way to "study phase transitions" of the Ising model 1.14 is as follows: fix $\vec{h}$, $\vec{J}$, and $T = 0$ and see what happens

if Γ, the transverse magnetic field strength is continuously varied. Depending on the values of the h_i's and J_{ij}'s, the energy landscape can be extremely complex[1]. This problem can become even more difficult if one also varies T–well at least until thermal effects are much stronger than quantum effects. At high enough T, it becomes easy: everything is randomly oriented and the bulk-sum magnetization sums to zero on average.

1.2.5 Quantum Spin Glasses and NP-Complete/NP-Hard

Intuitively, a quantum spin glass is the "hardest" instance of an Ising system in regards to understanding the phase transition properties. One way to create a spin glass is simply to take Eq. 1.14 and set $\vec{h} = \vec{0}$ while assigning random values to the $J'_{\text{ij}}s$ sampled from a Gaussian with zero mean and variance one [17]. For large systems, this is practically guaranteed to create a very rugged energy landscape. Typically, physicists consider these couplings fixed in time for an instance of a spin glass, and for historical reasons call these fixed couplings *quenched.* Since the J_{ij}'s are random, then any given qubit is potentially coupled to neighboring qubits with competing interactions, thereby making the entire system frustrated.

By "competing interactions" we mean something very specific: a single qubit which–due to neighboring qubit orientations and the interaction type–is compelled to be both spin up and down at the same time. We can understand this concretely with a somewhat contrived example[2]. Consider a 1-d qubit chain as in Fig. 1.1 with $J_{12} = J_{23} = 1$ (anti-ferromagnetic) and $J_{13} = 0$. One valid ground-state configuration would be $|101\rangle$–that is having qubits 1 and 3 pointing down while qubit 2 is pointing up. If we then turn on $J_{13} = 1$ abruptly when the system is in this state, then this creates frustration since qubits 1 and 3 now want to point

[1] These are often referred to as rugged landscapes.
[2] A physicist's favorite pedagogical tool

in opposite directions which competes with qubits 1 and 2 and 2 and 3 wanting to point in opposite directions. Of course, this system is more than contrived– it is not even a spin glass. "Turning on" J_{13} means we are discussing a system with time-varying J couplings, and this isn't allowed for a traditional spin glass. Nevertheless, I hope this contrived example helps readers understand the dynamics of a competing interaction that leads to a frustrated system without having to follow the details of a more complicated physical system.

We will not go further into the details of phase transitions in spin glasses as this could fill entire volumes. Needless to say, it is complicated, and only 1 dimensional systems can be solved analytically. For systems of dimension 2 and greater, the properties of spin glasses must be explored computationally, and in the computational world, there is a precise meaning of "just how hard is it really?" Such questions fall into the domain of *computational complexity theory.* Loosely speaking, this field studies how difficult it is to solve a computational problem by monitoring how much memory, time, or energy it takes to get to a solution. To organize the field, computer scientists have devised a number of *complexity classes* that group problems of similar difficulty together. For example, if I gave you a list of n elements and asked you to find out whether it contained a 5, you could find out the answer in at most n steps. Hence, this problem falls into the complexity class P–the set of problems that can be solved in polynomial time.

It turns out, spin glass Hamiltonians are hard to figure out. So hard, in fact, that asking "What is the ground-state of some spin glass H?" is an NP-Hard optimization problem whereas asking the slightly easier "Is the ground-state energy $E_{gs} \leq 0$?" is still an NP-complete optimization problem [18]. Defining these computational complexity classes precisely is difficult without going on a long tangent in the theory of computation. In short, problems that fall into these classes are difficult to solve and extremely important. Loosely speaking, it takes expo-

nential time to solve them, but this is only a half-truth. By extremely important we mean two things: they are pervasive in real world applications and solving one NP-complete problem efficiently means you've solved all other NP problems efficiently too[1]. For example, the question, "Given a set of houses a UPS driver needs to visit, is there a route that uses less gas than he or she has in her tank?" is NP-complete. If a computer scientist found an efficient solution to this problem, then they automatically have an efficient solution to, say, the seemingly unrelated *clique problem.* Put simply, the clique problem asks "Given a graph, is there a node that has at least K edges?" In other words, we might be asking something like: does at least one friend on your Facebook have at least 1500 friends?

The connection between the Ising model and NP-complete/NP-hard questions made physicists begin to wonder: could Feynman's idea of efficiently simulating quantum systems via quantum systems apply here too? That is, could we build an Ising model machine whose natural evolution computes the answer to difficult optimization problems that arise in the every-day world? This lead to a number of papers that sought to formalize this notion [12; 18; 19; 20]. When the solution uses the adiabatic theorem (see Eq. 1.10) to solve difficult optimization problems, we call the computational model *quantum annealing.*

This brings us nicely to our next chapter: Quantum Annealing on the D-Wave 2000Q, one modern physical implementation of this idea.

[1]NP-complete is a subset of NP problems for which a solution to one NP-complete problem means you can find a solution to any NP problem from there in polynomial time. In general, NP-hard problems are not in the complexity class NP–as confusing as that is.

Chapter 2

Quantum Annealing on the D-Wave 2000Q

Summary

Here we develop the basic theory behind quantum annealing (QA) [12; 21], a special case process of adiabatic quantum computation (AQC) that was developed in light of the adiabatic theorem (see Eq. 1.10).

2.1 The Physics of Annealing

Adiabatic quantum computation (AQC) starts by preparing a system into the ground-state of some Hamiltonian that is easy to prepare. Then, the system is evolved adiabatically to some final Hamiltonian that encodes the solution to a computational problem. If the final Hamiltonian can encode any computational problem (that can be solved, of course), the process is universal and equivalent to the gate-based model [1] of quantum computation [12]. Due to the engineering

[1] By gate-based model, we mean quantum computation via unitary transformation–aka quantum logic gates. This model is much more similar to that of classical computation via circuit

difficulties associated with building a universal AQC, D-Wave took a slightly less general route, instead opting for a final Hamiltonian suited to solve optimization problems [1]. Due to its similarity to simulated annealing, and to help differentiate it from universal AQC, optimization-based AQC is known as Quantum Annealing (QA).

Concretely, the time-dependent Hamiltonian used on D-Wave's QA can be written in the form of a transverse-field Ising model (TFIM) Hamiltonian,

$$H(s) = \frac{A(s)}{2} \sum_{i}^{N} \sigma_i^x + \frac{B(s)}{2} \left(\sum_{i}^{N} h_i \sigma_i^z + \sum_{i<j}^{N} J_{ij} \sigma_i^z \sigma_j^z \right), \tag{2.1}$$

where $s = t/T \in [0, 1]$ for total anneal time T, and $A(s)$ and $B(s)$ are hard-ware defined functions that evolve the Hamiltonian from the ground-state of the first term to the ground-state of the second term (also described as going form an equal superposition state to the ground-state of the problem Hamiltonian in the computational basis, z). In particular, $A(s)$ is a monotonically decreasing function whereas $B(s)$ is a monotonically increasing function that together satisfy $A(0) \gg B(0)$ and $A(1) \ll B(1)$. These functions can be seen in Fig. 2.1, plotted with respect to the average thermal energy associated with the finite temperature of the environment enclosing the 2000Q chip.

Although the form of $A(s)$ and $B(s)$ cannot be altered, users have control over several advanced anneal features [21]. These features are introduced by a user-specified *anneal schedule* that defines a $s(t)$. A user defined anneal schedule must be a piece-wise-linear function that satisfies several hard-ware constraints. An example of a valid anneal schedule is shown in Fig. 2.2.

Looking at Fig. 2.2, we see that different portions of $s(t)$ are given different

boards that simulate classical logical gates.

[1] This is hardly as limiting as it sounds as a very large class of computational problems can be asked as an optimization problem.

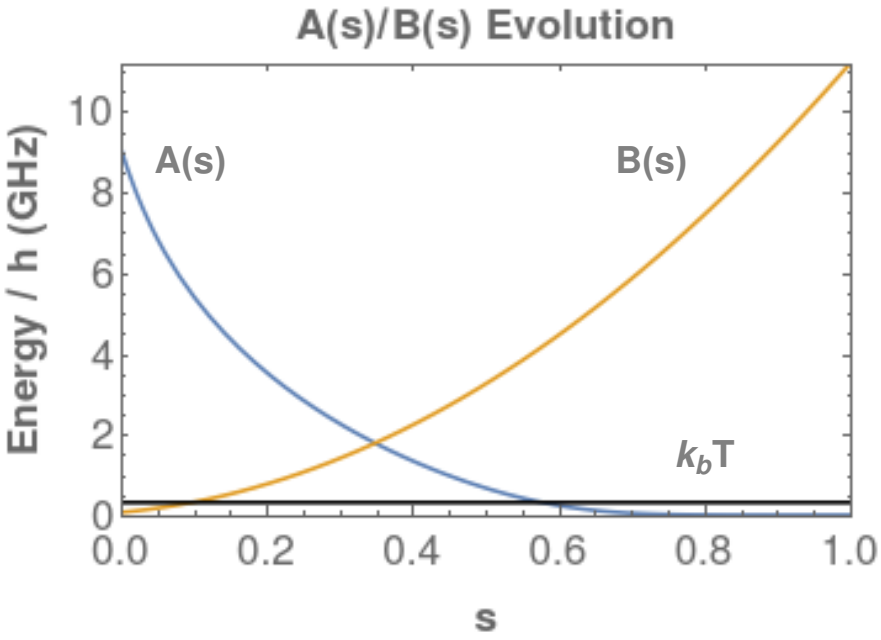

Figure 2.1: The functions, $A(s)$ and $B(s)$, defining the 2000Q's adiabatic evolution. The horizontal line shows the average thermal energy associated with the finite temperature of the chip, $T \approx 15mK$.

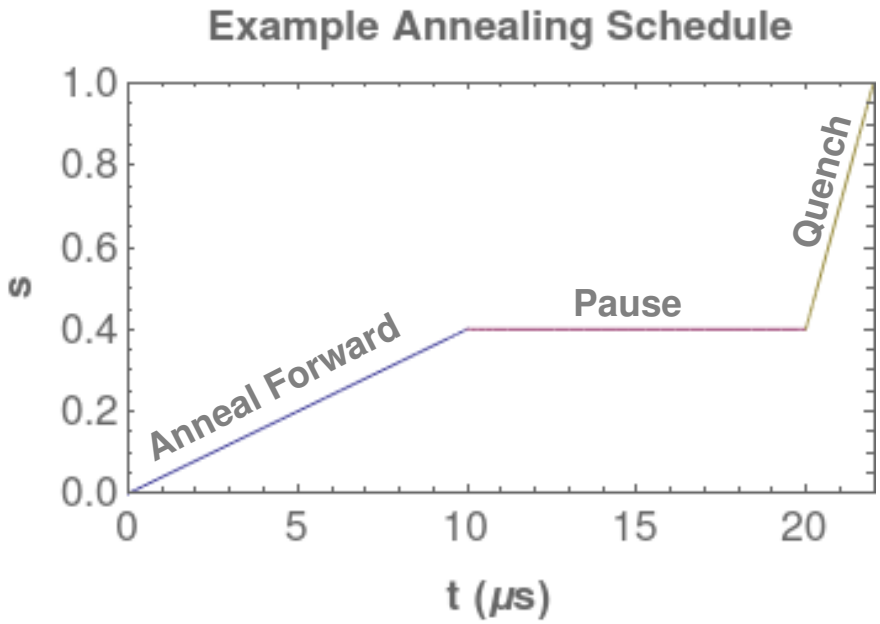

Figure 2.2: An example piece-wise-linear anneal schedule that defines the linear change in s as a function of t over an anneal on D-Wave's 2000Q quantum computer.

labels. This distinction is not entirely arbitrary–it has to do with how the slope of $s(t)$ at a point relates to the intended physics that should govern the change of the quantum state $|\psi\rangle$. The types of annealing schedules, along with their intended effects, can be summarized as follows.

1. A **Forward Anneal** is defined by a shallow positive slope. Ideally, this causes $H(s)$ to change slowly enough that $|\psi\rangle$ evolves adiabatically.

2. A **Reverse Anneal** is defined by a shallow negative slope. Ideally, $|\psi\rangle$ evolves adiabatically.

3. A **Pause** is defined by 0 slope. In a closed quantum system, this means that $|\psi(s')\rangle$ evolves solely due to the Hamiltonian $H(s')$. In an open quantum system, a pause allows thermal and quantum decoherence effects to play

a role over a longer time-span than normal. For example, more thermal excitations/ relaxations are probable in the case of a pause.

4. A **Quench** is defined by a steep positive slope. Ideally, this causes $H(s)$ to change quickly enough that $|\psi\rangle$ evolves according to the Sudden Approximation.

2.2 Submitting a Problem

Running a problem on D-Wave's 2000Q requires finding a representation of that problem encoded in Eq. 2.1, defining an appropriate anneal schedule in accordance with the desired evolution of $|\psi\rangle$, and finally, embedding the problem onto the native topology of the Chimera lattice that defines the connections available on the hardware, shown in Fig. 2.3–which may require adjusting the above two steps in some iterative process.

The process of finding an appropriate embedding, known as *minor-embedding*, is viewed in depth in several papers [22; 23; 24; 25], so we will only explain the idea and not discuss the details of any efficient algorithms to find them, like that of Cai, Macready, and Roy [25] available through D-Wave's pre-processing API [26]. A problem graph H is said to have an embedding onto the D-Wave Chimera lattice G if H is a minor of G. That is, if H can be constructed by deleting nodes and contracting edges of G. For example, the graph shown in Fig. 1.1 (the 1-d, 3 qubit graph with periodic boundary conditions) is a minor of 4 qubits in a single unit cell shown in Fig. 2.4. To see this, we will show how one can embed (short for finding a minor embedding) this 3 qubit problem into the 4 qubits 1, 2, 5, and 6 of the unit cell. One way is to contract the edge connected qubits 2 and 6, turning these two qubits into an effective qubit, q_{eff}. In this case, qubits 1, 5, and q_{eff} form the logical graph of Fig. 1.1. Of course, we could always contract as many edges

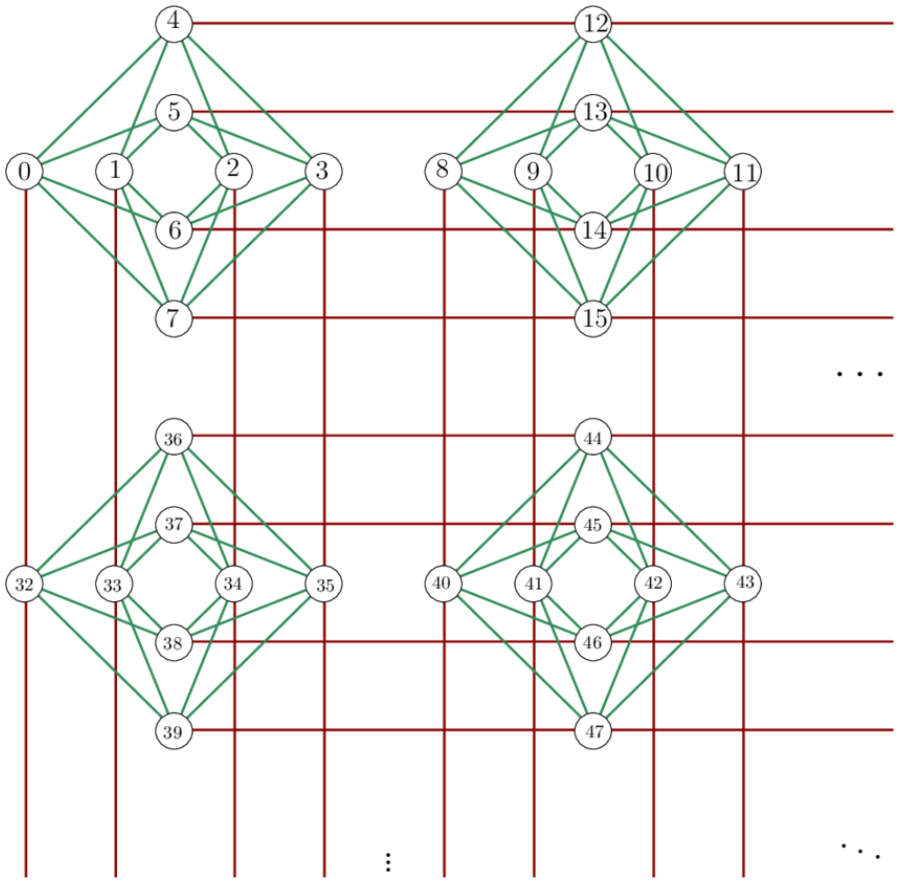

Figure 2.3: The upper-left corner of D-Wave's 2000Q Chimera lattice consisting of a 16×16 sparsely connected tiling of 8 qubit unit cells. Each unit cell is a complete bipartite graph. Image obtained from Calude [1].

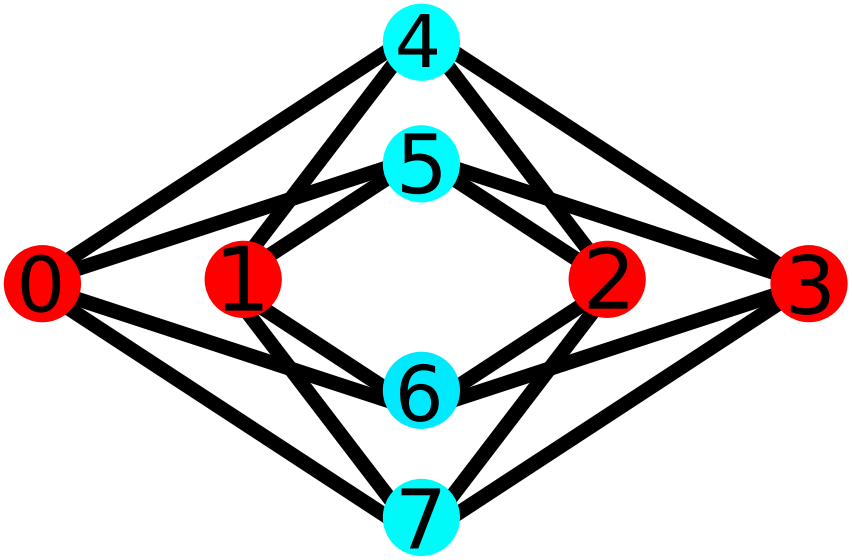

Figure 2.4: A graphical representation of a single 8 qubit unit cell on D-Wave's 2000Q Chimera lattice with the qubit coloring emphasizing the complete bipartite connectivity.

as we like to form one super q_{eff} whereby we use all 8 qubits of the unit cell to form the 3 qubit logical problem.

Obviously, these unit-cell couplers cannot actually be "contracted" on the device, so to actually perform this operation, one must instead adjust the weights in the Hamiltonian (Eq. 2.1) to produce the same effect. Going back to our previous example, if we wish to contract the edge connecting q_2 and q_6, we can instead "chain" them together. To do this, we add a strong ferromagnetic $J_{2,6} \ll -1$ penalty on the coupler between q_2 and q_6. This makes it very likely that the two qubits will always align. Hence, we can treat the two physical qubits as a single logical qubit in our problem, as long as we make sure to adjust the field-bias strengths (in this case we would apply half of the desired logical bias strength to each physical qubit) and do appropriate post-processing.

Before moving on, we'd like to address to natural questions that may arise from this idealized discussion of chaining to turn a logical problem in a physical

problem that works on the Chimera lattice. 1) What happens if two chained qubits actually point in opposite directions at the end of an anneal? How do we decide which direction the effective/logical qubit should be pointing pointing? 2) How high should the chaining coupling be with respect to the problem couplings, and what effect does different choices have on output statistics? For both of these questions, it is best to appeal to the thorough analysis that already exists in the literature [24; 25]. However, without running a problem otherwise, D-Wave pre- and post-processing routines heuristically decide the best way to handle these nuances with no user input.

Chapter 3

Our dwaveutils Software Package

Summary

The purpose of our dwaveutils package [27] is simple[1]. We wanted a single package that makes it easy to create an Ising Hamiltonian, submit the problem to D-Wave or a numerical quantum annealing routine, and post-process the data. In particular, we emphasized the ability to sweep over a large number of coupling values, as this problem came up time and time again with regular use of a D-Wave device, but was not easily supported with D-Wave Systems' Ocean API [26].

3.1 High Level Package Description

When constructing an Ising Hamiltonian, a user may choose to represent their problem in any number of different ways: an adjacency matrix, a Python dictionary, or a NetworkX graph[2]. The dwaveutils package is designed to be input-agnostic. Once a problem is represented by a certain data structure, all the class methods and output behavior behaves the same. That is, if Alice encodes her

[1]The source code can be found at github.com/naezzell/dwaveutils
[2]See their homepage: networkx.github.io/

problem as an adjacency matrix and Bob encodes his problem as a Python dictionary, they would still call the same exact method to queue their problem onto a D-Wave 2000Q processor. Further, they can both seamlessly go from running the problem on the D-Wave 2000Q processor to performing a numerical quantum annealing to performing direct diagonlization.

We enforce this behavior with an abstract base class (ABC) called ProbRep (aka Problem Representation). This construct allows us to design class attributes and methods without specifying an implementation. When we create a concrete class, inheriting ProbRep enforces a "reasonable behavior standard." For example, the class I use most often is DictRep (or Dictionary Representation). This class understands inputs in the form of a Python dictionary. If I failed to write a method that queued an instance of DictRep on the 2000Q annealer device, this would throw an error. Of course, once a concrete class is written, it can have its own methods that are not shared with ProbRep, but at minimum, it must have a working implementation of those methods specified in ProbRep.

Other than that, the code contains typical things: various helper functions contained in a dwavetools directory, scripts to run simulations that have been deemed useful to run over and over again, and tests.

3.2 Functionality I: Running Problems on D-Wave

Given the way the optimization process works on D-Wave quantum annealers, there are an infinite set of equivalent formulations of a problem. Even for a fixed graph topology, one can (theoretically) fine tune their parameters (h's and J's) as much as they like. Ideally, this wouldn't change the answer that D-Wave spits out, but the D-wave is not an ideal system, so it often does.

Because of this, users consistently run into the same problem: they want to

```
H1 = {(0, 0): 0.1, (1, 1): 0.1, (2, 2): 0.1, (0, 1): -0.1, (1, 2): -0.1, (0, 2): 0.1}
problem1 = DictRep(H = H1, qpu = 'dwave', vartype = 'ising', encoding = 'logical')
```

```
fig = plt.figure()
graph = problem1.visualize_graph()
plt.savefig("3qubit_problem.png")
```

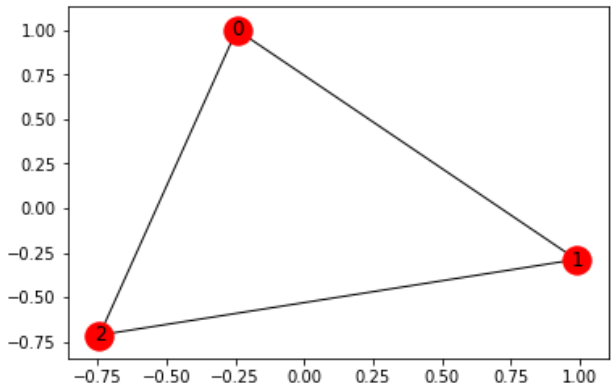

Figure 3.1: This code shows the creation of a logical encoding of a 3-qubit Ising problem with a triangular graph topology, along with the ease with which such a problem is visualized.

sweep over a fixed set of h and J hyper-parameters for an Ising model with a fixed graph topology. We will from now on refer to this as "hyper-parameter sweeping." Our code makes this process simple, but to understand it, we will first go over a couple of simpler examples that run only a single instance of a problem then get to the heart of what we are discussing.

In the first example, we queue a so-called "logical" Hamiltonian with actual numeric weights (Fig. 3.1). By this, we mean that we understand our problem may or may not have a native representation on the D-Wave 2000Q Chimera topology, and we allow the D-Wave pre-processing API to handle the minor-embedding process. The results of running this on an actual D-Wave 2000Q chip be seen in Fig. 3.2.

For contrast, we immediately run the same problem with the encoding parameter set to direct (Fig. 3.2). This throws an error on the D-Wave chip because

```
response1 = problem1.call_annealer() # this is the default OCEAN response
totaloutput = []
for energy, state, n in response1.data(['energy', 'sample', 'num_occurrences']):
    nthdata = [energy, state, n]
    totaloutput.append(nthdata)
print(totaloutput[0])
print(totaloutput[-1])
print(len(totaloutput))
```

```
[-0.4, {0: -1, 1: -1, 2: -1}, 1]
[0.20000000000000004, {0: -1, 1: 1, 2: -1}, 1]
246
```

Figure 3.2: The results of running the problem created in Fig. 3.1 on a D-Wave 2000Q chip.

```
H2 = {(0, 0): 0.1, (1, 1): 0.1, (2, 2): 0.1, (0, 1): -0.1, (1, 2): -0.1, (0, 2): 0.1}
problem2 = DictRep(H = H2, qpu = 'dwave', vartype = 'ising', encoding = 'direct')
try:
    response2 = problem2.call_annealer()
except:
    print("No three qubits are connected on Chimera with desired graph Topology.")
```

```
No three qubits are connected on Chimera with desired graph Topology.
```

Figure 3.3: Creates the same exact problem as Fig. 3.1 but with a direct graph encoding.

qubits 0, 1, and 2 do not have a triangular topology.

The next example in Fig .3.4 begins to show the actual power of our software. Here, we give each h and J coupling parameter a variable weight. This allows us to define a list of values we would like for these weights to take on. In particular, we collect all hyper-parameters that will be swept over in the parameters dictionary. By populating our problem with these parameters, we are essentially taking the Cartesian product over them all and insisting that a call the D-Wave quantum annealer try them all. The data from each run is saved to the problem's data parameter, so it is easy to view and post-process it afterwards as seen in Fig. 3.5. In this example, we use get_state_plot to show the distribution of final states produced by the sum total of all our trials thus far.

Next, we show just how powerful our hyper-parameter sweeping and post-

```
H3 = {(0, 0): 'h0', (1, 1): 'h0', (2, 2): 'h0', (0, 1): 'J1', (1, 2): 'J1', (0, 2): 'J2'}
problem3 = DictRep(H3, 'dwave', 'ising', 'logical')

# Set values for abstract parameters to take
h0 = np.linspace(0, 0.1, 2)
J1 = [-.1]
J2 = np.linspace(0, 0.1, 2)

# Create anneal schedules to try
sch1 = make_dwave_schedule('forward', s = 1, ta = 20)
sch2 = make_dwave_schedule('forward', s = 0.5, ta = 10, tp = 5, tq = 5)

#propogate these parameters to problem3
parameters = {'h0': h0, 'J1': J1, 'J2': J2, 'anneal_schedule': [sch1, sch2], 'num_reads': [1]}
problem3.populate_parameters(parameters)
data = problem3.call_annealer(cull = False)
data.head()
```

	J1	J2	anneal_schedule	energy	h0	num_reads	state
0	-0.1	0.0	[[0, 0], [20, 1]]	-0.2	0.0	1.0	(-1, -1, -1)
1	-0.1	0.0	[[0, 0], [20, 1]]	-0.2	0.0	1.0	(-1, -1, -1)
2	-0.1	0.0	[[0, 0], [20, 1]]	-0.2	0.0	1.0	(-1, -1, -1)
3	-0.1	0.0	[[0, 0], [20, 1]]	-0.2	0.0	1.0	(-1, -1, -1)
4	-0.1	0.0	[[0, 0], [20, 1]]	-0.2	0.0	1.0	(-1, -1, -1)

Figure 3.4: Creates the same exact problem as Fig. 3.1 but with variable weights. This makes it very easy to perform hyper-parameter sweeps over different h's and J's.

```
problem3.save_data('test.csv')
state_plot = problem3.get_state_plot(figsize = (12, 8), filename = 'test.png')
```

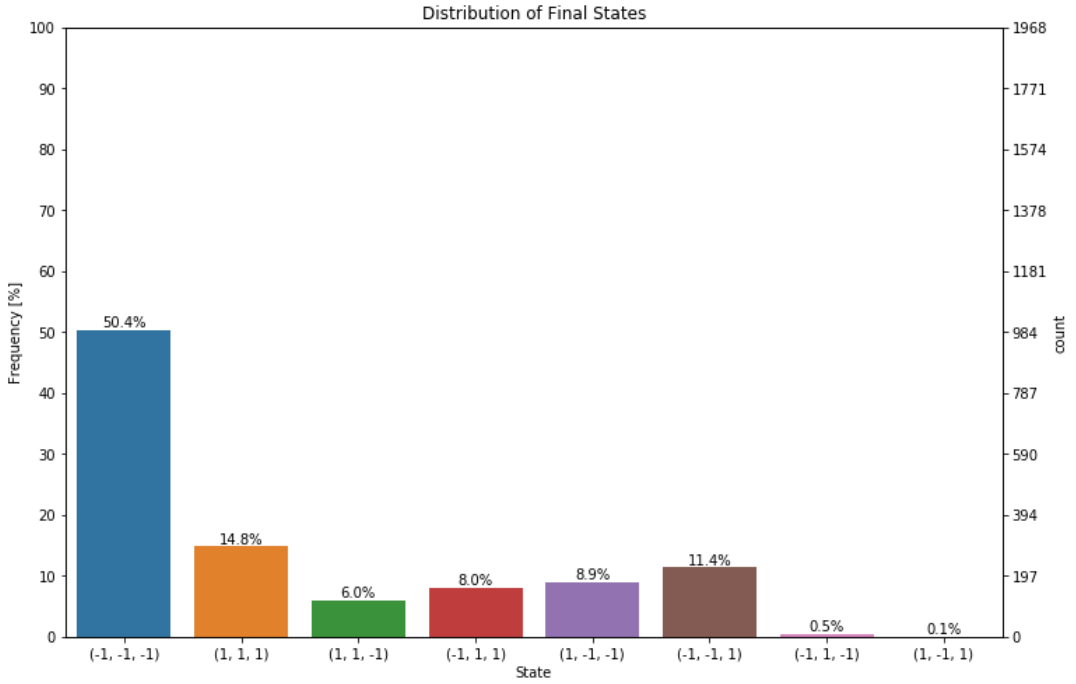

Figure 3.5: Keeping track of everything within the confines of a Python class means plotting bulk-results after running many trials can be made trivial. We illustrate this by plotting the states obtained from a D-Wave 2000Q device after trying out the parameter sweep of Fig. 3.4.

```
transH = {(0, 0): 'h', (1, 1): 'h', (2, 2): 'h', (0, 1): 'J1', (1, 2): 'J1', (0, 2): 'J2'}
problem4 = DictRep(transH, 'dwave', 'ising', 'logical')
# load in chip data (simplified csv file created with dwavetools)
heff_to_s = pd.read_csv("../dwaveutils/probrep/heff_to_s_DW_2000Q_2_June2018.csv")

#set h/ J vals
h = [0]
J1 = [-.1]
J2 = np.linspace(-0.4, 0.4, 2)
#set hxvals, find corresponding svals
hxvals = np.linspace(0, 0.4, 2)
hidxs = pd.Index(heff_to_s['heff'])
csv_svals = pd.Index(heff_to_s['s'])
s_to_hx = {}
svals = []
for hx in hxvals:
    s = csv_svals[hidxs.get_loc(hx, 'nearest')]
    svals.append(s)
    s_to_hx[s] = hx

#create anneal_schedules that correspond to quenching at s when heff = hx with various anneal lengths
atimes = [100]
schedules = [make_dwave_schedule('f', s, ta, 0, 1-s) for s in svals for ta in atimes]
```

Figure 3.6: This code shows how we set up the hyper-parameter sweep that corresponds to continuously varrying the effective transverse field strength of the D-Wave Ising Hamitlonian.

processing utilities can be by generating a transverse-field Ising quantum phase transition plot. This is quite the mouthful, and we will have much more to say about it in the next chapter when we discuss applications of our software. For now, we will only discuss the steps in the process. In Fig. 3.6 we set up the problem by associating the appropriate s' value to the desired effective transverse field strength (given by $A(s)/B(s)$). In Fig. 3.7 we show how this problem is passed to a D-Wave 2000Q device and print a few trial values. Finally, in Fig. 3.8, we use a get_ferro_diagram to plot the transverse-field quantum phase plot. In truth, we didn't do nearly enough trials in this example code to save time, so this plot is not illustrative of any actual results–it's meant merely to show what can be done.

Finally, we show a couple miscellaneous features that come up from time to time in Fig. 3.9. Suppose you don't want to trust the native D-Wave API to perform the minor-embedding process for you. While that's great, this can create a clash with hyper-parameter sweeping. In the past examples, we gave all couplings that

```
parameters4 = {'h': h, 'J1': J1, 'J2': J2, 'anneal_schedule': schedules, 'num_reads': [1]}
problem4.populate_parameters(parameters4)
data4 = problem4.call_annealer(cull = False, s_to_hx=s_to_hx)
print(data4.tail())
problem4.save_data("dwave_trans.csv")
```

```
      J1   J2                    anneal_schedule  energy    h   hx  \
979 -0.1  0.4  [[0, 0], [100, 0.44], [100.56, 1]]    -0.4  0.0  0.4
980 -0.1  0.4  [[0, 0], [100, 0.44], [100.56, 1]]    -0.4  0.0  0.4
981 -0.1  0.4  [[0, 0], [100, 0.44], [100.56, 1]]    -0.4  0.0  0.4
982 -0.1  0.4  [[0, 0], [100, 0.44], [100.56, 1]]     0.2  0.0  0.4
983 -0.1  0.4  [[0, 0], [100, 0.44], [100.56, 1]]     0.2  0.0  0.4

     num_reads         state
979        1.0    (-1, 1, 1)
980        1.0   (-1, -1, 1)
981        1.0    (1, 1, -1)
982        1.0     (1, 1, 1)
983        1.0  (-1, -1, -1)
```

Figure 3.7: Here, we submit our desired problem to a D-Wave 2000Q device and print out a few trial values.

we wish to vary together always with the same value the same name. For example, in $H_{embedding}$, we want qubit 0 and qubit 4 to both take on the same value of "h" regardless of which value that ends up being. But in this example, we are creating a 3 qubit triangle by embedding it into a 4 qubit square, a topology that does exist on the hardware. This means that weights on qubits 5 and 1 which together make up a single effective qubit should be halved to keep the energy effects the same. If we gave them a different name, the hyper-parameter sweeping would naively take a Cartesian product over all values–not producing the desired behavior. Instead, we added a feature that allows you to pre-pend a multiplier (or divisor) after a parameter as we have done here. Hence, it varies always with those values given only an h, but it always takes half it's value. Further, we show a simple feature where you can save the embedding for future use at the very end.

```
plt = problem4.get_ferro_diagram('J2', 'hx', 'J1')
plt.savefig("dwave_trans_contour.eps", dpi=300)
```

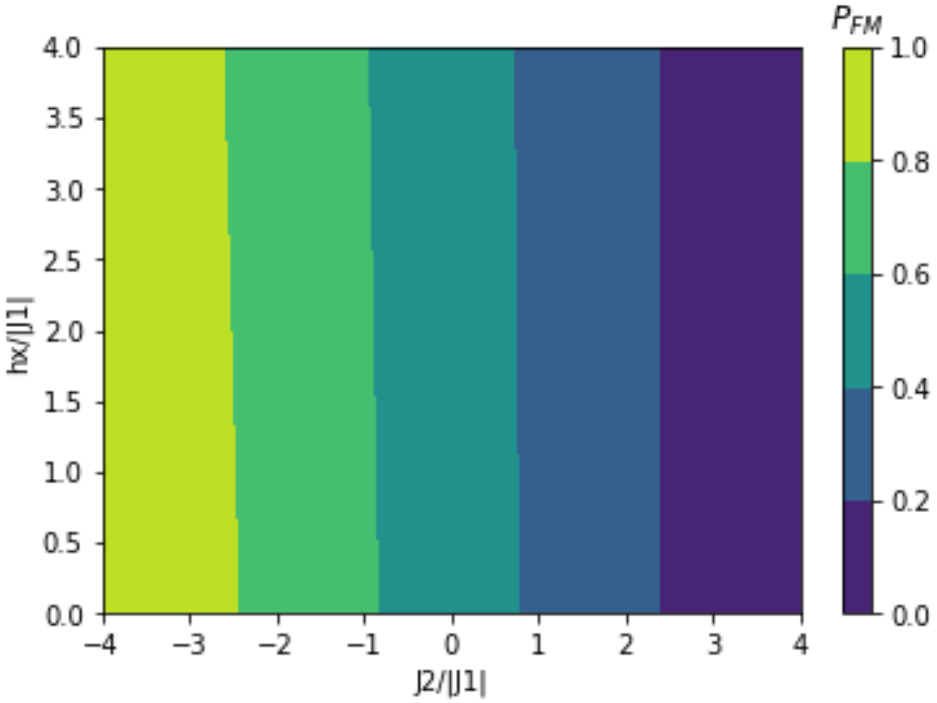

Figure 3.8: Finally, we use each saved value from our parameter sweep to create a transverse-field quantum phase transition plot. This is an example to show the power, so not enough trials were done to draw actual conclusions. We will have more to say about this in the next chapter.

```
Hembedding = {(0, 0): 'h', (4, 4): 'h', (1, 1): 'h/2', (5, 5): 'h/2', (0, 4): 'J2',
              (0, 5): 'J1', (1, 4): 'J1', (1, 5): 'J3'}
emproblem = DictRep(H = Hembedding, qpu = 'dwave', vartype = 'ising', encoding = 'direct')
emproblem.visualize_graph()
emproblem.save_config('square_embedding')
```

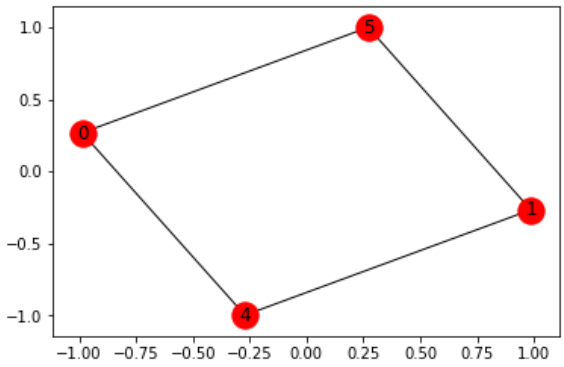

Figure 3.9: This code shows various miscellaneous features: the ability to create one's own custom embedding, to alter the variable weights by a multiplier, and to save the Ising problem for future uses.

3.3 Functionality II: Running Numerical Quantum Anneals

Being able to check the answer from a D-Wave run is invaluable. Our numerical utilities allow for both direct diagonalization and numerical quantum annealing. We use the Python QuTiP library [28; 29] as the base for these operations, but we've created our own numerical quantum annealing routine that follows the D-Wave quantum annealing theory of Sec. 2.1 closely. We've also tested augmenting the D-Wave algorithm in physically NISQy[1] ways which has lead to potential performance improvements[2].

Actually running a numerical anneal is simple: set up a problem as we've shown Sec. 3.2 but flag is as "numerical", create a numerical anneal schedule, and call

[1] As in, possible to engineer in the near future by not being far from what is already done.

[2] We'll have more to say about that soon in Chapter 4: Forward-reverse error-mitigation annealing. We think the answer is yes.

```
# Form a dictionary representation of a problem and create an anneal_schedule
dictparams = {(0, 0): 0.1, (1, 1): 0.1, (2, 2): 0.1, (0, 1): -0.1, (1, 2): -0.1, (0, 2): -0.1}
problem = DictRep(H = dictparams, qpu = 'dwave', vartype = 'ising', encoding = "logical")
ta, sa, tp, tq = 10, 0.4, 5, 0.6
T = ta + tp + tq
schedule = make_numeric_schedule(.1, **{'ta': ta, 'sa': sa, 'tp': tp, 'tq': tq})
times = schedule[0]
svals = schedule[1]

plt.plot(times, svals)
plt.title("Anneal Schedule")
plt.show()
```

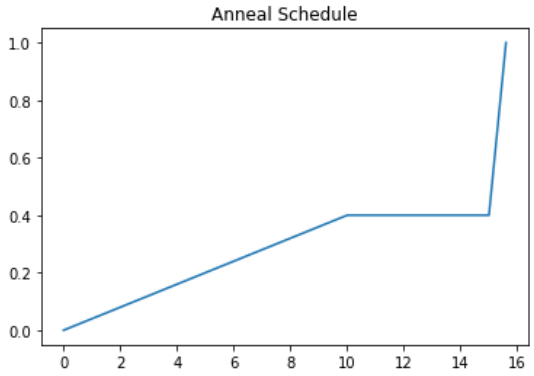

Figure 3.10: This code shows the creation of a numeric Ising Hamiltonin which now requires explicitely making a numeric anneal schedule.

the numerical anneal. For completeness, we will go through a detailed example that shows just how exactly our internals work, but all these steps are unnecessary when actually using the package.

First, we define the problem as shown alongside a numeric anneal schedule as shown in Fig. 3.10. Internally, this numeric anneal schedule is used to create the numeric representation of the time-varying Ising Hamiltonian as seen in Figs. 3.11 and 3.12.

After setting up the numeric Ising Hamiltonian, it's simple to utilize QuTiP built-in function to solve the time-varying Schr"ødinger equation as seen in Fig. 3.13. This can then be compared to direct diagonzlation of the final Hamiltonian as seen in Fig. 3.14.

```
# Create H(t)
processor_data = loadAandB()
sch_ABfuncs = time_interpolation(schedule, processor_data)
A, B = sch_ABfuncs['A(t)'], sch_ABfuncs['B(t)']
Hs = get_numeric_H(problem)
HX, HZ = Hs['HX'], Hs['HZ']

# Define H(t)
H = lambda t : A(t)*HX + B(t)*HZ

# Define list_H for QuTiP
list_H = [[HX, A], [HZ, B]]
```

Figure 3.11: This shows the "internals" of creating our discretized numeric Hamiltonian that is used for direct numerical diagonalization and QuTiP-assisted quantum annealing.

```
plt.plot(times, A(times))
plt.plot(times, B(times))
plt.title("A(t) and B(t)")
plt.ylabel('E/h (GHz)')
plt.xlabel('t')
plt.show()
```

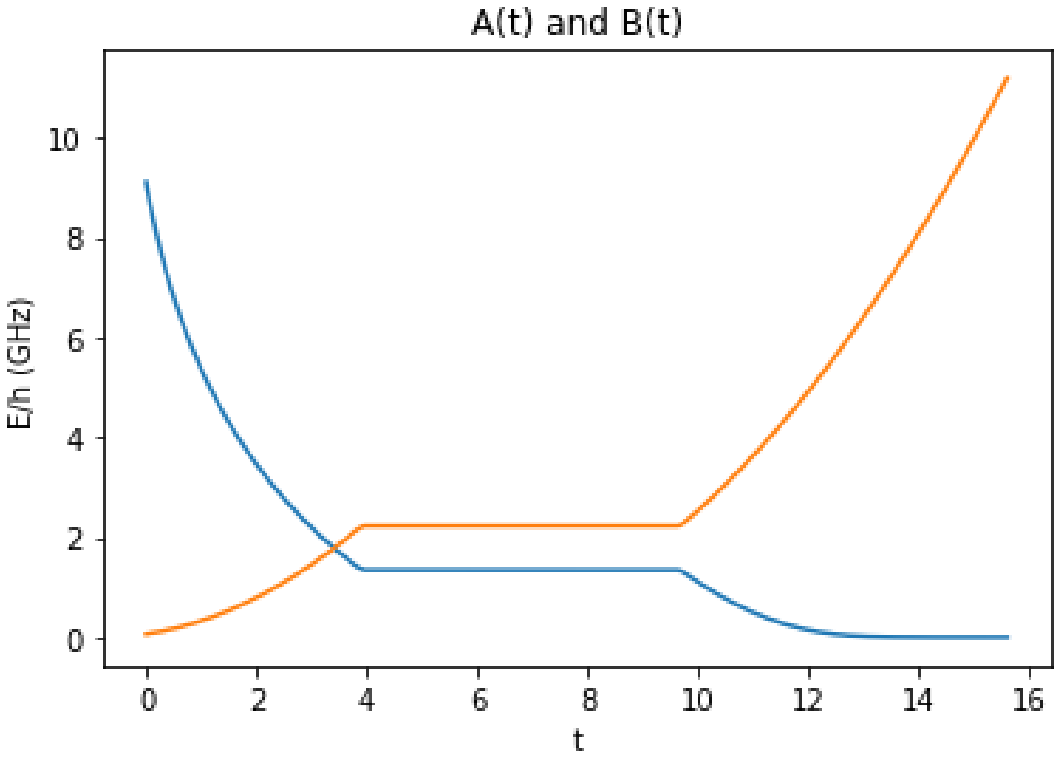

Figure 3.12: This code gives a clear picture of what we mean by "numeric anneal schedule." In particular, it's just the entire Hamiltonian of Eq. 2.1 adjusted by a discretized list of $A(t)$ and $B(t)$ weights at each time t during the anneal.

```
results = qt.sesolve(list_H, H(0).groundstate()[1], times)
amps = np.array([abs(results.states[-1][i])**2 for i in range(8)])
plt.bar([0, 1, 2, 3, 4, 5, 6, 7], amps.flatten())
plt.show()
```

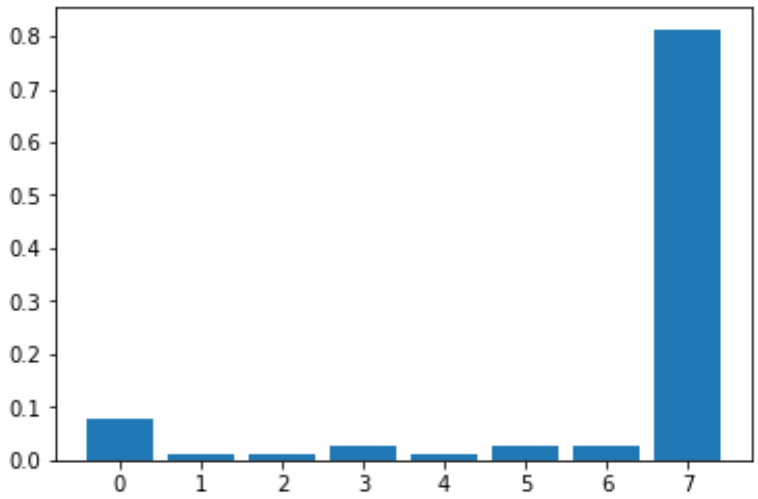

Figure 3.13: Here, we perform the numeric quantum anneal using QuTiP's built-in Schr"ødinger equation solver and plot the distribution of states.

```
# exact diagonlization of ground-state at the end
gs = H(T).groundstate()[1]
gs_amps = np.array([abs(gs[i])**2 for i in range(8)])
plt.bar([0, 1, 2, 3, 4, 5, 6, 7], gs_amps.flatten())
plt.show()
```

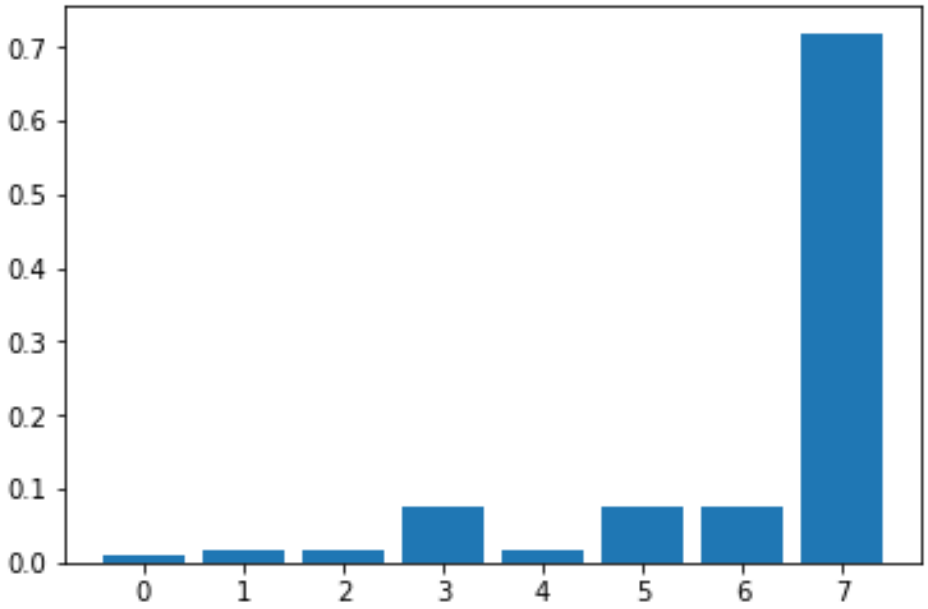

Figure 3.14: Here, we directly diagonalize the final QA Hamiltonian to compare it to the results of Fig. 3.13.

Chapter 4

Case Study I: Simulations of Transverse-field Ising Hamiltonians

Context and Summary of Investigation

In Summer 2018, I completed an internship at Oak Ridge National Laboratory (ORNL) under Dr. Travis Humble[1] and Paul Kairys[2]. They had recently come across an interesting paper by Xia, Bian, and Kais entitled "Electronic Structure Calculations and the Ising Hamiltonian." [30]. In short, this paper showed how to convert a quantum chemistry problem into an Ising problem: opening the door for quantum chemistry calculations on D-Wave quantum annealers.

It turns out, there is a pretty large qubit cost (overhead) in doing this conversion. But in summer 2018, the D-Wave 2000Q-2 had 2048 working qubits while the best gate-model architecture (IBM's Q20 Tokyo) had a qubit number of 20 [4].

[1]The director of the Quantum Computing Institute at ORNL

[2]Travis's newest graduate student who actually began working only about a week before I arrived

Could the D-Wave 2000Q outperform the gate-model that is theoretically more "natural" for quantum chemistry calculations? If it's easier to produce high qubit quantum annealing (QA) chips[1], could QA be the more promising NISQ era technology?

To make this comparison as concrete as possible, Paul and I decided to mirror each others calculations. He would submit problems to IBM's device, and I would work on the D-Wave front. As a proof-of-principle, we wanted to begin by showing that both devices could get the same answer with the simplest "non-trivial" problem imaginable: a 3 qubit Ising problem. To make comparison interesting and more relatable to our eventual goal of simulating interesting physics, we wanted to use these NISQ devices to generate a quantum phase transition plots. More than that, being able to effectively sample the Ising Hamiltonian with a transverse field present would reduce necessary ancilla qubits in quantum chemistry calculations. Ideally, this would let us simulate larger molecules as well. This proof-of-principle first step, however, ended up being an interesting exploration in and of itself, showing that some D-Wave features do not work as advertised (at least if used naively).

4.1 Statement of the Problem

In this work, we will focus on simulating quantum phase transitions (QPTs) of the 3-qubit Ising problem shown in Fig. 4.1. In particular, we will are looking for changes in the magnetic properties of our system through continuous changes in $\Gamma_{eff} = A(s)/B(s)$ (see Eq. 1.14), the $h_i's$ and the $J_{ij}'s$ as in Eq. 2.1. By magnetic properties, we are specifically referring to the probability that the ground-state (which, in general, is degenerate) is ferromagnetic, denoted $\mathcal{P}_{FM}$. While it is more

[1] ...and more importantly, keep improving this number

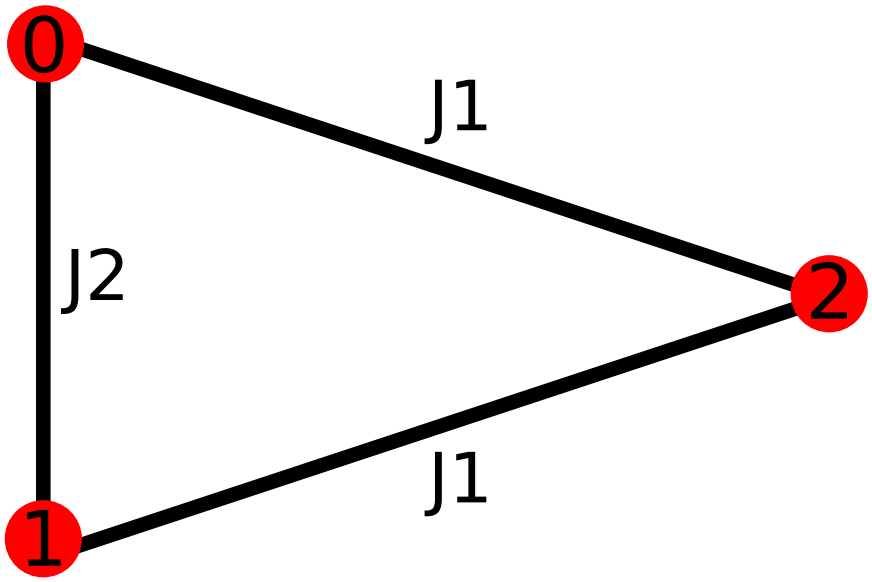

Figure 4.1: The 3-qubit Ising problem we simulate on D-Wave's 2000Q quantum computer.

informative, in general, to find an expectation of the magnetization, for our 3-qubit system, a state is either ferromagnetic or contains a single spin-flip (i.e. 2 up and 1 down or 1 up and 2 down). Hence, $\mathcal{P}_{FM}$, aside from the sign which is not important here, contains all the information about the magnetization.

More specifically, we will create contour plots to display QPTs for Longitudinal-field Ising Models (LFIM) and Transverse-field Ising Models (TFIM). Making contact with Eq. 2.1, the LFIM corresponds to an anneal all the way to $s' = 1$ with specified h_i, J_1, and J_2 values whereas the TFIM corresponds to an anneal to some $0 < s' < 1$ such that $\Gamma_{eff} = A(s')/B(s')$ is the effective transverse-field strength with $h_i = 0$ and J_1, J_2 specified. By fixing $J_1 < 0$ to a ferromagnetic value, we can see the effect of introducing frustration by tuning J_2 from a ferromagnetic coupling $J_2 < 0$ to an anti-ferromagnetic coupling $J_2 > 0$. Furthermore, in the LFIM, increasing h_i increases the chance that all the qubits will align with the external field whereas in the LFIM, increasing Γ_{eff} decreases the chance that qubits will

be aligned (scrambling analogous but not exactly the same as finite temperature fluctuations).

Though there has been recent success on solving a much larger $8 \times 8 \times 8$ cubic-lattice Ising problem [31], this problem is interesting and useful for a variety of reasons. In the near-term quantum hardware era, one of the major obstacles is predicting whether or not a problem will reproduce exact results *a priori*. We wish, among other things, to begin probing this question with simple test-problems that begin uncovering the relationship between a problem's parameters and its success-rate on current hardware so that non-domain experts can get an estimate on the success-rate of a posed problem. In addition, this problem was chosen because:

1. Being an Ising problem from the start, we are testing D-Wave architecture exactly on the types of problems it was built to solve without the added complexity of interpreting the success in the context of some other mathematical problem.
2. Our problem exhibits frustration due to competing objectives, giving rise to non-trivial quantum phase-transition plots. In addition, encoding the problem on a D-Wave requires utilizing advanced anneal features (i.e. quench and potentially pause).
3. This problem is small enough to run using the variational quantum eigensolver (VQE) method [32] on existing gate-based architectures such as Rigetti and IBM. Hence, comparisons between the different models of hardware can be made.
4. Furthermore, being small means numerical simulations are not only tractable, but fast enough to run quickly. This makes it trivial to check our answers with numerics and to obtain adequate statistics.

4.2 Methods[1]

We performed three types of simulation: (1) idealized numerical simulations, (2) closed-system numerical simulations, and (3) D-Wave 2000Q device simulations.

4.2.1 Idealized Numerical Simulations

In the ideal limit that D-Wave's device perfectly samples the ground-state of the desired Hamiltonian, then finding $\mathcal{P}_{FM}$ simply requires solving the time-independent Schrödinger equation by diagonlization (see Eqs. 1.2 and 1.3). Our algorithm was written in Mathematica and works as follows:

1. Loop over parameters $J_1 = -1$, $J_2 \in [-4, 4]$ and h_i (or $\Gamma_{eff} = A(s)/B(s)$) $\in [0, 4]$ for the LFIM(TFIM).
2. Construct ideal Hamiltonian (basically Eq. 1.14 without $A(s)$ and $B(s)$ as in Eq. 2.1) and diagonalize it.
3. Find the ground-state and compute the probability that it is ferromagnetic, $\mathcal{P}_{FM}$.

4.2.2 D-Wave 2000Q Simulations[2]

As discussed in Sec. 2.2, a particular embedding strategy must be chosen. In our work, we tried manually embedding our triangular graph onto a square graph than can be formed on a particular unit cell, choosing parameter values in the range of $[-0.4, 0.4]$ rather than $[-4, 4]$ to allow chained qubits to have a stronger coupling of $J = -1$, as this is the max coupling strength. However, doing a manual embedding

[1] Throughout, I will refer to the numerical simulations as having been done in Mathematica. This is true, as I originally simply adapted some of Travis's code for numerics. However, I used my hyper-parameter features for D-Wave submissions. In addition, I decided I wanted more flexibility in the numerics, so as discussed before, my own Python package could do all of this now. Redoing things would be a waste of time, though.

[2] As seen in Sec. 3.2, doing this type of hyper-parameter sweeping is simple with our package. This project was the motivation for those features.

gave the same results as allowing D-Wave's internal algorithms to automatically embed and adjust weights.

With these detail aside, all testing was done using D-Wave's native embedding algorithms and full-tiling onto the entire chip. Our simulation routine on the quantum computer went as follows:

1. Choose a set of parameters h_i, J_1, J_2, and s' (essentially choosing Γ_{eff}).
2. Anneal adiabatically to s' over $20\mu s$ then quench over $(1-s')\mu s$ (the fastest time) to $s = 1$ to sample $H(s')$. Note, for the LFIM, $s' = 1$, so only the adiabatic evolution matters.
3. Measure distribution of states at $s = 1$ and determine whether it is ferromagnetic.
4. Repeat these steps (both in discretization of parameters and number of samples for each set of parameters) until convergence of contour plot is obtained.

4.3 Results

4.3.1 Idealized Numerical Simulation Results

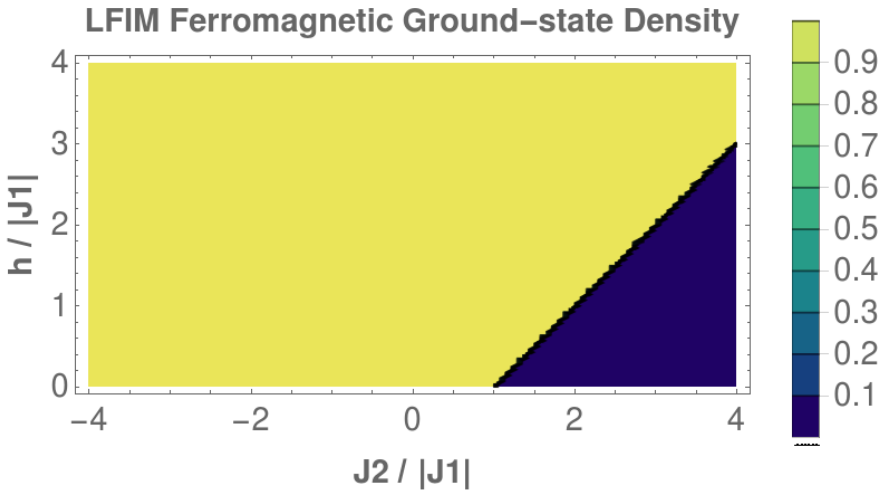

Figure 4.2: Quantum phase transition plot of idealized numerical simulations of longitudinal-field Ising model showing the probability that the ground-state of Fig. 4.1 is ferromagnetic for a given set of field-bias and coupling parameters.

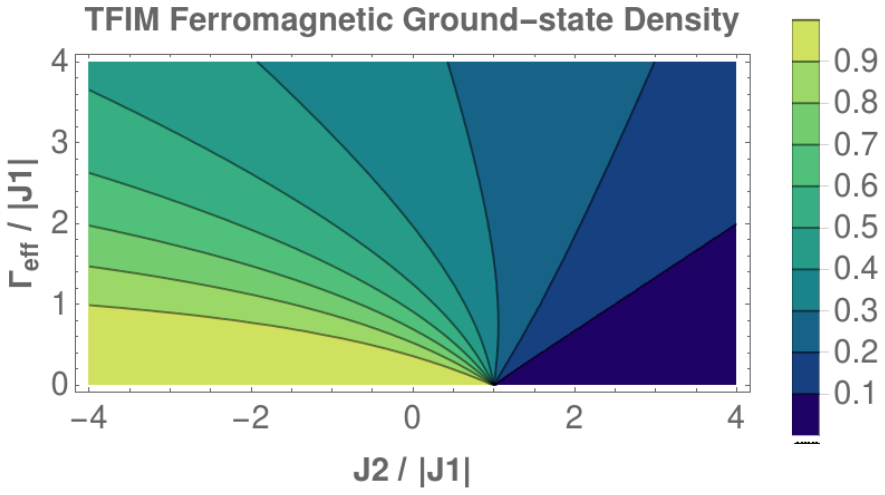

Figure 4.3: Quantum phase transition plot of idealized numerical simulations of transverse-field Ising model showing the probability that the ground-state of Fig. 4.1 is ferromagnetic for a given set of field-bias and coupling parameters.

4.3.2 D-Wave 2000Q Simulation Results

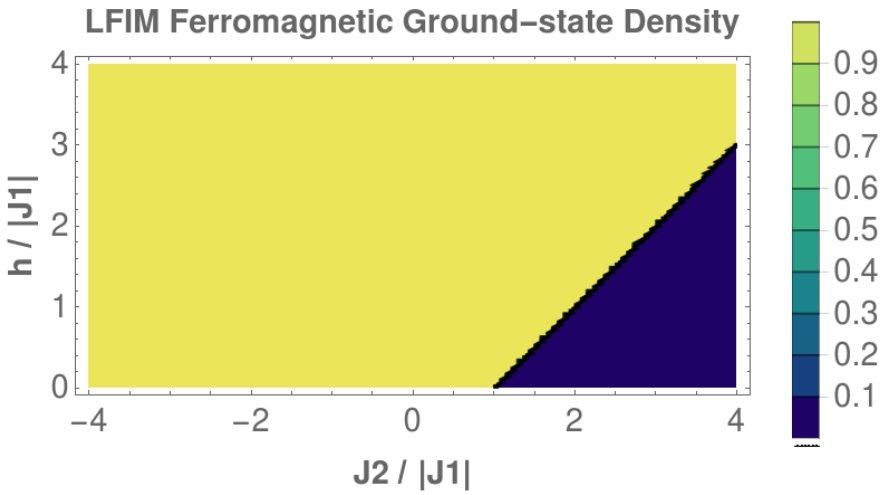

Figure 4.4: Quantum phase transition plot of D-Wave 2000Q simulations of longitudinal-field Ising model showing the probability that the ground-state of Fig. 4.1 is ferromagnetic for a given set of field-bias and coupling parameters.

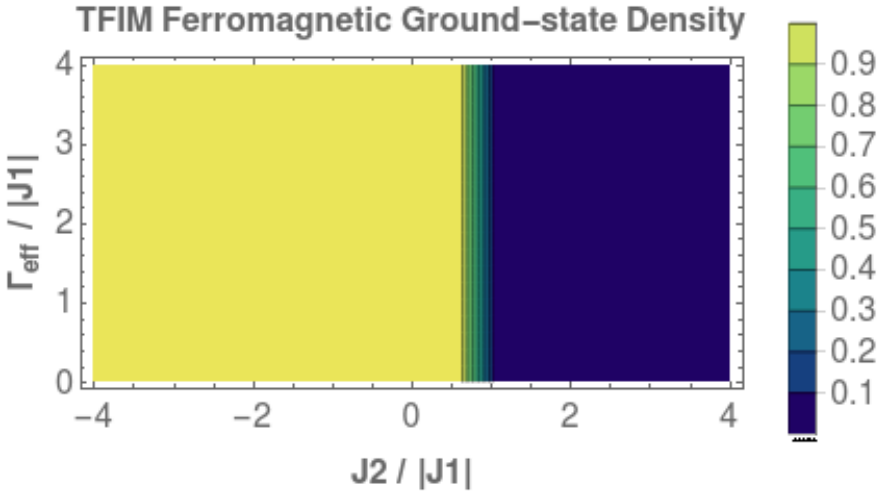

Figure 4.5: Quantum phase transition plot of D-Wave 2000Q simulations of transverse-field Ising model showing the probability that the ground-state of Fig. 4.1 is ferromagnetic for a given set of field-bias and coupling parameters.

4.4 Conclusion

From plots 4.2 and 4.4, we see that D-Wave's 2000Q quantum computer is very capable of simulating our 3-qubit LFIM. However, plots 4.3 and 4.5 show that the same cannot be said of the TFIM; in fact, the D-Wave simulation shows a QPT plot identical to idealized numerical calculations of a LFIM with $h_i = 0$, showing that Γ_{eff} played no role. Our future work will explore whether the TFIM can be simulated at all by lowering the Bohr frequency even further (Eq. 1.5 and Eq. 1.8) or whether this just makes thermal effects too dominant. In addition, we are interested in using the reverse-anneal feature in a Marokovian process to see if this could potentially make a difference. Whatever the case, any attempt to lower ancilla qubit requirements of quantum chemistry calculations on D-wave cannot rely on the quench feature.

4.5 Acknowledgements

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internship program.

Further, I would like to thank Dr. Travis Humble and Paul Kairys for allowing me the opportunity to work with them as well as for all their guidance and support.

Chapter 5

Case Study II: Forward-reverse Error-mitigation Annealing

Context and Summary of Investigation

Put succinctly, quantum annealing is a global search heuristic for a minimum. In classical heuristics, analogous routines are improved by local refinements, but due to the no-cloning theorem, such local improvements are not possible in traditional QA [33; 34]. Various alternative quantum annealing routines were proposed [35], and in 2017, D-Wave announced their solution: reverse (quantum) annealing[1] [33].

In summary, forward annealing explores the energy landscapes globally while reverse annealing does so locally. Our investigation asked a simple question: what happens if we try to combine them to get the "best of both worlds?" The end result was a new annealing protocol: forward-reverse error-mitigation (FREM) annealing. We guide you through our investigation by carefully defining FREM, outlining a few numerical experiments we ran with it, commenting on its poten-

[1] This is why we often refer to "standard quantum annealing" as simply forward (quantum) annealing.

tial, and discussing possible future directions. Ultimately, FREM has shown the potential to reduce errors compared to forward or reverse annealing alone without the cost of ancilla qubits. Further, it has uncovered a potentially fruitful line of investigation: graph introspection via critical qubits.

5.1 Defining FREM

To give a good definition of FREM, we're going to have to do a better job of defining reverse annealing than we did in Sec. 2.1.

5.1.1 Revisiting Reverse Annealing

In Eq. 2.1, we defined the Hamiltonian that D-Wave uses, and we saw it's of the form

$$H(s) = \frac{A(s)}{2} H_x + \frac{B(s)}{2} H_z, \tag{5.1}$$

where $s = t/T \in [0, 1]$, $H(0) = H_x$, and $H(1) = H_z$ based on the behavior the $A(s)/B(s)$ "control function." We saw that in forward annealing, the job of H_x is to put the system in an easily achievable ground-state (i.e. an equal superposition in the computational basis) and that it is H_z that actually encodes our optimization problem of interest. Operationally, forward annealing means starting out with $s = 0$ and ending with $s = 1$ over some anneal length T_f.

As the name implies, reverse annealing goes in the opposite direction. Operationally, reverse annealing

1. starts out with $s = 1$ at $t = 0$ and a classical state (typically a guess of the ground-state from a previous forward anneal)
2. reduces s to $0 < s' < 1$ over time T_r (anneals backwards)
3. forward anneals from $s = s'$ to $s = 1$ over time T_f

4. and ends with $s = 1$, $t = T_r + T_f$, and a possibly different classical state readout.

In other words, it does a local exploration of the energy landscape around the point corresponding to the initial classical bit-string fed as a guess for the ground-state. Ideally, it would return the original guess if it was correct and only a different bit-string if the new answer has lower energy.

In actual uses, it is not that simple. As the D-Wave White Paper points out [33], there is typically a problem-dependent "Goldilocks" region. In other words, if you anneal too far backwards (i.e. make s' too small), you run the risk of "forgetting" anything about your input guess, performing a standard forward-anneal in disguise. But if you don't go far enough (i.e. make s' too large), you aren't allowing the system to perform a local search at all–it simply spits the original answer back out every time.

5.1.2 The FREM Algorithm

Forward-reverse error-mitigation (FREM) annealing is an annealing protocol that performs forward and reverse annealing on a single Hamiltonian simultaneously. For this to make any sense in the context of D-Wave annealing, we must partition the qubits that make up our Ising Hamiltonian into two classes: those that will be forward annealed (say, team F) and those that will be reverse annealed (team R). Of course, a typical problem will also include some couplings that connect two qubits that are assigned to the different teams. We call these couplings *mixed*. An example partition is shown in Fig. 5.1. To actually perform FREM, these mixed edges must be assigned to one team or the other. Though we've yet to prove it, we suspect assigning each mixed edge to team R is the most effective way to reduce error. One of our goals, of course, is to see whether this is true.

Recapitulating, we can define FREM annealing operationally. To perform

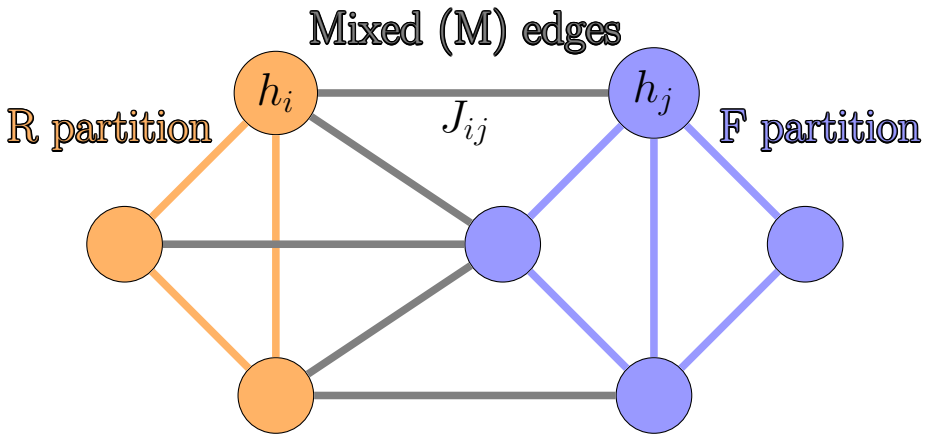

Figure 5.1: An example of a partition of an Ising Hamiltonian represented as a graph. In this case, we have not placed any of the mixed edges into either team R or F yet. A real partition must ultimately decide to which partition each of these edges belongs.

FREM, you must have an Ising Hamiltonian, a partition (generated from a scheme of your choice), a forward anneal schedule, a reverse anneal schedule, and a bit-string that encodes the initial state assigned to the qubits in team R. Given these, the job is straightforward: simply forward anneal team F while reverse annealing team R. Since pause is a valid operation even on current D-Wave devices, it is not necessary even that the two partitions finish annealing at the same time.

One thing we've glossed over is a good choice of the initial classical state for the team R qubits. Our choice in this work is simply to perform a forward anneal on the entire system first. In any given FREM anneal, we project the predicted ground-state of the forward anneal onto the qubits in team R.

Put in algorithmic steps, the FREM algorithm is:

1. input H_z; annealing schedule (T_f, T_r and s'); partitioning scheme
2. forward anneal (FA) H over time length T_f
3. reverse anneal (RA) starting with most probable classical state from FA over time length $T_r + T_f$

4. perform FREM annealing on each partition.

At the end, our algorithm dumps out two files: one that stores the raw data and one that summarizes all the major results we currently know to ask. In particular, the summary file keeps track of the following information

1. input data
2. number of FREM partitions tried
3. forward annealing (FA) ground-state probability (gsp)
4. reverse annealing (FA) gsp
5. the best FREM annealing gsp
6. average FREM gsp
7. percentage of FREM partitions that outperformed FA/RA gsp
8. the distribution of partition sizes for those FREM partitions that outperformed FA/RA gsp
9. "critical qubit" information.

It is somewhat obvious why we are interested in (1 - 5). This data answers the question: is it possible for FREM to improve ground-state sampling compared to the best available methods on quantum annealing hardware? Next, (6) and (7) help address: if so, how likely is that to occur if I choose a random partition? Datum (8) helps address the question: is there an ideal way to partition an arbitrary Hamiltonian? While this is a pretty coarse way to do so, we actually prefer this as it is typically not possible to ask extremely granular questions about spin-glasses. Datum (9) is slightly more complicated than the rest. Our hope is that investigations with (9) will help lead the way in answering granular questions about spin-glasses by allowing for "graph introspection," but we'll address it more thoroughly in Sec. 5.3.

5.2 Sidon Set Complete Graph (SK_n) Tests

5.2.1 The Sidon Spin Glass Hamiltonian

To test the efficacy of FREM as an error-mitigation protocol, we wanted to perform experiments on a "typical Ising Hamiltonian." One such useful notion is the quantum spin-glass discussed in Sec. 1.2.5 since answering a question about a spin-glass is at least as hard as answering the question on any other Ising Hamiltonian. To generate a spin-glass, one need only randomly assign J_{ij} coupling values from a Gaussian distribution with mean 0 and variance 1 while making all $h_i = 0$.

Unfortunately, assigning random real numbers to these couplings is generally a bad idea. For one, a computer will always truncate a real number due to memory constraints. Worse, even floating-point precision is lost once these coupling values are actually used on the D-Wave device due to hardware constraints (i.e. only the first 3 or 4 decimal places, in general, will matter). To avoid an ambiguity between the logical problem submitted and the physical problem that is actually solved using D-Wave (or even numerics), we instead randomly picked J_{ij}'s from the Sidon set $S_{28} = \{\pm 8/28, \pm 13/28, \pm 19/28, \pm 1\}$ [36]. This finite set of rational numbers has an extremely useful property: it creates spin-glasses just as well as selecting real numbers from a 0 mean Gaussian.

5.2.2 A Few Heuristic Decisions

We also decided to run our simulations on complete graphs. A complete graph is simply one where every vertex is connected to every other vertex. For this reason, a complete graph is unambiguously specified by the number of vertices and is typically denoted K_n. For this reason, we refer to these experiments as Sidon Complete SK_n tests. We made this choice for two reasons, (1) complete graphs have the potential to introduce as much frustration as is possible for a fixed number

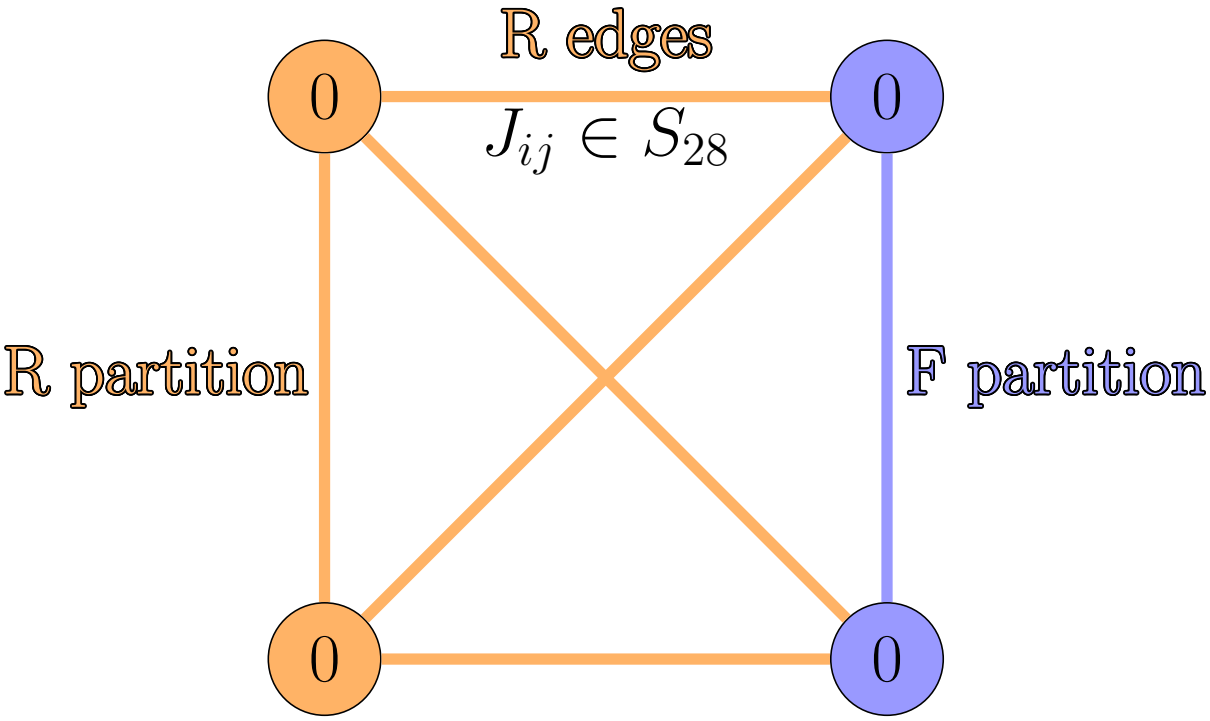

Figure 5.2: An example SK_4 partition.

of qubits and (2) our qubit numbers are limited by the exponential memory cost of adding more, so we wanted to have as "complex" of an Ising Hamiltonian as possible.

Next, our initial tests explicated here are "R biased" in that all mixed edges are assigned to team R. An example for SK_4 can be seen in Fig. 5.2. Our reasoning at this point is purely heuristic: we designed our algorithm so that it could globally explore the energy landscape while being influenced by a good guess on the team R qubits. In this model: we want team R qubits to influence the global search performed by team F. In this sense, it should be team R dictating when the mixed couplings should be turned on and off. In addition, this heuristic decision cuts down on the shear number of possible partitions that should be tried to generate useful statistics.

Even with an R-biased heuristic, the sample space of possible partitions for

an SK_n graph is much too large to try them all. The combinatorics are actually fairly simple for complete graphs (see Appendix B). For these experiments, there are $\sqrt{8^{(n^2-n)}}(2^n - 2)$ unique partitions. Trying every combination for a mere 10 qubits would mean running $\approx 4.45 \times 10^{43}$ FREM anneals... not exactly reasonable. Instead, we adopt the following heuristic: for a given set of couplings, try every partition. For example, an exhaustive test of SK_{10} with fixed random couplings now only takes $2^{10} - 2 = 1022$ FREM anneals.

5.2.3 Some Promising Results

Our eventual goal with FREM is to prove that it can improve ground-state sampling as compared to both forward annealing and reverse annealing alone. Ideally, we will also find a way to do so without having to explore an extremely large hyper-parameter space that nullifies the point of a quantum advantage (insofar as time complexity is concerned). We've yet to make significant progress on the latter goal, but the former goal has been realized by a suitable choice of parameters.

In particular, using $T_f = 1$, $T_r = 1$; $s' = 0.27$ (see D-Wave White Paper [33] for choice of s'), we find that FREM is capable of outperforming both FA and RA. These results are summarized in Fig. 5.3 which show the probability of measuring the ground-state after a forward anneal, a reverse anneal, and the best FREM anneal. By best, we mean the best ground-state probability attainable of those partitions tried–which as explained is every possible partition for fixed randomly chosen S_{28} couplings.

As discussed in Sec. 5.1.2, the FREM algorithm also outputs information about how likely a partition is to outperform FA/RA. This data is summarized in Fig. 5.4. In this particular example, FREM is more likely to outperform FA with a random partition for every graph size tried except for $n = 6$. The success rate compared to RA is not as good, but this is to be expected since RA is also intended to improve

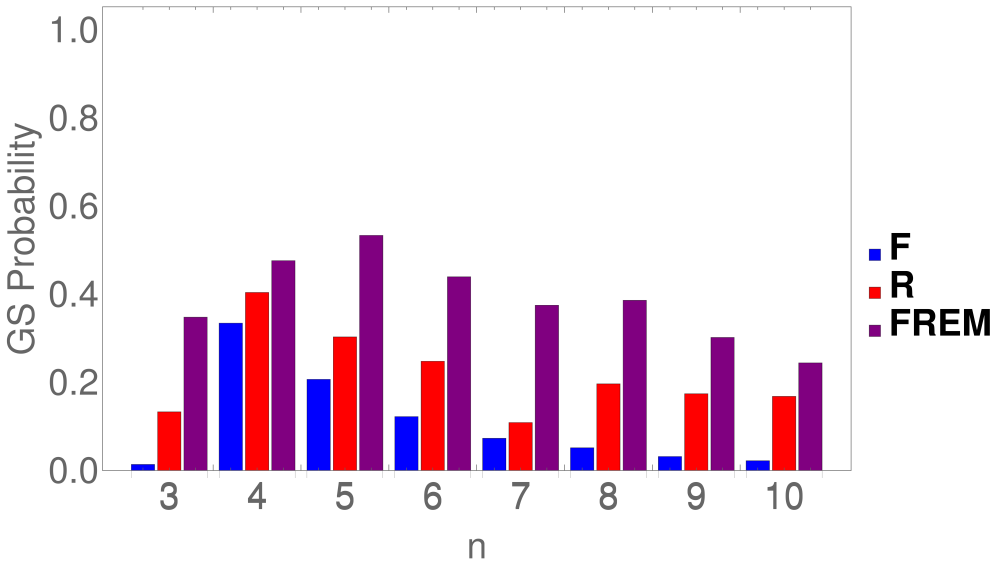

Figure 5.3: This graph summarizes the results for SK_n FREM tests for $T_f = T_r = 1$ and $s' = 0.27$. The x-axis shows the number of qubits used while the y-axis shows the probability of measuring the ground-state for forward annealing, reverse annealing, and (the best) FREM anneal, respectively.

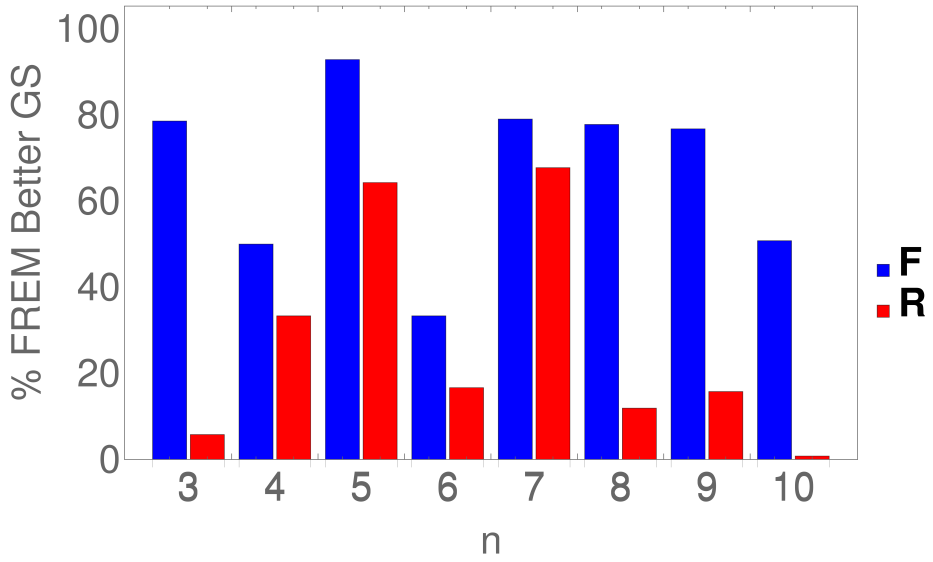

Figure 5.4: This graph summarizes the results for SK_n FREM tests for $T_f = T_r = 1$ and $s' = 0.27$. The x-axis shows the number of qubits used while the y-axis shows the percentage of partitions tried for which FREM annealing ground-state probability outperformed forward annealing and reverse annealing, respectively.

ground-state sampling.

5.3 Critical Qubit (CQ_n) Tests

This is still very much an experimental section of work, but we thought it was interesting enough to include here. This line of inquiry pursues the aforementioned "graph introspection." Is it possible that FREM can be used to identify "critical" regions of a graph? Our intuition of critical meaning "somehow more important than any other region in helping determine the eventual end-state that an anneal ends in." Of course, this is a somewhat vague and purely intuitive notion. Our eventual hope is that this line of inquiry will also shed light on a good definition of critical itself–though we suspect the notion of critical also depends on how one explores the energy landscape. Because of this, it is probably more accurate to call this "graph introspection in the context of an annealing protocol," but this is so wordy as to be cumbersome.

5.3.1 The CQ_n Hamiltonian

Succinctly, a CQ_n graph is a complete graph of n qubits for which $J_{ij} = 1\ \forall i, j$, $h_0 = -1$, and $h_i = 1\ \forall i \neq 0$. An example CQ_4 graph is shown in Fig. 5.5. Now for a little context...

The developed of the CQ_n graph began with our desire to create a highly degenerate graph given a complete topology of n qubits. Of course, this is very easy: set $J_{ij} = h_i = 1\ \forall i, j$. This creates the ultimate frustration: every qubit wants to point down due to its h_i bias. But due to the anti-ferromagnetic J_{ij} coupling, they also all want to point in opposite direction–which, of course, isn't possible. By mistake, I happen to set $h_0 = -1$ instead with all other couplings as described. This made me begin to think about the notion of a "critical qubit." Is

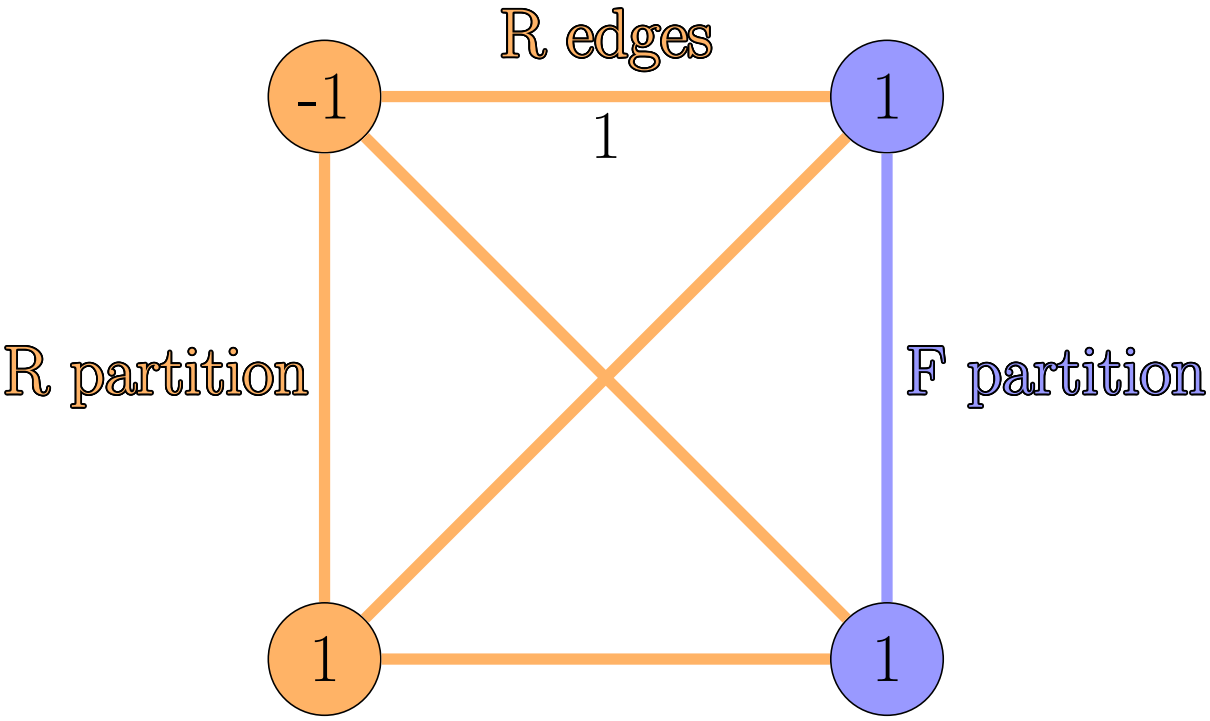

Figure 5.5: An example CQ_4 partition.

q_0 somehow special since it is the asymmetric qubit?

5.3.2 Some Promising Results

In regards to the procedure, we used the exact same methods as were used in the SK_n tests–except our couplings were actually fixed and we kept track of whether the R partition contained the critical qubit q_0 or not. Once again, we used $T_f = 1$, $T_r = 1$; $s' = 0.27$ as our annealing parameters, and we kept track of the same metrics as we did for SK_n. That is, Fig. 5.6 shows the ground-state sampling probability of forward, reverse, and (the best partition) FREM annealing while Fig. 5.7 shows how many partitions tried were successfully better than FA/RA. Once again, FREM shows promise in being an error-mitigation routine.

This time around, however, we have an addition graph of interest, Fig. 5.8. This

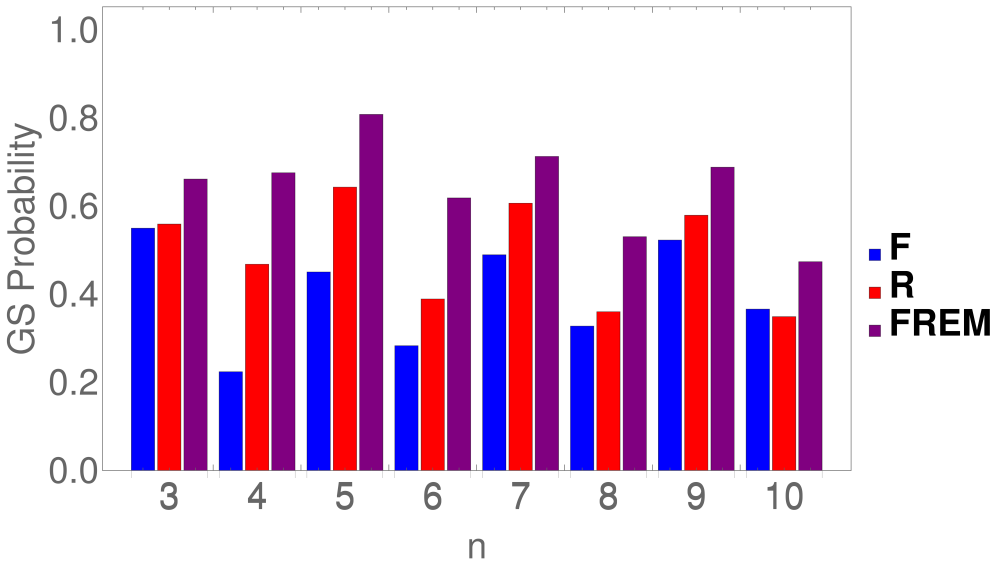

Figure 5.6: This graph summarizes the results for CQ_n FREM tests for $T_f = T_r = 1$ and $s' = 0.27$. The x-axis shows the number of qubits used while the y-axis shows the percentage of partitions tried for which FREM annealing ground-state probability outperformed forward annealing and reverse annealing, respectively.

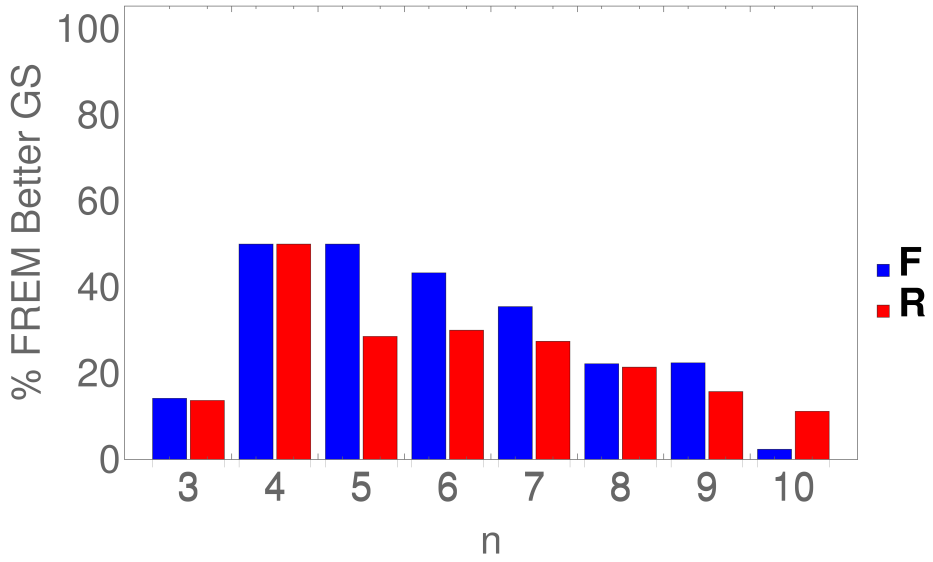

Figure 5.7: This graph summarizes the results for CQ_n FREM tests for $T_f = T_r = 1$ and $s' = 0.27$. The x-axis shows the number of qubits used while the y-axis shows the percentage of partitions tried for which FREM annealing ground-state probability outperformed forward annealing and reverse annealing, respectively.

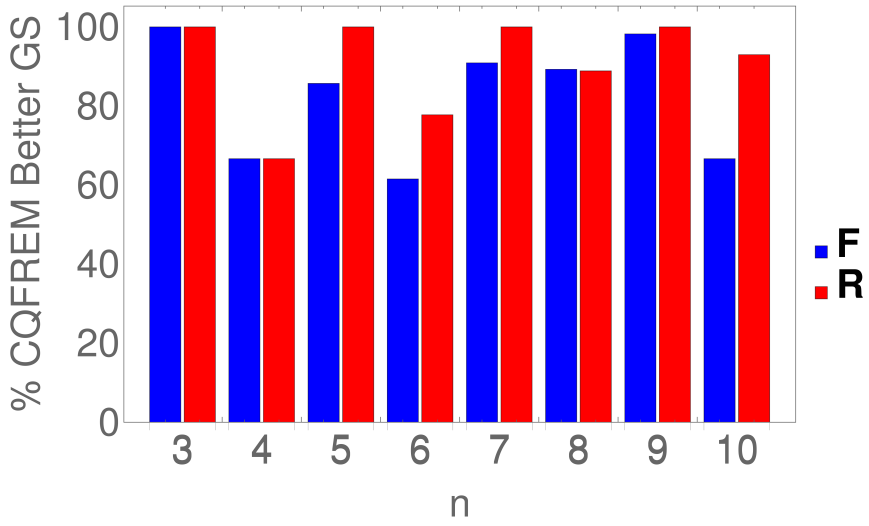

Figure 5.8: This graph summarizes the results for CQ_n FREM tests for $T_f = T_r = 1$ and $s' = 0.27$. The x-axis shows the number of qubits used while the y-axis shows the percentage of partitions where FREM outperformed FA/RA that included the critical qubit, q_0.

shows, given those FREM anneals that outperformed FA/RA, what percentage of these trials contained the critical qubit, q_0. In each case, the percentage is over 50, and is is typically in the 70-90 range. While this does not in any way conclusively show that FREM is capable of so-called "graph introspection," it is an interesting step in that direction, nevertheless.

5.4 Conclusions

No strong conclusions should be made at this point about the efficacy of FREM. In truth, this exploration has been but a mere proof-of-concept of our algorithm and only the beginning of collecting numerical evidence that it is effective. There are many avenues to explore. For one, we need to explore more of our enormous hyper-parameter space–ideally running some computations on super-computers where we can play with more qubits. Next, we should probably conduct our explorations

more realistically by intentionally giving FREM a *bad* guess for the ground-state of the R partition. Does it improve the answer then? If so, can it continually do so, iteratively?

Nevertheless, the little we have shown here has already sparked a number of interesting questions. For example, while Fig. 5.6 shows that FREM has the potential to correct errors in the same way that Fig. 5.3 did, there is an interesting difference between Fig. 5.7 and Fig. 5.4. In particular, a random partition performs worse, on average, for the CQ_n tests. Is this because CQ_n is simpler? More degenerate?

Finally, and most interestingly, our algorithm begins to probe at a very interesting line of thought regarding graph introspection. Is FREM capable of sussing out critical portions of a graph? If so, what does critical even mean? We hope that we can find answers soon as we explore this topic further in the near future!

5.5 Acknowledgements

Based on work supported by the Air Force Research Laboratory (AFRL) under agreement number FA8750-18-1-0096.

The views and conclusions herein are those of the author, and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL or the US Government.

Chapter 6

Conclusions

In summary, we have developed a package, dwaveutils, that allows users to study quantum annealing on the D-Wave as quickly and easily as possible. First, it makes pre- and post-processing, especially as it concerns hyper-parameter sweeping, extremely easy. Second, it makes checking one's answer numerically by direct diagonalization and numeric quantum annealing a breeze. We've illustrated the use of the package by carrying out two investigations we were already interested in with it: simulations of transverse-field Ising Hamiltonians and forward-reverse error-mitigation annealing.

In the first study, we found that, at least for our 3 qubit problem, the quench dynamic is not necessarily sufficient to approximate a sudden transition. Our leading hypothesis is that it is simply too slow for the problem tested, but we've yet to ascertain that this is the culprit. Whatever the case, it's certainly not possible to sample with a transverse field present for all problems.

In the second study, we say that FREM has the potential to be a viable error reduction protocol that does not require ancilla qubits. In particular, we mean FREM could increase the probability that D-Wave devices correctly find the solution to an optimization problem. Further, FREM may allow for information about the internals of a graph to be explored through critical qubit analysis.

Appendix A

The Postulates of Quantum Mechanics

To encapsulate the essence of quantum mechanics quickly, we will explicate the essential postulates as stated in Nielsen and Chuang's excellent textbook *Quantum Computation and Quantum Information*[7].

Postulate 1. Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the *state space* of the system. The system is completely described by its *state vector*, which is a unit vector in the system's state space.

Postulate 2. The evolution of a *closed* quantum system is described by a *unitary transformation*. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $|\psi'\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2,

$$|\psi'\rangle = U\,|\psi\rangle.$$

Postulate 3. Quantum measurements are described by a collection of $\{M_m\}$ of *measurement operators*. These are operators acing on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement, then the probability that result m occurs is given by

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle,$$

and the state of the system after the measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}.$$

The measurement operators satisfy the *completeness equation*,

$$\sum_m M_m^\dagger M_m = I.$$

The completeness relation equation expresses the fact that probabilities sum to one:

$$1 = \sum_m p(m) = \sum_m \frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}.$$

Postulate 4. The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$.

Appendix B

Some Complete Graph Combinatorics

First of all, every vertex of a complete graph is connected to every other vertex. Hence, the total number of edges for a complete graph with n vertices is

$$EC(n) = (n-1) + (n-2) + \ldots + 0 = \frac{n^2 - n}{2}. \tag{B.1}$$

Next, the total number of "R-biased" partitions of a graph is easy: we need only look at the unique combinations of qubits that could belong to team R. That's just the powerset throw away the entire graph and the null set, i.e.,

$$|2^{\{0,1,\ldots,n\}}| - 2 = 2^n - 2, \tag{B.2}$$

where $|\cdot|$ stands for the cardinality of the powerset.

Then, if each vertex is allowed to take on a random value from a set of 8 choices (the number of distinct choices in S_{28}), as in SK_n graphs, then the total number of unique partitions is simply

$$8^{EC(n)}(2^n - 2) = \sqrt{8^{n^2-n}}(2^n - 2). \tag{B.3}$$

References

[1] C. S. Calude, E. Calude, and M. J. Dinneen, "Guest column: Adiabatic quantum computing challenges," *SIGACT News*, vol. 46, pp. 40–61, Mar. 2015. vi, 22

[2] S. Wiesner, "Conjugate coding," *SIGACT News*, vol. 15, pp. 78–88, Jan. 1983. 2

[3] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, pp. 467–488, Jun 1982. 2

[4] D. Finke, "Quantum computing report: Where qubits entangle with commerce," 2018. 2, 40

[5] E. Tang, "A quantum-inspired classical algorithm for recommendation systems," 2018. 3

[6] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," 1995. 3

[7] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. New York, NY, USA: Cambridge University Press, 10th ed., 2011. 4, 5, 65

[8] J. Preskill, "Quantum computing in the nisq era and beyond," 2018. 4

[9] D. McIntyre, C. Manogue, J. Tate, and O. S. University, *Quantum Mechanics: A Paradigms Approach*. Pearson, 2012. 5

[10] W. Gerlach and O. Stern, "Der experimentelle nachweis der richtungsquantelung im magnetfeld," *Zeitschrift für Physik*, vol. 9, pp. 349–352, Dec 1922. 5

[11] Hitoshi, "University of California: Berkeley 221A Lecture notes in Time-Dependent Perturbation Theory," September 2006. 7, 8

[12] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Rev. Mod. Phys.*, vol. 90, p. 015002, Jan 2018. 9, 16, 17

[13] "Non-adiabatic crossing of energy levels," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 137, no. 833, pp. 696–702, 1932. 9

[14] B. A. Cipra, "An introduction to the ising model," *The American Mathematical Monthly*, vol. 94, no. 10, pp. 937–959, 1987. 10

[15] E. W. Weisstein, "Pauli matrices. From MathWorld—A Wolfram Web Resource." Last visited on 03/25/2019. 12

[16] M. Vojta, "Quantum phase transitions," *Reports on Progress in Physics*, vol. 66, no. 12, p. 2069, 2003. 13

[17] H. Rieger and A. P. Young, "Quantum spin glasses," 1996. 14

[18] A. Lucas, "Ising formulations of many NP problems," 2013. 15, 16

[19] A. Das and B. K. Chakrabarti, "Quantum annealing and analog quantum computation," 2008. 16

[20] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "a quantum adiabatic evolution algorithm applied to random instances of an np-complete problem," 16

[21] D-Wave: The Quantum Computing Company, "D-wave user manual 09-1109 a-j: Technical description of the d-wave quantum processing unit," March 2018. 17, 18

[22] V. Choi, "Minor-embedding in adiabatic quantum computation: I. The parameter setting problem," *Quantum Information Processing*, vol. 7, pp. 193–209, Oct 2008. 21

[23] V. Choi, "Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design," *Quantum Information Processing*, vol. 10, pp. 343–353, Jun 2011. 21

[24] W. Vinci, T. Albash, G. Paz-Silva, I. Hen, and D. A. Lidar, "Quantum annealing correction with minor embedding," *Phys. Rev. A*, vol. 92, p. 042310, Oct 2015. 21, 24

[25] J. Cai, W. G. Macready, and A. Roy, "A practical heuristic for finding graph minors," 2014. 21, 24

[26] D-Wave Systems Inc., "Ocean." https://github.com/dwavesystems/dwave-ocean-sdk, 2018. 21, 25

[27] N. Ezzell, "dwaveutils." 25

[28] J. Johansson, P. Nation, and F. Nori, "Qutip: An open-source python framework for the dynamics of open quantum systems," *Computer Physics Communications*, vol. 183, no. 8, pp. 1760 – 1772, 2012. 34

[29] J. Johansson, P. Nation, and F. Nori, "Qutip 2: A python framework for the dynamics of open quantum systems," *Computer Physics Communications*, vol. 184, no. 4, pp. 1234 – 1240, 2013. 34

[30] R. Xia, T. Bian, and S. Kais, "Electronic Structure Calculations and the Ising Hamiltonian," *The Journal of Physical Chemistry B*, vol. 122, no. 13, pp. 3384–3395, 2018. PMID: 29099600. 40

[31] R. Harris, Y. Sato, A. J. Berkley, M. Reis, F. Altomare, M. H. Amin, K. Boothby, P. Bunyk, C. Deng, C. Enderud, S. Huang, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, T. Lanting, R. Li, T. Medina, R. Molavi, R. Neufeld, T. Oh, I. Pavlov, I. Perminov, G. Poulin-Lamarre, C. Rich, A. Smirnov, L. Swenson, N. Tsai, M. Volkmann, J. Whittaker, and J. Yao, "Phase transitions in a programmable quantum spin glass simulator," *Science*, vol. 361, no. 6398, pp. 162–165, 2018. 43

[32] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a quantum processor," 2013. 43

[33] D.-W. T. Q. C. Company, "Reverse quantum annealing for local refinement of solutions." https://www.dwavesys.com/sites/default/files/14-1018A-A_Reverse_Quantum_Annealing_for_Local_Refinement_of_Solutions.pdf, 2017. 49, 51, 56

[34] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," , vol. 299, p. 802, Oct. 1982. 49

[35] N. Chancellor, "Modernizing quantum annealing using local searches," 2016. 49

[36] H. G. Katzgraber, F. Hamze, Z. Zhu, A. J. Ochoa, and H. Munoz-Bauza, "Seeking quantum speedup through spin glasses: The good, the bad, and the ugly," *Phys. Rev. X*, vol. 5, p. 031026, Sep 2015. 54

[37] E. E. Edwards, S. Korenblit, K. Kim, R. Islam, M. S. Chang, J. K. Freericks, G. D. Lin, L. M. Duan, and C. Monroe, "Quantum simulation and phase diagram of the transverse field ising model with three atomic spins," 2010.