Noname manuscript No.
(will be inserted by the editor)

Practical Integer-to-Binary Mapping for Quantum Annealers

Sahar Karimi · Pooya Ronagh

arXiv:1706.01945v1 [quant-ph] 6 Jun 2017

Received: date / Accepted: date

Abstract Recent advancements in quantum annealing hardware and numerous studies in this area suggests that quantum annealers have the potential to be effective in solving unconstrained binary quadratic programming problems. Naturally, one may desire to expand the application domain of these machines to problems with general discrete variables. In this paper, we explore the possibility of employing quantum annealers to solve unconstrained quadratic programming problems over a bounded integer domain. We present an approach for encoding integer variables into binary ones, thereby representing unconstrained integer quadratic programming problems as unconstrained binary quadratic programming problems. To respect some of the limitations of the currently developed quantum annealers, we propose an integer encoding, named *bounded-coefficient encoding*, in which we limit the size of the coefficients that appear in the encoding. Furthermore, we propose an algorithm for finding the upper bound on the coefficients of the encoding using the precision of the machine and the coefficients of the original integer problem. Finally, we experimentally show that this approach is far more resilient to the noise of the quantum annealers compared to traditional approaches for the encoding of integers in base two.

Keywords Adiabatic Quantum Computation, Integer Programming, Integer Encoding, Bounded-Coefficient Encoding

1 Introduction

Adiabatic quantum computation (AQC) has been proposed as a successful technique for solving certain classes of optimization problems (see [1,6,15,18]). In particular, quantum annealers (QA)

Sahar Karimi (corresponding author)
1QB Information Technologies (1QBit)
E-mail: sahar.karimi@gmail.com

Pooya Ronagh
1QB Information Technologies (1QBit)
E-mail: pooya.ronagh@1qbit.com

such as the ones manufactured by D-Wave Systems Inc. [8] approximate the adiabatic evolution in the presence of various sources of noise and finite temperature to solve Ising models of the form

$$\min_{s\in\{\pm 1\}^n} \; s^t JS + h^t s\,, \tag{Ising}$$

where diagonal entries of J are zeroes, and the nonzero entries of J create a subgraph of a sparse graph called a *Chimera* graph [4]. Many researchers have explored the applicability of quantum annealing to more-general optimization problems. Solving Ising models without the Chimera graph structure on a QA is discussed in [5]. Degree reduction techniques, such as the one described in [7], enable us to solve unconstrained binary polynomial programming problems of higher order using a QA. Several studies [16,17,20,22] have employed quantum annealing for solving real-world applications, and [9] discusses the possibility of using QAs for solving constrained problems.

Many optimization problems, on the other hand, involve integer-valued variables beyond binary values. Examples of such problems could be the number of vehicles or products travelling along each route of a supply chain (see [19]), or the number of traded assets in portfolio optimization (see [13]). This paper focuses on solving quadratic problems on bounded integer domains, alternatively referred to as unconstrained integer quadratic programming (UIQP) problems, i.e., problems of the following form:

$$\begin{aligned} &\min\; x^T Q x + q^t x, \\ &\text{s.t.}\;\; x_i \in \{0,1,2,\ldots,\kappa^{x_i}\} \quad \text{for } i=\{1,2,\ldots,n\}, \end{aligned} \tag{UIQP}$$

where $\kappa^{x_i}\in\mathbb{Z}_+$ is an upper bound on x_i; here, $\mathbb{Z}_+$ denotes the set of non-negative integers. Note that in problems where $x_i\in\{\alpha,\alpha+1,\alpha+2,\ldots,\alpha+\kappa^{x_i}\}$, we may shift x_i and substitute it with $x_i-\alpha$; hence, without loss of generality, we may assume that the domain of x_i starts at 0. It is worth mentioning that, although current QAs are limited to representing Ising models, an alternative approach to encoding bounded integer variables in terms of spin variables is to consider physical implementations of quantum processors as representing Potts-Ising models [3,11]. Our approach is different and is based on reformulating a UIQP problem as an Ising model. To this end, we represent each integer variable as a linear combination of several binary variables, i.e.,

$$x_i = \sum_{j=1}^{d^{x_i}} c_j^{x_i} y_j^{x_i} = (c^{x_i})^t y^{x_i}, \tag{1}$$

where $c_j^{x_i}\in\mathbb{Z}_+$, $y_j^{x_i}\in\{0,1\}$ for $j\in\{1,\ldots,d^{x_i}\}$, and the superscript x_i is used for clarity to denote that c^{x_i} and y^{x_i} correspond to the encoding of variable x_i. This representation is referred to as *integer encoding*. Some of the well-known integer encodings are binary and unary encodings in which $c_j^{x_i}=2^{j-1}$ and $c_j^{x_i}=1$, respectively. The *width* of an integer encoding, denoted by d^{x_i} in (1), refers to the number of binary variables required for encoding integer variable x_i. The width of binary and unary encodings are $\lfloor\log_2(\kappa^{x_i})\rfloor+1$ and κ^{x_i}, respectively. Since a binary variable y_j could be represented with a spin variable s_j via the affine transformation

$$y_j = \frac{1}{2}\,(s_j+1)\,, \tag{2}$$

integer variable x_i could be encoded into several spin variables as

$$x_i = \frac{1}{2}\left(\sum_{j=1}^{d^{x_i}} c_j^{x_i} + \sum_{j=1}^{d^{x_i}} c_j^{x_i} s_j^{x_i}\right), \tag{3}$$

which enables us to successfully represent a UIQP as an Ising model.

After derivation of an Ising model equivalent to (UIQP), heuristic methods including quantum annealing could be employed to find the ground state of the problem. The performance of the heuristic methods highly depends on the energy landscape of the problem. In particular, landscapes with tall barriers are challenging for the hill-climbing heuristics, and having wide barriers could impact the performance of methods like quantum annealing that could potentially benefit from quantum tunnelling (see [10] and references therein for more details). Having coefficients that are different in orders of magnitude is undesirable, as they could create energy landscapes with tall and wide barriers. Moreover, D-Wave quantum annealers have only a low precision of approximately 10^{-2} for couplings' strengths and local fields' biases (i.e., entries of J and h in (Ising)), scaling J and h to a very limited range, (e.g., $[-2, 2]$). Several sources of noise such as thermal excitations and control errors contribute to the low precision of the machines; in [2], Albash et al. propose a noise model for D-Wave devices that includes the control noise of the local field and couplings of the chip. Zhu et al. [21] show that increasing the classical energy gap beyond the intrinsic noise level of the machine can improve the success of the D-Wave Two QA. In [21], *resilience* is defined as a metric for measuring the resistance of a problem to noise. More precisely, resilience is the probability that the ground state does not change under random field fluctuations found on the chip. In this paper, we argue that restricting the range of the coefficients of a problem could improve resilience; and our results, presented in Section 4 of this paper, support this argument

Studies in [12,14] suggest that QAs could be biased in finding degenerate ground states; in other words, sampling with QAs does not return different degenerate solutions with uniform probability. Moreover, degenerate ground states are easier to reach; therefore, benchmarks created for QAs tend to avoid degeneracy and have a unique ground state [10]. While there is a need to rigorously study whether highly degenerate low-energy excited states could impact reaching the ground state, avoiding degeneracy could be an alternative. Recall that in integer encoding, an integer variable, x, is substituted with $c^t y$, where we assume $c \in \mathbb{Z}_+^d$ and $y \in \mathbb{B}^d$. Any binary vector y returns an integer value. If the total number of binary combinations, i.e., 2^d, is larger than the summation of entries of c, i.e., $\sum_{i=1}^{d} c_i$, some of the integers occur at more than one binary combination. The *code-word* of an integer value χ, for $\chi \in \{1, 2, \ldots, \kappa\}$, refers to the cardinality of the set $\{y : c^t y = \chi\}$. Code-words more than one is what we refer to as *redundancy* in this paper. Under the assumption that the QA is inclined to observe integers with higher code-words, it is ideal in an integer encoding to have a unique code-word for all integers. Binary encoding, for example, keeps the code-word of each integer uniformly at one, but the code-word in unary encoding is highly variant and the redundancy has its peak at $\lceil \frac{\kappa}{2} \rceil$ with word count $\binom{\kappa}{\lceil \frac{\kappa}{2} \rceil}$.

This work presents an integer encoding with minimal width that creates noise-resilient Ising models. Since one or several qubits is assigned to each spin variable and the number of qubits on a chip is limited, it is desirable to keep the size of the Ising model, and thus the width of the encoding, as small as possible. Binary encoding has the minimum width and uniform code-word; however, the coefficients c_j's in the binary encoding could get arbitrarily large and result in an error-prone

(noisy, non-resilient) Ising model. On the other hand, unary encoding does not expand the range of the coefficients of the problem, but it has an exceedingly large degree and redundancy.

The notation used in this paper is as follows. Generally, we reserve the upper-case letters, lower-case letters, and Greek alphabet for matrices, vectors, and scalars, respectively, with only a few exceptions; these exceptions, however, should be clear from the context. An all-ones vector and the identity matrix are denoted by e and I, respectively. Entries of a vector or matrix are differentiated with subscript indices. $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$ denote the ceiling and floor of a number, respectively. The $\log(\cdot)$ function is in base two; and $\mathrm{sgn}(\alpha)$ returns the sign of α. The function $\delta(A) : \mathbb{R}^{n\times n} \to \mathbb{R}^n$ returns a vector with the diagonal entries of A; and the function $\Delta(a) : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ returns a diagonal matrix with entries of a on the diagonal.

This paper is organized as follows. In the next section, we present bounded-coefficient encoding under the assumption that an upper bound on the coefficients of the integer encoding is available. In Section 3, we present how, given the precision, we may find an upper bound on the coefficients of the integer encoding. We test our algorithm by comparing binary and bounded-coefficient encodings in Section 4. Finally, we conclude our discussion in Section 5.

2 Bounded-Coefficient Encoding

Let x be an integer variable with upper bound κ^x, and μ^x be an upper bound on the coefficients of the integer encoding. In other words, we represent x as

$$x = c^t y, \tag{4}$$

where $y \in \mathbb{B}^{d^x}$, $c \in \mathbb{Z}_+^{d^x}$, and $c_i \leq \mu^x$ for $i = 1, \ldots, d^x$. The encoding we propose in this paper, which we refer to as *bounded-coefficient encoding*, is summarized in the following algorithm:

Notice that if $\kappa^x < 2^{\lfloor \log \mu^x \rfloor + 1}$, then the binary encoding respects the upper bound on the coefficients. These cases are captured in (5), and the width of encoding in these instances is $\lfloor \log(\kappa^x) \rfloor + 1$.

When $\kappa^x \geq 2^{\lfloor \log \mu^x \rfloor + 1}$, the encoding is derived using (6) and the width of the bounded-coefficient encoding is

$$d^x = \begin{cases} \rho + \eta + 1 & \text{if } \nu - \eta\mu \neq 0 \\ \rho + \eta & \text{otherwise} \end{cases}. \tag{7}$$

Here are a few demonstrative examples of bounded-coefficient encoding:

- $\kappa = 12$ and $\mu = 8$, the bounded-coefficient encoding is $c = [1,\ 2,\ 4,\ 5]^t$;
- $\kappa = 20$ and $\mu = 6$, the bounded-coefficient encoding is $c = [1,\ 2,\ 4,\ 6,\ 6,\ 1]^t$.

We assume $\kappa^x \gg 2^{\lfloor \log \mu^x \rfloor + 1}$ when we refer to the bounded-coefficient encoding, and our general statements are focused on these cases where we have multiple coefficients of size μ^x. Propositions 1 and 2, also, refer to these cases described by (6); these propositions hold true for binary encoding; hence, they hold for the cases that are derived with (5), such as our first example above.

Definition 1 *An encoding c is called κ-complete if it can encode only and all integers $\{0, 1, \ldots, \kappa\}$.*

Algorithm 1 Bounded-Coefficient Encoding

Inputs:
κ^x: upper bound on the integer variable x
$\mu^x \ll \kappa^x$: upper bound on the coefficients of the encoding
Output:
c^x: integer encoding coefficients

if $\kappa^x < 2^{\lfloor \log(\mu^x) \rfloor + 1}$
return

$$c^x = \left[2^0, 2^1, \ldots, 2^{\lfloor \log(\kappa^x) \rfloor - 1}, \kappa^x - \sum_{i=1}^{\lfloor \log(\kappa^x) \rfloor} 2^{i-1} \right] \tag{5}$$

else
compute $\rho = \lfloor \log \mu^x \rfloor + 1$, $\nu = \kappa^x - \sum_{i=1}^{\rho} 2^{i-1}$, and $\eta = \left\lfloor \frac{\nu}{\mu^x} \right\rfloor$
return vector c^x with entries

$$c_i^x = \begin{cases} 2^{i-1} & \text{for } i = 1, \ldots, \rho \\ \mu^x & \text{for } i = \rho + 1, \ldots, \rho + \eta \\ \nu - \eta\mu^x & \text{for } i = \rho + \eta + 1 \text{ if } \nu - \eta\mu^x \neq 0 \end{cases} \tag{6}$$

Proposition 1 *The bounded-coefficient encoding, generated by Algorithm 1, is κ^x-complete.*

By the first part of our encoding, i.e., $1, \ldots, 2^{\rho-1}$, we can encode all integers $\{0, 1, \ldots, \mu^x\}$. By adding μ^x to those binary combinations, we can generate all integers from μ^x to $2\mu^x$; if we add two μ^x factors, we get integers $2\mu^x$ to $3\mu^x$, and so on. Finally, we are guaranteed that we get integers $\eta\mu^x$ to κ^x because $\kappa^x - \eta\mu^x < \sum_{i=1}^{\rho} 2^{i-1} - \mu^x$ by the fact that $\nu - \eta\mu^x < \mu^x$.

Definition 2 *A sub-encoding $\tilde{c}$ of an encoding c is a choice of entries of c, with a fixed ordering arranged in a column vector $\tilde{c}$.*

Proposition 2 *Bounded-coefficient encoding, generated by Algorithm 1, has minimum width among all κ^x-complete integer encodings of the form* (1) *that respect an upper bound on their coefficients.*

Proof Our arguments rely on the optimality of binary encoding, or, more rigorously, the following points:

(a) binary encoding has minimum width; in other words, the binary encoding of values $\{0, 1, \ldots, \mu^x\}$ has width $d_B = \lfloor \log \mu^x \rfloor + 1$, and no other μ^x-complete integer encoding could have a width less than d_B.

(b) binary encoding of width d_B, i.e., $\left(2^0,\ 2^1,\ 2^2,\ \ldots,\ 2^{d_B-1}\right)$, encodes integers of maximum value $2^{d_B} - 1$; i.e., $\sum_{i=0}^{d_B-1} 2^i = 2^{d_B} - 1$. Notice that by (a), this is the largest number that can have an encoding of width d_B.

It is worth mentioning that if $\kappa^x < 2^{\lfloor \log \mu^x \rfloor + 1}$, then the bounded-coefficient encoding has width $\lfloor \log \kappa^x \rfloor + 1$, and by (a) has minimum width.

For more-general cases, our proof is by contradiction. Suppose c is the bounded-coefficient encoding derived by Algorithm 1, and f is a bounded-coefficient encoding of smaller width; i.e., $d_c > d_f$. Let f_μ be the minimal sub-encoding of f that encodes $\{0, 1, \ldots, \mu^x\}$, and let $d_{f_\mu} \leq d_f$ be the width of f_μ. Similarly, c_μ in the bounded-coefficient encoding derived by Algorithm 1 refers to the sub-encoding of c required to encode integers less than or equal to μ^x. As implied by Algorithm 1 and (6), $d_{c_\mu} = \rho$. By (a), we conclude $d_{f_\mu} \geq \rho$. We can, now, consider two cases: $d_{f_\mu} = \rho$ and $d_{f_\mu} > \rho$.

Consider the case where $d_{f_\mu} = \rho$. By (a) and (b), $\sum_{\alpha\in c_\mu} \alpha \geq \sum_{\alpha\in f_\mu} \alpha$; therefore, $\kappa^x - \sum_{\alpha\in f_\mu} \alpha \geq \nu$, where ν is as defined in Algorithm 1. Since all of the coefficients need to be bounded by μ^x, and by the fact that $\frac{\kappa^x - \sum_{\alpha\in f_\mu^x} \alpha}{\mu^x} \geq \frac{\nu}{\mu^x}$, we conclude that having $d_f < d_c$ is impossible.

Consider now the case that $d_{f_\mu} \geq \rho+1$. As all of our coefficients are bounded by μ^x, $\sum_{\alpha\in f_\mu} \alpha < 2\mu^x$; otherwise, this contradicts that f_μ is the minimal sub-encoding required to encode integers less than or equal to μ^x. While $\sum_{\alpha\in f_\mu} \alpha < 2\mu$, $\sum_{\alpha\in c_\mu} \alpha + \mu \geq 2\mu$. Similar to the previous case, we conclude that $\kappa^x - \sum_{\alpha\in f_\mu} \alpha > \kappa^x - \sum_{\alpha\in c_\mu} \alpha + \mu^x$; hence, $d_f < d_c$ would not be possible. □

As mentioned earlier, a potentially advantageous property of an integer encoding is uniform or low-variant redundancy. In general, the bounded-coefficient encoding has redundancy because we have more than one coefficient with a value of μ. It is, however, worth mentioning that forcing the following constraints on the binary variables of (4) makes the word count of each integer value unique:

$$\begin{array}{rl} \sum_{j=1}^{\rho} c_j y_j \geq (2^\rho - \mu^x) y_i & \text{for } i = \rho + 1, \\ y_i \geq y_{i+1} & \text{for } i = \rho+1, \ldots, d^x - 1, \\ \sum_{j=1}^{\rho} c_j y_j \geq (2^\rho - c_{d^x}) y_{d^x}. & \end{array} \tag{8}$$

Note that $(2^\rho - \mu^x) \leq \mu^x$; however, $(2^\rho - c_{d^x})$ may not necessarily be less than or equal to μ^x. If we wish to limit the coefficients of our constraints to μ^x, we can substitute $(2^\rho - c_{d^x}) y_{d^x}$ in the right-hand side of the last inequality with $\mu^x y_{d^x-1} y_{d^x} + (2^\rho - \mu^x - c_{d^x}) y_{d^x}$ whenever $2^\rho - c_{d^x} > \mu^x$. Handling constraints, despite being possible, is non-trivial for QA; it may introduce exceedingly large coefficients (see discussion in [9]), and could be contradictory to the purpose of this work.

As we presented earlier in (2) and (3), we may directly encode integer variables into spin variables. Moreover, by the fact that in the bounded-coefficient encoding $\sum_{j=1}^{d^x} c_j = \kappa^x$, the encoding of each integer variable into spin variables would be

$$x = \frac{1}{2}\left(\kappa^x + \sum_{j=1}^{d^x} c_j^x s_j^x\right). \tag{9}$$

In the next section, where we try to find the upper bound on the coefficients of the encoding, i.e., μ^x, we use the integer-to-spin transformation.

3 Finding the Upper Bound on the Coefficients of the Encoding

We explained earlier, in Section 1, that the QA scales the coefficients of an Ising model to a limited range and has low precision. We also mentioned that resilience to noise could be helpful in

overcoming the issue of precision. We propose that restricting the range of the coefficients of the problem could improve resilience. To this end, we wish to find upper bounds on the coefficients of the encoding, i.e., μ, such that after the encoding of the integer problem (UIQP) to an Ising model (Ising), the ratio of the smallest local field in magnitude to the largest one exceeds a threshold, i.e.,

$$\frac{\min_i |h_i|}{\max_i |h_i|} \geq \epsilon_l. \tag{10}$$

Similarly, we wish to bound the ratio of the smallest coupler to the largest one:

$$\frac{\min_{i,j} |J_{ij}|}{\max_{i,j} |J_{ij}|} \geq \epsilon_c. \tag{11}$$

Alternatively, one may desire to have the local fields biases and couplings' strengths well-separated to avoid the effect of noise, or in other words, having

$$\min_{i,j} |h_i - h_j| \geq \epsilon_l,$$

and

$$\min_{\substack{i,j \\ l,k}} |J_{ij} - J_{lk}| \geq \epsilon_c,$$

could be desirable. In this paper, however, we focus merely on inequalities (10) and (11).

Suppose we have a quadratic function

$$f(x) = x^t Q x + q^t x, \tag{12}$$

where Q is symmetric and the domain of $f(x)$ is $x = [x_1, x_2, \ldots, x_n]^t$ for $x_i \in \{0,\ 1,\ \ldots,\ \kappa^{x_i}\}$. Note that this is the function which we aim to minimize in (UIQP). Also, note that since $x_i x_j = x_j x_i$ for two integers x_i and x_j, we may substitute Q with $\frac{1}{2}(Q + Q^t)$ when Q is not symmetric. Of course, everything presented here is under the assumption that the coefficients of the problem (before any encoding) respect inequalities (10) and (11), i.e., unary encoding satisfies them. Let us define the encoding matrix denoted by C as

$$C = \begin{bmatrix} c_1^{x_1} & c_2^{x_1} & \ldots & c_{d^{x_1}}^{x_1} & 0 & 0 & \ldots & 0 & 0 & \ldots & \ldots & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & c_1^{x_2} & c_2^{x_2} & \ldots & c_{d^{x_2}}^{x_2} & 0 & \ldots & \ldots & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & \ldots & \ldots & 0 & c_1^{x_n} & c_2^{x_n} & \ldots & c_{d^{x_n}}^{x_n} \end{bmatrix}, \tag{13}$$

and the vector of spin variables as

$$s = \begin{bmatrix} s_1^{x_1} & s_2^{x_1} & \ldots & s_{d^{x_1}}^{x_1} & s_1^{x_2} & s_2^{x_2} & \ldots & s_{d^{x_2}}^{x_2} & \ldots & \ldots & s_1^{x_n} & s_2^{x_n} & \ldots & s_{d^{x_n}}^{x_n} \end{bmatrix}^t. \tag{14}$$

Using (9), we conclude that the encoding of integer variables, x, into spin variables, s, refers to the following substitution:

$$x = \frac{1}{2}(\kappa + Cs), \tag{15}$$

where $\kappa = [\kappa^{x_1},\ \kappa^{x_2},\ \ldots,\ \kappa^{x_n}]^t$ is the vector of all of the upper bounds on the integer variables. Using this substitution in (12), we get

$$f(s) = \frac{1}{4} s^t \left(C^t Q C - \Delta(\delta(C^t Q C)) \right) s + \frac{1}{2} \left(C^t Q \kappa + C^t q \right)^t s + \frac{1}{4} \left(\kappa^t Q \kappa + e^t \Delta(\delta(C^t Q C)) e + 2 q^t \kappa \right). \tag{16}$$

We are excluding the diagonal entries of $C^t Q C$ in the quadratic term because they correspond to the square of spin variables, i.e., s_i^2, and s_i^2 is a constant equal to one. The local field bias corresponding to each spin variable and the coupling strength corresponding to each pair of spin variables, derived in (16), is summarized below:

$$s_j^{x_i} : \frac{1}{2} \left[Q\kappa + q \right]_i c_j^{x_i} = \frac{1}{2} \left(q_i + \sum_{k=1}^{n} Q_{ik} \kappa_k \right) c_j^{x_i}, \tag{17}$$

$$s_k^{x_i} s_l^{x_i} : \frac{\left(Q_{ii} c_k^{x_i} c_l^{x_i} \right)}{2} \quad \text{for} \quad k, l \in \{1, \ldots, d^{x_i}\} \quad \text{and} \quad k < l, \tag{18}$$

$$s_k^{x_i} s_l^{x_j} : \frac{\left(Q_{ij} c_k^{x_i} c_l^{x_j} \right)}{2} \quad \text{for} \quad k \in \{1, \ldots, d^{x_i}\},\ l \in \{1, \ldots, d^{x_j}\}, \quad \text{and} \quad i, j \in \{1, \ldots, n\} : i < j, \tag{19}$$

where $[Q\kappa + q]_i$ in the first line denote the i-th entry of vector $Q\kappa + q$, as expected, and (18) and (19) incorporate the fact that $s_k^{x_i} s_l^{x_i} = s_l^{x_i} s_k^{x_i}$ and $s_k^{x_i} s_l^{x_j} = s_l^{x_j} s_k^{x_i}$.

Since all of the local fields biases and couplings' strengths share the $\frac{1}{2}$ factor and after rescaling our concern is their ratio, we will drop the $\frac{1}{2}$ factor for our calculations in the rest of this section. From the previous section and the derivation of the bounded-coefficient encoding, we conclude that the smallest coefficient used in the encoding is 1; hence, the minimum absolute value of the local fields biases is

$$m_l = \min_i \{\ |[Q\kappa + q]_i|\ \},$$

and the minimum absolute value of couplings' strengths is

$$m_c = \min_{i,j} \{\ |Q_{ii}|, |Q_{ij}|\ \}.$$

The maximum absolute value of the local fields biases among all spin variables corresponding to an integer variable x_i occurs at μ^{x_i}—to be found—for each $i \in \{1, \ldots, n\}$. Therefore, by (10), we wish to have:

$$\frac{m_l}{|\ [Q\kappa + q]_i\ | \mu^{x_i}} \geq \epsilon_l. \tag{20}$$

Similarly, (11) enforces the following conditions for the couplers:

$$\frac{m_c}{|Q_{ii}| (\mu^{x_i})^2} \geq \epsilon_c, \text{ and } \frac{m_c}{|Q_{ij}| \mu^{x_i} \mu^{x_j}} \geq \epsilon_c. \tag{21}$$

Using (20) and (21), we wish to obtain μ^{x_i}'s that satisfy the following set of inequalities:

$$\mu^{x_i} \leq \frac{m_l}{|[Q\kappa + q]_i|\,\epsilon_l} \tag{22}$$

$$\mu^{x_i} \leq \sqrt{\frac{m_c}{|Q_{ii}|\epsilon_c}}, \tag{23}$$

$$\mu^{x_i}\mu^{x_j} \leq \frac{m_c}{|Q_{ij}|\epsilon_c}. \tag{24}$$

We may solve a feasibility problem to find a solution to the above set of inequalities, i.e., an optimization or auxiliary objective function along with these inequalities. A well-justified objective function could be max $\min_i\{\mu^{x_i}\}$. Solving such problems is normally costly because (24) is non-convex and μ^{x_i}'s are required to be integers. Alternatively, we present an algorithm below that heuristically finds μ^{x_i}'s.

Note that any set of μ^{x_i}'s satisfying the above inequalities will guarantee that (10) and (11) hold. Our proposed algorithm for finding μ^{x_i}'s first initializes μ^{x_i}'s using inequalities (22) and (23). If (24) is satisfied for all i and j at this step, it terminates. Otherwise, it greedily decrease μ^{x_i} or μ^{x_j} for an i and j pair for which the failure happened. In this process, we take $\frac{\kappa^{x_i}}{\mu^{x_i}}$ as an estimate on the width of the encoding, so when we want to decrease either μ^{x_i} or μ^{x_j}, we choose the one that gives a lower combined width. See Algorithm 2 for a formal presentation of this algorithm.

Algorithm 2 Finding the Upper Bounds on the Coefficients of the Encoding

Inputs:
$\kappa,\ q,\ Q,\ \epsilon_l,\ \epsilon_c$
compute $Q\kappa + q$
set $m_l = \min_i \{\,|\,[Q\kappa+q]_i\,|\,\}$, and $m_c = \min_{i,j}\{|Q_{ii}|, |Q_{ij}|\}$
Output:
μ^{x_i} for $i = 1, 2, \ldots, n$

initialize $\mu^{x_i} = \left\lfloor \min\left\{ \frac{m_l}{|\,[Q\kappa+q]_i\,|\epsilon_l},\ \sqrt{\frac{m_c}{|Q_{ii}|\epsilon_c}} \right\} \right\rfloor$

while any $\left(\mu^{x_i}\mu^{x_j} > \frac{m_c}{|Q_{ij}|\epsilon_c}\right)$

let $i, j = \arg\max_{i,j} \left\{ \mu^{x_i}\mu^{x_j} - \frac{m_c}{|Q_{ij}|\epsilon_c} \right\}$

let $\xi_i = \frac{\kappa^{x_i}}{\mu^{x_i}-1} + \frac{\kappa^{x_j}}{\mu^{x_j}}$ and $\xi_j = \frac{\kappa^{x_i}}{\mu^{x_i}} + \frac{\kappa^{x_j}}{\mu^{x_j}-1}$

if $\xi_i < \xi_j$
$\mu^{x_i} = \mu^{x_i} - 1$
else
$\mu^{x_j} = \mu^{x_j} - 1$

It is worth mentioning that if we wish to keep μ^{x_i}'s for all integer variables equal, we may use the value

$$\mu = \min_i \{\mu^{x_i}\}. \tag{25}$$

Moreover, if in (12) the quadratic terms were missing, i.e., $f(x) = q^t x$, we would get

$$f(s) = \frac{1}{2} q^t (\kappa + Cs) = \frac{1}{2} q^t \kappa + \frac{1}{2} \left(C^t q\right)^t s. \tag{26}$$

Ignoring the $\frac{1}{2}$ factor (since it is common for all spin variables), we get $m_l = \min_i \{|q_i|\}$, and our ratio condition reduces to

$$\frac{m_l}{|q_i| \mu^{x_i}} \geq \epsilon_l; \tag{27}$$

therefore,

$$\mu^{x_i} = \left\lfloor \frac{m_l}{|q_i| \epsilon_l} \right\rfloor. \tag{28}$$

Finally, we would like to point out that unlike QA, in several heuristic methods, variables tend to be binary, i.e., in the $\{0, 1\}$ domain referred to as binary. In Appendix A, we present a modification of the method in this section that could be employed in this case.

4 Numerical Experiment

In this section, we test the bounded-coefficient encoding. We compare binary and bounded-coefficient encodings on ten randomly generated instances. To diversify our instances, however, we use several procedures to generate them. Let us define the set $U_\alpha = \{0, \pm 1, \pm 2, \ldots, \pm\alpha\}$. In all of our instances, we have five integer variables, i.e., $n = 5$; the upper-bound on all of the integer variables is 50, i.e., $\kappa^{x_i} = 50$ for $i = 1, \ldots, 5$; and the matrix Q in the quadratic term has a sparsity around 50%. In half of our instances, the generated Q is positive definite, and in the rest it is indefinite; these two categories are referred to as convex and non-convex instances, respectively.

For the convex instances, entries of matrix Q are initially drawn from U_2; then, λI is added to Q to make Q positive definite. Our choice of λ is $\lceil |\min\{\lambda_{\min}, 0\}| + r \rceil$, where $\lambda_{\min}$ is the minimum eigenvalue of Q and r is a random number between 0 and 1. A feasible sparse integer vector, i.e., x^*, where $x_i^* \in \{0, 1, \ldots, 50\}$ if $x_i^* \neq 0$, is then generated, and we set $q = -2Qx^*$. Note that by the fact that Q is positive definite and by our choice of q, x^* is the unique optimal solution to the generated instance of (UIQP). These instances are shown with the prefix convex in Tables 1 and 2. In the non-convex instances, entries of Q and q are from separate U_α's. We denote these instances with pairs (U_Q, U_q), where U_Q and U_q are distributions for the quadratic terms (Q) and linear terms (q), respectively. Our data sets are (U_2, U_{200}), (U_5, U_{200}), (U_5, U_{10}), (U_5, U_{100}), and (U_{10}, U_0).

As mentioned earlier, we take *resilience* as the measure of success. More specifically, we hypothesized that the bounded-coefficient encoding is a technique for representing a UIQP problem as a UBQP problem that is more robust against noise, and resilience directly measures the robustness of an instance of UBQP against noise. In [21], the resilience R of an instance is defined as

$$R = \frac{n_{\text{same}}}{n_{\text{trial}}}, \tag{29}$$

where n_{same} is the number of times, among all n_{trial}, that the original ground state does not change with random noise perturbations. In other words, n_{trial} different noise matrices with entries drawn from the normal distribution $\mathcal{N}(0, \epsilon)$ are generated; each is added to a scaled Ising model

Table 1 Resilience with Bounded-Coefficient Encoding.

Ins. \ ϵ	0.001	0.002	0.003	0.004	0.005	0.006	0.007	0.008	0.009	0.01
convex-1	1.0	1.0	0.6	0.4	0.8	0.5	0.2	0.1	0.1	0.0
convex-2	0.7	0.5	0.4	0.1	0.2	0.0	0.0	0.2	0.0	0.0
convex-3	1.0	0.7	0.6	0.5	0.2	0.1	0.0	0.1	0.0	0.0
convex-4	1.0	0.9	0.6	0.3	0.2	0.1	0.1	0.0	0.1	0.0
convex-5	1.0	1.0	0.5	0.6	0.3	0.4	0.2	0.2	0.4	0.1
(U_2, U_{200})	1.0	1.0	1.0	1.0	0.9	1.0	0.9	0.8	1.0	0.8
(U_5, U_{200})	1.0	0.9	0.9	0.9	0.8	0.4	0.7	0.8	0.5	0.6
(U_{10}, U_0)	1.0	0.8	0.4	0.3	0.4	0.1	0.4	0.2	0.1	0.2
(U_5, U_{10})	1.0	1.0	1.0	0.9	0.7	0.9	0.4	0.4	0.6	0.7
(U_5, U_{100})	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.7	0.6
average	0.97	0.88	0.7	0.6	0.55	0.45	0.39	0.35	0.35	0.3

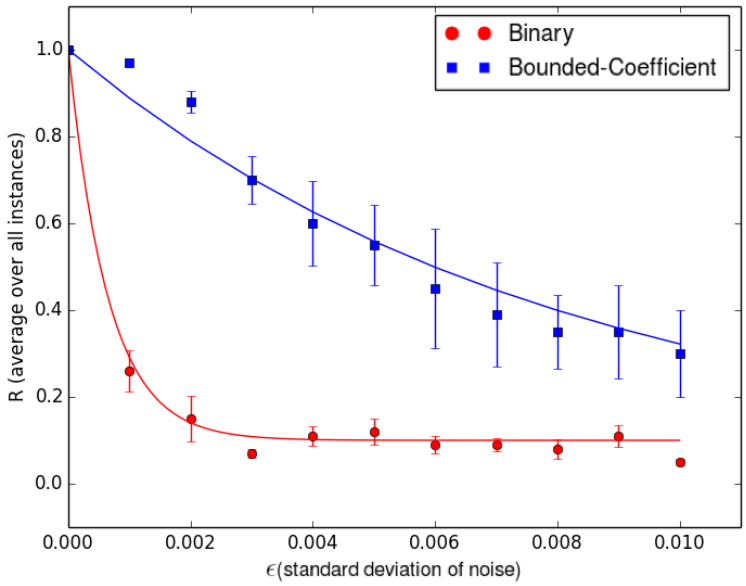
Table 2 Resilience with Binary Encoding.

Ins. \ ϵ	0.001	0.002	0.003	0.004	0.005	0.006	0.007	0.008	0.009	0.01
convex-1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
convex-2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
convex-3	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
convex-4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
convex-5	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
(U_2, U_{200})	0.6	0.4	0.2	0.1	0.3	0.1	0.3	0.1	0.4	0.2
(U_5, U_{200})	0.5	0.7	0.3	0.4	0.5	0.3	0.2	0.5	0.2	0.2
(U_{10}, U_0)	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
(U_5, U_{10})	0.3	0.1	0.1	0.1	0.1	0.0	0.1	0.1	0.1	0.0
(U_5, U_{100})	0.5	0.3	0.1	0.4	0.3	0.4	0.3	0.1	0.4	0.1
average	0.26	0.15	0.07	0.11	0.12	0.09	0.09	0.08	0.11	0.05

and the perturbed Ising model is solved exactly. n_{same}, then, refers to the number of times that the perturbed Ising model returns the same ground state as the unperturbed one. The number of trials, n_{trial}, in our experiment is set to 10. We take ϵ_l and ϵ_c to be 0.01, and we scale Ising models to $J \in [-1, 1]$ before adding the noise. After finding the upper bound on the coefficients of the encoding, with Algorithm 2 and $\epsilon_l = \epsilon_c = 0.01$, we obtain $\text{Ising}_{\text{bounded}}$ through the bounded-coefficient encoding. Also, by using the binary encoding, we derive $\text{Ising}_{\text{binary}}$. Then, we measure the resilience of each of these Ising models at $\epsilon \in \{0.001, 0.002, \ldots, 0.01\}$. The resilience of $\text{Ising}_{\text{bounded}}$ and $\text{Ising}_{\text{binary}}$ are summarized in Tables 1 and 2, respectively. The bounded-coefficient encoding has significantly outperformed the binary encoding; $\text{Ising}_{\text{bounded}}$ is five times more resilient to noise than $\text{Ising}_{\text{binary}}$, on average. Also, in most of our test cases, the convex instances and (U_{10}, U_0), the resilience is almost zero with the binary encoding, except for a few exceptions at $\epsilon = 0.001$, whereas for bounded-coefficient encoding, the resilience stays above zero and is considerably large for $\epsilon \leq 0.005$. In order to visualize the difference between the two encodings, we plot the average resilience (over all instances) with respect to ϵ, the standard deviation of the added noise, in Figure 1. It is obvious from this plot that the resilience with binary encoding stays marginally above 0, but for the bounded-coefficient encoding it decreases at a slower rate and remains considerably above 0.

Note that in [21], it is suggested to set $\epsilon = \frac{2}{J_{\max}}$, where $J_{\max}$ is the largest J_{ij}. Here, our approach slightly differs. We assume that we know the precision of the machine, and we derive the upper

Fig. 1 Resilience Averaged over All of the Instances.



bound on the coefficients of our encoding to accommodate this restriction. The precision of the current D-Wave machine is in order of 10^{-2}, which motivates our choice of $\epsilon = 10^{-2}$.

5 Conclusion and Discussion

In this paper, we presented an encoding to represent an unconstrained integer quadratic programming problem as an Ising model. To deal with the low precision of quantum annealers, we suggested bounding the value of the coefficients in the encoding. This restricts the range of the local fields biases and couplings' strengths in the derived Ising model, thereby creating Ising models that are more robust against noise after scaling. Resilience is used as a metric for the robustness of the Ising models. We compared bounded-coefficient encoding with the binary encoding. We infer from our results that bounding the coefficients of the encoding, as in the bounded-coefficient encoding, significantly improves the resilience of the model. The drawbacks are that the size of the derived Ising model is larger and it introduces redundancy.

In our proposed technique, we forced the ratios of the minimum absolute value of the local fields biases and the couplings' strengths to their respective maximums exceed a certain threshold tolerance that is correlated or equal to the quantum annealer's precision. However, all of our calculations are embedding-free, i.e., they ignore the Chimera graph structure. In theory, having an embedding with equal chain length for all variables and equal chain connectivity for any present quadratic terms will respect our calculation; however, finding such an embedding is nontrivial. Notice that, after the encoding, all spin variables corresponding to an integer variable are connected; additionally, all spin variables corresponding to x_i and x_j are connected if the term $x_i x_j$ appears in the integer for-

mulation; so, the underlying graph of the Ising model is quite dense. Even for our small instances, the resulting Ising models either exceed the size that can be embedded on the current chip, or the chains' lengths differ in orders of magnitude (e.g., minimum and maximum chain lengths in orders 1 and 10, respectively). Our algorithm can be tested on the future generation of the quantum annealers with improved connectivity.

According to our results in the previous section, using $\epsilon_l = \epsilon_c = 0.01$ to find the upper bound on the coefficients of the encoding, the derived Ising model stays robust against noise, with an average resilience above 0.5, and minimum resilience above 0.1 for $\epsilon = 0.005$, i.e., $\frac{\epsilon_c}{2}$. This could suggest using $2\epsilon_{QA}$ as ϵ_l and ϵ_c for finding the upper bounds on the coefficients of the encoding, where ϵ_{QA} is the precision of the quantum annealer. This, however, requires more experimentation and could be an avenue for future study.

A (Integer-to-Binary Encoding)

In this section, we aim to find the upper bounds on the coefficients of the integer encoding when we reduce an unconstrained integer quadratic programming (UIQP) problem to an unconstrained binary quadratic programming (UBQP) problem in the $\{0, 1\}$ domain. Similar to what we discussed earlier, we assume that our UIQP problem has the form

$$\begin{aligned} &\min\ x^T Q x + q^t x, \\ &\text{s.t.}\ \ x_i \in \{0, 1, 2, \ldots, \kappa^{x_i}\} \quad \text{for } i = \{1, 2, \ldots, n\}, \end{aligned}$$

with Q being symmetric.

We aim to represent the above problem as $f(y)$ by substituting $x = Cy$, where C is the encoding matrix we had earlier, i.e.,

$$C = \begin{bmatrix} c_1^{x_1} & c_2^{x_1} & \ldots & c_{d^{x_1}}^{x_1} & 0 & 0 & \ldots & 0 & 0 \ldots \ldots \ldots & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & c_1^{x_2} & c_2^{x_2} & \ldots & c_{d^{x_2}}^{x_2} & 0 \ldots \ldots \ldots & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \ddots \ddots \ddots \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 \ldots \ldots \ldots & 0 & c_1^{x_n} & c_2^{x_n} & \ldots & c_{d^{x_n}}^{x_n} \end{bmatrix}, \tag{30}$$

and

$$y = \begin{bmatrix} y_1^{x_1} & y_2^{x_1} & \ldots & y_{d^{x_1}}^{x_1} & y_1^{x_2} & y_2^{x_2} & \ldots & y_{d^{x_2}}^{x_2} & \ldots \ldots \ldots & y_1^{x_n} & y_2^{x_n} & \ldots & y_{d^{x_n}}^{x_n} \end{bmatrix}^t \in \mathbb{B}^{\sum_{i=1}^{n} d^{x_i}}. \tag{31}$$

After the substitution for x, we get the following equivalent binary formulation:

$$f(y) = y^t (C^t Q C) y + (C^t q)^t y. \tag{32}$$

Unlike the spin variables for which the diagonal of $C^t Q C$ became constant, the diagonal of $C^t Q C$ is added to the linear term in this scenario since $y_i^2 = y_i$ for $y_i \in \{0, 1\}$. Alternatively, we can represent $f(y)$ as

$$f_B = y^t Q_B y, \tag{33}$$

where

$$Q_B = \left[\begin{array}{c|c|c|c} Q_{11}[c^{x_1}][c^{x_1}]^t + \Delta(q_1[c^{x_1}]) & Q_{12}[c^{x_1}][c^{x_2}]^t & \cdots & Q_{1n}[c^{x_1}][c^{x_n}]^t \\ Q_{21}[c^{x_2}][c^{x_1}]^t & Q_{22}[c^{x_2}][c^{x_2}]^t + \Delta(q_2[c^{x_2}]) & \cdots & Q_{2n}[c^{x_2}][c^{x_n}]^t \\ \vdots & \vdots & \vdots & \vdots \\ Q_{n1}[c^{x_n}][c^{x_1}]^t & Q_{n2}[c^{x_n}][c^{x_2}]^t & \cdots & Q_{nn}[c^{x_n}][c^{x_n}]^t + \Delta(q_n[c^{x_n}]) \end{array}\right]. \tag{34}$$

The diagonal entries of Q_B are the linear terms, i.e., coefficients of the variables $y_j^{x_i}$. We use linear and quadratic terms for binary model $f(y)$ instead of local fields and couplers, respectively. We also refer to inequalities (10) and (11) as the ratio inequalities for linear and quadratic terms, respectively. Notice that in these ratios, minimum and maximum coefficients are measured in magnitude, so in our discussion that follows, we consider merely the magnitude of the coefficients.

Considering the fact that $y_k^{x_i} y_l^{x_i} = y_l^{x_i} y_k^{x_i}$ and $y_k^{x_i} y_l^{x_j} = y_l^{x_j} y_k^{x_i}$, the coefficients of linear and quadratic terms are listed below:

$$y_j^{x_i} : \left(Q_{ii} \left(c_j^{x_i} \right)^2 + q_i c_j^{x_i} \right), \tag{35}$$

$$y_k^{x_i} y_l^{x_i} : 2 \left(Q_{ii} c_k^{x_i} c_l^{x_i} \right) \quad \text{for} \quad k, l \in \{1, \ldots, d^{x_i}\} \quad \text{and} \quad k < l, \tag{36}$$

$$y_k^{x_i} y_l^{x_j} : 2 \left(Q_{ij} c_k^{x_i} c_l^{x_j} \right) \quad \text{for} \quad k \in \{1, \ldots, d^{x_i}\},\ l \in \{1, \ldots, d^{x_j}\}, \quad \text{and} \quad i, j \in \{1, \ldots, n\} : i < j. \tag{37}$$

The difference between this case and what was presented in Section 3 is that the coefficients of $y_j^{x_i}$ are no longer linear in $c_j^{x_i}$ (compare with (17)); therefore, the smallest coefficient may no longer occur at $c_j^{x_i} = 1$. Notice that $Q_{ii} \left(c_j^{x_i} \right)^2 + q_i c_j^{x_i}$ intersects 0 at $c_j^{x_i} = 0$ and $c_j^{x_i} = \frac{-q_i}{Q_{ii}}$; therefore, if $\frac{-q_i}{Q_{ii}} < 1$, then $Q_{ii} \left(c_j^{x_i} \right)^2 + q_i c_j^{x_i}$ takes its minimum at $c_j^{x_i} = 1$ and is increasing afterwards. When this is the case for all $i = 1, \ldots, n$, slight modification of Algorithm 2 is sufficient to find the μ^{x_i}'s. In the algorithm, the modification is at the initialization step of μ^{x_i}'s. Letting

$$m_l = \min_i \{|Q_{ii} + q_i|\} \quad \text{and} \quad m_c = \min_{i,j} \{|Q_{ii}|, |Q_{ij}|\}, \tag{38}$$

the condition that needs to be satisfied for the linear coefficients is

$$\frac{m_l}{|Q_{ii}|(\mu^{x_i})^2 + \operatorname{sgn}(q_i Q_{ii}) |q_i| \mu^{x_i}} \geq \epsilon_l. \tag{39}$$

Combined with the condition

$$\mu^{x_i} \leq \sqrt{\frac{m_c}{|Q_{ii}| \epsilon_c}}, \tag{40}$$

we need to initialize μ^{x_i} as

$$\mu^{x_i} = \left\lfloor \min \left\{ \tilde{\mu}^{x_i}, \sqrt{\frac{m_c}{|Q_{ii}| \epsilon_c}} \right\} \right\rfloor, \tag{41}$$

where

$$\tilde{\mu}^{x_i} = \frac{-\operatorname{sgn}(q_i Q_{ii}) |q_i| + \sqrt{|q_i|^2 + 4 |Q_{ii}| \frac{m_l}{\epsilon_l}}}{2 |Q_{ii}|}. \tag{42}$$

It is worth mentioning the special cases where $\operatorname{sgn}(Q_{ii}) = \operatorname{sgn}(q_i)$ (resulting in $\frac{-q_i}{Q_{ii}} < 0$), and $Q_{ii} = 0$ or $q_i = 0$ belong to the above category, where $Q_{ii} \left(c_j^{x_i} \right)^2 + q_i c_j^{x_i}$ attains its minimum at 1.

In the general cases where there exists an i such that $\frac{-q_i}{Q_{ii}} \geq 1$, satisfying the ratio condition on the linear terms is more complicated than what we discussed above. Not only might the minimum coefficient no longer occur at 1, but also the maximum could occur at either μ^{x_i} or $\frac{-q_i}{2Q_{ii}}$, where the derivative of the function $h(c_j^{x_i}) = Q_{ii} \left(c_j^{x_i} \right)^2 + q_i c_j^{x_i}$ is zero. Our approach for these general cases is to first derive μ^{x_i}'s that satisfy the ratio condition for the quadratic terms, and then adjust them accordingly to meet the ratio constraint on the linear terms.

Similar to the previous case, the minimum coefficient on the quadratic terms is

$$m_c = \min_{i,j} \{|Q_{ii}|, |Q_{ij}|\}, \tag{43}$$

and the ratio condition on quadratic terms enforces

$$\frac{m_c}{|Q_{ii}|(\mu^{x_i})^2} \geq \epsilon_c, \text{ and } \frac{m_c}{|Q_{ij}| \mu^{x_i} \mu^{x_j}} \geq \epsilon_c, \tag{44}$$

or, equivalently,

$$\mu^{x_i} \leq \sqrt{\frac{m_c}{|C_{ii}|\epsilon_c}}, \tag{45}$$

$$\mu^{x_i}\mu^{x_j} \leq \frac{m_c}{|C_{ij}|\epsilon_c}. \tag{46}$$

We may now initialize μ^{x_i} as $\left\lfloor \sqrt{\frac{m_c}{|Q_{ii}|\epsilon_c}} \right\rfloor$ and use the loop of Algorithm 2 to satisfy (46), i.e., Algorithm 3.

Algorithm 3 Finding μ^{x_i} for the Ratio Condition on Quadratic Terms

Inputs: $\kappa,\ q,\ Q,\ \epsilon_c$
  set $m_c = \min_{i,j}\{|Q_{ii}|, |Q_{ij}|\}$
Output:
  μ^{x_i} for $i = 1, 2, \ldots, n$

initialize $\mu^{x_i} = \left\lfloor \sqrt{\frac{m_c}{|Q_{ii}|\epsilon_c}} \right\rfloor$

while any $\left(\mu^{x_i}\mu^{x_j} > \frac{m_c}{|Q_{ij}|\epsilon_c}\right)$

  let $i, j = \arg\max_{i,j} \left\{\mu^{x_i}\mu^{x_j} - \frac{m_c}{|Q_{ij}|\epsilon_c}\right\}$

  let $\xi_i = \frac{\kappa^{x_i}}{\mu^{x_i}-1} + \frac{\kappa^{x_j}}{\mu^{x_j}}$ and $\xi_j = \frac{\kappa^{x_i}}{\mu^{x_i}} + \frac{\kappa^{x_j}}{\mu^{x_j}-1}$

  if $\xi_i < \xi_j$
    $\mu^{x_i} = \mu^{x_i} - 1$
  else
    $\mu^{x_j} = \mu^{x_j} - 1$

After μ^{x_i}'s are calculated to satisfy the conditions (45) and (46), we need to check the ratio condition for the linear terms. Although some of the integer values $\{1, 2, \ldots, \mu^{x_i}\}$ may not appear in our encoding, knowing which integers will appear prior to finding μ^{x_i}'s is not trivial. The algorithm presented below guarantees that the ratio condition on the linear terms holds if any of these integer values appear in the encoding. Let us categorize the indices based on where the minimum and maximum linear coefficients occur; we introduce the following sets of indices for this purpose:

$$\begin{aligned}
\mathcal{I}_0^m &= \left\{i : \text{sgn}(Q_{ii}) \neq \text{sgn}(q_i) \text{ and } \frac{-q_i}{Q_{ii}} = 1\right\},\\
\mathcal{I}_1^m &= \{i : \text{sgn}(Q_{ii}) = \text{sgn}(q_i)\} \cup\\
&\quad \left\{i : \text{sgn}(Q_{ii}) \neq \text{sgn}(q_i) \text{ and } 0 \leq \frac{-q_i}{Q_{ii}} < 1\right\} \cup\\
&\quad \left\{i : \text{sgn}(Q_{ii}) \neq \text{sgn}(q_i) \text{ but } \mu^{x_i} < \left\lfloor \frac{-q_i}{Q_{ii}} \right\rfloor\right\} \cup\\
&\quad \left\{i : \text{sgn}(Q_{ii}) \neq \text{sgn}(q_i) \text{ but } \frac{-q_i}{Q_{ii}} \text{ is an integer greater than } 1\right\},\\
\mathcal{I}_2^m &= \left\{i : \text{sgn}(Q_{ii}) \neq \text{sgn}(q_i) \text{ and } \frac{-q_i}{Q_{ii}} > 1 \text{ and } \frac{-q_i}{Q_{ii}} \text{ is not an integer and } \mu^{x_i} \geq \left\lfloor \frac{-q_i}{Q_{ii}} \right\rfloor\right\}.
\end{aligned} \tag{47}$$

and

$$\begin{aligned}\mathcal{I}_1^M &= \{i : \operatorname{sgn}(Q_{ii}) = \operatorname{sgn}(q_i)\} \cup \\ &\quad \left\{i : \operatorname{sgn}(Q_{ii}) \neq \operatorname{sgn}(q_i) \text{ and } 0 \leq \frac{-q_i}{2Q_{ii}} < \frac{1}{2}\right\} \cup \\ &\quad \left\{i : \operatorname{sgn}(Q_{ii}) \neq \operatorname{sgn}(q_i) \text{ but } \mu^{x_i} \leq \operatorname{round}\left(\frac{-q_i}{2Q_{ii}}\right)\right\}, \\ \mathcal{I}_2^M &= \left\{i : \operatorname{sgn}(Q_{ii}) \neq \operatorname{sgn}(q_i) \text{ and } \frac{-q_i}{2Q_{ii}} \geq \frac{1}{2} \text{ and } \mu^{x_i} > \operatorname{round}\left(\frac{-q_i}{2Q_{ii}}\right)\right\}. \end{aligned} \tag{48}$$

Note that μ^{x_i}'s returned by Algorithm 3 are integers. The sets with m and M superscripts are formed to facilitate computing minimum and maximum linear coefficients, respectively. For indices $i \in \mathcal{I}_0^m$, the minimum linear term happens at 2; for indices in $\mathcal{I}_1^m$, it happens at 1, and for indices in $\mathcal{I}_2^m$, it happens at one of the two integers closest to $\frac{-q_i}{Q_{ii}}$. Similarly, for the indices in $\mathcal{I}_1^M$, the maximum linear coefficient happens at μ^{x_i}, whereas for indices in $\mathcal{I}_2^M$, it happens at either μ^{x_i} or at the closest integer to $\frac{-q_i}{2Q_{ii}}$, i.e., round $\left(\frac{-q_i}{2Q_{ii}}\right)$.

After categorizing the indices in the above sets, we may form the arrays v^m and v^M, which represent the minimum and maximum coefficients of the linear terms for each variable, respectively:

$$v_i^m = \begin{cases} (|4Q_{ii} + 2q_i|, 2, i) & \text{if } i \in \mathcal{I}_0^m \\ (|Q_{ii} + q_i|, 1, i) & \text{if } i \in \mathcal{I}_1^m \\ \left(\min_{x \in \mathcal{S}_i} |Q_{ii}x^2 + q_i x|, \arg\min_{x \in \mathcal{S}_i} |Q_{ii}x^2 + q_i x|, i\right) : \mathcal{S}_i = \left\{x \in \left\{\left\lfloor \frac{-q_i}{Q_{ii}} \right\rfloor, \left\lceil \frac{-q_i}{Q_{ii}} \right\rceil\right\} : x \leq \mu^{x_i}\right\} & \text{if } i \in \mathcal{I}_2^m \end{cases} \tag{49}$$

$$v_i^M = \begin{cases} \left(|Q_{ii}(\mu^{x_i})^2 + q_i\mu^{x_i}|, \mu^{x_i}, i\right) & \text{if } i \in \mathcal{I}_1^M \\ \left(\max_{x \in \mathcal{S}_i} |Q_{ii}x^2 + q_i x|, \arg\max_{x \in \mathcal{S}_i} |Q_{ii}x^2 + q_i x|, i\right) : \mathcal{S}_i = \left\{\operatorname{round}\left(\frac{-q_i}{2Q_{ii}}\right), \mu^{x_i}\right\} & \text{if } i \in \mathcal{I}_2^M \end{cases} \tag{50}$$

We then sort entries of v^m (on the first entry) in increasing order; assume it results in vector $\bar{v}^m$, so $\bar{v}_1^m \leq \bar{v}_2^m \leq \cdots \leq \bar{v}_n^m$; and sort v^M in decreasing order; i.e., $\bar{v}^M$ such that $\bar{v}_1^M \geq \bar{v}_2^M \geq \cdots \geq \bar{v}_n^M$. We then whether $\frac{\bar{v}_1^m}{\bar{v}_1^M} \geq \epsilon_l$. If this inequality holds, then we have obtained our set of μ^{x_i}'s; otherwise, we have the following options for improvement:

- $\bar{v}_1^m$ happens at an $i \in \mathcal{I}_2^m$, where we can either change μ^{x_i} to increase the minimum coefficient, or change μ^{x_i} to decrease the maximum coefficient;
- $\bar{v}_1^m$ happens at $i \in \mathcal{I}_1^m \cup \mathcal{I}_0^m$, in which case we can only change μ^{x_i} such that the maximum coefficient decreases.

In the first scenario where we have the option to both increase the minimum or decrease the maximum coefficient, we use a greedy approach to make the decision. In other words, if the minimum coefficient is improved, it will be $v_{2,1}^m$ in the next iterate; similarly, the maximum coefficient will $\bar{v}_{2,1}^M$, if updated. In our approach, having the interval $[\bar{v}_{1,1}^m, \bar{v}_{1,1}^M]$ for the coefficients, we wish to update it to either $[\bar{v}_{2,1}^m, \bar{v}_{1,1}^M]$ or $[\bar{v}_{1,1}^m, \bar{v}_{2,1}^M]$. These two intervals will be $[\frac{\bar{v}_{2,1}^m}{\bar{v}_{1,1}^M}, 1]$ or $[\frac{\bar{v}_{1,1}^m}{\bar{v}_{2,1}^M}, 1]$, respectively, after the rescaling. We choose the option that gives us better lower bound, i.e., if

$$\frac{\bar{v}_{2,1}^m}{\bar{v}_{1,1}^M} > \frac{v_{1,1}^m}{\bar{v}_{2,1}^M} \;\equiv\; \bar{v}_{2,1}^m \bar{v}_{2,1}^M > \bar{v}_{1,1}^m \bar{v}_{1,1}^M \,,$$

we attempt to improve the lower bound, thus decreasing μ^{x_i} for $i = \bar{v}_{1,3}^m$. A formal presentation of what we have discussed is summarized in Algorithm 4.

Acknowledgements

We are thankful to Helmut G. Katzgraber and Gili Rosenberg for insightful discussions and helpful feedback; and Marko Bucyk for editing the manuscript.

Algorithm 4 Adjusting μ^{x_i} for Ratio Condition on Linear Terms

Inputs:
$q_i,\ Q_{ii},\ \epsilon_l$, and μ^{x_i} that is the output of Algorithm 3
Output:
μ^{x_i} for $i = 1, 2, \ldots, n$

initialize $\mathcal{I}_0^m, \mathcal{I}_1^m, \mathcal{I}_2^m, \mathcal{I}_1^M,$ and $\mathcal{I}_2^M$, using equations (47) and (48).

while 1
form v^m and v^M using equation (49) and (50).
sort v^m increasingly, and v^M decreasingly (on the first entry) to get $\bar{v}^m$ and $\bar{v}^M$

if $\frac{\bar{v}_{11}^m}{\bar{v}_{11}^M} \geq \epsilon_\ell$ **return**
else
if $k = \bar{v}_{1,3}^m \in \mathcal{I}_2^m$ AND $\bar{v}_{2,1}^m \bar{v}_{2,1}^M > \bar{v}_{1,1}^m \bar{v}_{1,1}^M$
set $\mu^{x_k} = \mu^{x_k} - 1$ (can also be $\bar{v}_{12}^m - 1$) ,
if $\mu^{x_k} < \left\lfloor \frac{-q_k}{Q_{kk}} \right\rfloor$ (OR $\bar{v}_{1,2}^m = \left\lfloor \frac{-q_k}{Q_{kk}} \right\rfloor$)
$\mathcal{I}_2^m = \mathcal{I}_2^m \setminus \{k\}$ and $\mathcal{I}_1^m = \mathcal{I}_1^m \cup \{k\}$
else
$k = \bar{v}_{1,3}^M$
if $k \in \mathcal{I}_2^M$ AND $\bar{v}_{1,2}^M = \text{round}\left(\frac{-q_k}{2Q_{kk}}\right)$
set $\mu^{x_k} = \text{round}\left(\frac{-q_k}{2Q_{kk}}\right) - 1$, $\mathcal{I}_2^M = \mathcal{I}_2^M \setminus \{k\}$ and $\mathcal{I}_1^M = \mathcal{I}_1^M \cup \{k\}$
else
set $\mu^{x_k} = \mu^{x_k} - 1$

References

1. T. Albash and D. A. Lidar. Adiabatic quantum computing. Nov. 2016 arXiv:quant-ph/1611.04471.
2. T. Albash, W. Vinci, A. Mishra, P. A. Warburton, and D. A. Lidar. Consistency tests of classical and quantum models for a quantum annealer. *Phys. Rev. A*, 91:042314, Apr. 2015. doi:10.1103/PhysRevA.91.042314.
3. Mohammad Amin, Neil Dickson, and Peter Smith. Adiabatic quantum optimization with qudits. *Quantum Information Processing*, 12(4):1819 – 1829, 2013. doi:10.1007/s11128-012-0480-x.
4. P. I. Bunyk, E. M. Hoskinson, M. W. Johnson, E. Tolkacheva, F. Altomare, A. J. Berkley, R. Harris, J. P. Hilton, T. Lanting, A. J. Przybysz, and J. Whittaker. Architectural considerations in the design of a superconducting quantum annealing processor. *IEEE Transactions on Applied Superconductivity*, 24(4):1–10, Aug. 2014. doi:10.1109/TASC.2014.2318294.
5. J. Cai, W. G. Macready, and A. Roy. A practical heuristic for finding graph minors. Jun. 2014. arXiv:1406.2741.
6. E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292(5516):472–476, 2001. doi:10.1126/science.1057726.
7. H. Ishikawa. Transformation of general binary MRF minimization to the first-order case. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(6):1234–1249, Jun. 2011. doi:10.1109/TPAMI.2010.91.
8. M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 05 2011. doi:10.1038/nature10012.
9. S. Karimi and P. Ronagh. A subgradient approach for constrained binary programming via quantum adiabatic evolution. Jan. 2017. arXiv:1605.09462.

10. H. G. Katzgraber, F. Hamze, Z. Zhu, A. J. Ochoa, and H. Munoz-Bauza. Seeking quantum speedup through spin glasses: The good, the bad, and the ugly. *Phys. Rev. X*, 5:031026, Sep. 2015. doi:10.1103/PhysRevX.5.031026.
11. L. W. Lee, H. G. Katzgraber, and A. P. Young. Critical behavior of the three- and ten-state short-range potts glass: A monte carlo study. *Phys. Rev. B*, 74:104416, Sep 2006. doi:10.1103/PhysRevB.74.104416.
12. Salvatore Mandrà, Zheng Zhu, and Helmut G. Katzgraber. Exponentially biased ground-state sampling of quantum annealing machines with transverse-field driving Hamiltonians. *Phys. Rev. Lett.*, 118:070502, Feb 2017. doi:10.1103/PhysRevLett.118.070502.
13. R. Mansini, W. Ogryczak, and M. G. Speranza. *Linear Models for Portfolio Optimization*. Springer International Publishing: 19–45, 2015. doi:10.1007/978-3-319-18482-1_2.
14. Yoshiki Matsuda, Hidetoshi Nishimori, and Helmut G Katzgraber. Ground-state statistics from annealing algorithms: quantum versus classical approaches. *New Journal of Physics*, 11(7):073021, 2009. doi:10.1088/1367-2630/11/7/073021.
15. C. C. McGeoch and C. Wang. Experimental evaluation of an adiabiatic quantum system for combinatorial optimization. In *Proceedings of the ACM International Conference on Computing Frontiers*, CF '13, pages 23:1–23:11, New York, NY, USA, 2013. ACM. doi:10.1145/2482767.2482797.
16. E. G. Rieffel, D. Venturelli, B. O'Gorman, M. B. Do, E. M. Prystay, and V. N. Smelyanskiy. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing*, 14:1–36, Jan. 2015. doi:10.1007/s11128-014-0892-x.
17. G. Rosenberg, P. Haghnegahdar, P. Goddard, P. Carr, K. Wu, and M. L. de Prado. Solving the optimal trading trajectory problem using a quantum annealer. *IEEE Journal of Selected Topics in Signal Processing*, 10(6):1053–1060, Sep. 2016. doi:10.1109/JSTSP.2016.2574703.
18. Giuseppe E Santoro and Erio Tosatti. Optimization using quantum mechanics: quantum annealing through adiabatic evolution. *Journal of Physics A: Mathematical and General*, 39(36):R393, 2006. doi:10.1088/0305-4470/39/36/R01.
19. Tadeusz Sawik. *Scheduling in Supply Chains Using Mixed Integer Programming*. John Wiley & Sons Inc., 2011. doi:10.1002/9781118029114.
20. D. Venturelli, D. J. J. Marchand, and G. Rojo. Quantum annealing implementation of job-shop scheduling. Jun. 2015. arXiv:1506.08479.
21. Z. Zhu, A. J. Ochoa, S. Schnabel, F. Hamze, and H. G. Katzgraber. Best-case performance of quantum annealers on native spin-glass benchmarks: How chaos can affect success probabilities. *Phys. Rev. A*, 93:012317, Jan. 2016. doi:10.1103/PhysRevA.93.012317.
22. K. M. Zick, O. Shehab, and M. French. Experimental quantum annealing: Case study involving the graph isomorphism problem. *Scientific Reports*, 5:11168, Jun. 2015. doi:10.1038/srep11168.