International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 13, Number 1, pages 2–17, January 2023

Applying Quantum Annealing for Shift Scheduling Problem for Call Centers

Natsuki Hamada
Department of Media and Governance, Graduate school of Keio University
Fujisawa, Kanagawa, 252-0882, Japan

Kazuhiro Saito
KDDI Research, Inc.
Fujimino, Saitama, 356-8502, Japan

Hideyuki Kawashima
Faculty of Environment and Information Studies, Keio University
Fujisawa, Kanagawa, 252-0882, Japan

Received: July 30, 2022
Revised: October 20, 2022
Accepted: November 24, 2022
Communicated by Masahiro Shibata

Abstract

Quantum annealing (QA) has shown good solutions quickly in benchmarks of combinatorial optimization problems. There are not many real-world problems that illustrate the effectiveness of QA compared to the typical optimization methods. We applied QA to a shift scheduling problem for call centers with four conditions. The shift scheduling problem for call centers optimizes shifts for a specific period of time while satisfying various constraints, such as satisfying the desired worker's shifts. In this paper, we formulated the shift scheduling problem for call centers in QUBO to apply QA. We tuned various parameters required to execute the QA. Then, we compared QA with six typical optimization methods using a classical computer to investigate QA's effectiveness for the shift scheduling problem for call centers. The results show that for three instances with up to 126 variables, QA is about 3-14 times faster than other methods and obtains enough quality of solutions. This shows that QA can be one of the effective methods to solve practical applications in the real-world.

Keywords: Shift scheduling, Quantum annealing, Combinatorial optimization

1 Introduction

The shift scheduling problem [13] solves work shifts that satisfy constraints such as meeting the number of workers requested for a specific period of time. Example of applications include work shifts for call center operators and hospital nurses [8, 16, 18, 20]. Work shifts should be created while

[0]The conference version of this paper is published in the proceedings of 24th Workshop on Advances in Parallel and Distributed Computational Models (APDCM 2022).
[0]This paper is partially based on results obtained from a project, JPNP16007, subsidized by the New Energy and Industrial Technology Development Organization (NEDO)

taking into account the slot requests by part-time workers and various other conditions such as the number of workers required for each time slot. The shift scheduling process with these constraints is often done manually, and as this requires an enormous amount of time and efforts, its automation is desired. Shift scheduling can be formulated as a combinatorial optimization problem that takes into account the shift conditions as constraints. This is difficult to solve efficiently by using classical computers even with approximation because the search space is so large.

Recently, quantum annealing (**QA**) is expected to be used to efficiently solve combinatorial optimization problems that are difficult to compute in polynomial time. QA is a kind of quantum algorithm that uses quantum tunneling to obtain a global minimum solution theoretically [22]. To apply QA for combinatorial optimization problems, we need to formulate the problems as the Ising model. Ising model represents a two-dimensional lattice model of ferromagnetic spins in statistical mechanics.

$$E(\boldsymbol{s}) = \sum_{i,j} J_{ij} s_i s_j + \sum_i h_i s_i \tag{1}$$

Eq. 1 represents the Ising model. The N variables (spins) $\boldsymbol{s} = [s_1, \ldots, s_N]$ are binary with $s_i = \{+1, -1\}$. J_{ij} and h_i are the coefficients representing the spin interaction and the value of the local field respectively. The combinatorial optimization problem can be solved by QA by formulating it into the Ising model as the minimization of the objective function. Minimizing the Ising model is known to be NP-hard [7]. We formulate the shift scheduling problem for call centers as the Ising model, and thus the complexity is NP-hard.

Since the formulation needs to be intuitive when solving real problems, we convert the spins s_i into binary variables $x_i \in \{0, 1\}$ as follows.

$$x_i = \frac{s_i + 1}{2} \tag{2}$$

To use QA, this is applied to the Ising model in Eq. 1 to obtain the following quadratic unconstrained binary optimization (**QUBO**).

$$E(\boldsymbol{x}) = \sum_{i,j} Q_{ij} x_i x_j \tag{3}$$

Therefore, it is possible to apply the shift scheduling problem for call centers to QA by formulating the objective function and constraints of the shift scheduling problem for call centers to the QUBO in Eq. 3. By using Eq. 2, the Ising model can be converted into QUBO and vice versa, so we can choose the formulation that is easier to formulate depending on the problem. In this paper, we formulate the shift scheduling problem for call centers in QUBO.

While there has been prior work that applies QA to the real problems (traffic flow modeling [19], air traffic management [21], multi-car paint shop [23], and nurse scheduling [8, 16]), ours is the first work that clarifies the effectiveness of the QA with a real quantum machine for the shift scheduling problem for call centers with comparison to classical methods to the best of our knowledge.

In this paper, we first formulate the shift scheduling for a call center with QUBO so that it can be treated with QA. Then, we tune the various parameters for applying QA. Then, we evaluate QA using a real quantum computer [1] and compare it with six existing methods on a classical computer. Our contribution of this paper is that we formulated the shift scheduling problem for call centers by QUBO and we revealed the performance comparison of classical methods and QA. Our results confirm that QA produces enough quality of solutions up to 14 times faster than classical methods.

The rest of this paper is organized as follows. In Sec. 2, we formulate the problem by QUBO. Sec. 3 describes the setup and tunes the parameters using a QA machine. Sec. 4 evaluates and compares several methods. Sec. 5 describes related work. Sec. 6 concludes this paper.

	DAY	1			2			3			4			5			6			7				
	PLAN	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	SUM	GAP
No. 1	(5)	x	o	o	!	=	x	=	x	x	o	=	=	o	=	x	=	x	=	o	x	=	6	1
No. 2	(5)	=	x	x	x	o	=	o	x	=	=	=	=	=	o	x	x	o	o	=	=	x	5	0
No. 3	(5)	x	=	x	=	=	x	x	x	=	x	o	=	x	x	=	o	x	!	x	o	o	5	0
No. 4	(5)	x	=	x	=	=	x	x	x	=	x	o	=	x	x	=	o	x	!	x	o	o	5	0
No. 5	(5)	o	o	=	=	=	x	=	=	o	x	x	o	=	x	x	=	o	=	o	x	x	6	1
No. 6	(5)	=	=	x	x	=	o	=	o	o	x	x	x	x	x	o	o	=	=	=	=	=	5	0
	SUM	1	2	1	1	1	1	1	1	2	1	2	1	1	1	1	3	2	3	2	2	2		
	GAP	-1	0	-1	-1	-1	-1	-1	-1	0	-1	0	-1	-1	-1	-1	0	-1	0	-1	-1	-1		

Assigned shift : o
Desired shift : =
Shift not desired : x
Shift not desired but assigned : !

Figure 1: An example of a shift table

Table 1: Symbols for formulation.

	Symbol	Explanation
A	$= \{\text{worker } 1, \text{worker } 2, \ldots, \text{worker } a\}$	Set of workers
D	$= \{\text{day } 1, \text{day } 2, \ldots, \text{day } d\}$	Set of days for schedule
T	$= \{\text{term } 1, \text{term } 2, \ldots, \text{term } t\}$	Set of terms in a day (e.g. morning, afternoon, evening).
S_{dt}	$= \{s \mid s$ is necessary number of booths for term t in day $d\}$	
R_a	$= \{r \mid r$ is the sum of the number of terms desired by worker a in a period$\}$	$\sum_A R_a \geq \sum_{D,T} S_{dt}$
r_{adt}	$= \{0, 1\}$	1 if worker a can be assigned to day d, term t; otherwise 0.
G_{ga}	$= \{0, 1\}$	1 if worker a is in group g; otherwise 0.
x_{adt}	$= \{0, 1\}$	1 if worker a is assigned to day d and term t; otherwise 0.

2 Applying Quantum Annealing to Shift Scheduling Problem for Call Centers

This paper considers a shift scheduling problem in a call center. In a call center, each worker is assigned to a booth and each day has shift terms (e.g., morning, afternoon, evening). The number of booths required for each term is determined, and workers are assigned to each term to keep the number of booths close to the number of workers. Each worker has a desired shift, and no shift should be assigned to a term that the worker does not desire.

Fig. 1 shows an example of a shift table. There are seven days in the period of time. A day is divided into three terms. Each term has a required number of booths, which are represented in the PLAN in the second row from the top. From the third to the eighth row represents the worker's shift information. In this example, we consider the case of six workers. Each worker has a desired shift and must take this into account when scheduling the worker's shift. There are four shift situations. The first, "o" , represents the shift assigned as the worker desired. The second, "=", represents a shift that was not actually assigned, but was desired by the worker. The third ,"x", represents a shift that the worker did not desire and was not assigned. The fourth, "!", represents a shift that

the worker did not want, but was assigned. A shift table is created by allocating each of these four situations to each term and worker. In addition, each worker has the number of terms he/she wants to work in a specific period of time. The number in parentheses to the right of the number of the worker represents this. The SUM in the second column from the right represents the sum of the number of shifts that each worker works in a specific period of time, and the GAP in the rightmost column represents the difference between the number of SUM and the number of terms that each worker desires to work. Workers No. 3 and No. 4 are colored blue to represent the constraint that they must work in the same shifts. In this example, the two workers are assigned to the same shift, so the constraint is satisfied. The second row from the bottom, SUM, represents the sum of the workers assigned in each term. The GAP on the bottom row represents the gap between the number of booths required for each term and the number of workers assigned to it. It can be said that the closer the two kinds of GAPs are to 0, the better the shift is. Thus, the shift scheduling problem is the problem of minimizing these GAPs while satisfying the desires of the workers.

The symbols used for formulation are listed in Table 1.

2.1 Objective Functions

There are two objective functions. The first objective function represents the degree of satisfaction of the actual number of assigned shifts against the number of shifts required for each term. This objective function is used to minimize the difference between the requested number of booths and the number of workers for each term. The second objective function represents the degree of satisfaction of the actual number of assigned shifts against the demanded number of shifts by each worker. In order to meet this condition, workers are asked in advance how many terms they want to work in the specific period. The second objective function is used to minimize the difference between the total number of terms that each worker wants and the total number of terms that are actually assigned. In this way, we can avoid a situation where shifts are concentrated on certain workers and other workers are unable to take shifts.

$$\arg\min_{x_{adt}} \sum_{d} \sum_{t} (\sum_{a} |x_{adt} - S_{dt}|) \tag{4}$$

$$\arg\min_{x_{adt}} \sum_{a} (\sum_{d} \sum_{t} |x_{adt} - R_a|) \tag{5}$$

Eq. 4 denotes the first objective function. It is the sum of the x_{adt} for term t on day d with worker a, and is subtracted from the originally set desired number of booths, S_{dt}. By minimizing this equation, we optimally allocate workers so as to approach the number of workers required for each term. Eq. 5 denotes the second objective function. It is the sum of the difference between the sum of the x_{adt} assigned to each worker a and the R_a of each set worker, where R_a is the total number of terms in the requested shifts for all workers.

2.2 Constraints

There are two constraints. Eq. 6 denotes the first constraint, which represents not to allocate shifts for terms that the worker does not want ($r_{adt} = 0$). Eq. 7 denotes the second constraint, which ensures all workers in a group are assigned to the same shift. The group means a set of workers that have to work together, such as both a trainee and his/her mentor. This equation is satisfied by either all workers in the same group or no workers in the same group are assigned. The left side of this equation is the number of assigned workers of a group, a day, and a term. This constraint is satisfied if the left side equals 0 or the sum of the number of workers in the group.

$$\begin{aligned} &\forall a \in A, \forall d \in D, \forall t \in T \\ &x_{adt} = 0, \text{if } r_{adt} = 0 \end{aligned} \tag{6}$$

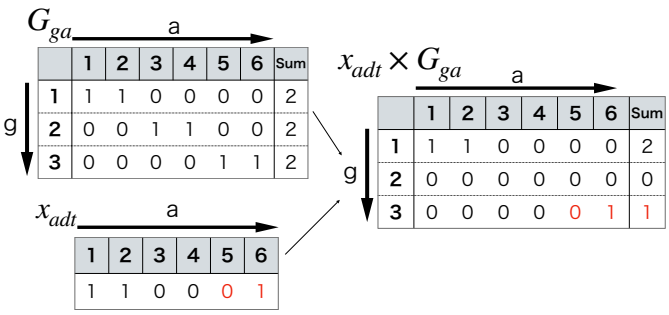


Figure 2: An example of constraint in Eq. 7. Group 1, to which workers 1 and 2 belong, satisfies the constraint because everyone is assigned to the shift ($x_{adt} = 1$). group 2, to which workers 3 and 4 belong, satisfies the constraint because everyone is not assigned to a shift ($x_{adt} = 0$). Group 3, to which workers 5 and 6 belong, does not satisfy the constraint because some workers are assigned ($x_{adt} = 1$) and some are not assigned ($x_{adt} = 0$) to the shift.

$$\forall a \in A, \forall d \in D, \forall g$$
$$\sum_a (x_{adt} \times G_{ga}) = (0 \text{ or } \sum_a G_{ga}) \tag{7}$$

An example of the second constraint (Eq. 7) is shown in Fig. 2. G_{ga} in Fig. 2 is that workers 1 and 2 belong to group 1, workers 3 and 4 belong to group 2, and workers 5 and 6 belong to group 3. The result of allocating shifts under these conditions with fixed workdays d and terms t is shown by x_{adt}. In this constraint, all workers belonging to the same group must be assigned to the same shift. Therefore, in Fig. 2, workers 5 and 6 belonging to group 3 violate the constraint. As can be seen in Fig. 2, group 1 is $\sum_a (x_{adt} \times G_{ga}) = \sum_a G_{ga} = 2$ and satisfies Eq. 7. Group 2 satisfies Eq. 7 with $\sum_a (x_{adt} \times G_{ga}) = 0$, but group 3, which violates the second constraint, does not satisfy Eq. 7 with $\sum_a (x_{adt} \times G_{ga}) \neq 0$ and $\sum_a (x_{adt} \times G_{ga}) \neq \sum_a G_{ga}$.

2.3 QUBO Formulation of Shift Scheduling Problem for Call Centers

From Eq. 9, 10, 11, and 12, the Hamiltonian $\mathscr{H}$ is defined as in Eq. 8. In order to minimize $\mathscr{H}$, QA tries to obtain a value close to the optimal solution. For the optimization property, a small value of $\mathscr{H}_i$ indicates that the number of workers assigned to each term is close to the required number of booths, and a small value of $\mathscr{H}_j$ indicates that the number of assignments of each worker is close to the total desired number of shifts for each worker. If $\mathscr{H}_k$ is 0, it means that all desired shifts of workers are satisfied. If $\mathscr{H}_l$ is 0, it means that the same group of workers were assigned to the same shift.

$$\mathscr{H} = \lambda_i \mathscr{H}_i + \lambda_j \mathscr{H}_j + W_k \mathscr{H}_k + W_l \mathscr{H}_l \tag{8}$$

$$\mathscr{H}_i = \sum_d \sum_t \{(\sum_a x_{adt}) - S_{dt}\}^2 \tag{9}$$

$$\mathscr{H}_j = \sum_a \{(\sum_d \sum_t x_{adt}) - R_a\}^2 \tag{10}$$

$$\mathscr{H}_k = \sum_a \sum_d \sum_t \{(1 - r_{adt}) \times x_{adt}\} \tag{11}$$

Table 2: Number of data and variables.

Worker a	Day d	Term t	#Variables
5	4	3	60
6	5	3	90
6	7	3	126

$$\mathscr{H}_l = \sum_d \sum_t \sum_g [\{(\sum_a G_{ga}) - \sum_a (x_{adt} \times G_{ga})\} \times \sum_a (x_{adt} \times G_{ga})] \tag{12}$$

Eq. 9, 10 define objective functions of Eq. 4, 5 to QUBO, and Eq. 11, 12 define the constraints in Eq. 6, 7 to QUBO, respectively. Eq. 9 is the application of Eq. 4 to the QUBO. It is the sum of the squares of the values obtained by summing the x_{adt} allocated to term t on day d and subtracting them from the desired number of booths, S_{dt}. Squaring the values allows us to take each difference as a positive value. Eq. 10 is the application of Eq. 5 to the QUBO. It is the sum of the squares of the differences obtained by summing the x_{adt} assigned to each worker a and subtracting them from the R_a of each worker. By squaring as in Eq. 9, each difference is received as a positive value.

If a constraint is satisfied, we formulate it to obtain 0. Otherwise, we formulate it to obtain a positive value. Eq. 11 defines the constraint in Eq. 6 as the QUBO. If a shift is assigned (x_{adt} = 1) for a term that the worker does not want ($r_{adt} = 0$), the constraint is violated and the value becomes $(1 - r_{adt}) \times x_{adt} = 1$, and the constraint term takes a positive value. In all other cases, the constraint term is defined to be $(1 - r_{adt}) \times x_{adt} = 0$. Eq. 12 defines the constraint term in Eq. 7 as a QUBO, and Fig. 2 is an example of this equation. Because $\sum_a G_{ga} \geq \sum_a (x_{adt} \times G_{ga})$, this constraint equation never takes negative values, and it always takes positive values when the constraint is violated.

λ_i and λ_j in Eq. 8 are the coefficients of the objective functions $\mathscr{H}_i$ and $\mathscr{H}_j$, respectively. The priority of the objective function in the solution can be adjusted by setting the values of the coefficients λ_i and λ_j differently from the others. For example, if we set $\lambda_i = 1$ and $\lambda_j = 2$, QA is performed to minimize the entire $\mathscr{H}$, so it is easier to obtain a solution with a smaller value of $\mathscr{H}_j$ compared to $\mathscr{H}_i$. This allows the user to decide which condition they want more satisfying. In this paper, we set $\lambda_i = 1$ and $\lambda_j = 1$ in evaluation because we assume a situation where we want both to be satisfied equally. W_k and W_l are the penalty coefficients of the constraint terms $\mathscr{H}_k$ and $\mathscr{H}_l$, respectively. By setting the values of W_k and W_l large enough to exceed the value of the objective function term, the solution can be derived by QA without violating the constraints.

3 Setup

3.1 Workload

As a shift schedule with limited constraints, we evaluate the relationship between execution time, energy, and feasible solution rate with the problem setup described in Sec. 2. We prepared instances of the size that can be executed on a quantum annealing machine. The number of terms is fixed, while the number of variables is changed as shown in Table 2. We created each instance by ourselves. We made sure that each instance had a feasible solution. Before the evaluation, we tune the penalty coefficients and `chain_strength` for QA in Sec. 4.1. This significantly reduces the number of constraint-violating solutions and the execution time in the evaluation. For ease of understanding, $\mathscr{H}$ is denoted by energy in Sec. 3 and 4.

3.2 Methods

The environment using a classical computer other than D-Wave was Amazon SageMaker notebook in Amazon Web Services. The server instance was ml.t3.medium with two vCPUs and 4GiB of

memory, using Python version 3.6.13. The method and solver used for the evaluation are described below:

Quantum annealing (D-Wave Advantage). We used the Advantage system 1.1 [1], a hardware implementation of the quantum annealing machine provided by D-Wave systems as a cloud service. We set three parameters, `num_reads`, `annealing_time`, and `chain_strength`, apart from the default QA settings to improve QA performance. We set `num_reads`=1,000 that is the number of run of the QA. `annealing_time` sets duration, in microseconds with a resolution of 0.01 μs for D-Wave Advantage, of quantum annealing time, per read. We set different `annealing_time` for each evaluation. We wrote the value of `annealing_time` in each section of the evaluation. `chain_strength` is used when embedding an all-coupled QUBO into a particular graph topology of qubits implemented in the D-Wave machine. We tuned and determined the value of `chain_strength` in Sec. 4.1.3. The process of determining the penalty coefficients is described in Sec. 4.1.1 and 4.1.2.

Simulated quantum annealing (SQA). We used OpenJij [5] as a quantum annealing simulator. This is an open source software with Python API for simulated QA (SQA). In this evaluation, we used version 0.1.1 of SQA. The SQA settings are beta=10, gamma=1.0, trotter=10, and `num_reads`=1000.

Satisfiability modulo theories (SMT). Satisfiability (SAT) is a problem to determine whether a constraint equation has a solution or not. Satisfiability modulo theories (SMT) is a version of SAT in which the interpretation of symbols is constrained by background theory. We formulated this problem using SMT and evaluated it with the Z3 Theorem Prover [9]. To apply the shift scheduling problem to SMT, we need to formulate QUBO's objective functions (Eq. 9, 10) as constraints. Eq. 13 is the initial condition for x_{adt} to be included in the SMT solver. Eq. 14 and 15 correspond to Eq. 9 and 10, respectively. We used Eq. 6 and 7 as is for constraints.

$$\begin{gathered}\forall a \in A,\ \forall d \in D,\ \forall t \in T \\ x_{adt} \in \mathbb{Z} \\ 0 \leq x_{adt} \leq 1\end{gathered} \tag{13}$$

$$\forall d \in D,\ \forall t \in T,\quad \sum_{a} x_{adt} \geq S_{dt} \tag{14}$$

$$\forall a \in A,\quad \sum_{d}\sum_{t} x_{adt} \geq R_a \tag{15}$$

Tabu search (TS). Tabu search (TS) is one of the metaheuristics used to solve QUBO. By swapping the variables of the solution, the search range is expanded to find the optimal solution. The swapped solution is added to the tabu tenure list to avoid finding the same solution in order not to search for the solution in this list. We used Dimod [2] from D-Wave Systems. The settings are default, except for `num_reads`=1,000 that is the number of run of the TS algorithm. We recorded the energy and execution time while varying the value of `timeout`.

Simulated annealing (SA). Simulated annealing (SA) [17] is a technique inspired by annealing in which a metal is heated and then the temperature is gradually lowered to obtain a clean crystal while maintaining a low energy state. Dimod [2] implements SA, so this is what we used. We recorded the energy and execution time while varying the value of `num_sweeps`. The settings are default, except for `num_sweeps`=1,000 that is the number of run of the SA.

Mixed integer programming (MIP). Mixed integer programming (MIP) is a linear programming problem that includes the integer decision variables. In order to apply our problem to an MIP solver, we introduced a new variable $y_{a_i d_i t_i a_j d_j t_j}$ to the second-order term $(x_{a_i d_i t_i} \times x_{a_j d_j t_j})$, and

the following constraints are added so that the variables are below.

$$\begin{gathered}\forall a_i \in A,\ \forall d_i \in D,\ \forall t_i \in T,\ \forall a_j \in A,\ \forall d_j \in D,\ \forall t_j \in T \\ 0 \leq x_{a_i d_i t_i} \leq 1 \\ 0 \leq x_{a_j d_j t_j} \leq 1 \\ y_{a_i d_i t_i a_j d_j t_j} \geq x_{a_i d_i t_i} + x_{a_j d_j t_j} - 1 \\ y_{a_i d_i t_i a_j d_j t_j} \leq x_{a_i d_i t_i} \\ y_{a_i d_i t_i a_j d_j t_j} \leq x_{a_j d_j t_j}\end{gathered} \tag{16}$$

From Eq. 3.2, we reshape Eq. 9-12 as follows.

$$\underset{x_{adt}, y_{a_i dt a_j dt}}{\arg\min} \sum_{d \in D} \sum_{t \in T} \{(\sum_{a_i \in A} \sum_{a_j \in A} y_{a_i dt a_j dt}) - 2(\sum_{a \in A} x_{adt}) \times S_{dt} + (S_{dt})^2)\} \tag{17}$$

$$\underset{x_{adt}, y_{ad_i t_i ad_j t_j}}{\arg\min} \sum_{a \in A} \{(\sum_{d_i \in D} \sum_{t_i \in T} \sum_{d_j \in D} \sum_{t_j \in T} y_{ad_i t_i ad_j t_j}) - 2(\sum_{d \in D} \sum_{t \in T} x_{adt}) \times R_a + (R_a)^2\} \tag{18}$$

$$\begin{gathered}\forall a \in A, \forall d \in D, \forall t \in T \\ (1 - r_{adt}) x_{adt} = 0\end{gathered} \tag{19}$$

$$\begin{gathered}\forall d \in D, \forall t \in T, \forall g \\ (\sum_{a} G_{ga}) \times \sum_{a} (x_{adt} G_{ga}) - \sum_{a_i \in A} \sum_{a_j \in A} (G_{ga_i} \times G_{ga_j} \times y_{a_i dt a_j dt}) = 0\end{gathered} \tag{20}$$

Eq. 17, 18, 19, and 20 correspond to Eq. 9, 10, 11, and 12, respectively. We used SCIP [6], which is provided as a third-party to Google OR-Tools' MIP solver [3]. SCIP has a method to judge whether a solution is an optimal solution or not, so we can obtain an optimal solution for each instance.

Mixed integer quadratic programming (MIQP). Whereas MIP can handle variables up to first-order expressions, mixed integer quadratic programming (MIQP) can handle variables up to second-order. The solver is Gurobi [4]. The version is 9.5.2. Gurobi, like SCIP, can judge whether a solution is an optimal solution or not.

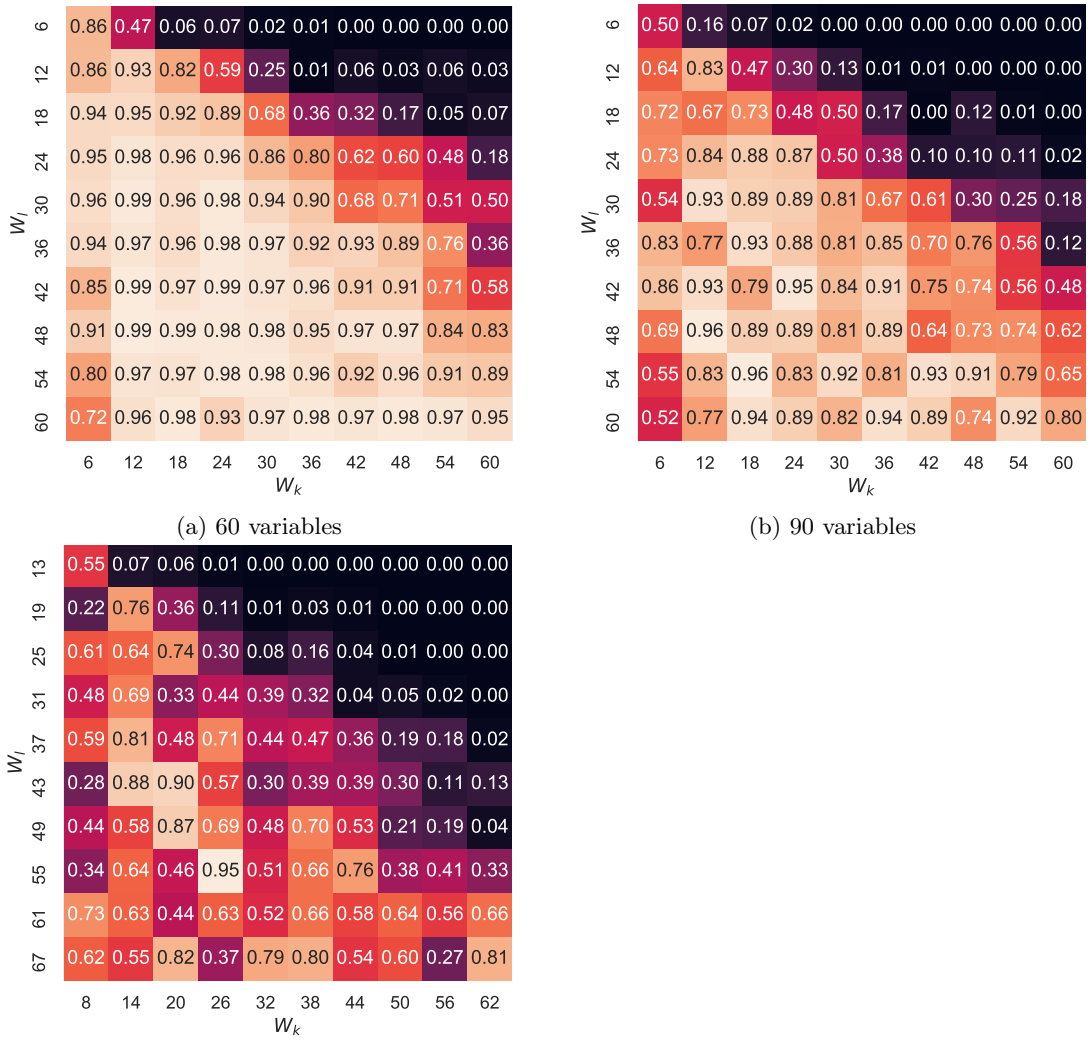
4 Evaluation

4.1 Parameter Tuning

We adjust the parameters of the QA machine, and W_k and W_l in Eq. 8 and `chain_strength` to lower the energy and upper the rate of feasible solution. We set $\lambda_i = 1$ and $\lambda_j = 1$ because we assume a situation where we want both objective function to be small equally. The number of executing annealing processes (`num_reads`) is set to 1,000 when adjusting the parameters.

4.1.1 Determination of ratio of penalty coefficients

First, we determine the ratio of W_k and W_l in Eq. 8. By changing the ratios, the constraint terms Eq. 11 and Eq. 12 are balanced so that the values of either expression are not too large. Note that we only determined the ratios of penalty coefficients in this section, not the actual ratios. Fig. 3 shows heat maps of the probabilities of feasible solutions obtained for each number of variables. By looking at the heat map, we select values that are balanced in the probability of feasible solutions.



(a) 60 variables

(b) 90 variables

(c) 126 variables

Figure 3: Heatmap with varied parameters

We can see here that the probability of obtaining a feasible solution decreases as the value of one of the variables becomes larger and the value of the other becomes smaller, regardless of the number of variables. Table 3 shows the ratio of W_k to W_l determined through these heat maps for each number of variables, which were then used for the following evaluations.

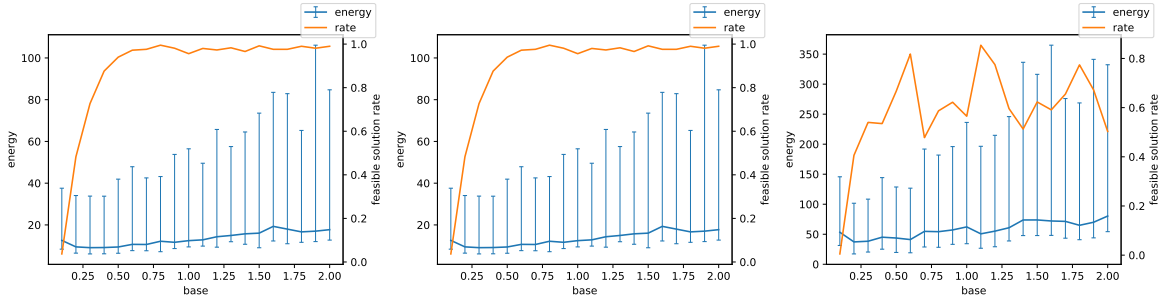
4.1.2 Determination of value to be multiplied by penalty coefficient

We introduce the value `base` to show how the QA results change as the value of the penalty coefficients increases. `base` is the value to be multiplied by the ratio of the penalty coefficient determined in Sec. 4.1.1. We execute QA with varying the value `base`, evaluate the result, and determine the value of the penalty coefficients. The goal is to set a `base` value that does not cause any violation of the constraints and that has low energy. Here, we measure the probability of a feasible solution and energy by varying the `base` value.

In Fig. 4, the vertical axis on the right shows the probability of obtaining a feasible solution when the `base` is varied, and the vertical axis on the left shows the minimum and maximum energy values for the solutions obtained for `num_reads` (=1,000) for each `base`, with the lower and upper limits of the error bars, and the average value shown as a line graph. When the `base` is small, the

Table 3: Ratio of W_k, W_l, `base`.

#Variables	W_k ratio	W_l ratio	`base`
60	15	24	0.5
90	27	12	0.6
126	55	26	0.3



(a) 60 variables (b) 90 variables (c) 126 variables

Figure 4: Feasible solutions and energy for varying base values

energy is stable at a low value, but the probability of obtaining a feasible solution is also low. On the other hand, when the `base` is large, the probability of obtaining a feasible solution is high, but the energy is large, so it can be said that there is a trade-off relationship.

The `base` is determined to be as small as possible, namely, the value at which the probability of obtaining a feasible solution begins to converge slowly. `base` values for each number are listed in Table 3. The values of the penalty coefficients determined by tuning are shown in Table 4, as multiplied by the ratio of the penalty coefficients determined in Sec. 4.1.1 and `base`.

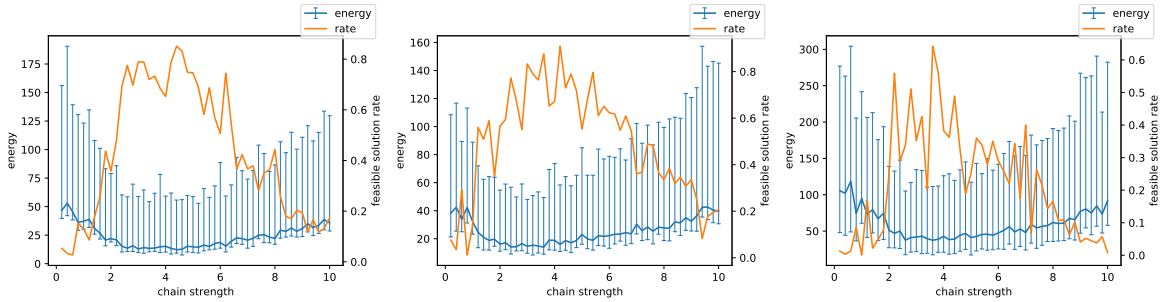
4.1.3 Determination of `chain_strength`

Finally, we tune the value of `chain_strength`. The parameter, `chain_strength`, is used when embedding an all-coupled QUBO into a particular graph topology of qubits implemented in the D-Wave machine. In particular, D-Wave advantage uses Pegasus topology. The shift scheduling problem for call centers is not always a Pegasus topology because the interrelationships between the variables change each time depending on the instances. Therefore, we need all-coupled embedding when solving the shift scheduling problem for call centers. To realize all-coupled embedding on hardware, we need to consider some qubits as a group and make them in the same direction, and the strength of the chain between the qubits is given by the parameter `chain_strength` in the D-Wave annealing machine. If the value of `chain_strength` is large, the strength of the chain is stronger. If the value of `chain_strength` is not sufficiently large, chain break will occur, so it is necessary to tune the `chain_strength`. In Fig. 5, the vertical axis on the right shows the probability of obtaining a feasible solution when the value of `chain_strength` is varied, and the vertical axis on the left shows the minimum and maximum energy values for the solutions obtained for `num_reads` (=1,000) for each `chain_strength`, with the lower and upper limits of the error bars, and the average value shown as a line graph. `annealing_time` is set to 20 in all variables.

We discuss the results of the 60 variables. Note the rate of feasible solution (orange line). Until the `chain_strength` value of 2.4, the rate of feasible solution keeps increasing up to about 0.7. While the `chain_strength` is 2.4-5.4, a high rate (0.7-0.85) is continuously obtained. When the `chain_strength` is greater than 5.4, the rate keeps decreasing with increasing value. In terms of energy, the minimum value of energy gradually decreases to 15 for a `chain_strength` of up to 2.4. As long as the `chain_strength` is between 2.4 and 5.2, the minimum value of energy continues to be low value (12.2-15). When `chain_strength` is higher than 5.4, the energy is larger as the

Table 4: Value of each parameter after tuning

#Variables	W_k	W_l	`chain_strength`
60	7.5	12	4.4
90	16.2	7.2	3.6
126	16.5	7.8	3.8



(a) 60 variables (b) 90 variables (c) 126 variables

Figure 5: Feasible solutions and energy for varying `chain_strength`

`chain_strength` increases. This means that we can get a good solution with a `chain_strength` of 2.4-5.2, because the rate of feasible solution is high and the energy is low. Similarly, for 90 and 126 variables, until the `chain_strength` reaches a specific value, the rate increases and the energy is lowered. Then, it maintains high energy and low energy, but if the `chain_strength` becomes too large, the rate decreases and energy becomes high. Taking this into account this, we determined each `chain_strength` as shown in Table 4.

4.2 Performance Metric: Time To Solution

Because QA, TS, SA, and SQA are a types of local search method using QUBO, they cannot be guaranteed to obtain a feasible solution in only one run, so it is necessary to run them a sufficient number of times. In this paper, time to solution means the execution time to obtain a feasible solution. We consider the probability of obtaining a feasible solution when the number of executions is increased. For the probability r_1 of obtaining a feasible solution in a single annealing process, the probability r_m of obtaining a feasible solution in m runs is as follows.

$$r_m = 1 - (1 - r_1)^m \tag{21}$$

From Eq. 21, the number of times m required to obtain a feasible solution with probability r_m is as follows.

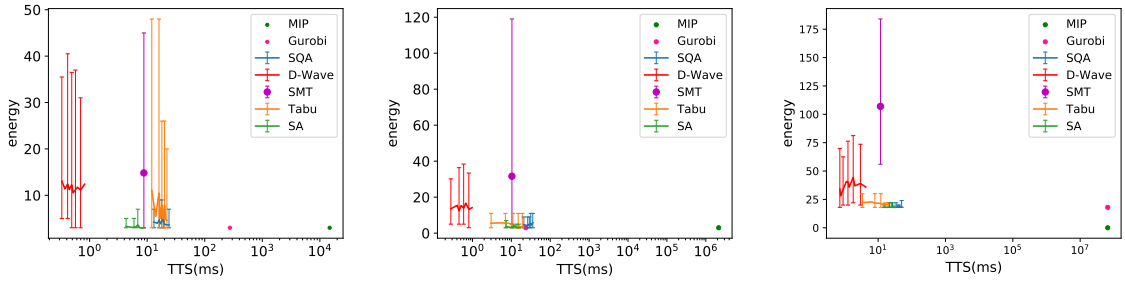
$$m = \left\lceil \frac{log(1 - r_m)}{log(1 - r_1)} \right\rceil \tag{22}$$

For example, if the probability of obtaining a feasible solution is 50% ($r_1 = 0.50$) and the execution time is τ, the TTS to obtain a feasible solution with a probability of 99% ($r_m = 0.99$) or more mustsol be executed $m = 7$ times by substituting the respective values in Eq. 23.

The Time to solution (TTS) is obtained by multiplying m by the execution time of one run τ.

$$TTS(\tau, r_1) = \tau \left\lceil \frac{log(1 - r_m)}{log(1 - r_1)} \right\rceil \tag{23}$$

If the probability of obtaining a feasible solution r_1 is small even if the execution time per execution (τ) is short, the number of executions m of Eq. 23 becomes large, and as a result, TTS



(a) 60 variables (b) 90 variables (c) 126 variables

Figure 6: TTS Comparison in log scale

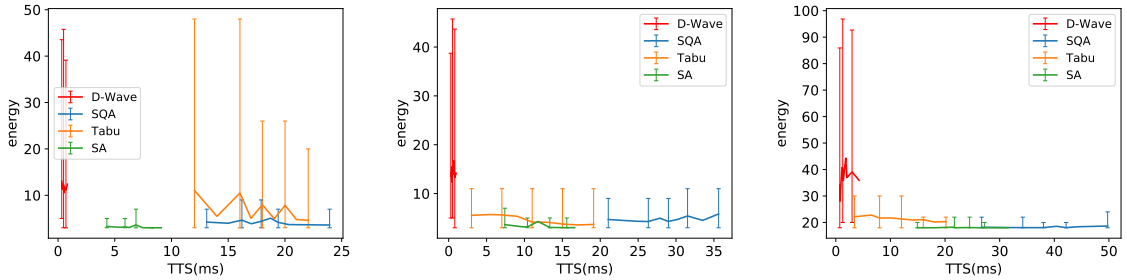
may become large. On the other hand, if the probability of obtaining a feasible solution is large even if the execution time per execution is long, the number of executions m can be reduced, and thus TTS can be small. In this paper, the value of r_m is set to 0.99. Therefore, by measuring TTS, it is possible to calculate the time required to obtain a feasible solution 99% of the time. TTS is used for D-Wave, SA, SQA, and TS, while MIP, MIQP, and SMT use the measured execution time directly. In addition, TTS obtained by these methods is changed by varying the parameters that depend on the execution time (e.g., the `annealing_time`). We also evaluate multiple TTSs on each method by changing the parameters. The details of D-Wave's TTS calculation are as follows. First, we executed QA with `num_reads`=1,000 and measured the number of times to obtain a feasible solution and the execution time. The number of times to obtain a feasible solution was divided by `num_reads` (=1,000) and the probability of obtaining a feasible solution, r_1, was obtained. An execution time τ was obtained by dividing the execution time obtained by D-Wave's execution time measurement function by `num_reads` (=1,000). We obtained TTS by applying the obtained τ and r_1 to Eq. 23.

4.3 Result

Fig. 6 shows the results of the seven methods we evaluated in TTS on the horizontal axis in log scale and the energy on the vertical axis. For all methods except MIP and Gurobi the range of energy of the obtained solution is represented by error bars and the average value is represented by a line graph, since the obtained solution varies with each run. Only when the variable 126, Gurobi could not obtain a solution in sufficient time. Therefore, we executed a timeout at the same execution time as the MIP, forcing the execution to be terminated. We can see that as the number of variables increases, the SMT solver obtains a higher energy than the other methods. This is due to the fact that the problem formulated in QUBO cannot be formulated exactly the same way in SMT, and the SMT solver does not have the ability to minimize energy. Therefore, it is not appropriate to compare it with other metaheuristics. As mentioned in section 3.2, MIP can be checked to see if the solution is an optimal solution. In all instances, the MIP's solution is an optimal solution, so the MIP provides the lowest energy. The TTS of the MIP worsens as the number of variables increases, and it is found to be 10^2-10^6 times slower than the other methods, although it provides an optimal solution.

Fig. 7 shows the comparison of the four methods from Fig. 6, excluding the MIP, Gurobi, and SMT. Note that the horizontal axis is no longer a log scale. First, we compare D-Wave with the other methods. D-Wave is the fastest in terms of TTS, obtaining a solution in 0.3-4.0 ms even with more variables, which is 3-14 times faster than the methods with the fastest TTS except D-Wave. In terms of energy, D-Wave has the largest energy range and average value, except for the TS with 60 variables. We now compare the minimum value of energy for D-Wave with the other methods.

For 60 variables, it has the same value as the other methods (energy=3). 90 variables, the minimum value is 5, which is only 2 away from the minimum value of the other methods (energy=3), which is close to the exact solution, so a good solution is obtained. 126 variables, the minimum value



(a) 60 variables (b) 90 variables (c) 126 variables

Figure 7: TTS Comparison in Metaheuristics

is 18. For 126 variables, the minimum value is 18, and the other methods also have a minimum value of 18. Therefore, the minimum value of D-Wave is almost the same as that of the other methods for all variables, and although the accuracy may be inferior to that of the other methods, a good solution can be obtained. In summary, D-Wave can obtain the fastest TTS, but the accuracy of the solution is not as good as the other methods. However, the minimum value is almost the same as that of the other methods, so it is possible to obtain a good solution.

Next, we compare the other three methods except D-Wave. First, we compare the TTS. The TTS of TS is 12.0 ms, and the TTS of SQA is 14.3 ms, which is almost equal. 90 variables TTS is 3.0 ms for TS, 7.4 ms for SA, and 21.1 ms for SQA. The TTS for 126 variables is the same as that for 90 variables: 3.4 ms for TS, 14.9 ms for SA, and 26.7 ms for SQA. Next, we look at energy. Looking at the minimum values, we can see that the three methods are able to obtain the same minimum values for all variables, indicating that good solutions can be obtained. Among them, the minimum values of the three methods for 60 variables and 90 variables are exact solutions.

The average value of energy for 60 variables is the smallest for SA (3.0-3.3), the second smallest for SQA (3.6-5.0), and the largest for TS (4.5-11.0). In the case of 90 variables, the mean value of energy for SA is the smallest at 3.0-4.3, SQA is 4.3-5.8, and TS is 3.6-5.4, so SQA and TS are almost equal. In the case of 126 variables, the mean value of energy for SA is 18.0-18.3, so they are almost equal, and TS is 20.2-22.8, which is larger than the other two.

Therefore, there is no significant difference in the energy obtained by any of the three methods, but SA has the highest accuracy in obtaining a good solution among the three. Since the minimum value of energy is the same for all three methods, it can be said that they are able to obtain good solutions.

4.4 Discussion

First, we focus on MIP. The results for the three instances in Fig. 6 show that MIP has the lowest energy value and that all of them are optimal solutions. However, in 126 variables, it was 10^1-10^4 times slower than the other solvers except for the MIQP solver, Gurobi. Since MIP cannot handle quadratic formulas, it is necessary to express linear variables in terms of quadratic variables, as in Eq. 3.2. When solving for instances with n variables, it is necessary to prepare $n^2 + n$ variables. Therefore, the search space for the solution is dramatically expanded, which may be the reason why it takes so long time to obtain the solution.

Gurobi (MIQP) obtains solutions faster than MIP except for 126 variables. In particular, in 90 variables, the optimal solution is obtained in execution times close to those of meta-heuristics. In 126 variables, the optimal solution could not be obtained because the solution was considered to be stuck in the local optimal solution. However, because the minimum energy is the same as that of the other methods except for MIP, it can be said that a good solution is obtained, although it is not the optimal solution.

Since SMT is a solver for solving constraint satisfaction problems, if an instance has a feasible

solution, then the feasible solution can certainly be obtained. According to Fig. 6 and Fig. 7, it is possible to obtain the solution in the same execution time as for metaheuristics, which means that this method is very fast. However, SMT does not have the ability to minimize the objective function, so the solution energy is randomly obtained from the set of feasible solutions, and it does not provide a good solution compared to QA.

In terms of execution time, SA requires more execution time (`num_sweep`) to obtain a feasible solution when the problem instance is large, and the difference in execution time between SA and QA is increasing. However, in terms of energy, it can be seen that in all instances SA has a small energy range and consistently provides good solutions.

TS has a slower execution time and a larger energy range than QA with 60 variables. However, as the number of variables increases, the difference between the execution time and QA becomes smaller, and the range of energy also becomes smaller. This result is against intuition. This indicates that, in addition to the size of the problem instance, other factors determine whether the TA solution is good or bad. Finding this reason is left for future work.

Since SQA is a simulation of QA on a classical computer, its execution time is longer than that of QA. However, since SQA is the second smallest energy range after SA, it is theoretically possible that QA also provides a low energy. However, the range of actual QA energies is large. Theoretically, QA is said to obtain the ground state by reducing the transverse magnetic field with a sufficiently long annealing time, but as shown in Fig. 7, there is no significant change in the energy of the solution obtained even if the annealing time is actually increased. Therefore, one of the possible reason why the solution is not stable is thought to be a quantum error that breaks the quantum state. To stabilize energy, device technology to increase coherent time and error correction technology are needed.

5 Related Work

Metaheuristics [10] have been expected to be a powerful method to solve combinatorial optimization problems and its continuously. Metaheuristics represent a higher level of heuristics, and it is expected to be able to obtain the optimal solution in a short time, even for very large problem sizes. Some examples are tabu search (TS) [14] [12], simulated annealing (SA) [17], and QA is one of them.

There are not many examples of QA being applied in the real-world, except for the following. Florian et al. [19] used QA to optimize traffic flow in Beijing. Tobias et al. [21] realized air traffic control using QA. Sheir et al. [23] used QA to solved the multi-car paint shop problem, which optimizes the number of color switches between cars in a paint shop queue during manufacturing. They then compare the five classical methods with three quantum annealers.

There are studies that automate shift scheduling problem for hospital nurses by classical methods. Ahmed et al. [20] studied the real-world nurse scheduling problem, modeled it carefully, and obtained a solution by using TS. Walter and Marion [15] solved the nurse scheduling problem by modeling the Vienna hospital compound consisting of 15 public hospitals by using the ant colony optimization [11] and obtaining a high quality solution compared to SA.

There are no studies that solve shift scheduling problem for call center operators by using QA, to our knowledge. Examples of applying QA to the nurse scheduling problem include the following: Ikeda et al. [16] formulated the nurse scheduling problem in QUBO, obtain the solution using QA and evaluate it. Arindam et al. [8] formulated not only the nurse scheduling problem, but also the physician scheduling problem and the nurse-physician scheduling problem assuming COVID-19 in QUBO. The quality of the solutions solved by QA is compared with that of SA. However, neither of the two researches applying QA to nurse scheduling problem compares the accuracy of the solution with other methods said to be effective for combinatorial optimization problems. They also do not compare execution time of QA with that of other methods.

6 Conclusion

In this paper, we formulated the QUBO for a shift scheduling problem for call centers. Then, we tuned the penalty coefficients of a small shift scheduling problem for call centers using the quantum annealing machine of D-Wave systems, which is a hardware implementation of QA, and showed the transition of the probability of obtaining a feasible solution and the energy. We then show the effectiveness of QA for this problem and compared QA to 6 methods using classical computers. The results showed that the QA provided a solution with a TTS of 0.3 ms to 4.0 ms, which is 3 to 14 times faster than the fastest among other methods.

Since its average value of energy was higher than that of the other methods, the accuracy of the solution was somewhat worse, but the minimum value of energy was almost the same as that of the other methods for all variables, indicating that it is possible to obtain a good enough solution. We also showed that not only QA but also other methods were effective for this problem.

Current quantum annealing machines are limited in the number of qubits and need embedding because they are not all-coupled graphs, so the size of the problem that can be solved is small. However, the actual shift scheduling problem for call centers is expected to have more than 10,000 variables, such as 100 workers, 30 days, and 4 terms. Advances in hardware and having more qubits will solve that problem, but not in the near future. The future work on software is to explore a method to process a large scale problem with high quality, low latency and minimum number of qubits.

References

[1] D-Wave Systems. `https://www.dwavesys.com/`.

[2] Dimod. `https://docs.ocean.dwavesys.com/en/stable/docs_dimod/`.

[3] Google OR-Tools. `https://developers.google.com/optimization`.

[4] Gurobi. `https://www.gurobi.com/`.

[5] OpenJij. `https://www.openjij.org/`.

[6] SCIP. `https://scipopt.org/#scipoptsuite`.

[7] Francisco Barahona. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241–3253, 1982.

[8] Kunal Das, Sahil Zaman, Arindam Sadhu, Asmita Banerjee, and Faisal Shah Khan. Quantum annealing for solving a nurse-physician scheduling problem in covid-19 clinics, 2020.

[9] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[10] Tansel Dokeroglua, Ender Sevincb, Tayfun Kucukyilmaza, and Ahmet Cosar. A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137:106040, 2019.

[11] Marco Dorigo and Thomas Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. *Handbook of metaheuristics*, pages 250–285, 2003.

[12] Fred Glover. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32, 1990.

[13] Fred Glover and Claude Mcmillan. The general employee scheduling problem: An integration of MS and AI. *Computers & operations research*, 13(5):563–573, 1986.

[14] Fred Grover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[15] Walter J Gutjahr and Marion S Rauner. An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria. *Computers & Operations Research*, 34(3):642–666, 2007.

[16] Kazuki Ikeda, Yuma Nakamura, and Travis S. Humble. Application of quantum annealing to nurse scheduling problem. *Scientific reports*, 9(1):1–10, 2019.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[18] Holmes E. Miller, William P. Pierskalla, and Gustave J. Rath. Nurse scheduling using mathematical programming. *Operations Research*, 24(5):857–870, 1976.

[19] Florian Neukart, Gabriele Compostella, Christian Seidel, David von Dollen, Sheir Yarkoni, and Bob Parney. Traffic flow optimization using a quantum annealer. *Frontiers in ICT*, 4:29, 2017.

[20] Ahmed Oughalime, Wan Rosmanira Ismail, and Liong Choong Yeun. A tabu search approach to the nurse scheduling problem. In *2008 International Symposium on Information Technology*, volume 1, pages 1–7. IEEE, 2008.

[21] Tobias Stollenwerk, Bryan O' Gorman, Davide Venturelli, Salvatore Mandra, Olga Rodionova, Hokkwan Ng, Banavar Sridhar, Eleanor Gilbert Rieffel, and Rupak Biswas. Quantum annealing applied to de-conflicting optimal trajectories for air traffic management. *IEEE transactions on intelligent transportation systems*, 21(1):285–297, 2019.

[22] Kadowaki Tadashi and Nishimori Hidetoshi. Quantum annealing in the transverse Ising model. *Physical Review E*, 58(5):5355–5363, 1998.

[23] Sheir Yarkoni, Alex Alekseyenko, Michael Streif, David Von Dollen, Florian Neukart, and Thomas Bäck. Multi-car paint shop optimization with quantum annealing. In *IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 35–41. IEEE, 2021.