IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 8, 2021, accepted September 19, 2021, date of publication September 28, 2021,
date of current version October 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3116038

A Three-Stage Annealing Method Solving Slot-Placement Problems Using an Ising Machine

KEISUKE FUKADA[1], (Member, IEEE), MATTHIEU PARIZY[1,2], YOSHINORI TOMITA[2], AND NOZOMU TOGAWA[1], (Member, IEEE)

[1]Department of Computer Science and Communications Engineering, Waseda University, Shinjuku, Tokyo 169-8555, Japan
[2]Fujitsu Laboratories Ltd., Kawasaki, Kanagawa 211-8588, Japan

Corresponding authors: Keisuke Fukada (keisuke.fukada@togawa.cs.waseda.ac.jp) and Nozomu Togawa (ntogawa@waseda.jp)

ABSTRACT Ising machines are promising alternatives to solve combinatorial optimization problems, which search for their quasi-optimal solutions with high speed and high accuracy. However, the obtained solution much depends on the initial spin states, since the computation time is finite. Moreover, due to their probabilistic nature, they cannot always satisfy the constraints given to combinatorial optimization problems. In this paper, we propose a three-stage annealing method, targeting a slot-placement problem as a typical but difficult example of combinatorial optimization problems. The proposed method is composed of an initial process, an annealing process, and a correction process. The initial process and the correction process are executed by a classical computer while the annealing process is executed by an Ising machine. In the initial process, we give initial spin values that lead to a relatively good solution to the combinatorial optimization problem, which satisfies the given constraints. Then, the annealing process is executed by an Ising machine, and the solution obtained by the annealing process is further corrected to satisfy the constraints. The experimental results demonstrate that the proposed method reduces a minimum total weighted wiring length by 0.0898%–2.45% on average depending on the initial process methods used, compared to the existing method. The mean total weighted wiring length is reduced by 2.79%–7.08% on average depending on the initial process methods used.

INDEX TERMS Ising machine, Ising model, QUBO model, slot-placement problem, initial process, correction process, combinatorial optimization problem.

I. INTRODUCTION

A. ISING MACHINES

A combinatorial optimization problem is a problem to find a combination of variables that maximizes or minimizes an objective function while satisfying its given constraints. In combinatorial optimization problems, there exist NP-hard problems, in which it is very difficult to search for an optimal solution by using a conventional von Neumann architecture machine.

In recent years, various Ising machines, non-von Neuman architecture machine, using the Ising model [1], [2] have been studied to efficiently solve combinatorial optimization problems [3]–[12]. Ising machines search for their quasi-optimal solution with high speed and high accuracy, where a quasi-optimal solution refers to a solution that does not always maximize or minimize the objective function.

The associate editor coordinating the review of this manuscript and approving it for publication was Abdullah Iliyasu.

When we solve a combinatorial optimization problem using an Ising machine, we map it onto a model in statistical mechanics called an *Ising model*, or its equivalent *quadratic unconstrained binary optimization (QUBO) model*. Nowadays, various combinatorial optimization problems are mapped to Ising models or QUBO models and solved by an Ising machine such as in [13]–[15].

B. SLOT-PLACEMENT PROBLEM

The slot-placement problem [16], [17] is known as one of those difficult combinatorial optimization problems. Given lattice slots and several items connected by a certain number of wires, the goal of the slot-placement problem is to find an optimal item assignment to lattice slots.

In the slot-placement problem, every item must be assigned to a single slot and no more than one item cannot be assigned to a single slot. This is called a *slot-placement constraint*. Several items are connected by a certain number of wires. The wiring length is given by the Manhattan distance between the

slots where the items are assigned. Then we can define the total weighted wiring length over all the item pairs with wires. Minimizing the total weighted wiring length is the objective of the slot-placement problem (See Section III in detail).

The slot-placement problem can be applied to many practical optimization problems such as labor shift scheduling and FPGA logic-block placement [18], [19]. It can be considered to be a variation of quadratic assignment problems, which are known as an NP-hard problem [20], and has been solved using simulated annealing (SA) [21] and branch-and-bound methods [22]. A method for mapping the slot-placement problem to the QUBO model and solving it using an Ising machine has also been proposed [16].

In this paper, we pick up the slot-placement problem as a typical but difficult example of combinatorial optimization problems for Ising machines.

C. HOW TO UTILIZE AN ISING MACHINE SOLVING COMBINATORIAL OPTIMIZATION PROBLEMS

Now we consider solving a combinatorial optimization problem by using an Ising machine. In practice, the obtained solution depends on the initial spin values and the optimal solution is not always obtained, since the computational time is finite. Moreover, the obtained solution can be a quasi-optimal solution and hence it cannot always satisfy the constraints given to the original combinatorial optimization problem. How to tackle those problems is one of the key issues to utilize Ising machines efficiently.

In [16], after the quasi-optimal solutions are obtained by using an Ising machine targeting the slot-placement problem, they are improved as post-processing using a classical computer, so that they satisfy the slot-placement constraint. However, no process is applied before the annealing process of the Ising machine, and all initial spin values are input as zero. As far as we know, there exist no previous researches where any initial process is applied before an Ising-machine-based annealing process.

D. OUR PROPOSAL

In this paper, we propose a three-stage annealing method solving the slot-placement problem. The proposed method is composed of three processes: an initial process, an annealing process, and a correction process. The initial process and the correction process are executed by a classical computer while the annealing process is executed by an Ising machine. In the initial process, we give initial values that give a relatively good solution to the slot-placement problem,[1] which satisfies the slot-placement constraints. Then, the annealing process is just executed by an Ising machine as usual. Finally, the correction process corrects the quasi-optimal solution obtained by the annealing process so that it can satisfy the slot-placement constraints.

As the initial process, we propose three initial process methods: a pair-wise exchange method, a random exchange method, and a cluster-growth method. Given a random feasible slot-placement solution, the pair-wise exchange method repeatedly exchanges the paired items or moves the item to an empty slot which most reduces the objective function, until no further improvement is seen. The random exchange method randomly exchanges the paired items or moves the item to an empty slot for the fixed number of times, if improved. In the cluster-growth method, we first place the item which is most connected to other items to the center of the lattice slots. Then we place an item connected most to the items already assigned to a neighboring slot in a one-by-one manner. The pair-wise exchange method gives a good initial solution but requires much time. The random exchange method runs within a limited time and gives an initial solution depending on its execution time. The cluster-growth method runs very fast but it may not give a good solution. Which method is better is depends on the situation where our proposed three-stage annealing method is utilized.

In the correction process, we correct the obtained quasi-optimal solution by an Ising machine so that it can satisfy the slot-placement constraint. We traverse the QUBO matrix horizontally and vertically and resolve any item assignment violations and/or slot assignment violations.

In our experiment, we apply our three-stage annealing method to various slot-placement problems and evaluate the solutions using an Ising machine hardware [23], [25], and confirm its effectiveness.

E. CONTRIBUTIONS OF THIS PAPER

The contributions of this paper are summarized as follows:

1) We propose a three-stage annealing method solving the slot-placement problem, which efficiently utilizes an Ising machine and obtains a feasible quasi-optimal solution to the slot-placement problem.
2) As the initial process, we propose a pair-wise exchange method, a random method, and a cluster-growth method. We also describe a correction process, which corrects an obtained quasi-optimal solution so that it satisfies the slot-placement constraint.
3) We applied the proposed three-stage annealing method to many types of slot-placement problems. Compared to the two-stage annealing method [16] not including any initial process, the proposed method reduces the minimum total weighted wiring length by approximately 2.42% on average in the case of using the pair-wise exchange method, approximately 2.45% on average in the case of using the random exchange method, and approximately 0.0898% on average in the case of using the cluster-growth method. The mean total weighted wiring length is reduced by approximately 6.92% on average in the case of using the

[1] We can input initial values to spins in an Ising machine as follows: In semiconductor-based Ising machines [3], [8], [23], initial spin values are directly given to them using their API environment; and, in quantum annealers like D-Wave 2000Q [24], initial values can be given to spins using external magnetic fields.

pair-wise exchange method, approximately 7.08% on average in the case of using the random exchange method, and approximately 2.79% on average in the case of using the cluster-growth method.

F. ORGANIZATION OF THIS PAPER

This paper is organized as follows: Section II summarizes the related works; Section III defines the slot-placement problem and its constraints and objective function; Section IV introduces the Ising model and QUBO model and explains the slot-placement problem mapping to the QUBO model; Section V proposes a three-stage annealing method, where we particularly propose three initial processes, i.e., a pair-wise exchange method, a random exchange method, and a cluster-growth method, and also a correction process; Section VI demonstrates the effectiveness of the proposed method using the Ising machine hardware; and Section VII summarizes this paper and gives several concluding remarks.

II. RELATED WORKS

In this section, we firstly summarize typical Ising-machine-applied combinatorial optimization problems. After that, we introduce several approaches to efficiently utilize an Ising machine to solve these combinatorial problems.

A. ISING-MACHINE-APPLIED COMBINATORIAL PROBLEMS

Firstly, we summarize typical combinatorial optimization problems solved by an Ising machine.

In [13], a rectangle packing problem is solved using an Ising machine. Given a set of rectangles, the problem arranges them into a small-sized area without overlapping. The problem is efficiently mapped onto the Ising model and solved by the Ising machine.

In [14], [26], a graph partitioning problem is solved using an Ising machine. Given a graph composed of vertices and edges, the problem is to minimize the number of edges connecting the two partitioned vertex sets, such that the number of elements in the two partitioned vertex sets must be equal. The problem is efficiently mapped onto the Ising model and solved by the Ising machine.

In [15], an induced subgraph isomorphism problem is solved using an Ising machine. Given two graphs, a substitution matrix X with binary variables as elements is introduced, which represents the mapping relation between the two graphs, and the problem is mapped to the QUBO model. The problem has two constraints: one is that there exists only one $(+1)$ binary variable in the row direction of X, and the other is that there exists only one $(+1)$ binary variable in the column direction of X. The objective of the problem is to find a substitution matrix X that satisfies $XA_2X^T = A_1$, given the adjacency matrices A_1 and A_2 of the two graphs.

In [16], a slot-placement problems is solved using an Ising machine. This problem is mapped to the QUBO model using t-by-m matrix with binary variables as elements, where t shows the number of slots and m shows the number of items. The objective of this problem is to minimize the total weighted wiring length under the slot-placement constraint.

In [27], a number partitioning problem is solved using an Ising machine. Given a set of numbers, the problem is to partition the set so that the sum of the elements of the two partitioned sets is equal. The problem is efficiently mapped onto the Ising model and solved by the Ising machine.

In [28], a traveling salesman problem is solved using an Ising machine. The problem is mapped to the QUBO model using an n-by-n matrix with binary variables as elements, where n refers to the number of cities to be traversed. The problem has two constraints: one is that a person cannot visit multiple cities at the same time, and the other is that the same city cannot be visited multiple times. The objective of the problem is to minimize the total travel cost in a route that visits all cities while satisfying the above two constraints.

In [29], a nurse scheduling problem is solved using an Ising machine. This problem has three constraints: upper and lower limit of the number of breaks, the number of nurses in duty for each shift slot, and upper and lower limit of time interval between two days of duty. The objective of the problem is to find an optimal schedule while satisfying the three constraints.

All the above combinatorial optimization problems except for [16] are directly solved by Ising machines. No pre-processes nor post-processes are applied to them to further improve the obtained solution. Due to the probabilistic nature of Ising machines, the obtained solution cannot always become an optimal solution and hence it cannot always satisfy the constraints given to the original combinatorial optimization problem.

B. HOW TO EFFICIENTLY UTILIZE AN ISING MACHINE

In order to efficiently solve combinatorial optimization problems, several approaches have been proposed to utilize an Ising machine and a classical computer.

In [30], a black-box optimization approach is proposed. Black-box optimization is composed of a machine-learning process by a classical computer and annealing process by an Ising machine. Black-box optimization is effective for particular applications as in material design, but is not applicable for general combinatorial optimization problems since its target is machine-learning-based optimization and it requires a large number of iterations.

In [16], an efficient two-stage annealing method utilizing an Ising machine and a classical computer is proposed to solve combinatorial optimization problems. Firstly, an Ising machine is applied to solve a combinatorial optimization problem. After that, a classical computer improves the obtained solution so that it satisfies the constraints given to the original combinatorial optimization problem. This approach can be effectively applied to the slot-placement problem and considered to be one of the state-of-the-art methods proposed before, since no other methods are proposed combining Ising machines and classical computers to satisfy the constraints as post-processing, as far as we know.

However, the obtained solutions by this method much depend on initial spin values in the Ising machine used, and the accuracy of the solutions may not be sufficient.

Based on these discussions, hereinafter, we pick up the slot-placement problem as an example in Sections III and IV and propose an effective annealing method combining an Ising machine and classical computer in Section V.

III. FORMULATION OF THE SLOT-PLACEMENT PROBLEM

In this section, we define and formulate the slot-placement problem as well as its constraints and objective function.

Let us define a set M of m items as $M = \{c_1, c_2, \ldots, c_m\}$ and a set S of slots consisting of h rows and v columns as $S = \{s_1, s_2, \ldots, s_t\}$, where $t = h \times v$ and $m \leq t$. When the slot s_a is positioned at a_1-th row and a_2-th column, and slot s_b is positioned at b_1-th row and b_2-th column ($1 \leq a_1, b_1 \leq h$ and $1 \leq a_2, b_2 \leq v$), the Manhattan distance $l(s_a, s_b)$ between the two slots s_a and s_b is defined by

$$l(s_a, s_b) = |a_1 - b_1| + |a_2 - b_2|. \tag{1}$$

Let $w(c_i, c_j)$ be the number of wires between two items c_i and c_j ($1 \leq i, j \leq m$). $w(c_j, c_i)$ is equal to $w(c_i, c_j)$ and $w(c_i, c_j)$ is zero when c_i and c_j are not connected. $w(c_i, c_j)$ is also zero when $i = j$. When an item c_i is assigned to any slot, the slot is denoted by $s(c_i)$. The weighted wiring length between items c_i and c_j is denoted by $w(c_i, c_j) \times l(s(c_i), s(c_j))$. The total weighted wiring length (TWWL), L, is defined by

$$L = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} w(c_i, c_j) l(s(c_i), s(c_j)). \tag{2}$$

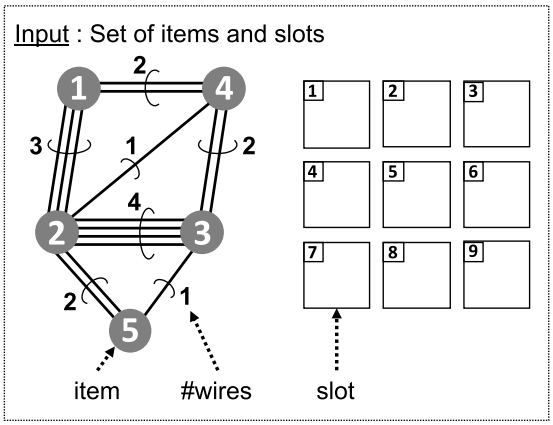
Eqn. (2) is the objective function of the slot-placement problem.

In addition, every item must be assigned to a single slot, which is called *item assignment constraint* and no more than one item cannot be assigned to a single slot, which is called *slot assignment constraint*. The *slot-placement constraint* refers to both of the item assignment constraint and slot assignment constraint.

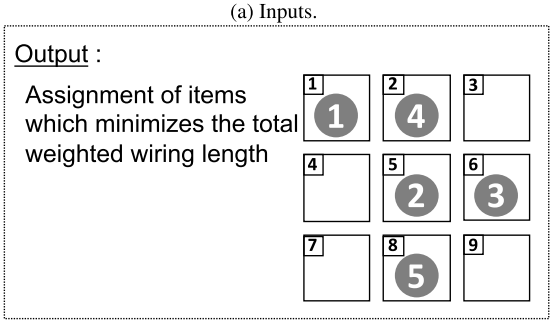
Hence, we define the slot-placement problem as follows:

Definition 1: Given $t = h \times v$ slots and m items which are connected by wires, the slot-placement problem is to find the slot assignment of items that minimizes the total weighted wiring length, L, while satisfying the slot-placement constraint.

Fig. 1 shows an example of the slot-placement problem of $h = v = 3$ and $m = 5$. As in Fig. 1(a), we have five items and 3×3 slots, and some items are connected to each other with wires. For example, the items 1 and 2 are connected by three wires, and the items 1 and 4 are connected by two wires. Fig. 1(b) shows the solution to the slot-placement problem. For example, the item 1 and item 4 are assigned to the slot 1 and the slot 2, respectively. The total weighted wiring length



(a) Inputs.



(b) Output.

FIGURE 1. Example of slot-placement problem.

is calculated as:

$$\begin{aligned} L &= \sum_{i=1}^{5-1} \sum_{j=i+1}^{5} w(c_i, c_j) l(s(c_i), s(c_j)) \\ &= 3 \times 2 + 2 \times 1 + 4 \times 1 \\ &\quad + 1 \times 1 + 2 \times 1 + 2 \times 2 + 1 \times 2 \\ &= 21. \end{aligned} \tag{3}$$

IV. QUBO MODEL MAPPING OF THE SLOT-PLACEMENT PROBLEM

In this section, we firstly introduce the Ising model and QUBO model. After that, we describe QUBO model mapping to the slot-placement problem.

A. ISING MODEL AND QUBO MODEL

The Ising model [1], [2] is a fundamental model in statistical mechanics, which describes the behavior of an entire system of microscopic elements called *spins*, depending on the interaction between the spins and the external magnetic field acting on each spin. As in Fig. 2, the Ising model is defined on an undirected graph $G = (V, E)$, where V is a set of vertices and E is a set of edges between the vertices. When two vertices $i, j \in V$ are connected, $(i, j) \in E$ denotes the connected edge between the vertices i and j. Let σ_i be the spin placed on the vertex i. σ_i takes either $(+1)$ or (-1), where $(+1)$ is the upward spin and (-1) is the downward spin. Let $J_{i,j}$ be the interaction coefficient between the two spins σ_i and σ_j and h_i be the external magnetic field coefficient acting on

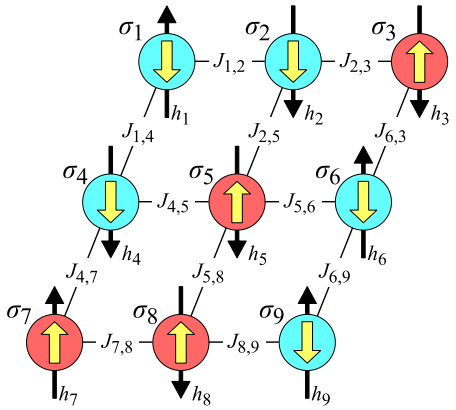


FIGURE 2. Ising model.

the spin σ_i. Then, the Hamiltonian $\mathcal{H}$ of the Ising model is expressed as:

$$\mathcal{H} = -\sum_{(i,j)\in E} J_{i,j}\sigma_i\sigma_j - \sum_{i\in V} h_i\sigma_i. \tag{4}$$

The smaller the value of $\mathcal{H}$ is, the more stable the state of the Ising model is. The state giving the minimum value of $\mathcal{H}$ is called the *ground state*.

The quadratic unconstrained binary optimization (QUBO) model [31] is an equivalent to the Ising model, which uses a binary variable n_i that takes the value of 0 or 1 instead of using a spin σ_i. The Hamiltonian $\mathcal{H}$ of the QUBO model is expressed as follows:

$$\mathcal{H} = -\sum_{(i,j)\in E} J'_{i,j} n_i n_j - \sum_{i\in V} h'_i n_i - \text{const} \tag{5}$$

where $J'_{i,j}$ is the interaction coefficient acting between the two binary variables n_i and n_j, h'_i is the external magnetic field coefficient acting on the binary variable n_i, and const is a constant. Eqn. (4) can be converted to Eqn. (5) by using Eqn. (6) below:

$$n_i = \frac{\sigma_i + 1}{2}. \tag{6}$$

In the rest of this paper, we focus on QUBO models. Note that, QUBO models can be equivalently converted into Ising models as described above.

B. QUBO MODEL MAPPING OF THE SLOT-PLACEMENT PROBLEM

In this subsection, we describe slot-placement problem mapping to the QUBO model based on [15], [16]. Let m be the number of items, $t = h \times v$ be the number of slots, c_i $(1 \le i \le m)$ be the i-th item, and s_a $(1 \le a \le t)$ be a-th slot. We define a binary variable $x_{a,i}$ as follows:

$$x_{a,i} = \begin{cases} 1 & (\text{if } c_i \text{ is assigned to the slot } s_a) \\ 0 & (\text{otherwise}). \end{cases} \tag{7}$$

Fig. 3 shows an example solution of the slot-placement problem when $m = 3$ and $t = 2 \times 2 = 4$. In this figure, an orange colored circle indicates $x_{a,i} = 1$ and a white colored circle indicates $x_{a,i} = 0$.

1) OBJECTIVE FUNCTION

The objective function in the slot-placement problem is given by Eqn. (2). Using the binary variables $x_{a,i}$ and $x_{b,j}$ defined above, Eqn. (2) can be re-written as follows:

$$\mathcal{H}_\mathcal{A} = \frac{1}{2}\sum_{a=1}^{t}\sum_{i=1}^{m}\sum_{b=1}^{t}\sum_{j=1}^{m} w(c_i, c_j)l(s_a, s_b)x_{a,i}x_{b,j}. \tag{8}$$

Let h_a be the minimum value of $\mathcal{H}_\mathcal{A}$. h_a depends on the slot-placement problem instance.

2) ITEM ASSIGNMENT CONSTRAINT

The item assignment constraint shows that every item c_i must be assigned to a single slot. Fig. 3(b) shows an example. In Fig. 3(b), the green check indicates that the constraint is satisfied and the red checks indicate that the constraint is violated. The constraint can be formulated as follows:

$$\sum_{a=1}^{t} x_{a,i} = 1 \quad (1 \le i \le m). \tag{9}$$

Then we introduce the Hamiltonian $\mathcal{H}_\mathcal{B}$ below:

$$\mathcal{H}_\mathcal{B} = \sum_{i=1}^{m}\left(1 - \sum_{a=1}^{t} x_{a,i}\right)^2. \tag{10}$$

$\mathcal{H}_\mathcal{B}$ takes the minimum value of zero if and only if all items satisfy Eqn. (9).

3) SLOT ASSIGNMENT CONSTRAINT

The slot assignment constraint shows that no more than one item cannot be assigned to every slot. Fig. 3(c) shows an example. In Fig. 3(c), the green checks indicate that the constraint is satisfied and the red check indicates that the constraint is violated. The constraint can be formulated as follows:

$$\sum_{i=1}^{m} x_{a,i} = 1 \text{ or } \sum_{i=1}^{m} x_{a,i} = 0 \quad (1 \le a \le t). \tag{11}$$

Then we introduce the Hamiltonian $\mathcal{H}_\mathcal{C}$ below:

$$\mathcal{H}_\mathcal{C} = \sum_{a=1}^{t}\left(\frac{1}{2} - \sum_{i=1}^{m} x_{a,i}\right)^2. \tag{12}$$

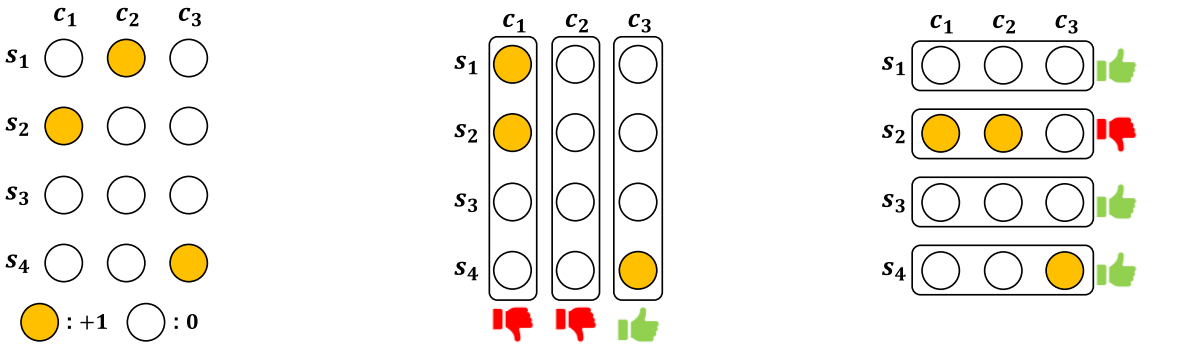
$\mathcal{H}_\mathcal{C}$ takes the minimum value of $t/4$ if and only if all slots satisfy Eqn. (11).

4) TOTAL HAMILTONIAN

Summing up Eqn. (8), Eqn. (10), and Eqn. (12) with a hyper-parameter $\alpha > 0$, we have the total Hamiltonian as follows:

$$\mathcal{H} = \mathcal{H}_\mathcal{A} + \alpha(\mathcal{H}_\mathcal{B} + \mathcal{H}_\mathcal{C}). \tag{13}$$

The Hamiltonian $\mathcal{H}$ takes the minimum value of $h_a + \alpha t/4$, if and only if the ground state is found and the binary variables



(a) Example of solution. (b) Item assignment constraint violations. (c) Slot assignment constraint violations.

FIGURE 3. QUBO model mapping examples where $m = 3$ and $t = 2 \times 2 = 4$.

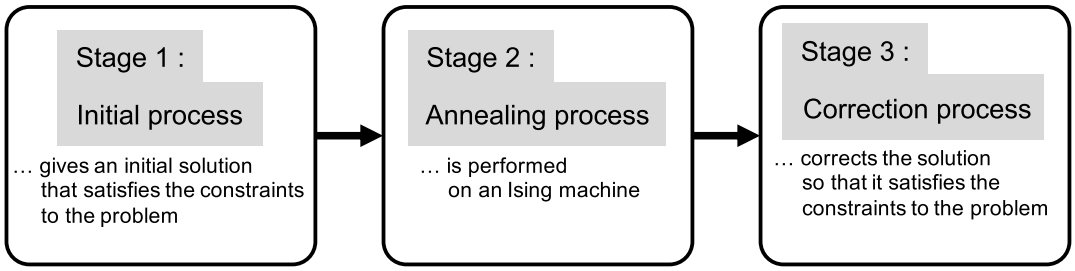


FIGURE 4. The flow of the proposed three-stage annealing method.

giving the ground state shows an optimal solution. How to set up the hyper-parameter α will be discussed in VI-C.

Note that the number of binary variables required to express Eqn. (13) becomes $(m \times t)$, where m shows the number of items and t shows the number of slots.

V. THREE-STAGE ANNEALING METHOD UTILIZING AN ISING MACHINE

A. THREE-STAGE ANNEALING STRATEGY

In general, the annealing process by an Ising machine tries to minimize the Hamiltonian expressed by Eqn. (4) or Eqn. (5). However, the solution obtained by the Ising machine does not necessarily become a ground-state solution and hence does not necessarily minimize Eqn. (4) or Eqn. (5). Particularly, the constraint terms such as Eqn. (10) and Eqn. (12) in the slot-placement problem cannot be minimized and hence those constraints cannot necessarily be satisfied.

Based on this discussion, we firstly introduce a *correction process* as in [16] after the annealing process by an Ising machine is over. The correction process is executed by a classical computer and corrects slightly the quasi-optimal solution obtained by the annealing process so that it satisfies the constraints to the original combinatorial optimization problem.

The annealing process globally searches for the solution space represented by the Ising model or QUBO model and reaches the ground state regardless of its initial spin values theoretically if it takes the infinite amount of time [32], [33]. However, in practice, the obtained solution depends on the initial spin values and the optimal solution is not always obtained, since the computational time is finite. There is a trade-off between time and solution accuracy, and finding a quasi-optimal solution in a short time is as demanding as finding an exact optimal solution in practice. Therefore, by appropriately setting the initial spin values or binary variables, we can obtain a better quasi-optimal solution in a short time by using an Ising machine. Furthermore, due to the existence of constraints in combinatorial optimization problems, the solution space can be multimodal and it may take much time to obtain a global optimal solution [34]. By providing initial values close to the optimal solution, we can improve the probability to reach a better quasi-optimal solution since we can start the search which may be closer to one of them.

Based on this discussion, we secondly introduce an *initial process* before the annealing process by an Ising machine starts. The initial process is executed by a classical computer and gives initial values for spins or binary variables so that the annealing process can obtain a better solution.

In summary, Fig. 4 shows the proposed three-stage annealing method including the initial process and the correction process. (Stage 1) First, the initial process gives an initial solution that satisfies the constraints to the combinatorial optimization problem, which may be close to the optimal solution. (Stage 2) Then, we perform the annealing process on an Ising machine [3], [4], [8], [23], [24] as usual. (Stage 3) Finally, the correction process corrects the quasi-optimal

solution obtained by the annealing process so that it satisfies the constraints to the combinatorial optimization problem. Note that, the time complexity of the proposed initial processes (Stage 1) and correction process (Stage 3) can be calculated as in the subsequent subsections, while the annealing time (Stage 2) using an Ising machine much depends on the Ising machine hardware used and its parameter settings (See Section VI-D for the discussion on execution time).

In the rest of this section, we propose an initial process and also describe a correction process targeting the slot-placement problem.

B. INITIAL PROCESS

As an initial process to the slot-placement problem, we propose a pair-wise exchange method (Section V-B1), a random exchange method (Section V-B2), and a cluster-growth method (Section V-B3).

1) PAIR-WISE EXCHANGE METHOD

We propose a pair-wise exchange method as an initial process to the slot-placement problem. Firstly, we generate a random feasible slot-placement solution, where a feasible solution shows the slot placement satisfying the slot-placement constraint. This is done by just assigning every item to a slot randomly without overlapping. Next, we select two slots and exchange the items assigned to them. If no item is assigned to one of the selected slots, we move the item to the empty slot. If no items are assigned to both the selected slots, no move is done. We try all those exchanges and moves and accept the one which most reduces the total weighted wiring length. This process is repeated until the total weighted wiring length no longer improves.

Algorithm 1 shows the pair-wise exchange method. By performing the pair-wise exchange method, a set of initial binary variables to the QUBO model is obtained.

The pair-wise exchange method starts with a random solution that satisfies the slot-placement constraint and greedily exchanges two items or moves one item to an empty slot to reduce the total weighted wiring length. Although it may lead to a locally optimal solution, it can provide a relatively good initial solution. Since the pair-wise exchange method repeats the exchange and move process until it well converges to a local minima, it requires several amount of time. In some cases, it may become longer than the annealing time (See the discussion in Section VI).

The time complexity of the pair-wise exchange method is calculated as follows: The single item exchange or item move requires $O(m)$ time to re-evaluate the total weighted wiring length, where m shows the number of items. Since we try all item exchanges and moves, we require $O(m \times t^2)$ time in every iteration, where t shows the number of slots. Assume that we repeat this process N_{pwe} times, i.e., after N_{pwe} iterations, no further improvement of the total weighted wiring length is seen. N_{pwe} depends on every slot-placement problem instance. Then the time complexity of the pair-wise exchange method becomes $O(N_{pwe} \times mt^2)$.

Algorithm 1 Pair-Wise Exchange Method

```
Generate a random feasible slot-placement solution;
while (TWWL is improved) do
    Try all item exchanges and moves;
    Accept the one which most reduces TWWL;
end while
return (Slot-placement solution at this time)
```

Algorithm 2 Random Exchange Method

```
Generate a random feasible slot-placement solution;
for (Iteration N) do
    Try item exchange or item move randomly;
    if (old TWWL > new TWWL) then
        Accept the trial above;
    end if
end for
return (Slot-placement solution at this time)
```

2) RANDOM EXCHANGE METHOD

Next, we propose a random exchange method as an initial process to the slot-placement problem. Firstly, we generate a random feasible slot-placement solution. Then, secondly, we try to exchange the items assigned to the slots or move the item to an empty slot for several times. When the total weighted wiring length is reduced, we accept the exchange or move. Algorithm 2 shows the random exchange method. The number N of iterations is given beforehand. By performing the random exchange method, a set of initial binary variables to the QUBO model is obtained.

The random exchange method starts with a random solution that satisfies the slot-placement constraint and randomly exchanges two items or moves an item to an empty slot for a certain number of iterations to reduce the total weighted wiring length. Depending on the number of iterations, a relatively good solution can be obtained as the initial solution. Note that, since the number of iterations can be given in the proposed random exchange method, the CPU time required to the initial process can be adjusted.

The time complexity of the random exchange method is calculated as follows: In the same way as the discussion in the pair-wise exchange method, we require $O(m \times t^2)$ time in every iteration. Since the iteration is repeated N times, time complexity of the random exchange method becomes $O(N \times mt^2)$.

3) CLUSTER-GROWTH METHOD

Lastly, we propose a cluster-growth method as an initial process to the slot-placement problem. Firstly, we assign an item i which is connected most to other items onto the center of the lattice slots. Next, among the items that have not yet been assigned to slots, we pick up the item j which is most connected to the item i and assign j to the slot neighboring to i. Similarly, we pick up an unassigned item which has the maximum connections to the items already assigned and

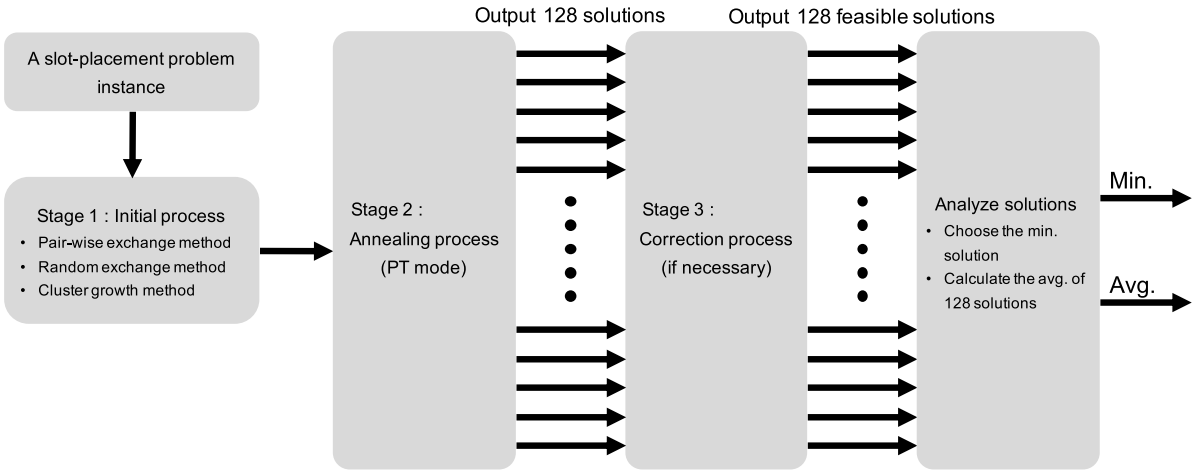


FIGURE 5. The flow of our proposed method.

Algorithm 3 Cluster-Growth Method

Assign an item which has the most connections to other items to the center of the lattice slots;
for $(m-1)$ **do**
Select an unassigned item j which is most connected to the items already assigned;
Assign the item j to the empty slot neighboring to the occupied slots;
end for
return (Slot-placement solutioin at this time)

assign it to an empty slot neighboring to the occupied slots. This process is repeated until all the items are assigned to slots. Algorithm 3 shows the cluster-growth method. By performing the cluster-growth method, a set of initial binary variables to the QUBO model is obtained.

Unlike the pair-wise exchange method and random exchange method, the cluster-growth method does not calculate the total weighted wiring length explicitly during its process. Therefore, the cluster-growth method runs fast even when the scale of the problem becomes larger.

The time complexity of the cluster-growth method is calculated as follows: When we find out the item most connected to other ones, we require $O(m^2)$ time, where m shows the number of items. Since we assign an item to a slot in a one-by-one manner, the time complexity of the cluster-growth method becomes $O(m^3)$.

C. CORRECTION PROCESS

When an annealing process in Fig. 4 outputs a slot-placement solution which does not satisfy the slot-placement constraint, the correction process corrects the slot-placement solution so that it satisfies the slot-placement constraint. Now the QUBO matrix $\{x_{a,i}\}$ refers to the matrix arranging the binary variables given by Eqn. (7) as in Fig. 3. According to [15], [16], we simply apply the processes below as the correction process:

1) DELETE REDUNDANT ITEMS FROM QUBO MATRIX COLUMNS AND ROWS

When two or more (+1) binary variables exist in any column in the QUBO matrix as in the first column of Fig. 3(b), we select one of the (+1) binary variables which minimizes the total weighted wiring length and set the other binary variables in this column to zero. We traverse the QUBO matrix from the first column to the last column to resolve the slot direction overlap.

After that, when two or more (+1) binary variables exist in any row in the QUBO matrix as in the second row of Fig. 3(c), we select one of the (+1) binary variables which minimizes the total weighted wiring length and set the other binary variables in this row to zero. We traverse the QUBO matrix from the first row to the last row to resolve the item direction overlap.

The time complexity of this process is calculated as follows: When verifying that a given QUBO matrix satisfies the slot-placement constraint, we require $O(mt)$ time, where m and t show the number of items and the number of slots, respectively. This is because we traverse the QUBO matrix just twice in the column direction and row direction. Assume that we find out the violation of item assignment constraint in the QUBO matrix column. We require $O(m^2 + t)$ time to try the operation that one of the binary variables is fixed to (+1) and the others to zero in this column, to flip binary variables in this column and re-evaluate the total weighted wiring length. Since we have totally m items and we try the operation for these items at worst, we require $O(m^3 + mt)$ time. The similar discussion holds for the violation of slot assignment constraint. Totally, the time complexity of this process becomes $O(m^3 + mt)$.

2) ADD ITEMS TO QUBO MATRIX COLUMNS

After deleting several items as above, there may exist no (+1) binary variables in a column in the QUBO matrix as in the second column of Fig. 3(b). In this case, we flip one

of the binary variables in the column which minimizes the total weighted wiring length. We traverse the QUBO matrix from the first column to the last column and add an item if necessary to satisfy the item assignment constraint.

The time complexity of this process is calculated as follows: In the worst case, we try to flip almost every binary variable in the QUBO matrix and re-evaluate its total weighted wiring length. Hence, the time complexity of this process becomes $O(mt \times m)$ time.

By applying the above processes, the slot-placement solution can satisfy the slot-placement constraint and hence we can always obtain a feasible slot-placement solution.

VI. EXPERIMENTAL EVALUATIONS

We have evaluated the proposed three-stage annealing method applying to various slot-placement problems.

A. COMPARISON METHOD

We have compared the following two methods to demonstrate the effectiveness of the proposed method:

1) OUR PROPOSED METHOD

The first one is the proposed method. In the proposed method, we use the classical computer environment of CentOS Linux 7.7.1908 and Intel Xeon Gold 6148 CPU processor for the initial process and the correction process. We run the pair-wise exchange (PWE) method, the random exchange (RE) method, and the cluster-growth (CG) method as an initial process. The number of iterations in the random exchange method is set to 10,000. ''Ours (PWE)'' refers to the proposed method with the pair-wise exchange method. ''Ours (RE)'' refers to the proposed method with the random exchange method. ''Ours (CG)'' refers to the proposed method with the cluster-growth method.

As an Ising machine, we use one of the semiconductor-based Ising machines, called Digital Annealer (DA) Unit 2 [23], [25], for the annealing process.[2] The number of iterations in the annealing process in DA is set to 1,000,000 (default). In DA, we use the parallel tempering mode (PT mode), where 128 quasi-optimal solutions are obtained simultaneously. We apply the correction process to these 128 solutions if necessary and obtain the 128 feasible slot-placement solutions. Fig. 5 summarizes the experimental flow of the proposed method.

2) BASELINE METHOD [16] (TWO-STAGE ANNEALING METHOD)

For comparison, we run the existing state-of-the-art method [16] which is composed of the annealing process and the correction process. Before starting the annealing process, we initialize all the binary variables to zero. In this method, we use the same classical computer environment as Ours. We also use a Digital Annealer Unit 2 for the annealing process. The number of iterations in the annealing process is set to 1,000,000. In the same way, we use PT mode, where 128 quasi-optimal solutions are obtained simultaneously. We apply the correction process to these 128 solutions if necessary and obtain the 128 feasible slot-placement solutions.

Note that, if we do not apply the correction process, the obtained slot-placement solutions cannot necessarily satisfy the slot-placement constraint. In fact, if we do not apply the correction process in our experiments, a maximum of 98% solutions violate the slot-placement constraint (See Table 6 in detail). Hence, we compare our proposed method to the two-stage annealing method above.

[2]The Ising machine that we use is implemented on ASIC and realized by hardware [23], [25]. It can deal with a maximum of 8,192 binary variables and hence we set up the problem instances as described in Table 1 where we can use at most 7,500 binary variables.

TABLE 1. The values of the hyper-parameters.

t	m	#variables (N_v)	α_{opt}
4×4	8	128	30
4×4	12	192	60
4×4	16	256	80
5×5	12	300	60
5×5	18	450	160
5×5	25	625	170
6×6	18	648	160
6×6	27	972	350
6×6	36	1296	270
7×7	24	1176	240
7×7	36	1764	530
7×7	49	2401	450
8×8	32	2048	466
8×8	48	3072	683
8×8	64	4096	899
9×9	40	3240	718
9×9	60	4860	1060
9×9	81	6561	1420
10×10	50	5000	1090
10×10	75	7500	1618

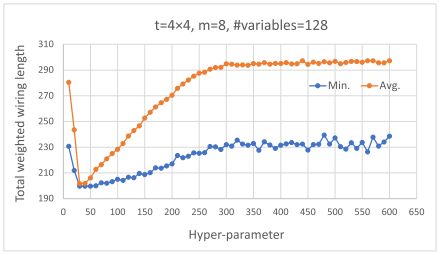
B. PROBLEM SETTING

We prepare 4×4 slots to 10×10 slots. For every $p \times p$ slot instance, we prepare three item instances, $m = (p \times p) \times (1/2)$, $m = (p \times p) \times (3/4)$, and $m = (p \times p)$. Note that, we did not prepare an item instance $m = 100$ for 10×10 slot instance, because the number of variables exceeds the upper limit of the Ising machine that we use for the annealing process. The items are randomly connected by wires such that $w(c_i, c_j) \in [0,\ 10]$ for any two items c_i and c_j.

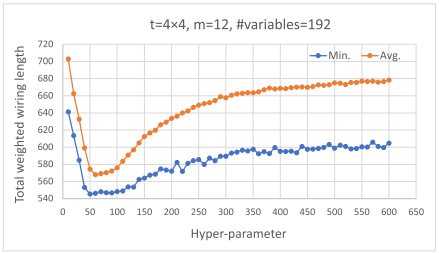
For every slot-placement problem instance, we randomly generate 10 different problems. For each of 10 different problems, we obtained 128 feasible slot-placement solutions by our methods or the baseline method [16]. As in Fig. 5, we picked up the best solution (Min. value) which minimizes the total weighted wiring length over these 128 solutions. We also calculated the mean value (Avg. value) of the total weighted wiring length over 128 solutions. Then, we averaged these Min. values and Avg. values over 10 different problems and summarized them.

C. SEARCH FOR THE OPTIMAL HYPER-PARAMETER

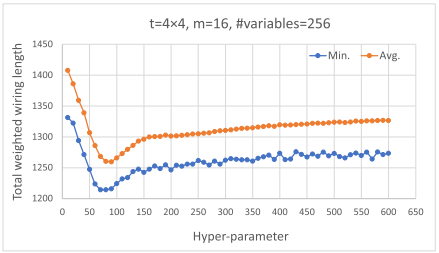
Before we evaluate our proposed method, we have searched for the optimal hyper-parameter α value described in



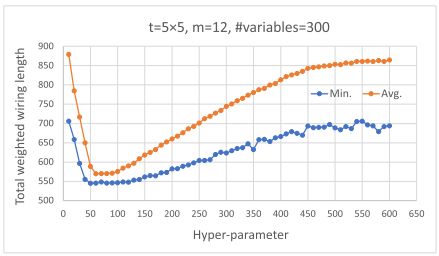
(a) $t = 4 \times 4,\ m = 8,$ #variables$= 128.$



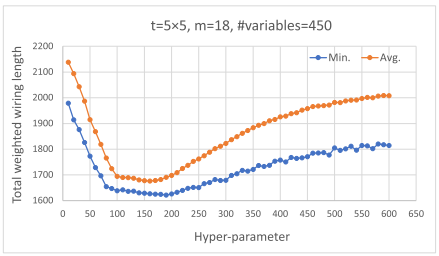
(b) $t = 4 \times 4,\ m = 12,$ #variables$= 192.$



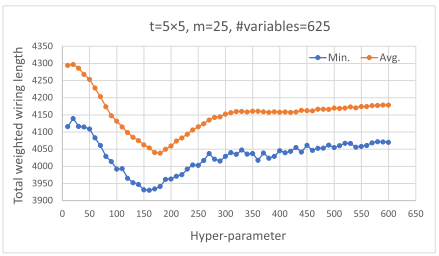
(c) $t = 4 \times 4,\ m = 16,$ #variables$= 256.$



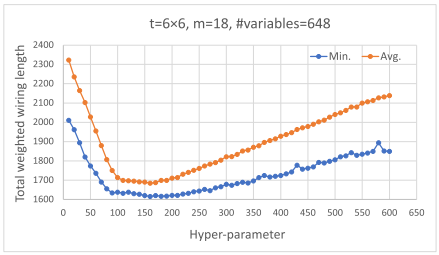
(d) $t = 5 \times 5,\ m = 12,$ #variables$= 300.$



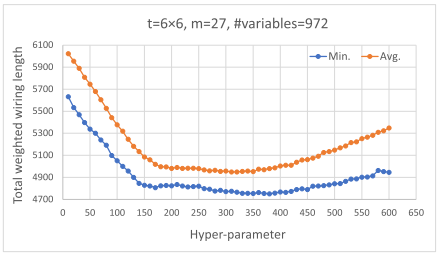
(e) $t = 5 \times 5,\ m = 18,$ #variables$= 450.$



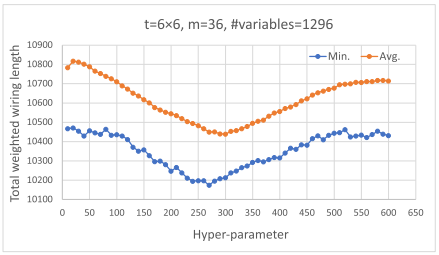
(f) $t = 5 \times 5,\ m = 25,$ #variables$= 625.$



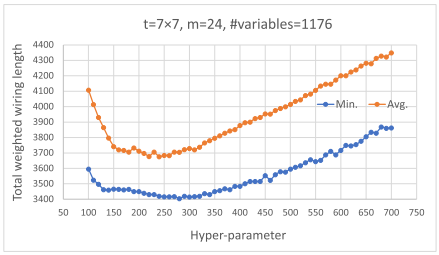
(g) $t = 6 \times 6,\ m = 18,$ #variables$= 648.$



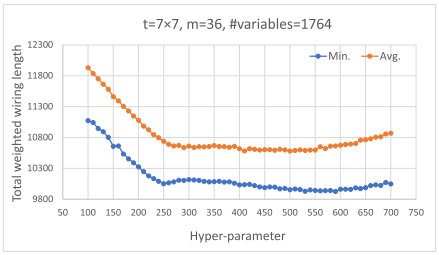
(h) $t = 6 \times 6,\ m = 27,$ #variables$= 972.$



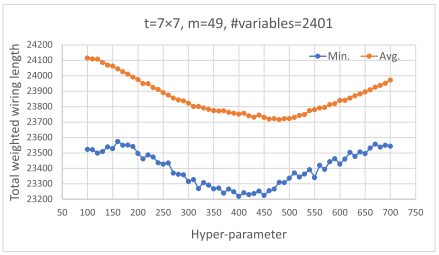
(i) $t = 6 \times 6,\ m = 36,$ #variables$= 1296.$



(j) $t = 7 \times 7,\ m = 24,$ #variables$= 1176.$



(k) $t = 7 \times 7,\ m = 36,$ #variables$= 1764.$



(l) $t = 7 \times 7,\ m = 49,$ #variables$= 2401.$

FIGURE 6. The result of the optimal hyper-parameter search.

Eqn. (13). In this search, we did not use any initial process and set zero to all the binary values at first. We used the correction process to obtain a feasible solution. We prepared 10 different problems for every slot-placement problem instance and searched for the hyper-parameter value in the range of $\alpha \in [10, 600]$ for 4×4, 5×5, and 6×6 slot instances and in the range of $\alpha \in [100, 700]$ for 7×7 slot instance. We also used PT mode as in Section VI-A and we obtained 128 feasible solutions at a time for every problem. We calculated and summarized averaged Avg. value and Min. value as described in Section VI-B for every slot-placement problem instance.

Fig. 6 shows the results. In most of the cases, Avg. value and Min. value are minimized at the particular α value, which gives the optimal hyper-parameter. This is because of the following reason: if the α value is too small, the constraint terms of $\mathcal{H}_\mathcal{B}$ and $\mathcal{H}_\mathcal{C}$ of Eqn. (13) cannot be minimized and hence the non-feasible solutions are obtained. Then the correction process corrects those solutions and their total weighted wiring lengths are not globally minimized. If the α value is too large, the constraint terms of $\mathcal{H}_\mathcal{B}$ and $\mathcal{H}_\mathcal{C}$ of Eqn. (13) can be minimized and the obtained solutions can satisfy the slot-placement constraint. But the first term $\mathcal{H}_\mathcal{A}$ of Eqn. (13) is not fully minimized and hence their total weighted wiring lengths are not minimized.

According to the results of Fig. 6, Fig. 7 plots the optimal hyper-parameter value versus the number of variables. As in Fig. 7, the optimal hyper-parameter value is

TABLE 2. The results of the minimum and mean values of the total weighted wiring length.

t	m	#variables	[16]		Ours (PWE)		Ours (RE)		Ours (CG)	
			Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
4×4	8	128	199.70	201.58	199.90 (0.10%)	201.86 (0.14%)	199.80 (0.050%)	201.35 (-0.11%)	199.70 (-0.00%)	201.75 (0.083%)
4×4	12	192	546.30	567.85	546.20 (-0.018%)	560.39 (-1.3%)	546.30 (0.00%)	560.45 (-1.3%)	547.10 (0.15%)	567.90 (0.0085%)
4×4	16	256	1214.5	1260.7	1207.9 (-0.54%)	1220.5 (-3.2%)	1203.5 (-0.91%)	1211.1 (-3.9%)	1214.0 (-0.041%)	1259.9 (-0.063%)
5×5	12	300	545.7	569.65	545.30 (-0.073%)	565.36 (-0.75%)	546.00 (0.055%)	557.36 (-2.2%)	545.90 (0.037%)	568.31 (-0.24%)
5×5	18	450	1626.9	1676.4	1609.7 (-1.1%)	1626.6 (-3.0%)	1609.6 (-1.1%)	1627.4 (-2.9%)	1628.3 (0.086%)	1674.2 (-0.13%)
5×5	25	625	3934.3	4040.4	3821.3 (-2.9%)	3823.8 (-5.4%)	3822.7 (-2.8%)	3823.0 (-5.4%)	3931.4 (-0.074%)	4036.2 (-0.11%)
6×6	18	648	1615.7	1684.7	1613.8 (-0.12%)	1636.3 (-2.9%)	1611.1 (-0.28%)	1624.7 (-3.6%)	1622.0 (0.39%)	1670.9 (-0.82%)
6×6	27	972	4754.0	4953.5	4681.9 (-1.5%)	4706.4 (-5.0%)	4696.5 (-1.2%)	4727.9 (-4.6%)	4747.0 (-0.15%)	4899.0 (-1.1%)
6×6	36	1296	10174	10450	9758.1 (-4.1%)	9761.6 (-6.6%)	9747.2 (-4.2%)	9752.7 (-6.7%)	10170 (-0.042%)	10410 (-0.38%)
7×7	24	1176	3419.9	3675.9	3378.1 (-1.2%)	3400.1 (-7.5%)	3358.0 (-1.8%)	3371.8 (-8.3%)	3416.3 (-0.11%)	3524.7 (-4.1%)
7×7	36	1764	9931.5	10592	9675.9 (-2.6%)	9685.8 (-8.6%)	9664.7 (-2.7%)	9687.8 (-8.5%)	9921.4 (-0.10%)	10214 (-3.6%)
7×7	49	2401	23226	23731	22147 (-4.6%)	22147 (-6.7%)	22091 (-4.9%)	22093 (-6.9%)	23275 (0.21%)	23620 (-0.47%)
8×8	32	2048	7342.7	8125.2	7259.0 (-1.1%)	7272.8 (-10%)	7194.8 (-2.0%)	7205.6 (-11%)	7337.8 (-0.067%)	7626.1 (-6.1%)
8×8	48	3072	21509	22789	20682 (-3.8%)	20683 (-9.2%)	20645 (-4.0%)	20653 (-9.4%)	21466 (-0.20%)	21931 (-3.8%)
8×8	64	4096	46080	46685	43674 (-5.2%)	43674 (-6.4%)	43824 (-4.9%)	43824 (-6.1%)	45955 (-0.27%)	46349 (-0.72%)
9×9	40	3240	13204	15312	12848 (-2.7%)	12887 (-16%)	12883 (-2.4%)	12930 (-16%)	13183 (-0.16%)	13679 (-11%)
9×9	60	4860	38078	41192	36530 (-4.1%)	36534 (-11%)	36658 (-3.7%)	36674 (-11%)	37912 (-0.44%)	38818 (-5.8%)
9×9	81	6561	84615	85834	80328 (-5.1%)	80328 (-6.4%)	80396 (-5.0%)	80396 (-6.3%)	84546 (-0.080%)	84837 (-1.2%)
10×10	50	5000	23937	27696	23056 (-3.7%)	23144 (-16%)	23071 (-3.6%)	23151 (-16%)	23776 (-0.67%)	24643 (-11%)
10×10	75	7500	67379	73171	64682 (-4.0%)	64690 (-12%)	64947 (-3.6%)	64966 (-11%)	67197 (-0.27%)	69077 (-5.6%)

TABLE 3. The results of the total weighted wiring length after the initial process and the best value.

t	m	#variables	Ours (PWE)		Ours (RE)		Ours (CG)	
			After initial process	Min.	After initial process	Min.	After initial process	Min.
4×4	8	128	206.10	199.90 (-3.0%)	205.00	199.80 (-2.5%)	215.40	199.70 (-7.3%)
4×4	12	192	562.10	546.20 (-2.8%)	562.80	546.30 (-2.9%)	619.70	547.10 (-12%)
4×4	16	256	1215.5	1207.9 (-0.63%)	1208.0	1203.5 (-0.37%)	1329.8	1214.0 (-8.7%)
5×5	12	300	570.30	545.30 (-4.4%)	560.80	546.00 (-2.6%)	619.70	545.90 (-12%)
5×5	18	450	1626.9	1609.7 (-1.1%)	1628.1	1609.6 (-1.1%)	1789.7	1628.3 (-9.0%)
5×5	25	625	3823.8	3821.3 (-0.065%)	3823.0	3822.7 (-0.0078%)	4174.9	3931.4 (-5.8%)
6×6	18	648	1639.1	1613.8 (-1.5%)	1624.6	1611.1 (-0.83%)	1789.7	1622.0 (-9.4%)
6×6	27	972	4694.4	4681.9 (-0.27%)	4714.8	4696.5 (-0.38%)	5138.0	4747.0 (-7.6%)
6×6	36	1296	9758.1	9758.1 (-0.00%)	9747.2	9747.2 (-0.00%)	10518	10170 (-3.3%)
7×7	24	1176	3399.8	3378.1 (-0.64%)	3372.9	3358.0 (-0.44%)	3666.0	3416.3 (-6.8%)
7×7	36	1764	9676	9675.9 (-0.0010%)	9675.8	9664.7 (-0.11%)	10518	9921.4 (-5.7%)
7×7	49	2401	22147	22147 (-0.00%)	22091	22091 (-0.00%)	23756	23275 (-2.0%)
8×8	32	2048	7261.7	7259.0 (-0.032%)	7195.3	7194.8 (-0.0069%)	7960.9	7337.8 (-7.8%)
8×8	48	3072	20683	20682 (-0.0058%)	20653	20645 (-0.037%)	22354	21466 (-4.0%)
8×8	64	4096	43674	43674 (-0.00%)	43824	43824 (-0.00%)	46516	45955 (-1.2%)
9×9	40	3240	12854	12848 (-0.048%)	12888	12883 (-0.040%)	14018	13183 (-6.0%)
9×9	60	4860	36534	36530 (-0.010%)	36674	36658 (-0.044%)	39514	37912 (-4.1%)
9×9	81	6561	80328	80328 (-0.00%)	80396	80396 (-0.00%)	84855	84546 (-0.36%)
10×10	50	5000	23057	23056 (-0.0026%)	23085	23071 (-0.060%)	24992	23776 (-4.9%)
10×10	75	7500	64686	64682 (-0.0059%)	64968	64947 (-0.032%)	70567	67197 (-4.8%)

proportional to the number of variables used and it is expressed by:

$$\alpha_{opt} = 0.2113 \times N_v + 33.571 \tag{14}$$

where N_v shows the number of variables and α_{opt} shows the optimal hyper-parameter value. Based on this discussion, we summarize the hyper-parameter values in Table 1 for every slot-placement problem instance.

D. EVALUATION RESULTS

Table 2 summarizes the averaged minimum and mean values (Min. values and Avg. values) of the total weighted wiring length over 10 different problems for 4×4 to 10×10 slot instances by using our methods and the baseline method [16]. The numbers in parentheses indicate the improvement of our methods over [16]. From Table 2, the minimum values are

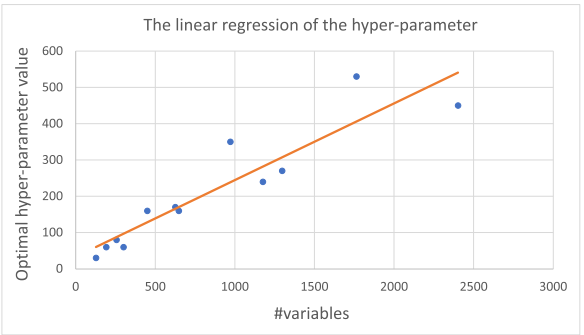


FIGURE 7. The linear regression of the hyper-parameter.

improved by approximately 2.42% on average in the case of Ours (PWE), 2.45% on average in the case of Ours (RE), and 0.0898% on average in the case of Ours (CG) compared

TABLE 4. The results of the averaged execution time.

t	m	#variables	Avg. execution time of initial process [s]				Avg. execution time of Ising machine [s]				Avg. execution time of correction process [$\times 10^{-2}$s]			
			[16]	Ours (PWE)	Ours (RE)	Ours (CG)	[16]	Ours (PWE)	Ours (RE)	Ours (CG)	[16]	Ours (PWE)	Ours (RE)	Ours (CG)
4×4	8	128	0	0.054212	0.52621	0.013915	11.389	11.249	11.062	11.184	0.029180	0.034850	0.032400	0.035780
4×4	12	192	0	0.091239	0.79240	0.014494	11.116	10.924	10.939	11.106	0.041780	0.030400	0.023220	0.043140
4×4	16	256	0	0.16258	1.2259	0.014878	11.167	10.753	10.861	10.894	0.091010	0.034660	0.017080	0.094320
5×5	12	300	0	0.16776	0.73747	0.014326	11.056	10.669	10.867	11.009	0.066910	0.036110	0.016910	0.059970
5×5	18	450	0	0.52202	1.1459	0.015811	10.949	10.805	11.034	10.946	0.20421	0.11438	0.073040	0.18774
5×5	25	625	0	1.0715	1.8169	0.019543	10.771	10.745	10.722	10.671	0.24705	0	0	0.12540
6×6	18	648	0	0.91002	1.0853	0.016379	10.587	10.817	10.573	10.946	0.42162	0.14280	0.10206	0.22814
6×6	27	972	0	2.3547	1.6650	0.019145	10.405	10.301	10.339	10.360	2.4192	0.44805	0.35665	0.91034
6×6	36	1296	0	4.5545	2.5912	0.025032	21.535	21.241	20.951	21.612	6.0998	0.015070	0.02218	0.81577
7×7	24	1176	0	3.2632	1.4062	0.019438	22.749	21.164	21.537	21.811	4.9575	0.087450	0.076890	0.21027
7×7	36	1764	0	9.1396	2.2049	0.022878	22.179	21.461	21.610	21.994	18.594	0.41961	0.51868	0.96635
7×7	49	2401	0	20.114	3.4721	0.14446	69.404	68.494	68.170	79.650	35.944	0.0025600	0.013810	1.39946
8×8	32	2048	0	10.607	1.8651	0.032375	21.249	20.991	21.366	21.364	26.416	0.54613	0.54558	1.29907
8×8	48	3072	0	28.981	2.9293	0.033662	69.183	67.771	68.243	69.125	63.894	0	0	0.0038300
8×8	64	4096	0	63.102	4.5936	0.041518	67.508	67.118	67.311	67.545	91.736	0	0	0
9×9	40	3240	0	28.008	2.2971	0.036731	69.006	68.491	68.752	68.981	85.008	1.6722	2.0496	3.2232
9×9	60	4860	0	72.705	3.5972	0.043053	97.418	92.374	94.278	97.493	169.10	0	0	0.020720
9×9	81	6561	0	162.67	5.7210	0.052643	96.158	92.292	92.794	94.77	206.68	0	0	0
10×10	50	5000	0	65.776	2.8512	0.043223	97.165	95.492	96.369	97.219	187.49	4.4806	4.6448	6.3012
10×10	75	7500	0	191.29	4.5540	0.053602	94.939	92.702	93.542	95.180	355.52	0.085860	0.22586	0.31674

TABLE 5. The results of the averaged total execution time.

t	m	#variables	Avg. total execution time [s]			
			[16]	Ours (PWE)	Ours (RE)	Ours (CG)
4×4	8	128	11.389	11.303	11.588	11.198
4×4	12	192	11.117	11.015	11.731	11.121
4×4	16	256	11.168	10.916	12.087	10.910
5×5	12	300	11.057	10.837	11.604	11.024
5×5	18	450	10.951	11.328	12.180	10.964
5×5	25	625	10.774	11.816	12.539	10.692
6×6	18	648	10.591	11.729	11.659	10.964
6×6	27	972	10.429	12.660	12.007	10.388
6×6	36	1296	21.596	25.796	23.542	21.645
7×7	24	1176	22.798	24.428	22.944	21.833
7×7	36	1764	22.365	30.605	23.820	22.026
7×7	49	2401	69.764	88.607	71.642	79.809
8×8	32	2048	21.513	31.603	23.236	21.409
8×8	48	3072	69.822	96.752	71.172	69.159
8×8	64	4096	68.425	130.22	71.905	67.586
9×9	40	3240	69.856	96.516	71.070	69.050
9×9	60	4860	99.109	165.08	97.875	97.536
9×9	81	6561	98.224	254.96	98.515	94.823
10×10	50	5000	99.040	161.31	99.266	97.325
10×10	75	7500	98.494	283.99	98.098	95.237

TABLE 6. The results of the number of the correction process runs.

t	m	#variables	# of the correction process runs			
			[16]	Ours (PWE)	Ours (RE)	Ours (CG)
4×4	8	128	14.5/128	15.3/128	14.8/128	15.4/128
4×4	12	192	17.4/128	14.4/128	10.2/128	18.4/128
4×4	16	256	37.2/128	13.6/128	6.4/128	37.8/128
5×5	12	300	19.5/128	9.6/128	6.3/128	17.6/128
5×5	18	450	51/128	30.6/128	19.2/128	50.6/128
5×5	25	625	35.1/128	0/128	0/128	32.6/128
6×6	18	648	53.6/128	29.8/128	21.7/128	48.3/128
6×6	27	972	113.6/128	52.9/128	41.6/128	103.7/128
6×6	36	1296	117.5/128	2.1/128	3.4/128	100.6/128
7×7	24	1176	41.1/128	11.3/128	9.9/128	26.5/128
7×7	36	1764	97.9/128	34.4/128	35.6/128	71.8/128
7×7	49	2401	121.6/128	0.2/128	1.3/128	87.6/128
8×8	32	2048	99.1/128	32.6/128	30.7/128	71.2/128
8×8	48	3072	37/128	0/128	0/128	0.2/128
8×8	64	4096	39.1/128	0/128	0/128	0/128
9×9	40	3240	125.3/128	46.7/128	52.8/128	83.7/128
9×9	60	4860	49.5/128	0/128	0/128	0.7/128
9×9	81	6561	51.5/128	0/128	0/128	0/128
10×10	50	5000	128/128	52.5/128	52.9/128	73.6/128
10×10	75	7500	63.5/128	2.1/128	5.1/128	6.9/128

to [16]. The mean values are improved by approximately 6.92% on average in the case of Ours (PWE), 7.08% on average in the case of Ours (RE), and 2.79% on average in the case of Ours (CG). The results indicate that Ours is superior to [16] in almost all the cases. The total weighted wiring length obtained by Ours (PWE) and Ours (RE) is almost the same. In any cases except for small examples, the total weighted wiring length is reduced by introducing the initial process, which definitely indicates that the three-stage annealing process is effective to the slot-placement problem.

Table 3 shows the total weighted wiring length after the initial process and the averaged minimum value (Min. value) after the correction process. The numbers in parentheses indicate the improvement of Min. value over "after initial process." The averaged minimum total weighted wiring length obtained after the correction process are improved approximately up to 4.4% in the case of Ours (PWE), 2.9% in the case of Ours (RE), and 12% in the case of Ours (CG) compared to those after initial process. These results demonstrate that the annealing process successfully improves the total weighted wiring length, even after the initial solution is given by the initial process.

Table 4 summarizes the averaged execution time for each stage and Table 5 summarizes the total averaged execution time for multiple stages. The initial process itself requires several amounts of time, but by introducing the initial process, the time required for the correction process is much reduced. Particularly, Ours (CG) requires less time than Ours (PWE) and Ours (RE), and the overall execution time of Ours (RE) and Ours (CG) are comparable to [16].

Table 6 shows the average number of the correction process runs. As described, we obtain 128 solutions simultaneously and hence the correction process can be applied to

each of them, i.e., the maximum number of the correction process runs becomes 128. If all the solutions obtained by the annealing process are feasible ones, then the number of the correction process runs becomes zero. Table 6 clearly indicates that, by introducing the initial process, the annealing process tends to output the feasible solutions and hence the average number of the correction process runs is much decreased.

Overall, the three-stage annealing process is effective enough to the slot-placement problem in terms of reducing the objective function, satisfying the slot-placement constraint, and computational time.

VII. CONCLUSION

In this paper, we have proposed a three-stage annealing method solving the slot-placement problem. The experimental results demonstrate that the proposed method reduces the minimum total weighted wiring length by 0.0898%–2.45% on average depending on the initial process methods used, compared to the existing method. The mean total weighted wiring length is reduced by 2.79%–7.08% on average depending on the initial process methods used. Totally, the proposed method is effective to the slot-placement problem. In the future, we will apply the proposed approach to other combinatorial problems and further confirm its effectiveness.

REFERENCES

[1] E. Ising, "Beitrag zur Theorie des Ferromagnetismus," *Zeitschrift Phys. A, Hadrons Nuclei*, vol. 31, no. 1, pp. 253–258, Feb. 1925.

[2] H. Nishimori, *Statistical Physics of Spin Glasses and Information Processing: An Introduction*, no. 111. Oxford, U.K.: Clarendon Press, 2001.

[3] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, "A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 303–309, Jan. 2016.

[4] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, and O. Tadanaga, "A coherent Ising machine for 2000-node optimization problems," *Science*, vol. 354, no. 6312, pp. 603–606, Nov. 2016.

[5] A. Marandi, Z. Wang, K. Takata, R. L. Byer, and Y. Yamamoto, "Network of time-multiplexed optical parametric oscillators as a coherent Ising machine," *Nature Photon.*, vol. 8, pp. 937–942, Oct. 2014.

[6] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, "Evidence for quantum annealing with more than one hundred qubits," *Nature Phys.*, vol. 10, pp. 218–224, Feb. 2014.

[7] R. Hamerly, T. Inagaki, P. L. McMahon, D. Venturelli, A. Marandi, T. Onodera, E. Ng, C. Langrock, K. Inaba, T. Honjo, and K. Enbutsu, "Experimental investigation of performance differences between coherent Ising machines and a quantum annealer," *Sci. Adv.*, vol. 5, no. 5, May 2019, Art. no. eaau0823.

[8] H. Goto, K. Tatsumura, and A. R. Dixon, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems," *Sci. Adv.*, vol. 5, no. 4, Apr. 2019, Art. no. eaav2372.

[9] M. Maezawa, G. Fujii, M. Hidaka, K. Imafuku, K. Kikuchi, H. Koike, K. Makise, S. Nagasawa, H. Nakagawa, M. Ukibe, and S. Kawabata, "Toward practical-scale quantum annealing machine for prime factoring," *J. Phys. Soc. Jpn.*, vol. 88, no. 6, Jun. 2019, Art. no. 061012.

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

[11] A. Lucas, "Ising formulations of many NP problems," *Frontiers Phys.*, vol. 2, pp. 1–15, Feb. 2014.

[12] S. Tanaka, R. Tamura, and B. K. Chakrabarti, *Quantum Spin Glasses, Annealing and Computation*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[13] K. Terada, D. Oku, S. Kanamaru, S. Tanaka, M. Hayashi, M. Yamaoka, M. Yanagisawa, and N. Togawa, "An ising model mapping to solve rectangle packing problem," in *Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT)*, Apr. 2018, pp. 1–4.

[14] H. Ushijima-Mwesigwa, C. F. A. Negre, and S. M. Mniszewski, "Graph partitioning using quantum annealing on the D-wave system," in *Proc. 2nd Int. Workshop Post Moores Era Supercomput.*, Nov. 2017, pp. 22–29.

[15] N. Yoshimura, M. Tawada, S. Tanaka, J. Arai, S. Yagi, H. Uchiyama, and N. Togawa, "Efficient ising model mapping for induced subgraph isomorphism problems using ising machines," in *Proc. IEEE 9th Int. Conf. Consum. Electron. (ICCE-Berlin)*, Sep. 2019, pp. 227–232.

[16] S. Kanamaru, K. Kawamura, S. Tanaka, Y. Tomita, H. Matsuoka, K. Kawamura, and N. Togawai, "Mapping constrained slot-placement problems to ising models and its evaluations by an ising machine," in *Proc. IEEE 9th Int. Conf. Consum. Electron. (ICCE-Berlin)*, Sep. 2019, pp. 221–226.

[17] M. Hanan and J. M. Kurtzberg, "A review of the placement and quadratic assignment problems," *SIAM Rev.*, vol. 14, no. 2, pp. 324–342, Apr. 1972.

[18] A. Hemani and A. Postula, "Cell placement by self-organisation," *Neural Netw.*, vol. 3, no. 4, pp. 377–383, Jan. 1990.

[19] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica: J. Econ. Soc.*, pp. 53–76, 1957.

[20] S. Srinivasan, V. Kamakoti, and A. Bhattacharya, "Towards quick solutions for generalized placement problem," in *Proc. Int. Symp. Electron. Syst. Design*, Dec. 2011, pp. 106–111.

[21] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated Annealing: Theory and Applications*. New York, NY, USA: Springer, 1987, pp. 7–15.

[22] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Oper. Res.*, vol. 14, no. 4, pp. 699–719, Jul./Aug. 1966.

[23] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. Katzgrabeer, "Physics-inspired optimization for quadratic unconstrained problems using a digital annealer," *Frontiers Phys.*, vol. 7, p. 48, Apr. 2019.

[24] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, and E. M. Chapple, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[25] M. Sao, H. Watanabe, Y. Musha, and A. Utsunomiya, "Application of digital annealer for faster combinatorial optimization," *Fujitsu Sci. Tech. J.*, vol. 55, no. 2, pp. 45–51, 2019.

[26] I. Hen and F. M. Spedalieri, "Quantum annealing for constrained optimization," *Phys. Rev. A, Gen. Phys.*, vol. 5, no. 3, Mar. 2016, Art. no. 034007.

[27] M. Yamaoka, T. Okuyama, M. Hayashi, C. Yoshimura, and T. Takemoto, "CMOS annealing machine: An in-memory computing accelerator to process combinatorial optimization problems," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2019, pp. 1–8.

[28] R. Martoňák, G. E. Santoro, and E. Tosatti, "Quantum annealing of the traveling-salesman problem," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 70, no. 5, Nov. 2004, Art. no. 057701.

[29] O. Titiloye and A. Crispin, "Quantum annealing of the graph coloring problem," *Discrete Optim.*, vol. 8, no. 2, pp. 376–384, May 2011.

[30] K. Kitai, J. Guo, S. Ju, S. Tanaka, K. Tsuda, J. Shiomi, and R. Tamura, "Designing metamaterials with quantum annealing and factorization machines," *Phys. Rev. Res.*, vol. 2, no. 1, Mar. 2020, Art. no. 013319.

[31] E. Boros, P. L. Hammer, and G. Tavares, "Local search heuristics for quadratic unconstrained binary optimization (QUBO)," *J. Heuristics*, vol. 13, no. 2, pp. 99–132, Feb. 2007, doi: 10.1007/s10732-007-9009-3.

[32] S. Morita and H. Nishimori, "Convergence theorems for quantum annealing," *J. Phys. A: Math. Gen.*, vol. 39, no. 45, pp. 13903–13920, Oct. 2006, doi: 10.1088/0305-4470/39/45/004.

[33] S. Morita and H. Nishimori, "Convergence of quantum annealing with real-time Schrödinger dynamics," *J. Phys. Soc. Jpn.*, vol. 76, no. 6, 2007, Art. no. 064002.

[34] P. Kerschke and C. Grimme, "An expedition to multimodal multi-objective optimization landscapes," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.* New York, NY, USA: Springer, 2017, pp. 329–343.

KEISUKE FUKADA (Member, IEEE) received the B.Eng. degree in communications and computer engineering from Waseda University, in 2021, where he is currently pursuing the M.Eng. degree. His research interests include mathematical optimization and quantum computing.

MATTHIEU PARIZY received the B.Eng. and M.Eng. degrees in computer science from ESIEE Paris, in 2006 and 2008, respectively. He then joined Fujitsu Laboratories Ltd., where he is currently working while being enrolled in a Ph.D. course from Waseda University. His research interests include combinatorial optimization, machine learning, and VLSI design.

YOSHINORI TOMITA received the B.Eng., M.Eng., and Ph.D. degrees from Tokyo Institute of Technology, in 1993, 1995, and 1999, respectively. He is currently working on application of Fujitsu Digital Annealer to real world combinatorial optimization problems at Fujitsu Laboratories Ltd. His research interests include optimization problems and simulation.

NOZOMU TOGAWA (Member, IEEE) received the B.Eng., M.Eng., and Dr.Eng. degrees in electrical engineering from Waseda University, 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 VLSI design, graph theory, and quantum computing. He is a member of IEICE and IPSJ.