Evaluating Ising Processing Units with Integer Programming

Carleton Coffrin(✉), Harsha Nagarajan, and Russell Bent

Los Alamos National Laboratory, Los Alamos, NM, USA
cjc@lanl.gov

Abstract. The recent emergence of novel computational devices, such as adiabatic quantum computers, CMOS annealers, and optical parametric oscillators, present new opportunities for hybrid-optimization algorithms that are hardware accelerated by these devices. In this work, we propose the idea of an Ising processing unit as a computational abstraction for reasoning about these emerging devices. The challenges involved in using and benchmarking these devices are presented and commercial mixed integer programming solvers are proposed as a valuable tool for the validation of these disparate hardware platforms. The proposed validation methodology is demonstrated on a D-Wave 2X adiabatic quantum computer, one example of an Ising processing unit. The computational results demonstrate that the D-Wave hardware consistently produces high-quality solutions and suggests that as IPU technology matures it could become a valuable co-processor in hybrid-optimization algorithms.

Keywords: Discrete optimization · Ising model · Quadratic unconstrained binary optimization · Integer programming · Large Neighborhood Search · Adiabatic quantum computation

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) [1], gate-based quantum computers [2–4], CMOS annealers [5–7], neuromorphic computers [8–10], memristive circuits [11,12], and optical parametric oscillators [13–15]. 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) [16]. Despite their entirely disparate physical implementations, AQCs, CMOS annealers, memristive circuits, 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

L.-M. Rousseau and K. Stergiou (Eds.): CPAIOR 2019, LNCS 11494, pp. 163–181, 2019.
https://doi.org/10.1007/978-3-030-19212-9_11

processes [17]. Furthermore, this kind of "Ising machine" [13,14] is already commercially available with more than 2000 decision variables in the form of AQCs developed by D-Wave Systems [18].

The emergence of physical devices that can quickly solve Ising models is particularly relevant to the constraint programming, artificial intelligence and operations research communities, because the impetus for building these devices is to perform discrete optimization. As this technology matures, it may be possible for this specialized hardware to rapidly solve challenging combinatorial problems, such as Max-Cut [19] or Max-Clique [20]. Preliminary studies have suggested that some classes of Constraint Satisfaction Problems may be effectively encoded in such devices because of their combinatorial structure [21–24]. Furthermore, an Ising model coprocessor could have significant impacts on solution methods for a variety of fundamental combinatorial problem classes, such as MAX-SAT [25–27] and integer programming [28]. At this time, however, it remains unclear how established optimization algorithms should leverage this emerging technology. This paper helps to address this gap by highlighting the key concepts and hardware limitations that an algorithm designer needs to understand to engage in this emerging and exciting computational paradigm.

Similar to an arithmetic logic unit (ALU) or a graphics processing unit (GPU), this work proposes the idea of an Ising processing unit (IPU) as the computational abstraction for wide variety of physical devices that perform optimization of Ising models. This work begins with a brief introduction to the IPU abstraction and its mathematical foundations in Sect. 2. Then the additional challenges of working with real-world hardware are discussed in Sect. 3 and an overview of previous benchmarking studies and solution methods are presented in Sect. 4. Finally, a detailed benchmarking study of a D-Wave 2X IPU is conducted in Sect. 5, which highlights the current capabilities of such a device. The contributions of this work are as follows,

1. The first clear and concise introduction to the key concepts of Ising models and the limitations of real-world IPU hardware, both of which are necessary for optimization algorithm designers to effectively leverage these emerging hardware platforms (Sects. 2 and 3).
2. Highlighting that integer programming has been overlooked by recent IPU benchmarking studies (Sect. 4), and demonstrating the value of integer programming for filtering easy test cases (Sect. 5.1) and verifying the quality of an IPU on challenging test cases (Sect. 5.2).

Note that, due to the maturity and commercial availability of the D-Wave IPU, this work often refers to that architecture as an illustrative example. However, the methods and tools proposed herein are applicable to all emerging IPU hardware realizations, to the best of our knowledge.

2 A Brief Introduction to Ising Models

This section introduces the notations of the paper and provides a brief introduction to Ising models, 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* $\boldsymbol{h}_i\ \forall i \in \mathcal{N}$ is specified for each node, and an interaction strength $\boldsymbol{J}_{ij}\ \forall i, j \in \mathcal{E}$ is specified for each edge. Given these data, the *energy* of the Ising model is defined as,

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

Applications of the Ising model typically consider one of two tasks. First, some applications focus on finding the lowest possible energy of the Ising model, known as a *ground state*. That is, finding the globally optimal solution of the following binary quadratic optimization problem:

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

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

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

where $\boldsymbol{\tau}$ is a parameter representing the *effective temperature* of the Boltzmann distribution [29]. It is valuable to observe that in the Boltzmann distribution, the lowest energy states have the highest probability. Therefore, 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:

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

Observe that, in this case, there are a number of optimal solutions such that $E(\boldsymbol{\sigma}^*) = -2$ but none such that $E(\sigma) = \sum_{i,j \in \mathcal{E}} -|\boldsymbol{J}_{ij}| = -3$. Note that frustration has important algorithmic implications as greedy algorithms are sufficient for optimizing Ising models without frustration.

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 σ-to-x is given by,

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

$$\sigma_i \sigma_j = 4x_i x_j - 2x_i - 2x_j + 1 \ \forall i,j \in \mathcal{E} \tag{8b}$$

and the inverse x-to-σ is given by,

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

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

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 that IPUs perform. However, in practice, a number of hardware implementation factors present additional challenges for computing with IPUs.

3 Features of Analog 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 [1,6,11,13]. 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.

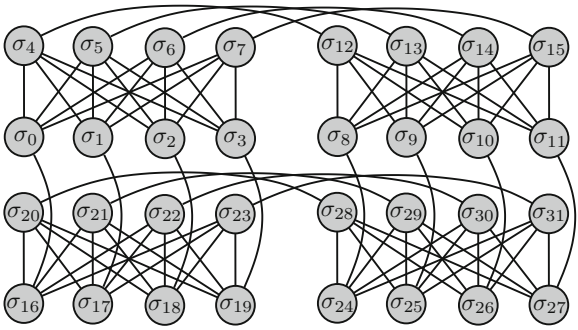

Fig. 1. A 2-by-2 chimera graph illustrating the variable product limitations of a D-Wave 2X IPU.

Effective Temperature: The ultimate goal of 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, $\boldsymbol{\tau}$ [30]. 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 [31]. This suggests that the IPU's performance can change based on the structure of the problem input.

Environmental Noise: One of the primary contributors to the sampling nature of IPUs are the 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.

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 of bias is often mitigated by performing multiple gauge transformations on the input and combining the results from all executions.

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, [1,6] whereas others support only the discrete set of $\{-1, 0, 1\}$ [13]. In either case, the mathematical Ising model must be rescaled into the IPU's operating range. However, this mathematically equivalent transformation can result in unexpected side effects because the coefficients used in the IPU hardware are perturbed by a constant amount of environmental noise and hardware bias, which can outweigh small rescaled coefficient values.

Topological Limitations: Another significant feature of IPUs is a restricted set of 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 Fig. 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 [32–34]. Research and development of algorithms for embedding various kinds of Ising models into a specific IPU topology is still an active area of research [21,35–37].

3.1 Challenges of Benchmarking Ising Processing Units

These analog hardware features present unique challenges for benchmarking IPUs that fall roughly into three categories: (1) comparing to established benchmark libraries; (2) developing Ising model instance generators for testing and; (3) comparing with classical optimization methods.

Benchmark Libraries: Research and development in optimization algorithms has benefited greatly from standardized benchmark libraries [38–40]. However, direct application of these libraries to IPUs is out of scope in 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 given problems onto the IPU's hardware graph can be prohibitive [41]; 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.

Instance Generation Algorithms: Due to these challenges, the standard practice in the literature is to generate a collection of instances for a given IPU and use these cases for the evaluation of that IPU [33,34,42,43]. The hope being that these instances provide a reasonable proxy for how real-world applications might perform on such a device.

Comparison with Classical Algorithms: Because of the radically different hardware of CPUs vs IPUs and the stochastic nature of the IPUs, conducting a fair comparison of these two technologies is not immediately clear [43–45]. Indeed, comparisons of D-Wave's IPU with classical algorithms have resulted in vigorous discussions about what algorithms and metrics should be used to make such comparisons [34,46,47]. It is widely accepted that IPUs do not provide optimality guarantees and are best compared to heuristic methods (e.g. local search) in terms of runtime performance. This debate will most likely continue for several years. In this work, our goal is not to answer these challenging questions but rather to highlight that commercial mixed integer programming solvers are valuable and important tools for exploring these questions.

4 A Review of Ising Processing Unit Benchmarking Studies

Due to the challenges associated with mapping established optimization test cases to specific IPU hardware [41], the IPU benchmarking community has adopted the practice of generating Ising model instances on a case-by-case basis for specific IPUs [33,34,42,43] and evaluating these instances on a variety of solution methods. The following subsections provide a brief overview of the instance generation algorithms and solution methods that have been used in various IPU benchmarking studies. The goals of this review are to: (1) reveal the lack of consistency across current benchmarking studies; (2) highlight the omission of integer programming methods in all of the recent publications and; (3) motivate the numerical study conducted in this work.

4.1 Instance Generation Algorithms

The task of IPU instance generation amounts to finding interesting values for $\mathbf{h}$ and $\mathbf{J}$ in (1). In some cases the procedures for generating these values are elaborate [33,48] and are designed to leverage theoretical results about Ising models [42]. A brief survey reveals five primary problem classes in the literature, each of which is briefly introduced. 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, this general class of problem was first proposed in [27] and was later refined into the RAN-k

problem in [34]. The RAN-k problem consists simply of assigning each value of $\mathbf{h}$ to 0 and each value of $\mathbf{J}$ uniformly at random from the set

$$\{-\boldsymbol{k}, -\boldsymbol{k}+1, \ldots, -2, -1, 1, 2, \ldots, \boldsymbol{k}-1, \boldsymbol{k}\} \tag{11}$$

The RANF-k problem is a simple variant of RAN-k where the values of $\mathbf{h}$ are also selected uniformly at random from (11). 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 [42] and then later refined to the FL-k problem in [48]. 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 two globally optimal solutions are maintained at $\sigma_i = -1 \; \forall i \in \mathcal{N}$ and $\sigma_i = 1 \; \forall i \in \mathcal{N}$ [48]. However, to obfuscate this solution, a gauge transformation is often applied to make the optimal solution a random assignment of σ.

A variant of the frustrated loop problem is the frustrated *cluster* loop problem, FCL-k [43]. The FCL-k problem is inspired by the chimera network topology (i.e., Fig. 1). The core idea is that tightly coupled variables (e.g., $\sigma_0 ... \sigma_7$ in Fig. 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. Note that a polynomial time algorithm is known for solving the FCL-k problem class on chimera graphs [45].

It is worthwhile to mention that the FL-k and FCL-k instance generators are solving a cycle packing problem on the IPU graph. Hence, the randomized algorithms proposed in [42,43] are not guaranteed to find a solution if one exists. In practice, this algorithm fails for the highly constrained settings of α and k.

Weak-Strong Cluster Networks (WSCNs): The WSCN problem was proposed in [33] 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., $\sigma_0 ... \sigma_7$ and $\sigma_8 ... \sigma_{15}$ in Fig. 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 [33]. The motivation for the WSCN design is that the clusters create deep local minima that are difficult for local search methods to escape.

4.2 Solution Methods

Once a collection of Ising model instances have been generated, the next step in a typical benchmarking study is to evaluate those instances on a variety of

solution methods, including the IPU, and compare the results. A brief survey reveals five primary solution methods in the literature, each of which is briefly introduced. For a detailed description, please refer to the source publications of the solution method.

Simulated Annealing: The most popular staw-man solution method for comparison is Simulated Annealing [49]. Typically the implementation only considers a neighborhood of single variable flips and the focus of these implementations is on computational performance (e.g. using GPUs for acceleration). The search is run until a specified time limit is reached.

Large Neighborhood Search: The state-of-the-art meta-heuristic for solving Ising models on the chimera graphs is a Large Neighborhood Search (LNS) method called the Hamze-Freitas-Selby (HFS) algorithm [50,51]. 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. A key to this method's success is the availability of a highly optimized open-source C implementation [52].

Integer Programming: Previous works first considered integer quadratic programming [27] and quickly moved to integer linear programming [53,54] as a solution method. The mathematical programming survey [55] provides a useful overview of the advantages and dis-advantages of various integer programming (IP) formulations.

Based on some preliminary experiments with different formulations, this work focuses on the following integer linear programming formulation of the Ising model, transformed into the Boolean variable space:

$$\min : \sum_{i,j\in\mathcal{E}} \boldsymbol{c}_{ij} x_{ij} + \sum_{i\in\mathcal{N}} \boldsymbol{c}_i x_i + \boldsymbol{c} \tag{12a}$$

s.t.:

$$\begin{aligned} &x_{ij} \geq x_i + x_j - 1,\ x_{ij} \leq x_i,\ x_{ij} \leq x_j\ \forall i,j \in \mathcal{E} \\ &x_i \in \{0,1\}\ \forall i \in \mathcal{N},\ x_{ij} \in \{0,1\}\ \forall i,j \in \mathcal{E} \end{aligned} \tag{12b}$$

where the application of (8) leads to,

$$\boldsymbol{c}_{ij} = \sum_{i,j\in\mathcal{E}} 4\boldsymbol{J}_{ij}\ \forall i,j \in \mathcal{E} \tag{13a}$$

$$\boldsymbol{c}_i = \sum_{i,j\in\mathcal{E}(i)} 2\boldsymbol{J}_{ij} + \sum_{i\in\mathcal{N}} 2\boldsymbol{h}_i\ \forall i \in \mathcal{N} \tag{13b}$$

$$\boldsymbol{c} = \sum_{i,j\in\mathcal{E}} \boldsymbol{J}_{ij} - \sum_{i\in\mathcal{N}} \boldsymbol{h}_i \tag{13c}$$

In this formulation, the binary quadratic program defined in (10) is converted 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 $x_{ij} = x_i \wedge x_j \ \forall i, j \in \mathcal{E}$ conjunction constraints. Preliminary experiments of this work confirmed the findings of [55], that this binary linear program formulation is best on sparse graphs, such as the hardware graphs of current IPUs.

Table 1. A chronological summary of IPU benchmarking studies

Publication	Problem classes					Solution methods				
	RAN	RANF	FL	FCL	WSCN	IP	SA	LNS	QMC	AQC
[27]	✓					✓				
[53]	✓					✓				
[54]		✓				✓				
[42]			✓				✓	✓		✓
[48]			✓				✓	✓		✓
[60]	✓		✓					✓	✓	✓
[33]					✓		✓		✓	✓
[43]				✓			✓	✓	✓	✓
This work	✓	✓	✓	✓	✓	✓		✓		✓

Adiabatic Quantum Computation: An adiabatic quantum computation (AQC) [56] is a method for solving an Ising model via a quantum annealing process [57]. This solution method has two notable traits: (1) the AQC dynamical process features quantum tunneling [58], which can help it to escape from local minima; (2) it can be implemented in hardware (e.g. the D-Wave IPU).

Quantum Monte Carlo: Quantum Monte Carlo (QMC) is a probabilistic algorithm that can be used for simulating large quantum systems. QMC is a very computationally intensive method [33,59] and thus the primary use of QMC is not to compare runtime performance but rather to quantify the possible value of an adiabatic quantum computation that could be implemented in hardware at some point in the future.

4.3 Overview

To briefly summarize a variety of benchmarking studies, Table 1 provides an overview of the problems and solution methods previous works have considered. Although there was some initial interest in integer programming models [27, 53,54], more recent IPU benchmark studies have not considered these solution methods and have focused exclusively on heuristic methods. Furthermore, there are notable inconsistencies in the type of problems being considered. As indicated by the last row in Table 1, the goal of this work is revisit the use of IP methods for benchmarking IPUs and to conduct a thorough and side-by-side study of all problem classes and solution methods proposed in the literature. Note that,

because this paper focuses exclusively on the quality and runtime of the Ising model optimization task (2), the study of SA and QMC are omitted as they provide no additional insights over the LNS [48] and AQC [33] methods.

5 A Study of Established Methods

This section conducts an in-depth computational study of the established instance generation algorithms and solution methods for IPUs. The first goal of this study is to understand what classes of problems and parameters are the most challenging, as such cases are preferable for benchmarking. The second goal is to conduct a validation study of a D-Wave 2X IPU, to clearly quantify its solution quality and runtime performance. This computational study is divided into two phases. First, a broad parameter sweep of all possible instance generation algorithms is conducted and a commercial mixed-integer programming solver is used to filter out the easy problem classes and parameter settings. Second, after the most challenging problems have been identified, a detailed study is conducted to compare and contrast the three disparate solution methods IP, LNS, and AQC.

Throughout this section, the following notations are used to describe the algorithm results: 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[1], TO denotes that the algorithm hit a time limit of 600 s, $\mu(\cdot)$ 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 classical computing algorithms are run on HPE ProLiant XL170r servers with dual Intel 2.10 GHz CPUs and 128 GB memory. After a preliminary comparison of CPLEX 12.7 [61] and Gurobi 7.0 [62], no significant difference was observed. Thus, Gurobi was selected as the commercial Mixed-Integer Programming (MIP) solver and was configured to use one thread. The highly specialized and optimized HFS algorithm [52] is used as an LNS-based heuristic and also uses one thread.

The IPU computation is conducted on a D-Wave 2X [63] adiabatic quantum computer (AQC). This computer has a 12-by-12 chimera cell topology with random omissions; in total, it has 1095 spins and 3061 couplers and an effective temperature of $\boldsymbol{\tau} \in (0.091, 0.053)$ depending on the problem being solved [64,65]. Unless otherwise noted, the AQC is configured to produce 10,000 samples using a 5-μs annealing time per sample and a random gauge transformation every 100 samples. The best sample is used in the computation of the upper bound value. 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, which adds an overhead of about three seconds.

[1] For MIP solvers, the runtime includes the computation of the optimally certificate.

Table 2. Parameter settings of various problems.

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 3. MIP runtime on various IPU benchmark problems (seconds)

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

All of the software used in this benchmarking study is available as open-source via: BQPJSON, a language-independent JSON-based Ising model exchange format designed for benchmarking IPU hardware; DWIG, algorithms for IPU instance generation; BQPSOLVERS, tools for encoding BQPJSON data into various optimization formulations and solvers.[2]

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 Sect. 4.1. Table 2 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 is generated using all of the decision variables available on the IPU. The computational results of this parameter sweep are summarized in Table 3.

The results presented in Table 3 indicate that, at this problem size, all variants of the FL, FCL, and WSCN problems are easy for modern MIP solvers. This is a stark contrast to [33], which reported runtimes around 10,000 s when applying Simulated Annealing to the WSCN problem. Furthermore, this result

[2] The source code is available at https://github.com/lanl-ansi/ under the repository names BQPJSON, DWIG and BQPSOLVERS.

Table 4. MIP runtime on RAN-k and RANF-k IPU benchmark problems (seconds)

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

suggests that these problems classes 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 useful for benchmarking. This result is consistent with a similar observation in the SAT community, where random SAT problems are known to be especially challenging [66,67]. 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 s time limit. The results of this parameter sweep are summarized in Table 4.

The results presented in Table 4 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.

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 interesting for IPU benchmarking, 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 a D-Wave 2X 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 5. 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.

Table 5. A comparison of solution quality and runtime as problem size increases on RAN-1 and RANF-1.

			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

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 s. 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 s. 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. The results of this study are summarized in Fig. 2. Note that the stochastic sampling nature of the IPU results in some noise for small numbers of samples. However, the overall trend is clear.

The results presented in Fig. 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 provides a strong validation that even if the D-Wave 2X IPU cannot guarantee a globally optimal solution, it produces high quality solutions reliably across a wide range of inputs.

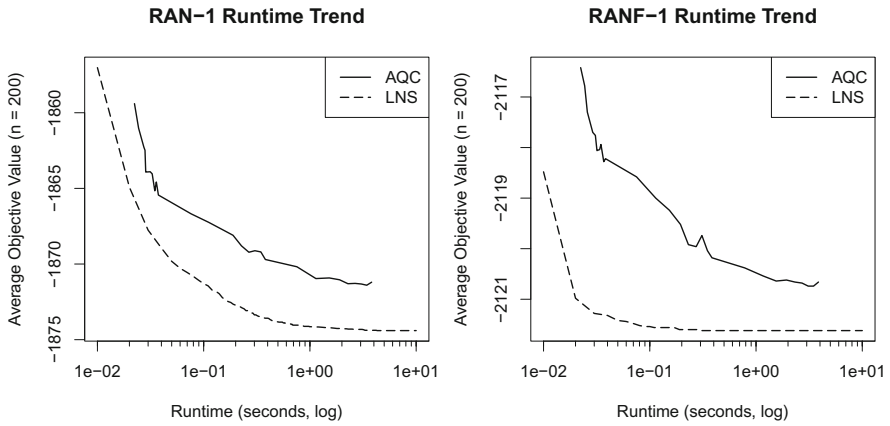

Fig. 2. Detailed runtime analysis of the AQC (D-Wave 2X) and LNS heuristic (HFS) on the RAN-1 (left) and RANF-1 (right) problem classes.

6 Conclusion

This work introduces the idea of Ising processing units (IPUs) as a computational abstraction for emerging physical devices that optimize Ising models.

It highlights a number of unexpected challenges in using such devices and proposes commercial mixed-integer programming solvers as a tool to help improve validation and benchmarking.

A baseline study of the D-Wave 2X IPU suggests that the hardware specific instance generation is a reasonable strategy for benchmarking IPUs. However, finding a class of challenging randomly generated test cases is non-trivial and an open problem for future work. The study verified that at least one commercially available IPU is already comparable to current state-of-the-art classical methods on some classes of problems (e.g. RAN and RANF). Consequently, as this IPU's hardware increases in size, one would expect that it could outperform state-of-the-art classical methods because of its parallel computational nature and become a valuable co-processor in hybrid-optimization algorithms.

Overall, we find that the emergence of IPUs is an interesting development for the optimization community and warrants continued study. Considerable work remains to determine new challenging classes of test cases for validating and benchmarking IPUs. We hope that the technology overview and the validation study conducted in this work will assist the optimization research community in exploring IPU hardware platforms and will accelerate the development of hybrid-algorithms that can effectively leverage these emerging technologies.

References

1. Johnson, M.W., et al.: Quantum annealing with manufactured spins. Nature **473**(7346), 194–198 (2011)
2. International Business Machines Corporation: IBM building first universal quantum computers for business and science (2017). https://www-03.ibm.com/press/us/en/pressrelease/51740.wss. Accessed 28 Apr 2017
3. Mohseni, M., et al.: Commercialize quantum technologies in five years. Nature **543**, 171–174 (2017)
4. Chmielewski, M., et al.: Cloud-based trapped-ion quantum computing. In: APS Meeting Abstracts (2018)
5. 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, February 2015
6. 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, September 2013
7. Fujitsu: Digital annealer, May 2018. http://www.fujitsu.com/global/digitalannealer/. Accessed 26 Feb 2019
8. Modha, D.S.: Introducing a brain-inspired computer (2017). http://www.research.ibm.com/articles/brain-chip.shtml. Accessed 28 Apr 2017
9. Davies, M., et al.: Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro **38**(1), 82–99 (2018)
10. Schuman, C.D., et al.: A survey of neuromorphic computing and neural networks in hardware (2017). arXiv preprint: arXiv:1705.06963
11. Caravelli, F.: Asymptotic behavior of memristive circuits and combinatorial optimization (2017)

12. Traversa, F.L., Di Ventra, M.: MemComputing integer linear programming (2018)
13. McMahon, P.L., et al.: A fully-programmable 100-spin coherent Ising machine with all-to-all connections. Science **354**, 614–617 (2016). https://doi.org/10.1126/science.aah5178
14. Inagaki, T., et al.: A coherent Ising machine for 2000-node optimization problems. Science **354**(6312), 603–606 (2016)
15. Kielpinski, D., et al.: Information processing with large-scale optical integrated circuits. In: 2016 IEEE International Conference on Rebooting Computing (ICRC), pp. 1–4, October 2016
16. Feynman, R.P.: Simulating physics with computers. Int. J. Theor. Phys. **21**(6), 467–488 (1982)
17. Brush, S.G.: History of the lenz-ising model. Rev. Mod. Phys. **39**, 883–893 (1967)
18. D-Wave Systems Inc.: Customers (2017). https://www.dwavesys.com/our-company/customers. Accessed 28 Apr 2017
19. Haribara, Y., Utsunomiya, S., Yamamoto, Y.: A coherent Ising machine for MAX-CUT problems: performance evaluation against semidefinite programming and simulated annealing. In: Yamamoto, Y., Semba, K. (eds.) Principles and Methods of Quantum Information Technologies. LNP, vol. 911, pp. 251–262. Springer, Tokyo (2016). https://doi.org/10.1007/978-4-431-55756-2_12
20. Lucas, A.: Ising formulations of many NP problems. Frontiers Phys. **2**, 5 (2014)
21. 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 ICT **3**, 14 (2016)
22. Bian, Z., Chudak, F., Israel, R., Lackey, B., Macready, W.G., Roy, A.: Discrete optimization using quantum annealing on sparse Ising models. Frontiers Phys. **2**, 56 (2014)
23. 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 Inf. Process. **14**(1), 1–36 (2015)
24. Venturelli, D., Marchand, D.J.J., Rojo, G.: Quantum annealing implementation of job-shop scheduling (2015)
25. de Givry, S., Larrosa, J., Meseguer, P., Schiex, T.: Solving Max-SAT as weighted CSP. In: Rossi, F. (ed.) CP 2003. LNCS, vol. 2833, pp. 363–376. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45193-8_25
26. 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)
27. McGeoch, C.C., Wang, C.: Experimental evaluation of an adiabiatic quantum system for combinatorial optimization. In: Proceedings of the ACM International Conference on Computing Frontiers, CF 2013, pp. 23:1–23:11. ACM (2013)
28. Nieuwenhuis, R.: The IntSat method for integer linear programming. In: O'Sullivan, B. (ed.) CP 2014. LNCS, vol. 8656, pp. 574–589. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10428-7_42
29. Zdeborova, L., Krzakala, F.: Statistical physics of inference: thresholds and algorithms. Adv. Phys. **65**(5), 453–552 (2016)
30. Bian, Z., Chudak, F., Macready, W.G., Rose, G.: The Ising model: teaching an old problem new tricks (2010). https://www.dwavesys.com/sites/default/files/weightedmaxsat_v2.pdf. Accessed 28 Apr 2017
31. 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 (2016)

32. Boixo, S., et al.: Evidence for quantum annealing with more than one hundred qubits. Nat. Phys. **10**(3), 218–224 (2014)
33. Denchev, V.S., et al.: What is the computational value of finite-range tunneling? Phys. Rev. X **6**, 031015 (2016)
34. King, J., Yarkoni, S., Nevisi, M.M., Hilton, J.P., McGeoch, C.C.: Benchmarking a quantum annealing processor with the time-to-target metric (2015). arXiv preprint: arXiv:1508.05087
35. Boothby, T., King, A.D., Roy, A.: Fast clique minor generation in chimera qubit connectivity graphs. Quantum Inf. Process. **15**(1), 495–508 (2016)
36. Cai, J., Macready, W.G., Roy, A.: A practical heuristic for finding graph minors (2014)
37. Klymko, C., Sullivan, B.D., Humble, T.S.: Adiabatic quantum programming: minor embedding with hard faults. Quantum Inf. Process. **13**(3), 709–729 (2014)
38. Koch, T., et al.: MIPLIB 2010: mixed integer programming library version 5. Math. Program. Comput. **3**(2), 103–163 (2011)
39. Gent, I.P., Walsh, T.: CSPLib: a benchmark library for constraints. In: Jaffar, J. (ed.) CP 1999. LNCS, vol. 1713, pp. 480–481. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-540-48085-3_36
40. Hoos, H.H., Stutzle, T.: SATLIB: An online resource for research on SAT (2000)
41. Coffrin, C., Nagarajan, H., Bent, R.: Challenges and successes of solving binary quadratic programming benchmarks on the DW2X QPU. Technical report, Los Alamos National Laboratory (LANL) (2016)
42. 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 (2015)
43. King, J., et al.: Quantum annealing amid local ruggedness and global frustration (2017)
44. 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 (2016)
45. Mandrà, S., Katzgraber, H.G., Thomas, C.: The pitfalls of planar spin-glass benchmarks: raising the bar for quantum annealers (again) (2017)
46. Aaronson, S.: D-wave: Truth finally starts to emerge, May 2013. http://www.scottaaronson.com/blog/?p=1400. Accessed 28 Apr 2017
47. Aaronson, S.: Insert d-wave post here, March 2017. http://www.scottaaronson.com/blog/?p=3192. Accessed 28 Apr 2017
48. King, A.D., Lanting, T., Harris, R.: Performance of a quantum annealer on range-limited constraint satisfaction problems (2015). arXiv preprint: arXiv:1502.02098
49. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
50. Hamze, F., de Freitas, N.: From fields to trees. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, UAI 2004, Arlington, Virginia, United States, pp. 243–250. AUAI Press (2004)
51. Selby, A.: Efficient subgraph-based sampling of Ising-type models with frustration (2014)
52. Selby, A.: Qubo-chimera (2013). https://github.com/alex1770/QUBO-Chimera
53. Puget, J.F.: D-wave vs cplex comparison. Part 2: Qubo (2013). https://www.ibm.com/developerworks/community/blogs/jfp/entry/d_wave_vs_cplex_comparison_part_2_qubo. Accessed 28 Nov 2018

54. Dash, S.: A note on qubo instances defined on chimera graphs (2013). arXiv preprint: arXiv:1306.1202
55. Billionnet, A., Elloumi, S.: Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. Math. Program. **109**(1), 55–68 (2007)
56. Farhi, E., Goldstone, J., Gutmann, S., Sipser, M.: Quantum computation by adiabatic evolution (2018)
57. Kadowaki, T., Nishimori, H.: Quantum annealing in the transverse Ising model. Phys. Rev. E **58**, 5355–5363 (1998)
58. Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., Preda, D.: A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. Science **292**(5516), 472–475 (2001)
59. Nightingale, M.P., Umrigar, C.J. (eds.): Quantum Monte Carlo Methods in Physics and Chemistry. Nato Science Series C, vol. 525. Springer, Netherlands (1998)
60. Parekh, O., Wendt, J., Shulenburger, L., Landahl, A., Moussa, J., Aidun, J.: Benchmarking adiabatic quantum optimization for complex network analysis (2015)
61. IBM ILOG CPLEX Optimizer. https://www.ibm.com/analytics/cplex-optimizer. Accessed 2010
62. Gurobi Optimization, Inc.: Gurobi optimizer reference manual (2014). http://www.gurobi.com
63. D-Wave Systems Inc.: The D-wave 2X quantum computer technology overview (2015). https://www.dwavesys.com/sites/default/files/D-Wave%202X%20Tech%20Collateral_0915F.pdf. Accessed 28 Apr 2017
64. 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, vol. 29, pp. 2595–2603. Curran Associates, Inc. (2016)
65. Lokhov, A.Y., Vuffray, M., Misra, S., Chertkov, M.: Optimal structure and parameter learning of Ising models (2016)
66. Mitchell, D., Selman, B., Levesque, H.: Hard and easy distributions of sat problems. In: Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI 1992, pp. 459–465. AAAI Press (1992)
67. Balyo, T., Heule, M.J.H., Jarvisalo, M.: Sat competition 2016: recent developments. In: Proceedings of the Thirty-First National Conference on Artificial Intelligence, AAAI 2017, pp. 5061–5063. AAAI Press (2017)