

QUBO Formulations for a System of Linear Equations

Kyungtaek Jun (✉ ktfriends@gmail.com)

Innovation on Quantum and Computed Tomography Inc.

Research Article

Keywords: quantum computers, computing algorithms, QUBO, D-Wave system

Posted Date: January 6th, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-558111/v1>

License:  This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

QUBO formulations for a system of linear equations

Kyungtaek Jun^{a,*}

^a Research Center, Innovation on Quantum and Computed Tomography, Seoul, Republic of Korea

* ktfriends@gmail.com

Abstract

With the advent of quantum computers, many quantum computing algorithms are being developed. Solving linear systems is one of the most fundamental problems in almost all science and engineering. The Harrow-Hassidim-Lloyd algorithm, a monumental quantum algorithm for solving linear systems on gate model quantum computers, was invented and several advanced variations have been developed. For a given square matrix $A \in \mathbb{R}^{n \times n}$ and a vector $\vec{b} \in \mathbb{R}^n$, we will find unconstrained binary optimization (QUBO) models for a vector $\vec{x} \in \mathbb{R}^n$ that satisfies $A\vec{x} = \vec{b}$. To formulate QUBO models for a linear system solving problem, we make use of a linear least-square problem with binary representation of the solution. We validate those QUBO models on the D-Wave system and discuss the results. For a simple system, we provide a Python code to calculate the matrix characterizing the relationship between the variables and to print the test code that can be used directly in the D-Wave system.

Introduction

Quantum computing has opened up a new paradigm for approaching computing problems where classical (i.e., traditional) computing can provide unparalleled speeds for specific problems. A specific subset of quantum computing is quantum annealing aimed at optimization problems. Quantum annealing processors naturally return low-energy solutions; some applications require the real minimum energy and others require good low-energy samples¹. One popular model of optimization problems on which quantum annealers are based is the Ising model².

A monumental quantum algorithm for solving linear systems on gate model quantum computers, was invented in 2008, and a variety of advanced variations were developed³. At that time, their algorithm and its variants

had many limitations, such as the quantum state of the solution, the need for quantum RAM, and the limitations of Matrix A ⁴. In particular, the quantum state of the solution limits the use of the algorithm to submodules of the entire system, or reduces the quantum speed improvement in real world implementations, as it requires iterative computations to read the quantum state of each base. In 2019, a method of using linear least squares was proposed to solve a linear problem⁵⁻⁷. They suggested a QUBO model of linear equations for m by n linear systems. In addition, when solving a linear problem using the QUBO model, they compared several classic methods such as QR factorization and SVD⁸. There has been remarkable research⁹⁻¹⁴ since the development of D-Wave quantum annealers.

In this paper, we propose two representative QUBO models for n by n linear systems. Additionally, we show the overall CPU calculations when performing the new QUBO modeling. The amount of calculation used here consists of simple addition and multiplication. These results will be of great advantage when solving linear equations for large n . We also provide Python code to create QUBO models that can be used in D-Wave quantum annealers. To date, the number of qubits that can be used in quantum annealers is approximately 200, so the size of the linear problem that can be solved using our QUBO model is not large. However, with the development of quantum annealers, the method we propose will have advantages. It is also expected that other issues could be approached in a similar way to ours.

Method

Background

The Ising model is a mathematical model for ferromagnetism in statistical mechanics. The energy Hamiltonian (the cost function) is formulated as follows:

$$H(\vec{\sigma}) = - \sum_{i=1}^N h_i \sigma_i - \sum_{i<j}^N J_{i,j} \sigma_i \sigma_j \quad \text{Equation 1}$$

where $\vec{\sigma} = (\sigma_1, \dots, \sigma_N)^T$ and $\sigma_i \in \{+1, -1\}$.

Quadratic unconstrained binary optimization (QUBO) is a combinatorial optimization problem in computer science. In this problem, a cost function f is defined on an n -dimensional binary vector space \mathbb{B}^n onto \mathbb{R} .

$$f(\vec{x}) = \vec{x}^T Q \vec{x} \quad \text{Equation 2}$$

where Q is an upper diagonal matrix, $\vec{x} = (x_1, \dots, x_N)^T$, and $x_i \in \{0,1\}$. The problem is finding x^* , which minimizes the cost function f . Since we have $x_i^2 = x_i$, the cost function is reformulated as follows:

$$f(\vec{x}) = \sum_{i=1}^N Q_{i,i} x_i + \sum_{i<j}^N Q_{i,j} x_i x_j. \quad \text{Equation 3}$$

In Q , the diagonal terms $Q_{i,i}$ and the off-diagonal terms $Q_{i,j}$ represent the linear terms and the quadratic terms, respectively. The unknowns of the Ising Model s and the unknowns of the QUBO Model x have the linear relation

$$s = 2x - 1 \quad \text{or} \quad x = \frac{1}{2}(s + 1) \quad \text{Equation 4}$$

Therefore, we can choose appropriate format depending on the binary unknown.

Given a matrix $A \in \mathbb{R}^{n \times n}$ and a column vector of variables $\vec{x} \in \mathbb{R}^n$ and a column vector $\vec{b} \in \mathbb{R}^n$. The linear least squares problem is to find the \vec{x} that minimizes $\|A\vec{x} - \vec{b}\|$. In other words, it can be described as follows:

$$\arg \min_{\vec{x}} \|A\vec{x} - \vec{b}\| \quad \text{Equation 5}$$

To solve Eq. 5 let us begin by writing out $\|A\vec{x} - \vec{b}\|$:

$$A\vec{x} - \vec{b} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \quad \text{Equation 6}$$

Taking the 2-norm square of the resultant vector of Eq. 6, we get the following:

$$\|A\vec{x} - \vec{b}\|_2^2 = \sum_{k=1}^n \left(\sum_{i=1}^n a_{k,i} x_i - b_k \right)^2 \quad \text{Equation 7}$$

$$= \sum_{k=1}^n \left(\left(\sum_{i=1}^n a_{k,i} x_i \right)^2 - 2b_k \sum_{i=1}^n a_{k,i} x_i + b_k^2 \right) \quad \text{Equation 8}$$

$$= \sum_{k=1}^n \left(\sum_{i=1}^n (a_{k,i} x_i)^2 + 2 \sum_{i<j} a_{k,i} a_{k,j} x_i x_j - 2b_k \sum_{i=1}^n a_{k,i} x_i + b_k^2 \right) \quad \text{Equation 9}$$

QUBO modeling 1

If we were solving binary least squares, then each x_i would be represented by the combination of qubits $q_{i,l} \in \{0,1\}$. In the work by O'Malley and Vesselinov¹⁵, the radix 2 representation of positive real value x_i is given by

$$x_i \approx \sum_{l=-m}^m 2^l q_{i,l} \quad \text{Equation 10}$$

where positive integer l is the number of digits of x_i and negative l is the number of digits of fractional terms.

Now, we can represent real value x_i is below

$$x_i \approx \sum_{l=-m}^m 2^l q_{i,l}^+ - \sum_{l=-m}^m 2^l q_{i,l}^- \quad \text{Equation 11}$$

To represent both positive and negative numbers, $q_{i,l}^+$ and $q_{i,l}^-$ are involved. Note that this representation can have the same value with different binary combinations.

To drive QUBO model, we insert Eq. 11 into Eq. 9. We obtain the two summation terms of the first term in Eq. 9 as below:

$$\sum_{k=1}^n \sum_{i=1}^n (a_{k,i} x_i)^2 \approx \sum_{k=1}^n \sum_{i=1}^n \left(a_{k,i} \left(\sum_{l=-m}^m 2^l q_{i,l}^+ - \sum_{l=-m}^m 2^l q_{i,l}^- \right)^2 \right) \quad \text{Equation 12}$$

$$= \sum_{k=1}^n \sum_{i=1}^n \left(a_{k,i}^2 \left(\left(\sum_{l=-m}^m 2^l q_{i,l}^+ \right)^2 + \left(\sum_{l=-m}^m 2^l q_{i,l}^- \right)^2 \right) \right) \quad \text{Equation 13}$$

$$= \sum_{k=1}^n \sum_{i=1}^n \left(a_{k,i}^2 \left(\sum_{l=-m}^m 2^{2l} \left((q_{i,l}^+)^2 + (q_{i,l}^-)^2 \right) + \sum_{l_1 < l_2} 2^{l_1+l_2+1} (q_{i,l_1}^+ q_{i,l_2}^+ + q_{i,l_1}^- q_{i,l_2}^-) \right) \right) \quad \text{Equation 14}$$

$$= \sum_{k=1}^n \sum_{i=1}^n \sum_{l=-m}^m a_{k,i}^2 2^{2l} (q_{i,l}^+ + q_{i,l}^-) + \sum_{k=1}^n \sum_{i=1}^n \sum_{l_1 < l_2} a_{k,i}^2 2^{l_1+l_2+1} (q_{i,l_1}^+ q_{i,l_2}^+ + q_{i,l_1}^- q_{i,l_2}^-) \quad \text{Equation 15}$$

Since x_i is a positive, zero, or negative number, $q_{i,l_1}^+ \times q_{i,l_2}^-$ for the same i can be zero. We can get Eq. 13 from Eq.

12. Additionally, since $q^2 = q$, Eq. 15 can be finally obtained. Here, the front summation represents a linear term, and the second summation represents a quadratic term.

The QUBO form of the second term in Eq. 9 can be obtained as below:

$$\sum_{k=1}^n \sum_{i < j} 2 a_{k,i} a_{k,j} x_i x_j \approx \sum_{k=1}^n \sum_{i < j} 2 a_{k,i} a_{k,j} \left(\sum_{l=-m}^m 2^l q_{i,l}^+ - \sum_{l=-m}^m 2^l q_{i,l}^- \right) \left(\sum_{l=-m}^m 2^l q_{j,l}^+ - \sum_{l=-m}^m 2^l q_{j,l}^- \right) \quad \text{Equation 16}$$

Equation 17

$$= \sum_{k=1}^n \sum_{i < j} \sum_{l_1=-m}^m \sum_{l_2=-m}^m 2^{l_1+l_2+1} \mathbf{a}_{k,i} \mathbf{a}_{k,j} (\mathbf{q}_{i,l_1}^+ \mathbf{q}_{j,l_2}^+ + \mathbf{q}_{i,l_1}^- \mathbf{q}_{j,l_2}^- - \mathbf{q}_{i,l_1}^+ \mathbf{q}_{j,l_2}^- - \mathbf{q}_{i,l_1}^- \mathbf{q}_{j,l_2}^+)$$

When we calculate the QUBO term, we use an upper triangular matrix. Therefore $q_{i,l}^+, q_{i,l}^-, q_{j,l}^+$, and $q_{j,l}^-$ have separate indices, and we can derive Eq. 17 from Eq. 16. This equation is part of the quadratic terms in our QUBO model.

The QUBO form of the third term in Eq. 9 can be obtained as below:

$$\sum_{k=1}^n \sum_{i=1}^n (-2 \mathbf{a}_{k,i} \mathbf{b}_k x_i) \approx \sum_{k=1}^n \sum_{i=1}^n \sum_{l=-m}^m 2^{l+1} \mathbf{a}_{k,i} \mathbf{b}_k (q_{i,l}^- - q_{i,l}^+) \quad \text{Equation 18}$$

The above equation is part of the linear terms in our QUBO model.

The last term in Eq. 9 represents the minimum value when the vector \vec{x} satisfies the linear equation in QUBO form. Therefore, our first QUBO model for linear systems is the summation of Eq. 15, Eq. 17, and Eq. 18.

QUBO modeling 2

To reduce the number of qubits to be used in Eq. 11, the new approximation of x_i was introduced as follows:

$$x_i \approx -2^{m+1} q_i^- + \sum_{l=-m}^m 2^l q_{i,l}^+ \quad \text{Equation 19}$$

Since the coefficient of q_i^- represents the amount of translation of the range of the summation part in Eq. 19, any coefficient of q_i^- that can represent the range of x_i we want can be used. We select the coefficient of q_i^- , which represents the largest range that x_i can have.

To drive QUBO model, we insert Eq. 19 into Eq. 9. We get another QUBO form of the first term in Eq. 9 as below:

$$\sum_{k=1}^n \sum_{i=1}^n (\mathbf{a}_{k,i} x_i)^2 \approx \sum_{k=1}^n \sum_{i=1}^n \left(\mathbf{a}_{k,i} \left(-2^{m+1} q_i^- + \sum_{l=-m}^m 2^l q_{i,l}^+ \right) \right)^2 \quad \text{Equation 20}$$

$$= \sum_{k=1}^n \sum_{i=1}^n \mathbf{a}_{k,i}^2 \left((-2^{m+1} q_i^-)^2 + \sum_{l=-m}^m (-2^{l+m+2} q_i^- q_{i,l}^+) + \left(\sum_{l=-m}^m 2^l q_{i,l}^+ \right)^2 \right) \quad \text{Equation 21}$$

$$= \sum_{k=1}^n \sum_{i=1}^n \mathbf{a}_{k,i}^2 \left(2^{2m+2} (q_i^-)^2 + \sum_{l=-m}^m (-2^{l+m+2} q_i^- q_{i,l}^+) + \sum_{l=-m}^m 2^{2l} (q_{i,l}^+)^2 + \sum_{l_1 < l_2} 2^{l_1+l_2} q_{i,l_1}^+ q_{i,l_2}^+ \right) \quad \text{Equation 22}$$

$$= \sum_{k=1}^n \sum_{i=1}^n a_{k,i}^2 \left(2^{2m+2} q_i^- + \sum_{l=-m}^m 2^{2l} q_{i,l}^+ \right) + \sum_{k=1}^n \sum_{i=1}^n a_{k,i}^2 \left(\sum_{l=-m}^m (-2^{l+m+2} q_i^- q_{i,l}^+) + \sum_{l_1 < l_2} 2^{l_1+l_2+1} q_{i,l_1}^+ q_{i,l_2}^+ \right) \quad \text{Equation 23}$$

Since $q^2 = q$, Eq. 23 can be finally obtained. The front summation represents linear terms, and the second summation represents quadratic terms.

The QUBO form of the second term in Eq. 9 can be obtained as below:

$$\sum_{k=1}^n \sum_{i < j} 2 a_{k,i} a_{k,j} x_i x_j \approx \sum_{k=1}^n \sum_{i < j} 2 a_{k,i} a_{k,j} \left(-2^{m+1} q_i^- + \sum_{l=-m}^m 2^l q_{i,l}^+ \right) \left(-2^{m+1} q_j^- + \sum_{l=-m}^m 2^l q_{j,l}^+ \right) \quad \text{Equation 24}$$

$$= \sum_{k=1}^n \sum_{i < j} a_{k,i} a_{k,j} \left(2^{m+3} q_i^- q_j^- - \sum_{l=-m}^m 2^{l+m+2} (q_i^- q_{j,l}^+ + q_j^- q_{i,l}^+) + 2 \sum_{l=-m}^m 2^l q_{i,l}^+ \sum_{l=-m}^m 2^l q_{j,l}^+ \right) \quad \text{Equation 25}$$

$$= \sum_{k=1}^n \sum_{i < j} a_{k,i} a_{k,j} \left(2^{m+3} q_i^- q_j^- - \sum_{l=-m}^m 2^{l+m+2} (q_i^- q_{j,l}^+ + q_j^- q_{i,l}^+) + \sum_{l_1=-m}^m \sum_{l_2=-m}^m 2^{l_1+l_2+1} q_{i,l_1}^+ q_{j,l_2}^+ \right) \quad \text{Equation 26}$$

Equation 26 can be derived in a similar way to Eq. 17. This equation is part of the quadratic terms in our second QUBO model.

The QUBO form of the third term in Eq. 9 can be obtained as below:

$$\sum_{k=1}^n \sum_{i=1}^n (-2 a_{k,i} b_k x_i) \approx \sum_{k=1}^n \sum_{i=1}^n 2^{m+2} a_{k,i} b_k q_{i,l}^+ + \sum_{k=1}^n \sum_{i=1}^n \sum_{l=-m}^m (-2^{l+1} a_{k,i} b_k q_{i,l}^+) \quad \text{Equation 27}$$

The last term in Eq. 9 represents the minimum value when the vector \vec{x} satisfies the linear equation in QUBO form.

Therefore, our second QUBO model for linear systems is the summation of Eq. 23, Eq. 26, and Eq. 27.

Implementation

Let's consider the following example¹⁶:

$$A = \begin{pmatrix} 3 & 1 \\ -1 & 2 \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} -1 \\ 5 \end{pmatrix}, \quad \text{and} \quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} q_{11} + 2q_{12} - q_{13} - 2q_{14} \\ q_{21} + 2q_{22} - q_{23} - 2q_{24} \end{pmatrix} \quad \text{Equation 28}$$

where q_{ij} is binary and the range of integer x_i is $-3 \leq x_i \leq 3$.

They used the related matrix below to solve Eq. 28.

$$Q_1 = \begin{pmatrix} 26 & 40 & -20 & -40 & 2 & 4 & -2 & -4 \\ & 72 & -40 & -80 & 4 & 8 & -4 & -8 \\ & & -6 & 40 & -2 & -4 & 2 & 4 \\ & & & 8 & -4 & -8 & 4 & 8 \\ & & & & -13 & 20 & -10 & -20 \\ & & & & & -16 & -20 & -40 \\ & & & & & & 23 & 20 \\ & & & & & & & 56 \end{pmatrix} \quad \text{Equation 29}$$

The diagonal entries and the upper diagonal entries represent coefficients of linear terms and quadratic terms, respectively, in Formula².

Our matrix from QUBO modeling 1 is below:

$$Q_2 = \begin{pmatrix} 26 & 40 & 0 & 0 & 2 & 4 & -2 & -4 \\ & 72 & 0 & 0 & 4 & 8 & -4 & -8 \\ & & -6 & 40 & -2 & -4 & 2 & 4 \\ & & & 8 & -4 & -8 & 4 & 8 \\ & & & & -13 & 20 & 0 & 0 \\ & & & & & -16 & 0 & 0 \\ & & & & & & 23 & 20 \\ & & & & & & & 56 \end{pmatrix} \quad \text{Equation 30}$$

Two different matrices have the same solution for Eq. 28.

Empirical Test

We tested the QUBO model of Eq. 30 on the D-Wave 2000Q system with 10,000 anneals, and compared the result from Eq. 29¹⁶. List 1 shows the Python code for our matrix and prints the Python code for running the QUBO model described in Formula 30. This code creates a QUBO model with nonzero coefficient terms.

```

1 import numpy as np
2 import random, math
3 import copy
4
5 Dimension = 2
6 qubits = 2
7 A = np.array([[3, 1], [-1, 2]])
8 b = np.array([-1, 5])
9
10 QM = np.zeros((2*qubits*Dimension, 2*qubits*Dimension))
11 for k in range(Dimension):
12     for i in range(Dimension):
13         for l in range(qubits):
14             cef1 = pow(2,2*i)*pow(A[k][i],2)
15             cef2 = pow(2,l+1)*A[k][i]*b[k]
16             po1 = 4*i + 1
17             po2 = 4*i + 1 + 2
18             QM[po1][po1] = QM[po1][po1] + cef1 - cef2
19             QM[po2][po2] = QM[po2][po2] + cef1 + cef2
20
21 for k in range(Dimension):
22     for i in range(Dimension):
23         for l1 in range(qubits-1):
24             for l2 in range(l1+1,qubits):
25                 qcef = pow(2, l1+l2+1)*pow(A[k][i],2)
26                 po1 = 4*i + 1
27                 po2 = 4*i + 1 + 2
28                 QM[po1][po2] = QM[po1][po2] + qcef
29                 po3 = 4*i + 1 + 2
30                 po4 = 4*i + 1 + 2 + 2
31                 QM[po3][po4] = QM[po3][po4] + qcef
32
33 for k in range(Dimension):
34     for i in range(Dimension-1):
35         for j in range(i+1,Dimension):
36             for l1 in range(qubits):
37                 for l2 in range(qubits):
38                     qcef = pow(2, l1+l2+1)*A[k][i]*A[k][j]
39                     po1 = 4*i + 1
40                     po2 = 4*j + 1
41                     QM[po1][po2] = QM[po1][po2] + qcef
42                     po3 = 4*i + 1 + 2
43                     po4 = 4*j + 1 + 2
44                     QM[po3][po4] = QM[po3][po4] + qcef
45                     po5 = 4*i + 1
46                     po6 = 4*j + 1 + 2
47                     QM[po5][po6] = QM[po5][po6] - qcef
48                     po7 = 4*i + 1 + 2
49                     po8 = 4*j + 1
50                     QM[po7][po8] = QM[po7][po8] - qcef
51
52 # Print Matrix Q
53 print("# Matrix Q is")
54 print(QM)

```

```

55
56 # Print Python code for the run in D-Wave quantum processing unit
57 print("from dwave.system import DWaveSampler, EmbeddingComposite")
58 print("sampler_auto = EmbeddingComposite(DWaveSampler(solver={'qpu': True}))\n")
59 print("linear = {", end = "")
60 for i in range(2*qubits*Dimension-1):
61     linear = i + 1
62     print ("('q",linear,"','q",linear,"): ",format(QM[i][i]),sep=",", end = ", ")
63 print ("('q",2*qubits*Dimension,"','q",2*qubits*Dimension,"): ",format(QM[2*qubits*Dimension-1][2*qubits*Dimension-1]),"}", sep=")
64
65 print("\nquadratic = {", end = "")
66 for i in range(2*qubits*Dimension-1):
67     for j in range(i+1,2*qubits*Dimension):
68         if QM[i][j] != 0:
69             qdrt1 = i + 1
70             qdrt2 = j + 1
71             if i == 2*qubits*Dimension-2 and j == 2*qubits*Dimension-1:
72                 print ("('q",qdrt1,"','q",qdrt2,"): ",format(QM[i][j]), " }", sep=")
73             else:
74                 print ("('q",qdrt1,"','q",qdrt2,"): ",format(QM[i][j]), sep=",", end = ", ")
75
76 print("\nQ = dict(linear)")
77 print("Q.update(quadratic)\n")
78
79 qa_iter = 1000
80 print("sampleset = sampler_auto.sample_qubo(Q, num_reads=",qa_iter,")", sep = "")
81 print("print(sampleset)")

```

82 **List 1.** Python code for 2×2 Matrix Q and the run in the D-Wave quantum processing unit (QPU)

For Q_1 , Table 1 shows the number of occurrences of the lowest energy -26 for invertible matrix A . The total occurrence of all combinations of qubits having the lowest energy is 6,316 from 10,000 anneals. In Eq. 9, we have the constant term 26 and the QUBO form in Eq. 29 is the same with Eq. 9 except the constant term. Since the possible minimum of Eq. 9 is zero, the possible minimum of our QUBO model is -26 calculated as the summation of b_k . For $x_1 = -1$, there are three qubit representations: $[q_{11}, q_{12}, q_{13}, q_{14}] = [0,0,1,0], [0,1,1,1]$ or $[1,0,0,1]$. For $x_2 = 2$, there are two qubit representations: $[q_{21}, q_{22}, q_{23}, q_{24}] = [0,1,0,0]$ or $[1,1,1,0]$. Thus, 6,316 anneals achieved the lowest energy state, with an energy equal to -26 .

q_{11}	q_{12}	q_{13}	q_{14}	q_{21}	q_{22}	q_{23}	q_{24}	energy	# of occurrences
0	1	1	1	0	1	0	0	-26.0	959
0	1	1	1	1	1	1	0	-26.0	2066
0	0	1	0	0	1	0	0	-26.0	499
1	0	0	1	1	1	1	0	-26.0	1256
0	0	1	0	1	1	1	0	-26.0	925
1	0	0	1	0	1	0	0	-26.0	611

Table 1. The Results of the solutions for the QUBO model, Eq. 29

For Q_2 with zero terms in QUBO, Table 2 shows the number of occurrences of the lowest energy -26 . The total occurrence of all combinations of qubits having the lowest energy is 8,400 from 10,000 anneals. Thus, 8,400 anneals achieved the lowest energy state, with an energy equal to -26 .

q_{11}	q_{12}	q_{13}	q_{14}	q_{21}	q_{22}	q_{23}	q_{24}	energy	# of occurrences
0	0	1	0	0	1	0	0	-26.0	8400

Table 2. The Results of the solutions for the QUBO model, Eq. 29 with zero terms

For Q_2 without zero terms in QUBO, Table 3 shows the number of occurrences of the lowest energy -26 . The total occurrence of all combinations of qubits having the lowest energy is 8,725 from 10,000 anneals. Thus, 8,725 anneals achieved the lowest energy state, with an energy equal to -26 .

q_{11}	q_{12}	q_{13}	q_{14}	q_{21}	q_{22}	q_{23}	q_{24}	energy	# of occurrences
0	0	1	0	0	1	0	0	-26.0	8725

Table 3. The Results of the solutions for the QUBO model, Eq. 29 without zero terms

Discussion

The most computationally intensive part of our first QUBO form is Eq. 17, one of the quadratic terms.

We want to discuss Eq. 17 below. By computing Eq. 17, we calculate variables k, i , and j . Variable k can vary from 1 to n , and i and j represent the (i, j) components of the upper triangular matrices without diagonal components, so the total amount of calculation can be expressed as follows:

$$\text{Major computing cost for } (i, j) = \frac{n(n-1)}{2}$$

Equation 31

$l_1 + l_2$ can have a value from $-2m$ to $2m$. Additionally, the product of $a_{k,i}a_{k,j}$ and $2^{l_1+l_2+1}$ can be calculated by addition. Therefore, the total amount of calculation is as follows:

$$\text{Major computing cost for } (i, j, l_1, l_2) = (4m + 1) + \frac{n(n-1)}{2}$$

Equation 32

$$\text{Major computing cost for } (i, j, k, l_1, l_2) = (4m + 1)n + \frac{n^2(n-1)}{2}$$

Equation 33

To make the QUBO model solve the linear equation, the total amount of calculation seems to be large as shown in Eq. 32. However, since each step consists of one multiplication or one addition, the solution can be solved faster than solving a linear equation in a classical computer. Our Python code List 1 did not minimize the amount of computation to improve readability.

To remove the fractional part of x_i in Eqs. 11 and 19, we can use $\|A(c\vec{x}) - c\vec{b}\|$ instead of $\|A\vec{x} - \vec{b}\|$. In more detail, we can use the $\|A\vec{y} - c\vec{b}\|$ equation and $y_i \approx \sum_{l=0}^m 2^l q_{i,l}$ for equation $\|A(c\vec{x}) - c\vec{b}\|$. If the solution requires an accuracy of one hundredth in the fractional part, then c is 100. In Eq. 9, b_k can be changed to cb_k . In the case of using the QUBO form to find a linear solution in quantum annealing, we already know the minimum value

for the invertible matrix. For example, if the range of x in Eq. 28 is expressed as a number with two decimal places and two digits, then the positive value of x_i must be expressed as $x_i \approx \sum_{l=-6}^6 2^l q_{i,l}^+$. For the new formula $\|A\vec{y} - c\vec{b}\|$, the positive value of y_i must be expressed as $y_i = \sum_{l=0}^{13} 2^l q_{i,l}^+$. In this case, one more qubit is used, but a more accurate solution can be obtained.

The greatest advantage of this model lies in parallel computing in the process of creating the QUBO equation. We can transform Eq. 17 as follows to calculate the cost for parallel computing:

$$\sum_{l_1=-m}^m \sum_{l_2=-m}^m \left(\sum_{k=1}^n \sum_{i<j} 2^{l_1+l_2+1} a_{k,i} a_{k,j} \right) (q_{i,l_1}^+ q_{j,l_2}^+ + q_{i,l_1}^- q_{j,l_2}^- - q_{i,l_1}^+ q_{j,l_2}^- - q_{i,l_1}^- q_{j,l_2}^+) \quad \text{Equation 34}$$

In Eq. 34, the coefficient of each quadratic term is divisible with respect to k . This means that it can be efficiently partitioned when distributing n parallel processes. At this time, each process requires computing cost as in Eq. 32. To calculate the remaining term the fastest, it can be calculated through $n(n-1)/4$ parallel systems for each k . In this case, the efficiency of the total computational cost is reduced, but the computational cost of the QUBO model is approximately $2\log_2 n$. When performing our QUBO modeling, as the total number of processes increases, the computing cost decreases. However, there is a large drawback to calculating linear equations in a quantum annealer. Our QUBO formula requires that one qubit be connected to all other qubits. However, each quantum annealer has its own topology, which limits the connectivity between qubits. To develop a QUBO model that can be used in a quantum annealer where the connection between qubits is limited, it is necessary to study the method of diagonalizing Q_2 based on a quantum annealer's topology.

QUBO modeling 1 has a major disadvantage in that it uses approximately twice as many qubits as QUBO modeling 2. However, we mainly discussed the results of QUBO modeling 1 to explain the important results for the constraints of the QUBO models. To solve the system of linear equations, each element x_i of \vec{x} can be divided by positive, negative, or zero. When Eq. 11 is used for the element x_i , and the result can be expressed as combinations of elements in q^+ and q^- . However, a solution x_i can be made with either $\sum 2^l q_{i,l}^+$ or $-\sum 2^l q_{i,l}^-$. Because of this property, we can assume this problem to be a constrained problem for certain quadratic terms. To make certain quadratic terms 0, it is common to add the coefficients of the terms to a large value. To make certain quadratic terms

equal to 0, it is common to add the coefficients of the terms to a large value. However, in this paper, we propose a new method of adding different values to each term so that it is set to 0 when the coefficient of each term is negative. Therefore, we set each coefficient of $q_{i,l_1}^+ q_{i,l_2}^-$ to zero for each x_i . Equation 30 was made by adding the same positive value to the coefficient of each negative constrained term in Eq. 29. In this way, we experimented by adding values of 100, 500, and 1000 to the constraint terms of Q matrices of various sizes with a size of 32 by 32 or less. However, we did not obtain better results than the two methods of setting each constrained term to 0 introduced in this paper. We obtained these results in D-wave 2000Q. The reason why we obtained the best results by removing the constrained terms from QUBO is probably because we obtained a physical advantage by simplifying the superposition and entanglement of qubits by reducing the terms.

References

1. S., B., Rønnow, T. & Isakov, S. *et. al.* Evidence for quantum annealing with more than one hundred qubits. *Nat. Phys* **10**, 218–224, DOI: <https://doi.org/10.1038/nphys2900> (2014).
2. Kadowaki, T. & Nishimori, H. Quantum annealing in the transverse ising model. *Phys. Rev. E* **58**, 5355–5363, DOI: [10.1103/PhysRevE.58.5355](https://doi.org/10.1103/PhysRevE.58.5355) (1998).
3. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. review letters* **103**, 150502 (2009).
4. Aaronson, S. Read the fine print. *Nat. Phys.* **11**, 291–293 (2015).
5. Borle, A. & Lomonaco, S. J. Analyzing the quantum annealing approach for solving linear least squares problems. In *WALCOM: Algorithms and Computation*, vol. 11355, 289–301, DOI: [10.1007/978-3-030-10564-8](https://doi.org/10.1007/978-3-030-10564-8) (2019).
6. Chang, T. H., Lux, T. C. & Tipirneni, S. S. Least-squares solutions to polynomial systems of equations with quantum annealing. *Quantum Inf. Process.* **18**, 1–17 (2019).
7. Potok, T. *et al.* Adiabatic quantum linear regression. *arXiv preprint arXiv:2008.02355* (2020).
8. Lee, D. Q. Numerically efficient methods for solving least squares problems (2012).
9. Adachi, S. H. & Henderson, M. P. Application of quantum annealing to training of deep neural networks (2015). [1510.06356](https://arxiv.org/abs/1510.06356).
10. B., O., R., B. & Perdomo-Ortiz, A. *et. al.* Bayesian network structure learning using quantum annealing. *Eur. Phys. J. Spec. Top.* **224**, 163–188 (2015).
11. Daniel, O., Vesselinov, V. V., Alexandrov, B. S. & Alexandrov, L. B. Nonnegative/binary matrix factorization with a d-wave quantum annealer. *PLOS ONE* **13**, 1–12, DOI: [10.1371/journal.pone.0206653](https://doi.org/10.1371/journal.pone.0206653) (2018).
12. Date, P., Arthur, D. & Pusey-Nazzaro, L. Qubo formulations for training machine learning models. *Sci. Reports* **11**, 1–10 (2021).
13. Arthur, D. *et al.* Balanced k-means clustering on an adiabatic quantum computer. *Quantum Inf. Process.* **20**, 1–30 (2021).
14. Pusey-Nazzaro, L. *et al.* Adiabatic quantum optimization fails to solve the knapsack problem. *arXiv preprint arXiv:2008.07456* (2020).
15. O’Malley, D. & Vesselinov, V. V. Toq. jl: A high-level programming language for d-wave machines based on julia. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–7 (IEEE, 2016).
16. Jun, K. *et. al.* Solving linear systems by quadratic unconstrained binary optimization on d-wave quantum

annealing device. vol. 11726, DOI: [10.1117/12.2591588](https://doi.org/10.1117/12.2591588) (International Society for Optics and Photonics, 2021).

Author contributions Statement

K. J conceived and designed the research problem and coded for simulations using Python Numpy. K.J performed the hardware testing and wrote the paper.

Competing financial interests

The authors declare no competing financial and non-financial interests.

Acknowledgements

The author would like to thank the sincere comments of the in-house editor, editor, and two reviewers.