scientific reports

Check for updates

OPEN

Enhanced convergence in p-bit based simulated annealing with partial deactivation for large-scale combinatorial optimization problems

Naoya Onizawa ✉ & Takahiro Hanyu

This article critically investigates the limitations of the simulated annealing algorithm using probabilistic bits (pSA) in solving large-scale combinatorial optimization problems. The study begins with an in-depth analysis of the pSA process, focusing on the issues resulting from unexpected oscillations among p-bits. These oscillations hinder the energy reduction of the Ising model and thus obstruct the successful execution of pSA in complex tasks. Through detailed simulations, we unravel the root cause of this energy stagnation, identifying the feedback mechanism inherent to the pSA operation as the primary contributor to these disruptive oscillations. To address this challenge, we propose two novel algorithms, time average pSA (TApSA) and stalled pSA (SpSA). These algorithms are designed based on partial deactivation of p-bits and are thoroughly tested using Python simulations on maximum cut benchmarks that are typical combinatorial optimization problems. On the 16 benchmarks from 800 to 5000 nodes, the proposed methods improve the normalized cut value from 0.8 to 98.4% on average in comparison with the conventional pSA.

In recent years, a new device model known as the probabilistic bit, or p-bit, has been proposed[1]. Unlike traditional bits which can only exist in a state of 0 or 1, a p-bit can exist in a range of states between 0 and 1, each state has a certain probability of occurring. The p-bit is a versatile computational model that can be implemented in software[2] or emerging probabilistic devices, such as Magnetoresistive Random Access Memory (MRAM)[3]. Furthermore, it can be approximated by digital circuits, such as Field-Programmable Gate Arrays (FPGAs)[4–7]. This probabilistic nature makes p-bits a useful tool in solving certain types of problems that require a degree of randomness or uncertainty. The output state of a p-bit is represented as follows:

$$\sigma_i(t+1) = \mathrm{sgn}\Big(\mathrm{r_i(t)} + \tanh\big(\mathrm{I_i}(t+1)\big)\Big), \tag{1}$$

where $\sigma_i(t+1) \in \{-1, 1\}$ is a binary output signal, $I_i(t+1)$ is a real-valued input signal, and $r_i(t) \in \{-1 : 1\}$ is a random signal. The utilization of p-bits is notably effective in the development of a specific neural network variant, the Boltzmann machine[8]. This model is particularly well-adapted for tasks that require invertible logic[9], where inputs and outputs can be interchanged. Moreover, it has significant applications in Bayesian inference[10], parallel tempering[11], Gibbs sampling[12], and simulated annealing (SA)[13].

SA is a stochastic optimization technique widely used for addressing combinatorial optimization problems[14,15]. Its applications span diverse real-world scenarios, including solving the maximum cut (MAX-CUT) problem in network analysis[16], optimizing communication systems[17], and enhancing various machine learning algorithms[18]. Combinatorial optimization problems can often be represented by Ising models, which are mathematical representations of networks or graphs. These problems are often categorized as NP-hard[19], meaning that the time required to find the optimal solution tends to grow exponentially with the size of the problem, making them computationally challenging to solve. The goal of simulated annealing in this context is to minimize the 'energy' of the Ising model, where 'energy' is a metaphor for the objective or cost function of the optimization problem. The global minimum energy state corresponds to the optimal solution to the combinatorial optimization problem.

Research Institute of Electrical Communication, Tohoku University, Sendai 980-8577, Japan. ✉email: naoya.onizawa.a7@tohoku.ac.jp

Diverse enhancements and adaptations of SA, including parallel tempering and stochastic simulated annealing (SSA), have been devised to improve efficiency in solving combinatorial optimization problems[20,21]. Additionally, hardware implementations of SA have been explored for rapidly addressing large-scale combinatorial optimization challenges[22–24]. More advanced computational methods such as quantum annealing (QA)[25,26] have been developed, with expectations of faster processing times compared to SA[27]. However, despite its potential, the realization of quantum annealing is currently restricted due to limitations in device performance. Large-scale problems remain challenging to solve using quantum annealing methods[28,29]. In addition to simulated annealing, various other algorithms have been developed for solving Ising models. These include coherent Ising machines[30], simulated bifurcation[31], and coupled oscillation networks[32].

Simulated annealing that utilizes p-bits (pSA) is grounded in a probabilistic computing paradigm, which enables its implementation on classical computers. This theoretical framework positions pSA as a potential tool for efficiently solving large-scale problems. A potential advantage of pSA is its ability to update nodes in parallel, as opposed to the serial updating method of traditional SA. This means that multiple nodes in the network can be updated simultaneously rather than one at a time, which could potentially lead to a faster convergence to the global minimum energy state, speeding up the process of finding the optimal solution. Preliminary studies have demonstrated that pSA can effectively solve small-scale problems[13]. On the other hand, the efficacy of pSA appears to diminish as the scale of the problem increases. Simulation studies have indicated that as the size of the problems increases, particularly in the cases of graph isomorphism and MAX-CUT problems, the effectiveness of finding solutions using pSA significantly diminishes[21]. One of the challenges is that the energy of the Ising model does not decrease as expected and remains high, indicating that pSA struggles to find optimal or near-optimal solutions for these larger problems. In pSA, the exact reasons behind this limitation remain unclear and are yet to be understood.

The initial part of this article will involve a detailed analysis of the issues encountered with pSA. This will involve using simulation techniques to study the behavior of the p-bits in detail, with the aim to identify the root cause of the aforementioned problems. The analysis identifies that the failure to lower the energy of the Ising model, which is an issue encountered during the optimization process, stems from oscillations occurring among the p-bits. This issue arises because pSA operates as a feedback system, where the output at one stage becomes the input for the next stage, causing the oscillations to occur and impede the reduction of energy. Based on the insights gained from the analysis, two new pSA algorithms are introduced: time average pSA (TApSA) and stalled pSA (SpSA). These algorithms aim to counteract the oscillations based on partial deactivation of p-bits, thereby overcoming the main issue identified with the current pSA process. These newly proposed algorithms are then put to the test through simulations conducted using Python. The simulations are applied to solve maximum cut (MAX-CUT) problems[33], which are typical examples of combinatorial optimization problems. The simulation results demonstrate that the newly proposed pSA algorithms significantly outperform both the conventional pSA algorithm and traditional SA algorithms This implies that these new algorithms may provide a more effective method for solving combinatorial optimization problems, especially for larger problem sizes where traditional methods struggle.

Methods

Problem identification of pSA

A combinatorial optimization problem is represented using an Ising model that represents an energy. The energy is represented by Hamiltonian that is defined as follows:

$$H(\sigma) = -\sum_i h_i\sigma_i - \sum_{i<j} J_{ij}\sigma_i\sigma_j, \tag{2}$$

where $\sigma_i \in \{-1, 1\}$ is a binary state, h are biases for p-bits, and J are weights between p-bits. Depending on combinatorial optimization problems, different h and J are assigned. Simulated annealing attempts to reach the global minimum energy of Eq. (2) by changing the states σ_i. An algorithm of chancing σ_i is different depending on SA algorithms[13,14,21,23].

pSA[13] is illustrated in Fig. 1. In pSA, each p-bit is biased with h and is connected with other p-bits with weights J. The input of p-bit $I_i(t+1)$ is calculated using the outputs of other p-bits that is defined as follows:

$$I_i(t+1) = I_0\left(h_i + \sum_j J_{ij}\cdot\sigma_j(t)\right), \tag{3}$$

where I_0 is a pseudo inverse temperature used to control the simulated annealing. During the simulated annealing process, I_0 is gradually increased in attempt to lower the energy of the Ising model. When I_0 is small, σ_i can be easily flipped between '− 1' and '+ 1' to search for many possible solutions of the combinatorial optimization problem. When I_0 is large, σ_i can be stabilized in attempt to reach the global minimum energy. σ_i can be found as the solution of the combinatorial optimization problem at the global minimum energy.

Let us explain an issue of pSA using a simulated result of a maximum-cut (MAX-CUT) problem that is a typical combinatorial optimization problem[33] (Fig. 2). The MAX-CUT problem aims to partition a graph into two groups in such a way that the sum of the weights of the edges crossing between the two groups is maximized. This process involves 'cutting' the graph into two separate sections, hence the term 'MAX-CUT'. A five-node MAX-CUT problem with edge weights of − 1 and +1 is illustrated (Fig. 2). The black circle illustrates a spin state of '+ 1', while the white circuit illustrates a spin state of '− 1'. In the graph, the weight associated with each edge, which can be either − 1 or + 1, is symbolized by the variable J. This variable is crucial in the MAX-CUT problem

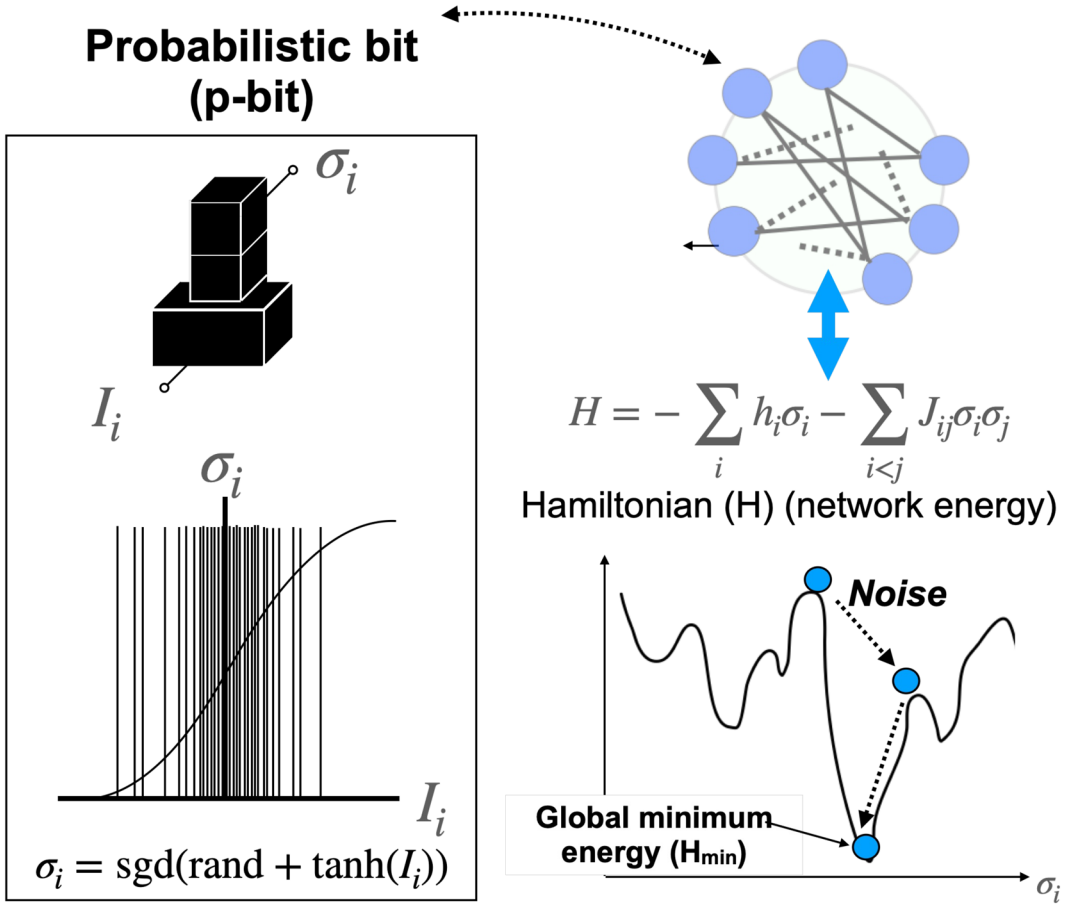

Figure 1. Simulated annealing based on p-bit (pSA). p-bits (left) probabilistically operates based on Eq. (1). A combinatorial optimization problem is represented by an Ising model that corresponds to an energy (Hamiltonian). Based on an Ising model, each p-bit is biased with h and is connected with other p-bits with weights J (right top). During the simulated annealing process, an pseudo inverse temperature I_0 is gradually increased to reach the global minimum energy (H_{min}). pSA attempts to lower the energy of the Ising model by changing the p-bit states σ_i. If the energy reaches the global minimum energy σ_i are a solution of the combinatorial optimization problem (right bottom).

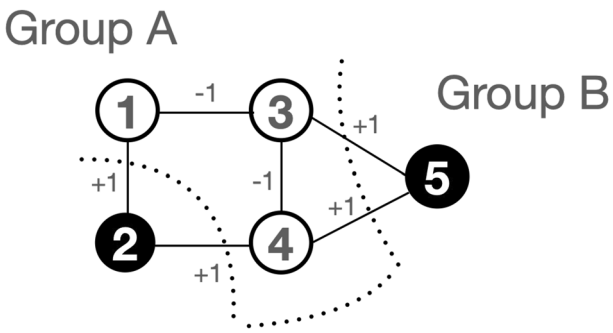

Figure 2. A five-node maximum cut (MAX-CUT) problem with edge weights of −1 and +1. MAX-CUT problem is a typical combinatorial optimization problem. The line cuts the edges to divide the graph into two groups while the sum of the edge weights is maximized. The graph is divided into Group A (nodes 1, 3, and 4) and Group B (nodes 2 and 5), with a sum of edge weights equal to 4.

as it determines the optimal partition of the graph. During the simulated annealing process, the spin states are flipped to lower the energy. The goal is to find the optimal solution, which corresponds to the minimum energy state. After the process, the graph is divided into Group A (nodes 1, 3, and 4) and Group B (nodes 2 and 5), with a sum of edge weights equal to 4.

pSA with the G1 problem is simulated to identify and understand the current issue of pSA. The pSA algorithm is executed using Python 3.11 on Apple M1 Ultra with 128 GB memory. G1 is a specific combinatorial optimization challenge called the MAX-CUT problem from the G-set benchmark[34]. The G1 graph consists of 800 nodes and 19,176 edges that are randomly interconnected. To manage the simulated annealing process of pSA, the pseudo inverse temperature I_0 is gradually increased over time from I_{0min} to I_{0max}, following the formula $I_0(t+1) = I_0(t)/\beta$, where β is 0.995, I_{0min} is 0.0149, and I_{0max}=1.49 (Fig. 3a). A method of determining these hyperparameters will be explained in the last subsection. During the simulated annealing process, all the p-bit states σ_i start to oscillate between '– 1' and '+ 1'. The average value of all the p-bit states is changed between '– 1' and '+ 1' at every cycle (Fig. 3b). An unexpected issue arises due to this oscillation: the energy starts to increase rather than decrease towards a global minimum (Fig. 3c). This suggests that pSA is not reaching the optimal solution, as we would typically expect the energy to minimize in a successful simulated annealing process.

Proposed algorithms based on partial deactivation of p-bits

SA based on time average p-bit (TApSA)

In this article, two new pSA algorithms with nonlinear functions are introduced, which partially deactivates p-bits to mitigate the oscillations. The first algorithm is SA based on time average p-bit (TApSA). The TApSA algorithm draws its inspiration from stochastic simulated annealing (SSA)[21]. SSA approximates the behavior of p-bits using a method called stochastic computing, which is particularly suited for simulated annealing processes. Stochastic computing is a computational approach where values are represented as the frequencies (time averages) of 1s in bit streams[35,36]. This allows for the efficient operation of time series computations in a way that is hardware efficient in terms of physical area usage[37]. The use of stochastic computing has been successfully applied in a range of computational applications, such as in low-density parity-check decoders, image processing, digital filters, and deep neural networks[38–41]. The SSA approach approximates the tanh function (Eq. 1), a key component of the pSA operation, using a saturated updown counter, which results in an operation that calculates tanh in a time series manner. Contrary to pSA, SSA has been shown to be capable of effectively solving large-scale combinatorial optimization problems. This suggests that the new TApSA algorithm, which is inspired by SSA, could potentially address the limitations identified in traditional pSA when dealing with large problem sizes.

Based on the previously discussed explanations, the time average operation is a key element to solve the issue of pSA. The proposed TApSA incorporates a time-average operation into the pSA algorithm. This operation, which is added to Eq. (3) of pSA, is defined as follows:

$$TI_i(t+1) = h_i + \sum_j J_{ij} \cdot \sigma_j(t), \tag{4a}$$

$$I_i(t+1) = I_0\left(\frac{1}{\alpha}\sum_{i=0}^{\alpha-1} TI_i(t+1-i)\right), \tag{4b}$$

where $TI_i(t+1)$ represents a temporary value used for the time averaging operation and the variable α is the size of the time window over which $TI_i(t+1)$ is averaged. $I_i(t+1)$ is the input for the p-bit, as defined in Eq. (1) that is also used in TApSA. The set of equations in Eq. (4) is responsible for calculating the time average of the p-bit input signal. This operation effectively smooths out the signal over a certain time window, which helps to reduce random fluctuations or 'noise' in the signal. Another consequence of this time-averaging operation is that it highlights or emphasizes the lower frequency components of the signal while simultaneously reducing or attenuating the higher frequency components.

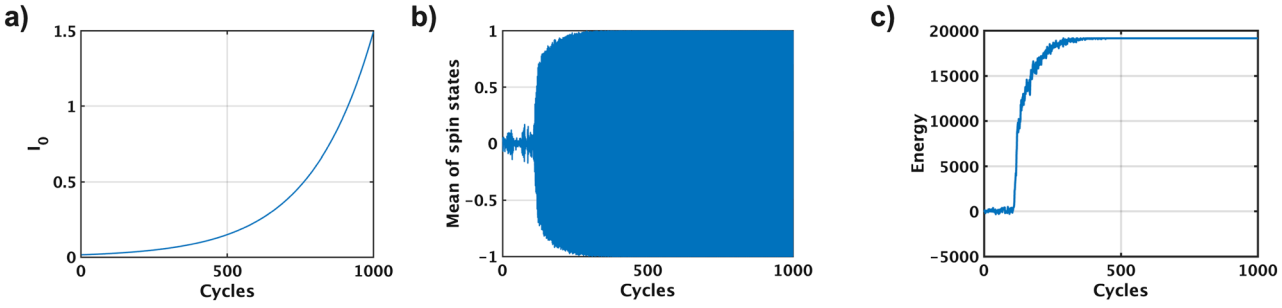

Figure 3. Issue of pSA. pSA is simulated by Python with G1 that is a MAX-CUT problem of the G-set benchmark. During the simulated annealing process, the pseudo inverse temperature I_0 is increased to control pSA (**a**). A mean of all the p-bits states is changed between '– 1' and '+ 1' at every cycle (**b**). Because of this oscillation, the energy starts to increase after the oscillation, although the energy is expected to be lower to the global minimum energy (**c**).

SA based on stalled p-bit (SpSA)

The second algorithm is simulated annealing based on stalled p-bit (SpSA). SpSA takes inspiration from the sparse random signals used in invertible logic operations[42]. Invertible logic is an application of p-bits, which permits bidirectional operations of any function. Sparse random signals are applied in such a way that they probabilistically halt, or "stall," the addition of random signals to the invertible logic operations, subsequently reducing error rates.

The SpSA algorithm employs a similar approach, where it probabilistically stalls the behavior of p-bits. Specifically, in SpSA, the input of a p-bit, represented as $I_i(t)$, is probabilistically stalled and maintains the same value $I_i(t)$ from the previous time step. The equation for SpSA as mentioned above is:

$$I_i(t+1) = \begin{cases} I_i(t), & \text{with probability of getting stalled } p \\ I_0\left(h_i + \sum_j J_{ij} \cdot \sigma_j(t)\right). & \text{with probability}(1-p) \end{cases} \quad (5)$$

In this equation, the input of the p-bit at time t+1, $I_i(t+1)$, can either be stalled (i.e., be the same as the input at time t, $I_i(t)$), with a probability p, or take on a new value with a probability of $(1-p)$. This approach is a significant deviation from traditional pSA, where Eq. (3) is replaced by Eq. (5) in SpSA.

MAX-CUT problems and annealing parameters for evaluation

The proposed TApSA and SpSA algorithms are evaluated in MAX-CUT problems using Python 3.11 on Apple M1 Ultra with 128 GB memory. Two MAX-CUT benchmarks, namely G-set and K2000, are used for these simulations (Table 1). The G-set includes Gxx graphs that vary in node sizes and edge connections[34]. On the other hand, K2000 represents a fully-connected graph with edge weights of either '– 1' or '+ 1'[43]. Before the simulated annealing process begins, J of the Ising model is assigned based on the graph weights.

Performance of the proposed TApSA and SpSA algorithms is compared with the traditional pSA. The annealing algorithms are outlined in Table 2. In these simulated annealing algorithms, a crucial factor is the manipulation of the pseudo inverse temperature, a value which has significant implications on annealing to explore the solution space. During the simulated annealing process, the pseudo inverse temperature I_0 is gradually increased over time from the initial value I_{0min} to the maximum value I_{0max}, following the formula $I_0(t+1) = I_0(t)/\beta$

Graph	# nodes	Structure	Weights (J)	# Edges	Best known value
G1	800	Random	+ 1	19,176	11,624
G6	800	Random	+ 1, – 1	19,176	2178
G11	800	Troidal	+ 1, – 1	1600	564
G14	800	Planar	+ 1	4694	3064
G18	800	Planar	+ 1, – 1	4694	992
G22	2000	Random	+ 1	19,990	13,359
G34	2000	Troidal	+ 1, – 1	4000	1384
G38	2000	Planar	+ 1	11,779	7688
G39	2000	Planar	+ 1, – 1	11,778	2408
G47	1000	Random	+ 1	9990	6657
G48	3000	Troidal	+ 1, – 1	6000	6000
G54	1000	Random	+ 1	5916	3852
G55	5000	Random	+ 1	12,498	10,299
G56	5000	Random	+ 1, – 1	12,498	4017
G58	5000	Planar	+ 1	29,570	19,293
K2000	2000	Full	+ 1, – 1	1,999,000	33,337

Table 1. Summary of MAX-CUT benchmarks used for evaluating simulated annealing algorithms, including TApSA, SpSA, and traditional SA and pSA. The Gxx graphs are part of the G-set benchmark[34], and K2000 is a fully-connected graph benchmark[43].

Algorithm	Equations
pSA[13]	Eqs. (1) and (3)
TApSA (proposed)	Eqs. (1) and (4)
SpSA (proposed)	Eqs. (1) and (5)

Table 2. Summary of the determined hyperparameters for pSA, TApSA, and SpSA. In the simulated annealing process, all the p-bit states are updated using these equations in parallel in attempt to reach the global minimum energy of an Ising model.

(Table 3). The hyperparameters for the simulated annealing processes, such as I_{0min}, I_{0max}, and β, are not arbitrarily selected. Rather, they are determined in accordance with a specific statistical method, which is designed to optimize the performance of the simulated annealing algorithm (SSA)[44]. In addition to these, a traditional SA algorithm, a well-established method for optimization problems, is also implemented for the sake of performance comparison[16]. In the traditional SA algorithm, the annealing temperature T is managed in a slightly different manner: the temperature is gradually decreased at each cycle by a factor of Δ_{IT}, following the equation $T \leftarrow 1/(1/T + \Delta_{IT})$. In this experiment, the initial temperature is set to 1, and the final temperature is set to 1/1000.

All of the simulated annealing algorithms, including TApSA, SpSA, pSA, and traditional SA, are simulated for a total of 1,000 cycles each. This number of cycles allows the annealing system ample opportunity to explore the solution space and converge to a solution. Due to the inherent randomness in these probabilistic algorithms, the evaluation is not based on a single trial. Instead, to get a more accurate understanding of their performance, 100 separate trials are executed for each algorithm. The outcomes of these trials are then used to calculate the minimum, average, and maximum cut values of the MAX-CUT problems, providing a comprehensive assessment of the algorithms' performance.

Results

Simulation analysis of TApSA

The TApSA algorithm is simulated on the G1 graph with varying window size α (Fig. 4). During the annealing process, the pseudo inverse temperature I_0 is gradually increased over time for 1,000 cycles from $I_{0min} = 0.0149$

Parameter	Value
s_i	$\sqrt{(n-1)\cdot \mathrm{Var}(\mathrm{J}_{i,:})}$
I_{0min}	$\frac{\gamma}{\mathrm{mean}(\mathrm{s_i})}$
I_{0max}	$\frac{\delta}{\mathrm{mean}(\mathrm{s_i})}$
β	$\left(\frac{I_{0min}}{I_{0max}}\right)^{\left(\frac{1}{cycle-1}\right)}$

Table 3. Statistically determined hyperparameters for pSA, TApSA, and SpSA. A pseudo inverse temperature is gradually increased over time from I_{0min} to I_{0max}, following the formula $I_0(t+1) = I_0(t)/\beta$. $\gamma = 0.1$ and $\delta = 10$ are used in this article.

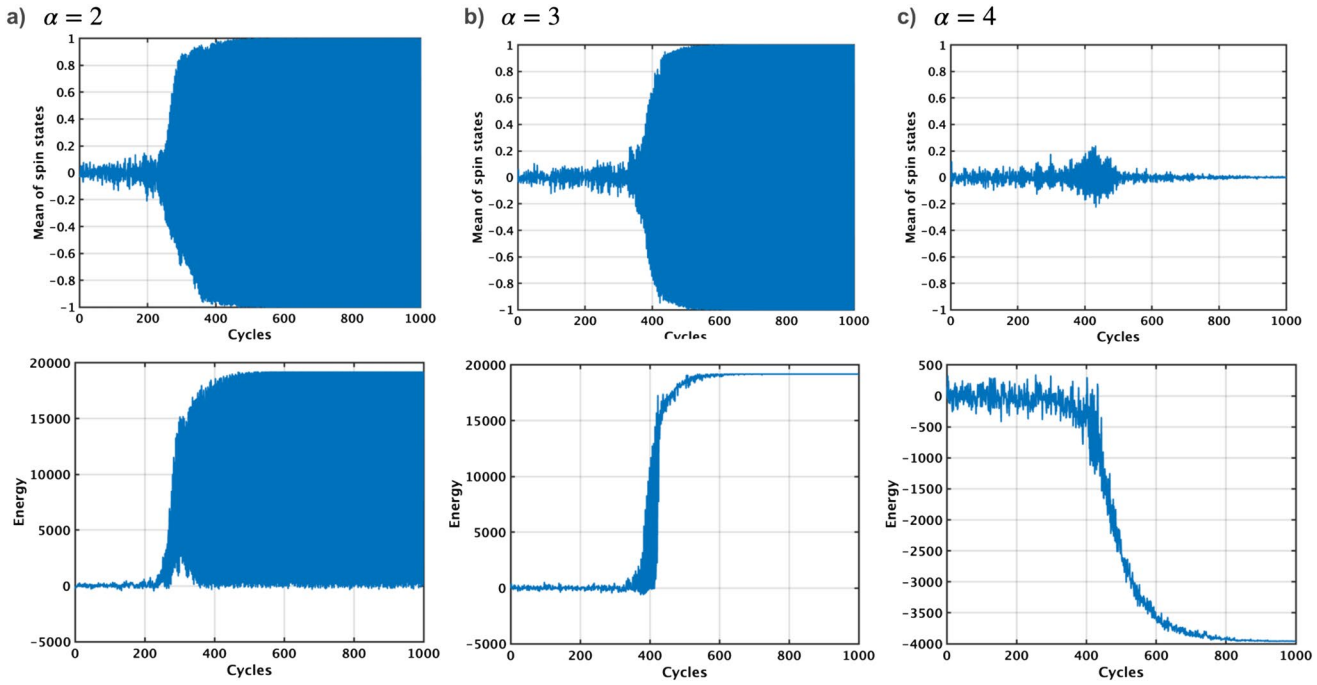

Figure 4. Simulation analysis of TApSA on the G1 graph with different window size α. The mean values of all the p-bit states are oscillated between '− 1' and '+ 1' in case of α of two and three (top of **a** and **b**), resulting in the energy increase of the Ising model instead of decrease (bottom of **a** and **b**). When α is four, no oscillation occurs and hence the energy goes down to the global minimum energy (**c**). TApSA can solve the oscillation issue of pSA.

to $I_{0max} = 1.49$, following the formula $I_0(t+1) = I_0(t)/0.995$. α is incrementally increased from two to four to observe the corresponding changes in the behavior of TApSA. Note that when α equals to one, TApSA operates exactly the same as pSA. Increasing α can make signals of p-bits smoother. For $\alpha = 2$, the mean of all the p-bit states oscillates between '− 1' and '+ 1'. This oscillatory pattern implies that the algorithm alternates between two distinct states throughout its execution. This behavior, however, does not lead to the energy of the system decreasing towards the global minimum. Instead, it causes an increase in the energy level. Similarly, when α is increased to three, the oscillation persists, but its start cycle is delayed in comparison to when $\alpha = 2$. Despite this delayed onset of oscillation, the energy of the Ising model still increases, failing to converge to the global minimum. However, a significant change in behavior is observed when α is increased to four. In this case, no oscillation is observed in the mean of the p-bit states. This allows the energy of the Ising model to decrease steadily, eventually reaching the global minimum. Thus, for this specific graph and set of conditions, an α value of four appears to facilitate the effective optimization, leading the annealing system towards the global minimum energy state.

Next, the TApSA algorithm is simulated to evaluate the normalized cut value on the G1, G11, G58, and K2000 MAX-CUT problems by varying the windows size α from one to ten (Fig. 5). By varying the window size, the behavior of the TApSA algorithm and the resulting cut values are influenced. To evaluate the performance, the normalized cut values are calculated using the minimum, mean and maximum cut value divided by the best-known value for each benchmark graph. This normalization process allows for fair comparisons across different graph structures and scales. The parameters to control the pseudo inverse temperature I_0 are summarized in Table 4. These parameters are determined based on Table 3.

As α is increased to a specific value, the cut values tend to get closer to the best-known values due to the elimination of oscillation in the algorithm. However, when α surpasses this optimal value, the normalized cut value starts to decrease slightly as α continues to increase. This suggests that while increasing α can improve

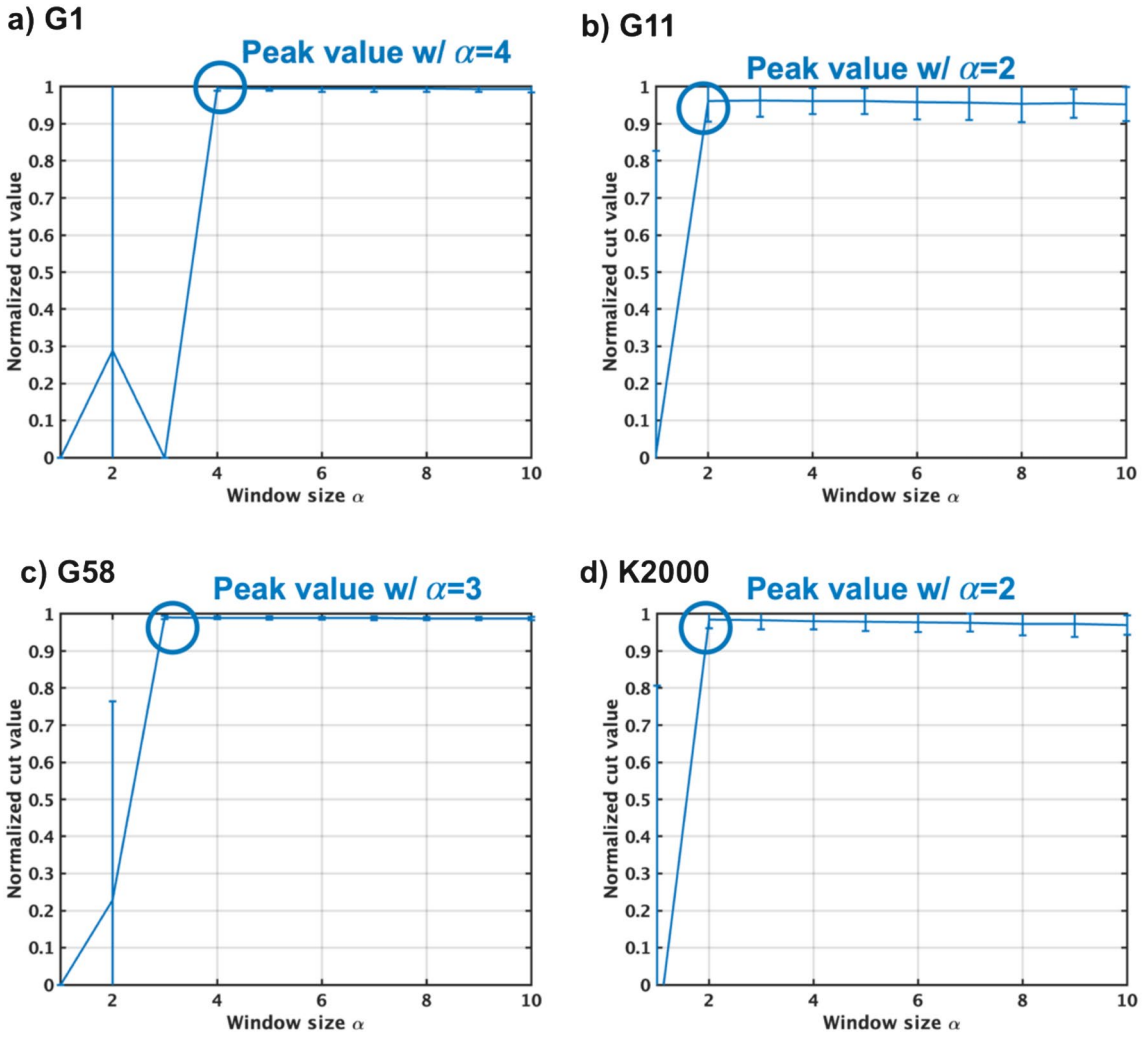

Figure 5. Normalized cut values using the TApSA algorithm on the G1, G11, G58 and K2000 MAX-CUT problems by varying the windows size α from one to ten. This window size α plays a key role in determining the quality of the solution and can control the behavior of the TApSA algorithm. When α is increased to a specific value, the cut values can be closer to the best-known values because of no oscillation. The peak of the normalized mean cut value is obtained with different α depending on the graph.

Parameter	G1	G11	G58	K2000
s_i	6.69	1.99	3.22	44.7
I_{0min}	0.0149	0.0501	0.0311	0.00224
I_{0max}	1.49	5.01	3.11	0.224
β	0.995	0.995	0.995	0.995

Table 4. Summary of hyperparameters to control I_0 for pSA, TApSA, and SpSA on the G1, G11, G58, K2000 benchmarks. The number of cycles is 1000.

performance up to a point, overly large window sizes may have a detrimental effect. Interestingly, the specific value of α that yields the peak normalized mean cut values varies depending on the graph. This indicates that the optimal window size is not universal but instead depends on the particular characteristics of the graph being analyzed. The source of these differing optimal α values can be traced to the weights present in the respective graphs, as listed in Table 1. For instance, G1 and G58 only contain weights of '+ 1', while G11 and K2000 contain both '− 1' and '+ 1' weights. This imbalance in weights can lead to strong oscillations in the algorithm. Large α values can help to mitigate these oscillations, effectively smoothing the search through the solution space and improving its ability to find the global minimum.

Simulation analysis of SpSA

The G58 graph is used to evaluate the SpSA algorithm, considering varying probabilities of p-bits getting stalled, denoted as p (Fig. 6). The parameter p is systematically incremented from 0.1 to 0.5 in order to thoroughly understand how changes in p impact the behavior and effectiveness of the SpSA algorithm. When p equals zero, there is no discernible difference in how SpSA and the pSA algorithm function. When p is 0.1, the average of all p-bit states starts to oscillate, alternating between '− 1' and '+ 1'. This oscillation mirrors the behavior observed in the TApSA algorithm with small α, leading to an increase in the energy level of the system, which usually signifies a less optimal solution. When p rises to 0.3, though the oscillation continues, its magnitude is diminished compared to the scenario where $p = 0.1$. The most significant change in behavior of the SpSA algorithm is observed when p reaches 0.5. In this situation, the oscillation of the average p-bit states ceases completely. The absence of oscillation allows the energy of the Ising model to decrease steadily. As the energy reduces, the energy moves closer to the most optimal solution, ultimately achieving the global minimum, which represents the best possible solution.

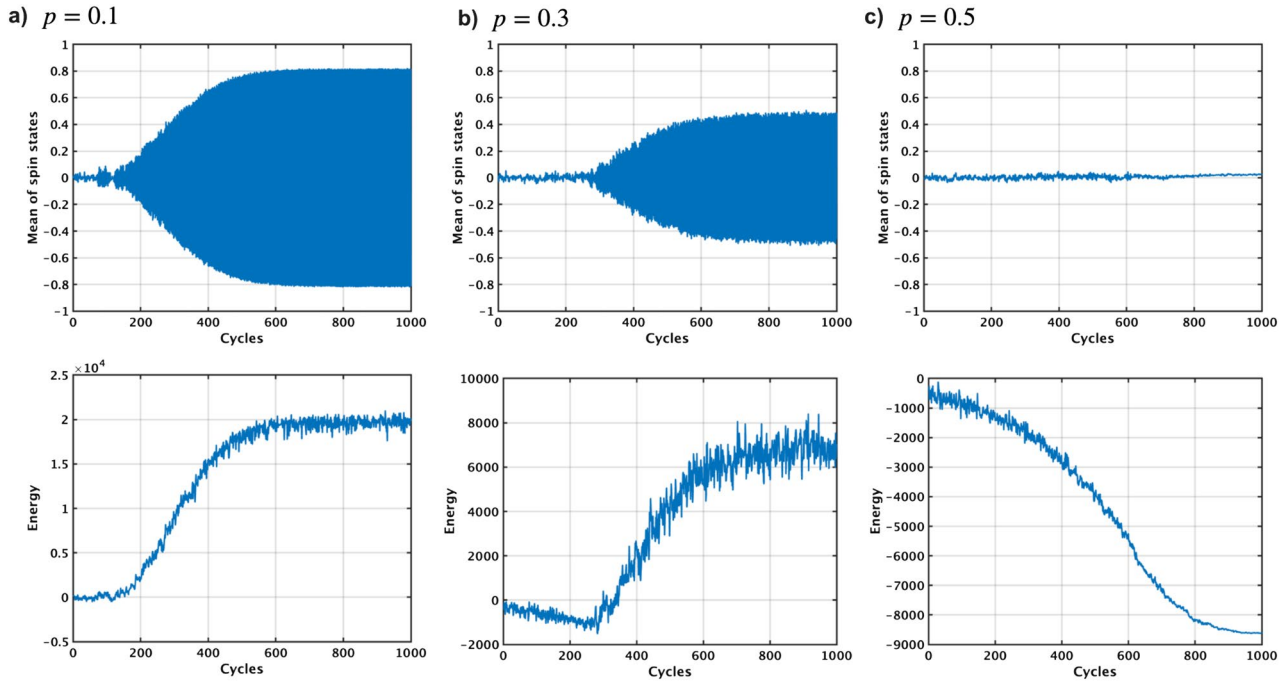

Figure 6. Simulation analysis of SpSA on the G58 graph with different probabilities of getting stalled on p-bits p. The mean values of all the p-bit states are oscillated between '− 1' and '+ 1' in case of p of 0.1 and 0.3 (top of **a** and **b**), resulting in the increase of energy instead of decrease (bottom of **a** and **b**). When p is 0.5, no oscillation occurs and hence the energy goes down to the global minimum energy (**c**). SpSA can also solve the oscillation issue of pSA.

Next, the SpSA algorithm is simulated to evaluate the normalized cut value on the G1, G11, G58, and K2000 MAX-CUT problems by varying the probability of p-bits getting stalled p from 0 to 0.9 (Fig. 7). As p is increased to a specific value, the cut values tend to get closer to the best-known values due to the elimination of oscillation as well as TApSA. When p surpasses this optimal value, the normalized cut value starts to decrease slightly as p continues to increase. The specific value of p that yields the peak normalized mean cut values varies depending on the graph.

Based on the simulated results of TApSA and SpSA, increasing α exhibits the similar effect to increasing p in terms of eliminating the oscillation, which is crucial for effective optimization. In both TApSA and SpSA, there appears to be an optimal value of α and p, respectively, which yields the best performance in terms of the normalized cut value. However, this optimal value is not universal and depends on the specifics of the graph. Moreover, surpassing these optimal values can actually lead to a decrease in performance.

Performance comparisons

A comparative analysis of the cut values on the G1 graph are conducted for three different simulated annealing algorithms: pSA, TApSA, and SpSA (Fig. 8). To assess the effectiveness of these algorithms in finding cut values, simulations involving 1000 cycles are performed for each, and these are repeated 100 times. The repetition of these simulations leads to the collection of a substantial amount of data, enabling a robust evaluation of the minimum, mean, and maximum cut values. All cut values using pSA are found to be zero, an intriguing outcome attributable to the oscillation that is observed during the annealing process. This suggests that the pSA algorithm is unstable under these conditions, leading to a failure to produce any viable cut values. In the case of the TApSA and SpSA algorithms, they are simulated with the most advantageous parameters, namely $\alpha = 4$ and $p = 0.6$, respectively. The choice of these specific values is determined on previous experiments and analysis, which demonstrated superior performance. When comparing the results, it is shown that both TApSA and SpSA

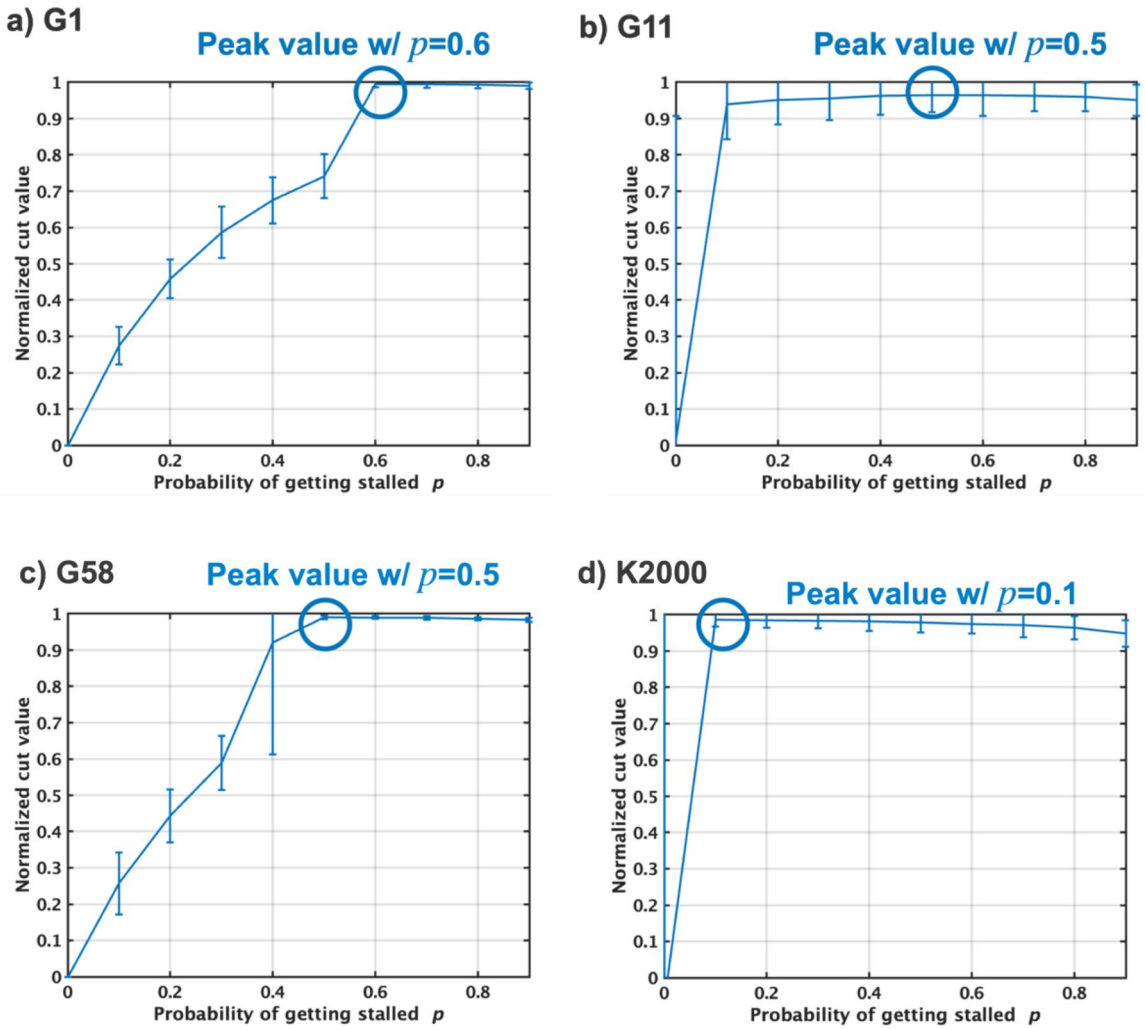

Figure 7. Normalized cut values using the SpSA algorithm on the G1, G11, G58 and K2000 MAX-CUT problems by varying the stalled probability p from 0 to 0.9. When p is increased to a specific value, the cut values can be closer to the best-known values because of no oscillation as well as SpSA.

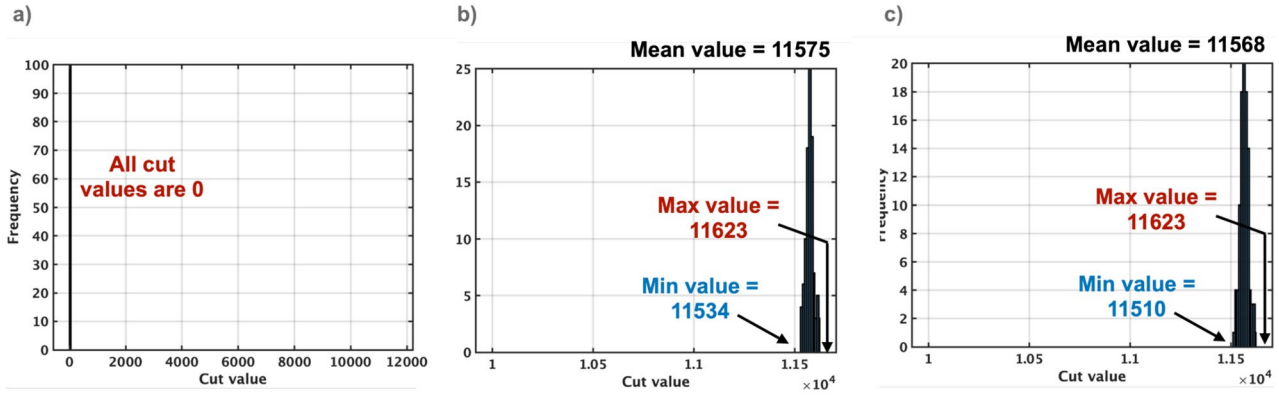

Figure 8. Cut values obtained using simulated annealing on the G1 graph for 100 trials. The conventional pSA causes all 0 values due to the oscillation (**a**). In contrast, TApSA with $\alpha = 4$ and SpSA with $p = 0.6$ can reduce the energy of the Ising model, which results in good cut values that are closer to the best-known values (**b** and **c**).

outperform pSA, achieving near-optimal solutions. This is largely due to the elimination of oscillations observed in the pSA algorithm, made possible by the design of the TApSA and SpSA algorithms.

Furthermore, a comparison is also conducted between the proposed algorithms (TApSA and SpSA) and the traditional SA method, specifically using the G-set and K2000 benchmarks. The detailed results of this comparison are summarized in Table 5. The pSA algorithm is found to consistently fail to lower the energy of the Ising model, leading to particular patterns in the mean cut values. When the weight values of the graphs are solely '+ 1', the mean cut values turns out to be 0. Conversely, when the weight values are either '− 1' or '+ 1', the mean cut values are negative. On the 16 benchmarks, the normalized mean cut value of pSA is only 0.8% on average. The traditional SA algorithm, on the other hand, delivers better cut values compared to pSA. However, it is worth noting that these values are still significantly off from the best-known values, where the normalized mean cut value is 44.4% on average. Remarkably, the proposed algorithms, TApSA and SpSA, demonstrate substantial superiority over both pSA and traditional SA. TApSA and SpSA achieve the normalized mean cut value of 98.3% and 98.4% on average, respectively, in all 16 benchmarks used in this study. This underlines the effectiveness of these proposed methods and their potential for practical applications in solving similar optimization problems.

	SA[14]	pSA[13]	TApSA (proposed)		SpSA (proposed)	
Graph	Mean cut value	Mean cut value	Mean cut value	Window size α	Mean cut value	Probability of getting stalled p
G1	10757.91	0	11574.69	4	11567.89	0.6
G6	1270.88	173.48	2150.49	2	2151.23	0.1
G11	336.72	6.18	542.7	3	543.78	0.5
G14	2801.84	0	3035.74	3	3034.78	0.5
G18	591.5	− 49.79	968.31	2	968.94	0.1
G22	11161.58	0	13277.55	3	13271.27	0.5
G34	469.22	− 29.62	1331.22	2	1335.72	0.5
G38	6638.96	0	7617.3	3	7610.48	0.5
G39	854.57	− 489.49	2343.52	2	2349.57	0.2
G47	5851.46	0	6623.31	3	6618.35	0.6
G48	3563.94	3057.22	5867.16	2	5897	0.1
G54	3479.42	0	3815.16	3	3811.77	0.5
G55	6968.09	0.03	10184.66	2	10193.41	0.2
G56	696.99	− 185.22	3900.35	2	3912.14	0.1
G58	15787.85	0	19108.08	3	19096.28	0.5
K2000	11369.62	− 4889.64	32812.64	2	32860.58	0.1

Table 5. Comparisons of mean cut values in the MAX-CUT benchmarks. On the 16 benchmarks from 800 to 5000 nodes, the proposed methods improve the normalized cut value from 0.8 to 98.4% on average in comparison with the conventional pSA.

Discussion

In this article, we have critically examined the limitations of the simulated annealing using probabilistic bits (pSA) algorithm, specifically with large-scale combinatorial optimization problems such as the maximum cut (MAX-CUT) problem. Our detailed analysis has identified disruptive oscillations as the root cause of energy stagnation in the pSA process. To mitigate this, we have proposed and rigorously tested two novel algorithms, time average pSA (TApSA) and stalled pSA (SpSA). The results suggest significant performance improvements over traditional methods (SA and pSA), highlighting the potential of our proposed algorithms in effectively tackling large-scale optimization tasks.

Stochastic simulated annealing (SSA) is another p-bit-based simulated annealing that outperforms pSA and SA in several combinatorial optimization problems[21]. SSA is implemented using digital hardware because the tanh function is approximated using stochastic computing. In terms of energy consumption, TApSA and SpSA can reduce energy consumption than SSA because of the nature of their implementation. When the p-bit is implemented using an emerging device, the energy consumption can be 10 times smaller than that implemented using a traditional digital circuit[3]. In addition, the p-bit is a single device, while it can be approximated using several hundreds of transistors in digital implementation, resulting in a more compact hardware implementation. In the future, large-scale p-bit-based simulated annealing, TApSA and SpSA could gain in terms of energy and area consumption in comparison with SSA.

In terms of computation cost, TApSA and SpSA require extra computation for nonlinear functions in comparison with pSA. In Fig. 9a, the simulation time for TApSA is plotted against the parameter α, while Fig. 9b illustrates the simulation time for SpSA as a function of p, with both scenarios considering 1000 cycles for solving problems G1, K2000, and G58. It is observed that TApSA's simulation time remains relatively constant regardless of variations in α, across the same number of cycles. In contrast, SpSA exhibits longer simulation times at smaller values of α, as opposed to shorter times at larger p values. This consistent behavior is further detailed in Fig. 9c, which shows the total simulation time, inclusive of the duration spent tuning α or p. Prior to the annealing process, parameter tuning for TApSA and SpSA involves adjusting α or p over 100 cycles to identify the optimal parameter settings. When compared to pSA, it is noteworthy that the additional computational costs associated with TApSA and SpSA are less significant at higher cycle counts. From another point of view, it would be preferable to realize new emerging devices that mimic these algorithms for future implementation.

To determine if pSA induces oscillation in optimization problems other than MAX-CUT, a 100-spin graph isomorphism (GI) problem is employed. This problem involves ascertaining the isomorphism of two 10-node graphs. GI problems are inherently more complex than MAX-CUT problems, primarily due to the presence of non-zero values of h in GI, in contrast to the all-zero values of h in MAX-CUT. Previous literature[21] noted that pSA failed to converge in GI problems with more than 25 spins, though it did not explicitly address the oscillation of p-bit states. To further explore this issue, both pSA and TApSA are applied to the 100-spin GI problem over 1000 cycles as shown in Fig. 10. The result shows that pSA, similar to its performance in MAX-CUT problems, is unable to reduce energy due to p-bit state oscillation. On the other hand, TApSA effectively resolves this oscillation issue, achieving a reduction in energy to the global minimum. These findings suggest that while TApSA shows promise in addressing oscillation issues in various algorithms, a more detailed analysis in other contexts is planned for future work.

The performance of the newly proposed TApSA algorithm is benchmarked against other notable methods, specifically the GPU-based asynchronous parallel algorithm[45] and the coherent Ising machine (CIM)[30], with details presented in Table 6. This comparison uses normalized mean cut values, calculated by dividing the mean cut value by the best-known value for each problem. The GPU-based approach evaluates mean cut values over 1000 annealing steps. Conversely, the performance data for CIM, sourced from the literature[47], is based on simulations using SimCIM[46] for a substantially longer duration of 50,000 annealing steps. In comparison with the GPU-based method, TApSA achieves comparable normalized mean cut values. While TApSA's performance is marginally lower than CIM, it shows potential for improvement with an increased number of annealing steps.

Initially, p-bits are modeled with uniform random signals, where $r_i(t) \in \{-1 : 1\}$, as outlined in Eq. (1). To explore the impact of random signals on performance, this study introduces random signals derived from a Poisson distribution for the p-bits. Specifically, the random signal $r_i(t) = 1/\lambda \cdot X - 1$ is generated according to the Poisson probability formula $P(X = k) = e^{-\lambda}\lambda^k/k!$, with λ set to 10. A comparison of normalized mean cut values,

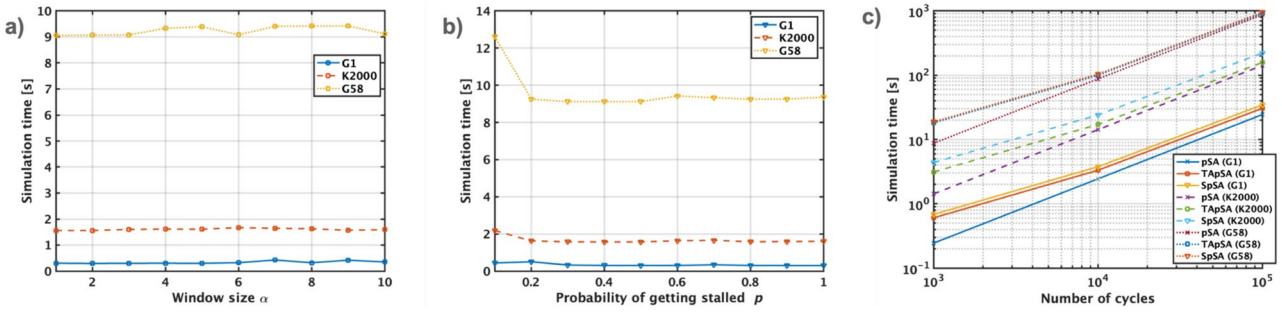

Figure 9. The simulation duration for TApSA and SpSA was assessed by varying the parameters α and p, respectively, across 1000 cycles (as shown in figures **a** and **b**). This evaluation includes the total time spent on the simulations over the number of cycles, incorporating the time required for tuning the parameters α or p (**c**).

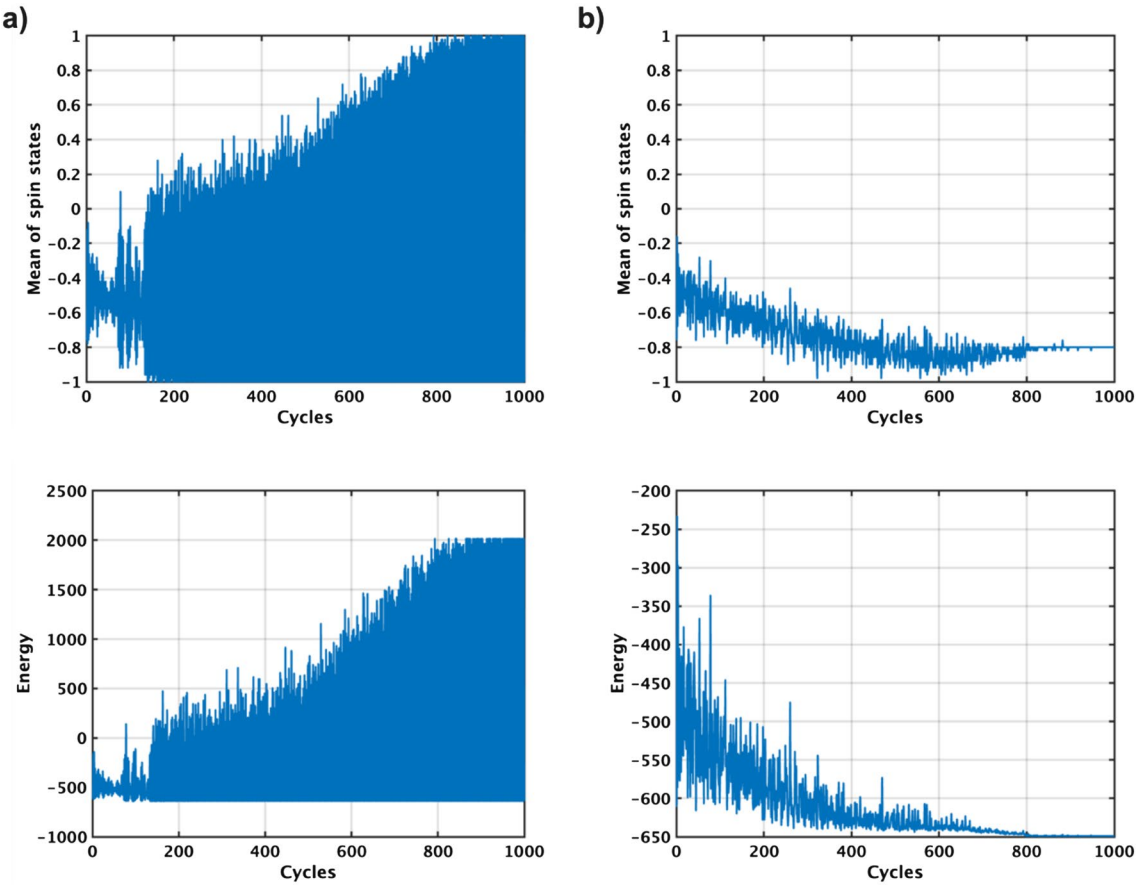

Figure 10. In the context of a 100-spin graph isomorphism (GI) problem, which involves determining if two 10-node graphs are isomorphic, distinct performance characteristics of pSA and TApSA with $\alpha = 10$ are observed. In case (**a**), pSA struggles to minimize energy due to oscillations in p-bit states, a challenge similarly noted in the MAX-CUT problem. Conversely, TApSA effectively addresses this oscillation issue, as shown in case (**b**), leading to a significant reduction in energy that can reach the global minimum.

Benchmark	GPU[45]	SimCIM[47]	TApSA
G22	N/A	99.6%	99.4%
G39	N/A	97.9%	97.3%
G47	99.4%	N/A	99.5%
G54	97.7%	N/A	99.0%
K2000	N/A	99.6%	98.4%

Table 6. Comparisons of normalized mean cut values obtained in this research with those reported in related works[45,47]. In this comparison, both the GPU-based approach and TApSA are configured to perform 1000 annealing steps. In contrast, the SimCIM approach, as referenced in the literature, requires a significantly higher number of annealing steps, totaling 50,000. This disparity in the number of annealing steps highlights the efficiency differences among these methods.

$r_i(t)$	pSA	TApSA	SpSA
Uniform	0.828%	98.3%	98.4%
Poisson	3.94%	97.9%	97.9%

Table 7. Normalized mean cut values on average for all 16 benchmarks with different random signals.

using both uniform random and Poisson distribution-based signals, is presented in Table 7. The results indicate that the type of random signals has a negligible effect on the performance of all three algorithms under study.

P-bits have found application in various domains, one of which includes Gibbs sampling[12]. A key distinction between Gibbs sampling and pSA lies in the approach to node updates: Gibbs sampling typically operates serially, while pSA updates nodes in parallel. It is important to note that although extended versions of Gibbs sampling, such as chromatic Gibbs sampling, have the capability to operate in parallel[7], the scope of their applications is relatively limited. Simulation results have shown that pSA faces an oscillation issue due to its parallel update mechanism, a problem which the proposed algorithms aim to address. However, applying techniques based on TApSA and SpSA to Gibbs sampling, which inherently operates serially, presents a significant challenge. Exploring a Gibbs sampling method that incorporates the proposed techniques is an intriguing direction for future research.

In conclusion, our research has broadened the understanding of the pSA process and has led to the development of more effective algorithms for complex optimization tasks. The proposed TApSA and SpSA algorithms offer promising avenues for overcoming the limitations of the traditional pSA approach and could be crucial for future progress in combinatorial optimization.

Data availability

All data generated or analyzed during this study are included in this published article. The Python codes are available at https://github.com/nonizawa/pSA.

Received: 14 September 2023; Accepted: 8 January 2024
Published online: 16 January 2024

References

1. Camsari, K., Faria, R., Sutton, B. & Datta, S. Stochastic p-bits for invertible logic. *Phys. Rev. X* **7**, 156 (2017).
2. Pervaiz, A. Z., Ghantasala, L. A., Camsari, K. Y. & Datta, S. Hardware emulation of stochastic p-bits for invertible logic. *Sci. Rep.* **7**, 10994. https://doi.org/10.1038/s41598-017-11011-8 (2017).
3. Borders, W. A. *et al.* Integer factorization using stochastic magnetic tunnel junctions. *Nature* **573**, 390–393. https://doi.org/10.1038/s41586-019-1557-9 (2019).
4. Pervaiz, A. Z., Sutton, B. M., Ghantasala, L. A. & Camsari, K. Y. Weighted *p* -bits for FPGA implementation of probabilistic circuits. *IEEE Trans. Neural Netw. Learn. Syst.* **30**, 1920–1926 (2019).
5. Smithson, S. C., Onizawa, N., Meyer, B. H., Gross, W. J. & Hanyu, T. Efficient CMOS invertible logic using stochastic computing. *IEEE Trans. Circ. Syst. I Regul. Pap.* **66**, 2263–2274 (2019).
6. Sutton, B. *et al.* Autonomous probabilistic coprocessing with petaflips per second. *IEEE Access* **8**, 157238–157252 (2020).
7. Aadit, N. A., Grimaldi, A., Finocchio, G. & Camsari, K. Y. Physics-inspired ising computing with ring oscillator activated p-bits. In *2022 IEEE 22nd International Conference on Nanotechnology (NANO)* 393–396 (2022).
8. Hinton, G. E., Sejnowski, T. J. & Ackley, D. H. Boltzmann machines: Constraint satisfaction networks that learn. In *Tech. Rep. CMU-CS-84-119, Department of Computer Science, Carnegie-Mellon University* (1984).
9. Onizawa, N., Smithson, S. C., Meyer, B. H., Gross, W. J. & Hanyu, T. In-hardware training chip based on cmos invertible logic for machine learning. *IEEE Trans. Circ. Syst. I Regul. Pap.* **67**, 1541–1550 (2020).
10. Kaiser, J. *et al.* Hardware-aware in situ learning based on stochastic magnetic tunnel junctions. *Phys. Rev. Appl.* **17**, 014016. https://doi.org/10.1103/PhysRevApplied.17.014016 (2022).
11. Grimaldi, A. *et al.* Spintronics-compatible approach to solving maximum-satisfiability problems with probabilistic computing, invertible logic, and parallel tempering. *Phys. Rev. Appl.* **17**, 024052. https://doi.org/10.1103/PhysRevApplied.17.024052 (2022).
12. Aadit, N. A. *et al.* Massively parallel probabilistic computing with sparse ising machines. *Nat. Electron.* **5**, 460–468. https://doi.org/10.1038/s41928-022-00774-2 (2022).
13. Camsari, K. Y., Sutton, B. M. & Datta, S. p-bits for probabilistic spin logic. *Appl. Phys. Rev.* **6**, 011305 (2019).
14. Kirkpatrick, S., Gelatt, C. D. Jr. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).
15. Johnson, D. S., Aragon, C. R., McGeoch, L. A. & Schevon, C. Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning. *Oper. Res.* **39**, 378–406 (1981).
16. Mykleburst, T. Solving maximum cut problems by simulated annealing. In *CoRR* 1110.2574 (2015). arXiv: org/abs/1505.03068.
17. Elmitwalli, E., Ignjatovic, Z. & Köse, S. Utilizing multi-body interactions in a CMOS-based ising machine for LDPC decoding. *IEEE Trans. Circ. Syst. I Regul. Pap.* **2023**, 1–11 (2023).
18. Park, H.-K., Lee, J.-H., Lee, J. & Kim, S.-K. Optimizing machine learning models for granular ndfeb magnets by very fast simulated annealing. *Sci. Rep.* **11**, 3792. https://doi.org/10.1038/s41598-021-83315-9 (2021).
19. Reiter, E. E. & Johnson, C. M. *Limits of Computation: An Introduction to the Undecidable and the Intractable* (Chapman and Hall/CRC, 2012).
20. Earl, D. J. & Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Phys. Chem. Chem. Phys.* **7**, 3910–3916. https://doi.org/10.1039/B509983H (2005).
21. Onizawa, N., Katsuki, K., Shin, D., Gross, W. J. & Hanyu, T. Fast-converging simulated annealing for Ising models based on integral stochastic computing. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–7 (2022).
22. Aramon, M. *et al.* Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. *Front. Phys.* **7**, 156. https://doi.org/10.3389/fphy.2019.00048 (2019).
23. Gyoten, H., Hiromoto, M. & Sato, T. Enhancing the solution quality of hardware ising-model solver via parallel tempering. In *Proceedings of the International Conference on Computer-Aided Design, ICCAD '18* (Association for Computing Machinery, New York, NY, USA, 2018). https://doi.org/10.1145/3240765.3240806.
24. Shin, D., Onizawa, N., Gross, W. J. & Hanyu, T. Memory-efficient fpga implementation of stochastic simulated annealing. *IEEE J. Emerg. Sel. Top. Circ. Syst.* **13**, 108–118 (2023).
25. Kadowaki, T. & Nishimori, H. Quantum annealing in the transverse ising model. *Phys. Rev. E* **58**, 5355–5363 (1998).
26. Boixo, S. *et al.* Evidence for quantum annealing with more than one hundred qubits. *Nat. Phys.* **10**, 218–224. https://doi.org/10.1038/nphys2900 (2014).
27. Neven, H. When can quantum annealing win? (2016). https://ai.googleblog.com/2015/12/when-can-quantum-annealing-win.html.
28. Zick, K. M., Shehab, O. & French, M. Experimental quantum annealing: Case study involving the graph isomorphism problem. *Sci. Rep.* **5**, 11168. https://doi.org/10.1038/srep11168 (2015).
29. Yarkoni, S., Raponi, E., Bäck, T. & Schmitt, S. Quantum annealing for industry applications: Introduction and review. *Rep. Prog. Phys.* **85**, 104001 (2022).

30. Wang, Z., Marandi, A., Wen, K., Byer, R. L. & Yamamoto, Y. Coherent ising machine based on degenerate optical parametric oscillators. *Phys. Rev. A* **88**, 063853. https://doi.org/10.1103/PhysRevA.88.063853 (2013).
31. Goto, H., Tatsumura, K. & Dixon, A. R. Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems. *Sci. Adv.* **5**, eaav2372. https://doi.org/10.1126/sciadv.aav2372 (2019).
32. Dutta, S. *et al.* An Ising Hamiltonian solver based on coupled stochastic phase-transition nano-oscillators. *Nat. Electron.* **4**, 502–512. https://doi.org/10.1038/s41928-021-00616-7 (2021).
33. Burer, S., Monteiro, R. D. C. & Zhang, Y. Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs. *SIAM J. Optim.* **12**, 503–521 (2001).
34. Ye, Y. Computational optimization laboratory (1999). http://web.stanford.edu/~yyye/Col.htm.
35. Gaines, B. R. Stochastic computing systems. *Adv. Inf. Syst. Sci. Plenum* **2**, 37–172 (1969).
36. Brown, B. D. & Card, H. C. Stochastic neural computation. I. Computational elements. *IEEE Trans. Comput.* **50**, 891–905 (2001).
37. Gaudet, V. C. & Gross, W. J. *Stochastic Computing: Techniques and Applications* (Springer International Publishing, 2019).
38. Gaudet, V. C. & Rapley, A. C. Iterative decoding using stochastic computation. *Electron. Lett.* **39**, 299–301 (2003).
39. Li, P., Lilja, D. J., Qian, W., Bazargan, K. & Riedel, M. D. Computation on stochastic bit streams digital image processing case studies. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **22**, 449–462 (2014).
40. Liu, Y. & Parhi, K. K. Architectures for recursive digital filters using stochastic computing. *IEEE Trans. Signal Process.* **64**, 3705–3718 (2016).
41. Ardakani, A., Leduc-Primeau, F., Onizawa, N., Hanyu, T. & Gross, W. J. VLSI implementation of deep neural network using integral stochastic computing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **25**, 2588–2599 (2017).
42. Onizawa, N. *et al.* Sparse random signals for fast convergence on invertible logic. *IEEE Access* **9**, 62890–62898 (2021).
43. Inagaki, T. *et al.* A coherent ising machine for 2000-node optimization problems. *Science* **354**, 603–606 (2016).
44. Onizawa, N., Kuroki, K., Shin, D. & Hanyu, T. Local energy distribution based hyperparameter determination for stochastic simulated annealing. *Science* **2304**, 11839 (2023).
45. Cook, C., Zhao, H., Sato, T., Hiromoto, M. & Tan, S.X.-D. GPU-based ising computing for solving max-cut combinatorial optimization problems. *Integration* **69**, 335–344 (2019) https://www.sciencedirect.com/science/article/pii/S0167926019301348.
46. Tiunov, E. S., Ulanov, A. E. & Lvovsky, A. I. Annealing by simulating the coherent ising machine. *Opt. Express* **27**, 10288 (2019).
47. Yavorsky, A., Markovich, L. A., Polyakov, E. A. & Rubtsov, A. N. Highly parallel algorithm for the ising ground state searching problem. *Opt. Express* **1907**, 05124 (2019).

Acknowledgements

This work was supported in part by JST CREST Grant Number JPMJCR19K3, and JSPS KAKENHI Grant Number JP21H03404.

Author contributions

N.O. conducted and analyzed the experiments. T.H. discussed the experiment. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to N.O.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.