Hybrid Annealing Method Based on subQUBO Model Extraction With Multiple Solution Instances

Yuta Atobe, Masashi Tawada, *Member, IEEE*, and Nozomu Togawa, *Member, IEEE*

Abstract—Ising machines are expected to solve combinatorial optimization problems efficiently by representing them as Ising models or equivalent quadratic unconstrained binary optimization (QUBO) models . However, upper bound exists on the computable problem size due to the hardware limitations of Ising machines. This paper propose a new hybrid annealing method based on partial QUBO extraction, called subQUBO model extraction, with multiple solution instances. For a given QUBO model, the proposed method obtains N_I quasi-optimal solutions (quasi-ground-state solutions) in some way using a classical computer. The solutions giving these quasi-optimal solutions are called *solution instances*. We extract a size-limited subQUBO model as follows based on a strong theoretical background: we randomly select N_S ($N_S < N_I$) solution instances among them and focus on a particular binary variable x_i in the N_S solution instances. If x_i value is much *varied* over N_S solution instances, it is included in the subQUBO model; otherwise, it is not. We find a (quasi-)ground-state solution of the extracted subQUBO model using an Ising machine and add it as a new solution instance. By repeating this process, we can finally obtain a (quasi-) ground-state solution of the original QUBO model. Experimental evaluations confirm that the proposed method can obtain better quasi-ground-state solution than existing methods for large-sized QUBO models.

Index Terms—Ising machine, Ising model, hybrid annealing method, QUBO model, subQUBO model extraction

1 Introduction

The combinatorial optimization problem involves finding the combination of variables that minimizes (or maximizes) the objective function within a given set of constraints and has many applications in the real world (e.g., route optimization for transportation or logistics and optimal control of robot behavior) [1], [2], [3]. Some of these combinatorial optimization problems are categorized into a problem class, called the NP-hard problem, which is difficult to solve in a practical time using conventional computers because the number of variable combinations explosively increases as the number of variables increases [3]. To tackle this problem, Ising machines have recently been focused on as a new computer architecture [4], [5], [6].

An Ising machine solves an original combinatorial optimization problem by transforming it into a statistical mechanics model, called an *Ising model*, or its equivalent quadratic unconstrained binary optimization model, called a *QUBO model*, and finding its minimum energy state. This state is called the *ground state*. The QUBO model consists of multiple 0-1 binary variables, and a combination of which gives one solution. In particular, the combination of binary variables giving the minimum energy state of the QUBO model is called the ground-state solution. Ising machines include, for example, D-Wave Systems' 2000Q [4] and Fujitsu's digital annealer [5], [6]. Although these Ising machines are expected to quickly solve combinatorial optimization problems, there is an upper limit to the variable size of a QUBO model that can be solved because of hardware limitations. For example, the D-Wave 2000Q machine has approximately 2000 qubits but, due to its hardware limitations, it can only deal with a QUBO model having a maximum of 64 variables if all the variables are connected to each other [7], [8].

This work investigates a hybrid annealing method for solving large-sized QUBO models using classical computers and Ising machines. The hybrid annealing method is based on extracting several binary variables from the original QUBO model and constructing a small-sized QUBO model, called a *subQUBO model*, using classical computers. The method then finds its (quasi-)ground-state solution using Ising machines. In this process, how to extract the subQUBO model from the original QUBO model is the key to efficiently obtaining the ground-state solution of the original QUBO model.

Several methods for extracting subQUBO models from an original QUBO model have been proposed [9], [10], [11]. The simplest subQUBO model extraction method randomly chooses variables from the original QUBO model [9]. This method may have a probabilistic effect depending on the

- *Yuta Atobe and Masashi Tawada are with Green Computing Systems Research Organization, Waseda University, Tokyo 162-0042, Japan. E-mail: yutaatobe@aoni.waseda.jp, tawada@togawa.cs.waseda.ac.jp.*
- *Nozomu Togawa is with the Department of Computer Science and Communications Engineering, Waseda University, Tokyo 169-8555, Japan. E-mail: togawa@togawa.cs.waseda.ac.jp.*

Manuscript received 7 May 2021; revised 30 Nov. 2021; accepted 22 Dec. 2021. Date of publication 28 Dec. 2021; date of current version 8 Sept. 2022.
This work was supported by the results obtained from a Project No. JPNP 16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO), Japan.
(Corresponding authors: Yuta Atobe and Nozmu Togawa.)
Recommended for acceptance by M. Lipasti.
Digital Object Identifier no. 10.1109/TC.2021.3138629

QUBO model structure. The next possible method is based on the changes of the objective value. A method based on the *impact values* was proposed in [10], while a method based on *k-opt local search* [12], [13] was proposed in [9]. Ref. [9] also proposed a method for extracting subQUBO models from relatively good solutions reached in the past process. In [11], a subQUBO model was extracted based on the solutions solved by a classical computer. As discussed in Section 3.2, these methods are all heuristic and do not necessarily have a theoretical background. Furthermore, the extracted subQUBO models tend to be fixed because they are extracted from fixed solutions. We cannot expect that a globally optimal solution is obtained for an original QUBO model by using these methods.

This paper proposes a new hybrid annealing method that can efficiently obtain the (quasi-)ground-state solution for a QUBO model. For a given QUBO model, the proposed method first obtains N_I quasi-ground-state solutions, called *solution instances*, in some way using a classical computer. Next, N_S ($N_S < N_I$) solution instances are randomly selected from the N_I solutions instances. We then extract a size-limited subQUBO model as follows based on the strong theoretical background: we examine how much every binary variable in the N_S solution instances is *varied*. A binary variable x_i is the most varied when it has the same value in just the $N_S/2$ solution instances. On the contrary, a binary variable x_i is called the least varied when it has the same value in all the N_S solution instances. We sort the binary variables from the most varied to the least varied and construct the subQUBO model using the binary variables in the sorted order, with its size not exceeding the hardware limit. The proposed method tends not to fix the extracted subQUBO models by randomly selecting the N_S solution instances from the N_I solution instances. Repeating this process, we can obtain a (quasi-)ground-state solution of the original QUBO model.

The main contributions of this paper are summarized as follows:

1) We prove ideal subQUBO model extraction and propose a subQUBO model extraction method based on this theorem. As far as we know, this is the first theorem to theoretically describe the ideal subQUBO model extraction.
2) We implement the proposed method and give insight into the optimal parameter setting thorough experimental evaluations.
3) We demonstrate that the proposed method can obtain better quasi-ground-state solutions compared to existing methods in both CPU simulation evaluations and evaluations using a quantum annealing machine.

The rest of this paper is organized as follows: Section 2 first defines the QUBO models and then introduces the Ising machines and their hardware limitations; Section 3 formulates the subQUBO model extraction method and summarizes the existing hybrid annealing methods for solving a large-sized QUBO model; Section 4 first proves the ideal subQUBO model extraction with illustrative examples, then proposes a hybrid annealing method based on it; Section 5 demonstrates the results of the CPU simulation evaluations and those using a quantum annealing machine and discusses these results; and finally, Section 6 concludes the paper and provides an idea of the future works.

2 Preliminaries

2.1 QUBO Model

A QUBO model is represented by Eq. (1) using a binary variable x_i that takes 0 or 1 ($x_i = 0$ or $x_i = 1$)

$$f(X) = \sum_i a_i x_i + \sum_{i \neq j} b_{ij} x_i x_j \tag{1}$$

where $X = \{x_1, x_2, \cdots\}$ is a set of binary variables; a_i is the external magnetic field coefficient representing the force on the binary variable x_i; and b_{ij} is the interaction coefficient representing the weight of the connection between the binary variables x_i and x_j. A *solution* of a QUBO model is one of the combinations of binary variables. The *ground-state solution* of the QUBO model is the combination of binary variables that minimizes $f(X)$. The value of $f(X)$ for a given combination of binary variables is called the *objective value*.

In general, the combinatorial optimization problem can be mapped onto the problem of finding a combination of binary variables that minimizes Eq. (1) [1], [14], [15].

2.2 Ising Machines and Their Size Limitations

An Ising machine is a computer that specializes in finding combinations of binary variables that minimize the objective value of the QUBO model. Ising machines include, for example, D-Wave Systems' quantum annealing machines [4], [16], [17], Hitachi's CMOS annealing machines [18], [19], and Fujitsu's digital annealers [5], [6]. D-Wave machines are based on quantum annealing. CMOS annealing machines and digital annealers are based on simulated annealing. Ising machines based on computational principles different from quantum annealing and simulated annealing include NTT's coherent Ising machines [20] and Toshiba's simulated bifurcation machines [21].

Although the D-Wave 2000Q machine [4] has approximately 2000 qubits, it can only deal with a maximum of 64 binary variables of a QUBO model if all the binary variables are connected to each other due to hardware limitations [7], [8]. Even for the CMOS annealing machines with approximately 100,000 bits, they can only deal with a maximum of approximately 300 binary variables when they are fully connected in the QUBO model [22]. The digital annealers, coherent Ising machines, and simulated bifurcation machines have similar hardware limitations.

3 Hybrid Annealing Method

3.1 General Approach

This subsection describes a general approach for finding a (quasi-)ground-state solution of a QUBO model, whose size is larger than the Ising machine hardware.

In the QUBO model represented by Eq. (1), let $X = \{x_1, x_2, \cdots, x_n\}$ be a set of binary variables. The set X contains n binary variables (the size of the QUBO model is n at this time), and the number of binary variables dealt with in the Ising machine is m ($< n$). Algorithm 1 presents a basic hybrid annealing approach.

First, the binary variable set X is initialized by setting each binary variable to 0 or 1 randomly (line 2). X gives a

tentative solution (line 3). The following processes are repeated while the solution does not converge (lines 4–6). Based on the tentative solution X, a subQUBO model with the size m is extracted from the QUBO model with the size n (line 5). A (quasi-)ground-state solution of the subQUBO model is found by an Ising machine. In this process, the $(n-m)$ variables are set to be the fixed values (line 6). By repeating this process, we obtain a (quasi-)ground-state solution for the original QUBO model with the size n by iteratively obtaining the (quasi-)ground-state solutions of subQUBO models with the size m using an Ising machine.

Suppose that for a set $X = \{x_1, x_2, \cdots, x_n\}$ of binary variables, we are given a certain combination of binary variables called a *tentative solution*. Let $\hat{X} = \{\hat{x_1}, \hat{x_2}, \cdots, \hat{x_n}\}$ be a tentative solution, where $\hat{x_i}$ is 0 or 1. Let $S \subset X$ be a subset of binary variables to be extracted as a subQUBO model. We now consider the interaction coefficient between variables $x_i \in S$ and $x_j \in X \backslash S$. As mentioned above, we have a tentative solution in which $x_j = \hat{x_j}$. For the quadratic term $b_{ij}x_ix_j$ consisting of variables x_i and x_j in Eq. (1), $b_{ij}x_ix_j = (b_{ij}\hat{x_j})x_i = c_ix_i$ ($c_i = b_{ij}x_j$), and the quadratic term $b_{ij}x_ix_j$ can be written as a linear term of x_i because x_j is fixed to be $\hat{x_j}$, and b_{ij} is a constant. In addition, for $x_j, x_k \in X \backslash S$, the linear term of x_j (or x_k) and the quadratic term of x_jx_k can be replaced with $x_j = \hat{x_j}$ and $x_k = \hat{x_k}$ using the tentative solution (i.e., they all become constants). Consequently, when we fix the binary variable $x_j \in X \backslash S$ using a tentative solution, Eq. (1) can be re-written as Eq. (2)

$$f_s(S) = \sum_{x_i \in S} c_i x_i + \sum_{\substack{i \neq j \\ x_i, x_j \in S}} b_{ij} x_i x_j + const \tag{2}$$

where

$$c_i = a_i + \sum_{x_j \in X \backslash S} b_{ij} \hat{x}_j, \tag{3}$$

$$const = \sum_{x_i \in X \backslash S} a_i \hat{x}_i + \sum_{\substack{i \neq j \\ x_i, x_j \in X \backslash S}} b_{ij} \hat{x}_i \hat{x}_j \tag{4}$$

Eq. (2) shows the subQUBO model extracting only the binary variables in S from Eq. (1) assuming the tentative solution $\hat{X}$.

Algorithm 1. Basic Hybrid Annealing Approach

1: **procedure** BASIC HYBRID ANNEALING APPROACH
2: $X \leftarrow$ Initialize(QUBO)
3: $X_{best} \leftarrow X$
4: **while** not converged **do**
5: subQUBO $\leftarrow$ Extract(QUBO, m, X)
6: $X \leftarrow$ Optimize(subQUBO, X)
▷ given the fixed $n - m$ variables
7: **return** $f(X_{best}), X_{best}$

3.2 Existing subQUBO Model Extraction Methods

This subsection summarizes the existing subQUBO model extraction methods other than a simple random subQUBO model extraction method.

Ref. [9] proposed a method based on *k-opt local search* [12], [13]. In that study, a method for extracting a subQUBO model based on relatively good solutions reached in the past was also proposed. This method utilized *path relinking* [23]. However, the extracted subQUBO models tend to be fixed because they depend only on the existing solutions.

In [10], an extracting method, called *qbsolv*, based on *impact values* was proposed. The impact value is defined as how much the objective function value is increased (i.e., less optimal) when the variable is negated from the current solution. By extracting the subQUBO model according to the impact value and updating the current solution, an effective search is expected. However, the impact values depend only on the current solution; hence, the extracted subQUBO model also tends to be fixed, falling into a local optimal solution.

In [11], a genetic algorithm (GA) was first used to explore the solution space in a given QUBO model by running a classical computer. The subQUBO model was then constructed based on the final population in the GA process. This method must run the GA process until it sufficiently converges; thus, it is impractical to apply it to large-sized QUBO models.

Note that several studies proposed subQUBO model extraction methods for specific problems, such as the maximum-clique problem in [24], [25], [26], the constraint satisfaction problem in [27], and the maximum constraint satisfaction problem in [28]. However, these methods can only be applied to these specific problems.

4 Proposed Hybrid Annealing Method

The subQUBO model extraction methods discussed in Section 3.2 are all heuristic and do not necessarily have a theoretical background. On the contrary, the theoretically ideal property of the subQUBO model extraction can be obtained by focusing on the difference between the tentative solutions and the ground-state solution of a QUBO model, as will be proven in Section 4.1. It is quite important to utilize the theoretical background to extract the subQUBO models and then find globally optimal solutions in the original QUBO model.

In this section, we first focus on the difference between the tentative solution and the ground-state solution of a QUBO model and theoretically prove the ideal property of the subQUBO model extraction. Based on this theorem, we then propose a subQUBO model extraction method and a hybrid annealing method that utilizes it.

4.1 Theoretical Background and Strategy

Assuming a small-sized QUBO model exapmle, we considered the ideal subQUBO model extraction. For illustrative purposes, the QUBO model of Eq. (1) is shown as an undirected graph. For example, when the number of binary variables in the QUBO model, called the QUBO model size, is 5, Eq. (1) is shown as in Fig. 1, where every node shows the binary variable x_i, and the edge shows the node interaction. Coefficients a_i and b_{ij} in Eq. (1) are associated with every node and edge, respectively. We assumed that the QUBO model size is $n = 5$. The coefficients were assigned as shown in Fig. 1. Moreover, the computable variable size of an Ising machine is $m = 3$.

Suppose that we obtain a tentative solution of ($x_1 = 1$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$, $x_5 = 0$) as shown in Fig. 2. The

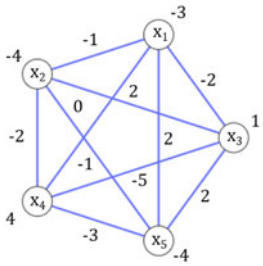


Fig. 1. QUBO model.

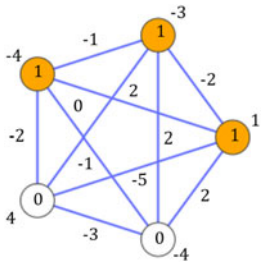


Fig. 2. A tentative solution of the QUBO model.

objective value of the QUBO model given by Eq. (1) is (-7) in this case. In Fig. 2, the orange nodes indicate that their values are equal to the ground-state solution. When all the variables became one for this QUBO model, its objective value was minimized to (-14).

We assume that x_1, x_3, and x_5 are extracted as a subQUBO model from Fig. 2, as shown in the red box in Fig. 3a, where x_5 is included in the extracted subQUBO model but x_4 is not included in the extracted subQUBO model. x_4 and x_5 are binary variables with a different value from the ground-state solution of the original QUBO model. When the ground-state solution of this subQUBO model is obtained as in Fig. 3b, we have $x_1 = 1, x_3 = 0$ and $x_5 = 1$. The objective value of the original QUBO model becomes (-11). In this case we cannot obtain the ground-state solution of the original QUBO model. The value of x_3 must be $(+1)$ in the ground-state solution of the original QUBO model.

We further assume that x_3, x_4 and x_5 are extracted as a subQUBO model from Fig. 2, as shown in the red box in Fig. 4a, where both x_4 and x_5 are included in the extracted subQUBO model. When the ground-state solution of this subQUBO model is obtained as in Fig. 4b, we have $x_3 = 1$, $x_4 = 1$, and $x_5 = 1$, which leads to the objective value of (-14), the ground state of the original QUBO model. Even if x_2, x_4, and x_5 are extracted as a subQUBO model, and its ground-state solution are obtained, we can also obtain the ground-state solution of the original QUBO model.

The examples imply the following property: assume that we extract binary variables, including *all* the binary variables that currently have a different value from the ground-state solution and construct a subQUBO model including all these binary variables. Obtaining the ground-state solution of the extracted subQUBO model also gives the ground-state solution of the original QUBO model. This property leads to a very strong insight to the extraction of the subQUBO models.

This property can be generalized as the following theorem:

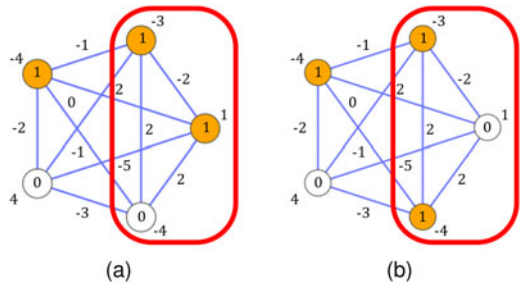


Fig. 3. x_1, x_3, and x_5 are extracted as a subQUBO model (a). The ground state of the extracted subQUBO model is obtained, where $x_1 = 1$, $x_3 = 0$, and $x_5 = 1$ (b).

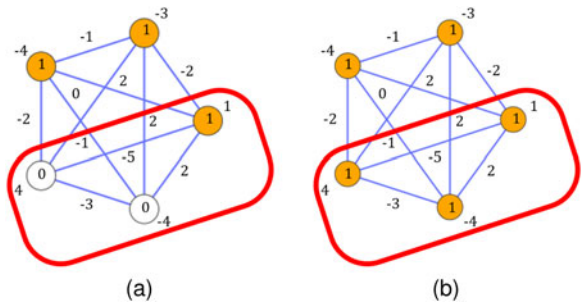


Fig. 4. x_3, x_4, and x_5 are extracted as a subQUBO model (a). The ground state of the extracted subQUBO model is obtained, where $x_3 = 1$, $x_4 = 1$, and $x_5 = 1$ (b).

Theorem 1. *Consider a subQUBO model obtained by extracting a subset $S \subset X$ of binary variables from a QUBO model consisting of a set X of binary variables. Let $\hat{X}$ be a tentative solution of the QUBO model and $X^* = \{x_i^*\}$ be the ground-state solution. Let $X_s^* = \{s_i^*\}$ for every $x_i \in S$ be the ground-state solution of the extracted subQUBO model. In the tentative solution $\hat{X}$ of the QUBO model, let N be the set of binary variables with different values from the ground-state solution. In this case, if $S \supseteq N$, then $f_s(X_s^*) = f(X^*)$.*

Proof. From Eq. (2), the ground-state objective value $f_s(X_s^*)$ of the subQUBO model is presented as

$$\begin{aligned} f_s(X_s^*) = &\sum_{x_i \in S} a_i s_i^* + \sum_{\substack{x_i \in S \\ x_j \in X \backslash S}} b_{ij} s_i^* \hat{x}_j \\ &+ \sum_{\substack{i \neq j \\ x_i, x_j \in S}} b_{ij} s_i^* s_j^* + \sum_{x_i \in X \backslash S} a_i x_i \\ &+ \sum_{\substack{i \neq j \\ x_i, x_j \in X \backslash S}} b_{ij} x_i x_j \end{aligned} \tag{5}$$

In the same manner, from Eq. (1), the ground-state objective value $f(X^*)$ of the QUBO model is presented as

$$\begin{aligned} f(X^*) = &\sum_{x_i \in S} a_i x_i^* + \sum_{x_i \in X \backslash S} a_i x_i^* + \sum_{\substack{x_i \in S \\ x_j \in X \backslash S}} b_{ij} x_i^* x_j^* \\ &+ \sum_{\substack{i \neq j \\ x_i, x_j \in S}} b_{ij} x_i^* x_j^* + \sum_{\substack{i \neq j \\ x_i, x_j \in X \backslash S}} b_{ij} x_i^* x_j^* \end{aligned} \tag{6}$$

If $x_i \in X \backslash S$, then $x_i \in X \backslash N$ because $S \subseteq N$. If $x_i \in X \backslash N$, then $x_i = x_i^*$. The following equations then hold

$$a_i x_i = a_i x_i^* \quad (x_i \in X \backslash S) \tag{7}$$

$$b_{ij} x_j = b_{ij} x_j^* \quad (x_j \in X \backslash S) \tag{8}$$

$$b_{ij} x_i x_j = b_{ij} x_i^* x_j^* \quad (x_i, x_j \in X \backslash S) \tag{9}$$

Therefore, the ground-state objective value $f_s(X_s^*)$ of the subQUBO model becomes

$$\begin{aligned} f_s(X_s^*) = & \sum_{x_i \in S} a_i s_i^* + \sum_{\substack{x_i \in S \\ x_j \in X \backslash S}} b_{ij} s_i^* x_j^* \\ & + \sum_{\substack{i \neq j \\ x_i, x_j \in S}} b_{ij} s_i^* s_j^* + \sum_{x_i \in X \backslash S} a_i x_i^* \\ & + \sum_{\substack{i \neq j \\ x_i, x_j \in X \backslash S}} b_{ij} x_i^* x_j^* \end{aligned} \tag{10}$$

If $f_s(X_s^*) > f(X^*)$, then $X_s^* = \{s_i^*\}$ does not give the ground-state solution of the subQUBO model, contradicting the assumption that X_s^* is the ground-state solution of the subQUBO model. If $f_s(X_s^*) < f(X^*)$, then $X^* = \{x_i^*\}$ does not give the ground-state solution of the QUBO model, contradicting the assumption that X^* is the ground-state solution of the QUBO model. In other words, $f_s(X_s^*) = f(X^*)$. □

The above theorem shows that the ideal subQUBO model extraction aims to evaluate the tentative solution and include all variables with a different value from the ground-state solution. However, the variables having a different value from the ground-state solution cannot be clearly determined. Therefore, we will discuss herein how to construct a method to find the variables in the tentative solution that are *likely* to have a different value from the ground-state solution.

When focusing on a specific binary variable x_i, it is very natural that, if it is 0 (or 1) in many quasi-ground-state solutions obtained, its value of the ground-state solution is also likely to be $x_i = 0$ (or $x_i = 1$). In contrast, if the binary variable x_i value is not fixed at 0 or 1 over many quasi-ground-state solutions, it is unlikely to have the same value as the ground-state solution and likely to have a different value from the ground-state solution.

Based on this strategy, we newly design a subQUBO model extraction method in Section 4.2. As shown in that section, the proposed method is based on the abovementioned strategy. Furthermore, it tends not to fix the extracted subQUBO models; hence, we can expect that we can extract various subQUBO models and finally find a better quasi-optimal solution of the original QUBO model.

4.2 Hybrid Annealing Method With Multiple Solution Instances

In this subsection, we propose a subQUBO model extraction method based on the discussion in the previous subsection. We also propose a hybrid annealing method utilizing this subQUBO model extraction.

4.2.1 subQUBO Model Extraction Method

Based on the strategy discussed earlier, we first consider a situation in which we obtained $N_I \geq 3$ quasi-ground-state solutions of the original QUBO model, called *solution instances*. These solution instances can be obtained by running any QUBO solver on classical computers, such as PyQUBO [29] and the ocean library [30] to the original QUBO model. We then execute the following steps:

(Step 1) Focusing on a particular binary variable x_i of the solution instances, we count the number of solution instances with $x_i = 1$ in these N_I solution instances. This count is calculated for each binary variable in the original QUBO model.

(Step 2) When the count of a binary variable x_i is equal to $N_I/2$, it is called the most *varied*. When the count of a binary variable x_i is equal to N_I or 0, it is called the least *varied*. We sort the binary variables from the most varied to the least varied and construct the subQUBO model in the sorted order while its size not exceeding the subQUBO size limit.

Using this method, we can collect the binary variables not fixed at 0 or 1 into the extracted subQUBO model.

However, the proposed method leads to the fixed subQUBO model because we always use N_I solution instances. Hence, we randomly pick up N_S $(2 \leq N_S < N_I)$ solution instances from the N_I solution instances and construct the subQUBO model from them.

In the abovementioned method, we only focus on the N_S solution instances. The $(N_I - N_S)$ solution instances are not considered. The pick-up process is repeated N_E times to fully utilize all the solution instances. We also construct N_E subQUBO models. We can construct various patterns of subQUBO models; thus, we can obtain various solution instances by adding their solution into the instance pool. Accordingly, we expect that a better quasi-optimal solution of the original QUBO model can be finally obtained. Note that when we have N_E or more Ising machines, we can simultaneously obtain (quasi-)ground-state solutions for these N_E subQUBO models.

4.2.2 Proposed Hybrid Annealing Method

Algorithm 2 presents the proposed hybrid annealing method based on the subQUBO model extraction. In this method, n and m show the QUBO model size and subQUBO model size, respectively. In the proposed method, we first prepare a predetermined number N_I of the solution instances and register it in the instance pool (lines 2–4). Among the solution instances in the instance pool, that with the best objective value is selected as the best solution X_{best} (line 5). The following process is repeated while the solution does not converge (lines 6–19). A quasi-ground-state solution is found by running any QUBO solver on a classical computer for every solution instance in the instance pool. At that time, the previously obtained solution is used as an input to the QUBO solver (lines 7 and 8). The N_E subQUBO models are extracted using the subQUBO model extraction method described above (lines 9–17).

To extract the subQUBO model, the N_S solution instances $(X_1, X_2, \cdots, X_{N_S})$ are randomly selected from the instance pool (line 10). Let $x_{k,j}$ be the jth binary variable in the kth solution instance. Focusing on the jth binary variable over the N_S selected solution instances, we count the number of solution instances, in which its value is one. Let

c_j be the count. The absolute difference between c_j and $N_S/2$ is then calculated (lines 11–14). In the ascending order of this absolute difference, the binary variables are extracted as the subQUBO model, whose size does not exceed m. At that time, a solution X_t randomly selected from the N_S solution instances is used as a tentative solution (line 15). For every subQUBO model extracted, a (quasi-)ground-state solution is found by an Ising machine, which is combined with X_t. We then obtain a new solution X' (line 16). We add it into the instance pool as a new solution instance (line 17).

Algorithm 2. Proposed Hybrid Annealing Method

```
 1: procedure PROPOSED METHOD WITH MULTI-INSTANCES
 2:   for (i = 1; i ≤ N_I; i++) do
 3:     X_i ← Initialize(QUBO)
 4:     Pool ← AddInstancePool(Pool, X_i)
 5:   X_best ← FindBest(Pool)
 6:   while not converged do
 7:     for (i = 1; i ≤ N_I; i++) do
 8:       X_i ← Optimize(QUBO, X_i)
                                  ▷ Using a classical computer
 9:     for (i = 1; i ≤ N_E; i++) do
10:       X_1, X_2, ···, X_{N_S} ← SelectInstance(Pool, N_S)
11:       for (j = 1; j ≤ n; j++) do
12:         for (k = 1; k ≤ N_S; k++) do
13:           c_j ← c_j + x_{k,j}
14:         d_j ← |c_j − N_S/2|
15:       subQUBO ← Extract(ArgSort(d_1, d_2, ···, d_n), m, X_t)
16:       X' ← Optimize(subQUBO, X_t)
                                  ▷ Using an Ising machine
17:       Pool ← AddInstancePool(Pool, X')
18:     X_best ← FindBest(Pool)
19:     Pool ← ArrangeInstancePool(Pool, N_I)
20:   return f(X_best), X_best
```

After these steps, X_{best} is updated to the best solution in the instance pool (line 18). Finally, the top N_I solution instances with good objective values from the instance pool are redefined as the instance pool (line 19).

5 Experimental Evaluations

5.1 Evaluation Purpose

The following four items are evaluated in this section:

1) performance of the proposed method with respect to three parameters (i.e., N_I, N_E and N_S) under a CPU simulation;
2) comparison of the performance of the proposed method to that of the existing methods under a CPU simulation;
3) performance of the proposed method when using a real Ising machine; and
4) comparison of the performance of the proposed method to that of the existing methods using a real Ising machine.

First, we clarified the relationship among the three parameters, namely the number N_I of solution instances, the number N_E of subQUBO models to be extracted, and the number N_S of solution instances to be selected for the subQUBO model extraction, and the obtained (quasi-) ground-state solutions/processing time. N_I was specifically set to three patterns, that is, 20, 40, and 60. Similarly, N_E was set to three patterns, that is, $N_I/4$, $N_I/2$, and N_I. Meanwhile, N_S was set to three patterns depending on the N_I value, that is, 2, $N_I/4$, and $N_I/2$, because its maximum value is N_I itself. We tried a total of 27 patterns.

Next, we compared the proposed method with the existing methods, including the simplest subQUBO model extraction method, which randomly selects variables (henceforth referred to as the *random method*), and qbsolv [10], which is the most effective in real-world problems, such as in [1], from the viewpoints of the obtained quasi-ground-state solutions and processing time.

An Ising machine was simulated on the CPU for evaluations (1) and (2). For evaluations (3) and (4), a real Ising machine was used. We evaluated several conditions as above in Evaluation (1) and verified that the proposed method was effective, even when using a real Ising machine in Evaluation (3). Evaluation (2) compared the proposed method to the existing ones. Finally, in Evaluation (4), we implemented the existing methods on a real Ising machine and compared the results to the proposed method, which showed the effectiveness of the proposed method.

5.2 Implementation

5.2.1 Implementation Environment

For evaluations (1) and (2), the Ising machine was simulated by the tabu search [31] on the CPU of a local machine. The local machine used was an Intel Xeon (CPU: 28 cores, 2.60 GHz/ RAM: 96GB). We used the function, hybrid.samplers.InterruptableTabuSampler(), provided by the ocean library [30]. This function runs the annealing process until the solution is well converged. For evaluations (3) and (4), D-Wave 2000Q machine was used as an Ising machine. When using the D-Wave 2000Q machine, one subQUBO model was annealed once for each trial. We used the function, dwave.system. AutoEmbeddingComposite(dwave.system.DWaveSampler()).sample(num_reads=1), provided by the ocean library [30]. The Ising machine under the CPU simulation and the D-Wave 2000Q machine were set to be able to deal with subQUBO models of up to 50 variables ($m = 50$).[1] The implementation language was Python. We also used the NumPy library [32].

5.2.2 Implementation of the Proposed Method

In this implementation, the tabu search process was performed by the CPU to obtain each solution instance in the instance pool during 0.010 [s], which was short enough, but led to reasonable solutions, in lines 7 and 8 in Algorithm 2. We used the function tabu.TabuSampler(), provided by the ocean library [30].

In Algorithm 2, we used the average hamming distance between the solution instances in the instance pool to judge if the solution is well converged or not. When the average hamming distance was less than or equal to the Ising machine size ($= 50$), we considered the search to be sufficiently completed, and we terminated the process.

1. As mentioned in Section 2.2, D-Wave 2000Q machine logically deals with a maximum of 64 variables. However, several qubits cannot be used due to defects; thus, the effective number of binary variables dealt with becomes less than 64. We then set the maximum subQUBO size $m = 50$ in this experiment.

5.2.3 Implementation of the Random Method

Algorithm 3 presents the implemented random method. In this method, each variable in the variable set X was initialized to 0 or 1 (line 2), and the tentative solution was prepared (line 3). The following steps were repeated until the solution is converged (lines 4–9). The tabu search process was executed for 0.010 [s] as in the proposed method (line 5). Accordingly, m random variables were extracted into the subQUBO model (line 6). m is the number of variables that can be computed by an Ising machine. The (quasi-) ground-state solution of the subQUBO model was obtained. If it was better than the tentative solution, the tentative solution was updated (lines 7–9). The convergence condition was set to the case where the tentative solution in lines 8 and 9 was not updated c_n times consecutively.[2]

Algorithm 3. Random Method

```
1: procedure RANDOM METHOD
2:   X ← Initialize(QUBO)
3:   X_best ← X
4:   while not converged do
5:     X ← TabuSeach(QUBO, X)
6:     subQUBO ← RandomExtract(QUBO, m, X)
7:     X ← Optimize(subQUBO, X)
8:     if f(X) < f(X_best) then
9:       X_best ← X
10:  return f(X_best), X_best
```

5.2.4 Implementation of Qbsolv

Algorithm 4 presents the qbsolv method [10], in which each binary variable in a variables set X was initialized to 0 or 1 (line 2), and the variable set after the tabu search process was set as the tentative solution (line 3). For the tentative solution, the variable indices were sorted in the impact values of the variables. The following process was repeated until the solution is converged (lines 5–13). Multiple subQUBO models were extracted by decomposing the binary variables in the order of the sorted indices (line 6–7). The (quasi-)ground-state solutions of the extracted subQUBO models were then obtained. These solutions were composed as the temporal solution of the original QUBO model (lines 8 and 9). As in lines 3 and 4, for the temporal solution, the tabu search was executed, and the variable indices were sorted in the impact values (lines 10 and 11). Subsequently, the temporal solution was compared with the tentative solution. If it was better than the tentative solution, then the tentative solution was updated (lines 12 and 13). As in the random method, the convergence condition was set to the case where the tentative solution was not updated c_n times consecutively in lines 12 and 13.

5.2.5 Implementation of Direct QUBO Search

When the Ising machine is simulated by the CPU as in evaluations (1) and (2), it can deal with a larger QUBO model. In order to evaluate this case, we input an original large-sized QUBO model directly into the CPU-simulated Ising machine. In this case, we also used the function, hybrid. samplers.InterruptableTabuSampler(), provided by the ocean library [30]. This function runs the annealing process until the solution is well converged.

Algorithm 4. Qbsolv[10]

```
1: procedure QBSOLV
2:   X ← Initialize(QUBO)
3:   X_best ← TabuSeach(QUBO, X)
4:   index ← OrderByImpact(QUBO,X_best)
5:   while not converged do
6:     for (i = 0; i < Size(QUBO); i+ = Size(subQUBO)) do
7:       subQUBO ← Decompose(QUBO,
                    index[i : i+Size(subQUBO)-1], X_best)
8:       subX ← Optimize(subQUBO, X_best)
9:       X[index[i : i+Size(subQUBO)-1]] ← subX
10:    X ← TabuSearch(QUBO, X)
11:    index ← OrderByImpact(QUBO, X)
12:    if f(X) < f(X_best) then
13:      X_best ← X
14:  return f(X_best), X_best
```

5.3 Measurement Results

This subsection presents the measurement results of the proposed method and the existing methods. In these measurements, three quadratic assignment problems (QAPs), namely tai20a, tho30 and tho40 from QAPLIB [33] were converted into QUBO models and used. When the QAPs were converted into QUBO models, their variable sizes were 400, 900, and 1600, respectively, which are larger than the maximum variable size of the Ising machine.

We measured the number of loops, solution accuracy, processing time on the Ising machine and local search. The number of loops is the number of while loops required for the solution convergence. The solution accuracy is defined as $f(X)/f(X^*)$, where the resulting output solution is X, and the ground-state solution of the QUBO model is X^*. X^* can be obtained from QAPLIB [33].

The processing time of the Ising machine is the total time spent to search for the (quasi-)ground-state solutions of the subQUBO models. For the CPU simulation, it is the time of the tabu search process for the subQUBO models, while in the case of using the real Ising machine, it is the time devoted to the annealing process.[3] The local search time is the total execution time of the tabu search for the entire original QUBO model. We measured them 50 times. The mean values were summarized in the results. The best accuracies over 50 trials were also shown in Tables 1, 2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13, and 14.

Tables 1, 2, 3, 4, 5, 6, 7, 8, and 9 show the measurement results of the proposed method for the QUBO models for Evaluation (1). Tables 1, 2, and 3 present the results for tai20a with N_I as 20, 40, and 60. Similarly, Tables 4, 5, and 6 show the results for tho30, and Tables 7, 8, and 9 present the

2. As discussed in Sections 5.3 and 5.4, we have tried various c_n values and determined that $c_n = 3$ is the best. No further improvement in solution accuracy can be seen, even if we set $c_n = 4$ or more in the random method and qbsolv.

3. Other than the annealing time, pre-process time such as QPU (Quantum Processing Unit) cooling time is needed for D-Wave machine. However, it is not directly related to annealing process time [34], [35]. In this experiment, we measured the anneal time and evaluated the effectiveness of the proposed method.

TABLE 1
Results of the Proposed Method for tai20a (CPU Simulation): $N_I = 20$

Conditions					Results			
Problem	QUBO model size	N_I	N_E	N_S	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tai20a	400 variables	20	5	2	6.82	0.966 (0.975)	2.696	1.364
				5	7.56	0.969 (0.989)	2.964	1.512
				10	7.58	0.970 (0.983)	2.967	1.516
			10	2	6.40	0.969 (0.981)	5.090	1.280
				5	7.16	0.975 (0.989)	5.690	1.432
				10	7.08	0.971 (0.990)	5.637	1.416
			20	2	6.22	0.972 (0.987)	9.647	1.244
				5	6.84	0.977 (0.990)	10.618	1.368
				10	6.54	0.973 (0.989)	10.248	1.308

TABLE 2
Results of the Proposed Method for tai20a (CPU Simulation): $N_I = 40$

Conditions					Results			
Problem	QUBO model size	N_I	N_E	N_S	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tai20a	400 variables	40	10	2	7.20	0.970 (0.988)	5.694	2.880
				10	8.22	0.976 (0.989)	6.602	3.288
				20	7.98	0.971 (0.989)	6.415	3.192
			20	2	6.56	0.972 (0.982)	10.520	2.624
				10	8.22	0.980 (0.991)	13.040	3.288
				20	7.46	0.975 (0.991)	11.955	2.984
			40	2	6.22	0.975 (0.987)	20.041	2.488
				10	7.20	0.981 (0.998)	22.653	2.880
				20	6.78	0.979 (0.991)	21.328	2.712

TABLE 3
Results of the Proposed Method for tai20a (CPU Simulation): $N_I = 60$

Conditions					Results			
Problem	QUBO model size	N_I	N_E	N_S	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tai20a	400 variables	60	15	2	7.16	0.971 (0.979)	8.707	4.296
				15	9.30	0.979 (0.991)	11.116	5.580
				30	8.52	0.973 (0.989)	10.219	5.112
			30	2	6.62	0.973 (0.987)	16.038	3.972
				15	8.28	0.982 (0.990)	19.915	4.968
				30	8.32	0.976 (0.990)	19.852	4.992
			60	2	6.46	0.974 (0.987)	30.391	3.876
				15	7.22	0.982 (0.991)	34.255	4.332
				30	7.00	0.979 (0.991)	33.511	4.200

TABLE 4
Results of the Proposed Method for tho30 (CPU Simulation): $N_I = 20$

Conditions					Results			
Problem	QUBO model size	N_I	N_E	N_S	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tho30	900 variables	20	5	2	7.66	0.955 (0.969)	3.055	1.532
				5	6.00	0.953 (0.968)	2.281	1.200
				10	6.40	0.955 (0.969)	2.430	1.280
			10	2	7.06	0.958 (0.969)	5.454	1.412
				5	6.62	0.956 (0.970)	5.098	1.324
				10	6.34	0.955 (0.968)	4.921	1.268
			20	2	6.78	0.956 (0.976)	10.295	1.356
				5	7.20	0.955 (0.969)	10.963	1.440
				10	6.86	0.955 (0.969)	10.436	1.372

results for tho40. The problem name, size of the QUBO model, N_I, N_E, N_S, number of loops, solution accuracy, and processing time on the CPU-simulated Ising machine and the local search are indicated in each table.

Tables 10, 11, and 12 show the results of the random method, qbsolv, and direct QUBO search for Evaluation (2). Each table presents the results for each QUBO model, where the problem name, size of the QUBO model, method, number of loops, solution accuracy, and processing time on the CPU-simulated Ising machine and the local search are also presented from the leftmost to the rightmost column.

Fig. 5 summarizes the mean accuracies when the parameter c_n is varied in the random method and qbsolv. Figs. 6, 7, and 8 show the boxplot of the data from evaluations (1) and (2). The boxplot shows the distribution of the obtained data. The box extends from the first quartile to third quartile with the line inside indicating the median. The whisker shows the maximum and minimum values. In these figures, we set $c_n = 3$ for the random method and qbsolv based on the discussion in Section 5.4.

Table 13 shows the measurement results of the proposed method for Evaluation (3),[4] including the problem name, size of the QUBO model, N_I, N_E, N_S, number of loops, solution accuracy, and annealing time on the real Ising machine and the local search time from the leftmost to the rightmost column. How to setup the parameters in Table 13 is discussed in Section 5.4.

4. In Table 13, the total annealing time on D-Wave 2000Q, t_a [ms], is the same as the processing time on local search, t_l [s], because of the following reason. In D-Wave 2000Q, every annealing requires 0.020 [ms]. t_a [ms] is given by $t_a[\mathrm{ms}] = N_E \times \#Loop \times 0.020[\mathrm{ms}]$. t_l [s] is given by $t_l[\mathrm{s}] = N_I \times \#Loop \times 0.010[\mathrm{s}]$ because every tabu search requires 0.010 [s], as discussed in Section 5.2.2. We set $N_I = 2 \times N_E$, thus, we have $t_a[\mathrm{ms}] = t_l[\mathrm{s}]$.

TABLE 5
Results of the Proposed Method for tho30 (CPU Simulation): $N_I = 40$

Conditions					Results			
Problem	QUBO model size	N_I	N_E	N_S	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tho30	900 variables	40	10	2	7.26	0.959 (0.970)	5.476	2.904
				10	5.82	0.958 (0.968)	4.393	2.328
				20	5.58	0.959 (0.969)	4.169	2.232
			20	2	7.88	0.961 (0.971)	11.827	3.152
				10	6.90	0.958 (0.973)	10.561	2.760
				20	6.50	0.957 (0.970)	9.925	2.600
			40	2	7.64	0.959 (0.972)	23.146	3.056
				10	7.52	0.959 (0.971)	22.718	3.008
				20	6.90	0.957 (0.968)	20.875	2.760

TABLE 6
Results of the Proposed Method for tho30 (CPU Simulation): $N_I = 60$

Conditions					Results			
Problem	QUBO model size	N_I	N_E	N_S	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tho30	900 variables	60	15	2	7.68	0.960 (0.973)	8.839	4.608
				15	5.90	0.959 (0.968)	6.751	3.540
				30	5.66	0.958 (0.968)	6.489	3.396
			30	2	8.28	0.964 (0.976)	18.865	4.968
				15	6.24	0.960 (0.969)	14.306	3.744
				30	6.16	0.960 (0.968)	14.158	3.696
			60	2	9.32	0.963 (0.973)	42.907	5.592
				15	6.22	0.957 (0.969)	28.748	3.732
				30	6.26	0.956 (0.968)	29.078	3.756

TABLE 7
Results of the Proposed Method for tho40 (CPU Simulation): $N_I = 20$

Conditions					Results			
Problem	QUBO model size	N_I	N_E	N_S	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tho40	1600 variables	20	5	2	10.94	0.964 (0.975)	4.172	2.188
				5	8.88	0.959 (0.973)	3.399	1.776
				10	9.52	0.960 (0.973)	3.687	1.904
			10	2	10.46	0.965 (0.977)	7.971	2.092
				5	11.00	0.963 (0.979)	8.385	2.200
				10	9.86	0.961 (0.971)	7.556	1.972
			20	2	10.08	0.964 (0.977)	15.518	2.016
				5	9.78	0.964 (0.979)	14.974	1.956
				10	9.34	0.963 (0.978)	14.282	1.868

TABLE 8
Results of the Proposed Method for tho40 (CPU Simulation): $N_I = 40$

Conditions					Results			
Problem	QUBO model size	N_I	N_E	N_S	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tho40	1600 variables	40	10	2	10.88	0.965 (0.976)	8.216	4.352
				10	9.26	0.963 (0.973)	7.026	3.704
				20	8.58	0.961 (0.971)	6.559	3.432
			20	2	13.20	0.971 (0.980)	20.118	5.280
				10	12.04	0.966 (0.981)	18.254	4.816
				20	10.16	0.963 (0.980)	15.470	4.064
			40	2	12.00	0.972 (0.985)	35.363	4.800
				10	11.28	0.966 (0.980)	34.284	4.512
				20	10.60	0.964 (0.983)	32.269	4.240

TABLE 9
Results of the Proposed Method for tho40 (CPU Simulation): $N_I = 60$

Conditions					Results			
Problem	QUBO model size	N_I	N_E	N_S	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tho40	1600 variables	60	15	2	11.52	0.969 (0.981)	13.311	6.912
				15	9.04	0.962 (0.974)	10.512	5.424
				30	9.60	0.964 (0.974)	11.238	5.760
			30	2	13.00	0.969 (0.981)	30.067	7.800
				15	10.40	0.964 (0.984)	24.375	6.240
				30	11.34	0.965 (0.982)	26.691	6.804
			60	2	12.16	0.973 (0.983)	55.035	7.296
				15	12.34	0.969 (0.984)	56.319	7.404
				30	10.66	0.965 (0.976)	48.900	6.396

Table 14 presents the measurement results of the random method and qbsolv for Evaluation (4). The problem name, size of the QUBO model, method, number of loops, solution accuracy, and annealing time on the real Ising machine and local search time are shown.

Fig. 9 shows the boxplot of the data from evaluations (3) and (4). The boxplot shows the distribution of the obtained data, as in Figs. 6, 7, and 8.

5.4 Discussions

This subsection discusses the evaluation results of the proposed method compared to the existing methods. First, we investigate the effect of the three parameters, namely N_I, N_E, and N_S, on the solution accuracy in the proposed method from Tables 1, 2, 3, 4, 5, 6, 7, 8, and 9. Table 15 summarizes the results of the averaged mean solution accuracy and the processing time of the CPU-simulated Ising machine for each

TABLE 10
Results of the Existing Method for tai20a (CPU Simulation)

Conditions				Results			
Problem	QUBO model size	Method	c_n	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tai20a	400 variables	Random method	2	3.12	0.948 (0.963)	0.241	0.031
			3	4.10	0.948 (0.963)	0.325	0.041
			5	6.34	0.948 (0.969)	0.481	0.063
			10	12.50	0.948 (0.965)	0.951	0.125
		qbsolv	2	3.98	0.954 (0.968)	2.589	0.040
			3	5.76	0.957 (0.972)	3.776	0.058
			5	8.10	0.956 (0.968)	5.210	0.081
			10	14.36	0.957 (0.976)	8.552	0.144
		Direct QUBO search	-	-	0.954 (0.966)	1.942	-

TABLE 11
Results of the Existing Method for tho30 (CPU Simulation)

Conditions				Results			
Problem	QUBO model size	Method	c_n	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tho30	900 variables	Random method	2	3.60	0.936 (0.957)	0.287	0.036
			3	4.84	0.940 (0.957)	0.385	0.048
			5	6.68	0.940 (0.957)	0.528	0.067
			10	12.20	0.940 (0.943)	0.954	0.122
		qbsolv	2	4.00	0.938 (0.945)	5.803	0.040
			3	5.48	0.942 (0.974)	7.314	0.055
			5	7.40	0.942 (0.948)	10.720	0.074
			10	13.05	0.940 (0.946)	18.767	0.130
		Direct QUBO search	-	-	0.940 (0.952)	9.758	-

TABLE 12
Results of the Existing Method for tho40 (CPU Simulation)

Conditions				Results			
Problem	QUBO model size	Method	c_n	#Loop	Mean accuracy (Best accuracy)	Processing time on CPU-simulated Ising machine [s]	Processing time on local search [s]
tho40	1600 variables	Random method	2	5.56	0.944 (0.966)	0.426	0.056
			3	5.50	0.947 (0.959)	0.413	0.055
			5	7.96	0.947 (0.963)	0.597	0.080
			10	12.94	0.947 (0.963)	0.960	0.129
		qbsolv	2	5.16	0.946 (0.968)	13.631	0.052
			3	6.40	0.949 (0.969)	15.218	0.064
			5	8.86	0.949 (0.964)	22.007	0.089
			10	14.30	0.949 (0.962)	35.473	0.143
		Direct QUBO search	-	-	0.956 (0.970)	33.737	-

TABLE 13
Results of the Proposed Method When Using D-Wave 2000Q

Conditions						Results		
Problem	QUBO model size	N_I	N_E	N_S	#Loop	Mean accuracy (Best accuracy)	Total annealing time on D-Wave 2000Q [ms]	Processing time on local search [s]
tai20a	400 variables	20	10	5	5.58	0.958 (0.966)	1.116	1.116
tho30	900 variables	20	10	5	9.38	0.955 (0.968)	1.876	1.876
tho40	1600 variables	20	10	5	11.92	0.963 (0.978)	2.384	2.384

TABLE 14
Results of the Existing Method When Using D-Wave 2000Q

Conditions					Results		
Problem	QUBO model size	Method	c_n	#Loop	Mean accuracy (Best accuracy)	Total annealing time on D-Wave 2000Q [ms]	Processing time on local search [s]
tai20a	400 variables	Random method	3	6.02	0.950 (0.969)	0.120	0.060
		qbsolv	3	6.52	0.951 (0.967)	1.043	0.065
tho30	900 variables	Random method	3	7.04	0.944 (0.957)	0.141	0.070
		qbsolv	3	5.90	0.943 (0.956)	2.124	0.059
tho40	1600 variables	Random method	3	9.48	0.945 (0.963)	0.190	0.095
		qbsolv	3	5.98	0.935 (0.950)	3.827	0.060

value of N_I. According to the table, the larger the value of N_I, the better the solution accuracy. The larger the value of N_I, the longer the processing time of the CPU-simulated Ising machine.

Next, we examined the effect of N_E on the solution accuracy and processing time. We calculated the averaged mean solution accuracy and the processing time of the CPU-simulated Ising machine for each value of N_E from Tables 1, 2, 3,

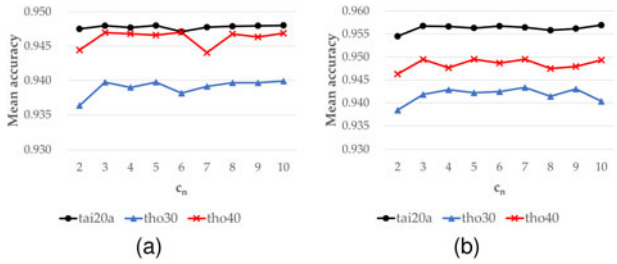


Fig. 5. No further improvement in solution accuracy can be seen, even if we set $c_n = 4$ or more in the random method (a) and qbsolv (b).

4, 5, 6, 7, 8, and 9. Table 16 summarizes the results. According to the table, the larger the value of N_E, the better the solution accuracy and the longer the processing time of the CPU-simulated Ising machine.

Table 17 summarizes the effect of N_S on the performance, where the mean solution accuracies and the processing times of the CPU-simulated Ising machine were averaged for each value of N_S. Table 17 shows that we cannot see well the clear relationship between N_S and the performance.

The results indicate that parameters N_I and N_E should be set large enough as long as the processing time allows. N_S may not affect the solution accuracy, and it may be good to set N_S to a reasonable value according to the available Ising machine hardware and processing time.

We then compared the performances of the proposed method and the existing methods through Tables 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12. For instance, looking at Table 10 for tai20a, the random method showed the mean solution accuracy of 0.948, qbsolv showed the mean solution accuracy of 0.954–0.957, and direct QUBO search showed the mean solution accuracy of 0.954. In the proposed method, the mean solution accuracies ranged from 0.966 to 0.982 for tai20a. The solution accuracy was 0.970 on average in the condition of $N_I = 20$, $N_E = 5$, and $N_S = 10$, which is almost equivalent in the processing time of the CPU-simulated Ising machine in qbsolv under the condition of $c_n = 3$. The similar discussions hold true when looking at the best accuracies. These results indicate that the proposed method can find better quasi-ground-state solutions compared to the existing methods. As we can also see from the distribution of solution accuracies shown in Figs. 6, 7, and 8, the proposed method can obtain better quasi-ground-state solutions compared to the existing methods. As in these figures and tables, the direct QUBO search cannot obtain better

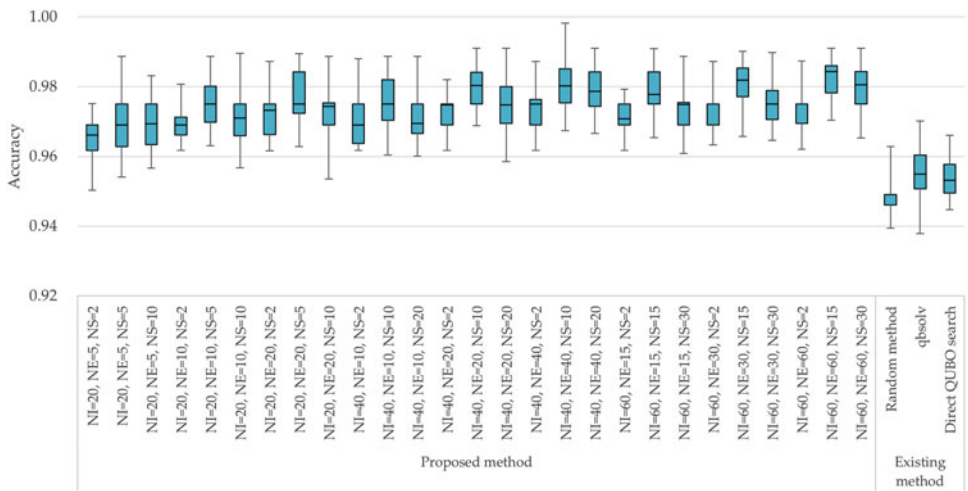


Fig. 6. Boxplot of data from the evaluations for tai20a.

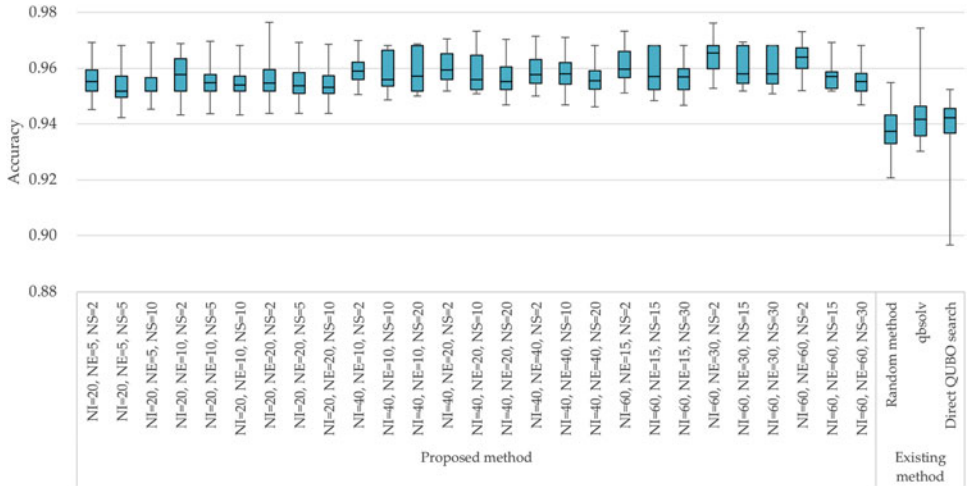


Fig. 7. Boxplot of data from the evaluations for tho30.

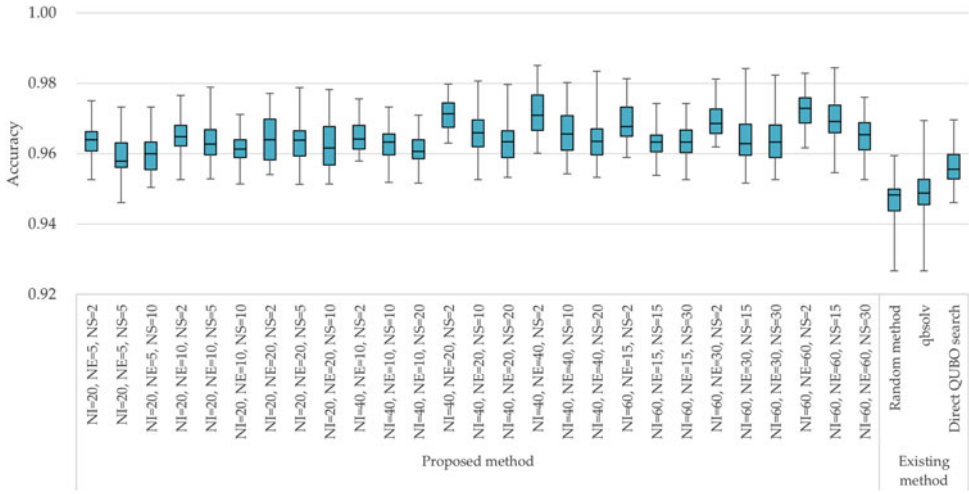


Fig. 8. Boxplot of data from the evaluations for tho40.

results, compared to the proposed method. This is because the CPU-simulated Ising machine cannot anneal the solution enough, if large-sized QUBO models are given. In summary, the proposed method can obtain better quasi-ground-state solutions compared to the existing methods for all the QUBO models of tai20a, tho30, and tho40.

The discussions showed the effectiveness of the proposed method on the CPU simulation. We found that the proposed method can obtain a better solution accuracy than the existing methods even in the case of $N_I = 20$. Table 18 shows the averaged mean solution accuracy and the processing time of the CPU-simulated Ising machine of the proposed method for the nine patterns of N_E and N_S with a fixed $N_I = 20$. Table 18 illustrates that the highest solution accuracy is obtained on $N_I = 20$, $N_E = 20$, and $N_S = 5$; however, the processing time of the CPU-simulated Ising machine becomes relatively long. In the case of $N_I = 20$, $N_E = 10$, and $N_S = 5$, the solution accuracy was almost the same as that of $N_I = 20$, $N_E = 20$, and $N_S = 5$, but the processing time of the CPU-simulated Ising machine can be reduced to half. From this viewpoint, in Evaluation (3) using the real Ising machine, we focused on the parameter set of $N_I = 20$, $N_E = 10$, and $N_S = 5$.

Fig. 5 shows that we cannot see no further improvement in solution accuracy, even if $c_n = 4$ or more are set in the

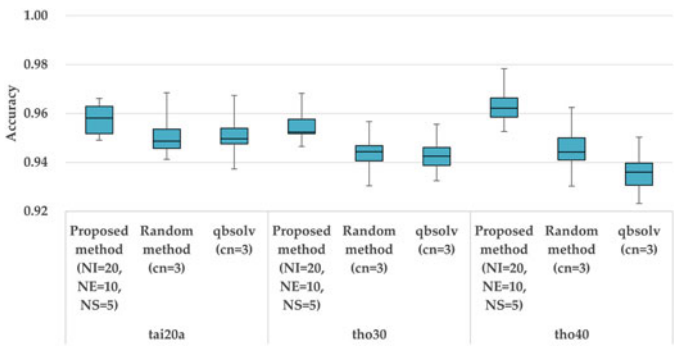


Fig. 9. Boxplot of data from the evaluations when using D-Wave 2000Q.

TABLE 15
Effect of N_I on the Solution Accuracy and Processing Time

N_I	Mean accuracy	Processing time on CPU-simulated Ising machine [s]
20	0.963	7.053
40	0.967	15.183
60	0.968	23.355

TABLE 16
Effect of N_E on the Solution Accuracy and Processing Time

N_I	N_E	Mean accuracy	Processing time on CPU-simulated Ising machine [s]
20	5	0.9614	3.072
	10	0.9638	6.200
	20	0.9642	11.887
40	10	0.9647	6.061
	20	0.9670	13.412
	40	0.9678	25.964
60	15	0.9662	9.687
	30	0.9681	20.474
	60	0.9688	39.905

TABLE 17
Effect of N_S on the Solution Accuracy and Processing Time

N_I	N_S	Mean accuracy	Processing time on CPU-simulated Ising machine [s]
20	2	0.9633	7.100
	5	0.9635	7.152
	10	0.9625	6.907
40	2	0.9670	15.711
	10	0.9674	15.504
	20	0.9652	14.334
60	2	0.9686	24.907
	15	0.9683	22.922
	30	0.9662	22.237

TABLE 18
Effect of N_E, N_S, and Performance Under $N_I = 20$

N_I	N_E	N_S	Mean accuracy	Processing time on CPU-simulated Ising machine [s]
20	5	2	0.9619	3.308
		5	0.9607	2.881
		10	0.9617	3.028
	10	2	0.9640	6.172
		5	0.9648	6.391
		10	0.9624	6.038
	20	2	0.9640	11.820
		5	0.9651	12.185
		10	0.9635	11.655

random method and qbsolv. Therefore, in Evaluation (4) using the real Ising machine, we set $c_n = 3$.

Table 13 shows the performance of the proposed method on the D-Wave 2000Q machine with the parameter set of $N_I = 20$, $N_E = 10$, and $N_S = 5$. The table illustrates that the proposed method can find the same level of quasi-ground-state solutions with the mean accuracy difference of 0.00–0.017 compared to the CPU simulation. Finally, from Tables 13 and 14, we compared the results of the proposed method to the existing methods when using the D-Wave 2000Q machine. Similar to the CPU simulation results, the proposed method can obtain better quasi-ground-state solutions in terms of accuracy compared to the existing methods for all the QUBO models of tai20a, tho30, and tho40. Fig. 9 also shows that the distribution of the solutions obtained by the proposed method is better than that of the existing methods, especially in tho30 and tho40.

6 Conclusion

In this paper, we proposed a hybrid annealing method that extracts a subQUBO model from a large-sized QUBO model on a classical computer and solves the extracted subQUBO model on an Ising machine. The proposed method prepared multiple solution instances and focused on the variation of each binary variable in the solution instances. The (quasi-) ground-state solutions of the extracted subQUBO models were obtained using an Ising machine and pooled into solution instances. By repeating this process, we obtained a (quasi-)optimal solution of the original QUBO model.

From the implementation evaluations of the simulated Ising machine on the CPU, we clarified the effect of the design parameters of the proposed method. In both evaluations of the simulated and real Ising machines, we demonstrated that the proposed method can obtain better solutions compared to the existing methods.

According to the evaluation results, the larger the number of extracted subQUBO models, the better the solution accuracy. In the proposed method, the number of subQUBO models to be extracted can be flexibly set. Therefore, the performance of the proposed method can be maximized by extracting the subQUBO models according to the available Ising machine hardware. Using the maximum number of Ising machine hardware in parallel, we can simultaneously obtain the (quasi-)ground-state solutions of the extracted subQUBO models. This will be an important part of the future work.

In this paper, we picked up the QAPs, typical difficult combinatorial optimization problems, and evaluated the proposed method. However, if a given combinatorial optimization problem instance tends to satisfy Theorem 1, the proposed method may more outperform the existing methods and we can see the more significant benefits from the proposed method. In the future, we will theoretically define such a problem class and evaluate the proposed method furthermore.

Recently, an effective quantum annealing scheme, called ensemble quantum annealing (EQUAL), is proposed in [36]. In the conventional quantum annealing, an optimal solution cannot be obtained due to the *system bias* caused by the noise and imperfections in a quantum annealing machine, even when a single quantum machine instruction (QMI) is executed many times. In EQUAL, QMI is repeatedly executed by adding controlled *perturbations*, called *perturbation Hamiltonian*, and hence it much improves the solution quality. The scheme is effective when using the quantum annealing machine hardware like D-Wave 2000Q and can be combined with the proposed hybrid annealing method. In this case, how to generate an effective perturbation Hamiltonian is the key and this is also an important future work.

References

[1] F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney, "Traffic flow optimization using a quantum annealer," *Front. Inform. Comm. Techno.*, vol. 4, 2017, Art. no. 29. [Online]. Available: https://www.frontiersin.org/article/10.3389/fict.2017.00029

[2] W. Yanling, Y. Deli, and Y. Guoqing, "Logistics supply chain management based on multi-constrained combinatorial optimization and extended simulated annealing," in *Proc. Int. Conf. Logistics Syst. Intell. Manage.*, 2010, pp. 188–192.

[3] T. C. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin, "Multirobot routing algorithms for robots operating in vineyards," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1184–1194, Jul. 2020.

[4] D-Wave Systems Inc., "D-Wave announces D-Wave 2000Q quantum computer," Accessed: Jan., 2021. [Online]. Available: https://www.dwavesys.com/press-releases/d-wave-announces-d-wave-2000q-quantum-computer

[5] S. Tsukamoto, M. Takatsu, S. Matsubara, andH. Tamura,"An accelerator architecture for combinatorial optimization problems," *Fujitsu Sci. Tech. J.*, vol. 53, no. 5, pp. 8–13, 2017.

[6] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, "Physics-inspired optimization for quadratic unconstrained problems using a digital annealer," *Front. Phys.*, vol. 7, 2019, Art. no. 48. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphy.2019.00048

[7] V. Choi, "Minor-embedding in adiabatic quantum computation: I. the parameter setting problem," *Quantum Inf. Process.*, vol. 7, no. 5, pp. 193–209, Oct. 2008.

[8] V. Choi, "Minor-embedding in adiabatic quantum computation: Ii. minor-universal graph design," *Quantum Inf. Process.*, vol. 10, no. 3, pp. 343–353, Jun. 2011.

[9] G. Rosenberg, M. Vazifeh, B. Woods, and E. Haber, "Building an iterative heuristic solver for a quantum annealer," *Comput. Optim. Appl.*, vol. 65, no. 3, pp. 845–869, Apr. 2016.

[10] M. Booth, S. P. Reinhardt, and A. Roy, "Partitioning optimization problems for hybrid classical/quantum execution technical report," D-Wave Tech. Rep., 2017. [Online]. Available: https://docs.ocean.dwavesys.com/projects/qbsolv/en/latest/_downloads/bd15a2d8f32e587e9e5997ce9d5512cc/qbsolv_techReport.pdf

[11] G. Bass, C. Tomlin, V. Kumar, P. Rihaczek, and J. Dulny, "Heterogeneous quantum computing for satellite constellation optimization: Solving the weighted k-clique problem," *Quantum Sci. Technol.*, vol. 3, no. 2, Mar. 2018, Art. no. 024010.

[12] G. A. Croes, "A method for solving traveling-salesman problems," *Oper. Res.*, vol. 6, no. 6, pp. 791–812, 1958. [Online]. Available: http://www.jstor.org/stable/167074

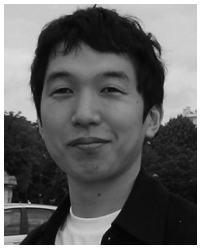
[13] S. LIN, "An effective heuristic algorithm for the traveling salesman problem," *Oper. Res.*, vol. 21, pp. 498–516, 1973. [Online]. Available: https://ci.nii.ac.jp/naid/30041574372/

[14] K. Tanahashi, S. Takayanagi, T. Motohashi, and S. Tanaka, "Application of Ising machines and a software development for Ising machines," *J. Physical Soc. Japan*, vol. 88, no. 6, 2019, Art. no. 061010.

[15] A. Lucas, "Ising formulations of many NP problems," *Front. Phys.*, vol. 2, 2014, Art. no. 5. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphy.2014.00005
[16] M. W. Johnson *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, May 2011. [Online]
[17] P. I. Bunyk *et al.*, "Architectural considerations in the design of a superconducting quantum annealing processor," *IEEE Trans. Appl. Supercond.*, vol. 24, no. 4, pp. 1–10, Aug. 2014.
[18] C. Yoshimura, M. Yamaoka, H. Aoki, and H. Mizuno, "Spatial computing architecture using randomness of memory cell stability under voltage control," in *Proc. Eur. Conf. Circuit Theory Des.*, 2013, pp. 1–4.
[19] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, "24.3 20k-spin Ising chip for combinational optimization problem with CMOS annealing," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2015, pp. 1–3.
[20] T. Inagaki *et al.*, "A coherent Ising machine for 2000-node optimization problems," *Science*, vol. 354, no. 6312, pp. 603–606, 2016. [Online]. Available: https://science.sciencemag.org/content/354/6312/603
[21] H. Goto, K. Tatsumura, and A. R. Dixon, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems," *Sci. Advances*, vol. 5, no. 4, 2019, Art. no. eaav2372. [Online]. Available: https://advances.sciencemag.org/content/5/4/eaav2372
[22] Y. Sugie *et al.*, "Minor-embedding heuristics for large-scale annealing processors with sparse hardware graphs of up to 102,400 nodes," 2020, *arXiv:2004.03819v1*.
[23] Y. Wang, Z. Lü, F. Glover, and J.-K. Hao, "Path relinking for unconstrained binary quadratic programming," *Eur. J. Oper. Res.*, vol. 223, no. 3, pp. 595–604, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221712005334
[24] A. Narimani, S. S. C. Rezaei, and A. Zaribafiyan, "Combinatorial optimization by decomposition on hybrid CPU–non-CPU solver architectures," 2017, *arXiv:1708.03439v3*.
[25] G. Chapuis, H. N. Djidjev, G. Hahn, and G. Rizk, "Finding maximum cliques on the D-wave quantum annealer," 2018, *arXiv:1801.08649v3*.
[26] E. Pelofske, G. Hahn, and H. N. Djidjev, "Solving large maximum clique problems on a quantum annealer," 2019, *arXiv:1901.07657v3*.
[27] Z. Bian, F. Chudak, R. Israel, B. Lackey, W. G. Macready, and A. Roy, "Discrete optimization using quantum annealing on sparse Ising models," *Front. Phys.*, vol. 2, 2014, Art. no. 56. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphy.2014.00056
[28] Z. Bian, F. Chudak, R. Israel, B. Lackey, W. G. Macready, and A. Roy, "Mapping constrained optimization problems to quantum annealing with application to fault diagnosis," *Front. Inform. Comm. Techno.*, vol. 3, 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/fict.2016.00014
[29] Recruit Communications Co., "PyQUBO," Accessed: Feb., 2021. [Online]. Available: https://pyqubo.readthedocs.io/en/latest/index.html
[30] D-Wave Systems Inc., "D-wave ocean software documentation," Accessed: Sep., 2020. [Online]. Available: https://docs.ocean.dwavesys.com/en/stable/
[31] F. Glover, G. A. Kochenberger, and B. Alidaee, "Adaptive memory tabu search for binary quadratic programs," *Manage. Sci.*, vol. 44, no. 3, pp. 336–345, 1998. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.44.3.336
[32] "NumPy," Accessed: Feb., 2021. [Online]. Available: https://numpy.org/
[33] "QAPLIB - a quadratic assignment problem library," Accessed: Sep., 2020. [Online]. Available: http://anjos.mgi.polymtl.ca/qaplib/inst.html
[34] C. D. Gonzalez Calaza, D. Willsch, and K. Michielsen, "Garden optimization problems for benchmarking quantum annealers," *Quantum Inf. Process.*, vol. 20, no. 9, Sep. 2021, Art. no. 305.
[35] S. Feld *et al.*, "A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer," *Front. Inf. Commun. Technol.*, vol. 6, 2019, Art. no. 13. [Online]. Available: https://www.frontiersin.org/article/10.3389/fict.2019.00013
[36] R. Ayanzadeh, P. Das, S. S. Tannu, and M. Qureshi, "EQUAL: Improving the fidelity of quantum annealers by injecting controlled perturbations," 2021, *arXiv:2108.10964v1*.

Yuta Atobe received the BEng and MEng degrees in computer science and engineering from the Waseda University, Tokyo, Japan, in 2012 and 2014, respectively. He is currently a guest junior researcher with the Green Computing Systems Research Organization, Waseda University. His research interests include quantum computation and IoT system design.

Masashi Tawada (Member, IEEE) received the BEng, MEng, and DrEng degrees in computer science from the Waseda University, Tokyo, Japan, in 2010, 2012, and 2015, respectively. He is currently an assistant professor with the Green Computing Systems Research Organization, Waseda University. His research interests include cache design, embedded architecture, and non-volatile memory. He is a member of IEICE and IPSJ.



Nozomu Togawa (Member, IEEE) received the BEng, MEng, and Dr Eng degrees in electrical engineering from the Waseda University, Tokyo, Japan, in 1992, 1994, and 1997, respectively. He is currently a professor with the Department of Computer Science and Communications Engineering, Waseda University. His research interests include quantum computation and integrated system design. He is a member of ACM, IEICE, and IPSJ.