IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 December 2023, accepted 13 January 2024, date of publication 22 January 2024, date of current version 31 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3357144

RESEARCH ARTICLE

Nebula: Network Enhanced Boltzmann Machine With Universal Local Search Architecture

YASUHIRO WATANABE[1], **HIROTAKA TAMURA**[2], **(Fellow, IEEE), YUKI FURUE**[1], **AND FANG YIN**[1]
[1]Quantum Laboratory, Fujitsu Ltd., Nakahara-ku, Kawasaki, Kanagawa 211-8588, Japan
[2]DXR Laboratory Inc., Kohoku-ku, Yokohama, Kanagawa 223-0066, Japan
Corresponding author: Yasuhiro Watanabe (yas.watanabe@fujitsu.com)

ABSTRACT We propose *Nebula*, a computational method for combinatorial optimization and stochastic sampling. The proposed method is designed for a generic local search engine utilized for digital annealing in an optimization system for multidimensional binary variables, and it can handle a variety of cost functions of multidimensional binary variables with multi-body interactions in a unified manner. In addition to optimization, the method can be used to perform sampling to reproduce the Boltzmann distribution. To achieve this, we extend the network of conventional Boltzmann machines or Ising machines to include dependent variables mediating various interactions. The extended network enables fast operation by predicting the total energy change due to state transitions in all one-flip neighborhoods. The energy function, which is limited to the quadratic form in conventional Ising machines, is extended here to represent inequality constraints and higher-order products. The ability to handle higher-order spin products enables the implementation of arbitrary energy functions with k-body interactions based on the Walsh transform. We conducted numerical experiments to demonstrate the concept of our proposed method and show that it can solve problems with inequality constraints and higher order terms, which are difficult to solve with conventional Ising machines. The numerical experiments also show that our method can exploit the expansion of the energy landscape by the Walsh function and reproduce the Boltzmann distribution by sampling.

INDEX TERMS Boltzmann machine, combinatorial optimization, higher-order interaction, HOBO, Ising machine, MCMC, parallel tempering, quantum-inspired computing, QUBO, Walsh transform.

I. INTRODUCTION

The evolution of modern information processing technology has been driven by processor performance improvements through the miniaturization of CMOS devices. At the same time, the search for computing paradigms other than processors has been underway since before the advent of integrated circuit processors [1], [2]. Now that processor performance improvements through CMOS device miniaturization have reached their limit [3], research into various problem-specific computational methods is becoming increasingly important [4].

Representative methods of such alternative paradigms are those that utilize artificial spins inspired by physical phenomena [5] or artificial neurons inspired by brain functions [6], followed by different circuit implementations [7], [8]. Among these proposals, Boltzmann machines have an important position in the integration of the physical phenomena and inference and learning mechanisms of the brain [6]. The Boltzmann machine is a neural network of recursively coupled binary neurons capable of both learning and inference operations. The learning capability of Boltzmann machines was improved by the Restricted Boltzmann machine (RBM) [9], [10], which has connections only between the visible and hidden layers. Historically, making the RBM multi-layered opened the door for the emergence of deep learning.

A similar line of research is that of artificial-spin computing, which has been brought back into prominence by D-Wave's quantum annealing machines [11], [12]. Breaking away from technologies that utilize real physical spin, artificial spins have evolved in a direction that does not

The associate editor coordinating the review of this manuscript and approving it for publication was Shih-Wei Lin.

necessarily reproduce the analog or quantum behavior of real spin due to simplification and idealization based on digital technology. This direction of technology is known as quantum-inspired computing.

The goal of quantum-inspired computing is touted as instantly optimizing the cost function of a wide range of problems. This recognition is what many people outside of quantum computing and quantum annealing expect from quantum technology. While the desire for high-performance computing is legitimate, there are still many obstacles to achieving this goal, whether quantum technology or anything else. Therefore, our vision for quantum-inspired computing is to get as close as possible to this expectation while simultaneously achieving the goal of high speed (i.e., faster than conventional computing schemes) to obtain solutions with practical accuracy and the flexibility to handle diverse problems.

Examples of such "quantum-inspired" computing include various classical (i.e., non-quantum) circuits designed to optimize the energy functions given by the quadratic functions of binary variables [5], [6], [7], [8]. To achieve the desired speedups, various parallel processing schemes have been proposed, such as Hitachi's CMOS Annealer [13], [14], Toshiba's Simulated Bifurcation Machine (SBM) [15], [16], [17], and Fujitsu's Digital Annealer [18], [19].

To cope with the variety of problems to be handled, we need to enhance the problem-solving performance so that it can address higher-order binary optimization (HOBO) problems [20], [21], which are not limited to quadratic unconstrained binary optimization (QUBO). Our observation is that while regular QUBO has high expressive power for formulating a wide range of combinatorial optimization problems [22], it has difficulties when it comes to problems involving inequality constraints and higher-order products: specifically, the size of the problem tends to increase when expressed in ancillary bits [23], and they are often intractable when formulated in slack variables [24]. As a possible extension of the Digital Annealer, the extended Ising machine has been proposed to add higher-order terms and rectified linear unit (ReLU)-type inequality constraint functions to the binary quadratic form [25].

Expanding on this basic idea, we propose a network-enhanced Boltzmann (or Ising) machine with a universally applicable local search method, coined *Nebula*, which features an architecture with k-body interactions implemented using an additional network of dependent variables. The value of k and the number of interactions m are limited only by the processing power and memory capacity of the computing platform. Within this limit, k is as large as the dimension of the problem, n, and m can be beyond n.

Our proposal utilizes an algorithm that takes parallel processing into account to improve the performance when dedicated circuits are used. However, our goal is to achieve a sufficient performance using not only dedicated hardware but also the latest processors with excellent parallel processing capabilities, thus enabling early deployment on various platforms according to market demand.

We should briefly mention a note on terminology here. Our goal is to extend the energy functions of conventional Ising and Boltzmann machines in the binary quadratic form to more general functions. Since Boltzmann machines are initially named for networks that include a learning mechanism, and we do not discuss learning in this paper, we refer to them as Ising machines, regardless of whether the variables are spin or 0/1 binary.

Section II of this paper presents the dependent variable-based extension technique that forms the basis of our proposal and explains how to represent energy of an arbitrary order. In Section III, we describe *Nebula*'s expressiveness for different energy landscapes, and in Section IV, we present numerical experiments as a proof of concept. We conclude in Section V with a brief summary and mention of future work.

II. ARCHITECTURE FOR EXTENDED FUNCTIONALITY

A. ENERGY FORMULATION

Conventional Ising machines with quadratic energy are represented as networks with bidirectional two-body coupling $W_{ij} = W_{ji}$ (Fig. 1(a)). Adding dependent-variable networks mediating various many-body couplings in addition to two-body couplings yields an extended Ising machine architecture, *Nebula*, with improves the problem representation power (Fig. 1(b)). Here, both the conventional Ising machine and *Nebula*'s network assume a fully connected network of arbitrary topology within the limits imposed by memory capacity and the number of processing elements, thereby enabling various problems to be mapped.

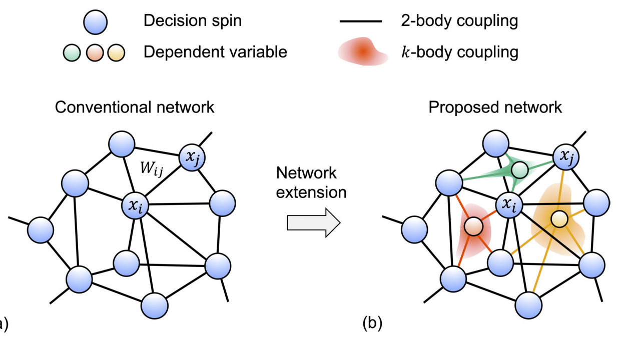

FIGURE 1. Network extension by additional couplings. (a) Conventional Ising machines use two-body coupling to represent the binary quadratic energy of the decision spins. (b) In contrast, the proposed architecture implements k-body coupling by adding coupling through dependent variables. k-body coupling enables inequality constraints and higher-order products to be added, thereby improving the expressive power of energy functions.

Let the energy $H(\boldsymbol{x})$ of the system be a function of n-dimensional 0/1 binary decision variable $\mathbf{x} = (x_1, \cdots, x_n) \in \{0, 1\}^n$ that represents the value of artificial spin in Fig. 1. $H(\boldsymbol{x})$ is composed of $E(\boldsymbol{x})$, the binary quadratic form of $\boldsymbol{x}$, and additional energy functions $G(\boldsymbol{x})$:

$$H(\boldsymbol{x}) = E(\boldsymbol{x}) + G(\boldsymbol{x}), \tag{1}$$

$$E(\boldsymbol{x}) = -\sum_{i=1}^{n} b_i x_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} x_i x_j, \tag{2}$$

$$G(\boldsymbol{x}) = \sum_{k=1}^{m} \lambda_k G_k(\boldsymbol{x}), \tag{3}$$

where b_i and W_{ij} are the linear and second-order coefficients of the quadratic energy of the system. G is the summation of m additional functions G_k multiplied by a factor λ_k, which is a coefficient to balance the linear and quadratic energy E with the additional function G_k and is positive for constraints such as inequality constraints. We assume that $W_{ij} = W_{ji}$, $W_{ii} = 0$, as in conventional Ising machines.

When using a formulation with the spin variable $\sigma_i \in \{-1, +1\}$ instead of $x_i \in \{0, 1\}$, we convert the coefficients of the quadratic energy b_i and W_{ij} accordingly and use the same basic algorithm in both representations. In contrast, for cases where higher-order products of spin variables are needed, we use a function G_k that represents the spin product, as described in Section II-C.

The method we propose is aimed at energy-based models, where energy (1)–(3) is defined as a univalent function of multidimensional binary variables. Note that $\pi(\boldsymbol{x})$, the occupancy probability of the state $\boldsymbol{x}$ of the system, is given by the Boltzmann distribution:

$$\pi\left(\mathbf{x}\right) = \exp\left(-\beta H\left(\mathbf{x}\right)\right)/Z, \tag{4}$$

where $Z = \sum_{\mathbf{x}} \exp\left(-\beta H\left(\mathbf{x}\right)\right)$ is the normalization constant.

Since the energy function is univalent, its value is uniquely determined independent of the path by starting from the initial-state energy and integrating the energy change at each motion step. This simple fact guarantees that we can construct a network of artificial spins that produce energy differences for reproducing the total given energy. If we can efficiently calculate the energy difference for varying variables, we can apply diverse local search methods based on the energy difference.

To allow as much parallel processing as possible, we restrict the form of G_k to be a univalent function of the weighted linear sum r_k of the variables x_j rather than an arbitrary function of $\boldsymbol{x}$. Even with this restriction, G_k can in principle be any computable function and can handle a variety of functions useful for optimization, thus providing high problem representation power thanks to handling higher-order interactions. Hence, we have

$$G_k(\boldsymbol{x}) = G_k(r_k), \tag{5}$$

$$r_k = \sum_{j=1}^{n} Z_{kj} x_j + c_k, \tag{6}$$

where r_k is regarded as a resource variable that contributes to the total energy through the function G_k, or the kth dependent variable. G_k can be utilized as a penalty function that generates a positive penalty if r_k violates a certain condition. The coefficient Z_{kj} is used to represent the contribution of variable x_j to resource r_k, while Z_{ik} is used for the coefficient representing the influence from $G_k(r_k)$ to variable x_i. Coefficient c_k is used, for example, to bias r_k to give an inequality constraint threshold.

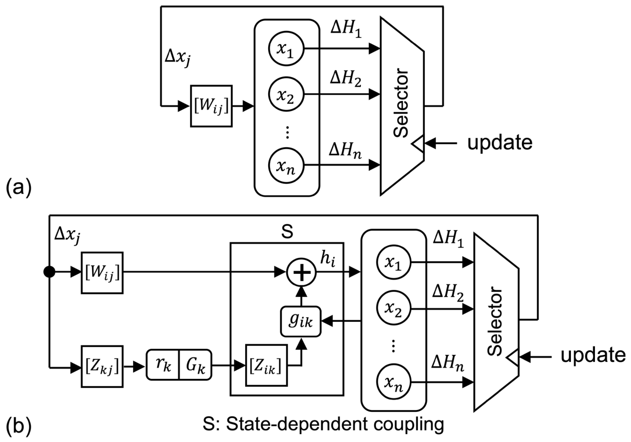

FIGURE 2. Comparison of proposed method and conventional Ising machine with parallel calculation of the energy difference. (a) Conventional method: A single-layer network predicts the change in energy *H* when the decision variable x_i is reversed. (b) Proposed method: An additional one-layer network of dependent variable G_k is connected to the decision spin through a state-dependent coupling S, which calculates g_{ik}, or the contribution of G_k to the local field h_i.

B. NETWORK FOR ENERGY DIFFERENCE CALCULATION

We perform a local search guided by the energy differences, not by the energy values themselves. Calculations based on energy differences are the traditional method used in Markov chain Monte Carlo (MCMC) for binary quadratic forms [26], [27]. The energy calculation using (1) through (3) requires $O(n^2) + O(nm)$ operations per search iteration, while the difference calculation requires much less.

Therefore, we derive from (1)–(3) a network that operates on the basis of the energy difference with artificial spins in a parallelizable and easy to implement form with modern digital circuit technology. The main objective of this network is to calculate and output the energy difference ΔH_i for all i assuming that only one decision variable x_i is reversed. In other words, this network predicts the energy of all n possible futures generated by a one-flip variable inversion. It has been applied in [18] for conventional Ising machines (Fig. 2(a)) and in [25] for Ising machines with extended energy functions (Fig. 2(b)).

We calculate the incremental energy ΔH_i when the variable x_i changes as $x_i \to x_i + \Delta x_i$, where $\Delta x_i = 1 - 2x_i$. ΔH_i is expressed as the sum of the changes in the quadratic energy E and the additional energy G (denoted as ΔE_i and ΔG_i, respectively).

$$\Delta H_i = \Delta E_i + \Delta G_i \tag{7}$$

From (1) through (7), we obtain the value of ΔH_i as follows.

$$\Delta H_i = -h_i \Delta x_i \tag{8}$$

$$h_i = \sum_{j=1}^{n} W_{ij} x_j + b_i - \sum_{k=1}^{m} \lambda_k g_{ik} \tag{9}$$

$$g_{ik} = \frac{\left[G_k\left(r_k + Z_{ki}\Delta x_i\right) - G_k\left(r_k\right)\right]}{\Delta x_i} \tag{10}$$

The energy change ΔH_i is proportional to Δx_i with the proportionality constant $-h_i$. h_i represents the sum of forces acting on the artificial spin x_i from the other interacting spins and the bias term b_i, and is called the local field [25]. g_{ik} represents the contribution of the dependent variable G_k to the local field h_i and can be interpreted as the force exerted by G_k on the spin x_i, i.e., the equivalent signal strength.

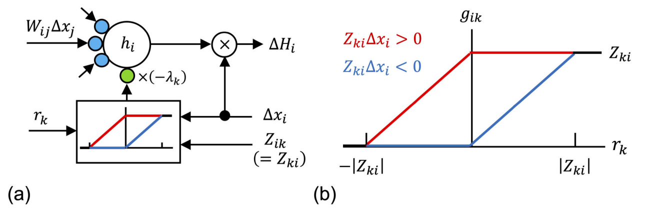

FIGURE 3. State-dependent coupling. (a) The effect of the kth constraint on the energy increment depends on the decision variable x_i. (b) g_{ik} as a contribution of the kth constraint to the local field h_i is shown in the graph of g_{ik} versus r_k.

As r_k varies by a finite value of $Z_{ki}\Delta x_i$, g_{ik} varies piecewise linearly with respect to r_k. Note that the slope of the G_k-r_k characteristic varies stepwise, as in a rectified-linear type penalty function with inequality and equality constraints. Also, g_{ik} has different values depending on the value of x_i, even for the same input value r_k. Therefore, the effect of G_k on spin x_i depends on the state of the spin receiving the signal and thus requires state-dependent coupling. The state-dependent coupling results from the different slopes of the $G_k(r_k)$ property depending on whether the reversal of the variable x_i increases ($Z_{ki}\Delta x_i > 0$) or decreases ($Z_{ki}\Delta x_i < 0$) the resource variable r_k. The graph of the g_{ik}-r_k characteristic for an inequality constraint is shown in Fig. 3.

The energy-based model expressed in (1)–(6) is equivalent to the network model in (8)–(10). Note that the total energy H is obtained by integrating ΔH_j each time the variable x_j is reverted. Since the values of ΔH_i are obtained for all i, various local search algorithms can be run on these values. We will discuss the local search algorithm we used in the next subsection, but for now, let us assume that the inversion of variable x_j was chosen by some local search algorithm based on the available ΔH_i information.

Once the variable x_j to be flipped is determined, the state variables x, h_i, and r_k are updated by a difference calculation. The update is performed by propagating the influence of Δx_j through the network to obtain the final value of h_i. For the direct path from Δx_j, h_i and r_k are updated using $\Delta x_j = 1 - 2x_j$ with the following formula:

$$\begin{aligned} H &\leftarrow H - h_j\Delta x_j \\ h_i &\leftarrow h_i + W_{ij}\Delta x_j \\ r_k &\leftarrow r_k + Z_{kj}\Delta x_j. \end{aligned} \tag{11}$$

After updating H, h_i, and r_k using (11), x_j is changed to $x_j + \Delta x_j$ and we perform a differential update of g_{ik} in the second pass for k where r_k was updated:

$$\begin{aligned} &h_i \leftarrow h_i - \lambda_k \left[g_{ik}(r_k) - g_{ik}\left(r_k^{(old)}\right)\right] \ (i \neq j), \\ &h_i : no\ change\ (i = j). \end{aligned} \tag{12}$$

Here, $r_k^{(old)}$ is the value just before r_k is updated in (12). If r_k is updated, g_{ik} is calculated only for index i with $Z_{ik} \neq 0$.

The variable update described above requires a total of $O(nm)$ multiply-accumulate operations. Doing this in a single pass of parallel operations requires $O(nm)$ of parallelism, which becomes difficult to implement when n and m number several hundred or more. Therefore, we take the approach of performing $O(n)$ and $O(n + m)$ parallel updates in two passes. To take full advantage of the two-pass update method, the computing platform must be configured so that when one of the decision or dependent variables is updated, all of the coupling coefficients associated with that variable are read, and r_k and h_i can be updated in parallel.

This configuration allows a complex problem structure (such as the one in Fig. 1(b)) to be achieved in a shallow recurrent network (such as the one in Fig. 2(b)), where the dependent variable part is a two-layer network added in parallel to the single-layer network of the conventional Ising machine. Compared to the implementation of Ising machines based on the so-called "natural computing" concept, there is an advantage in that embedding is not required due to the high degree of freedom of the problem topology. To be fair, we should emphasize that our approach is limited to one variable inversion at a time and has the potential to create a bottleneck in accessing the coefficient memory in exchange for the problem-topology flexibility. Natural computing, in contrast, allows for $O(n)$ parallelism of operations but may lack flexibility in the problem topology.

C. IMPLEMENTATION OF VARIOUS ENERGY FUNCTIONS

In this section, we present several useful $G_k(r_k)$ functions and the form of g_{ik} derived from them. To treat the different forms of G_k in a unified manner, after computing the resource variable r_k using the matrix $[Z_{kj}]$, the intermediate variable y_k is generated using the appropriate activation function $y_k = f_k(r_k)$. The activation function f_k can be an identity function with real inputs and outputs for an additional energy function G_k representing inequality and equality constraints. Depending on the nature of G_k, the amount of data can be reduced by using integers instead of real numbers for the input and output of f_k or by restricting the output range (Fig. 4).

If the computing platform allows it, the processing elements that generate the resource variable r_k run in parallel to receive the change in x_j and compute r_k. The coefficients Z_{kj} needed for this calculation are assumed to be stored in a memory that can be quickly accessed by the arithmetic circuit that generates the resource variables r_k. Note that c_k is utilized only for the initialization of r_k and does not need to be stored in memory.

The intermediate variables y_k calculated from r_k are sent en masse to the decision variable x_i side. In the calculation of the value of the spin x_i, g_{ik} is computed by (10) using the information Z_{ik} and x_i stored in local memory that can be quickly accessed from the x_i side. In this case, Z_{ik} stores the same value as Z_{ki} to use (10) (Fig. 5).

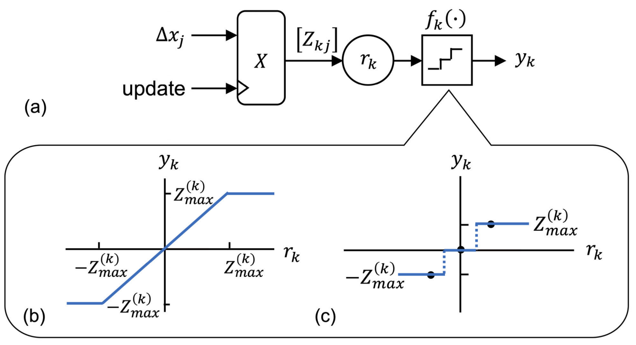

FIGURE 4. **Use of intermediate variable y_k. (a) The resource value r_k is calculated with a weighted sum of the decision variables x_j with weight Z_{kj}, and the intermediate variable y_k is generated with the activation function f_k and sent to the processing element for the decision variable. The activation function can be (b) an identity function with the output range restricted as needed or (c) various discrete functions, depending on the type of the dependent variable G_k.**

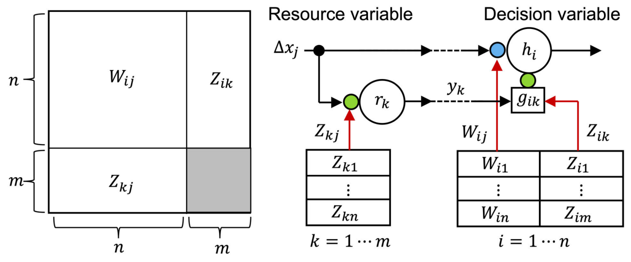

FIGURE 5. **Memory configuration for calculating changes in different variables. The memory of Z_{kj} is placed in the neighborhood of the processing element (PE) that generates the resource variable r_k to obtain the coefficients needed for the calculation, and the memory of Z_{ik} is placed in the neighborhood of the PE that calculates the local field h_i of the decision variable x_i.**

1) INEQUALITY AND EQUALITY CONSTRAINTS

When the coefficients Z_{kj} determining the resource variable r_k are various different real numbers, the additional energy function G_k that represents inequality constraints is given by

$$G_k(r_k) = \max(0,\ r_k), \tag{13}$$

$$y_k = f(r_k) = r_k, \tag{14}$$

where the activation function f_k is an identity function. From (10), (13), and (14), and refering to Fig. 3(b), considering that we should use Z_{ik}, which is identical to Z_{ki} but quickly accessible from the dependent variable, g_{ik} is calculated for $Z_{ik}\Delta x_i < 0$ by

$$g_{ik} = sign(Z_{ik}) \min[\max(y_k,\ 0),\ |Z_{ik}|]. \tag{15}$$

If $Z_{ik}\Delta x_i > 0$, g_{ik} is computed by shifting the argument of (15) to $y_k \leftarrow y_k + |Z_{ik}|$. We use a method in which an equality constraint is expressed as a combination of two inequality constraints.

From (14) and (15), the value of g_{ik} is constant if r_k exceeds the maximum value of $|Z_{ik}|$. Therefore, we can use an activation function that limits the absolute value of r_k to within $|Z_{ik}|$, as in Fig. 4(b).

On the other hand, there are also many inequality constraints where the change in the constraint resource variable r_k is limited to an increase or decrease of a constant value. For example, an inequality constraint may be imposed on the number of items that can be selected. In this case, the constraint resource variable r_k is an integer, and the change in r_k by inverting the decision variable is either $+1$, 0, or -1. In such cases, the three-step quantization function shown in Fig. 4(c) can be used.

2) HIGHER-ORDER AND PRODUCTS

The AND product of the positive and negative literals of a 0/1 binary variable produces 1 when all literals in the product are True, and 0 otherwise. Such a product can be utilized to provide a positive or negative energy bias for combinations of multiple variables, giving complex constraints that cannot be expressed in quadratic form without ancillary bits. G_k is expressed as

$$G_k = \prod_{j\in S_k} z_j, \tag{16}$$

where S_k is the set of variable indices j contained in the kth product and z_j is x_j if x_j is a positive literal in the product and $1 - x_j$ if it is a negative literal.

We generate the AND product (16) by using the inequality constraint (13). For this purpose, r_k is computed as a weighted sum with a weight of $Z_{kj} = 1$ if the variable x_j is a positive literal and $Z_{kj} = -1$ if it is a negative literal. If the bias coefficient c_k is 1 minus the number of positive literals in the product, then $r_k = 1$ only when all literals in the product are 1. Since the value of Z_{kj} is $+1/0/-1$, we can use the intermediate variable y_k with r_k quantized to three values:

$$y_k = \begin{cases} -1 & r_k \le -1 \\ 0 & r_k = 0 \\ +1 & r_k \ge 1. \end{cases} \tag{17}$$

3) HIGHER-ORDER XOR PRODUCTS

We include higher-order XOR products to enrich the expressiveness of the energy function H. The XOR product corresponds to the product of spin variables. Let G_k be the value of the kth higher-order XOR product and $\lambda_k G_k$ be its contribution to the Hamiltonian:

$$G_k = \prod_{j\in S_k} \sigma_j, \tag{18}$$

where S_k is the set of variable indices j contained in the kth higher-order product. We assume the spin representation of x_i is $\sigma_i = (-1)^{x_i} = 1 - 2x_i$. The spin product G_k is -1 if the number of σ_j that is -1 in the product is odd and 1 if it is even. Therefore, G_k is determined from the even and odd values of r_k, which is calculated from (6) by assuming $Z_{kj} = 1$ if x_j is included in the spin product G_k, $Z_{kj} = 0$ if x_j is not included, and c_k equals zero.

$$G_k = y_k = \begin{cases} +1 & even\ r_k \\ -1 & odd\ r_k. \end{cases} \tag{19}$$

If the variable σ_i is contained in the kth product, i.e., $Z_{ki}(= Z_{ik}) \neq 0$, then $G_k \to -G_k$. Therefore g_{ik} is expressed as

$$g_{ik} = -2Z_{ik} y_k \Delta x_i. \tag{20}$$

Many higher-order spin products can be made by multiplying a common product of the lower order by several different spins. In such cases, the dependent variable and the required memory space can be reduced by formulating $y_q = y_{ki} = y_k \Delta x_i$ instead of y_k as the dependent variable and the index pair (k, i) as a one-dimensional index q.

D. LOCAL SEARCH ALGORITHM

1) REJECTION-FREE SELECTION

The energy change $\Delta H_i (i = 1, \cdots, n)$ output by the network of artificial spins is used to guide the local search. For the lowest-level local search, we use the rejection-free MCMC method [28], [29], which utilizes the following selection rule:

$$j = \underset{i}{argmin}[\max(0,\ \Delta H_i) + T\ log\ (-log\,(r_i))], \tag{21}$$

where T is the temperature and r_i is a random number uniformly distributed with $0 < r_i < 1$, independent for each variable x_i. This selection rule reduces the time to reach the optimum value for problems with small acceptance probabilities of proposed moves because the transition from a state happens without rejection.

In rejection-free MCMC, the probability distribution of the transition-destination state is identical to a normal serial selection MCMC continued until the transition is accepted. As suggested by this, the MCMC chain that implements the rejection-free selection rule is equivalent to a stochastic process for the original chain with the repetition of the same state removed. The probability that a bit flip is accepted by the original chain is given by

$$A_i = min\left[1, \exp\left(-\beta \Delta H_i\right)\right]. \tag{22}$$

The probability of transitioning to a state different from the current one for each trial in the original chain, i.e., the escape probability, α, is given by

$$\alpha = \left(\frac{1}{n}\right) \sum_{i=1}^{n} A_i. \tag{23}$$

Escape probabilities are needed to compute the importance weights of the samples obtained from the rejection-free MCMC, i.e., the weights to be multiplied on the samples in formulas such as expectation calculations, as will be shown later.

2) PARALLEL TEMPERING (EXCHANGE MONTE CARLO)

As modern processing hardware designs have shifted heavily toward multi-core systems, there are many methods available for running multiple local search instances in parallel and combining their results via a higher-level algorithm, such as genetic algorithms [30] or population annealing [31]. Among these methods, we utilize the replica-exchange Monte Carlo methods, also known as Parallel Tempering (PT) [32], [33], with multiple replicas to improve the efficiency of the search in addition to rejection-free MCMC.

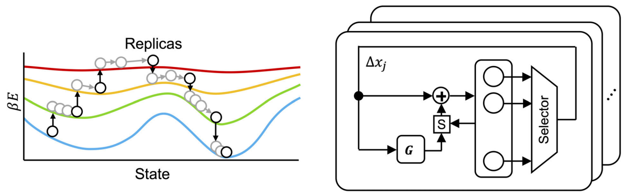

FIGURE 6. **Parallel tempering (Exchange Monte Carlo). Parallel tempering runs multiple searches (replicas) with different temperatures on the same problem instance. Each time a variable is inverted a certain number of times, the replicas exchange temperatures between neighboring temperatures. Since the replicas with relatively higher energy move to a higher temperature, the temperature exchange increases the likelihood of escaping the local solution and reaching a better solution.**

In the PT system, P parallel stochastic local search instances or replicas each run at a different inverse temperature β from β_{min} to β_{max} in a ladder, as shown in Fig. 6. Let $p(= 1, 2, \cdots, P)$ be the index of the replicas and $\mathbf{x}_p$ and β_p be the state and inverse temperature of the pth replica. After running MCMC operations for a certain number of time steps (iterations) for each replica, temperatures are exchanged between neighboring temperature replicas with a probability determined by the difference between the energy and the inverse temperature. In the case of rejection-free selection, the ratio of the escape probability $\alpha(\mathbf{x}, \beta)$ at inverse temperature β and state $\mathbf{x}$ appears in the exchange probability A_p:

$$A_p = min\left[1, \mathcal{R}_\alpha\ exp\left[\Delta\beta_p \Delta H_p\right]\right], \tag{24}$$

$$\mathcal{R}_\alpha = \frac{\alpha\left(\mathbf{x}_p,\ \beta_{p+1}\right)\alpha\left(\mathbf{x}_{p+1},\ \beta_p\right)}{\alpha\left(\mathbf{x}_{p+1},\ \beta_{p+1}\right)\alpha\left(\mathbf{x}_p,\ \beta_p\right)}, \tag{25}$$

where $\Delta\beta_p = \beta_{p+1} - \beta_p$ and $\Delta H_p = H_{p+1} - H_p$ [29].

As far as we have experienced, approximating $\mathcal{R}_\alpha$ appearing in (24) as 1 between nearby temperatures does not cause any major problems when solving optimization problems [34]. However, this approximation introduces errors in the probability distribution obtained by sampling. Therefore, when reproducing the probability distribution by sampling, it is necessary to use (25) without approximating $\mathcal{R}_\alpha$ as 1.

III. EXPRESSIVE POWER OF NEBULA ARCHITECTURE

A. ENERGY FUNCTION WITH MANY-BODY INTERACTIONS

In the previous section, we showed that *Nebula* can have an additional energy function G_k with a weighted sum of decision variables as arguments. The number of additional functions, or dependent variables G_k, is only limited by the number of coupling coefficients the system can handle. Since the functional form of G_k is arbitrary, in principle the proposed extended Ising machine can implement interactions of any order (up to problem size n). Note that the scheme is expressive enough even when restricting the functional form of G_k to that described above.

In addition to the convenience of ReLU-type additive energy functions in formulating various optimization problems, AND and XOR products (i.e., spin products) can be used to produce arbitrary energy landscapes, or pseudo Boolean functions, subject to certain constraints, as shown below.

1) EXPANSION USING AND PRODUCTS

The AND product of a positive or negative literal will be 1 only for a certain bit pattern in the product, and zero otherwise. Therefore, if the number of such interactions mediated by the bit group is q, $q \cdot 2^K$ dependent variables are needed to express an arbitrary energy landscape with K-body higher order interactions.

If there is no bound on the value of K, which can be equal to the size of the problem n, an exponential number of 2^n dependent variables is needed to implement an arbitrary interaction. Note, however, that if $K \ll n$ and there are q K-bounded couplings and q is $O(n)$, the number of dependent variables required, $q \cdot 2^K$, is much less than 2^n.

2) EXPANSION USING WALSH FUNCTIONS

Another example of where arbitrary functions can be constructed is the XOR product, or spin product, as discussed in II-C. The XOR product G_k can be written as

$$G_k = \varphi_k(\mathbf{x}) = (-1)^{\sum_{j=0}^{n-1} k_j x_j}, \tag{26}$$

$$k = \sum_{j=0}^{n-1} k_j 2^j, \tag{27}$$

$$k_j = Z_{kj} = \begin{cases} 1 & j \in S_k \\ 0 & j \notin S_k. \end{cases} \tag{28}$$

The value of G_k is equal to the Walsh function $\varphi_k(\mathbf{x})$ characterized by two parameters n and k, where k is an integer with the value of Z_{kj} as the coefficient k_j of the binary expansion (27). Thanks to the orthogonality and completeness of the Walsh function, any energy landscape can be expanded. Also, as in the case of the AND product, if there are q K-bounded interactions, the number of dependent variables required is just $q \cdot 2^K$.

Note that the linear and quadratic terms formulated in 0/1 binary variables can also be formulated in spin variables if the coefficients W_{ij} and b_i are appropriately transformed. Therefore, the entire energy function can be expanded by using Walsh functions.

Walsh transforms and Walsh analysis are useful in evolutionary computation for analyzing the energy landscapes of many well-known combinatorial optimization problems and real-world models [35]. For example, Walsh analysis can be utilized to derive algebraically closed expressions for the various statistical moments (i.e., expectation and variance) of various combinatorial optimization problem landscapes [36], [37], [38]. Such an approach enables us to derive useful information about the structure of the landscape of combinatorial problems without having to sample the entire objective function of the problem in the neighborhood of interest.

Our proposed method can directly treat the Walsh function as a component of the energy function to be used in the optimization algorithm. It has the potential to not only analyze the nature of the problem but also implement efficient optimization algorithms based on the analysis.

3) SAMPLING FROM BOLTZMANN DISTRIBUTION

Since the rejection-free MCMC method does not reject proposed transitions to the next state, it yields a jump chain that stays in the same state only once [29]. This gives the possibility of speeding up the optimization, but the samples must be properly weighted to reproduce the probability distribution by sampling.

The number of times to stay in the same state in the original chain is called the multiplicity and is denoted by M, where M is a random variable that follows a geometric distribution defined by the escape probability α from the state. Using the samples obtained from the jump chain, the expected value of the state function can be obtained as a sample average with sample weights M. Thus, the extended Ising machine can be utilized not only as an optimization solver but also as a sampler under the target distribution $\pi(\boldsymbol{x})$ [29]. If J_k is the kth sampled state and $s(J_k)$ is an arbitrary function of state J_k, the expected value of $s(J_k)$ is given by the following weighted sampling:

$$E_\pi(h) = \lim_{L\to\infty} \frac{\sum_{k=1}^{L} M_k s(J_k)}{\sum_{k=1}^{L} M_k} \tag{29}$$

Here, M_k is the number of stays calculated from the escape probability α_k for the transition from the kth state to the next state. Note that (29) is also valid when $1/\alpha_k$ is used instead of the random number M_k that follows a geometric distribution.

The possibility of sampling provides a potential application of the proposed scheme to learning. Since the expression for $H(\boldsymbol{x})$ is explicitly given, the gradient of $H(\boldsymbol{x})$ with various parameters (e.g., W_{ij} and b_i) is also explicitly obtained, and its expected value can be obtained by sampling. Thus, the ability to perform sampling can be applied to various types of learning. We leave this topic to future research.

4) COMPARISON WITH TWO-BODY BOLTZMANN MACHINE

The proposed method has an overhead associated with the dependent variable that is not present in the two-body Boltzmann machine. However, considering the increase in problem size and the complexity of handling higher-order interactions with only two-body coupling, our experience shows that the proposed method outperforms the two-body Boltzmann machine in many situations, as shown in Section IV. To prove this with high confidence by theory and experiment is our future work. Instead, we briefly discuss the difference in the expressiveness between the proposed method and the Boltzmann machine.

Since the proposed method is based on n-bit fully coupled decision variables connected to a one-layer dependent variable network with m units, the comparison is made between a Boltzmann machine with fully coupled n visible units and a single-layer m hidden units added. The energy of this Boltzmann network is as follows:

$$H\left(\mathbf{x}, \mathbf{x}^{(\mathbf{h})}\right) = E\left(\mathbf{x}\right) - \sum_{k=1}^{m} c_k x_k^{(h)} - \sum_{i=1}^{n}\sum_{k=1}^{m} x_i Z_{ik} x_k^{(h)}, \quad (30)$$

where $\mathbf{x}$ is the visible variable, $\mathbf{x}^{(\mathbf{h})}$ is the hidden variable, E is the quadratic energy term of the fully coupled Boltzmann machine, Z_{ik} is the coupling coefficient between the visible and hidden units, and c_k is the bias constant of the kth hidden unit. From (30), the state occupancy probability $P\left(\mathbf{x},\ \mathbf{x}^{(\mathbf{h})}\right)$ is calculated assuming the Boltzmann distribution and marginalized by $\mathbf{x}^{(\mathbf{h})}$ to obtain $P\left(\mathbf{x}\right)$

$$P\left(\mathbf{x}\right) \propto \exp\left\{-\beta E + \sum_{k=1}^{M} \ln\left[1 + exp\left[\beta\left(c_k + \sum_{i=1}^{N} Z_{ik} x_i\right)\right]\right]\right\}. \quad (31)$$

Letting the exponential part of (31) be $-\beta H_{eff}$ and approximating H_{eff} at $\beta \to \infty$, we obtain

$$H_{eff}\left(\mathbf{x}\right) \approx E\left(\mathbf{x}\right) - \sum_{k=1}^{m} \max\left[0,\ c_k + \sum_{i=1}^{n} Z_{ik} x_i\right]. \quad (32)$$

The second term on the right side of (32) is the sum of the rectified-linear type functions, which can represent any function if the number of neurons in the hidden layer is sufficiently large. Note that the sign attached to the max function in the second term of the right-hand side of $H_{eff}(\boldsymbol{x})$ in (32) is limited to negative. To reproduce the penalty function for inequality constraints, which is often used in optimization problems, this limitation requires a larger number of hidden layer neurons per inequality constraint. The method proposed in this paper has the advantage that the penalty function resulting from an inequality constraint can be expressed in a single dependent variable, which simplifies the formulation.

IV. NUMERICAL RESULTS

This section presents numerical experiments as a proof of concept for the proposed scheme. All numerical experiments were performed on a 3.60 GHz Intel@Core i9-10850K CPU with ten CPU cores and the cache size of 20408 kB. For all problem instances, 16 threads were used. The value of the time to reach the optimal solution (Time-to-Solution: TtS) is the average value for ten different random-number seeds.

A. QPLIB INSTANCES

We selected eight instances from the QPLIB [39] that consisted of binary variables, had quadratic cost terms, and contained multiple inequality linear constraints (but no equality constraints). The energy H to be minimized is given in the form (1), (2), (3), and G_k is expressed as

$$G_k = \max\left(0, \sum_{i \in D} w_{ki} x_i - w_{\max}\right). \quad (33)$$

The value of λ_k is a fixed value determined by

$$\lambda_k = 2 \max_i \left(\frac{b_i + \sum_{j=1}^{n} \left|W_{ij}\right|}{\left|Z_{ki}\right|}\right). \quad (34)$$

The instance ID, number of variables n, number of constraints m, and TtS in seconds are listed in Table 1. The number of replicas was 32, and the parallel temperatures ranged from T = 0.01 to 64 for 3803 and from T = 0.01 to 16 for the remaining instances. The ratio of adjacent temperatures was kept constant. All instances reached the correct answer with ten random seeds.

For comparison, Gurobi's solution times from Mittelmann's list [40] are given in the format a/b (sec), where a is the time to obtain a solution with guaranteed correctness and b is the time until the incumbent solution matches the best known solution. Comparison with these times shows that our method is about the same or slightly better than Gurobi's in the time it takes for the incumbent solution to match the best known value.

None of the QPLIB instances shown in Table 1 could be solved using a pure QUBO formulation, which does not utilize dependent variables in the proposed method and represents inequality constraints in slack variables. Instances with as few as 5–11 inequality constraints in QPLIB (10043, 10044, 10048, 10058, 10067, 10069) were also tried. The QUBO formulation did not solve those instances even with 10^7 iterations, but the proposed method solved all of them in less than a second.

TABLE 1. Value of time-to-solution (TtS) for the selected QPLIB instances. These instances have quadratic costs and multiple linear inequality constraints. Note that TtS was compared to Gurobi in Mittelmann's list.

Instance	n	m	Gurobi (sec/sec)*	This work (sec)
2357	240	2240	43/2	0.33
2359	306	3264	13/3	1.33
3757	552	8096	57/8	14.4
3762	90	480	2/0	0.03
3772	380	4560	261/5	1.55
3803	190	2280	19/11	1.57
3883	182	1456	775/268	0.14
10048	150	5	48/0	0.11

*In the form a/b, where a is the time it takes to obtain a solution that is guaranteed to be correct and b is the time it takes for the incumbent solution to match the best known solution.

B. WEIGHTED MaxSAT PROBLEMS

In this subsection, as a typical example of a problem with higher-order terms, we solve weighted maximum satisfiability (MaxSAT) problems with higher-order products of different orders.

Note that the purpose of this numerical experiment is not to show that the proposed method can compete with dedicated SAT solvers but rather to demonstrate that it can provide formulation freedom for optimization problems involving

TABLE 2. TtS of the weighted MaxSAT instances, which are formulated by an energy function involving higher-order AND products.

Instance	n	m	k*	TtS (sec)
yagi10b	10	133	2-8	0.008
yagi20b	20	237	2-10	0.022
yagi40b	40	456	2-14	0.562
jnh01.sat	100	850	2-14	2.303

*Number of literals per clause

cost terms represented by various logic functions with orders well beyond 3.

MaxSAT is the problem of finding a set of Boolean variables $\{x_1, \cdots, x_n\}$ that maximizes the number of satisfied clauses $\{C_1, \cdots, C_M\}$ in a conjunction normal form $\Psi = C_1 \wedge C_2 \cdots \wedge C_M$, where

$$C_i = z_{i_1} \vee \cdots \vee z_{i_k}, 1 \leq i \leq M. \tag{35}$$

The variables $z_{i_1}, \cdots, z_{i_k}$ in (35) are selected from another set of Boolean variables $x_1, \cdots, x_n, \overline{x}_1, \cdots, \overline{x}_n$. The weighted MaxSAT problem is a generalization of the maximum satisfiability problem in which each clause C_i is assigned a positive weight w_i. The goal of the problem is to determine the variables $x_1, \cdots, x_n$ that maximize the sum of the weights of the satisfied clauses.

When solving the weighted MaxSAT problem with the proposed extended Ising machine, the positive and negative literals that make up the clauses are inverted to create AND products, which are additional energy functions G_k. Since each AND product takes the value of 1 if the corresponding clause is not satisfied, the weighted sum of satisfied clauses can be maximized by minimizing the weighted sum of G_k. H in this problem is given in the following form with constraint terms only:

$$H(\mathbf{x}) = \sum_{k=1}^{m} \lambda_k \prod_{j \in S_k} z_j. \tag{36}$$

The problem instances solved were yagi10b, yagi10b, yagi20b, and yagi40b (available at [41]), and jnh1.sat, which is one of the random SAT instances with variable length clauses [42]. These are problems with decision variables in the range of ten to 100, number of clauses in the range of 133 to 850, and number of literals per clause, k, in the range of two to 14. Using the AND product of the proposed scheme, these problems are formulated directly with decision spins equal to the number of decision variables n and the number of dependent variables equal to the number of clauses m. Here, the order of the AND product is equal to k, and even the 14th order can easily be handled.

For these problems, we used 16 replicas. The temperature of the replicas was set to a power of 2 in the range T = 0.125–4096. As shown in Table 2, the proposed scheme reached the best known values of these problems within 8 msec to 2.3 sec.

C. WALSH TRANSFORM AND BOLTZMANN SAMPLING

Here, we show an example of how our proposed scheme can generate arbitrary energy landscapes. For this purpose, we generated a 10-bit pseudo-Boolean function, i.e., an energy landscape, using yagi10b, one of the weighted MaxSAT instances listed in Table 2. Under the hypothetical situation that we only know the value of the generated energy landscape $H(\mathbf{x})$, we generated from this energy landscape a problem defined as a weighted sum of higher-order spin products by a Walsh transform:

$$w_k = \frac{1}{2^n} \sum_{\mathbf{x} \in \{0,1\}^n} H(\mathbf{x}) \cdot \varphi_k(\mathbf{x}), \tag{37}$$

$$H(\mathbf{x}) = \sum_{k=0}^{2^n-1} w_k \varphi_k(\mathbf{x}) = \sum_{k=0}^{2^n-1} w_k \prod_{j \in S_k} \sigma_j. \tag{38}$$

We then performed optimization and Boltzmann sampling from the energy landscape re-formulated by (38) in terms of the higher-order spin products (Fig. 7).

The results of Boltzmann sampling from this problem by our proposed method for the top 20 occupancy probabilities are shown in Fig. 8. As we can see, the theoretical value of the probability of occurrence of the state and the value obtained from the sampling mean (29) with the sampling weight as $1/\alpha_k$ are in good agreement. The value of the total variation distance is less than 0.01 when the number of iterations is 10^6. The CPU time required to sample 10^6 times from this problem instance was about 20 seconds for the AND product formulation and 70 seconds for the Walsh expansion formulation. Not surprisingly, the optimization and sampling

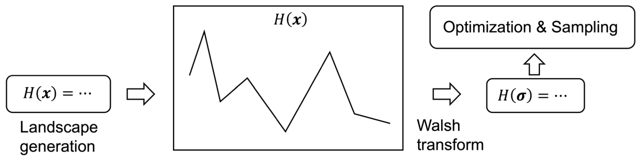

FIGURE 7. Setup for the energy landscape synthesis experiment, where the energy landscape is generated from the yagi10b instance. By Walsh transforming this energy landscape, we obtained a problem instance formulated as a weighted sum of higher-order spin products, which was sampled and optimized by an extended Ising machine.

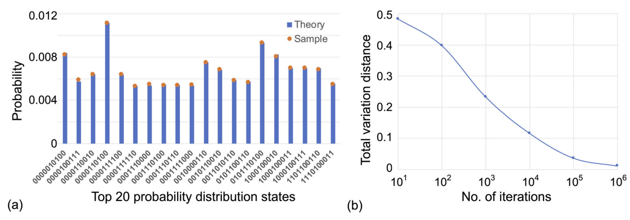

FIGURE 8. Sampling from energy landscape synthesized from yagi10b's landscape via Walsh transform. (a) Comparison of theoretical and measured top 20 state occupancy probabilities. (b) Dependence of total variation distance on the number of iterations.

results were exactly the same for the formulation using the AND products of the Boolean variables and for the formulation using the spin products obtained by the Walsh transform. Moreover, the number of variable inversions that reach the correct solution in the two formulations when starting from the same initial values is also exactly the same. This is because the MCMC search in our proposed method is guided by the energy change due to the decision-variable flip, and the energy change is designed to be independent of what type of dependent variables are used.

V. CONCLUSION

We have proposed an algorithm and a computational architecture called *Nebula* for constructing Ising machines with the higher-order interactions needed to handle inequality constraints and higher-order energy terms. Specifically, a network of dependent variables is added to express an extended energy function with interactions mediated by these variables, and a recursive network composed of the decision spins and dependent variables computes the energy change when a decision spin is reversed for all variables in parallel. The energy change is then utilized to minimize the energy function by local search or for sampling to reproduce the Boltzmann distribution.

Although the dependent-variable network for higher-order interactions is a modest addition, it has the effect of expanding the range of problems that the Ising machine can handle. Since the dependent variable mediating the various interactions is uniquely determined by the decision spins, there is no increase in the problem dimensions due to the addition of dependent variables, so it does not complicate the problem as usual ancillary spins do. Numerical experiments on inequality constraints and higher-order products confirm that the use of dependent variables enables a straightforward formulation of these problems, thus increasing the value of the Ising machines in both solving and sampling a wider range of problems.

The primary contribution of this work is, as stated above, to extend the energy functions available in the Ising machine to facilitate the formulation of optimization problems. The flexibility of the energy function opens up a wide range of possibilities. Since *Nebula* can handle higher-order products, arbitrary energy functions with k-body interactions can be implemented within the computational elements and memory capacity of the platform. Since the higher-order spin product corresponds to the Walsh function, it is expected to have applications such as extracting information from the energy landscape based on the Walsh expansion and using it to implement efficient optimization methods. While the *Nebula* architecture is capable of fast sampling as well as optimization, which could be effectively used for learning, this paper only points out the possibility. It is also necessary to compare how the dependent variable extension of the energy function compares to the well-known restricted Boltzmann machine in terms of the learning and flexibility of the resulting energy function. These are topics for future work.

ACKNOWLEDGMENT

The authors sincerely appreciate Prof. Jun Okubo of Saitama University and Prof. Kazue Kudo of Ochanomizu University for their participation in technical discussions. They also thank Prof. Ali Sheikholeslami and his team at the University of Toronto for their insightful comments and advice. They also thank Ayaka Fujimoto and Haruka Akishima of Ochanomizu University helped create the code for the numerical experiments.

REFERENCES

[1] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

[2] K. Fukushima, "Cognitron: A self-organizing multilayered neural network," *Biol. Cybern.*, vol. 20, nos. 3–4, pp. 121–136, 1975.

[3] K. Rupp. (2022). *50 Years of Microprocessor Trend Data*. [Online]. Available: https://github.com/karlrupp/microprocessor-trend-data

[4] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2014, pp. 10–14, doi: 10.1109/ISSCC.2014.6757323.

[5] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, 1982.

[6] D. Ackley, G. Hinton, and T. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognit. Sci.*, vol. 9, no. 1, pp. 147–169, Mar. 1985.

[7] C. R. Schneider and H. C. Card, "Analog CMOS deterministic Boltzmann circuits," *IEEE J. Solid-State Circuits*, vol. 28, no. 8, pp. 907–914, Feb. 1993.

[8] B. E. Boser, E. Sackinger, J. Bromley, Y. Le Cun, and L. D. Jackel, "An analog neural network processor with programmable topology," *IEEE J. Solid-State Circuits*, vol. 26, no. 12, pp. 2017–2025, May 1991.

[9] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, vol. 1. Cambridge, MA, USA: MIT Press/Bradford Books, 1986, ch. 6, pp. 195–281.

[10] Y. Freund and D. Haussler, "Unsupervised learning of distributions on binary vectors using 2-layer networks," in *Proc. 4th Int. Conf. Neural Inf. Process. Syst.*, 1991, pp. 912–919.

[11] *D-Wave Systems*. Accessed: May 6, 2021. [Online]. Available: https://www.dwavesys.com

[12] P. I. Bunyk, E. M. Hoskinson, M. W. Johnson, E. Tolkacheva, F. Altomare, A. J. Berkley, R. Harris, J. P. Hilton, T. Lanting, A. J. Przybysz, and J. Whittaker, "Architectural considerations in the design of a superconducting quantum annealing processor," *IEEE Trans. Appl. Supercond.*, vol. 24, no. 4, pp. 1–10, Aug. 2014, doi: 10.1109/TASC.2014.2318294.

[13] T. Takemoto, M. Hayashi, C. Yoshimura, and M. Yamaoka, "A 2 × 30k-spin multichip scalable annealing processor based on a processing-in-memory approach for solving large-scale combinatorial optimization problems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2019, pp. 52–54, doi: 10.1109/ISSCC.2019.8662517.

[14] T. Takemoto, K. Yamamoto, C. Yoshimura, M. Hayashi, M. Tada, H. Saito, M. Mashimo, and M. Yamaoka, "A 144Kb annealing system composed of 9 ×16 Kb annealing processor chips with scalable chip-to-chip connections for large-scale combinatorial optimization problems," in *Proc. IEEE Int. Solid- State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Feb. 2021, pp. 64–66, doi: 10.1109/isscc42613.2021.9365748.

[15] 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, doi: 10.1126/sciadv.aav2372.

[16] H. Goto, K. Endo, M. Suzuki, Y. Sakai, T. Kanao, Y. Hamakawa, R. Hidaka, M. Yamasaki, and K. Tatsumura, "High-performance combinatorial optimization based on classical mechanics," *Sci. Adv.*, vol. 7, no. 6, Feb. 2021, Art. no. eabe7953, doi: 10.1126/sciadv.abe7953.

[17] K. Tatsumura, A. R. Dixon, and H. Goto, "FPGA-based simulated bifurcation machine," in *Proc. 29th Int. Conf. Field Program. Log. Appl. (FPL)*, Barcelona, Spain, Sep. 2019, pp. 59–66, doi: 10.1109/FPL.2019.00019.

[18] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, "Physics-inspired optimization for quadratic unconstrained problems using a digital annealer," *Frontiers Phys.*, vol. 7, p. 48, Apr. 2019, doi: 10.3389/fphy.2019.00048.

[19] S. Matsubara, M. Takatsu, T. Miyazawa, T. Shibasaki, Y. Watanabe, K. Takemoto, and H. Tamura, "Digital annealer for high-speed solving of combinatorial optimization problems and its applications," in *Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Beijing, China, Jan. 2020, pp. 667–672, doi: 10.1109/ASP-DAC47756.2020.9045100.

[20] F. Glover, J. K. Hao, and G. Kochenberger, "Polynomial unconstrained binary optimisation—Part 1," *Int. J. Metaheuristics*, vol. 1, no. 3, p. 232, 2011, doi: 10.1504/ijmheur.2011.041196.

[21] M. K. Bashar and N. Shukla, "Designing Ising machines with higher order spin interactions and their application in solving combinatorial optimization," *Sci. Rep.*, vol. 13, no. 1, p. 9558, Jun. 2023, doi: 10.1038/s41598-023-36531-4.

[22] A. Lucas, "Ising formulations of many NP problems," *Frontiers Phys.*, vol. 2, p. 5, Feb. 2014, doi: 10.3389/fphy.2014.00005.

[23] R. Xia, T. Bian, and S. Kais, "Electronic structure calculations and the Ising Hamiltonian," *J. Phys. Chem. B*, vol. 122, no. 13, pp. 3384–3395, Apr. 2018, doi: 10.1021/acs.jpcb.7b10371.

[24] L. Pusey-Nazzaro and P. Date, "Adiabatic quantum optimization fails to solve the knapsack problem," 2020, *arXiv:2008.07456*.

[25] F. Yin, H. Tamura, Y. Furue, M. Konoshima, K. Kanda, and Y. Watanabe, "Extended Ising machine with additional non-quadratic cost functions," *J. Phys. Soc. Jpn.*, vol. 92, no. 3, Mar. 2023, Art. no. 034002, doi: 10.7566/jpsj.92.034002.

[26] A. DeGloria, P. Faraboschi, and M. Olivieri, "Efficient implementation of the Boltzmann machine algorithm," *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 159–163, May 1993.

[27] S. V. Isakov, I. N. Zintchenko, T. F. Rønnow, and M. Troyer, "Optimised simulated annealing for Ising spin glasses," *Comput. Phys. Commun.*, vol. 192, pp. 265–271, Jul. 2015, doi: 10.1016/j.cpc.2015.02.015.

[28] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, "A new algorithm for Monte Carlo simulation of Ising spin systems," *J. Comput. Phys.*, vol. 17, no. 1, pp. 10–18, Jan. 1975.

[29] J. S. Rosenthal, A. Dote, K. Dabiri, H. Tamura, S. Chen, and A. Sheikholeslami, "Jump Markov chains and rejection-free metropolis algorithms," *Comput. Statist.*, vol. 36, no. 4, pp. 2789–2811, Dec. 2021, doi: 10.1007/s00180-021-01095-2.

[30] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*, 2nd ed. Cambridge, MA, USA: MIT Press, 1992.

[31] K. Hukushima, "Population annealing and its application to a spin glass," in *Proc. AIP Conf.*, 2003, pp. 200–206, doi: 10.1063/1.1632130.

[32] R. H. Swendsen and J.-S. Wang, "Replica Monte Carlo simulation of spin-glasses," *Phys. Rev. Lett.*, vol. 57, no. 21, pp. 2607–2609, Nov. 1986.

[33] K. Hukushima and K. Nemoto, "Exchange Monte Carlo method and application to spin glass simulations," *J. Phys. Soc. Jpn.*, vol. 65, no. 6, pp. 1604–1608, Jun. 1996.

[34] K. Dabiri, M. Malekmohammadi, A. Sheikholeslami, and H. Tamura, "Replica exchange MCMC hardware with automatic temperature selection and parallel trial," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1681–1692, Jul. 2020, doi: 10.1109/TPDS.2020.2972359.

[35] P. F. Stadler, "Towards a theory of landscapes," in *Complex Systems and Binary Networks* (Lecture Notes in Physics). Berlin, Germany: Springer, 1995.

[36] R. B. Heckendorn, S. Rana, and D. Whitley, "Polynomial time summary statistics for a generalization of MAXSAT," in *Proc. 2nd Annu. Conf. Genetic Evol. Comput.*, 2000, pp. 919–926.

[37] A. M. Sutton, L. Darrell Whitley, and A. E. Howe, "Computing the moments of k-bounded pseudo-Boolean functions over Hamming spheres of arbitrary radius in polynomial time," *Theor. Comput. Sci.*, vol. 425, pp. 58–74, Mar. 2012, doi: 10.1016/j.tcs.2011.02.006.

[38] F. Chicano and E. Alba, "Elementary landscape decomposition of the 0–1 unconstrained quadratic optimization," *J. Heuristics*, vol. 19, no. 4, pp. 711–728, Aug. 2013, doi: 10.1007/s10732-011-9170-6.

[39] *QPLIB A Library of Quadratic Programming Instances*. Accessed: Aug. 19, 2023. [Online]. Available: https://qplib.zib.de

[40] H. Mittelmann. *Binary Non-Convex QPLIB Benchmark*. Accessed: Aug. 26, 2023. [Online]. Available: http://plato.asu.edu/ftp/qplib.html

[41] *Prof. Yagiura's Weighted MaxSAT Problem Example*. Accessed: Aug. 19, 2023. [Online]. Available: https://www-or.amp.i.kyoto-u.ac.jp/members/yagiura/maxsat.html

[42] H. H. Hoos and T. Stãtzle. *SATLIB: An Online Resource for Research on SAT*. Accessed: Aug. 19, 2023. [Online]. Available: https://www.cs.ubc.ca/∼hoos/SATLIB/benchm.html

YASUHIRO WATANABE received the B.S. and M.S. degrees in electronic science and engineering from Kyoto University, Japan, in 1994 and 1996, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan, in 1996, and was involved in the research and development of digital TV LSIs and video codec LSIs. Since 2015, he has moved into the field of domain-specific computing. He is currently with Fujitsu Ltd., and engaged in research and development on computing systems for combinatorial optimization. His research interests include domain-specific circuit architectures, novel computing hardware architectures, and high-performance computing systems.

HIROTAKA TAMURA (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Tokyo University, Japan, in 1977, 1979, and 1982, respectively.

After joining Fujitsu Laboratories, in 1982, he initially worked on superconducting devices, such as Josephson junctions and high-temperature superconducting devices. From 1988 to 1989, he was a Visiting Scholar with the University of California at Berkeley. In 1996, he entered the field of high-speed CMOS signaling and was involved in the development of high-speed multi-channel I/O for server interconnects. Since then, he has been working in the area of architectural and transistor-level design for CMOS high-speed signaling circuits. Since 2014, he has expanded his expertise to devices, circuits, and architectures for post-Moore era computing. After leaving Fujitsu, in 2020, he founded DXR Laboratory Inc., where he is actively engaged in these research areas. He was a member of the Far East Program Committee of ISSCC, from 1997 to 1999, and on the International Technical Program Committee of ISSCC as part of the Wireline Communication Subcommittee. He received the 51st Okochi Memorial Prize for the development and commercialization of multi-gigabit CMOS high-speed I/O technology.

YUKI FURUE received the B.E. and M.E. degrees in information engineering from Saitama University, Japan, in 2021 and 2023, respectively.

In graduate school, he studied the scalability of Ising machines in continuous variable optimization problems. While in graduate school, from 2020 to 2023, he was a part-time Employee with DXR Laboratory Inc., under the supervision of Dr. Hirotaka Tamura to extend the functionality of optimization algorithms with Markov chain Monte Carlo methods, and was involved in creating proof-of-principle codes for inequality constraints and extension to higher-order terms. In 2021, he was a Summer Intern with Fujitsu Ltd. He joined Fujitsu Ltd., in 2023, where he is currently working on extending the functionality of digital annealing systems.

FANG YIN received the M.S. degree in computer science and systems engineering from Kobe University, Japan, in 2008.

After graduation, she joined Fujitsu Laboratories Ltd., where she was engaged in the research and development of video codec LSI and partial image retrieval accelerators. She is currently working at Fujitsu Ltd., and engaged in research and development of software algorithms to speed up the process of solving combinatorial optimization problems.