Description: Third Generation Digital Annealer Technology

Hiroshi Nakayama(*1), Junpei Koyama(*1), Noboru Yoneoka(*1), Toshiyuki Miyazawa(*1)

1. Introduction

Digital Annealer (DA) is a new computing technology that Fujitsu Laboratories has started researching and developing in the mid-2010s for high-speed solving of combinatorial optimization problems, which are difficult to solve with existing general-purpose computers. The DA can handle combinatorial optimization problems mapped to an Ising model that is expressed by the energy function in which the spin states (−1, +1) are converted into binary values (0, 1). The DA can rapidly solve those problems by searching for a ground state, based on the Markov-Chain Monte Carlo (MCMC) method, where the energy of the Ising model system is the lowest. The inspiration for the architecture of the DA comes from quantum computers, which repeatedly apply unitary operations to multiple qubits, whereas the DA repeatedly applies stochastic transitions to classical bits which correspond to the aforementioned binary values. In order to accelerate this iterative operation of stochastic transitions for digital circuits, we have developed an architecture that performs parallel computation of subtraction operations for the finite differences of the energy function and its stochastic evaluation, in the collaboration with the University of Toronto in Canada. This parallel structure of the circuits eliminates the data transfer bottleneck between the memory and the arithmetic units by reducing movement of Ising model coupling coefficient data as much as possible [1].

In May 2018, Fujitsu commercialized the results of this research by launching its first generation DA cloud service for 1,024-bit problems [2], followed by the launch of a second generation cloud service in December 2018 characteristically equipped with a dedicated processor (Digital Annealing Unit: DAU) [3][4] capable of handling 8,192 bits. Compared with the first generation, this greatly increased the scale of problems that could be handled, and offered better problem-solving performance [5]. Unlike quantum annealing machines, DA technology has various advantages that make it suitable for practical use, such as not requiring cryogenic environment and having the features of fully coupled bit connectivity and high coupling resolution up to 64bits. Solving business problems with the DA has resulted in great benefits for many organizations. This technology has been applied to various fields in the real world, including logistics, finance, and drug discovery [6][7][8]. Research using the DA is also being actively conducted at universities and research institutes all over the world to develop applications in new fields [9–14].

With the first and second generation DAs, energy functions to be minimized are handled as a quadratic unconstrained binary optimization (QUBO). Often, combinatorial optimization problems are subject to various constraints. In these situations, penalty terms, whose evaluation value increases with the violation of constraints, are also expressed in a binary quadratic form and added to the cost terms representing the original optimization targets. However, it is known that as the

(*1) Optimization Computing Project, ICT Systems Laboratory, Research Unit of Fujitsu Research

numbers of constraints and variables increase, it becomes notoriously difficult to reach optimal or sub-optimal solutions [15]. To tackle this problem we have developed the third generation DA. This is a problem-solving system with a hybrid software and hardware configuration. The software component effectively finds good solutions in a large-scale solution space of 100,000 bits by analyzing constraint-violating conditions. The hardware component searches for optimal solutions in the neighborhood of these good solutions. The features of this third generation DA are described in Section 2 of this paper, and its performance evaluation results are presented in Section 3.

2. Third generation DA

2.1 Overview

The overall configuration of the third generation DA is shown in Fig. 1. This is a hybrid problem-solving system in which a software intervention layer (SIL) cooperates with a search core to find optimal or sub-optimal solutions to a binary quadratic programming (BQP) problem of up to 100,000 bits.

The energy function of the system input consists of the two kinds of separated terms: an aggregated cost term and an aggregated penalty term, each of which can be set in binary quadratic form. It enables the system to analyze the violation of constraints, which was not possible with the first and second generation. To better handle constraints that frequently arise in practical problems, we have added two types of input: one is for a group of variables in which the sum of binary variables is 1 (one-hot constraints), and the other is for linear inequality constraints. Functionalities for these equality and inequality constraints will be described in Section 2.2. By allowing the user to provide constraints explicitly, the third generation DA achieves better solving performance for these constrained problems. Furthermore, this interface simplifies the formulation of problems by eliminating the need to transform linear inequality constraints into penalty terms, as it was required with the previous generations of DA. These new interfaces, which take constrained problems into account in the third generation DA, are called BQP IF (as shown in Fig. 1) to distinguish it from QUBO IF in the first and second generation DAs(*2).

The SIL (as shown in Fig. 1) is a software layer that controls the search core to have it perform efficient searches in the large-solution space, based on the analysis on the problem-solving progress. Its main functions are automatic temperature control, search start points generation, and automatic penalty coefficient adjustment.

One of characteristics of the DA is MCMC-based search which controls the probability of state transitions by a control parameter called “temperature.” The DAU hardware (described below) incorporates a replica exchange method that executes parallel MCMC-based searches at multiple temperatures and exchanges the solution states between adjacent temperatures at appropriate times in order to escape from local minima

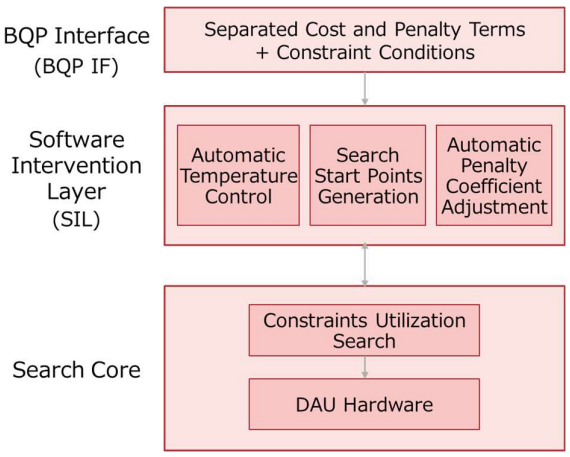

Fig. 1: Overall configuration of the third generation DA

(*2) The third generation DA receives a QUBO prepared for the second generation DA, by putting the QUBO to the cost terms of the energy function of the BQP IF and not putting any penalty terms and inequality constraints.

[3][4][16]. In the replica exchange method, the maximum /minimum temperature settings and the temperature interval setting have a significant effect on the solution quality and convergence time. The automatic temperature control sets these parameters appropriately by analyzing the distribution of the input energy function.

The search start points generation is a function that generates new search start points from a set of past solution candidates obtained by the search core. When the search core has stalled in a local minimum, it generates new starting points, with randomness taken into consideration, which aims at guaranteeing diversity of search in a large space.

The automatic penalty coefficient adjustment analyzes the influence of a cost term and a penalty term from the solution result obtained by the search core, and automatically adjusts the penalty coefficient (i.e., the weight of penalty term) so that the search core can easily find better solutions in subsequent search process.

The SIL stops the search process when the evaluated energy as the lowest in the process is not being updated over a criteria time, when the total elapsed time exceeds the time limit, or when the target energy is attained, if the value is provided by the user. The SIL outputs a user-specified number of solution candidates, including the best solution, including the pest-found solution, after the solution search is completed.

The search core consists of a software-implemented constraint utilization search and DAU hardware. The constraint utilization search actively utilizes constraints in its search process, such as penalty term, one-hot constraints and inequality constraints, all set via BQP IF. The search process starts from a point generated by the SIL in the large-scale solution space to finally find a good solution by evaluating the violation of constraints above. The DAU hardware performs ultra-fast search based on MCMC to find the optimal solution in the neighborhood of this good solution.

By operating cooperatively to make use of the characteristics of each search, the third generation DA is able to handle large-scale problems that could not be handled by the second generation DA, and by tackling constrained problems that are frequently encountered in practical problems, it can be used for a wider range of applications.

2.2 BQP IF and constraint-related functions

This section provides a detailed description of the constraint-related functions of the third generation DA including the BQP IF specification.

(1) Separated cost and penalty terms interface

Most practical optimization problems are subject to constraints that define the feasible search space. To handle those problems as minimizing a QUBO energy function, constraints are incorporated into the energy function as a penalty term with strictly positive values when any constraint is violated (Eq. (1)), and the energy function is expressed as a binary quadratic formula (Eq. (2)).

$$E(\boldsymbol{x}) = C(\boldsymbol{x}) + \alpha P(\boldsymbol{x}) \quad (1)$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$: A set of binary variables
$C(\boldsymbol{x})$: Cost term to be minimized as a target
$P(\boldsymbol{x})$: Penalty term (constraint violation when $P(x) > 0$)
α : Penalty coefficient (positive)

$$E(\boldsymbol{x}) = \sum_{i<j} J_{ij} x_i x_j + \sum_i h_i x_i + c \quad (2)$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$: A set of binary variables
J_{ij} : Coupling coefficient between variables x_i and x_j
h_i : Bias coefficient for variable x_i
c : Constant term

In the first and second generation DA, this binary quadratic function (Eq.(2)), in which $C(\boldsymbol{x})$ and $P(\boldsymbol{x})$ are up-front combined, should be set via the corresponding QUBO IF. On the other hand, in the third generation DA, those terms are set as separate binary

quadratic forms (Eqs.(3) and (4)) via the corresponding BQP IF.

$$C(\boldsymbol{x}) = \sum_{i<j} J_{ij}^{c} x_i x_j + \sum_i h_i^{c} x_i + c^{c} \quad (3)$$

$$P(\boldsymbol{x}) = \sum_{i<j} J_{ij}^{p} x_i x_j + \sum_i h_i^{p} x_i + c^{p} \quad (4)$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$: A set of binary variables

J_{ij}^{c}, h_i^{c}, c^{c} : Coupling coefficient, bias coefficient, and constant term of Cost term

j_{ij}^{p}, h_i^{p}, c^{p} : Coupling coefficient, bias coefficient, and constant term of Penalty term

The penalty coefficient α in Eq. (1), which represents the weight of the penalty term, can be manually set to a strictly positive integer value or automatically adjusted.

The remainnder of this section uses the Quadratic Assignment Problem (QAP) to explain in more details the difference between the conventional QUBO formulation process and the formulation process when using the third generation DA.

The facility location problem, which is one kind of QAP, involves assigning n facilities to n locations in such a way as to minimize the cost term expressed as the sum of the distance between each pair of facilities multiplied by the flow of goods between these facilities.

$$C(\boldsymbol{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ij} d_{kl}\, x_{ik} x_{jl} \quad (5)$$

x_{ik} : Binary variables, 1 if facility i is assigned to location k, 0 otherwise

f_{ij} : A quantity representing the flow of goods between facilities i and j

d_{kl} : Distance between locations k and l

This problem is subject to the constraints that only a single facility should be assigned to each location, and only a single location should be assigned to each facility, which are expressed in Eqs. (6) and (7) respectively.

$$\sum_{k=1}^{n} x_{ik} = 1 \quad (i = 1, 2, \cdots, n) \quad (6)$$

$$\sum_{i=1}^{n} x_{ik} = 1 \quad (k = 1, 2, \cdots, n) \quad (7)$$

The penalty term that satisfies the constraints of Eqs. (6) and (7) can therefore be expressed as follows.

$$P(\boldsymbol{x}) = \sum_{i=1}^{n}(\sum_{k=1}^{n} x_{ik} - 1)^2 + \sum_{k=1}^{n}(\sum_{i=1}^{n} x_{ik} - 1)^2 \quad (8)$$

Equation (8) has value 0 if all the constraints are satisfied, has a strictly positive value if any of them are violated, and increases with the number of violations, thereby it works as an appropriate penalty term. The energy function of the facility location problem is obtained by substituting Eqs. (5) and (8) into Eq. (1).

$$\begin{aligned} E(\boldsymbol{x}) &= C(\boldsymbol{x}) + \alpha P(\boldsymbol{x}) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ij} d_{kl}\, x_{ik} x_{jl} + \\ &\quad \alpha[\sum_{i=1}^{n}(\sum_{k=1}^{n} x_{ik} - 1)^2 + \sum_{k=1}^{n}(\sum_{i=1}^{n} x_{ik} - 1)^2] \end{aligned} \quad (9)$$

The two-dimensional array of binary variables in this problem

$$\boldsymbol{x} = (x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, x_{22}, \ldots\ldots, x_{nn})$$

can be expanded into a one-dimensional array with n^2 elements,

$$\boldsymbol{x} = (x_1, x_2, x_3, \ldots\ldots, x_{n^2-1}, x_{n^2})$$

and when redefined in this way, Eq. (9) can be transformed by reorganizing the coefficients as follows:

$$E(\boldsymbol{x}) = \sum_{i<j} J_{ij} x_i x_j + \sum_i h_i x_i + c \quad (10)$$

allowing it to be treated as a QUBO problem. However, the user needs to adjust α by iteratively changing its value when solving the QUBO, analyze the obtained solution by checking if it satisfies the constraints and its cost value is low enough, and repeat the process until the application requirements are satisfied. In general, it is difficult and cumbersome to obtain those solutions because of this trial-and-error approach.

In the BQP IF of the third generation DA, the cost term in Eq. (5) and the penalty term in Eq. (8) are converted into the binary quadratic form of Eqs. (3) and (4), respectively, and can be set separately so that the automatic penalty coefficient adjustment part in the SIL analyzes the influence of the penalty term in the search process and adjusts α appropriately, so that the solution converges to optimal solutions.

Though the above QAP has two types of constraints (Eqs. (6) and (7)), practical problems often have more constraints. Since the third generation DA only handles one penalty term separately from the cost term, the user has to adjust the individual weights for constraints to be incorporated in $P(\boldsymbol{x})$ if there are multiple constraints. Equation (8) is an example where the weights of the constraints in Eqs. (6) and (7) are all the same.

(2) One-hot constraints

Optimization problems that deal with binary variables often include equality constraints where a group of variables has a sum value of 1, such as Eqs. (6) and (7) in the QAP. This type of constraint is called a one-hot constraint. The third generation DA incorporates a mechanism that makes it possible to solve problems faster by explicitly specifying groups of variables subject to one-hot constraints.

As a specific example of a one-hot constrained problem, we will discuss a facility location problem with four facilities and four locations. In this case, the number of variables is 16, and the variables can be arranged in a two-dimensional matrix as shown in Fig. 2 reflecting the relationships between the facilities and locations. The variable groups indicated by rounded-rectangles in Fig. 2 (a) and (b) become the one-hot constraint groups corresponding to Eqs. (6) and (7), respectively. Since the variable groups in Fig. 2 (a) and (b) point in different directions respectively, this is called a 2way1hot constraint. If we redefine the two-dimensional matrix x_{ij} of Eq. (8) into a one-dimensional array $(x_1, x_2, \ldots, x_{16})$, the penalty term turns into Eq. (11).

$$P(\boldsymbol{x}) = \sum_{k=0}^{3}\left(\sum_{i=4k+1}^{4k+4} x_i - 1\right)^2 + \sum_{k=0}^{3}\left(\sum_{i=0}^{3} x_{k+4i+1} - 1\right)^2 \quad (11)$$

When this is rearranged into the form of Eq. (4), the coupling coefficient matrix of the penalty term is the matrix $\boldsymbol{J}$ shown in Fig.3, the bias coefficients are all -2, and the constant term is 8. The user has to set these values as a penalty term. Furthermore, successive variable array $(x_1, x_2, \ldots, x_{16})$ are specified as a group subject to 2way1hot constraints. In this way, the search core's constraint utilization search can flip the state of variables, avoiding the violation of the constraint specified as a 2way1hot group, thereby obtaining a good solution at high speed. Section 3 shows the results of evaluating the performance of the third generation DA in solving QAPs.

In the facility location problem as shown in Fig. 2, only one 2way1hot group is specified, but it is possible to specify more than one 2way1hot groups with different numbers of variables and the value of the number should be square. And the index of variable should be consequtive within group and have no gap between groups.

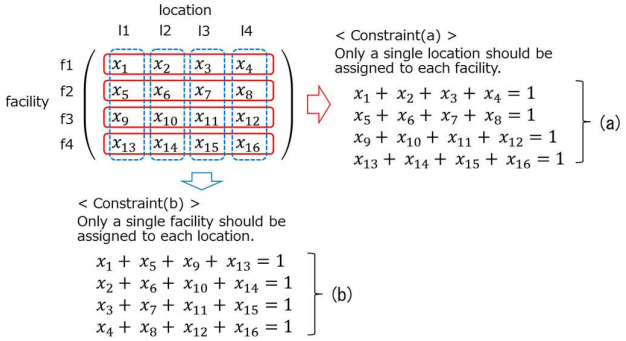

Fig. 2: 2way1hot constraint in a facility location problem

$$\boldsymbol{J} = \begin{pmatrix} \boldsymbol{J_1} & \boldsymbol{J_2} & \boldsymbol{J_2} & \boldsymbol{J_2} \\ \boldsymbol{O} & \boldsymbol{J_1} & \boldsymbol{J_2} & \boldsymbol{J_2} \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{J_1} & \boldsymbol{J_2} \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{J_1} \end{pmatrix}, \quad \boldsymbol{J_1} = \begin{pmatrix} 0 & 2 & 2 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{J_2} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \quad \boldsymbol{O} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 3: Coupling coefficient matrix of the penalty term in a facility location problem (Fig. 2)

Other than specifying 2way1hot constraint group, it is also possible to specify another type of one-hot group as a constraint where the sum of a one-dimensional series of variables is 1. This is called a 1way1hot constraint [(*3)]. As an example of 1way1hot constrained problem, we will discuss a Graph Coloring Problem (GCP). In one type of GCP called a vertex coloring problem, a set of n vertices defines a graph, and one color out of m colors is assigned to each vertex so that no edge in the graph connects two vertices of same color, which can be applied to practical problem such as the channel assignment in wireless communication systems.

In GCP, if variables are defined such that x_{ij} = 1 when color c_j is assigned to v_i (and x_{ij} = 0 when it is not), then the constraint can be expressed as follows:

$$\sum_{j=1}^{m} x_{ij} = 1 \quad (i = 1, 2, \cdots, n) \tag{12}$$

This means that only one of m colors can be assigned to each vertex. Fig. 4 shows states corresponding to constraint satisfaction (Fig. 4 (a)) and constraint violation (Fig. 4 (b)) for the example of a graph coloring problem with m = 3 and n = 5. In the second equation of one-hot constraint in Fig. 4 (b), right-hand-side value of the equation is 0, which means no color is assigned to the vertex 2.

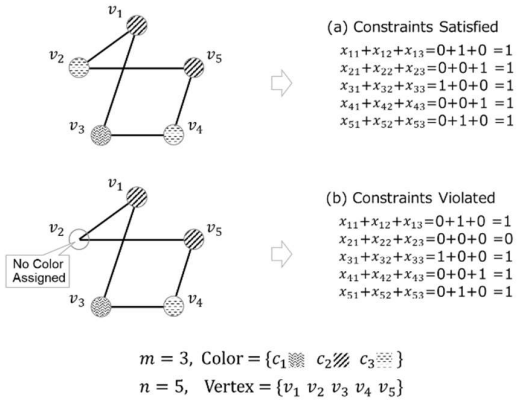

Fig. 4: Example of 1way1hot constraints in a graph coloring problem

Another constraint of GCP which prohibits the same color from being assigned to a pair of vertices of edges can be expressed as follows:

$$\sum_{(i,j)\in E} \sum_{k=1}^{m} x_{ik}\, x_{jk} \tag{13}$$

In Eq. (13), (i, j) indicates an edge that connects vertices v_i and v_j, and E is a set of edges in the graph. Eq. (13) equals to 0 when no edge in the graph connects the vertices of same color, and increases with the number of violated edges

If the two-dimensional variable array in Fig. 4 is redefined into a one-dimensional array consisting of 15 variables ($x_1, x_2, \ldots, x_{15}$), and the weights of each one-hot constraint are uniform, the penalty term related to one-hot constraint becomes

$$\sum_{k=0}^{4} \left(\sum_{i=3k+1}^{3k+3} x_i - 1 \right)^2 \tag{14}$$

To make up one penalty term of the BQP IF from all two types of constraints in GCP corresponding to Eqs. (12) and (13) respectively, the user also transforms Eq. (13) into a binary quadratic formula using the variables $(x_1, x_2, \ldots, x_{15})$ above, applies a proper weight to it and adds its result to Eq. (14). Finally the user redefines this into Eq. (4) and has to set it as the penalty term. Furthermore, five consecutive variable groups $(x_1, x_2, x_3), (x_4, x_5, x_6)$, $\cdots$, (x_{13}, x_{14}, x_{15}) are specified as 1way1hot constraint groups. By this specification, the constraint utilization search flips the state of variables, avoiding the violation of the constraints specified as 1way1hot groups.

In the GCP as shown in Fig. 4, all five 1way1hot groups have same number of variables. but it is possible to specify different number per each group. And the index

(*3) Although a 1way1hot constraint does not have similar geometric orientation as a 2way1hot constraint, it is referred to as "1way" in the third generation DA specification in order to clearly distinguish it from a "2way" constraint.

of variable should be consequtive within group and have no gap between groups.

(3) Linear inequality constraints

A linear inequality constraint is a constraint that frequently appears in practical combinatorial optimization problems. In the BQP IF of the third generation DA, multiple linear inequality constraints can be set directly without being converted to a quadratic form of equal constraints. For linear inequality constraints defined by Eq.(15) with a set of variables $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$, user can set the coefficients a_{ji} and right-hand-side values b_j directly.

$$\sum_{i=1}^{n} a_{ji} x_i \leq b_j \qquad (j = 1, 2, \cdots, k) \tag{15}$$

To demonstrate easier handling of problems with linear inequality constraints on the third generation DA, we explain the differences between the QUBO formulation and the formulation for BQP IF for a knapsack problem with one knapsack and n items as an example. Let a_i and c_i denote the weight and the value of each item i respectively, and b the weight capacity of the knapsack. Let us define variable x_i for each item i so that it has the value 1 when the item i is in the knapsack, and 0 otherwise. A knapsack problem, in which the sum of the values of the items in the knapsack is maximized while satisfying the constraint on the capacity such that the sum of the weights of items does not exceed the knapsack capacity, is formulated as the minimization of the following energy function.

$$\begin{aligned} E(\boldsymbol{x}, \boldsymbol{y}) = & -\sum_{i=1}^{n} c_i x_i + \\ & \alpha\Big[\Big(b - \sum_{m=0}^{a_{max}-1} m y_m\Big) - \sum_{i=1}^{n} a_i x_i\Big]^2 + \\ & \beta\Big(\sum_{m=0}^{a_{max}-1} y_m - 1\Big)^2 \end{aligned} \tag{16}$$

In Eq.(16), a_{max} and y_m are the maximum weight over all items and the binary auxiliary variable, respectively. Considering the room of the knapsack and the maximum weight over all items, you can formulate the knapsack problem using the value a_{max}.

The first term is the cost term that represents the value to be maximized. The second and the third terms are linear inequality constraints converted to equality penalty terms regarding the sum of items, which implies:

$$\sum_{i=1}^{n} a_i x_i = (b - m) \leq b \tag{17}$$

By expanding Eq. (16), we now get the energy function to be minimized in a form of Eq. (2), which means we can handle a problem with linear inequality constraint as a QUBO problem. However, as mentioned above, the QUBO formulation of inequality constraints is complex, and the introduction of auxiliary variables has the disadvantage of increasing the scale of the problem. Furthermore, since practical problems often have multiple inequality constraints, the user needs to repeat the conversion discussed above for the number of inequality constraints. Moreover, the user needs to determine a proper weight for each constraints (a penalty coefficient for each penalty term converted from a constraint) when putting them together into the energy function, making it even harder to obtain a solution.

To resolve this problem, the third generation DA has an interface for the user to directly specify linear inequality constraints in the form of Eq. (15), thereby eliminating the process of incorporating the inequality constraints to a QUBO problem. This makes things more convenient for the user, and greatly enhances the problem-solving performance. In the case of the knapsack problem described above, only the first term of Eq. (16)

$$E(\boldsymbol{x}) = -\sum_{i=1}^{n} c_i x_i \tag{18}$$

is set as a cost term, and there is no need to set a penalty term. When there are multiple inequality constraints, for example, a volume for each item and a volume capacity of the knapsack are given in addition to the problem discussed above, the user can input the inequality

constraints for the weight capacity and the volume capacity separately. By directly setting linear inequality constraints, the third generation DA can perform optimization with the constraint utilization search evaluating the penalty of inequality constraints. The weighting between inequality constraints is automatically adjusted in the SIL. In Section 3, we present the results of evaluating the performance of solving problems with inequality constraints.

This Section 2 described the features of the third generation DA, its interfaces and functions related to constraints. Table 1 shows a comparison of the second and third generation DAs.

3. Performance evaluation

In this section, we compare the search performance of the second and third generation DAs with regard to one-hot and inequality constraints, which are new features of the third generation DA. We also show the performance in solving large-scale problems that are not handled by the second generation DA.

(1) One-hot constraints

We evaluated the performance in problems with one-hot constraints by using the QAP described in Section 2.2(1). As a benchmark problem, we used Lipa70a (a facility location problem with 70 facilities and 70 locations) from the QAPLIB dataset [17], which is referenced in numerous papers. The scale of this problem is 4,900 (=70×70) bits.

The second generation DA solves this problem as a QUBO that integrates the cost term and the penalty term with a penalty coefficient α manually adjusted in advance. On the other hand, the third generation DA uses a separated cost and penalty terms interface. The cost term consists of the products of distances and flows between facilities as represented by Eq. (5), and is set separately from the penalty term related to the constraints for the facility allocations and the locations shown in Eq. (8). Furthermore, all variables are specified as a 2way1hot group. By setting input such way, we can expect that the automatically adjusted penalty coefficient α and the constraint utilization search work to contribute to solve the problem quickly. We conducted 10-time trials of optimization with a different random seed, a parameter that affects stochastic transitions in the MCMC method, for each DA, and compared the each transition of the value of the energy function against the execution time.

The evaluation results are shown in Fig. 5. The horizontal axis shows the execution time, and the vertical axis shows the solution energy. The median of the 10 trials is shown as a solid line, and the region between the maximum and minimum energies of the 10 trials is shown as a shaded area to indicate the variation of the energy transition between trials. Even after 300 seconds of search, the second generation DA was not able to obtain solutions with an energy lower than 170,374. On the other hand, the

Table 1: Comparison of second and third generation DAs

	Problem Scale	Full-Coupling Coefficient Precision	Specific Function for Constraints	Search Engine Implementation
The Second Generation	up to 4,096 bits	64-bit signed integer	None	Dedicated Processor
	up to 8,192 bits	16-bit signed integer		
The Third Generation	up to 100,000 bits	64-bit signed integer	・ Separated Cost and Penalty Terms ・ 2way/1way 1hot Constraints ・ Linear Inequality Constrains	Software and Dedicated Processor Hybrid Architecture

third generation DA obtained the optimal solution with the energy of 169,755 within 11 seconds at most. We confirmed the significant performance improvement by the separated cost and penalty terms and the 2way1hot constraint group specification.

(2) Linear inequality constraints

The performance of the third generation DA for problems with inequality constraints was demonstrated by applying it to the quadratic knapsack problem (QKP) – a variant of the knapsack problem described in Section 2.2(3) that uses quadratic terms for value calculations. It is defined as follows:

$$E(\boldsymbol{x},\boldsymbol{y}) = -\sum_{i=1}^{n} c_i x_i - \sum_{i<j}^{n} c_{ji} x_i x_j + \alpha\left[\left(b - \sum_{m=0}^{a_{max}-1} m y_m\right) - \sum_{i=1}^{n} a_i x_i\right]^2 + \beta\left(\sum_{m=0}^{a_{max}-1} y_m - 1\right)^2 \quad (19)$$

The definition of the variables here is the same as in Eq. (16). Compared with the knapsack problem shown in Eq.(16), another value is added to the cost term. This additional value accrues when variables x_i and x_j are both 1, i.e., when two items enter the knapsack at the same time. QKP is a problem that the value is determined based on the relationship between two items selected from the candidates under the constraints not to exceed the weight capacity of the knapsack. An example of possible QKP applications is the selection of the construction locations for wireless base stations or airports under the constraint of a limited budget. Here, we used a benchmark problem with a size of n= 300 items generated by the method shown in Billionnet et al. [18].

In the second generation DA, we added 50 bits of auxiliary variables in the formulation of Eq. (19), and solved the problem as a QUBO with 350 variables. On the other hand, for the third generation DA we formulated a binary quadratic problem with 300 variables and the cost term set up as follows:

$$C(\boldsymbol{x}) = -\sum_{i}^{n} c_i x_i - \sum_{i<j}^{n} c_{ij} x_i x_j \quad (20)$$

and we directly set the coefficients of the linear inequality (weight for each item) and the comparison value (weight capacity of the knapsack) separately from the cost terms. As before in Section 3(1), we performed 10 optimization trials with different random seed values for each DA.The evaluation results are shown in Fig. 6. The notation of the graph is the same as in Fig. 5. The second generation DA could only find a score of about -960,000 after 300 seconds, while the third generation DA found the best known solution of -996,070 in one second. We therefore confirmed that there was a significant performance improvement in solving problems with inequality constraints.

(3) Large-scale problems

As shown in Section 2, the third generation DA can efficiently handle problems with a scale of 100,000 bits

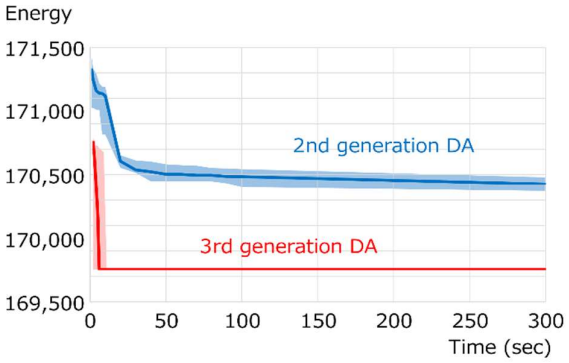

Fig. 5: Energy transition of the second and third generation DAs in a QAP (Lipa70a)

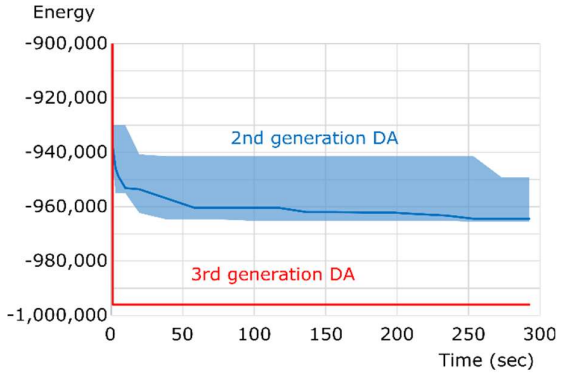

Fig. 6: Energy transition when solving QKP with the second and third generation DAs

and find good solutions by using a hybrid configuration of software and hardware. To evaluate the performance with large-scale problems, we applied the third generation DA to five large-scale problems from the QAPLIB: sko100e, esc128, tho150, tai150b, and tai256c. For each problem, the execution time was limited to 300 seconds. As in Section 3(1), the cost term and the penalty term of 2way1hot constraint were set separately, 2way1hot group is specified, and 10 trials with different random seeds were conducted for each problem.

For each problem, Table 2 shows the number of facilities/locations n, the best known solution, the number of times the third generation DA reached the best known solution in 10 trials (#BKS), the percentage gap between the average minimum energy reached in 10 trials and the best known solution, and the average time taken to reach the solution. For problems with n=100 and n=128, the best known solutions were reached in 108.6 and 8.0 seconds on average respectively, and for problems with a larger number of n= 150 and n= 256, an energy level with a very small gap was reached after 300 seconds.

4. Conclusion

We have shown how the DA, which can perform fast MCMC-based search through the practical means of digital circuits, has evolved into a new generation by combining software that controls effective transitions in the large-scale solution space and search techniques that actively utilize constraints. This third generation DA has been available as a cloud service since February 2021.

Table 2: Performance of third generation DA on large-scale QAP

Instance name	n	Best known	#BKS	Gap (%)	Time (sec)
sko100e	100	149,150	10	0	108.6
esc128	128	64	10	0	8.0
tho150	150	8,133,398	0	0.144828	300.0
tai150b	150	498,896,643	0	0.467121	300.0
tai256c	256	44,759,294	0	0.212259	300.0

Fujitsu will continue to pursue research and development aimed at further advancing the DA technology to tackle a variety of social issues and support the ongoing digital transformation of its customers.

Acknowledgements

The authors thank Makiko Konoshima, Kouichi Kanda, Norihiro Kakuko, and Shinichi Sazawa for their efforts to develop key technologies, Nobuyuki Hara for his contribution to performance evaluation, and Hisanori Fujisawa, Hidetoshi Matsumura, and Parizy Matthieu for their helpful comments on the manuscript. Finally the authors thank all members related with the development of the third generation Digital Annealer.

References

[1] Fujitsu : Fujitsu Laboratories Develops New Architecture that Rivals Quantum Computers in Utility, https://www.fujitsu.com/global/about/resources/news/press-releases/2016/1020-02.html.

[2] Fujitsu : Fujitsu Quantum-Inspired Digital Annealer Cloud Service to Rapidly Resolve Combinatorial Optimization Problems, https://www.fujitsu.com/global/about/resources/news/press-releases/2018/0515-01.html.

[3] S.Matsubara, M.Takatsu, T.Miyazawa, T.Shibasaki, Y.Watanabe, K.Takemoto, H. Tamura : Digital Annealer for High-Speed Solving of Combinatorial Optimization Problems and Its Applications, ASP-DAC 2020 - 25th Asia and South Pacific Design Automation Conference (ASP-DAC).

[4] K.Takemoto, S.Matsubara, Y.Watanabe, D.Shimada, T.Kurita, H. Tamura : "Digital Annealer" Technology for Efficiently Solving Combinatorial Optimization Problems and Its Applications in Manufacturing and Materials Science, C-Abstracts of IEICE TRANSACTIONS on Electronics Vol.J104-C No.4, pp.101-109, ISSN: 1881-0217 (in Japanese).

[5] Fujitsu : Fujitsu Launches Next Generation Quantum-Inspired Digital Anneale Service, https://www.fujitsu.com/global/about/resources/news/press-releases/2018/1221-01.html.

[6] Fujitsu : How Digital Annealer is pushing the boundaries of financial services, https://www.fujitsu.com/downloads/GLOBAL/vision/2020/download-center/melcoinvestments_EN.pdf.

[7] Fujitsu : Fujitsu and Toyota Systems Optimize Large-Scale Supply Chain Logistics using Quantum-Inspired Technology, https://www.fujitsu.com/global/about/resources/news/press-releases/2020/0910-02.html.

[8] Fujitsu : Fujitsu and PeptiDream Achieve Milestone in Joint Research for Peptide Drug Candidates with High-Speed, High-Precision Exploration Technology,

https://www.fujitsu.com/global/about/resources/news/press-releases/2020/1013-01.html.

[9] Z.Naghsh, M.Javad-Kalbasi, S.Valaee : Digitally Annealed Solution for the Maximum Clique Problem with Critical Application in Cellular V2X, ICC 2019 – 2019 IEEE International Conference on Communications (ICC).

[10] M.Javad-Kalbasi, K.Dabiri, S.Valaee, A.Sheikholeslami : Digitally Annealed Solution for the Vertex Cover Problem with Application in Cyber Security, ICASSP 2019 – 2019 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

[11] H.Jippo, M.Ohfuchi, C.Terashima : Fundamental and Application to Material Development of DA, Vacuum and surface scienceI, 63(3), pp.117-122, 2020, ISSN : 2433-5835 (in Japanese).

[12] T.Saito, S.Katayama, A.Yoshida, T.Kashikawa, K.Kimura, Y.Amano, Y.Hayashi : Fast hierarchical coordination using price signal for town-scale home-EMSs aggregation with digital annealer, ECOS 2020 - 33rd International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems(ECOS).

[13] F.Mo, H.Jiao, S.Morisawa, M.Nakamura, K.Kimura, H.Fujisawa, M.Ohtsuka, H.Yamana : Real-Time Periodic Advertisement Recommendation Optimization using Ising Machine, IEEE BigData 2020 – 2020 IEEE International Conference on Big Data(BigData).

[14] Fujitsu : Fujitsu and University of Toronto Researchers Develop Quantum-Inspired Technology to Optimize Radiation Treatment Plans for Brain Tumors and Other Diseases, https://www.fujitsu.com/global/about/resources/news/press-releases/2021/0226-01.html.

[15] F.Glover, G.Kochenberger, Y.Du : A Tutorial on Formulating and Using QUBO Models, arXiv:1811.11538v6 [cs.DS], 2019.

[16] K.Hukushima, K.Nemoto: Exchange Monte Carlo method and application to spin glass simulations, Journal of the Physical Society of Japan, vol.65, pp. 1604-1608, 1996.

[17] R.E.Burkard, S.E.Karisch, F.Rendl : QAPLIB – A Quadratic Assignment Problem Library. Journal of Global Optimization vol.10, pp.391–403, 1997, https://doi.org/10.1023/A:1008293323270.

[18] A.Billionnet, F.Calmels : Linear programming for the 0–1 quadratic knapsack problem, European Journal of Operational Research, vol.92, no.2, pp.310–325, 1996.