Title: Ising Processing Units: Potential and Challenges for Discrete Optimization

Author(s): Coffrin, Carleton James
Nagarajan, Harsha
Bent, Russell Whitford

Intended for: Web

Issued: 2017-07-05 (rev.1)

Ising Processing Units: Potential and Challenges for Discrete Optimization

Carleton Coffrin, Harsha Nagarajan, and Russell Bent

Los Alamos National Laboratory, Los Alamos, New Mexico, USA

Abstract. The recent emergence of novel computational devices, such as adiabatic quantum computers, CMOS annealers, and optical parametric oscillators, presents new opportunities for hybrid-optimization algorithms that leverage these kinds of specialized hardware. In this work, we propose the idea of an Ising processing unit as a computational abstraction for these emerging tools. Challenges involved in using and benchmarking these devices are presented, and open-source software tools are proposed to address some of these challenges. The proposed benchmarking tools and methodology are demonstrated by conducting a baseline study of established solution methods to a D-Wave 2X adiabatic quantum computer, one example of a commercially available Ising processing unit.

1 Introduction

As the challenge of scaling traditional transistor-based Central Processing Unit (CPU) technology continues to increase, experimental physicists and high-tech companies have begun to explore radically different computational technologies, such as adiabatic quantum computers (AQCs) [29], gate-based quantum computers [28,43], CMOS annealers [52,53], neuromorphic computers [42], and optical parametric oscillators [40,27,30]. The goal of all of these technologies is to leverage the dynamical evolution of a physical system to perform a computation that is challenging to emulate using traditional CPU technology (e.g., the simulation of quantum physics) [19]. Despite their entirely disparate physical implementations, AQCs, CMOS annealers, and optical parametric oscillators are unified by a common mathematical abstraction known as the Ising model, which has been widely adopted by the physics community for the study of naturally occurring discrete optimization [12]. Furthermore, this kind of "Ising machine" [40,27] is already commercially available with more than 2000 decision variables in the form of AQCs developed by D-Wave Systems [16].

The emergence of physical devices that can quickly solve Ising models is particularly relevant to the optimization and operations research communities, because the core purpose of these devices is to perform discrete optimization. As this technology matures, it may be possible for this specialized hardware to rapidly solve classical combinatorial problems, such as Max-Cut [24] or Max-Clique [37]. Preliminary studies have even suggested that some classes of Constraint Satisfaction Problems may be effectively encoded in such devices because of their combinatorial structure [6,5,47,50]. Furthermore, an Ising model coprocessor could have significant impacts on solution methods for a variety of fundamental combinatorial problem classes, such as MAX-SAT [21,44] and integer programming [46]. At this time, however, it remains unclear how established optimization algorithms should leverage this emerging technology.

Similar to an arithmetic logic unit or a graphics processing unit (GPU), we propose the idea of an Ising processing unit (IPU) as the computational abstraction for physical devices that perform optimization of Ising models. This work first provides a brief introduction to the IPU abstraction and develops the mathematical foundations and real-world considerations of this novel hardware. Second, it conducts a baseline benchmarking study of an IPU to demonstrate the current capabilities of such a device. Hence, this work makes both literature survey and technical contributions. The survey contributions include (1) presenting foundational Ising model results from other disciplines in terminology that is familiar to the optimization community (Section 2), and (2) highlighting key features of IPUs to provide context for future algorithmic developments utilizing these analog devices (Section 3). The technical contributions include (1) developing a benchmarking methodology for IPUs, which is enabled by proposed open-source software tools (Section 4); and (2) demonstrating the proposed benchmarking tools by conducting a baseline evaluation of a D-Wave 2X IPU to support future work in the area (Section 5). To the best of our knowledge, this is the first benchmarking study to use an integer programming solver for identifying challenging IPU test cases. Because of the maturity and commercial availability of the D-Wave IPU, this work often refers to that architecture as an illustrative example. However, based on our understanding of other IPU technologies, the methods and tools proposed herein are applicable to all emerging IPU hardware realizations.

2 A Brief Introduction to Ising Models

This section introduces the notations of the paper and provides a brief introduction to Ising models because they are the core mathematical abstraction of IPUs. The Ising model refers to the class of graphical models where the nodes, $\mathcal{N}$, represent *spin* variables (i.e., $\sigma_i \in \{-1, 1\} \; \forall i \in \mathcal{N}$) and the edges, $\mathcal{E}$, represent *interactions* of spin variables (i.e., $\sigma_i \sigma_j \; \forall i,j \in \mathcal{E}$). A local *field* $\boldsymbol{h}_i \; \forall i \in \mathcal{N}$ can be specified for each node, and an interaction strength $\boldsymbol{J}_{ij} \; \forall i,j \in \mathcal{E}$ can be specified for each edge. Given these data, the *energy* of the Ising model is defined as

$$E(\sigma) = \sum_{i,j \in \mathcal{E}} \boldsymbol{J}_{ij} \sigma_i \sigma_j + \sum_{i \in \mathcal{N}} \boldsymbol{h}_i \sigma_i \qquad (1)$$

Applications of the Ising model typically consider one of two tasks. Some applications focus on finding the lowest possible energy of the Ising model, namely finding the globally optimal solution of the following binary quadratic optimization problem:

$$\begin{aligned} &\min : E(\sigma) \\ &\text{s.t.: } \sigma_i \in \{-1, 1\} \; \forall i \in \mathcal{N} \end{aligned} \tag{2}$$

Other applications are interested in sampling from the Boltzmann distribution of the Ising model's states:

$$Pr(\sigma) \propto \boldsymbol{e}^{\frac{-E(\sigma)}{\boldsymbol{\tau}}} \tag{3}$$

where $\boldsymbol{\tau}$ is a parameter representing the *effective temperature* of the Boltzmann distribution [54]. It is valuable to observe that in the Boltzmann distribution, the lowest energy states have the highest probability. Hence, the task of sampling from a Boltzmann distribution is similar to the task of finding the lowest energy of the Ising model. Indeed, as $\boldsymbol{\tau}$ approaches 0, the sampling task smoothly transforms into the aforementioned optimization task. This paper focuses exclusively on the mathematical program presented in (2), the optimization task.

Frustration The notion of frustration is common in the study of Ising models and refers to any instance of (2) where the optimal solution, σ^*, satisfies the property

$$E(\sigma^*) > \sum_{i,j \in \mathcal{E}} -|\boldsymbol{J}_{ij}| - \sum_{i \in \mathcal{N}} |\boldsymbol{h}_i| \tag{4}$$

A canonical example is the following three node problem:

$$\boldsymbol{h}_1 = 0, \boldsymbol{h}_2 = 0, \boldsymbol{h}_3 = 0 \tag{5a}$$
$$\boldsymbol{J}_{12} = -1, \boldsymbol{J}_{23} = -1, \boldsymbol{J}_{13} = 1 \tag{5b}$$

Observe that, in this case, there are a number of optimal solutions such that $E(\boldsymbol{\sigma}^*) = -2$ but none such that $E(\sigma) = \sum_{i,j \in \mathcal{E}} -|\boldsymbol{J}_{ij}| = -3$.

Gauge Transformations A valuable property of the Ising model is the gauge transformation, which characterizes an equivalence class of Ising models. For illustration, consider the optimal solution of Ising model S, $\boldsymbol{\sigma}^{s*}$. One can construct a new Ising model T where the optimal solution is the same, except that $\boldsymbol{\sigma}_i^{t*} = -\boldsymbol{\sigma}_i^{s*}$ for a particular node $i \in \mathcal{N}$ is as follows:

$$\boldsymbol{J}_{ij}^t = -\boldsymbol{J}_{ij}^s \; \forall i, j \in \mathcal{E}(i) \tag{6a}$$
$$\boldsymbol{h}_i^t = -\boldsymbol{h}_i^s \tag{6b}$$

where $\mathcal{E}(i)$ indicates the neighboring edges of node i. This S-to-T manipulation is referred to as a gauge transformation. Given a complete source state $\boldsymbol{\sigma}^s$ and a complete target state $\boldsymbol{\sigma}^t$, this transformation is generalized to all of σ by

$$\boldsymbol{J}_{ij}^t = \boldsymbol{J}_{ij}^s \boldsymbol{\sigma}_i^s \boldsymbol{\sigma}_j^s \boldsymbol{\sigma}_i^t \boldsymbol{\sigma}_j^t \; \forall i, j \in \mathcal{E} \tag{7a}$$
$$\boldsymbol{h}_i^t = \boldsymbol{h}_i^s \boldsymbol{\sigma}_i^s \boldsymbol{\sigma}_i^t \; \forall i \in \mathcal{N} \tag{7b}$$

It is valuable to observe that by using this gauge transformation property, one can consider the class of Ising models where the optimal solution is $\sigma^*_i = -1 \; \forall i \in \mathcal{N}$ or any arbitrary vector of $-1, 1$ values without loss of generality.

Bijection of Ising and Boolean Optimization It is also useful to observe that there is a bijection between Ising optimization (i.e., $\sigma \in \{-1, 1\}$) and Boolean optimization (i.e., $x \in \{0, 1\}$). The transformation of $\sigma \Rightarrow x$ is given by

$$x_i = \frac{\sigma_i + 1}{2} \; \forall i \in \mathcal{N} \tag{8a}$$

$$x_i x_j = \frac{\sigma_i \sigma_j + \sigma_i + \sigma_j + 1}{4} \; \forall i, j \in \mathcal{E} \tag{8b}$$

and the inverse $x \Rightarrow \sigma$ is given by

$$\sigma_i = 2x_i - 1 \; \forall i \in \mathcal{N} \tag{9a}$$

$$\sigma_i \sigma_j = 4x_i x_j - 2x_i - 2x_j + 1 \; \forall i, j \in \mathcal{E} \tag{9b}$$

Consequently, any results from solving Ising models are also immediately applicable to the following class of Boolean optimization problems:

$$\begin{aligned} \min : & \sum_{i,j \in \mathcal{E}} \boldsymbol{c}_{ij} x_i x_j + \sum_{i \in \mathcal{N}} \boldsymbol{c}_i x_i \\ \text{s.t.: } & x_i \in \{0, 1\} \; \forall i \in \mathcal{N} \end{aligned} \tag{10}$$

The Ising model provides a clean mathematical abstraction for understanding the computation of IPUs. However, in practice, a number of hardware implementation factors present additional challenges for computing with IPUs.

3 Features of Ising Processing Units

The core inspiration for developing IPUs is to take advantage of the natural evolution of a discrete physical system to find high-quality solutions to an Ising model (2) [29,40,53]. Consequently, to the best of our knowledge, all IPUs developed to date are analog machines, which present a number of challenges that the optimization community is not accustomed to considering.

Effective Temperature The ultimate goal of current IPUs is to solve the optimization problem (2) and determine the globally optimal solution to the input Ising model. In practice, however, a variety of analog factors preclude IPUs from reliably finding globally optimal solutions. As a first-order approximation, current IPUs behave like a Boltzmann sampler (3) with some hardware-specific effective temperature, $\boldsymbol{\tau}$ [7]. It has also been observed that the effective temperature of an IPU can vary around a nominal value based on the Ising model that is being executed [4]. This suggests that the IPU's performance can change based on the structure of the problem input.

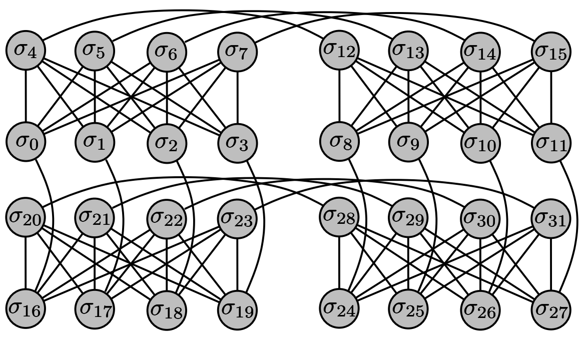

Fig. 1. A 2-by-2 Chimera Graph Illustrating the Variable Product Limitations of a D-Wave 2X IPU.

Environmental Noise One of the primary contributors to the sampling nature of IPUs is environmental factors. All analog machines are subject to faults due to environmental noise; for example, even classical computers can be affected by cosmic rays. However, given the relative novelty of IPUs, the effects of environmental noise are noticeable in current hardware. The effects of environmental noise contribute to the perceived effective temperature $\boldsymbol{\tau}$ of the IPU.

Problem Coefficients In traditional optimization applications, the problem coefficients are often rescaled to best suit floating-point arithmetic. Similarly, IPUs have digital-to-analog converters that can encode a limited number of values; typically these values are represented as numbers in the range of -1 to 1. Some IPUs allow for hundreds of steps within this range, [29,53] whereas others support only the discrete set of {-1, 0, 1} [40]. In either case, the Ising model must be rescaled into the IPU's operating range. The values of the coefficients used in the IPU are critically important because these values are often perturbed by environmental noise and hardware biases.

Coefficient Biases Once an Ising model is input into an IPU, its coefficients are subject to at least two sources of bias. The first source of bias is a model programming error that occurs independently each time the IPU is configured for a computation. This bias is often mitigated by programming the IPU multiple times with an identical input and combining the results from all executions. The second source of bias is a persistent coefficient error, which is an artifact of the IPU manufacturing and calibration process. Because this bias is consistent across multiple IPU executions, this source bias is often mitigated by performing multiple gauge transformations on the input and combining the results from all executions.

Topological Limitations Another significant feature of IPUs is a restricted topology for variable products. In classical optimization (e.g., (2)), it is assumed that every variable can interact with every other variable, that is, an Ising model where an edge connects every pair of variables. However, because of the hardware implementation of an IPU, it may not be possible for some variables to interact. For example, the current D-Wave IPUs are restricted to the *chimera* topology, which is a two-dimensional lattice of *unit cells*, each of which consist of a 4-by-4 bipartite graph (e.g., see Figure 1). In addition to these restrictions, fabrication errors can also lead to random failures of nodes and edges in the IPU hardware. Indeed, as a result of these minor imperfections, every D-Wave IPU developed to date has a unique topology [9,18,32]. Research and development of algorithms for embedding various kinds of Ising models into a specific IPU topology is still an active area of research [6,11,13,34].

3.1 Challenges of Benchmarking Ising Processing Units

These novel IPU features present unique challenges for benchmarking these devices. These challenges fall into roughly two categories: finding interesting Ising models for testing and comparing with classical methods.

Standardized Benchmark Libraries Research and development in optimization algorithms has benefited greatly from standardized benchmark libraries [35,20,26]. However, direct application of these libraries to IPUs is out of scope for the near term for the following reasons: (1) the Ising model is a binary quadratic program, which is sufficiently restrictive to preclude the use of many standard problem libraries; (2) even in cases where the problems of interest can be mapped directly to the Ising model (e.g., Max-Cut, Max-Clique), the task of embedding the given problems onto the IPU's hardware graph can be prohibitive; and (3) even if an embedding can be found, it is not obvious that the problem's coefficients will be amenable to the IPU's operating range [14].

Another option is to build a new standardized benchmark library specifically for the evaluation of IPUs. However, given that every IPU is unique and within an IPU each variable also has individual properties (e.g., persistent bias), it is not immediately clear that a new static benchmark library could be applied to multiple IPUs.

Comparison with Classical Algorithms Because of the radically different hardware of CPUs vs IPUs and the sampling nature of the IPUs, how to conduct a fair comparison of these two technologies is not immediately clear [38,39,33]. For example, is the comparison more fair or less fair if classical algorithms utilize multiple cores or GPUs? Indeed, comparisons of D-Wave's IPU with classical algorithms have resulted in vigorous discussion about what algorithms and metrics should be used to make such comparisons [1,32,2]. It is widely accepted that IPUs do not provide optimality guarantees and should be compared to classical heuristic methods rather than methods providing optimality proofs. However, it is less clear whether heuristics should be specialized to solve problems for a specific IPU architecture (e.g., [49,48]) or whether classical methods should be designed for the most general form of the Ising model (2). This debate will most likely continue for several years. In this work, our goal is not to answer these challenging questions but rather to develop benchmarking tools that will assist researchers in exploring these questions and replicating previous results as the subject of IPUs evolves.

4 Tools for Benchmarking Ising Processing Units

To help address a number of the previously discussed challenges, in this work we propose three open-source tools for assisting in benchmarking IPUs:

- BQPJSON: a language-independent json-based test case exchange format designed with IPUs in mind[1]
- DWIG: a collection of algorithms for IPU test case generation[2]
- BQPSOLVERS: a collection of tools for encoding BQPJSON data into various optimization solvers[3]

We now review the design motivations and core features of each of these tools.

4.1 Data Management

The primary design goals of BQPJSON are to (1) encode all of the information required to replicate the exact execution that was performed on a specific IPU hardware; (2) allow the IPU test case to be transformed back to the original problem domain, if one exists; and (3) not be prescriptive about the mathematical representation of the test case (i.e., (2) or (10)). To that end, BQPJSON encodes the following class of binary quadratic programs, which is a generalization of (2) and (10):

$$\begin{aligned} \min : \boldsymbol{s}\left(\sum_{i,j\in\mathcal{E}} \boldsymbol{c}_{ij} b_i b_j + \sum_{i\in\mathcal{N}} \boldsymbol{c}_i b_i + \boldsymbol{o}\right) \quad (11) \\ \text{s.t.: } b_i \in \{0,1\}\ \forall i \in \mathcal{N} \text{ or } b_i \in \{-1,1\}\ \forall i \in \mathcal{N} \end{aligned}$$

The key features of this model are (1) the decision variables can be represented as either spin (i.e., $\{-1,1\}$) or Boolean (i.e., $\{0,1\}$) values; (2) the objective offset term, $\boldsymbol{o}$, is required in order to ensure an idempotent transformation between the spin and Boolean variable domains; (3) the explicit scaling term, $\boldsymbol{s}$, allows for the problem coefficients (i.e., $\boldsymbol{c}$) to be specified within an IPU's operating range while preserving the units of the problem's original objective function; and (4) the collection of decision variables $\mathcal{N}$ allows the variable identifiers to have arbitrary values, such as the names used by specific IPU hardware (unlike traditional test case formats, where the variables are numbered from 1 to n).

[1] The source code is available at `https://github.com/lanl-ansi/bqpjson`
[2] The source code is available at `https://github.com/lanl-ansi/dwig`
[3] The source code is available at `https://github.com/lanl-ansi/bqpsolvers`

In addition to these problem specification features, BQPJSON also has extensive support for documenting test cases. A description section provides human-readable information about the test case, whereas a metadata section is used to encode useful machine-readable information. BQPJSON also supports encoding solutions so that test cases can include details about target or best-known solutions to specific cases.

Implementation and Tools The current implementation of BQPJSON is provided as a free and open-source Python package that can be installed from the source code repository or from the Python Package Index (PyPI). This package includes (1) detailed documentation of the BQPJSON file format; (2) tools for validation of JSON data to ensure they conform to the BQPJSON specification; (3) BQPJSON-to-BQPJSON idempotent transformations between the spin and Boolean variable domains; (4) translation of BQPJSON data into other established problem formats, such as QUBO [10] and MiniZinc [45]; and (5) a collection of command line tools for bash script processing of BQPJSON data.

4.2 Case Generation

Because of the challenges associated with mapping established optimization test cases to the IPU hardware, the IPU benchmarking community has adopted the practice of generating tests on a case-by-case basis for specific IPUs [32,25,33,18]. This practice amounts to finding interesting values for **h** and **J** in (1). In some cases the procedures for generating these values are elaborate [18,31] and are designed to leverage theoretical results about Ising models [25]. To ensure the experimental reproducibility in the context of IPU-specific case generation, in this paper we developed the D-Wave Instance Generator (DWIG). DWIG represents a collection of algorithms that generate well-studied random Ising models. Our hope is that as long as the same test case generation algorithm is used, the average values of previous experimental results will be replicable across multiple IPUs. The DWIG script currently implements five problem classes proposed in the literature [32,31,33,18], each of which we will briefly introduce. For a detailed description, please refer to the source publication of the problem class.

Random (RAN-k and RANF-k) To the best of our knowledge, the RAN-k problem was first proposed in [32] and consists simply of assigning each value of **h** to 0 and each value of **J** uniformly at random from the set

$$\{-\boldsymbol{k}, -\boldsymbol{k}+1, \ldots, -2, -1, 1, 2, \ldots, \boldsymbol{k}-1, \boldsymbol{k}\} \tag{12}$$

The RANF-k problem is a simple variant of RAN-k where the values of **h** are also selected uniformly at random from (12). As we will later see, RAN-1 and RANF-1, where $\boldsymbol{h}, \boldsymbol{J} \in \{-1, 1\}$, are an interesting subclass of this problem.

Frustrated Loops (FL-k and FCL-k) The frustrated loop problem was originally proposed in [25] and then later refined to the FL-k problem in [31]. It consists of generating a collection of random cycles in the IPU graph. In each cycle, all of the edges are set to -1 except one random edge, which is set to 1 to produce *frustration.* A scaling factor α is used to control how many random cycles should be generated, and the parameter k determines how many cycles each edge can participate in. A key property of the FL-k generation procedure is that a globally optimal solution is maintained at $\sigma_i = -1 \; \forall i \in \mathcal{N}$ and $\sigma_i = 1 \; \forall i \in \mathcal{N}$ [31]. However, by default, DWIG uses a gauge transformation to obfuscate this solution pair to a random assignment of σ.

A variant of the frustrated loop problem is the frustrated *cluster* loop problem, FCL-k [33]. The FCL-k problem is inspired by the chimera network topology (i.e., Figure 1). The core idea is that tightly coupled variables (e.g., $\sigma_1..\sigma_7$ in Figure 1) should form a *cluster* where all of the variables take the same value. This is achieved by setting all of the values of $\boldsymbol{J}$ within the cluster to -1. For the remaining edges between clusters, the previously described frustrated cycles generation scheme is used. It is important to note that a polynomial time algorithm is known for solving the FCL-k problem class on chimera graphs [39].

It is worthwhile to mention that the FL-k and FCL-k case generators are effectively solving a cycle packing problem on the IPU graph. Hence, the randomized algorithm implemented in DWIG is not guaranteed to find a solution if one exists, and in practice it fails for the highly constrained settings of α and k.

Weak-Strong Cluster Networks (WSCNs) The WSCN problem was proposed in [18] and is highly specialized to the chimera network topology. The basic building block of a WSCN is a pair of spin clusters in the chimera graph (e.g., $\sigma_1..\sigma_7$ and $\sigma_8..\sigma_{15}$ in Figure 1). In the *strong* cluster the values of $\boldsymbol{h}$ are set to the strong force parameter *sf* and in the *weak* cluster the values of $\boldsymbol{h}$ are set to the weak force parameter *wf*. All of the values of $\boldsymbol{J}$ within and between this cluster pair are set to -1. Once a number of weak-strong cluster pairs have been placed, the strong clusters are connected to each other using random values of $\boldsymbol{J} \in \{-1, 1\}$. The values of $sf = -1.0$ and $wf = 0.44$ are recommended by [18]. The motivation for the WSCN design is that the clusters create many deep local minima that are difficult for local search methods to escape.

Implementation The current implementation of DWIG is provided as a Python script. The implementation takes an IPU hardware specification as input and produces BQPJSON test cases for each of the problem classes described above.

4.3 Baseline Solution Methods

Because of the novelty of BQPJSON, solvers do not currently support loading BQPJSON cases directly. Hence, to make BQPJSON usable we provide the BQPSOLVERS code base, which includes a collection of simple tools for running BQPJSON data on a variety of established Ising model optimization algorithms. BQPSOLVERS is designed as a collaborative space where state-of-the-art BQPJSON solvers can be community-curated and leveraged for future IPU evaluations.

At this time, BQPSOLVERS includes connections to the following solution methods: (1) mixed integer programming (MIP) [17,8], (2) large neighborhood search (LNS) via the Hamze-Freitas-Selby (HFS) algorithm [23,49], and (3) adiabatic quantum computation (AQC) via the D-Wave IPU [29]. Each of these is briefly introduced.

Mixed Integer Programming Modern MIP solvers are highly optimized for Boolean variables, so we choose to develop a MIP model based on the Boolean optimization variant of the Ising model, (10), as follows:

$$\min : \boldsymbol{s} \left(\sum_{i,j \in \mathcal{E}} \boldsymbol{c}_{ij} x_{ij} + \sum_{i \in \mathcal{N}} \boldsymbol{c}_i x_i + \boldsymbol{o} \right) \tag{13a}$$

s.t.:

$$x_{ij} \geq x_i + x_j - 1 \;\forall i,j \in \mathcal{E} \tag{13b}$$

$$x_{ij} \leq x_i \;\forall i,j \in \mathcal{E} \tag{13c}$$

$$x_{ij} \leq x_j \;\forall i,j \in \mathcal{E} \tag{13d}$$

$$x_{ij} \in \{0,1\} \;\forall i,j \in \mathcal{E}$$

$$x_i \in \{0,1\} \;\forall i \in \mathcal{N}$$

In this formulation we convert the binary quadratic program defined in (10) to a binary linear program by lifting the variable products $x_i x_j$ into a new variable x_{ij} and adding linear constraints to capture the $x_{ij} = x_i \wedge x_j \;\forall i,j \in \mathcal{E}$ constraints [8]. This formulation was implemented using Gurobi [22].

Large Neighborhood Search The state-of-the-art heuristic for solving Ising models on the chimera graphs is an LNS-based method called HFS [23,49]. The core idea of this algorithm is to extract low treewidth subgraphs of the given Ising model and then use dynamic programming to compute the optimal configuration of these subgraphs. This extract and optimize process is repeated until a specified time limit is reached. This solver utilizes a highly optimized open-source C implementation of HFS [48] and leverages the BQPJSON tools to convert BQPJSON data into the HFS's proprietary data format.

Adiabatic Quantum Computation The AQC solver runs the provided BQPJSON model natively on a D-Wave IPU. This solver assumes that the BQPJSON model was generated on the specified IPU and that the variable identifiers used in the BQPJSON model will map directly to the IPU hardware. No attempt is made to transform a general BQPJSON model into the specified IPU hardware. Recalling that the IPU behaves like a Boltzmann sampler rather than a classical optimization solver, this code takes a number of samples from the IPU (typically 10,000) and reports the best solution found among all of the samples, making the IPU's output consistent with the other optimization tools considered here.

Implementation All of the BQPSOLVERS are provided as independent Python scripts with consistent input and output interfaces.

5 A Study of Established Methods

In this section we conduct an in-depth computational study of the IPU instance generation procedures and possible solution methods. The first goal of this study is to understand what classes of problems are most useful for the study of IPUs. The second goal is to provide a baseline of comparison for future works in this area. This computational study is divided into two phases. First, we focus on a broad parameter sweep of all possible instance generation procedures and use a MIP solver to determine which of the problem classes are most challenging. Second, after the most challenging problem classes have been identified, a detailed study is conducted to compare and contrast the MIP, LNS, and AQC solution methods.

Throughout this section, the following notations are used to describe the runtime analysis of the algorithms: UB denotes the objective value of the best feasible solution produced by the algorithm within the time limit, LB denotes the value of the best lower bound produced by the algorithm within the time limit, T denotes the algorithm runtime in seconds, TO denotes that the algorithm hit a time limit of 600 seconds, $\mu(\cdot)$ denotes the mean of a collection of values, $sd(\cdot)$ denotes the standard deviation of a collection of values, and $max(\cdot)$ denotes the maximum of a collection of values.

Computation Environment The IPU computation is conducted on a D-Wave 2X AQC [15]. This computer is a 12-by-12 chimera topology with random omissions; in total, the IPU has 1095 qubits and 3061 couplers and an effective temperature of $\boldsymbol{\tau} \in (0.091, 0.053)$ depending on the problem being solved [51,36]. Unless otherwise noted, the AQC is configured to produce 10,000 samples using a 5-microsecond annealing time per sample and a random gauge transformation every 100 samples. The reported runtime of the AQC reflects the amount of time used on the IPU hardware; it does not include the overhead of communication or scheduling of the computation.

The classical computing algorithms are run on HPE ProLiant XL170r servers with dual Intel 2.10GHz CPUs and 128GB memory. Gurobi 7.0.2 [22] is used as the MIP solver and is configured to use one thread. The highly specialized and optimized HFS algorithm [48] is used as an LNS-based heuristic and also uses one thread.

5.1 Identifying Challenging Cases

Broad Parameter Sweep In this first experiment, we conduct a parameter sweep of all the inputs to the problem generation algorithms described in Section 4.2. Table 1 provides a summary of the input parameters for each problem class. The values of each parameter are encoded with the following triple: (start..stop : step size). When two parameters are required for a given problem class, the cross product of all parameters is used. For each problem class and each combination of parameter settings, 250 random problems are generated in order to produce a reasonable estimate of the average difficulty of that configuration. Each problem

Problem	First Param.	Second Param.
RAN-k	$k \in (1..5:1)$	NA
RANF-k	$k \in (1..5:1)$	NA
FL-k	$k \in (1..5:1)$	$\alpha \in (0..1:0.1)$
FCL-k	$k \in (1..5:1)$	$\alpha \in (0..1:0.1)$
WSCN	$wf \in (-1..1:0.2)$	$sf \in (-1..1:0.2)$

Table 1. Parameter Settings of Various Problems.

Problem	# Cases	$\mu(\lvert\mathcal{N}\rvert)$	$\mu(\lvert\mathcal{E}\rvert)$	$\mu(T)$	$sd(T)$	$max(T)$
RAN	1250	1095	3061	TO	—	TO
RANF	1250	1095	3061	TO	—	TO
FL	6944	1008	2126	1.82	1.06	16.80
FCL	8347	888	2282	4.19	2.81	41.40
WSCN	30250	949	2313	0.25	0.87	17.90

Table 2. MIP Runtime on Various IPU Benchmark Problems (seconds).

is generated using all of the decision variables available on the IPU. The results of this parameter sweep are summarized in Table 2. It is important to note that the FL and FCL generation methods are randomized algorithms and are not guaranteed to succeed. This explains why the number of cases for these problems is not a multiple of 250.

The results presented in Table 2 indicate that, at this problem size, all variants of the FL, FCL, and WSCN problems are easy for modern MIP solvers. This suggests that these problems are not ideal candidates for benchmarking IPUs. In contrast, the RAN and RANF cases consistently hit the runtime limit of the MIP solver, suggesting that these problems are more challenging optimization tasks. This result is consistent with a similar observation in the SAT community, where random SAT problems are known to be especially challenging [41,3]. To get a better understanding of these RAN problem classes, we next perform a detailed study of these problems for various values of the parameter k.

The RAN and RANF Problems In this second experiment, we focus on the RAN-k and RANF-k problems and conduct a detailed parameter sweep of $k \in (1..10:1)$. To accurately measure the runtime difficulty of the problem, we also reduce the size of the problem from 1095 variables to 194 variables so that the MIP solver can reliably terminate within a 600-second time limit. The results of this parameter sweep are summarized in Table 3.

The results presented in Table 3 indicate that (1) as the value of k increases, both the RAN and RANF problems become easier; and (2) the RANF problem is easier than the RAN problem. The latter is not surprising because the additional linear coefficients in the RANF problem break many of the symmetries that exist in the RAN problem. These results suggest that it is sufficient to focus on the RAN-1 and RANF-1 cases for a more detailed study of IPU performance. This is a serendipitous outcome for IPU benchmarking because restricting the problem coefficients to $\{-1,0,1\}$ reduces artifacts caused by noise and the numeral precision of the analog hardware.

k	# Cases	$\mu(\lvert\mathcal{N}\rvert)$	$\mu(\lvert\mathcal{E}\rvert)$	$\mu(T)$	$sd(T)$	$max(T)$	$\mu(T)$	$sd(T)$	$max(T)$
Problems of Increasing k				RAN-k			RANF-k		
1	250	194	528	340.0	195.0	TO	14.10	15.20	82.70
2	250	194	528	89.3	64.3	481	2.97	3.41	22.70
3	250	194	528	64.8	28.3	207	1.67	1.48	10.70
4	250	194	528	58.0	29.5	250	1.25	0.83	6.10
5	250	194	528	49.0	23.0	131	1.12	0.77	6.98
6	250	194	528	49.0	22.4	119	1.05	0.59	4.47
7	250	194	528	45.0	22.8	128	1.04	0.75	7.60
8	250	194	528	44.8	23.7	121	1.01	0.62	5.43
9	250	194	528	42.3	22.3	110	0.98	0.60	5.08
10	250	194	528	39.8	22.1	107	0.91	0.43	3.09

Table 3. MIP Runtime on RAN-k and RANF-k IPU Benchmark Problems (seconds)

5.2 An IPU Evaluation using RAN-1 and RANF-1

Now that the RAN-1 and RANF-1 problem classes have been identified as the most challenging IPU test cases, we perform two detailed studies on these problems using all three algorithmic approaches (i.e., AQC, LNS, and MIP). The first study focuses on the scalability trends of these solution methods as the problem size increases, whereas the second study focuses on a runtime analysis of the largest cases that can be evaluated on our IPU hardware.

Scalability Analysis In this experiment, we increase the problem size gradually to understand the scalability profile of each of the solution methods (AQC, LNS, and MIP). The results are summarized in Table 4. Focusing on the smaller problems, where the MIP solver provides an optimality proof, we observe that both the AQC and the LNS methods find the optimal solution in all of the sampled test cases, suggesting that both heuristic solution methods are of high quality. Focusing on the larger problems, we observe that, in just a few seconds, both AQC and LNS find feasible solutions that are of higher quality than what the MIP solver can find in 600 seconds. This suggests that both methods are producing high-quality solutions at this scale. As the problem size grows, a slight quality discrepancy emerges favoring LNS over AQC; however, this discrepancy in average solution quality is less than 1% of the best known value.

Detailed Runtime Analysis Given that both the AQC and the LNS solution methods have very similar solution qualities, it is prudent to perform a detailed runtime study to understand the quality vs runtime tradeoff. To develop a runtime profile of the LNS algorithm, the solver's runtime limit is set to values ranging from 0.01 to 10.00 seconds. In the case of the AQC algorithm, the number of requested samples is set to values ranging from 10 to 10,000, which has the effect of scaling the runtime of the IPU process.[4] The results of this study

[4] It is important to emphasize that the IPU runtime does not include any communication or scheduling overheads.

			AQC		LNS		MIP		
# Cases	$\mu(\lvert\mathcal{N}\rvert)$	$\mu(\lvert\mathcal{E}\rvert)$	$\mu(UB)$	$\mu(T)$	$\mu(UB)$	$\mu(T)$	$\mu(UB)$	$\mu(LB)$	$\mu(T)$
RAN-1 Problems of Increasing Size									
250	30	70	-44	3.53	-44	10	-44	-44	0.05
250	69	176	-110	3.57	-110	10	-110	-110	0.48
250	122	321	-199	3.60	-199	10	-199	-199	15.90
250	194	528	-325	3.64	-325	10	-325	-327	340.00
250	275	751	-462	3.68	-462	10	-461	-483	TO
250	375	1030	-633	3.73	-633	10	-629	-673	TO
250	486	1337	-821	3.77	-822	10	-814	-881	TO
250	613	1689	-1038	3.77	-1039	10	-1021	-1116	TO
250	761	2114	-1296	3.76	-1297	10	-1262	-1401	TO
250	923	2578	-1574	3.77	-1576	10	-1525	-1713	TO
250	1095	3061	-1870	3.80	-1873	10	-1806	-2045	TO
RANF-1 Problems of Increasing Size									
250	30	70	-53	3.53	-53	10	-53	-53	0.02
250	69	176	-127	3.56	-127	10	-127	-127	0.13
250	122	321	-229	3.61	-229	10	-229	-229	0.67
250	194	528	-370	3.66	-370	10	-370	-370	14.10
250	275	751	-526	3.71	-526	10	-526	-527	128.00
250	375	1030	-719	3.76	-719	10	-719	-727	471.00
250	486	1337	-934	3.81	-934	10	-933	-954	588.00
250	613	1689	-1179	3.82	-1179	10	-1178	-1211	TO
250	761	2114	-1472	3.82	-1472	10	-1470	-1520	TO
250	923	2578	-1786	3.82	-1787	10	-1778	-1856	TO
250	1095	3061	-2121	3.86	-2122	10	-2110	-2212	TO

Table 4. A Comparison of Solution Quality and Runtime as Problem Size Increases on RAN-1 and RANF-1.

are summarized in Figure 2. Note that the stochastic sampling nature of the IPU results in some noise for small numbers of samples. However, the overall trend is still clear.

The results presented in Figure 2 further illustrate that (1) the RAN problem class is more challenging than the RANF problem class, and (2) regardless of the runtime configuration used, the LNS heuristic slightly outperforms the AQC; however, the average solution quality is always within 1% of each other. Combining all of the results from this section suggests that the D-Wave 2X IPU is comparable to state-of-the-art optimization methods on commodity classical computing hardware.

6 Conclusion

In this work we have introduced the idea of an IPU as a computational abstraction for emerging physical devices that optimize Ising models. We have demonstrated a number of unexpected challenges in benchmarking such devices and have proposed BQPJSON , DWIG , and BQPSOLVERS to help address these

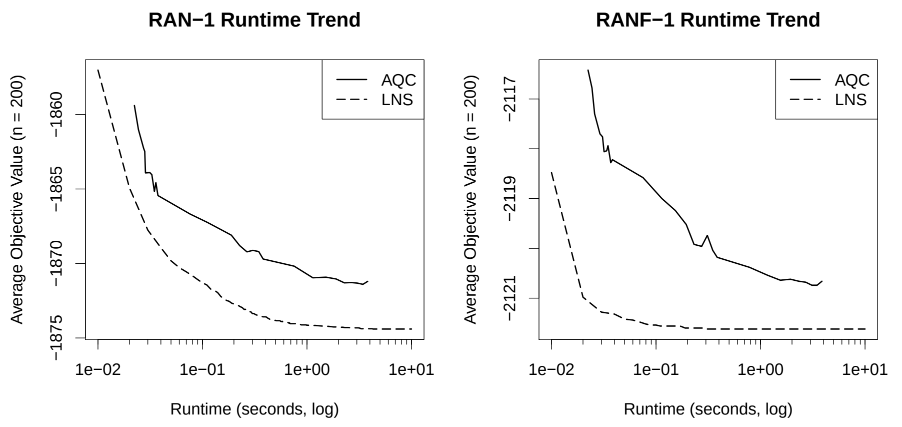

Fig. 2. Detailed Runtime Analysis of the AQC (D-Wave 2X) and LNS Heuristic (HFS) on the RAN-1 (left) and RANF-1 (right) Problem Classes.

challenges. We hope that these tools will assist the research community in developing novel algorithms that leverage IPUs.

The baseline study of the D-Wave 2X IPU suggests that averaging over collections of randomly generated test cases is a reasonable strategy for benchmarking IPUs. However, finding a class of challenging randomly generated test cases is not trivial. The study verified that at least one commercially available IPU is comparable to current state-of-the-art classical methods. However, a detailed runtime analysis did not demonstrate any time vs quality configuration where the IPU outperformed a state-of-the-art classical method. That said, the IPU hardware seems to be largely invariant in quality and runtime across a wide variety of problem classes and instance sizes. If this trend continues as the IPU's hardware increases in size, one would expect the IPU to overtake the current state-of-the-art classical methods because of its parallel computational nature.

Overall, we conclude that the emergence of IPUs is an interesting development for the optimization community and warrants continued study. Considerable work remains to determine the best classes of test cases for benchmarking IPUs and to develop a better understanding of how to fairly compare this heterogeneous hardware to established classical computing methods.

Acknowledgments

We gratefully acknowledge Denny Dahl for his feedback on the tools proposed herein and the U.S. Department of Energy for supporting this work through Los Alamos National Laboratory's LDRD Program.

References

1. Aaronson, S.: D-wave: Truth finally starts to emerge. Published online at http://www.scottaaronson.com/blog/?p=1400 (May 2013), accessed: 04/28/2017
2. Aaronson, S.: Insert d-wave post here. Published online at http://www.scottaaronson.com/blog/?p=3192 (Mar 2017), accessed: 04/28/2017
3. Balyo, T., Heule, M.J.H., Jarvisalo, M.: Sat competition 2016: Recent developments. In: Proceedings of the Thirty-First National Conference on Artificial Intelligence. pp. 5061–5063. AAAI'17, AAAI Press (2017)
4. Benedetti, M., Realpe-Gómez, J., Biswas, R., Perdomo-Ortiz, A.: Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning. Phys. Rev. A 94, 022308 (Aug 2016), https://link.aps.org/doi/10.1103/PhysRevA.94.022308
5. Bian, Z., Chudak, F., Israel, R., Lackey, B., Macready, W.G., Roy, A.: Discrete optimization using quantum annealing on sparse ising models. Frontiers in Physics 2, 56 (2014), http://journal.frontiersin.org/article/10.3389/fphy.2014.00056
6. Bian, Z., Chudak, F., Israel, R.B., Lackey, B., Macready, W.G., Roy, A.: Mapping constrained optimization problems to quantum annealing with application to fault diagnosis. Frontiers in ICT 3, 14 (2016), http://journal.frontiersin.org/article/10.3389/fict.2016.00014
7. Bian, Z., Chudak, F., Macready, W.G., Rose, G.: The ising model: teaching an old problem new tricks. Published online at https://www.dwavesys.com/sites/default/files/weightedmaxsat_v2.pdf (2010), accessed: 04/28/2017
8. Billionnet, A., Elloumi, S.: Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. Mathematical Programming 109(1), 55–68 (2007), http://dx.doi.org/10.1007/s10107-005-0637-9
9. Boixo, S., Ronnow, T.F., Isakov, S.V., Wang, Z., Wecker, D., Lidar, D.A., Martinis, J.M., Troyer, M.: Evidence for quantum annealing with more than one hundred qubits. Nat Phys 10(3), 218–224 (Mar 2014), http://dx.doi.org/10.1038/nphys2900, article
10. Booth, M.: qbsolv. https://github.com/dwavesystems/qbsolv (2017)
11. Boothby, T., King, A.D., Roy, A.: Fast clique minor generation in chimera qubit connectivity graphs. Quantum Information Processing 15(1), 495–508 (Jan 2016), http://dx.doi.org/10.1007/s11128-015-1150-6
12. Brush, S.G.: History of the lenz-ising model. Rev. Modern Phys. 39, 883–893 (Oct 1967), https://link.aps.org/doi/10.1103/RevModPhys.39.883
13. Cai, J., Macready, W.G., Roy, A.: A practical heuristic for finding graph minors (2014), https://arxiv.org/abs/1406.2741
14. Coffrin, C., Nagarajan, H., Bent, R.: Challenges and Successes of Solving Binary Quadratic Programming Benchmarks on the DW2X QPU. Tech. rep., Los Alamos National Laboratory (LANL) (2016)
15. D-Wave Systems Inc.: The d-wave 2x quantum computer technology overview. Published online at https://www.dwavesys.com/sites/default/files/D-Wave%202X%20Tech%20Collateral_0915F.pdf (2015), accessed: 04/28/2017
16. D-Wave Systems Inc.: Customers. Published online at https://www.dwavesys.com/our-company/customers (2017), accessed: 04/28/2017
17. Dash, S.: A note on qubo instances defined on chimera graphs. arXiv preprint arXiv:1306.1202 (2013), https://arxiv.org/abs/1306.1202

18. Denchev, V.S., Boixo, S., Isakov, S.V., Ding, N., Babbush, R., Smelyanskiy, V., Martinis, J., Neven, H.: What is the computational value of finite-range tunneling? Phys. Rev. X 6, 031015 (Aug 2016), `https://link.aps.org/doi/10.1103/PhysRevX.6.031015`
19. Feynman, R.P.: Simulating physics with computers. International Journal of Theoretical Physics 21(6), 467–488 (1982)
20. Gent, I.P., Walsh, T.: CSPlib: A Benchmark Library for Constraints, pp. 480–481. Springer Berlin Heidelberg, Berlin, Heidelberg (1999), `http://dx.doi.org/10.1007/978-3-540-48085-3_36`
21. de Givry, S., Larrosa, J., Meseguer, P., Schiex, T.: Solving Max-SAT as Weighted CSP, pp. 363–376. Springer Berlin Heidelberg, Berlin, Heidelberg (2003), `http://dx.doi.org/10.1007/978-3-540-45193-8_25`
22. Gurobi Optimization, Inc.: Gurobi optimizer reference manual. Published online at `http://www.gurobi.com` (2014)
23. Hamze, F., de Freitas, N.: From fields to trees. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence. pp. 243–250. UAI '04, AUAI Press, Arlington, Virginia, United States (2004), `http://dl.acm.org/citation.cfm?id=1036843.1036873`
24. Haribara, Y., Utsunomiya, S., Yamamoto, Y.: A Coherent Ising Machine for MAX-CUT Problems: Performance Evaluation against Semidefinite Programming and Simulated Annealing, pp. 251–262. Springer Japan, Tokyo (2016), `http://dx.doi.org/10.1007/978-4-431-55756-2_12`
25. Hen, I., Job, J., Albash, T., Rønnow, T.F., Troyer, M., Lidar, D.A.: Probing for quantum speedup in spin-glass problems with planted solutions. Phys. Rev. A 92, 042325 (Oct 2015), `https://link.aps.org/doi/10.1103/PhysRevA.92.042325`
26. Hoos, H.H., Stutzle, T.: Satlib: An online resource for research on sat (2000)
27. Inagaki, T., Haribara, Y., Igarashi, K., Sonobe, T., Tamate, S., Honjo, T., Marandi, A., McMahon, P.L., Umeki, T., Enbutsu, K., Tadanaga, O., Takenouchi, H., Aihara, K., Kawarabayashi, K.i., Inoue, K., Utsunomiya, S., Takesue, H.: A coherent ising machine for 2000-node optimization problems. Science 354(6312), 603–606 (2016), `http://science.sciencemag.org/content/354/6312/603`
28. International Business Machines Corporation: Ibm building first universal quantum computers for business and science. Published online at `https://www-03.ibm.com/press/us/en/pressrelease/51740.wss` (2017), accessed: 04/28/2017
29. Johnson, M.W., Amin, M.H.S., Gildert, S., Lanting, T., Hamze, F., Dickson, N., Harris, R., Berkley, A.J., Johansson, J., Bunyk, P., Chapple, E.M., Enderud, C., Hilton, J.P., Karimi, K., Ladizinsky, E., Ladizinsky, N., Oh, T., Perminov, I., Rich, C., Thom, M.C., Tolkacheva, E., Truncik, C.J.S., Uchaikin, S., Wang, J., Wilson, B., Rose, G.: Quantum annealing with manufactured spins. Nature 473(7346), 194–198 (May 2011), `http://dx.doi.org/10.1038/nature10012`
30. Kielpinski, D., Bose, R., Pelc, J., Vaerenbergh, T.V., Mendoza, G., Tezak, N., Beausoleil, R.G.: Information processing with large-scale optical integrated circuits. In: 2016 IEEE International Conference on Rebooting Computing (ICRC). pp. 1–4 (Oct 2016)
31. King, A.D., Lanting, T., Harris, R.: Performance of a quantum annealer on range-limited constraint satisfaction problems. arXiv preprint arXiv:1502.02098 (2015)
32. King, J., Yarkoni, S., Nevisi, M.M., Hilton, J.P., McGeoch, C.C.: Benchmarking a quantum annealing processor with the time-to-target metric. arXiv preprint arXiv:1508.05087 (2015)

33. King, J., Yarkoni, S., Raymond, J., Ozfidan, I., King, A.D., Nevisi, M.M., Hilton, J.P., McGeoch, C.C.: Quantum annealing amid local ruggedness and global frustration (2017), `https://arxiv.org/abs/1701.04579`
34. Klymko, C., Sullivan, B.D., Humble, T.S.: Adiabatic quantum programming: minor embedding with hard faults. Quantum Information Processing 13(3), 709–729 (2014), `http://dx.doi.org/10.1007/s11128-013-0683-9`
35. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R., Danna, E., Gamrath, G., Gleixner, A., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D., Wolter, K.: Miplib 2010: Mixed integer programming library version 5. Mathematical Programming Computation 3(2), 103–163 (2011)
36. Lokhov, A.Y., Vuffray, M., Misra, S., Chertkov, M.: Optimal structure and parameter learning of ising models (2016), `https://arxiv.org/abs/1612.05024`
37. Lucas, A.: Ising formulations of many np problems. Frontiers in Physics 2, 5 (2014), `http://journal.frontiersin.org/article/10.3389/fphy.2014.00005`
38. Mandrà, S., Zhu, Z., Wang, W., Perdomo-Ortiz, A., Katzgraber, H.G.: Strengths and weaknesses of weak-strong cluster problems: A detailed overview of state-of-the-art classical heuristics versus quantum approaches. Phys. Rev. A 94, 022337 (Aug 2016), `https://link.aps.org/doi/10.1103/PhysRevA.94.022337`
39. Mandr, S., Katzgraber, H.G., Thomas, C.: The pitfalls of planar spin-glass benchmarks: Raising the bar for quantum annealers (again) (2017), `https://arxiv.org/abs/1703.00622`
40. McMahon, P.L., Marandi, A., Haribara, Y., Hamerly, R., Langrock, C., Tamate, S., Inagaki, T., Takesue, H., Utsunomiya, S., Aihara, K., et al.: A fully-programmable 100-spin coherent ising machine with all-to-all connections. Science p. aah5178 (2016)
41. Mitchell, D., Selman, B., Levesque, H.: Hard and easy distributions of sat problems. In: Proceedings of the Tenth National Conference on Artificial Intelligence. pp. 459–465. AAAI'92, AAAI Press (1992), `http://dl.acm.org/citation.cfm?id=1867135.1867206`
42. Modha, D.S.: Introducing a brain-inspired computer. Published online at `http://www.research.ibm.com/articles/brain-chip.shtml` (2017), accessed: 04/28/2017
43. Mohseni, M., Read, P., Neven, H., Boixo, S., Denchev, V., Babbush, R., Fowler, A., Smelyanskiy, V., Martinis, J.: Commercialize quantum technologies in five years. Nature 543, 171174 (2017), `http://www.nature.com/news/commercialize-quantum-technologies-in-five-years-1.21583`
44. Morgado, A., Heras, F., Liffiton, M., Planes, J., Marques-Silva, J.: Iterative and core-guided maxsat solving: A survey and assessment. Constraints 18(4), 478–534 (2013), `http://dx.doi.org/10.1007/s10601-013-9146-2`
45. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: MiniZinc: Towards a Standard CP Modelling Language, pp. 529–543. Springer Berlin Heidelberg, Berlin, Heidelberg (2007), `http://dx.doi.org/10.1007/978-3-540-74970-7_38`
46. Nieuwenhuis, R.: The IntSat Method for Integer Linear Programming, pp. 574–589. Springer International Publishing, Cham (2014), `http://dx.doi.org/10.1007/978-3-319-10428-7_42`
47. Rieffel, E.G., Venturelli, D., O'Gorman, B., Do, M.B., Prystay, E.M., Smelyanskiy, V.N.: A case study in programming a quantum annealer for hard operational planning problems. Quantum Information Processing 14(1), 1–36 (2015), `http://dx.doi.org/10.1007/s11128-014-0892-x`

48. Selby, A.: Qubo-chimera. https://github.com/alex1770/QUBO-Chimera (2013)
49. Selby, A.: Efficient subgraph-based sampling of ising-type models with frustration (2014), https://arxiv.org/abs/1409.3934
50. Venturelli, D., Marchand, D.J.J., Rojo, G.: Quantum annealing implementation of job-shop scheduling (2015), https://arxiv.org/abs/1506.08479
51. Vuffray, M., Misra, S., Lokhov, A., Chertkov, M.: Interaction screening: Efficient and sample-optimal learning of ising models. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 2595–2603. Curran Associates, Inc. (2016)
52. Yamaoka, M., Yoshimura, C., Hayashi, M., Okuyama, T., Aoki, H., Mizuno, H.: 24.3 20k-spin ising chip for combinational optimization problem with cmos annealing. In: 2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers. pp. 1–3 (Feb 2015)
53. Yoshimura, C., Yamaoka, M., Aoki, H., Mizuno, H.: Spatial computing architecture using randomness of memory cell stability under voltage control. In: 2013 European Conference on Circuit Theory and Design (ECCTD). pp. 1–4 (Sept 2013)
54. Zdeborova, L., Krzakala, F.: Statistical physics of inference: thresholds and algorithms. Advances in Physics 65(5), 453–552 (2016), http://dx.doi.org/10.1080/00018732.2016.1211393

LA-UR-17-23494