D-Wave
The Quantum Computing Company™

Quantum Annealing amid Local Ruggedness and Global Frustration

TECHNICAL REPORT

J. King, S. Yarkoni, J. Raymond, I. Ozfidan, A. D. King, M. Mohammadi Nevisi, J. P. Hilton, and C. C. McGeoch

2017-01-16

Overview

We introduce a problem class with two attributes crucial to the evaluation of quantum annealing processors: local ruggedness (i.e., tall, thin energy barriers in the energy landscape) so that quantum tunneling can be harnessed as a useful resource, and global frustration so that the problems are combinatorially challenging and representative of real-world inputs. We evaluate the new 2000-qubit D-Wave quantum processing unit (QPU) on these inputs, comparing it to software solvers that include both GPU-based solvers and a CPU-based solver which is highly tailored to the D-Wave topology. The D-Wave QPU solidly outperforms the software solvers: when we consider pure annealing time, the D-Wave QPU is three to four orders of magnitude faster than software solvers in both optimization and sampling evaluations.

14-1003A-C
D-Wave Technical Report Series

CONTACT

Corporate Headquarters
3033 Beta Ave
Burnaby, BC V5G 4M9
Canada
Tel. 604-630-1428

US Office
2650 E Bayshore Rd
Palo Alto, CA 94303

Email: info@dwavesys.com

www.dwavesys.com

Notice and Disclaimer

Summary

Context

A recent Google study [1] compared a D-Wave 2X quantum processing unit (QPU) to two classical Monte Carlo algorithms: simulated annealing (SA) and quantum Monte Carlo (QMC). The study showed the D-Wave 2X to be up to 100 million times faster than the classical algorithms.

The Google inputs are designed to demonstrate the value of collective multiqubit tunneling, a resource that is available to D-Wave QPUs but not to simulated annealing. But the computational hardness in these inputs is highly localized in gadgets, with only a small amount of complexity coming from global interactions, meaning that the relevance to real-world problems is limited. Later work [2] compared D-Wave 2X performance on these instances to a wider selection of algorithms. HFS, a specialized combinatorial algorithm, handles the gadgets of the Google problems using localized brute force. Because there is only a small amount of computational hardness from the global interactions, HFS solves the Google problems with relative ease.

Contributions

In this study we provide a new synthetic problem class that addresses the limitations of the Google inputs while retaining their strengths. We use simple clusters instead of more complex gadgets and more emphasis is placed on creating computational hardness through global interactions like those seen in interesting real-world inputs.

We use these inputs to evaluate the new 2000-qubit D-Wave QPU. We include the HFS algorithm—the best performer in a broader analysis of Google inputs [2]—and we include state of the art GPU implementations of SA and QMC. The D-Wave QPU solidly outperforms the software solvers; when we consider pure annealing time (computation time), the D-Wave QPU reaches ground states up to 2600 times faster than the competition (see Figure 3). In the task of zero-temperature Boltzmann sampling from challenging multimodal inputs, the D-Wave QPU holds a similar advantage and does not see significant performance degradation due to quantum sampling bias.

Our input class has the additional benefit of parameter-tunable ruggedness of the associated energy landscapes. Ruggedness correlates with classical hardness, and more rugged inputs can benefit more from quantum tunneling. We show that quantum annealing shows greater resilience to ruggedness than simulated annealing, and the more closely a classical Monte Carlo algorithm approximates quantum annealing, the better it handles increasing ruggedness (see Figure 4).

Contents

1 Introduction

Quantum annealers are designed to take advantage of quantum tunneling to find good solutions to hard optimization problems. When constructing a family of synthetic inputs to test the potential of a quantum annealing platform, one should therefore ensure that the inputs a) are such that solvers can benefit from quantum tunneling, and b) are hard optimization problems with global frustration.

For a solver to benefit from quantum tunneling, the energy landscape associated with the input must have tall, thin energy barriers. For an input to be computationally hard, the input must have constraints that interact with each other in nontrivial ways.

Quantum processing units (QPUs) developed by D-Wave Systems that use the quantum annealing algorithm have been commercially available since 2011. These QPUs solve Ising model inputs defined on the underlying working graph of the chip. There have been various efforts to evaluate the performance of the D-Wave systems using synthetic inputs generated randomly from different distributions, or *input classes*.

This study has two main contributions: to propose a new problem class ideal for evaluating D-Wave QPUs, and to use this problem class to evaluate the 2000-qubit D-Wave QPU.

1.1 Proposing a new problem class

Previous evaluations of D-Wave QPUs have used problem classes that benefit either too little or too much from quantum tunneling to be ideal for evaluating quantum annealers.

On one side of this spectrum we have problems such as random unstructured ± 1 problems on the Chimera topology native to D-Wave QPUs. These were used by Rønnow et al. [3] in their evaluation of the D-Wave Two QPU in 2014, but they are now known [4] to lack a finite-temperature phase transition, meaning that quantum tunneling is unlikely to play a significant role when solving them.

On the other side of the spectrum, Denchev et al. [1] recently introduced an input class designed to benefit immensely from quantum tunneling. We refer to these inputs as Google problems. Their study showed a massive speed increase (up to 100 million times faster) of a D-Wave 2X system over simulated annealing (SA) and quantum Monte Carlo (QMC), also known as *simulated quantum annealing*. This provided strong evidence for the ability of quantum annealing to leverage quantum tunneling in a computationally relevant way. However, the spin-glass backbones of the Google problems are easy to solve, meaning that a) the problems have limited relevance to real-world problems, and b) certain cluster-detecting algorithms can solve them with relative ease [2].

In this study we provide a problem class that aims to retain the advantages of Google problems while being more reflective of real-world problems. They are more reflective of real-world problems because, rather than relying too heavily on finely-tuned gadgets, they derive much of their computational hardness from larger spin-glass backbones with planted frustration.

Our problems are synthetic and are easy to solve using knowledge of the problem class.[1]

[1] For example, the super-spin heuristic [2] that relies on hard-coded knowledge of clusters would be far faster

However, they have properties such as tunable ruggedness that make them useful for the evaluation of quantum annealing and classical approximations thereof. In this way, they are similar to Kauffman's NK model that has proved very useful in the analysis of evolutionary algorithms [5–7].

1.2 Evaluation of the 2000-qubit D-Wave QPU

We use this new problem class to evaluate the latest-generation D-Wave QPU. We measure its performance in absolute terms and we analyze its response to the ruggedness parameters of the problem class.

The software competition we consider is much stronger than that considered by Denchev et al. [1], and includes GPU implementations of SA, QMC, and SVMC, and also includes Selby's implementation [8, 9] of the Hamze-de Freitas-Selby (HFS) algorithm [8, 10]. In the study of Mandrà et al. [2] that used a wide array of algorithms to solve Google problems, Selby's implementation of HFS was the fastest software solver in terms of both scaling and absolute speed.

We find that the D-Wave QPU is able to find ground states up to 2600 times faster than the software competition. We also consider the problem of sampling from ground states and find that the D-Wave QPU maintains a similar advantage and does not struggle to find a diverse set of optimal solutions.

The remainder of the paper is organized as follows. In Section 2 we provide a description of the 2000-qubit D-Wave system and a history of D-Wave QPUs. In Section 3 we present the problem class analyzed in this paper and discuss the concept of ruggedness and its relevance to optimization problems. In Section 4 we discuss the software solvers used in our evaluations, as well as notable solvers that were not suitable. In Section 5 we present our experimental results on optimization. In Section 6 we present our experimental results on sampling from ground states. In Section 7 we argue that constant pre-factors are important and that scaling is not the only thing we should be interested in; this argument is based on power consumption of classical algorithms. In Section 8 we provide further discussion and conclude the paper.

2 D-Wave quantum processing units

We start with an overview of D-Wave design features and introduce notation that will be used throughout. For details about underlying technologies see Bunyk et al. [11], Dickson et al. [12], Harris et al. [13], Johnson et al. [14] or Lanting et al. [15].

than the software solvers we consider. However, such heuristics do not generalize to other problem classes and it would not make sense to include them as competition solvers.

2.1 Ising minimization

D-Wave annealing-based QPUs are designed to find minimum-cost solutions to the Ising minimization (IM) problem, defined on a graph $G = (V, E)$ as follows. Given a collection of fields $h = \{h_i : i \in V\}$ and couplings $J = \{J_{ij} : (i, j) \in E\}$, assign values from $\{-1, +1\}$ to *n spin variables* $s = \{s_i\}$ so as to minimize the *energy function*

$$E(s) \quad = \quad \sum_{i \in V} h_i s_i + \sum_{(i,j) \in E} J_{ij} s_i s_j. \tag{1}$$

The spin variables s can be interpreted as magnetic poles in a physical particle system; in this context, negative J_{ij} is *ferromagnetic* and positive J_{ij} is *antiferromagnetic*, the optimal solution is called a *ground state*, and nonoptimal solutions are *excited states*. IM instances can be trivially transformed to Quadratic Unconstrained Boolean Optimization (QUBO) instances defined on integers $s = \{0, 1\}$, or to Maximum Weighted 2-Satisfiability (MAX W2SAT) instances defined on Booleans $s = \{\text{true}, \text{false}\}$, all of which are NP-hard.

2.2 Chimera topology

The native connectivity topology for the D-Wave QPU is based on a C_{16} *Chimera graph* containing 2048 vertices (qubits) and 6016 edges (couplers).

A Chimera graph of size C_s is an $s \times s$ grid of Chimera cells (also called unit tiles or unit cells), each containing a complete bipartite graph on 8 vertices (a $K_{4,4}$). Each vertex is connected to its four neighbors inside the cell as well as two neighbors (north/south or east/west) outside the cell: therefore every vertex has degree 6 excluding boundary vertices.

In this study, as in others, we vary the problem size using square subgraphs of the full graph, from size C_4 (128 vertices) up to C_{16} (2048 vertices). Note that the number of problem variables $n = 8s^2$ grows quadratically with Chimera size. The reason we measure algorithm performance as a function of the Chimera size and not the number of qubits is that problem difficulty tends to scale exponentially with the Chimera size, i.e., with the square root of the number of qubits, since the treewidth of a Chimera graph C_s is linear in s [16, 17].

Because the chip fabrication and trapped magnetic flux leave some small number of qubits unusable, each QPU has a specific *hardware working graph* $H \subset C_{16}$. The qubit yield—the fraction of qubits that are operational—is typically around 98% for the 2000-qubit D-Wave system whereas 95% was typical for the D-Wave 2X. The working graph used in this study has 2035 working qubits out of 2048.

2.3 Quantum annealing

D-Wave QPUs solve Ising problems by *quantum annealing* (QA) in the form proposed by Kadowaki and Nishimori [18]. The QA algorithm is implemented in hardware using a framework of analog control devices to manipulate a collection of qubit states according to

a time-dependent Hamiltonian shown below.

$$\mathcal{H}(t) \quad = \quad A(t) \cdot \mathcal{H}_{init} + B(t) \cdot \mathcal{H}_{prob}. \tag{2}$$

QA carries out a gradual transition in time $t : 0 \rightarrow t_a$, from an initial ground state in $\mathcal{H}_{init}$, to a state described by the *problem Hamiltonian* $\mathcal{H}_{prob} = \sum_i h_i \sigma_i^z + \sum_{ij} J_{ij} \sigma_i^z \sigma_j^z$. The problem Hamiltonian matches the energy function (1), so that a ground state for $\mathcal{H}_{prob}$ is a minimum-cost solution to $E(s)$.

QA is closely related to *adiabatic quantum computing* (AQC). The AQC model of computation was proposed by Farhi et al. [19] who showed that if the transition is carried out slowly enough the algorithm will find a ground state (i.e., an optimal solution) with high probability.

Theoretical guarantees about solution times for quantum algorithms (found in [19]) assume that the computation takes place in an ideal closed system, perfectly isolated from energy interference from ambient surroundings. The 2000-qubit D-Wave chip is housed in a highly shielded chamber and cooled to near absolute zero; nevertheless, as is the case with any real-world quantum device, it must suffer some amount of interference, which has the general effect of reducing the probability of landing in a ground state. Thus, theoretical guarantees on performance may not apply to these systems. We consider any D-Wave QPU to be a *heuristic* solver, which requires empirical approaches to performance analysis.

The D-Wave QPU studied here contains 2035 active qubits (quantum bits) and 5912 active couplers made of microscopic loops of niobium connected to a large and complex analog control system via an arrangement of Josephson Junctions. Thermometry on the refrigerator of the D-Wave QPU and fits of single qubit measurements to a thermodynamic model indicate that $T \lesssim 15\,\text{mK}$. When cooled to temperatures below 9.3 K, niobium becomes a superconductor and is capable of displaying quantum properties including superposition, entanglement, and quantum tunneling. Because of these properties, the qubits on the chip behave as a quantum mechanical particle process that carries out a transition from initial state described by $\mathcal{H}_{init}$ to a problem state described by $\mathcal{H}_{prob}$ [12, 15, 20].

2.4 Modeling performance

Given input instance (h, J), a D-Wave computation involves the following steps.

a. **Program.** Load (h, J) onto the chip; denote the elapsed programming/initialization time t_i.

b. **Anneal.** Carry out the QA algorithm. Anneal time t_a can be set by user to some value $5\,\mu\text{s} \leq t_a \leq 1000\,\mu\text{s}$.

c. **Read.** Record qubit states to obtain a solution; denote the elapsed readout time t_r.

d. **Repeat.** Repeat steps b and c k times to obtain a sample of k solutions.

We define *sample time* t_s and *total time* T as follows:

$$\begin{aligned} t_s &= (t_a + t_r) \\ T &= t_i + k\, t_s. \end{aligned} \tag{3}$$

For the D-Wave system studied in this paper, the median programming time t_i is 9.5 ms and the median readout time t_r is 123 µs.

In this study, both for software solvers and for the D-Wave QPU, we typically report annealing time rather than total time. Annealing time is the measure of the algorithm proper, and measuring total time often obscures trends in data. Scaling plots are particularly susceptible to this because the overhead of programming time makes scaling—typically presented on a semilog plot—look totally flat except for an uptick at the very largest problem sizes. Further, we are most interested in the future potential of D-Wave QPUs, and we expect that programming time and readout time will be reduced to small fractions of their current values; minimum annealing times will similarly be reduced, allowing us better control over the algorithm parameters. For reference, since many people will be interested in total wall clock time, rather than annealing time, a 1000 times speedup over software solvers in annealing time, typical for the D-Wave QPU in this study, translates roughly to a 30 times speedup in total wall clock time including programming and readout.

System characteristics of D-Wave QPUs such as yield can vary within a generation. If we compare this specific 2000-qubit D-Wave system to the specific D-Wave 2X QPU studied in 2015 [21], programming time has decreased by 20%, readout is three times faster, and yield has improved from 95% to 99%.

3 Frustrated Cluster Loop problems

3.1 Ruggedness and clusters

Ruggedness is a feature of certain optimization problems—more specifically their energy landscapes—characterized by tall energy barriers and many local optima [22, 23]. Typically, rugged problems are harder to solve, particularly with Markov chain Monte Carlo (MCMC) methods [24, 25]. In the late 1980s, when ruggedness was first being explored in the context of evolutionary biology and bio-inspired computing, Kauffman's NK model was put forward as a model with tunable ruggedness inspired by genetic fitness functions under varying degrees of *epistasis*, or how many other genetic loci affect the fitness contribution of a given locus [5–7]. The tunable ruggedness of the NK model has proved very valuable in the study of optimization heuristics, particularly evolutionary algorithms [26].

Closely related to ruggedness is the analysis of spin overlap, in which landscape features are inferred from the distribution of overlap of two random states sampled from the Boltzmann distribution [27, 28]. Tall, thin peaks in the spin overlap distribution tend to correspond to tall, thin energy barriers; the presence of these features correlates not only with ruggedness and classical hardness, but also with applicability of quantum annealing, since quantum tunneling is likely to be a useful computational resource in the presence of these tall, thin barriers. Zhu et al. [29] have used spin overlap features to predict whether a problem can be solved by QA more efficiently than by SA, showing promising preliminary results for optimization problems such as weighted partial MAX-2SAT, minimum vertex cover, satisfiability, graph partitioning, circuit fault diagnosis, and certain spin-glass instances [29, 30]. This work points to the potential of quantum annealers to have a place in portfolio solvers [31] and hybrid algorithms running on heterogeneous computing systems

alongside CPUs, GPUs, and other coprocessors [32, 33].

To induce ruggedness using tall, thin energy barriers, Denchev et al. [1] used ferromagnetically coupled unit tiles as clusters. Flipping such a cluster in the absence of fields or external couplings involves jumping over or tunneling through an energy barrier that is 16 Ising units high and has a width of 8 in Hamming space. Denchev et al. [1] actually go beyond using single-tile clusters and use two-tile gadgets studied previously by Boixo et al. [20]. This gadget is made up of two clusters that form a deceptive trap to draw annealers into a local minimum using local fields; annealers must then go over or through an energy barrier to reach the gadget's ground state.

Instead of using two-cluster gadgets, we simply use single-cell ferromagnetic clusters to induce ruggedness, leaving us with a simpler problem class.

3.2 FCL problem generation

We create local ruggedness by treating unit cells of the Chimera graph as ferromagnetically-coupled clusters. We create global frustration by joining these clusters together using a problem generated on the logical graph of clusters. This creates an energy landscape that is macroscopically interesting and in which the clusters induce wells separated by tall, thin energy barriers.

The logical graph of clusters is a square lattice, with a logical 16×16 lattice of clusters spanning the working graph of a 2000-qubit D-Wave QPU.[2] The problems we generate on the logical graph are *frustrated loop problems*, constraint satisfaction problems first used in the evaluation of D-Wave QPUs by Hen et al. [34] and modified to allow precision limits by King et al. [35].

We refer to the final inputs as *frustrated cluster loop* (FCL) problems. For a given Chimera graph G_C that may or may not have missing qubits or couplers, an FCL problem is generated from three parameters, α (the clauses-to-variables ratio), ρ (the range, or precision), and $R \geq \rho$ (the ruggedness) as follows:

a. Define each unit cell as a *logical spin* if it has no missing qubits or couplers. Use $c(v)$ to denote the logical spin index corresponding to qubit v.

b. Wherever all four couplers connecting two logical spins are present, define these couplers as a *logical coupler*.

c. Define the logical graph G_L as the graph comprising the logical spins and logical couplers.

d. Generate a range-bounded frustrated loop problem Hamiltonian (h_L, J_L) on G_L using parameters α and ρ as per King et al. [35] (note that h_L is the zero vector).

[2] Note that a 16×16 logical lattice is significantly larger than the largest logical lattice, 4×4, of the Google problems considered by Denchev et al. [1]—their two-tile gadgets take up more space than our one-tile clusters and the D-Wave 2X has a smaller working graph than the 2000-qubit D-Wave system. These larger logical graphs in the problems we consider mean that the spin-glass backbones of these problems are significantly more computationally challenging.

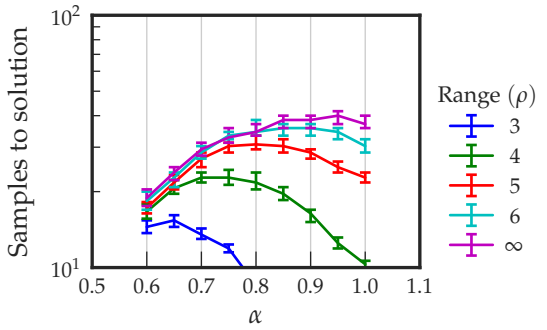

Figure 1: Logical problem difficulty as measured by expected samples to solution for simulated annealing. Error bars show the 95% confidence intervals for the medians, grouped over α and ρ. Difficulty is maximized at $\alpha = 0.65$ for precision 3, $\alpha = 0.75$ for precision 4, $\alpha = 0.8$ for precision 5, and $\alpha = 0.85$ for precision 6.

e. Define the native Chimera Hamiltonian (h_C, J_C) with h_C as the zero vector and J_C as:

$$J_C(u,v) = \begin{cases} -1, & \text{if } c(u) = c(v) \\ \frac{1}{R} \cdot J_L(c(u), c(v)), & \text{otherwise.} \end{cases}$$

It is worth repeating that these Hamiltonians have no fields (i.e., h_L and h_C are both zero vectors). Note also that in-tile couplings in J_C are all -1 and inter-tile couplings take values in

$$\{j/R \mid j \in \{-R, -R+1, \ldots, R\}\}.$$

Since we ensure that $\rho \leq R$, and $\rho \geq |j|$ for any logical coupling j, all couplings in J_C are in the range $[-1, 1]$

The logical frustrated loop problems may be disconnected and have multiple components; we reject such disconnected inputs at generation time.

While these problems are large enough to span the entire working graph of the latest D-Wave QPUs, the repetition code inherent in logical couplers and spins makes them relatively robust to analog errors [36].

3.3 Problem class parameters

The FCL problem class has three parameters: the clauses-to-variables ratio α, the range ρ, and the ruggedness R. We would like to restrict our experiments to the most interesting region of the parameter space.

First we aim to determine the value of α that maximizes the difficulty of the logical problem. If α is too low, a problem is underconstrained and is easy to solve. If α is too high, the planted solution is expressed too strongly and the problem's features approach those of a ferromagnet, making it easy to solve. The difficulty of the logical problem depends only on α and ρ, not on R. For various values of ρ, we perform a sweep of α to determine the value that maximizes the hardness of the logical problem (see Figure 1).

The impacts of ρ are more nuanced. First, the value of ρ provides an upper bound on the limit of α because packing in more loops eventually raises the maximum coupler range. Second, for a fixed value of α, problems with a lower ρ value have their loops spread out more evenly over the logical spins. Finally, for the native problem, coupler values are scaled down by a factor of $R \geq \rho$ so that inter-tile couplings are in the range $[-1, 1]$ (in-tile couplings are always -1). Thus higher values of ρ constrain R to be higher, and make problems more locally rugged relative to the global Hamiltonian. In Section 5, we attempt to deconvolve the impacts of ρ and R.

The ability to tune the ruggedness of the inputs by varying R, either by specifying $R = \rho$ and varying ρ, or by varying R independently, gives FCL problems an additional degree of utility, particularly when assessing the value of quantum tunneling and the potential of quantum annealing. Varying ρ and specifying $R = \rho$ makes problem generation simpler by reducing the number of free parameters whereas fixing ρ and varying R allows us to isolate the impact of ruggedness without altering the complexity of the logical problem.

3.4 Confirming correlation between ruggedness and classical hardness

We expect to see a positive correlation between ruggedness and classical hardness. Here we characterize classical hardness using the decorrelation time of an MCMC procedure. To validate this assumption we measure the decorrelation time for a parallel tempering (PT) procedure that uses the Metropolis algorithm in combination with the standard replica exchange rule [37]; we use the autocorrelation of temperature as our measure of decorrelation [38]. For more details of this method, see Appendix A.

Figure 2 illustrates the relationship between ruggedness and classical hardness. Confirming our intuition, FCL problems with greater ruggedness are characterized by greater classical hardness.

4 Software solvers

The four software solvers we consider are GPU implementations of SA, QMC, and SVMC, and Selby's CPU implementation of HFS [8, 9].

Recent studies of D-Wave QPUs have not included GPU-based software solvers despite the fact that SA is very amenable to GPU implementation [39]. The addition of GPU solvers is a significant raising of the bar in terms of software competition, and means that solvers that can be implemented on GPUs have taken a leap forward relative to solvers that cannot. Run on modern hardware, our GPU-based algorithm implementations are roughly 1000 times faster than the corresponding single-core CPU implementations.

Mandrà et al. [2] analyzed the performance of a diverse set of solvers on the inputs of Denchev et al. [1]. However the lack of GPU implementations of these solvers means that most are unlikely to be competitive in an absolute sense. Indeed, of the three classes of solvers that they study, only sequential algorithms, which they find to have the worst performance, have the massive parallelizability and low memory requirements that make

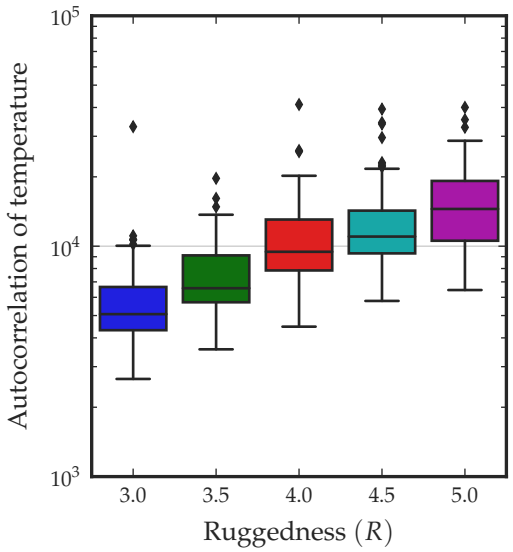

Figure 2: Box plots showing ruggedness versus classical hardness. We hold ρ and α fixed at 3 and 0.65, respectively, and vary the ruggedness R of the native Chimera problem. At each value of R we generated 100 instances; points indicate outliers. Classical hardness is measured using the autocorrelation of temperature. There is a clear positive correlation between ruggedness and classical hardness.

efficient GPU implementations possible. It is also possible to implement SA in a field-programmable gate array (FPGA), but the additional speedup over GPU implementation is limited and generally not worth the increased cost of hardware.

In Appendix B we give further details of the software solvers and parameterizations we used. We also discuss algorithms we omitted because of prohibitive runtimes.

5 Optimization

We measure the expected time to solution (TTS) of different solvers on the inputs, calculated as

$$\text{TTS} = \frac{\text{time per anneal}}{\text{ground state probability}}.$$

We consider only annealing times and exclude programming and readout times from our analysis as these are not part of the algorithms proper.

For a given value of ρ, we choose α to maximize the difficulty of the logical problem (see Figure 1). For each selection of ρ and R, we generate 100 FCL problems at each problem size and solve each problem with each solver.

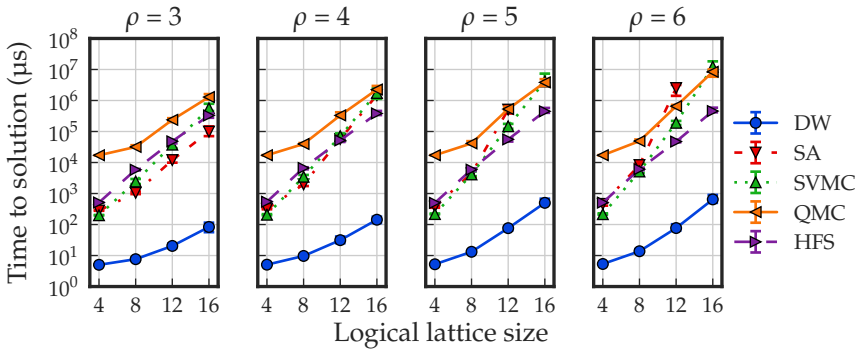

Figure 3: Time to solution for D-Wave and software solvers with range values $\rho \in \{3,4,5,6\}$. For each value of ρ, α is chosen to maximize logical hardness. Shown are median values (over 100 inputs at each size) with 95% confidence intervals.

5.1 Varying ruggedness via logical complexity

In our first experiment, we vary ρ and set $R = \rho$. In this case the only free parameter ρ controls both the ruggedness and the logical complexity of the inputs. Time-to-solution plots are shown in Figure 3. At the largest problem size, the D-Wave QPU is three orders of magnitude faster than the fastest software solver for each value of ρ. D-Wave's speedup over software peaks at 2600 times for $\rho = 4$.

As ρ increases, the impact of local ruggedness increases as the logical Hamiltonian is compressed relative to the local wells induced by the clusters. The performance of SA drops off sharply while the performance of DW and QMC declines gracefully. The performance of HFS decreases only very slightly. HFS is not affected by the local ruggedness because it is tailored to the Chimera topology and uses updates that contain entire clusters; the performance degradation is due to the slight increase in logical problem hardness.

All solvers except HFS have strictly convex scaling curves because the anneal lengths are optimized for the largest problem size and are too long for the smaller problems. HFS does not use fixed-length anneals and ends up using shorter anneals on smaller inputs.

Though true scaling is masked by the inability to optimize parameters for smaller inputs [3, 40], we note that the performance of the D-Wave QPU scales at least as well as the software solvers between the two largest problem sizes.

5.2 Varying ruggedness by scaling

In our second experiment, we fix $\rho = 3$ and vary the ruggedness R. This keeps the logical complexity constant, allowing us to isolate the impact of ruggedness on the various solvers. Here we consider only the largest problem size having a 16×16 logical lattice.

Consistent with our findings when varying ρ, tuning the ruggedness directly by varying R increases difficulty dramatically for simulated annealing and less so for other solvers (see Figure 4). Excluding HFS, whose behaviour is constant in this example, the work required by a solver essentially scales according to its quantumness. The D-Wave QPU is

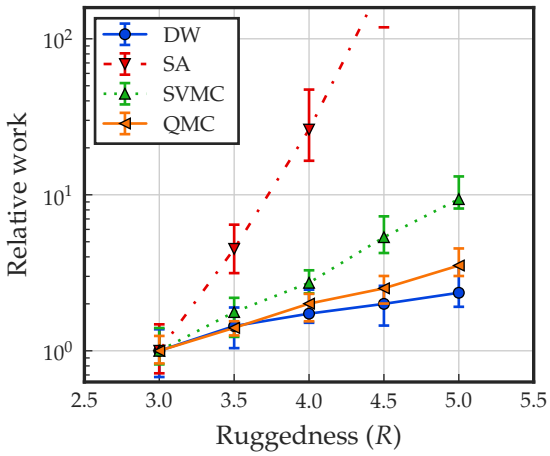

Figure 4: Ruggedness (increasing from left to right) versus relative work for various solvers. Relative work for each solver is calculated as TTS divided by median TTS at $R = 3.0$. The solvers have notably different responses to increasing ruggedness, with SA struggling the most, followed by SVMC, then QMC, then the D-Wave QPU. HFS deals with these energy barriers using exponential brute force; therefore the parameter R does not affect its performance. Markers indicate medians (over 100 inputs) and error bars indicate 95% confidence intervals for the median.

most capable of dealing with ruggedness. QMC—the most faithful classical simulation of quantum annealing—comes next, followed by SVMC, which is a mean-field approximation to QMC. Bringing up the rear is SA, a simulation of a fully classical process.

The improved scaling (versus ruggedness) of QMC over SVMC indicates that crucial information is being lost in the mean-field approximation. The improved scaling of QA (i.e., the D-Wave QPU) over QMC may indicate that QMC is failing to faithfully simulate the dynamics of the QA processor, or it may simply be an artifact of our inability to use faster D-Wave anneals. This bears further investigation using future D-Wave QPUs with faster annealing times, again utilizing the tunable ruggedness of FCL problems.

6 Sampling

The ability of an Ising solver to sample diverse optima has both practical and theoretical importance. Ground state sampling in combinatorial problems is the basis for construction of space-efficient SAT-based membership filters [41, 42]. The associated complexity class, #P —the counting analog of NP—has been the subject of extensive research in theoretical computer science since the 1970s [43]. Sampling from the Boltzmann distribution, in which states with equal energy are sampled with equal probability, is of particular interest in machine learning. Boltzmann samples are used to train Boltzmann machines, a task known to be both hard and useful [44].

While machine learning applications typically depend on finite-temperature Boltzmann sampling, using near-optimal states as well as optimal states, we focus on zero temperature sampling to simplify our investigation. This saves us from having an additional input

parameter β—the inverse temperature—that we would have to either set arbitrarily or determine empirically. Empirical estimation of β can be challenging [45] and basing the target β on the output of a solver would arguably give that solver an unfair advantage.

6.1 Sampling from all valleys

The expected time required for a solver to find all ground states of a problem is known, both in the equiprobable case and the biased case [46]. In the case of an Ising spin problem, ground states often lie in connected valleys in Hamming space, and given one ground state in the cluster it is easy to find the rest. We therefore adopt a more practical metric based on the time required to sample all *valleys* of ground states.

In ground states of FCL problems, all clusters have their spins in agreement; therefore the distance between any two ground states in the native Hamming space is a multiple of 8. However, ground states can be adjacent (i.e., differ by a single spin) in the logical space. We define a *valley* as a set of ground states that are connected in the logical Hamming space. While it is nontrivial to move from one state to another in the same valley because of the single tall, thin energy barrier, it can still be done with a modest amount of postprocessing. We also note that, since FCL problems do not have fields, ground states come in antipodal pairs,[3] and by extension so do valleys. We treat each pair of antipodal valleys as a single valley since it is trivial to move from one to the other.

With valleys defined in this way, we define the time-to-all-valleys (TTAV) metric as the expected amount of annealing time required to draw at least one sample from each valley. This metric captures the hardest part of sampling from these distributions—finding ground states in every mode—and ensures a diverse set of solutions. With at least one sample from each valley, it is possible to find all ground states using only a modest amount of postprocessing. The TTAV metric is most meaningfully interpreted relative to TTS since hitting valleys directly depends on hitting ground states.

6.2 Mining for interesting valley structure

Sampling from all valleys is not always much harder than finding a single ground state—an input may have only a single valley or may have valleys that are all very close in Hamming space. We wish to generate inputs with multiple valleys that are well-separated. Sampling from distributions with multiple, well-separated valleys is particularly hard [47] and has important applications such as classification using deep Boltzmann machines [48].

Because we define valleys as clusters of ground states in the *logical* space, analyzing the valley structure of an input is tractable. The largest logical graphs are 16×16 lattices having treewidth 16, so solving a logical problem using dynamic programming and returning some fixed number of ground states typically takes less than a second. This allows us to mine for inputs having interesting valley structure.

We quantify interesting valley structure using the distribution of spin overlap $P(q)$ [27, 28] at zero temperature, i.e., for two ground states sampled uniformly with replacement,

[3] For Hamiltonians with no fields, flipping all spins of a state does not change the energy. Therefore the antipode (negation) of any ground state is also a ground state.

what fraction of spins do they have in common? The random variable $P(q)$ takes values in the range $[-1, 1]$. For inputs without fields the distribution is symmetric about zero; we can therefore consider the distribution of the absolute value $P(|q|)$. We define the *mean overlap* as the expectation of $P(|q|)$. Inputs with mean overlap near 1 tend to resemble ferromagnets—if there are multiple valleys they will be close together. Inputs with lower mean overlap tend to have valleys that are well-separated.

Inputs that are hard to sample from have multiple valleys that are well-separated. We mine for such inputs as follows. First we reject any input with more than 1000 ground states, as these slow down our analysis and may be too easy. Second, we reject any input that does not have at least 4 valleys since we want valley collection to be nontrivial. Finally, we reject any input with a mean overlap of 0.7 or higher since we want valleys to be well-separated.

6.3 Sampling results

We generated problems at the 16×16 lattice size with $\alpha = 0.85$ and $\rho = R = 6$ and mined them for interesting valley structure as described above. We generated 50,000 inputs and rejected all but 74. This gave us an acceptance rate of roughly 0.15% of inputs. We sought to answer the question, after x seconds of annealing, what fraction of the valleys has each solver seen? For each problem we drew a number of samples according to the solver as follows:

D-Wave QPU:	100,000 samples at 5 µs (in batches of 100 per spin-reversal transform)
SA:	100,000 samples
QMC:	5000 samples
HFS:	5000 samples

6.3.1 Time to all valleys

Results for the TTAV metric are shown in Figure 5. The 2000-qubit D-Wave QPU is the fastest of the solvers, hitting all valleys in a median time of roughly 30 ms. The fastest software competition was HFS, which hit all valleys in a median time of roughly 30 s.

In certain situations quantum annealing in the transverse-field Ising model is subject to inherent sampling bias [49–52], although that does not prove to be a significant problem here. While the slope of the D-Wave curve is slightly less steep than the HFS curve, indicating that its samples might be less diverse, the D-Wave QPU still manages to outperform the competition by about three orders of magnitude.

6.3.2 KL-divergence of valley distributions

The TTAV results shown in Figure 5 fail to address a specific fear—that in a significant minority of inputs there are valleys that the D-Wave QPU would be simply unable to find due to quantum sampling bias. To address this, we calculate for each (solver, problem) pair the KL-divergence between empirical valley distributions and exact valley distributions

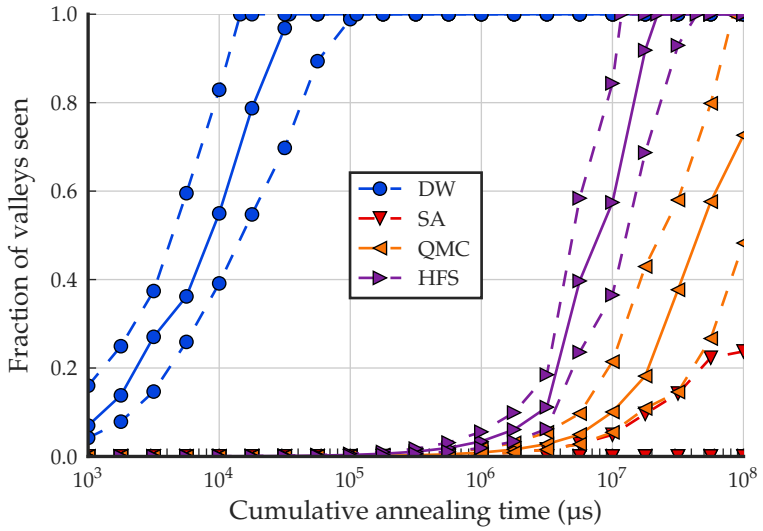

Figure 5: Time to all valleys (TTAV) for various solvers. The x-axis shows elapsed annealing time and the y-axis shows the fraction of valleys that a solver has hit up to that point in time. Solid lines show medians (over 74 inputs) and dashed lines show the 25th and 75th percentiles.

(i.e., relative valley sizes). KL-divergence is an *asymmetric* measure of the distance between two probability distributions; we calculate it such that it is infinite if the solver fails to see all valleys, i.e.,

$$\mathrm{KLD} = \sum_{\text{valleys } v} P(v) \log \frac{P(v)}{\widehat{P}(v)},$$

where $P(v)$ is the true Boltzmann probability of valley v and $\widehat{P}(v)$ is the sample estimate of $P(v)$, conditioned on samples being ground states. This KL-divergence measure includes two types of error. First, there is a distributional error, since each solver samples from a distribution that differs from the Boltzmann distribution. Second, there is a sample size error, since our sample estimate has finite size and therefore differs from the solver's true distribution. In this context it is appropriate to include both types of error.

Figure 6 shows histograms of KL-divergence for the different solvers. For these FCL problems, fears of valleys suppressed by quantum sampling bias are unfounded. The D-Wave QPU has a superior KL-divergence distribution than any of the software solvers even when annealing for three orders of magnitude less time. On the single input for which the D-Wave KLD was infinite because at least one valley was never seen, it was also infinite for all other solvers.

6.3.3 Raw error on model marginals

The TTAV metric and valley distributions can be thought of as representing what sample quality would look like with postprocessing. We would also like a more raw metric that does not have this implicit postprocessing. For this we consider marginals of the zero-temperature Boltzmann distribution. Specifically, we consider the spin-spin expectations,

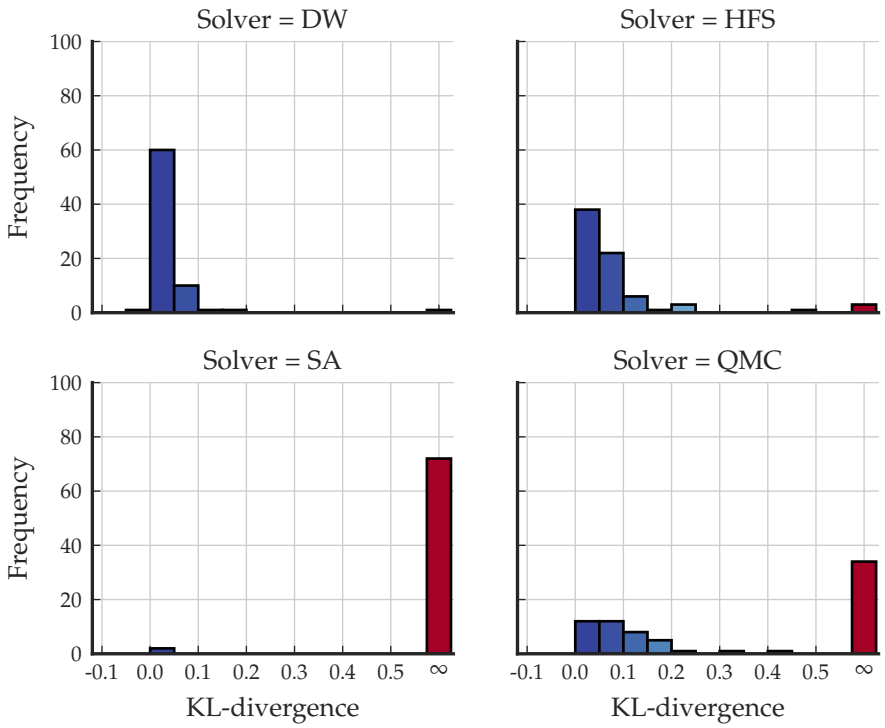

Figure 6: KL-divergence histograms. Shown are the empirical distributions (out of 74 inputs) of the KL-divergence achieved by each solver in estimating the valley distributions. Where the value is infinite, the solver failed to see one or more of the valleys. The D-Wave QPU had the best performance in this metric—even with three orders of magnitude less annealing time—followed by HFS, then QMC, then SA.

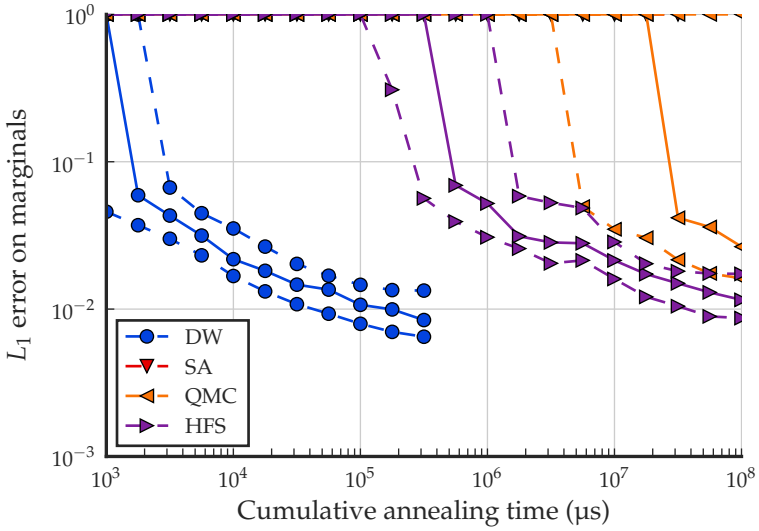

Figure 7: Elapsed annealing time versus L_1 error of marginal estimation for various solvers. Solid lines show medians (over 74 inputs) and dashed lines show the 25th and 75th percentiles. The D-Wave QPU achieves the same error as software solvers in roughly three orders of magnitude less time.

i.e., for each coupler, what is the expected product of the two incident spins in the zero-temperature Boltzmann distribution? The L_1 error on these marginals (i.e., the empirical estimates of spin-spin expectations minus true expectations) is a well-established metric of interest in the study of undirected graphical models (see, e.g., [53]).

Figure 7 shows the decay in error as more samples are taken. We measure errors in the logical space, so couplers within a cluster are ignored. Again, the D-Wave QPU is roughly three orders of magnitude faster than the fastest software solver. We expect that the error on marginals will decay to a certain point, then plateau, with the level of the plateau corresponding to the bias of the solver. As with the TTAV data, potential concerns about the bias of quantum annealers do not seem to play out here. The L_1 error of the D-Wave QPU is still decaying after 100,000 samples; at this point in time (0.5 s) the D-Wave QPU has achieved a median error of less than 1%, a value that software solvers fail to reach in 100 s to match.

7 Power analysis

While much discussion of D-Wave QPUs has centered around various forms of quantum speedup [2, 3], the focus on scaling behavior alone ignores current pain points of high-performance computing (HPC) and hyperscale cloud computing. One of the most pressing concerns for HPC is energy consumption.

The US Department of Energy's Exascale Computing Initiative has the stated goal of deploying an exascale supercomputer—one capable of 1 exaflops, or 10^{18} floating point operations per second—that draws only 20–30 MW of power [54]. This translates to an effi-

ciency of up to 50 gigaflops per watt. By contrast, the world's most powerful supercomputer as of 2017—the Sunway TaihuLight—performs 93 petaflops at an efficiency of 6.1 gigaflops per watt excluding cooling power and 2.2 gigaflops per watt including cooling power [55].

Including the cooling, TaihuLight requires 42 MW of power. The average hydroelectric facility in the US produces 57 MW of power. Using the average price for industrial power in the US which is approximately $600,000 per year per MW, the operating costs are staggering.

More efficient computation is needed and can be achieved using specialized coprocessors. As an example we consider the NVIDIA DGX-1 [56], a highly optimized GPU server capable of 170 teraflops[4] that draws 3.2 kW of power for an efficiency of 53 gigaflops per watt. More efficient computation comes at the expense of generality; for example, it is impossible to run the HFS algorithm on an NVIDIA DGX-1 efficiently, if at all.

If we go to an even more highly-specialized coprocessor, the D-Wave QPU, the benefits in terms of energy efficiency can be massive. In this study the D-Wave QPU scales similarly to QMC and solves problems 10,000 times faster than QMC run on an NVIDIA GTX 1080. Extrapolating based solely on flop rate and computation time, this would be equivalent to roughly 500 NVIDIA DGX-1 servers.[5] The power draw of the D-Wave system is under 25 kW whereas 500 NVIDIA DGX-1 servers would draw 1.6 MW—roughly as much as 1300 American households [57]. The gap shrinks significantly if we include programming and readout time for the D-Wave QPU, but it would still be on the order of 10 times faster than an NVIDIA DGX-1.

In this study HFS has been more energy efficient than QMC because a) it is faster than QMC, and b) it was run on a single CPU core drawing 20 W rather than on a GPU drawing 180 W. Considering pure annealing time, HFS is roughly on par with the 2000-qubit D-Wave QPU in terms of ground state throughput per watt. However, we have noted that HFS is not future proof against denser topologies [1].

Almost all of the power drawn by D-Wave systems is used by the dilution refrigerator. This has remained constant since the introduction of the first generation of D-Wave system in 2011 and is expected to stay constant as computing power scales with successive generations of QPU.

8 Conclusions

We have introduced a class of synthetic inputs on which to evaluate the performance of annealing-based QPUs, specifically the QPUs developed by D-Wave. This problem class is more representative of real-world problems and provides an alternative to the Google problems of Denchev et al. [1], which were more highly tuned to highlight the utility of

[4] The NVIDIA DGX-1 achieves this flop rate for half-precision 16-bit floating point operations. Reducing precision in exchange for faster operations is often beneficial in machine learning.

[5] This back-of-the-envelope calculation of ground state throughput rate favors classical solvers for two reasons. First, it is valid only in the case where we have a large number of independent jobs to run in parallel. In practice, parallelizability across devices will be limited by the number of concurrent jobs that can be run since all of the algorithms we consider are dominated by sequential loops. Second, our calculation ignores communication time between devices, though in this case we would not expect that to be dominant.

quantum tunneling.

The D-Wave QPU is up to 2600 times faster than all software solvers considered and typically on the order of 1000 times faster at the largest problem size. These software solvers now include GPU implementations of SA, QMC, and SVMC as well as a CPU implementation of HFS, making the competition much stronger than that analyzed by Denchev et al. [1]. The set of software solvers we used was representative, but not exhaus-tive—in particular, we hope to include more of the solvers used by Mandrà et al. [2] in future studies.

These inputs have tunable ruggedness controlled either by the range parameter ρ or by the scaling parameter S. Of particular interest is the fact that QMC performed better relative to SA when the ruggedness is increased, and physical quantum annealing performed better still.

We also evaluated the 2000-qubit D-Wave QPU on the task of zero-temperature Boltzmann sampling, i.e., sampling uniformly from ground states. While concerns have been raised that quantum and analog sampling bias might make it difficult for quantum annealers to sample from Boltzmann distributions [49–51], there was little evidence for such a struggle in this study. In several metrics considered, the 2000-qubit D-Wave QPU maintains its speed advantage of roughly three orders of magnitude and provides sample diversity that is as good as, or better than, the software competition.

References

[1] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, "What is the computational value of finite-range tunneling?" *Phys. Rev. X*, vol. 6, p. 031 015, 3 2016.

[2] S. Mandrà, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, "Strengths and weaknesses of weak-strong cluster problems: A detailed overview of state-of-the-art classical heuristics versus quantum approaches," *Phys. Rev. A*, vol. 94, p. 022 337, 2 2016.

[3] T. Rønnow, Z. Wang, J. Job, S. Boixo, S. Isakov, D. Wecker, J. Martinis, D. Lidar, and M. Troyer, "Defining and detecting quantum speedup," *Science*, vol. 345, no. 6195, pp. 420–424, 2014.

[4] H. Katzgraber, F. Hamze, and R. Andrist, "Glassy Chimeras could be blind to quantum speedup: Designing better benchmarks for quantum annealing machines," *Physical Review X*, vol. 4, no. 2, p. 021 008, 2014.

[5] S. Kauffman and S. Levin, "Towards a general theory of adaptive walks on rugged landscapes," *Journal of theoretical Biology*, vol. 128, no. 1, pp. 11–45, 1987.

[6] S. A. Kauffman and E. D. Weinberger, "The NK model of rugged fitness landscapes and its application to maturation of the immune response," *Journal of theoretical biology*, vol. 141, no. 2, pp. 211–245, 1989.

[7] E. Weinberger, "NP completeness of Kauffman's N-k model, a tuneable rugged fitness landscape," Santa Fe Institute, Working Papers, 1996.

[8] A. Selby, "Efficient subgraph-based sampling of ising-type models with frustration," *ArXiv preprint arXiv:1409.3934*, 2014.

[9] ——, *Qubo-chimera*, `https://github.com/alex1770/QUBO-Chimera`, GitHub repository, 2013.

[10] F. Hamze and N. de Freitas, "From fields to trees," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, AUAI Press, 2004, pp. 243–250.

[11] P. Bunyk, E. Hoskinson, M. Johnson, E. Tolkacheva, F. Altomare, A. Berkley, R. Harris, J. Hilton, T. Lanting, A. Przybysz, *et al.*, "Architectural considerations in the design of a superconducting quantum annealing processor," *IEEE Transactions on Applied Superconductivity*, 2014.

[12] N. G. Dickson, M. W. Johnson, M. H. Amin, R Harris, F Altomare, *et al.*, "Thermally assisted quantum annealing of a 16-qubit problem.," *Nature communications*, vol. 4, no. May, 2013, ISSN: 2041-1723.

[13] R. Harris, M. Johnson, T. Lanting, A. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, *et al.*, "Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor," *Physical Review B*, vol. 82, no. 2, p. 024 511, 2010.

[14] M. Johnson, M. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. Berkley, J. Johansson, P. Bunyk, *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[15] T. Lanting, A. Przybysz, A. Smirnov, F. Spedalieri, M. Amin, A. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, *et al.*, "Entanglement in a quantum annealing processor," *Physical Review X*, vol. 4, no. 2, p. 021 041, 2014.

[16] N. Robertson and P. D. Seymour, "Graph minors. II. Algorithmic aspects of tree-width," *Journal of algorithms*, vol. 7, no. 3, pp. 309–322, 1986.

[17] S. Boixo, T. Albash, F. Spedalieri, N. Chancellor, and D. Lidar, "Experimental signature of programmable quantum annealing," *Nature communications*, vol. 4, 2013.

[18] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Physical Review E*, vol. 58, no. 5, p. 5355, 1998.

[19] 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," *Science*, vol. 292, no. 5516, pp. 472–475, 2001.

[20] S. Boixo, V. Smelyanskiy, A. Shabani, S. Isakov, M. Dykman, V. Denchev, M. Amin, A. Smirnov, M. Mohseni, and H. Neven, "Computational multiqubit tunnelling in programmable quantum annealers," *Nature communications*, vol. 7, 2016.

[21] J. King, S. Yarkoni, M. Mohammadi Nevisi, J. P. Hilton, and C. C. McGeoch, "Benchmarking a quantum annealing processor with the time-to-target metric," p. 29, 2015. arXiv: 1508.05087.

[22] E. Weinberger, "Correlated and uncorrelated fitness landscapes and how to tell the difference," *Biological cybernetics*, vol. 63, no. 5, pp. 325–336, 1990.

[23] V. K. Vassilev, T. C. Fogarty, and J. F. Miller, "Information characteristics and the structure of landscapes," *Evolutionary computation*, vol. 8, no. 1, pp. 31–60, 2000.

[24] W. S. Kendall, F. Liang, and J.-S. Wang, *Markov chain Monte Carlo: Innovations and applications*. World Scientific, 2005, vol. 7.

[25] W. Janke, *Rugged free energy landscapes: Common computational approaches to spin glasses, structural glasses and biological macromolecules*. Springer, 2007, vol. 736.

[26] M. Pelikan, K. Sastry, D. E. Goldberg, M. V. Butz, and M. Hauschild, "Performance of evolutionary algorithms on nk landscapes with nearest neighbor interactions and tunable overlap," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ACM, 2009, pp. 851–858.

[27] B. Yucesoy, J. Machta, and H. G. Katzgraber, "Correlations between the dynamics of parallel tempering and the free-energy landscape in spin glasses," *Phys. Rev. E*, vol. 87, p. 012 104, 1 2013.

[28] H. G. Katzgraber, H. Firas, Z. Zhu, A. J. Ochoa, and H. Munoz-Bauza, "Seeking quantum speedup through spin glasses: The good, the bad, and the ugly," *Physical Review X*, vol. 5, no. 031026, 2015.

[29] Z. Zhu, A. Feldman, S. Isakov, H. G. Katzgraber, S. Mandrà, H. Munoz-Bauza, A. Ochoa, A. Ozaeta, and A. Perdomo-Ortiz, "Predicting quantum tunneling advantage in random spin-glass and application problems," Adiabatic Quantum Computing Conference 2016, 2016.

[30] H. G. Katzgraber, S. Boixo, V. V. Denchev, F. Hamze, S. Isakov, *et al.*, "Quantum vs classical optimization: A status update on the arms race," Adiabatic Quantum Computing Conference 2016, 2016.

[31] C. P. Gomes and B. Selman, "Algorithm portfolios," *Artificial Intelligence*, vol. 126, no. 1, pp. 43–62, 2001.

[32] S. Venkatasubramanian, R. W. Vuduc, *et al.*, "Tuned and wildly asynchronous stencil kernels for hybrid cpu/gpu systems," in *Proceedings of the 23rd international conference on Supercomputing*, ACM, 2009, pp. 244–255.

[33] D. Grewe and M. F. P. O'Boyle, "A static task partitioning approach for heterogeneous systems using OpenCL," in *International Conference on Compiler Construction*, Springer, 2011, pp. 286–305.

[34] I. Hen, J. Job, T. Albash, T. F. Rønnow, M. Troyer, and D. A. Lidar, "Probing for quantum speedup in spin-glass problems with planted solutions," *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 92, no. 4, pp. 1–22, 2015, ISSN: 10941622. arXiv: 1502.01663v2.

[35] A. D. King, T. Lanting, and R. Harris, "Performance of a quantum annealer on range-limited constraint satisfaction problems," *ArXiv preprint arXiv:1502.02098*, 2015.

[36] K. L. Pudenz, T. Albash, and D. A. Lidar, "Quantum annealing correction for random Ising problems," *Phys. Rev. A*, vol. 91, no. 4, p. 42 302, 2015.

[37] K. Hukushima and K. Nemoto, "Exchange Monte Carlo method and application to spin glass simulations," *Journal of the Physical Society of Japan*, vol. 65, no. 6, pp. 1604–1608, 1996.

[38] R. A. Baños, A. Cruz, L. A. Fernandez, J. M. Gil-Narvion, A. Gordillo-Guerrero, *et al.*, "Nature of the spin-glass phase at experimental length scales," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2010, no. 06, P06026, 2010.

[39] A. M. Ferreiro, J. A. García, J. G. López-Salas, and C. Vázquez, "An efficient implementation of parallel simulated annealing algorithm in GPUs," *Journal of Global Optimization*, vol. 57, no. 3, pp. 863–890, 2012, ISSN: 1573-2916.

[40] M. H. Amin, "Searching for quantum speedup in quasistatic quantum annealers," *Physical Review A*, vol. 92, no. 5, p. 052 323, 2015, ISSN: 1050-2947. arXiv: 1503.04216.

[41] S. A. Weaver, K. J. Ray, V. W. Marek, A. J. Mayer, and A. K. Walker, "Satisfiability-based set membership filters," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 8, pp. 129–148, 2014.

[42] A. Douglass, A. D. King, and J. Raymond, "Theory and applications of satisfiability testing – sat 2015: 18th international conference, austin, tx, usa, september 24-27, 2015, proceedings," in, M. Heule and S. Weaver, Eds., Cham: Springer International Publishing, 2015, ch. Constructi, pp. 104–120, ISBN: 978-3-319-24318-4.

[43] L. G. Valiant, "The complexity of computing the permanent," *Theoretical computer science*, vol. 8, no. 2, pp. 189–201, 1979.

[44] P. M. Long and R. Servedio, "Restricted boltzmann machines are hard to approximately evaluate or simulate," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 703–710.

[45] J. Raymond, S. Yarkoni, and E. Andriyash, "Global warming: temperature estimation in annealers," *Frontiers in ICT*, vol. 3, p. 23, 2016, ISSN: 2297-198X.

[46] P. Flajolet, D. Gardy, and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207–229, 1992, ISSN: 0166-218X.

[47] R. M. Neal, "Sampling from multimodal distributions using tempered transitions," *Statistics and Computing*, vol. 6, no. 4, pp. 353–366, 1996, ISSN: 1573-1375.

[48] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep Boltzmann machines," in *Advances in neural information processing systems*, 2012, pp. 2222–2230.

[49] T. Albash, W. Vinci, A. Mishra, P. A. Warburton, and D. A. Lidar, "Consistency tests of classical and quantum models for a quantum annealer," *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 91, no. 4, 2015, ISSN: 10941622. arXiv: 1403.4228.

[50] Y. Matsuda, H. Nishimori, and H. G. Katzgraber, "Ground-state statistics from annealing algorithms: quantum versus classical approaches," *New Journal of Physics*, vol. 11, no. 7, p. 73 021, 2009.

[51] S. Mandrà, Z. Zhu, and H. G. Katzgraber, "Exponentially-biased ground-state sampling of quantum annealing machines with transverse-field driving Hamiltonians," p. 6, 2016. arXiv: 1606.07146.

[52] A. D. King, E. Hoskinson, T. Lanting, E. Andriyash, and M. H. Amin, "Degeneracy, degree, and heavy tails in quantum annealing," *Phys. Rev. A*, vol. 93, no. 5, p. 52 320, 2016.

[53] A. Globerson and T. S. Jaakkola, "Approximate inference using conditional entropy decompositions," *International Workshop on Artificial Intelligence and Statistics*, 2007.

[54] U.S. Department of Energy Office of Science and National Nuclear Security Administration, *Preliminary conceptual design for an exascale computing initiative*, http://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20141121/Exascale_Preliminary_Plan_V11_sb03c.pdf, 2014.

[55] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao, *et al.*, "The Sunway TaihuLight supercomputer: System and applications," *Science China Information Sciences*, vol. 59, no. 7, p. 072 001, 2016.

[56] *NVIDIA DGX-1 Deep Learning System*, http://images.nvidia.com/content/technologies/deep-learning/pdf/61681-DB2-Launch-Datasheet-Deep-Learning-Letter-WEB.pdf, NVIDIA Corporation, 2016.

[57] US Energy Information Administration, *Frequently asked questions*, https://www.eia.gov/tools/faqs/faq.cfm?id=97&t=3, 2016.

[58] A. Kone and D. A. Kofke, "Selection of temperature intervals for parallel-tempering simulations," *The Journal of Chemical Physics*, vol. 122, no. 20, 206101, 2005.

[59] W. Janke, "Statistical analysis of simulations: Data correlations and error estimation," *Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms*, vol. 10, pp. 423–445, 2002.

[60] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[61] B. Heim, T. F. Rønnow, S. V. Isakov, and M. Troyer, "Quantum versus classical annealing of Ising spin glasses," *Science*, vol. 348, no. 6231, pp. 215–217, 2015.

[62] S. W. Shin, G. Smith, J. A. Smolin, and U. Vazirani, "How "quantum" is the D-Wave machine?" *ArXiv preprint arXiv:1401.7087*, 2014.

[63] U. Wolff, "Collective Monte Carlo updating for spin systems," *Phys. Rev. Lett.*, vol. 62, pp. 361–364, 4 1989.

[64] D. Venturelli, S. Mandrà, S. Knysh, B. O'Gorman, R. Biswas, and V. Smelyanskiy, "Quantum optimization of fully-connected spin glasses," *ArXiv preprint arXiv:1406.7553*, 2014.

[65] Y. Komura and Y. Okabe, "GPU-based single-cluster algorithm for the simulation of the Ising model," *Journal of Computational Physics*, vol. 231, no. 4, pp. 1209–1215, 2012, ISSN: 00219991.

[66] Z. Zhu, C. Fang, and H. G. Katzgraber, "Borealis – a generalized global update algorithm for boolean optimization problems," pp. 1–19, 2016. arXiv: arXiv:1605.09399v1.

[67] B. Bollobas and O. Riordan, *Percolation*. Cambridge University Press, 2006.

A Calculation of decorrelation

The parallel tempering implementation we use to measure decorrelation time is parameterized by a sequence of n increasing inverse temperatures $\beta_i \in [0, \beta = 3]$. A replica sample is initialized randomly for each temperature. The replicas are then iterated, undertaking a random walk in temperature space combined with MCMC sweeps controlled by the energy landscape[6]. Inference on the lower temperature distributions is hardest. The quality of inference is limited by the time scale associated with the temperature random walk. For a sample to decorrelate at low temperature on a practical time scale, the random walk must pass through a high temperature state (where decorrelation is fast) and return to the low temperature state. The time scale is approximated by the integrated autocorrelation of the temperature index [38].

In our setup, temperatures are selected independently for every instance, with β spanning $[0, 30]$ such that the replica exchange rate is equalized at close to 40% (no lower than 33%, no higher than 50%). Equalization of exchange rates is an intuitive and well-studied heuristic for temperature selection and is optimal in special cases [58].[7] Equalization of exchange rates is achieved heuristically by iterating PT, refining the temperature set by linear interpolation of the log empirical exchange rates. To measure autocorrelation time, we undertake a long run of 600,000 sweeps, discarding a conservative portion (10%) of the initial samples which we took as sufficient for burn in (this assumption was tested self-consistently). We then extract the integrated autocorrelation time from the empirical values by an initial sequence estimator [59], we average over the autocorrelations on the n available chains to reduce noise.

B Details of software solvers

[6] At $\beta = 0$, the hottest replica, we replace the sample with a new random uniformly drawn sample on each iteration so that decorrelation is perfect.

[7] We note that, as would be true in studying any problem class, better choices for temperatures and transition operators can lower the autocorrelation times relative to those presented.

B.1 Included software solvers

Our experiments and analyses focus on four algorithms commonly used in comparisons with D-Wave QPUs—three that are highly amenable to GPU implementation and one that is highly tailored to the Chimera topology.

B.1.1 Simulated annealing

Simulated annealing [60] is a simulation of thermal annealing that is widely used as an optimization algorithm. It is the classical analog to quantum annealing. Since simulated annealing is a simple algorithm with very low memory requirements and a high degree of parallelizability, it is ideal for implementation on a GPU.

B.1.2 Quantum Monte Carlo

Quantum Monte Carlo, also known as simulated quantum annealing, is a classical approximation to quantum annealing. For the algorithm to work efficiently on a GPU, we implement the discrete time variant of QMC and fix the number of Trotter slices at 64 so that a worldline can be packed as bits in a word.

While the continuous time variant of QMC is a more faithful simulation of quantum annealing, in particular serving as a bias-free sampler that approaches the quantum Boltzmann distribution in the limit, it has been shown that discrete time QMC can have superior performance as an optimizer [61].

B.1.3 Spin vector Monte Carlo

Spin vector Monte Carlo (SVMC), also known as the O(2)-rotor model, is a mean-field approximation to QMC. SVMC can be thought of as falling between SA and QMC. Proposed for use as an approximation to D-Wave QPUs by Shin, Smith, Smolin, and Vazirani [62], it is also known as the SSSV algorithm. We use a GPU implementation of SVMC that is a minor modification of our implementation of SA.

B.1.4 Hamze-de Freitas-Selby

The Hamze-de Freitas-Selby (HFS) algorithm optimizes by repeatedly optimizing the spins in a low-treewidth induced subgraph of the input, subject to the rest of the input being fixed. The subgraph over which the input is optimized changes at each step. The HFS algorithm is a greedy search algorithm in which reassignment of many variables is considered at once. We used Selby's implementation [8, 9] that is heavily tailored to the Chimera topology; we modified this solver to return each stopping state for consistency with the other heuristic solvers. We note that the HFS algorithm cannot be efficiently implemented on GPU because the memory requirements are too high.

B.2 Excluded software solvers

In addition to these four algorithms, we considered several other software solvers that were prohibitively slow; due to limited time and resources it was not feasible to perform the long software runs needed to optimize parameters and determine ideal performance.

B.2.1 Nontailored HFS

We tested an implementation of HFS that, rather than using subgraphs tailored to the Chimera topology, is topology-agnostic and generates subgraphs dynamically. This nontailored version of HFS performed far worse than Selby's tailored implementation, to the point where we failed to hit ground states in the largest problems. The failure of this algorithm highlights the extent to which Selby's HFS implementation, and specifically the hardcoded subgraphs to update, exploit the sparsity and modularity of the Chimera topology [2]. It is very likely that this type of exploitation will be impossible in future quantum annealer topologies [1].

B.2.2 Wolff cluster Monte Carlo

The Wolff algorithm [63] dynamically detects clusters of spins that should be flipped together. We used a modified implementation that considers the potential change in energy when deciding whether to flip a cluster, similar to Venturelli et al. [64]. This algorithm would, at first glance, be ideal for FCL problems due to the crucial role of clusters. However, finding clusters is slow and our CPU implementation was not competitive with other solvers. Note that the Wolff algorithm is not particularly amenable to GPU implementation; such implementations exist for topologies such as lattices [65] but they achieve only modest speedups over CPU implementations.

B.2.3 Parallel tempering

Parallel tempering runs multiple replicas of a Monte Carlo simulation at different temperatures in parallel, and can exchange information between replicas according to certain exchange rules. It is the algorithm of choice for approximately calculating features and statistics of an energy landscape when exact calculation is prohibitive. Our GPU implementation used 64 replicas, with replicas of a spin packed bitwise into a word, similar to our implementation of QMC. Parallel tempering is more powerful than simulated annealing, but in this case proved to be uncompetitive due to the increased cost of each step.

B.2.4 PT-ICM

In vanilla parallel tempering, replica exchange steps are combined with single-spin Monte Carlo updates. However, replica exchanges can be combined with other types of updates. Isoenergetic cluster move (ICM) updates can be combined with replica exchange and simple Monte Carlo updates. This is sometimes called the PT-ICM algorithm [2] and it has

Resource	CPU	GPU
Model	Intel®Xeon®CPU E5-2643 v3	NVIDIA®GeForce® GTX 1080
Clock rate	3.4 GHz	1.6 GHz
Cores	6	2560
Concurrent workers	6	1
Power	135 W	180 W

Table 1: Specifications for classical processors used for software solvers.

been implemented as the *borealis* solver by Zhu et al. [66]. We implemented PT-ICM as described by Zhu et al. and found that its scaling was similar to Selby's implementation of HFS, but absolute performance was an order of magnitude slower even when using a precomputed ideal temperature ladder (i.e., set of β values) specific to each input.

The advantage that PT-ICM has over Selby's implementation of HFS is that PT-ICM detects clusters dynamically and is not tailored to the underlying topology. Because of this, there may be a misconception that PT-ICM will be future proof against denser quantum processor topologies. However, problems on denser topologies will have smaller site-percolation thresholds [67], and the effectiveness of PT-ICM depends crucially on a large site-percolation threshold [66]. Thus increased processor density will erode the computational value of cluster updates and consequently the advantage of PT-ICM over vanilla PT.

B.3 Classical hardware

Table 1 contains the specifications for the classical processors used. CPU algorithms were run single-threaded on one core each; multiple workers used cores concurrently for independent jobs. GPU algorithms are highly parallelized and each GPU job uses the entire GPU.

B.4 Parameter tuning

For the D-Wave QPU, optimal performance at all problem sizes was achieved at the minimum allowed annealing time of 5 µs. It therefore makes sense to optimize the parameters of software solvers only at the largest problem size; optimizing on a per-size basis would only make scaling look worse for the software solvers. Since we cannot properly optimize the performance of all solvers at all problem sizes, we focus on results at the largest problem size and take scaling results with a grain of salt.

For SA, we chose values of β that increase linearly from 0.01 to 3 and used 10^5 sweeps. For QMC, we used a fixed β of 30 and 10^4 sweeps with the transverse field $A(t)$ decreasing linearly from 1 to 0 and the longitudinal field $B(t)$ increasing linearly from 0 to 1. For SVMC we again used a β of 30 and the same annealing schedule, but used 10^5 sweeps. These values of β were chosen to optimize performance. For HFS, we used Selby's strategy GS-TW2 [8]. For the D-Wave QPU, we use the minimum annealing time of 5 µs.