Journal of Advances in Applied Mathematics, Vol. 3, No. 3, July 2018
https://dx.doi.org/10.22606/jaam.2018.33002

Mathematical Methods for a Quantum Annealing Computer

Richard H. Warren

Lockheed Martin Corporation-Retired
Email: rhw3@psu.edu

Abstract. This paper describes the logic and creativity needed in order to have high probability of solving discrete optimization problems on a quantum annealing computer. Current features of quantum computing via annealing are discussed. We illustrate the logic at the forefront of this new era of computing, describe some of the work done in this field, and indicate the distinct mindset that is used when programming this type of machine. The traveling salesman problem is formulated for solving on a quantum annealing computer, which illustrates the methods for this computer.

Keywords: Quantum annealing, Ising objective function, Boolean logic, penalty functions, traveling salesman problem

1 Introduction

Quantum computing has been called an emerging technology that is a paradigm shift from digital computing. From our viewpoint, it is more than a shift; it is an entirely new way to solve optimization problems. Old problems are being reformulated so that quantum techniques can be applied with high probability of finding a global minimum. Hardware is being built that is stable. Operating systems are being designed that include error correction. Several small quantum computers are available to researchers.

The design and fabrication of several generations of quantum annealing computers (QACs) to solve hard optimization problems has aroused interest in their expected speedup [8, 14, 26], real-world applications [1, 2, 4, 9, 21, 22, 25], and benchmarks [14 - 16]. Current research is focused on framing old problems to quantum annealing [10, 18, 30], embedding problems into quantum bits and their connections [5, 13], suppressing errors due to the nature of the apparatus [11, 21, 24], scaling large data to fit QACs [23], and comparing quantum annealers with thermal annealers [19, 21]. The above list of topics and [32] indicate that we are working in an interdisciplinary field.

This paper shows the logic for implementing discrete optimization problems on a D-Wave quantum annealing computer (QAC). Some of these techniques are scattered on three websites [3, 6, 23] and we develop others. This paper brings together the basic principles for solving discrete optimization problems on a D-Wave QAC with great likelihood.

D-Wave's initial processor had 128 quantum bits (called qubits). It has been upgraded and now has 2048 qubits. There are limited connections between qubits. Johnson and his colleagues [12] have an excellent description and diagram of the physical makeup of a qubit in a D-Wave computer. When super cooled, a qubit reaches its low energy state. Thus, when a family of qubits is associated with the variables in an optimization problem, the low energy state of the qubits corresponds to binary values associated with the variables, thus showing which variables form a minimum for the problem. Globally, this can be regarded as examining all solutions simultaneously and selecting a near minimum one. Authors [7, 12, 13, 20] describe D-Wave QACs.

In essence, a QAC does one thing: it finds near optimal answers to discrete minimization problems extremely fast. In order to do this, it needs an objective function and constraints expressed with binary variables. This requires declaring the variables, their types, and their coefficients based on the problem. There are no procedural instructions, such as "if X, then do Y."

A QAC is not the popularized circuit model on which Shor's factorization algorithm is designed to operate [28]. Shor's algorithm factors products of prime numbers in time growing polynomially in the size of the integer to be factored.

2 Outline and Terminology

Section 3 of this paper contains the unique input function for a QAC, equation (2). In Section 4 elementary Boolean operations are translated into QAC inputs. Penalty functions are introduced in Section 5. Logic gates are the topic in Section 6. The paper concludes in Section 7 with a formulation of the traveling salesman problem for quantum annealing.

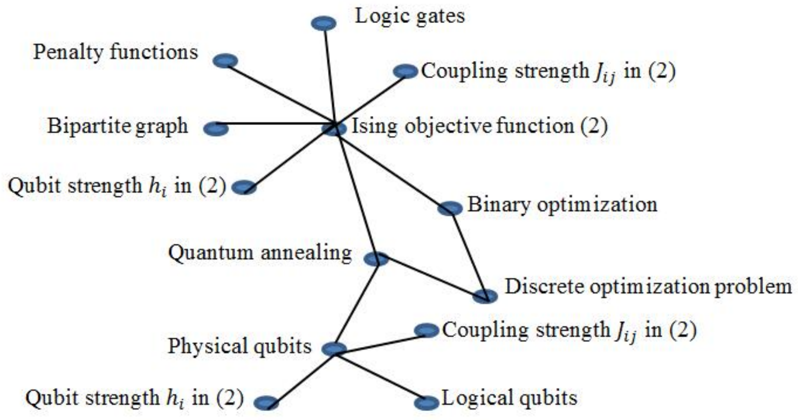
Logic gates
Penalty functions
Coupling strength J_{ij} in (2)
Bipartite graph
Ising objective function (2)
Qubit strength h_i in (2)
Binary optimization
Quantum annealing
Discrete optimization problem
Physical qubits
Coupling strength J_{ij} in (2)
Qubit strength h_i in (2)
Logical qubits

Figure 1. Terminology and relationships for quantum annealing computers.

3 Ising Model for Quantum Annealing

The theory for quantum annealing [7] implies that the qubits will achieve an optimal state of low energy when super cooled. This is represented by the following expression where an initial Hamiltonian H_0 evolves to its low energy state in a final Hamiltonian H_p according to

$$H(t) = \left(1 - S\left(\frac{t}{T}\right)\right)H_0 + S\left(\frac{t}{T}\right)H_p \text{ for } 0 \le t \le T \tag{1}$$

as $S(\tau)$ increases from $S(\tau) = 0$ to $S(1) = 1$ and if H_0 and H_p do not commute [18]. In theory, T is the time imposed by the Scrödinger equation for the initial Hamiltonian to evolve to its low energy state [27]. On a D-Wave QAC, time T is in microseconds. The Hamiltonian H_0 is established by D-Wave for all problems. The Hamiltonian H_p represents the combinatorial problem to be solved and is an input to a QAC. Determining H_p is a subject of this paper. Besides H_p, other inputs include the number of samples of the problem, the time T within a given range, and scaling factors. Essentially, a result $H(t)$ is a sample from a Boltzmann distribution.

A physical implementation of (1) does not strictly meet the conditions for quantum annealing. In addition, values loaded by the user may differ slightly from the machine interpretation of the numbers. These difficulties are overcome by making multiple samples and choosing a valid solution that has minimum energy.

The Ising objective function [13, 17, 23, 29] for the optimal state H_p in (1) is

$$\min\left(\sum_{i>j} s_i J_{ij} s_j + \sum_i h_i s_i\right) \tag{2}$$

where i and j are qubits and s_i is the final spin state of qubit i. Also h_i is the energy strength for qubit i and J_{ij} is the coupling energy between qubits i and j. The problem to be minimized dictates the values for the coefficients h_i and J_{ij}. An optimal state for (2) is an assignment of ±1 to the spin states s_i and s_j

by the QAC so that (2) is minimized.

The interface between the variables and the qubits in a QAC is the Hamiltonian H_p, which is a square, symmetric matrix with a row and column for each spin variable s_i that applies to the problem. The entries in H_p originate in the minimization problem. The diagonal entries of H_p are the energy strengths h_i assigned to qubits. The off-diagonal entries are the coupling energies J_{ij} assigned to the connections between qubits. The entries in H_p are adjusted for penalty functions that will be described in Section 5.

In summary, (2) is the unique format for a discrete optimization problem to be processed on a QAC. The variables h_i and J_{ij} in (2) are from the problem and are inputs to a QAC.

4 Boolean Applications

Let i and j designate qubits in a QAC. We will use a transcribed model where $x_i, x_j \in \{0,1\}$. This is convenient for Boolean algebra. If $s_i \in \{-1, 1\}$ and $x_i \in \{0, 1\}$, the relationship $s_i = 2x_i - 1$ equates the spin state model (2) and a Boolean model. So we may replace s_i and s_j in (2) with x_i and x_j, and assume that the values 0, 1 are assigned to the qubits in the minimization process. Next we will show how to set some Boolean operations so they fit (2) and can be used on a QAC.

Suppose we want to set the state x_i of qubit i to be a constant 0 or 1. Let h_i be the local field at i where $h_i x_i$ is a term in (2). If we set h_i sufficiently less than 0, then x_i will be minimized to 1 in the annealing process represented by (2). Or if we set h_i sufficiently greater than 0, then x_i will be minimized to 0.

Suppose we want to set $x_k = x_i$ which represents the Boolean states of qubits k and i being identical, i.e., we want two physical qubits, each with 5 connections to other qubits, to form a logical qubit with 8 connections to other qubits[1]. The equation $x_k = x_i$ does not conform to (2). So we look for a relationship between x_k and x_i that have the same value, say v, when $x_k = x_i$ and at least $v+1$ when $x_k \neq x_i$. This is represented by the catalog

x_k	x_i	Value
1	1	v
1	0	at least $v+1$
0	1	at least $v+1$
0	0	v

From (2) there are three variables that can be used: h_i, h_k and J_{ik}. A solution is $-2x_i x_k + x_i + x_k$ which minimizes at $v = 0$ if and only if $x_k = x_i$. Thus $h_i = 1$, $h_k = 1$ and $J_{ik} = -2$. This is the first entry in Table 2.

Table 1. Truth tables for basic Boolean operations on two binary variables

		Conjunction (And)	Disjunction (Inclusive Or)	Implication	Exclusive Or	Equivalence
x_i	x_j	$x_i \wedge x_j$	$x_i \vee x_j$	$x_i \rightarrow x_j$	$x_i \oplus x_j$	$x_i \equiv x_j$
1	1	1	1	1	0	1
1	0	0	1	0	1	0
0	1	0	1	1	1	0
0	0	0	0	1	0	1

[1] Physical connectivity between qubits is limited. The connections between 8 qubits in a unit in a D-Wave QAC form a complete bipartite graph on 4 vertices $K_{4,4}$ [13 Figure 1, 14 Figure 1]. In addition, each qubit connects to one qubit in an adjacent unit. The 1152 qubit chip is composed of a two dimensional array of 12 × 12 units, each containing 8 qubits.

Let $\overline{x_k}$ be the negation of x_k where $\overline{x_k} = 0$ if $x_k = 1$ and $\overline{x_k} = 1$ if $x_k = 0$. The quantum annealing adjustments for this operation are in Table 2.

Table 1 contains the truth tables for the Boolean operations addressed in Table 2.

Table 2. Expressions for Boolean logic

Boolean Logic	Corresponding Equation(s)	Corresponding Penalty Function
$x_k = x_i$	$x_k = x_i$	$-2x_ix_k + x_i + x_k$
$x_k = \overline{x_i}$	$x_k = 1 - x_i$	$2x_ix_k - x_i - x_k$
$x_k = x_i \wedge x_j$	$x_k = x_ix_j$	$x_ix_j - 2\left(x_i + x_j\right)x_k + 3x_k$
$x_k = x_i \vee x_j$	$x_k = x_i + x_j - x_ix_j$	$x_ix_j + \left(x_i + x_j\right)\left(1 - 2x_k\right) + 1$
$x_k = \left(x_i \to x_j\right)$	$x_k = \overline{x_i} \vee x_j = 1 - x_i + x_ix_j$	$4x_ix_j + 2x_ix_k - 6\left(x_i + x_j\right)x_a - 2x_kx_a - x_i - x_k + 9x_a$
$x_k = x_i \oplus x_j$	$x_k = \left(x_i \vee x_j\right) \wedge \left(\overline{x_i} \vee \overline{x_j}\right)$ $= x_i + x_j - 2x_ix_j$ $= \left(x_i + x_j\right) mod\ 2$ $= \left(x_i - x_j\right) mod\ 2$	$2x_ix_j - 2\left(x_i + x_j\right)x_k - 4\left(x_i + x_j\right)x_a +$ $4x_kx_a + x_i + x_j + x_k + 4x_a$ There is no quadratic penalty function using only variables x_i, x_j and x_k.
$x_k = \left(x_i \equiv x_j\right)$	$x_k = \overline{\left(x_i \oplus x_j\right)} = 1 - x_i - x_j + 2x_ix_j$	$2x_ix_j + 2\left(x_i + x_j\right)x_k - 4\left(x_i + x_j\right)x_a -$ $4x_kx_a - x_i - x_j - x_k + 8x_a$

In Table 2 expressions for Boolean logic are translated to penalty functions that can be implemented on a QAC. The notation a represents an ancillary qubit so that $x_a = x_ix_j$ and is introduced to reduce a product of three variables to a quadratic term. This is necessary because the Ising objective function (2) is quadratic.

The penalty functions for $\wedge, \vee$ and $\oplus$ are from [23 Table 4.1]. The penalty function for $\equiv$ is the negation of the penalty function for $\oplus$ added to 4 times the penalty function for $\wedge$. Similarly, the penalty function for $\to$ is a sum of two penalty functions. The first is derived from the algebraic equation for $\to$. The other is 3 times the penalty function for $\wedge$.

5 Penalty Functions

Equations, inequalities and expressions of Boolean logic cannot be entered directly in (2) for computing. Instead, penalty functions are used to represent them. We will explain how to convert to penalty functions.

Table 2 shows a conversion of expressions for Boolean logic to penalty functions. To verify that a penalty function represents its Boolean logic, it is necessary to show that the penalty function yields the same value, say v, for each assignment of truth values to the expression of the Boolean logic, and that the penalty function yields at least v + 1 for each assignment of false values to the expression of the Boolean logic. The latter is needed because (2) is a minimization statement.

A constraint that is an equation can be changed to a penalty function by noting that a parable which opens upward has a minimum value at its vertex. We convert an equation to a generalized parable by reversing the algebraic sign of all terms on one side of the equation and deleting the equality sign. Next, we square the result, simplify it with the property $x^2 = x$ for binary variables 0 and 1, and delete the constant term so that the minimum value is 0. (A minimum value of 0 does not change when the penalty function is multiplied by a positive scalar.) It is easily seen that a binary solution occurs for the constraint equation if and only if its penalty function attains a global minimum at this solution. We illustrate this general method for the equation $x_k = x_i$ from Section 4.

Step 1. Reverse the algebraic sign of the term on the left side of the equation and delete the equality sign: $-x_k + x_i$.

Step 2. Square the result: $\left(-x_k + x_i\right)^2 = -2x_i x_k + x_i^2 + x_k^2$.

Step 3. Simplify by the property $x^2 = x$ for binary variables 0 and 1: $-2x_i x_k + x_i + x_k$ which is the result in Table 2.

Step 4. Delete the constant term. This step is not needed for this example.

A constraint inequality can be converted to a constraint equation by inserting slack variables, as is done in the simplex algorithm for linear programming.

See [23 Section 4.1.3] for a systematic method to construct constraint equations and constraint inequalities for a specific problem. It also has techniques to derive penalty functions.

The largest degree of the terms in (2) is 2. If a term in a penalty function has degree greater than 2, the degree can be reduced by introducing an ancillary qubit that represents the product of two variables. There are pitfalls, such as an ancillary qubit substituted in a quadratic term may destroy the logic of the penalty function. The penalty functions for Boolean operations $\rightarrow, \oplus$ and $\equiv$ in Table 2 require an ancillary qubit. These penalty functions were tested for false values of the dependent variables x_k and x_a occurring simultaneously, in addition to testing them separately.

6 Logic Gates

A logic gate is a device, ideal or real, that implements a Boolean operation [5, 31]. Logic gates are usually separate steps in the popularized circuit model of quantum computing. However, in quantum annealing, according to (2) there are no separate steps. This implies that a logic gate in quantum annealing needs to be part of the single input. As result, logic gates do not seem to appear in quantum annealing unless they are an intrinsic part of the minimization problem. We will discuss a gate for one of the five basic Boolean operations in Table 1, Exclusive Or.

The controlled-NOT gate (also called the CNOT gate or XOR gate) in a circuit model quantum computer negates the target bit if and only if the control bit is 1. The truth table for the CNOT gate is:

Input		Output	
Control	Target	Control	Target
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

The CNOT gate is fundamentally important in a circuit model quantum computer, because all gates can be generated by one-bit gates and the CNOT gate [28 page 1489].

The truth table for Exclusive Or in Table 1 is identical to the truth table for the CNOT gate. Therefore, a natural way to implement a CNOT gate for a QAC is to use Exclusive Or. Thus, a CNOT gate in quantum annealing needs to act like the penalty function for $\oplus$ in Table 2.

The CNOT gate in the circuit model is composed of two qubits, a target and a control. According to Table 2, since the CNOT gate equates to $\oplus$, the CNOT gate in a QAC uses four qubits. They are target i, control j, result k and ancillary a. We will form a Hamiltonian for a CNOT gate using these four qubits. Let x_i, x_j, x_k and x_a be their corresponding value in {0, 1}. From Table 2 the penalty function for the CNOT gate is

$$2x_i x_j - 2\left(x_i + x_j\right)x_k - 4\left(x_i + x_j\right)x_a + 4x_k x_a + x_i + x_j + x_k + 4x_a \tag{3}$$

In (3), $x_a = x_i x_j$ and is used to reduce $x_i x_j x_k$ to a quadratic term which is required by (2). The coefficients of the linear terms in (3) are placed on the diagonal of the Hamiltonian and the coefficients of the quadratic terms are placed off the diagonal. Based on (3), the Hamiltonian for the CNOT gate is:

	i	j	k	a
i	1	2	-2	-4
j	2	1	-2	-4
k	-2	-2	1	4
a	-4	-4	4	4

A QAC assigns values to x_i, x_j, x_k and x_a from {0, 1} so that (3) is minimized.

A CNOT gate is reversible in quantum computing. After the gate is executed, if the control is applied to the result, then the original target is recovered.

In summary, in quantum annealing a logic gate may be regarded as a Hamiltonian that implements its corresponding Boolean operation. This Hamiltonian is combined with a Hamiltonian that represents an optimization problem for a single input to a QAC.

7 The Traveling Salesman Problem

To illustrate the logic in this paper, the traveling salesman problem (TSP) is formulated for implementation on a QAC. This is an improvement and expansion of the presentations in [30] by incorporating the redundancy of one city. Given a set of cities and the directed distances between each pair of cities, the task of the TSP is to find a shortest route that visits each city exactly once and returns to the starting city.

Let the cities be designated 1, 2, ...,n. Let d_{ij} be the distance from city i to city j. We do not assume that $d_{ij} = d_{ji}$. A *tour* is a cyclic permutation of the cities, i.e., a permutation that has one cycle. The TSP asks for a tour of the cities that returns to the starting city and minimizes the distance traveled, called an *optimal tour.* As in [30], our binary variables for the TSP are V_{it} representing city i occurring in position t of a tour. For an n-city problem, the indices satisfy $1 \leqslant i \leqslant n$ and $1 \leqslant t \leqslant n$. We want city 1 to be in the first position of all tours. This is accomplished by setting $V_{11} = 1$ and not using V_{i1} for $2 \leqslant i \leqslant n$.

A *subtour* is a cyclic permutation of a proper subset of the cities. Subtours must be prevented from spoiling the selection of an optimal tour. Both [18, 30] do this in the objective function by having the subscripts on the binary variables represent the position of a city in a tour. Moreover, the distance from city i to city j counts toward the optimal tour length only when cities i and j are consecutive in the tour and city i precedes city j. A bonus of this formulation is that both symmetric and asymmetric TSPs can be represented.

Employing the pattern of (2), our objective function for the TSP is

$$\sum_{j=2}^{n} d_{1j} V_{11} V_{j2} + \sum_{\substack{i,j=2 \\ i \neq j}}^{n} \sum_{t=2}^{n-1} d_{ij} V_{it} V_{j,t+1} + \sum_{i=2}^{n} d_{i1} V_{in} V_{11}; \text{and} \quad V_{11} = 1$$

This is equivalent to

$$E = \sum_{j=2}^{n} d_{1j} V_{j2} + \sum_{\substack{i,j=2 \\ i \neq j}}^{n} \sum_{t=2}^{n-1} d_{ij} V_{it} V_{j,t+1} + \sum_{i=2}^{n} d_{i1} V_{in} \tag{4}$$

We want the D-Wave processor to assign 0, 1 to the binary variables V_{it} so that (4) is minimized and the variables are subject to the following constraints.

$$\text{For each } t \in \{2,3,\ldots,n\} \sum_{i=2}^{n} V_{it} = 1 \tag{5}$$

$$\text{For each } i \in \{2,3,\ldots,n\} \sum_{t=2}^{n} V_{it} = 1 \tag{6}$$

Constraint (5) ensures that each position $t \geq 2$ has exactly one city in an outcome. Constraint (6) ensures that each city $i \geq 2$ occurs exactly once in an outcome. The notation in (4) ensures that subloops do not occur.

Constraints (5) and (6) need to be modified so they conform to (2). Using the steps in Section 5, the result for constraint (5) is

$$F_t = -\sum_{i=2}^{n} V_{it} + 2\sum_{j=2}^{n-1}\sum_{k=j+1}^{n} V_{jt}V_{kt} \text{ for } t \in \{2,3,\ldots,n\}. \tag{7}$$

We note that F_t attains its minimum when $V_{it} = 1$ for exactly one subscript i and $V_{it} = 0$ for all subscripts $j \neq i$. Thus, F_t attains its minimum when constraint (5) is true.

Similarly, constraint (6) can be implemented in the form of (2) by

$$G_i = -\sum_{t=2}^{n} V_{it} + 2\sum_{r=2}^{n-1}\sum_{s=r+1}^{n} V_{ir}V_{is} \text{ for } i \in \{2,3,\ldots,n\}. \tag{8}$$

The input to the D-Wave quantum computer is

$$\lambda_1 * E + \lambda_2 * \left(\sum_{t=2}^{n} F_t + \sum_{i=2}^{n} G_i\right). \tag{9}$$

The λ_i are Lagrange multipliers to scale the pieces of the input in order to control errors and improve precision.

The TSP as formulated in (4) – (6) has been verified on a D-Wave QAC for n = 6, 7, and 8, $\lambda_1 = 1$, λ_2 =1500 and distances between real cities ranging from 302 to 2230. When n = 9, the embedding requires more than 1200 qubits. The D-Wave software "dw" was used. It has the advantage of finding an embedding of the variables into the qubits, avoiding the Hamiltonian matrix which is tedious to construct, and determining whether solutions satisfy assertions that describe the constraints.

However, the D-Wave processor and the "dw" software have difficulties. They sample from 1000 solutions in order to have a high probability of finding a valid solution. However, sometimes all 1000 solutions of a TSP are invalid. Also, the documentation about scaling factors seems sketchy and incomplete. We set the parameters by experimenting without fully knowing what was being affected.

8 Conclusion

We have shown that computation on a quantum annealing computer has rich mathematical methods which are different than those for digital computing. The major source of the difference is the single input statement (2) for computing on a quantum annealer.

Acknowledgments. Several colleagues are thanked for comments and suggestions that significantly improved previous versions of the manuscript. Denny Dahl and Joel Gottlieb of D-Wave Systems are thanked for helping to install the TSP on a D-Wave quantum annealing computer. The Lord Jesus is thanked for providing motivation and financial resources to investigate these topics and write this paper.

References

1. S. H. Adachi and M. P. Henderson. 2015. Application of quantum annealing to training of deep neural networks. arXiv:1510.06356, 18 pages.
2. M. Benedetti, J. Realpe-Gomez, R. Biswas, and A. Perdomo-Ortiz. 2016. Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning. *Phys. Rev. A* 94, 022308.
3. Z. Bian, F. Chudak, W. G. Macready, G. Rose. 2010. The Ising model: teaching an old problem new tricks. D-Wave Systems. http://www.dwavesys.com/sites/default/files/weightedmaxsat_v2.pdf
4. Z. Bian, F. Chudak, W. G. Macready, L. Clark, F. Gaitan. 2013. Experimental determination of Ramsey numbers. *Phys. Rev. Lett.* 111, 130505.
5. Z. Bian, F. Chudak, R. Israel, B. Lackey, W. G. Macready, A. Roy. 2014. Discrete optimization using quantum annealing on sparse Ising models. *Front. Phys.* 2 Article 56.

6. E. Boros and P. L. Hammer. 2001. Pseudo-Boolean Optimization.
http://rutcor.rutgers.edu/~boros/Papers/2002-DAM-BH.pdf
7. S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, M. Troyer. 2014. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics* 10, 218-224. DOI: 10.1038/nphys2900
8. S. Boixo, G. Ortiz, R. Somma. 2015. Fast quantum methods for optimization. *Eur. Phys. J. Special Topics* 224, 35-49.
9. M. Henderson, J. Novak, T. Cook. 2018. Leveraging adiabatic quantum computation for election forecasting. arXiv:1802.00069
10. V. Horan, S. Adachi, S. Bak. 2016. A comparison of approaches for finding minimum identifying codes on graphs. *Quantum Inf. Process.* 15, 1827-1848.
11. Z. Jiang and E. G. Rieffel. 2017. Non-commuting two-local Hamiltonians for quantum error suppression. *Quantum Inf. Process.* 16, Article 89. https://doi.org/10.1007/s11128-017-1527-9
12. M. W. Johnson et al. 2011. Quantum annealing with manufactured spins. *Nature* 473, 194-198.
DOI: 10.1038/nature10012
13. K. Karimi, N. G. Dickson, F. Hamze, M. H. S. Amin, M. Drew-Brook, F. A. Chudak, P. I. Bunyk, W. G. Macready, G. Rose. 2012. Investigating the performance of an adiabatic quantum optimization processor. *Quantum Inf. Process.* 11, 77-88.
14. H. G. Katzgraber, F. Hamze, R. S. Andrist. 2014. Glassy chimeras could be blind to quantum speedup: Designing better benchmarks for quantum annealing machines. *Phys. Review X* 4, 021008.
15. J. King, S. Yarkoni, M. M. Nevisi, J. P. Hilton, C. C. McGeoch. 2015. Benchmarking a quantum annealing processor with the time-to-target metric. arXiv:1508.05087
16. D. Korenkevych, Y. Xue, Z. Bian, F. Chudak, W. G. Macready, J. Rolfe, E. Andriyash. 2016. Benchmarking quantum hardware for training of fully visible Boltzmann machines. arXiv:1611.04528
17. T. Lanting et al. 2014. Entanglement in a quantum annealing processor. *Phys. Rev. X* 4, 021041. DOI: 10.1103/PhysRevX.4.021041
18. A. Lucas. 2014. Ising formulations of many NP problems. *Front. Phys.* 2 Article 5, 15 pages. DOI:10.3389/fphy.2014.00005
19. V. Martin-Mayor and I. Hen. 2015. Unraveling quantum annealers using classical hardness. *Scientific Reports* 5, 15324.
20. C. C. McGeoch and C. Wang. 2013. Experimental evaluation of an adiabatic quantum system for combinatorial optimization, Proceedings of the ACM International Conference on Computing Frontiers, Article No. 23, ACM Press. DOI:10.1145/2482767.2482797
21. C. C. McGeoch. 2014. *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*, Morgan & Claypool.
22. F. Neukart, D. Von Dollen, C. Seidel, G. Compostella. 2018. Quantum-enhanced reinforcement learning for finite-episode games with discrete state spaces. Front. Phys. 5, Article 71. DOI:10.3389/fphy.2017.00071
23. Programming with QUBOs. 2016. Release 2.4, D-Wave Systems.
Available upon request at inquiry@dwavesys.com.
24. K. L. Pudenz, T. Albash, D. A. Lidar. 2014. Error-corrected quantum annealing with hundreds of qubits. *Nature Communications* 5, Article No. 3243.
25. E. G. Rieffel, D. Venturelli, B. O'Gorman, M. B. Do, E. M. Prystay, V. N. Smelyanskiy. 2015. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Inf. Process.* 14, 1-36.
26. T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, M. Troyer. 2014. Defining and detecting quantum speedup. *Nature* 345, 420-424.
27. G. E. Santoro and E. Tosatti. 2006. Optimization using quantum mechanics: quantum annealing through adiabatic evolution. *J. Phys. A* 39, R393-R431. DOI:10.1088/0305-4470/39/36/R01
28. P. Shor. 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* 26, 1484-1509.
29. S. Suzuki, J. Inoue, B. K. Chakrabarti. 2013. *Quantum Ising Phases and Transitions in Transverse Ising Models*, 2nd edition, Springer-Verlag.

30. R. H. Warren. 2013. Adapting the traveling salesman problem to an adiabatic quantum computer, *Quantum Inf. Process*. 12, 1781-1785. DOI:10.1007/s11128-012-0490-8
31. J. D. Whitfield, M. Faccin, J. D. Biamonte. 2012. Ground-state spin logic. *Europhysics Letters* 99, 57004. DOI:10.1209/0295-5075/99/57004
32. T. Albash and D. A. Lidar. 2018. Adiabatic quantum computing, *Rev. Mod. Phys.* 90, 015002. DOI:10.1103/RevModPhys.90.015002