Binary optimization by momentum annealing

Takuya Okuyama,[1,*] Tomohiro Sonobe,[2] Ken-ichi Kawarabayashi,[2] and Masanao Yamaoka[1]
[1]*Hitachi, Ltd., 1-280, Higashi-Koigakubo, Kokubunji-shi, Tokyo 185-8601, Japan*
[2]*National Institute of Informatics, Hitotsubashi 2-1-2, Chiyoda-ku, Tokyo 101-8403, Japan*

(Received 3 December 2018; revised manuscript received 12 February 2019; published 10 July 2019)

One of the vital roles of computing is to solve large-scale combinatorial optimization problems in a short time. In recent years, methods have been proposed that map optimization problems to ones of searching for the ground state of an Ising model by using a stochastic process. Simulated annealing (SA) is a representative algorithm. However, it is inherently difficult to perform a parallel search. Here we propose an algorithm called momentum annealing (MA), which, unlike SA, updates all spins of fully connected Ising models simultaneously and can be implemented on GPUs that are widely used for scientific computing. MA running in parallel on GPUs is 250 times faster than SA running on a modern CPU at solving problems involving 100 000 spin Ising models.

DOI: 10.1103/PhysRevE.100.012111

I. INTRODUCTION

Problems in various applications such as social network analysis, circuit layout, and machine learning are difficult to solve precisely [1–5]. In fact, many of them are NP-hard [6]. In such cases, the techniques of combinatorial optimization are considered essential to arrive at adequate solutions. In particular, computing with Ising models has attracted a lot of interest. An Ising model consists of N binary spins $\sigma_i \in \{-1, 1\}$ with couplings J_{ij} and local fields h_i. Its Hamiltonian is defined as

$$H(\sigma_1, \ldots, \sigma_N) = -\sum_{i<j} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i. \tag{1}$$

The spin-spin connection can be represented by a graph G with a set of vertices $V = \{v_1, \ldots, v_N\}$; spins σ_i and σ_j on vertices v_i and v_j are connected by couplings J_{ij}. Here the computation works by first mapping a quadratic binary optimization problem to an Ising problem and then finding the ground state of the Hamiltonian [7,8]. Most such Ising machines mimic the behavior of an Ising model by using quantum mechanics or some algorithm [9–15]. A quantum annealer, which is a device to perform quantum annealing [16], has been studied theoretically and experimentally. It has been shown that a quantum annealer has better solution performance than conventional algorithms in a class of nonconvex optimization problems [17,18]. Other Ising machines also have attracted interest regarding computational time. However, to date they are not able to simulate large Ising models; for example, the number of spins in quantum annealers [9] and optical network systems [10] is currently limited to about 2000. These size limitations make it hard to use them to solve real-world problems.

Despite the above problems, state-of-the-art digital computers running heuristic algorithms and having huge memories can potentially solve large problems. Here simulated annealing (SA) has been shown to find solutions for a wide range of optimization problems [19,20]. SA is a probabilistic approximate algorithm for finding a state such that a given function $f(\boldsymbol{x})$ is minimized [21]. It consists of discrete-time Markov processes that converge to a Boltzmann distribution. The existence probability of state $\boldsymbol{x}$ is proportional to $\exp[-f(\boldsymbol{x})/T]$, where T is a parameter called temperature. Sampling from the distribution at low temperature attains an optimal or near-optimal solution with high probability. Convergence is achieved by making stochastic transitions based on a Markov chain Monte Carlo algorithm. Typically, a random Ising problem with Hamiltonian (1) is solved through single spin updates. Although spins that are not connected to any others can be independently and simultaneously updated, simultaneous updates cannot be made under any other conditions [22]; that is, SA can only flip single spins for all-to-all connected random Ising models. This requirement is a bottleneck that makes it hard to shorten the computing time, because it means we cannot perform parallel processing on the spins. Meanwhile we can parallelize spin updates of a sparse Ising model. The factor of speed-up by the parallelization depends on the chromatic number of G. In this computing flow, we must solve a graph coloring problem before Monte Carlo simulations, but it is an NP-hard. It is theoretically difficult even to obtain an approximate solution [23], and it actually takes some time. If we know that the Ising model to be solved is on the sparse graph with a small chromatic number such as a grid or chimera graph, we can avoid the coloring process, and the parallelization is effective [24]. However, such a situation is generally not to be expected.

In this paper, we propose a method called momentum annealing (MA), which enables simultaneous spin updates of any Ising model. Our algorithm performs Monte Carlo simulations of Ising models on a trivial two-colorable graph to find the ground state of any Ising model. The advantages of MA are that it can be executed using parallel processing, and thus we implemented MA on GPUs. In fact, we have found that, compared with an SA program optimized for running on multiple processors, a GPU implementation of

*takuya.okuyama.mn@hitachi.com

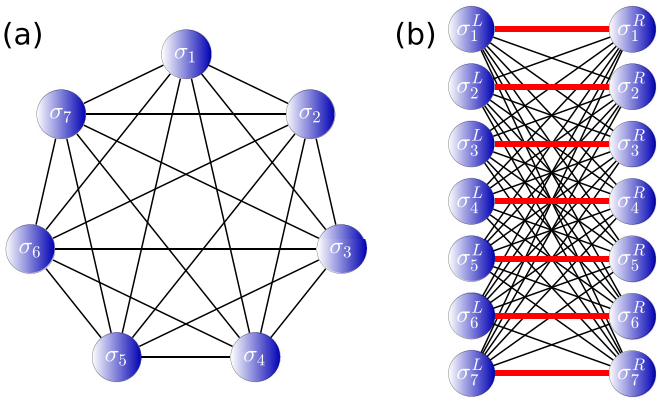


FIG. 1. Conversion of Ising model into bipartite graph. (a) A fully connected Ising model with $N = 7$. (b) An Ising model on a complete bipartite graph G'. If the values of red (bold) couplings are sufficiently large, the spin configurations on the left match those on the right in the ground state.

our algorithm is 247 times faster at finding an approximate solution to a problem mapped to a fully connected Ising model with 100 000 spins.

II. THEORETICAL BACKGROUND

A. Mapping to a bipartite graph

Figure 1 shows the conversion of an Ising model into a representation on a bipartite graph. In Fig. 1(b),the left and right spins of the Ising model on the bipartite graph G' are denoted as $\sigma_1^{\mathrm{L}}, \ldots, \sigma_N^{\mathrm{L}}$ and $\sigma_1^{\mathrm{R}}, \ldots, \sigma_N^{\mathrm{R}}$, respectively. The coupling between spin σ_i^{L} and σ_j^{R} is set to $J_{ij}\,(= J_{ji})$, and the coupling between spins σ_i^{L} and σ_i^{R} is denoted as w_i. The Hamiltonian on G' is expressed as follows:

$$H' = -\sum_{i,j} J_{ij}\sigma_i^{\mathrm{L}}\sigma_j^{\mathrm{R}} - \sum_i h_i\left(\sigma_i^{\mathrm{L}} + \sigma_i^{\mathrm{R}}\right) - \sum_i w_i\sigma_i^{\mathrm{L}}\sigma_i^{\mathrm{R}}. \tag{2}$$

If $w_1, \ldots, w_N$ are sufficiently large, the spin configurations on the left and right side become equivalent in the ground state [25]. Denoting the values of $\sigma_1^{\mathrm{L}}, \ldots, \sigma_N^{\mathrm{L}}$ in the ground state by $b_1, \ldots, b_N$, we can express the Ising energy as $H_{\min} = -2\sum_{i<j} J_{ij}b_ib_j - 2\sum_i h_ib_i - \sum_i w_i$. Because the third term on the right side is a constant independent of b_i, the values $b_1, \ldots, b_N$ can be regarded as the spin configuration that minimizes Eq. (1). Thus, in the ground state, the spin configuration on the left (or right) side of Fig. 1(b) also minimizes the Ising energy of Fig. 1(a). Updating all spins on each side in parallel is possible. We expect that this will shorten the computational time.

Similar to SA, our algorithm explores spin configuration in the ground state by using the Metropolis algorithm [26]. Here we consider the case of updating a spin on the left side of the Ising model on G'. The transition probability depends on the energy change in the state transition. If the ith spin flips, energy increases by $\Delta := 2\sigma_i^{\mathrm{L}}(h_i + \sum_{j=1}^N J_{ij}\sigma_j^{\mathrm{R}})$. Since the Metropolis algorithm accepts the transition with a probability of $\min\{1, \exp(-\Delta/T)\}$, a flip occurs if and only if $\Delta \leqslant 0$ or $u \leqslant \exp(-\Delta/T)$, where u is a uniformly distributed random variable between 0 and 1. This condition is equivalent to $\Delta \leqslant T\gamma_i$, where γ_i is a gamma random variable with shape and scale parameters of value one. When we update only the ith spin, the condition that the spin is 1 through the update is expressed as follows:

$$\begin{aligned}
&\Delta > T\gamma_i \Leftrightarrow h_i + \sum_{j=1}^N J_{ij}\sigma_j^{\mathrm{R}} > T\gamma_i/2 \quad (\sigma_i^{\mathrm{L}} = 1)\\
&\Delta \leqslant T\gamma_i \Leftrightarrow h_i + \sum_{j=1}^N J_{ij}\sigma_j^{\mathrm{R}} \geqslant -T\gamma_i/2 \quad (\sigma_i^{\mathrm{L}} = -1).
\end{aligned} \tag{3}$$

After the update, a new value of the spin becomes 1 if and only if $h_i + \sum_{j=1}^N J_{ij}\sigma_j^{\mathrm{R}} \geqslant \sigma_i^{\mathrm{L}}T\gamma_i/2$ and is computed by $\sigma_i^{\mathrm{L}} \leftarrow \mathrm{sgn}(h_i + \sum_{j=1}^N J_{ij}\sigma_j^{\mathrm{R}} - \sigma_i^{\mathrm{L}}T\gamma_i/2)$. The function $\mathrm{sgn}(\boldsymbol{x})$ returns $\boldsymbol{y}$ such that the ith element of $\boldsymbol{y}$ is 1 if the ith element of $\boldsymbol{x}$ is positive and is -1 otherwise. To simulate the Markov process of spins, we update all spins on the left sequentially. Each calculation depends on the spin on the right and not on the left, and it means that simultaneous flips of all spins on the left are allowed. Therefore, we can update all spins on the left side by using Eq. (4) with $\alpha = \mathrm{L}$ and $\bar{\alpha} = \mathrm{R}$:

$$\begin{aligned}
\begin{bmatrix}\sigma_1^{\alpha}\\ \vdots \\ \sigma_N^{\alpha}\end{bmatrix} \leftarrow \mathrm{sgn}\Bigg(\boldsymbol{h} + \Bigg(J + \begin{bmatrix} w_1 & & O \\ & \ddots & \\ O & & w_N\end{bmatrix}\Bigg)\begin{bmatrix}\sigma_1^{\bar{\alpha}}\\ \vdots \\ \sigma_N^{\bar{\alpha}}\end{bmatrix}\\
- \frac{T}{2}\begin{bmatrix}\gamma_1 & & O \\ & \ddots & \\ O & & \gamma_N\end{bmatrix}\begin{bmatrix}\sigma_1^{\alpha}\\ \vdots \\ \sigma_N^{\alpha}\end{bmatrix}\Bigg).
\end{aligned} \tag{4}$$

Here $J = (J_{ij})_{N\times N}$ is the adjacency matrix of G. Due to the symmetry of G', an update rule of spins on the right side can be written as Eq. (4) with $\alpha = \mathrm{R}$ and $\bar{\alpha} = \mathrm{L}$. By repeating these calculations alternately while decreasing temperature T, we can execute annealing for the Ising model on G'.

Moreover, we can consider the stochastic time evolution of Ising spins from another point of view. Let k be the index of the above alternate iteration ($k \geqslant 1$) and $\boldsymbol{s}_k$ be the spin values obtained at the kth step. Here we start the update from the spins on the left side. Then $\boldsymbol{s}_k$ represents values after $(k+1)/2$ updates in the left spins if k is odd, and values after $k/2$ updates in the right otherwise. We refer the initial spin values on the left and right sides as $\boldsymbol{s}_{-1}$ and $\boldsymbol{s}_0$, respectively. Then we can rewrite the update rule as follows:

$$\boldsymbol{s}_k = \mathrm{sgn}\left(\boldsymbol{h} + [J + \mathrm{diag}(w_1, \ldots, w_N)]\boldsymbol{s}_{k-1} - \frac{T_k}{2}\Gamma_k\boldsymbol{s}_{k-2}\right). \tag{5}$$

Here T_k and Γ_k are the temperature and a diagonal matrix of γ_i at the kth step, respectively. This means the update rule of MA to be a second-order Markov chain of the original Ising model. The proposed Markov chain originates from the single-spin flip Monte Carlo simulation. MA transits states with the balance condition and ergodicity, and thus a sufficiently slow annealing schedule allows spin configuration to reach the ground state [27].

B. Strength of momentum couplings

How should we determine the values of w_i so that the spin configurations on both sides of G' match in the ground state? Too large a value of w_i will make it difficult to obtain a good solution, because the spin configuration will get stuck in a local minimum immediately. To avoid this problem, it is better for the values of w_i to be as small as possible. Here we have proved that the following equality satisfies the condition for the matching of spin values in the ground state (see the Supplemental Material [28] for the proof):

$$w_i = \begin{cases} \sum_{v_j \in V} |J_{ij}| - \frac{1}{2} \sum_{v_j \in C} |J_{ij}| & (v_i \in C) \\ \frac{\lambda}{2} & (v_i \notin C) \end{cases}, \tag{6}$$

where λ is the largest eigenvalue of $-J$, and C is an arbitrary subset of V. In particular, we choose $C = \{v_i \,|\, \lambda \geqslant \sum_{v_j \in V} |J_{ij}|\}$. Without Eq. (6), we must make w_i greater than $\sum_{v_j \in V} |J_{ij}|$, but in that case, the spin configuration would not change during single-spin flip annealing [25]. For example, we consider the all-to-all connected Ising model whose couplings are chosen from $\{-1, 1\}$ with equal probability, i.e., $\mathbb{P}(J_{ij} = 1) = \mathbb{P}(J_{ij} = -1) = 1/2$. According to the random matrix theory, the largest eigenvalue of J is close to $2\sqrt{N}$ with a high probability as N becomes large [29], which means $w_i = O(\sqrt{N})$. Meanwhile, the lower bound of w_i obtained with the current inequality is $N - 1$. Equation (6) allows to decrease the values of w_i. It is because of this equation that our algorithm works properly as a Monte Carlo heuristics based on thermal annealing process.

The coupling w_i is a momentum effect on the temporal behavior of the spin σ_i. MA achieves simultaneous spin flips owing to this effect. It is better to decrease the energy barrier between spin configurations for improving the search efficiency. Here we use two techniques: dropout and momentum scaling. Our dropout is inspired by the method for improving generalization performance of neural network [30]. In dropout, we choose several couplings w_i with a certain probability and treat their strengths as zero temporarily. The probability decreases to zero gradually. Momentum scaling increases the coupling w_i from a small value to the original one computed by Eq. (6). Although the momentum couplings become small by this equation, they might still be larger than other couplings. If some of the momentum couplings have larger strengths than other couplings, dropout and momentum scaling relieve the inertia effects of the coupling. As annealing proceeds, all $w_1, \ldots, w_N$ return to their original values.

The MA computation can be summarized as follows. First, we compute the largest eigenvalue of the adjacency matrix and calculate $w_1, \ldots, w_N$ by using Eq. (6). We prepare the current spin configuration and its previous state. In the initialization, each spin is typically chosen from $\{-1, 1\}$ at random. Then we update the spin configuration by repeating the stochastic behavior based on the second-order Markov process with Eq. (5), dropout, and momentum scaling. We can exploit the parallelism of not only spin update but also dropout, momentum scaling, and random number generation. Once the process finishes according to some criteria, we consider the spin configuration at that time to be a solution of the original Ising problem. We show the steps of MA in Algorithm 1.

Algorithm 1. Momentum annealing for Ising model.

```
Calculate w_1, ..., w_N by using Eq. (6)
Initialize spin configurations s_{-1} and s_0
k ← 0
while criteria are not satisfied do
    k ← k + 1
    Update temperature T_k, dropout rate p_k, and momentum scaling factor c_k
    for all i = 1, ..., N do in parallel
        Set w_i to zero with a probability of p_k temporarily
        Decrease w_i to c_k w_i temporarily
        Sample a random variable from the gamma distribution with shape and scale parameters of value one, and set it to the ith diagonal element of Γ_k
        Calculate the ith element of s_k by using Eq. (5)
    end for
end while
Return current spin configuration as a solution
```

III. RESULTS

A. Performance to find the exact solution

We compared MA with SA in terms of their solution performance and computation time to find the ground state. Using a machine-independent graph generator [31], we prepared Ising models with N spins. We chose the value of the coupling J_{ij} from integers between $-2^{M-1} + 1$ and $2^{M-1} - 1$ uniformly at random. The value of N was varied from 20 to 120 for $M = 2$ and from 20 to 90 for $M = 10$. Zero values are excluded from the coupling range only if $M = 2$. For each case, we generated 100 fully connected and 100 sparse Ising models with an edge density of 10%, and obtained the exact solution of all problems using BiqCrunch [32]. On the basis of a preliminary experiment, we used the temperature schedule

$$T_k = \frac{1}{\beta_0 \ln(1 + k)} \quad (k \geqslant 1), \tag{7}$$

where $\beta_0 = 1.0 \times 10^{-1}$ for $M = 2$ and $\beta_0 = 3.0 \times 10^{-4}$ for $M = 10$ (see the Supplemental Material [28] for details of graph generation and parameter determination). To run MA, we set the kth dropout rate p_k and the momentum scaling factor c_k to be

$$p_k = \max\left\{0,\ 0.5 - \frac{k}{2000}\right\},\ c_k = \min\left\{1,\ \sqrt{\frac{k}{1000}}\right\}. \tag{8}$$

These effects gradually disappear as annealing proceeds. After 1000 sweeps, our algorithm simulated the original system. We ran a highly optimized program of SA on an Ubuntu 16.04.4 Linux server with multiple cores of a POWER8 CPU and 512 GB of memory. It was based on a previous study program [33], and we utilized OpenMP to parallelize the energy calculations at each step. While changing the number of threads, we investigated the optimal number of threads. For each case, we performed MA and SA 1000 times, respectively. We used *MCS-to-solution* and *time-to-solution* as a metric

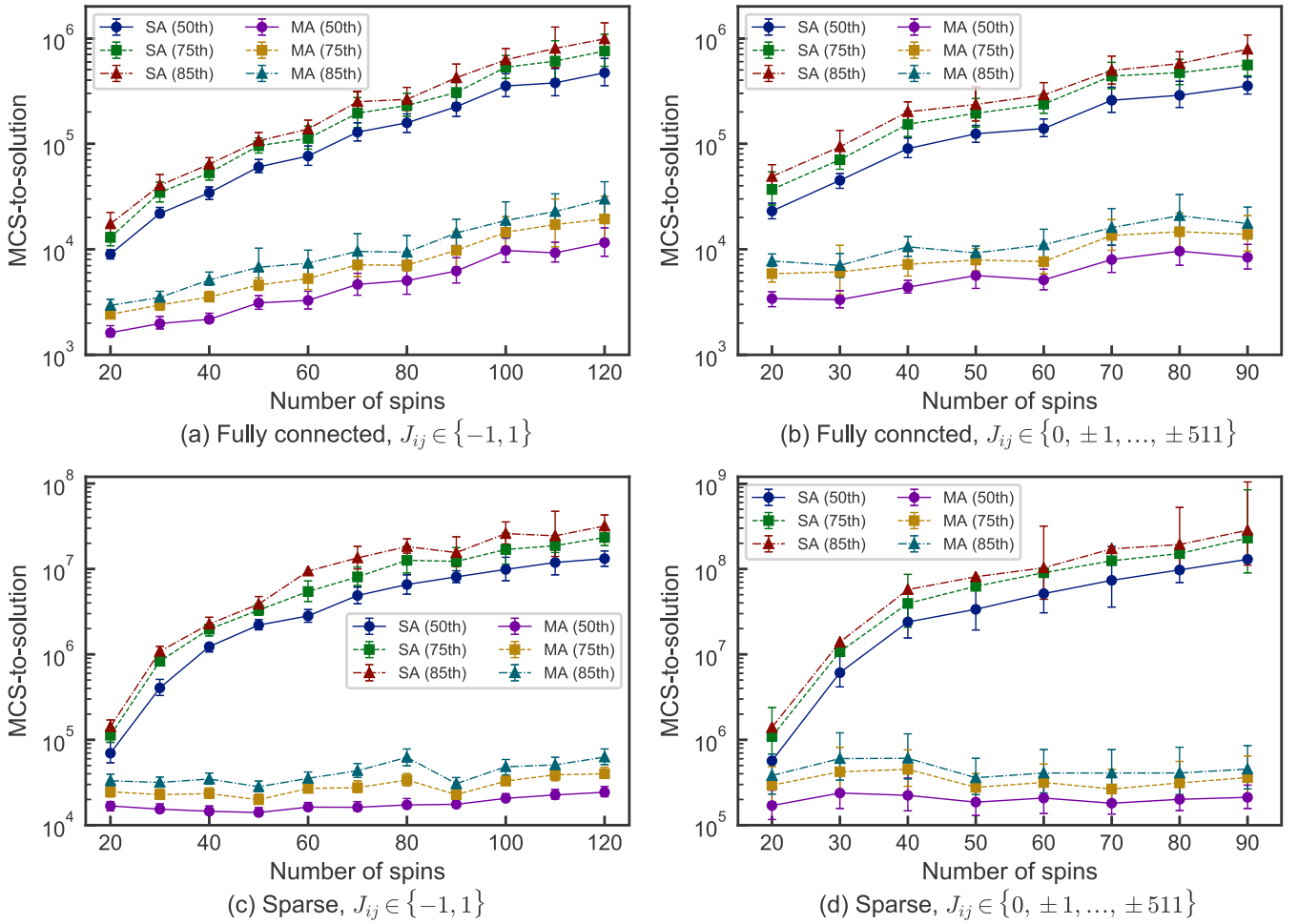


FIG. 2. Scaling of the optimal Monte Carlo steps to find the ground state with a probability of 99%. Shown are the 50th, 75th, and 85th percentiles over a set of 100 instances. The error bars represent 95% confidence intervals.

to evaluate the performance. They are the minimum Monte Carlo steps and computing time to find the ground state with a probability of 99%, respectively. One sweep entailed attempting to flip each spin once. Here MA and SA carried out one sweep in one and N Monte Carlo steps, respectively. We took for granted that MA updates all spins once, because the algorithm pays the cost so as to achieve the parallelization. The MCS-to-solution is calculated as

$$\min_k \left\{ k \left\lceil \frac{\ln(1-0.99)}{\ln(1-q_k)} \right\rceil \right\}, \tag{9}$$

where q_k is a probability that a single annealing with k Monte Carlo steps finds the ground state. Similarly time-to-solution is defined as

$$\min_k \left\{ t_k \left\lceil \frac{\ln(1-0.99)}{\ln(1-q_k)} \right\rceil \right\}, \tag{10}$$

where t_k represents the time that annealing takes to carry out k Monte Carlo steps [34]. MCS-to-solution and time-to-solution show the performance to find the ground state from the viewpoint of calculation amount and wall-clock time, respectively.

Figure 2 compares the MCS-to-solution of both annealing methods. Were the number of sweeps to increase drastically as a result of our momentum couplings, the parallel update of MA would not decrease the Monte Carlo steps enough to be effective. Fortunately, the results show that MA can find the ground states and has a lower scaling of MCS-to-solution than SA for any cases. This relative reduction is just what we expected.

For the parallel processing, we implemented MA on a GPU (NVIDIA Tesla P100). The implementation provides couplings with single-precision floating point numbers. Figure 3 plots the time-to-solution as a function of N. When the number of spins is small, MA takes a longer time than SA. However, since MA has a lower increase rate of MCS-to-solution and time-to-solution than SA especially for fully connected Ising models, we can expect that MA will be faster than SA when the number of spins is more enormous than hundreds of spins.

B. Performance to find an approximated solution

1. Comparison of SG and GW-SDP

We investigate the computational performance of our algorithm for large-scale instances. Before the evaluation, we discuss about some algorithms to solve the quadratic binary optimization problem. To the best of our knowledge, no

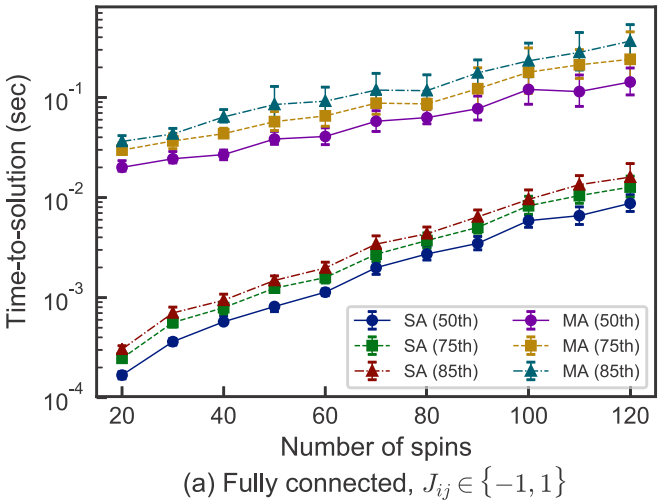


FIG. 3. Scaling of optimal annealing time to find the ground state with a probability of 99%. Shown are the 50th, 75th, and 85th percentiles over a set of 100 instances. The error bars represent 95% confidence intervals. SA was run on a single thread.

algorithm can find the ground state of an Ising model exactly in a practical amount of time when the number of spins is greater than hundreds of spins. It is thus reasonable to use instead an approximate algorithm for benchmarking. One of the well-known methods is the Goemans-Williamson SDP (GW-SDP) algorithm, which solves the maximum-cut problem with a guarantee of solution quality [35]. This algorithm relaxes the original problem to one of semidefinite programming (SDP). However, it is difficult to use SDP to solve large problems, because of its long computation time. Therefore, we used the Sahni-Gonzales (SG) algorithm for large instances [36].

SG is a greedy heuristic, and its solution quality is not bounded theoretically. To confirm the performance of SG, we compared the solution qualities obtained with SG and GW-SDP. Using the graph generator, we prepared Ising models with N spins and M bit widths ($N = 500, 2000, 5000$ and $M = 2, 4, 6, 8, 10$). For each case, we generated 10 different Ising models with different seeds. In GW-SDP, we took a random hyperplane through the origin 1000 times and chose the best solution. Here H_{SG} and H_{GW} denote Ising energies

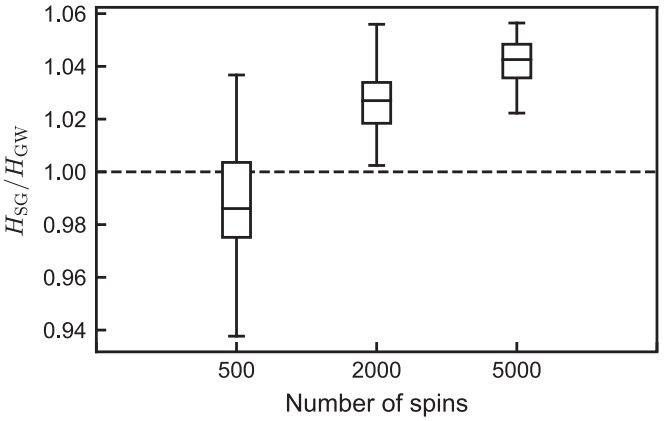


FIG. 4. Comparison of Ising energies obtained with SG and GW-SDP. For each number of spins, 50 different instances are investigated. Each box plot indicates interquartile range (each horizontal line in the box is the median).

obtained with SG and GW-SDP, respectively. Figure 4 plots the distribution of $H_{\mathrm{SG}}/H_{\mathrm{GW}}$. The ratios are nearly one for each case. SG thus has a solution performance comparable to GW-SDP. Therefore, we conclude that it is reasonable to use it to obtain the reference values.

2. *Fully connected Ising model*

We compared MA with SA in terms of their solution performance and computation time on fully connected Ising models. Using the graph generator, we prepared Ising models with $N = 500, 2000, 10\,000, 50\,000, 100\,000$ and $M = 2, 4, 6, 8, 10$. We used the same annealing schedule as Eqs. (7) and (8) with $\beta_0 = 1.0 \times 10^{-1}, 2.5 \times 10^{-2}, 6.0 \times 10^{-3}, 1.3 \times 10^{-3}$, and 3.0×10^{-4} for each bit width M. For each case, we performed MA and SA 100 times, respectively.

Figure 5(a) compares the number of sweeps that SA and MA take to reach the Ising energy obtained with SG. The results show that MA kept the number of Monte Carlo steps almost constant, whereas SA increased the number of steps on the order $O(N^{1.0})$. For massively parallel processing, we implemented MA on GPUs (NVIDIA Tesla P100 × 4 connected to a POWER8 CPU by NVLink and 64 GB memory in total). Figure 5(b) plots the computational time as a function of N. By increasing the number of GPUs, we can attempt to flip more spins at once. However, communications between GPUs take time. Here MA with an optimal number of GPUs and SA on an optimal threads CPU took $O(N^{0.9})$ and $O(N^{2.0})$ time, respectively. The computational time of MA includes the communication time. Thus, despite the need for communications, our implementation of MA on GPUs is superior in terms of computation time to SA on a CPU.

Figure 6(a) plots temporal energy curves when using MA and SA to solve the fully connected Ising model with 100 000 spins and 10 bit-width couplings. Here both methods performed 2000 sweeps. To reach the Ising energy obtained with SG, MA, and SA took 0.16 and 38.90 sec, respectively. Namely, MA was 247 times faster than SA at reaching an approximate solution. Figure 6(b) shows a histogram of

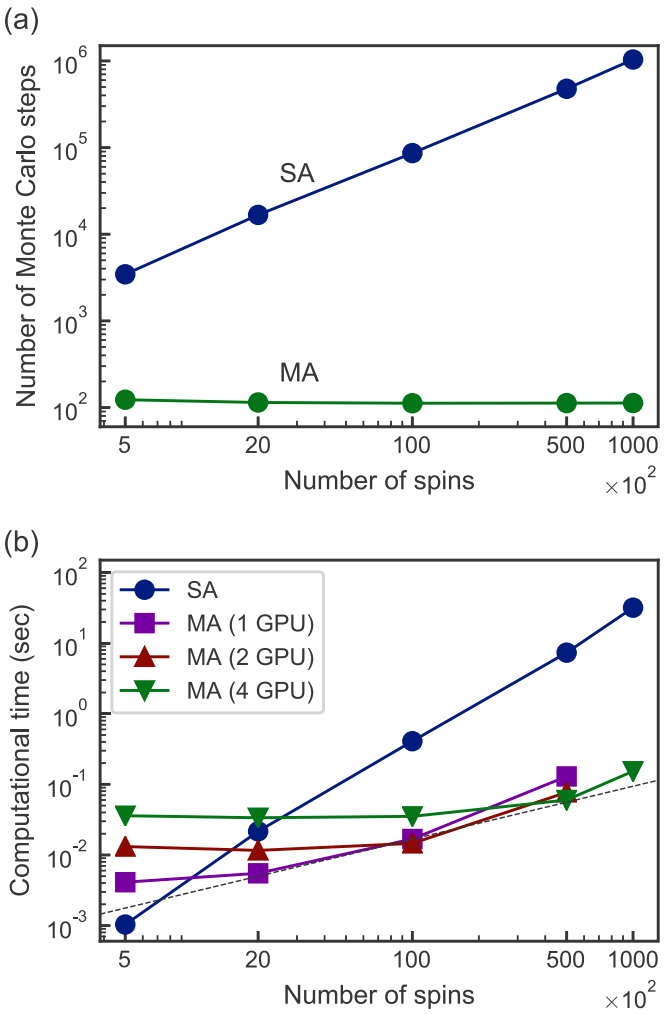


FIG. 5. Scaling of time to reach the Ising energy obtained by SG. For each number of spins, five types of all-to-all connected Ising models were tested ($M = 2$, 4, 6, 8, 10). For each case, we performed MA and SA 100 times, respectively. Dots show the average time of the five cases. (a) We assumed that MA and SA take one and N Monte Carlo steps for one sweep, respectively. (b) The minimum computational time for $N = 2000$, 5000, and 10 000 was found for 1, 2, and 4 GPUs, respectively. From these results we extrapolated the dashed line, which shows the optimal scaling of the computational time of MA. The optimal number of threads in the SA for fully connected Ising models was 1 for 500 and 2000 spins, 32 for 10 000 spins, 60 for 50 000 spins, and 40 for 100 000 spins.

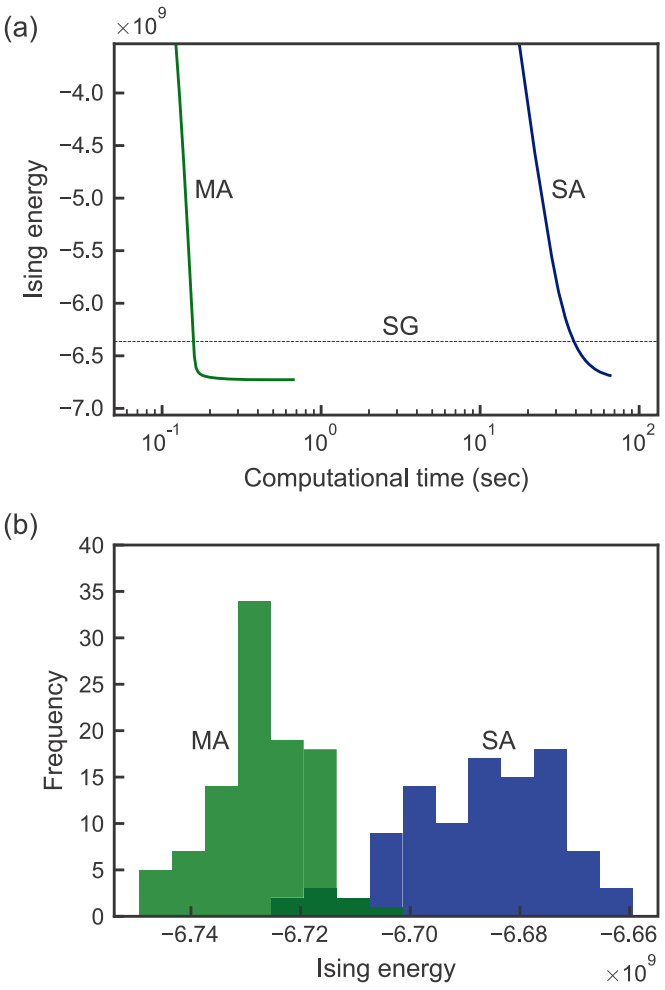


FIG. 6. Experimental results obtained with MA and SA solving fully connected Ising model with 100 000 spins and 10 bit widths. (a) Temporal curve of Ising energy. The dashed line corresponds to the target Ising energy of −6 363 591 595 obtained with SG. (b) Histograms of Ising energy after both methods performed 2000 sweeps.

Ising energies after the annealing. For the same number of sweeps, on average, MA reached a less energetic state than SA did.

3. Sparse Ising model

Next, we compared the solution performance and computation time when solving Ising models on a sparse graph. In this experiment, we solved the maximum-cut problem, which is equivalent to finding the ground state of the Ising model [37]. We tested three different maximum-cut problems: G61, G81, and torusg3-15. G61 and G81 are random and toroidal grid graphs taken from the G-set [38] whose edge weights are limited to $\{-1, 1\}$. Torusg3-15 is taken from the DIMACS library [39]. Its edge weights are Gaussian distributed. MA was implemented with a single GPU, because it does not require a large memory for sparsity. The optimal β_0 of the fully connected Ising models was inversely proportional to the average value of $|J_{ij}|$. From this observation, we used the temperature schedule by Eq. (7) with $\beta_0 = 1.0 \times 10^{-1}$ for G61 and G81, and $\beta_0 = 1.0 \times 10^{-6}$ for torusg3-15.

The results are summarized in Fig. 7. The listed times are those to reach the cut value obtained with SG. The upper and lower values in each row are those obtained with SA and MA, respectively. For each instance, MA reached the target cut value in 0.01 sec, whereas SA took 20–800 times longer. The histograms are cut values after annealing for 1 sec. The average cut values with MA are better than those with SA. These results suggest that MA on a GPU outperforms SA on a modern CPU at solving not only dense- but also sparse-graph problems.

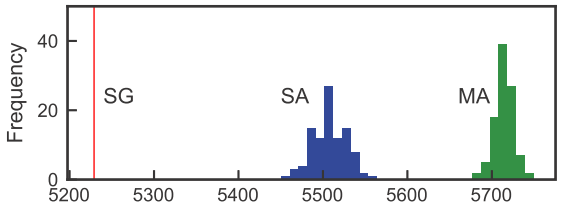
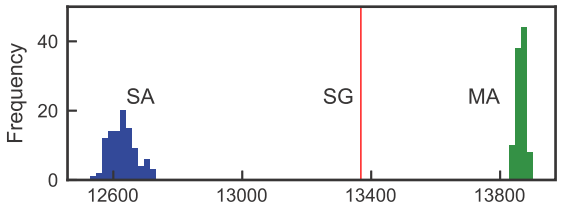
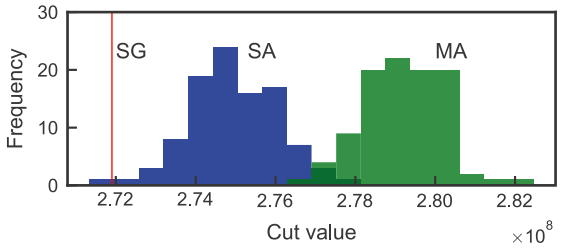
Dataset	Problem |V|	Problem |E|	Problem Edge weight	Results Time (sec)	Results Histogram
G61	6957	17148	{-1, 1}	0.164 0.007	
G81	20000	40000	{-1, 1}	10.364 0.013	
torus g3-15	3375	10125	[-372602 ,375700]	0.574 0.013	

FIG. 7. Experimental results of maximum-cut problem obtained with MA and SA. We tested three sparse instances: G61, G81, and torusg3-15. Here $|V|$ and $|E|$ represent the number of vertices and nonzero edge weights, respectively. Vertical lines correspond to cut values obtained with SG. The histograms show cut values obtained after annealing for 1 s. In all cases, single-threaded SAs were faster than multithreaded SAs. In contradiction to Fig. 6(b), it is better for the distribution to be located on the right side because we solve the maximum problem.

IV. DISCUSSION

A. Eigenvalue computation

Let us examine the total computing time. The eigenvalue computation is required once in MA. Since MA calculates only the largest eigenvalue of $-J$, we used the shifted power method [40]. All eigenvalues of $K := -J + \mu I$, where I is an identity matrix and μ is a constant, are greater than those of $-J$ by μ. If μ is sufficiently large, the largest eigenvalue and the absolute largest eigenvalue of K are the same. Since K is a real and symmetric matrix, we can obtain the absolute largest eigenvalue of it by using the power method. Hence, we compute the largest eigenvalue of $-J$ by subtracting μ from the absolute largest eigenvalue of K. This method performs a matrix-vector multiplication repeatedly, and a GPU is a suitable architecture for it. In addition, an approximate eigenvalue computation is sufficient for MA.

How should we determine the value of μ so that the largest eigenvalue and the absolute largest eigenvalue of K are the same? If $\mu \geqslant \max_i \{\sum_{j\neq i} |J_{ij}|\}$, it satisfies the condition above because K is diagonally dominant, and it means K is positive semidefinite. However, it will be difficult to obtain the largest eigenvalue rapidly by the power method due to its slow convergence. In this paper, we chose $\mu = \max_i \{\sum_{j\neq i} |J_{ij}|\}/100$ and perform the power method. If the value ξ obtained by the calculation is non-negative, it approximates the largest eigenvalue of K. Otherwise, it does the smallest eigenvalue of K, and thus we increase μ by $-\xi$ and restart the power method. The recalculation always returns the largest eigenvalue of K.

Now let us investigate the whole calculation time of the shifted power method. Figure 8 shows a temporal curve of the estimated largest eigenvalue when we calculate the eigenvalues of the adjacency matrix of the Ising model with $N = 100\,000$ and $M = 10$. From this result, we conclude that

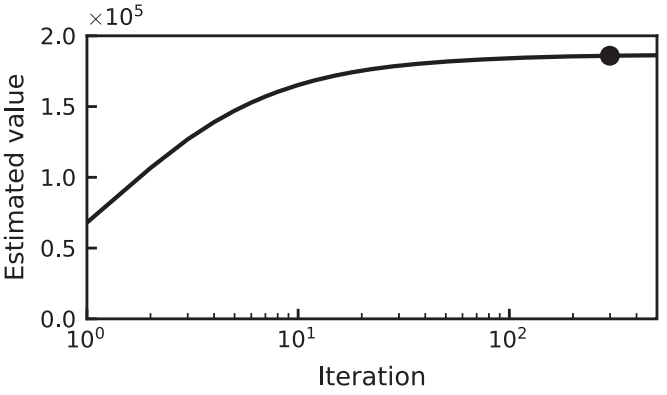


FIG. 8. Convergence of largest eigenvalue by the power method. We show an example when calculating the largest eigenvalue of the adjacency matrix on an Ising model with $N = 100\,000$ and $M = 10$. The dot shows the result at 300 iterations.

it is sufficient to iterate the shifted power method 300 times. This calculation takes 6.8 sec on the four GPUs, and it is required only once at the beginning of MA even though the annealing is performed dozens of times. As shown in Fig. 6(a), single annealing of MA takes about 0.1–1.0 sec. Therefore, the eigenvalue computation is not the most time-consuming part in MA.

B. Applicability to MCMC methods

The proposed algorithm is an extension of SA. What bothers SA is that it is difficult to find a path to the ground state in the problem where the number of local minima increases exponentially with the number of variables. To overcome the problem, an optimization algorithm via MCMC sampling has been widely proposed besides SA. Parallel tempering [41,42] improves the solution performance by promoting convergence to the Boltzmann distribution. It simulates independent Markov chains in which a temperature is constant and exchanges them stochastically at a specified interval. We can perform the independent runs by using Eq. (5) with fixed temperatures. Another improved algorithm is simulated quantum annealing by a path integral Monte Carlo method (SQA), which is a quantum-inspired algorithm that simulates the quantum tunneling phenomena of an Ising model with a transverse field. It has been reported that SQA has better scalability of computational time than SA [17,43]. The SQA for Ising model can be achieved by MCMC for the Ising model one-dimensional higher. Also, efficient nonlocal update algorithms for spin glasses have been developed [44,45]. We can execute MA with these techniques, and it will increase the success probability to find the ground state.

The Boltzmann sampling is an MCMC method to carry out a statistical evaluation of an Ising model and is studied for machine learning [46–48]. It samples a spin configuration from equilibrium state at a finite temperature. Our algorithm focuses on the ground state search, which can be regarded as sampling from the equilibrium state at zero temperature. To apply MA to the Boltzmann sampling, we have to evaluate the error incurred by the mapping of the Ising model with Eq. (6). Not only combinatorial optimization but also statistical evaluation of the system is promising due to its broad applications, and the analysis is open problem.

Finally, we refer to the effect of dropout and momentum scaling. In a preliminary experiment, we confirmed that a spin configuration gets stuck in a local minimum easily, and it is hard to reach the ground state without these techniques. Further study of the techniques is also future work.

V. CONCLUSION

We have presented an MCMC-based annealing algorithm for finding the ground state of Ising models. Our proposed algorithm, which we call momentum annealing (MA), enables us to update all spins simultaneously even if all spins are fully connected. We have demonstrated that MA can obtain the exact solution with lower Monte Carlo steps than SA. When solving a fully connected Ising model with 100 000 spins, we have confirmed that MA running on four NVIDIA Tesla P100 is 250 times faster than SA running on an IBM POWER8 CPU at reaching the approximate solution obtained with Sahni-Gonzales algorithm. Furthermore, we have shown that MA is faster than SA for not only dense- but also sparse-graph instances. In this paper, we have tested the Ising problem with up to 100 000 spins due to the memory size. MA on parallel computers with larger memory will be able to solve Ising problems with more spins while keeping scalability of computational time.

[1] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature (London) **435**, 814 (2005).

[2] A. Javanmard, A. Montanari, and F. Ricci-Tersenghi, Proc. Natl. Acad. Sci. USA **113**, E2218 (2016).

[3] A. Braunstein, L. Dall'Asta, G. Semerjian, and L. Zdeborová, Proc. Natl. Acad. Sci. USA **113**, 12368 (2016).

[4] F. T. Leighton, *Complexity Issues in VLSI: Optimal Layouts for the Shuffle-Exchange Graph and Other Networks* (MIT Press, Cambridge, MA, 1983).

[5] H. Neven, V. S. Denchev, M. Drew-Brook, J. Zhang, W. G. Macready, and G. Rose, in *Demonstrations at NIPS-09, 24th Annual Conference on Neural Information Processing Systems* (2009), pp. 1–17.

[6] F. Barahona, J. Phys. A **15**, 3241 (1982).

[7] A. Lucas, Front. Phys. **2**, 5 (2014).

[8] T. Albash and D. A. Lidar, Rev. Mod. Phys. **90**, 015002 (2018).

[9] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk *et al.*, Nature (London) **473**, 194 (2011).

[10] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, O. Tadanaga, H. Takenouchi, K. Aihara, K. Kawarabayashi, K. Inoue, S. Utsunomiya, and H. Takesue, Science **354**, 603 (2016).

[11] M. Steffen, W. van Dam, T. Hogg, G. Breyta, and I. Chuang, Phys. Rev. Lett. **90**, 067903 (2003).

[12] H. Suzuki, J. Imura, Y. Horio, and K. Aihara, Sci. Rep. **3**, 1610 (2013).

[13] I. Mahboob, H. Okamoto, and H. Yamaguchi, Sci. Adv. **2**, e1600236 (2016).

[14] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, IEEE J. Solid-State Circuits **51**, 303 (2016).

[15] T. Okuyama, M. Hayashi, and M. Yamaoka, in *2017 IEEE International Conference on Rebooting Computing (ICRC)* (IEEE, Piscataway, NJ, 2017), pp. 1–6.

[16] T. Kadowaki and H. Nishimori, Phys. Rev. E **58**, 5355 (1998).

[17] S. Mandrà, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, Phys. Rev. A **94**, 022337 (2016).

[18] C. Baldassi and R. Zecchina, Proc. Natl. Acad. Sci. USA **115**, 1457 (2018).

[19] V. Černý, J. Optim. Theory Appl. **45**, 41 (1985).

[20] M. Rosvall and C. T. Bergstrom, Proc. Natl. Acad. Sci. USA **104**, 7327 (2007).

[21] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Science **220**, 671 (1983).
[22] D. P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, 4th ed. (Cambridge University Press, Cambridge, 2014).
[23] D. Zuckerman, Theor. Comput. **3**, 103 (2007).
[24] T. Preis, P. Virnau, W. Paul, and J. J. Schneider, J. Comput. Phys. **228**, 4468 (2009).
[25] V. Choi, Quantum Inf. Process **7**, 193 (2008).
[26] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).
[27] B. Hajek, Math. Operations Res. **13**, 311 (1988).
[28] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevE.100.012111 for details of graph generation and parameter determination.
[29] X. Ding and T. Jiang, Ann. Appl. Probab. **20**, 2086 (2010).
[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, J. Mach. Learn. Res. **15**, 1929 (2014).
[31] G. Rinaldy, "Rudy", https://www-user.tu-chemnitz.de/~helmberg/rudy.tar.gz, (1996).
[32] N. Krislock, J. Malick, and F. Roupin, ACM Trans. Math. Softw. **43**, 1 (2017).
[33] S. Isakov, I. Zintchenko, T. Rønnow, and M. Troyer, Comput. Phys. Commun. **192**, 265 (2015).
[34] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, Science **345**, 420 (2014).
[35] M. X. Goemans and D. P. Williamson, J. ACM **42**, 1115 (1995).
[36] S. Kahruman, E. Kolotoglu, S. Butenko, and I. V. Hicks, Int. J. Comput. Mater. Sci. Eng. **3**, 211 (2007).
[37] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, Oper. Res. **36**, 493 (1988).
[38] C. Helmberg and F. Rendl, SIAM J. Optim. **10**, 673 (2000).
[39] S. Schmieta, The DIMACS library of mixed semidefinite-quadratic-linear programs, http://dimacs.rutgers.edu/archive/Challenges/Seventh/Instances/lib.tar.gz, (2000).
[40] Y. Saad, *Numerical Methods for Large Eigenvalue Problems* (Society for Industrial and Applied Mathematics, 2011).
[41] R. H. Swendsen and J.-S. Wang, Phys. Rev. Lett. **57**, 2607 (1986).
[42] K. Hukushima and K. Nemoto, J. Phys. Soc. Jpn. **65**, 1604 (1996).
[43] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, Phys. Rev. X **6**, 031015 (2016).
[44] J.-S. Wang, Phys. Rev. E **72**, 036706 (2005).
[45] Z. Zhu, A. J. Ochoa, and H. G. Katzgraber, Phys. Rev. Lett. **115**, 077201 (2015).
[46] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, Phys. Rev. X **8**, 021050 (2018).
[47] H. Sakaguchi, K. Ogata, T. Isomura, S. Utsunomiya, Y. Yamamoto, and K. Aihara, Entropy **18**, 365 (2016).
[48] S. Yamanaka, M. Ohzeki, and A. Decelle, J. Phys. Soc. Jpn. **84**, 024801 (2015).