Quantized Simulated Bifurcation for the Ising Model

Tingting Zhang and Jie Han

***Abstract*— Developed from the Ising model, Ising machines are promising to be efficient domain-specific accelerators for solving combinatorial optimization problems (COPs). A simulated bifurcation (SB) algorithm enables an Ising machine to achieve massive parallelism in the simulation of Hamiltonian dynamics. Although SB significantly accelerates the search, more resources are required due to the use of continuous variables for the position of oscillators to obtain discrete spin states, compared to conventional simulated annealing. This article proposes ternary and multiple-value quantized SB (qSB) algorithms by discretizing the position variables used for the hardware-consuming multiply-accumulate (MAC) operations in SB. These quantization schemes do not only reduce the computational complexity, but also improve the solution quality for COPs. Specifically, the ternary qSB with dynamic threshold settings converts the MAC into addition and subtraction. To improve the precision in number representation when solving large-scale COPs, a uniform quantization scheme is applied to provide multiple-valued quantization. Alternatively, a logarithmic qSB leverages the evolution characteristics of position variables and implements multiplication by using simple shift operations. We demonstrate that using the proposed qSB improves the solution quality in a long search and accelerates energy convergence in a short search for solving COPs tackled by up to 2000 fully connected spins.**

I. Introduction

The performance improvement of general-purpose processors is slowing down as the feature size of transistors is reaching the physical limit. Domain-specific hardware accelerators based on unconventional computing paradigms are expected to lead to additional performance and efficiency gains for a particular class of applications. Combinatorial optimization problems (COPs) exist in various data-intensive applications, such as cell placement, wire routing, and logic minimization in very large-scale integration designs [1]. They are computationally non-deterministic polynomial-time (NP)-hard to solve by using an enumeration method on a general-purpose processor. A domain-specific computing architecture provides a novel paradigm for solving COPs in the post-Moore era.

The Ising model describes the spin dynamics in a ferromagnet, where spins naturally orient, leading the system to converge to the lowest energy state. Many COPs can be mapped to a problem of searching for the ground state of the Ising model, referred to as an Ising problem [2]. Various types of domain-specific hardware platforms for efficiently solving Ising problems, called Ising machines, are broadly categorized into two classes. One utilizes physical models, including quantum annealers with superconducting circuits [3], coherent Ising machines with pulse lasers [4] and oscillator-based Ising machines [5]. The other class is implemented by conventional circuits for various heuristic algorithms to numerically simulate spin dynamics, such as simulated annealing (SA) [6], stochastic cellular automata annealing [7] and simulated bifurcation (SB) [8]. These heuristic algorithm-based machines support solving Ising problems with dense spin-to-spin interactions with a high precision, which however may be challenging for the implementation of the first class of Ising machines.

Inspired by quantum mechanics, SB originates from simulations of Hamiltonian dynamics with quantum adiabatic bifurcation in a nonlinear oscillator network [8]–[10]. Compared with the commonly used SA, it does not require a one-by-one spin update to guarantee energy convergence. It benefits from the massive parallelism in updating spin states, thus reducing the search time. However, SB uses continuous variables for oscillator positions to obtain spin states (-1 or $+1$). Thus, it incurs a high hardware cost for computation, especially for the multiply-accumulate (MAC) operation.

Quantization reduces the precision of input samples into discrete levels, which is a lossy process by using less memory space. Quantization of the position values for the MAC can reduce the hardware for computation and also introduce some unexpected noise. This noise is likely to bring beneficial randomness to help the system jump out of the local minima, but at a risk of dramatic loss in solution quality due to the possibility of moving to another local minimum with higher energy [11]. The recently developed discrete SB (dSB) binarizes the position values by using their signs for MAC [11]. It achieves a significant reduction of computational complexity, but at the cost of a longer search time than the original SB [8]. Therefore, it becomes interesting to investigate the effect of quantizing the position values in SB for a better trade off between the computational cost and search time.

In this paper, quantized SB (qSB) is proposed for solving COPs quickly in a short search and also accurately in a long search. Different quantization schemes are applied for position values as inputs to the MAC in SB. The ternary qSB quantizes the position values into $\{0, \pm 1\}$, depending on a dynamic threshold that changes with time. It does not only convert the costly MAC into addition and subtraction, but also compresses the data by quantizing a large number of values to zero. Multiple-value qSB algorithms based on uniform and logarithmic quantizations are further

*This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Numbers: RES0048688, RES0051374 and RES0054326). T. Zhang was supported by a Ph.D. scholarship from the China Scholarship Council.

T. Zhang and J. Han are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada (ttzhang@ualberta.ca, jhan8@ualberta.ca).

developed to achieve a more accurate number representation for solving large-scale COPs. Especially, logarithmic qSB simplifies multiplication to shift operations. It produces more accurate computation results for updating variables near the bifurcation point and introduces randomness when the magnitudes of position values become larger during the search. The performance of the proposed qSB is evaluated by solving instances of Ising problems with up to 2000 fully and sparsely connected spins.

In the remainder of this paper, Section II presents preliminaries. The SB algorithms with quantization schemes are discussed in Section III. The experiment results are reported in Section IV. Finally, Section V concludes the paper.

II. Preliminaries

The Ising model is an abstract mathematical model to describe the ferromagnetism of magnetic particles interacting with each other in the presence of an external magnetic field in statistical physics. An N-spin Ising problem is to find the spin states that minimize the total energy (i.e., Hamiltonian) of the Ising model. The Hamiltonian is given by [2]

$$H(\mathbf{s}) = -\frac{1}{2}\sum_{i,j}^{N} J_{ij}s_is_j - \sum_{i}^{N} h_is_i, \tag{1}$$

where s_i (or s_j, $\in \{-1,+1\}$) denotes the state of the ith (or jth) spin, J_{ij} indicates the magnetic interaction strength between s_i and s_j (so $J_{ij} = J_{ji}$ and $J_{ii} = 0$), and h_i is the external magnetic field (or bias) placed on s_i.

SB was derived from a quantum bifurcation machine [9]. It applies the bifurcation phenomena, adiabatic processes, and ergodic processes to solving COPs. An SB algorithm essentially searches for an approximate solution by solving a pair of differential equations related to the positions and momenta of oscillator networks [8]. To restrain the errors introduced by using continuous position variables to represent discrete spin states, two variants, called ballistic SB (bSB) and dSB [11] are developed. Both bSB and dSB follow the Hamiltonian equations of motion, given by [11]

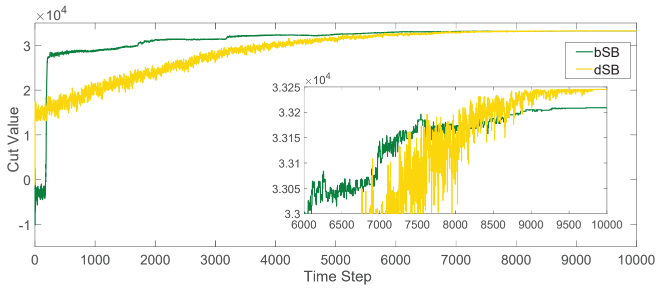
$$\dot{x_i} = a_0y_i, \tag{2}$$

$$\dot{y_i} = -\{a_0 - a(t)\}x_i + c_0JX_i + \eta(t)h_i, \tag{3}$$

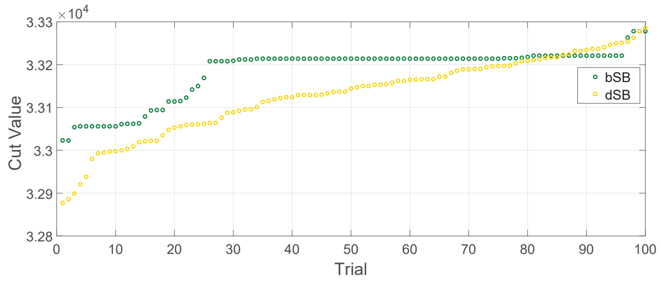
where x_i and y_i are the position and the momentum of the ith oscillator, respectively; $\dot{x_i}$ and $\dot{y_i}$ denote their derivatives with respect to time; a_0 and c_0 are manually tuned constants; $a(t)$ and $\eta(t)$ are time-dependent control parameters to guarantee the adiabatic evolution; $\boldsymbol{JX}$ is the computation related to the interactions among spins ($\boldsymbol{J}$) and position variables ($\boldsymbol{x}$), where JX_i is used for updating y_i. x_i is replaced by its sign (denoted by $sgn(x_j)$) and y_i is reset to 0 when $|x_i| > 1$. The bSB and dSB differ from the expressions of JX_i, given by [11]

$$JX_i = \begin{cases} \sum_{j=1}^{N} J_{ij}x_j & \text{in bSB} \\ \sum_{j=1}^{N} J_{ij}sgn(x_j) & \text{in dSB} \end{cases}. \tag{4}$$

To solve an Ising problem using SB, positions are first randomly initialized. Then, the semi-implicit Euler method is usually utilized to solve a pair of differential equations in (2) and (3) with a time step δ_t. At the end of a search, the sign of x_i indicates the state of the ith spin.



(a) Time evolutions of cut values with the time step in a single trial



(b) Distribution of cut values in multiple trials

Fig. 1: Solving a 2000-spin fully connected Ising problem K_{2000} by using bSB and dSB. Parameters are $\delta_t = 0.5$, $a_0 = 1$, and $c_0 = 0.01$, $a(t)$ linearly increases from 0 to 1. A fast convergence in Hamiltonian (indicated by the quickly increasing cut values) is preferred for accelerating the search. The fluctuations in Hamiltonian (indicated by the instability of cut values) are beneficial for jumping out of the local minima in a long search.

The max-cut problem (MCP) is to split N vertices in an undirected weighted graph into two groups (G_0 and G_1) to maximize the cut value (f_{MCP}), which is the total weight of edges between G_0 and G_1. This is formulated as [2]

$$f_{MCP} \quad = \frac{1}{2}\sum_{i,j}^{N} w_{ij}\frac{1-s_is_j}{2} = -\frac{1}{2}H_{MCP} + \frac{1}{4}\sum_{i,j}^{N} w_{ij}, \tag{5}$$

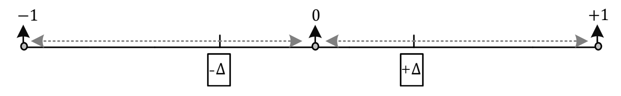
where s_i (or s_j) is a state variable for the ith (or jth) vertex that belongs to G_0 (as -1) or G_1 (as $+1$), and w_{ij} ($= w_{ji}$) denotes the edge weight between vertices i and j.

As a typical COP, the MCP is equivalent to an N-spin Ising problem with the Hamiltonian (H_{MCP}) given by [2]

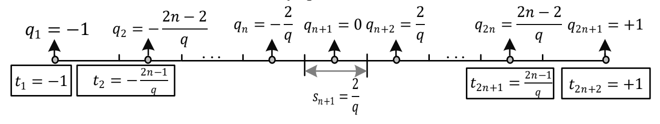
$$H_{MCP} \quad = -\frac{1}{2}\sum_{i,j}^{N} J_{ij}s_is_j, \tag{6}$$

where $J_{ij} = -w_{ij}$.

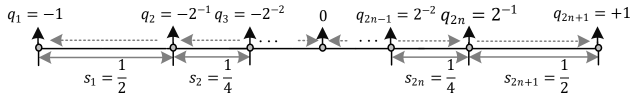
Fig. 1 presents instances of solving a benchmark MCP K_{2000} by bSB and dSB, which is an Ising problem with 2000 fully connected spins [7], [8]. Fig. 1(a) shows the cut values with time evolutions when using $T_s = 10^4$ time steps (i.e., iterations). Compared with the dSB, the cut value quickly increases by using bSB at the beginning of the search. Near the end of the search, the cut value remains nearly unchanged because the spin states in bSB become stable. However, the cut value obtained by using dSB fluctuates even at the end, which increases the probability of jumping out of the local minima.



(a) Ternary quantization in tSB



(b) Uniform multiple-valued quantization in uSB



(c) Logarithmic multiple-valued quantization in lSB

Fig. 2: Quantization schemes in SB algorithms.

Fig. 1(b) presents the distribution of cut values in 100 trials. Each trial adopts $T_s = 10^3$ time steps and differs in the random initialization of position values. For bSB, the majority of trials provides a large cut value between 33200 and 33300, which shows the general effectiveness of bSB. As indicated by a broader distribution of cut values, the dSB yields an unstable solution quality, but it is more likely to find a better solution (with a larger cut value).

Unlike those gradient method-based Ising machines, such as SimCIM [12], which utilizes a stochastic search process, the SB algorithm uses a deterministic process except for the random initialization of positions and momenta. For bSB, the energy is converged to a local minimum in a short search, while the binarized x_i used as the inputs to the MAC in dSB, as in (4), may introduce beneficial randomness to prevent the Ising model from being stuck at the local minima in a long search. In this work, we apply quantization schemes for the MAC in SB to not only achieve fast energy convergence in a short search but also increase the probability of jumping out of the local minima in a long search.

III. Quantized Simulated Bifurcation

Depending on the sign bit, dSB binarizes x_j to $\{-1, 1\}$ when computing JX_i, as in (4). Thus, the MAC is simplified to addition and subtraction, so dSB achieves a significant reduction in hardware. However, as discussed in Section II, it results in an unstable solution quality and relatively slow convergence in energy. In this section, we improve the SB with different quantization methods for the position values used as inputs to the MAC for a better trade-off between solution quality and computational complexity.

A. Ternary Simulated Bifurcation

The ternary SB (tSB) utilizes three-valued position variables (such as $x_j \in \{-1, 0, 1\}$, where $j \in [1, N]$) to compute $\boldsymbol{JX}$. Similar to dSB, the computation of $\boldsymbol{JX}$ avoids using multipliers. Moreover, $\boldsymbol{x}$ is compressed as a result of a high probability of quantizing position values to zeros.

Let $\hat{\boldsymbol{x}}$ be the quantized position values. As shown in Fig. 2(a), a dynamic threshold Δ determines the mapping from $\boldsymbol{x}$ to $\hat{\boldsymbol{x}}$. In this way, JX_i is computed as

$$JX_i = \textstyle\sum_{j=1}^{N} J_{ij} tri(x_j), \quad (7)$$

where

$$tri(x_j) = \begin{cases} 0 & |x_j| \le \Delta \\ sgn(x_j) & others \end{cases} . \quad (8)$$

An empirically optimized value for Δ when using the ternary quantization method is $\Delta^* = \frac{0.7\|\boldsymbol{x}\|_{l1}}{N}$ to minimize the Euclidean distance between $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}$ [13].

To compute Δ^*, the L1 norm of $\boldsymbol{x}$ needs to be evaluated and it requires the accumulation of N elements in $\boldsymbol{x}$ in each time step, thus incurring a high cost. The elements in $\boldsymbol{x}$ evolve from random values around 0 to $+1$ or -1. Therefore, Δ^* increases from an extremely small positive value to around 0.7 with time. We then use a less computationally expensive function related to time $\Delta(t)$ to simulate the function of Δ^*. Linear, exponential, and logarithmic increasing functions are considered, given by

$$\Delta(t) = \begin{cases} 0.7 \times \frac{t}{T_s} & \text{linear} \\ 0.7 \times 2^{\frac{t}{T_s}-1} & \text{exponential} \\ 0.7 \times log_2(\frac{t}{T_s}+1) & \text{logarithmic} \end{cases} , \quad (9)$$

where t and T_s denote the current time step and the total number of time steps, respectively.

B. Uniformly Quantized Simulated Bifurcation

Multiple-valued quantization is further considered to improve computational accuracy. It is relatively costly to use a dynamic threshold to identify different quantization levels. Thus, a fixed threshold is used in the multiple-value qSB. Assume that the position values are quantized to q $(= 2n+1)$ levels. The gap between two adjacent levels is called the quantization step size. Let q_l be the quantized value of the lth level of the quantization intervals from t_l to t_{l+1}. Its step size is denoted by s_l $(= t_{l+1} - t_l)$.

As shown in Fig. 2(b), uniform quantization shares the same step size s_l $(= \frac{2}{q})$ in different quantization levels. To simplify the computation, the position variables in the 1st and $(2n+1)$th quantization levels, i.e., when $\frac{2n-1}{q} < |x_j| \le 1$, are quantized to $sgn(x_j)$. Except for these two levels, for the lth level, q_l is given by the middle value in $[t_l, t_{l+1})$, as $q_l = -1 + \frac{2l-1}{q}$ to maintain the accuracy. In this way, x_j is quantized to $\{0, \pm\frac{2}{q}, ..., \pm\frac{2n-2}{q}, \pm 1\}$. Thus, assuming $p_{x_j} = round(\frac{q|x_j|}{2})$, where $round()$ changes a value to its nearest integer, the q-level uniform qSB (uSB) computes JX_i as

$$JX_i = \textstyle\sum_{j=1}^{N} J_{ij} u(x_j), \quad (10)$$

where

$$u(x_j) = \begin{cases} sgn(x_j) \cdot \frac{2p_{x_j}}{q} & p_{x_j} <= n-1 \\ sgn(x_j) & others \end{cases} . \quad (11)$$

C. Logarithmic Quantized Simulated Bifurcation

The bifurcation that occurs at the beginning of a search plays an important role in SB. Thus, the computation with a high precision is preferred for small position values. When the magnitudes of position values become larger with time, it will be harder to change their sign bits, which makes the system vulnerable to local minima. Therefore, less accurate computation for positions with relatively large absolute values may help the system search for a better solution.

Logarithmic quantization [14] is then applicable in the multiple-value qSB. Different from uniform quantization, logarithmic quantization results in an exponential difference in s_l and s_{l+1}. The two quantized values are uniformly distributed in the base 2 logarithmic domain. In this way, as presented in Fig. 2(c), the q-level logarithmic qSB (lSB) quantizes x_j into $\{0, \pm 2^0, \pm 2^{-1}, ..., \pm 2^{-(n-1)}\}$, so the multiplications in $\boldsymbol{JX}$ are realized directly through shift operations with a wide numerical representation range for x_j. Assume $|x_j| = 2^{\widetilde{x_j}}$, then $\widetilde{x_j} = \lceil log_2|x_j| \rceil$. In the q-level lSB, JX_i is computed by

$$JX_i = \textstyle\sum_{j=1}^{N} sgn(x_j)s(J_{ij}, \widetilde{x_j}), \qquad (12)$$

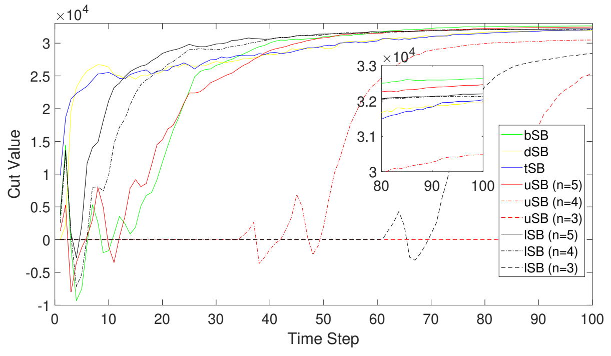
where $s()$ right shifts the binary representation of J_{ij} by $|\widetilde{x_j}|$ bits, when $\widetilde{x_j} > -n$; otherwise $s()$ outputs 0.

IV. Experimental Results

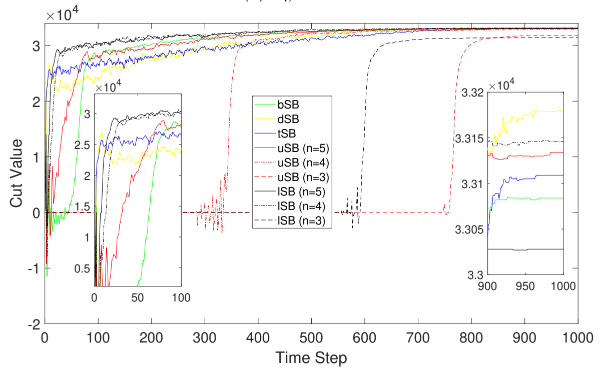
A. Time Evolutions of tSB, uSB and lSB

Fig. 3 compares the time evolutions of Hamiltonian when using tSB, uSB, and lSB on solving the K_{2000} benchmark in a single trial. Since tSB using different kinds of Δ (Δ^* and $\Delta(t)$) shows a similar pattern in energy convergence, the tSB with Δ^* is considered as an example. The obtained solution quality partly depends on the random initialization, which can not be accurately evaluated by one trial experiment. Thus, instead of evaluating the solution quality at the end of a search, this section discusses the convergence and fluctuation in Hamiltonian in short and long searches.

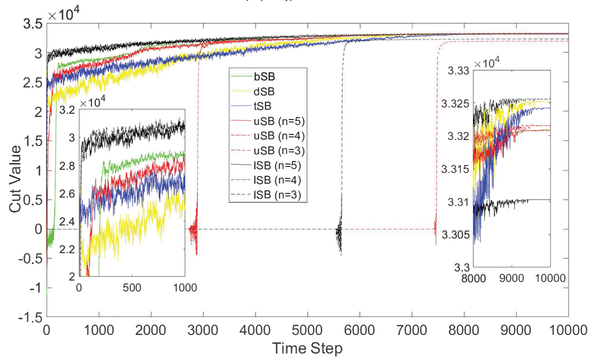
The tSB shows a similar tendency in energy convergence as dSB due to their similar mechanism. At the beginning of the search, the cut value obtained by using tSB increases more quickly than using dSB when $T_s = 1000$ and 10000. Near the end of the search, the tSB has a stronger fluctuation, which indicates that the tSB is more likely to find a better solution. The uSB with $n = 3, 4$ and the lSB with $n = 3$ lead to a slow convergence at the beginning due to the lack of ability to recognize the small position values (quantized to zeros) when computing $\boldsymbol{JX}$. Benefiting from the wide range of numerical representation, the lSB with $n \geq 4$ increases the cut value quickly at the beginning, whereas the uSB requires $n \geq 5$ for a quick convergence. The uSB with $n = 5$ performs similarly as bSB. Although bSB sharply increases the cut value at the beginning, it cannot continuously decrease the energy throughout the iterations, especially for a large number of iterations. However, in the uSB, the randomness introduced by quantization makes the system transverse more local minima. Thus, the energy still shows a decreasing tendency even near the end of the search. The lSB with $n \geq 4$ shows a better convergence compared with other SB algorithms. It decreases the energy in a short time at the beginning of a search and maintains a slight fluctuation at the end.



(a) $T_s = 100$



(b) $T_s = 1000$

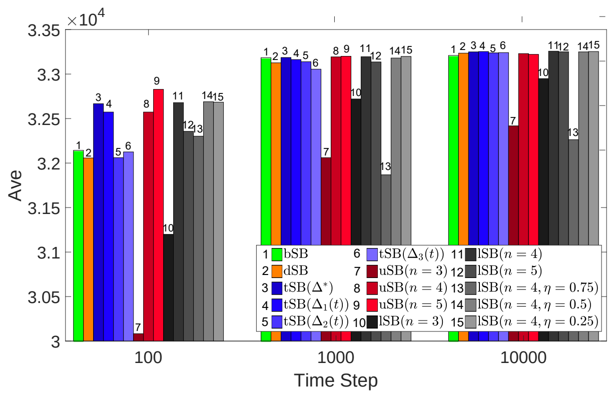


(c) $T_s = 10000$

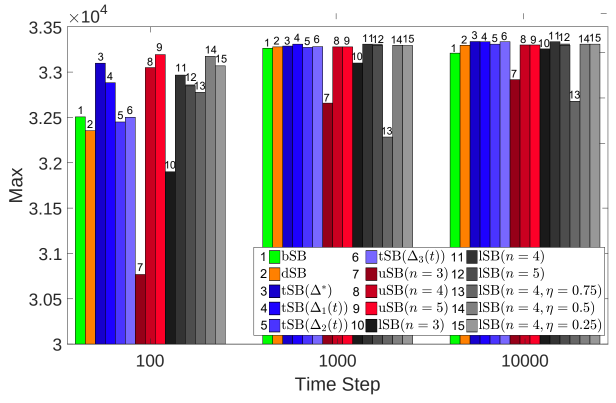
Fig. 3: Time evolutions of cut values in a single trial on the K_{2000} benchmark by using SB algorithms with different time steps T_s.

B. Solution Quality of using tSB, uSB and lSB

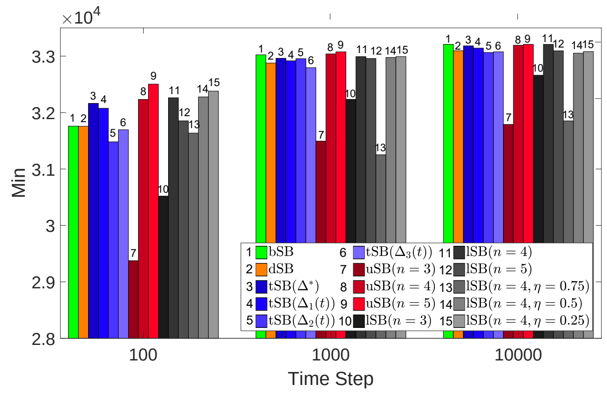
Fig. 4 illustrates the solution quality on the K_{2000} benchmark by using the proposed qSB, bSB, and dSB algorithms from 100 trials. Compared with the dSB, a higher solution quality (in Ave, Max and Min) can be obtained by using the bSB due to the fast convergence when $T_s = 100$. However, with the increase of T_s, the randomness from the initialization has less effect on energy convergence than the influence from $\boldsymbol{JX}$. It is vulnerable for bSB to be stuck on the local minima. Therefore, when $T_s = 10000$, the standard deviation (Std) obtained by using bSB becomes 0. Due to the computational accuracy loss from the binarized position values used for computing $\boldsymbol{JX}$, the dSB struggles to achieve a large cut value in a short time. However, the dSB shows its advantage in a long search, as discussed in Section II.



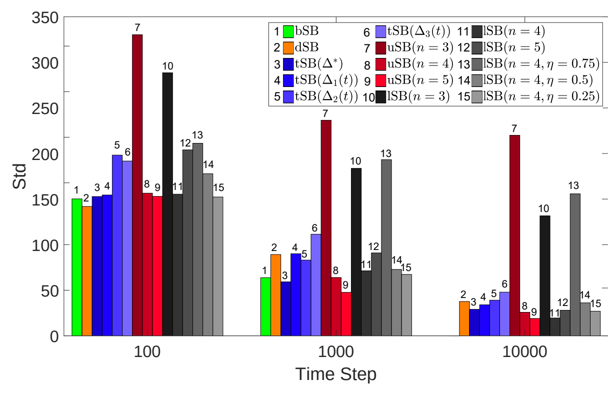
(a) The average (Ave) of the obtained cut values.



(b) The maximum (Max) of the obtained cut values.



(c) The minimum (Min) of the obtained cut values.



(d) The standard deviation (Std) of the obtained cut values.

Fig. 4: Solution quality (in the statistics of cut values from 100 trials) of using qSB, bSB, and dSB algorithms to solve the K_{2000} benchmark with T_s = 100, 1000, and 10000. When a single SB machine is used to search for the solution, larger Ave and Min values are preferable. When multiple SB machines are used to search for solutions, higher Max and Std values are preferable because the best one can be selected as the result.

TABLE I: Summary of Simulated Bifurcation Algorithms

SB Algorithms	**Complexity of JX**				**Solution Search**	
	Compression		**Operation**		**Time**	**Structure**
bSB [8]	N/A		MUL	H	SS	SM
dSB [8]	N/A		SIN	L	LS	MM
tSB	DYN	H	SIN	L	SS, LS	SM, MM
uSB	STA	M	MUL	M	SS, LS	SM, MM
lSB	STA	L	SHIF	L	SS, LS	SM, MM

Compression: $\boldsymbol{x}$ is dynamically (DYN) or statically (STA) compressed. Operation: $J_{ij}x_j$ in $\boldsymbol{JX}$ is implemented by multiplication (MUL), sign conversion (SIN), or shift (SHIF) operations. "H", "M", and "L" indicate that the compression degree or the complexity of the operation is high, moderate, or low. Time: the algorithm is suitable for a short search (SS) or a long search (LS). Structure: the algorithm is suitable for a single SB machine (SM) or multiple SB machines (MM) for choosing the best solution.

In the tSB, linear, exponential and logarithmic $\Delta(t)$ are denoted by $\Delta_1(t)$, $\Delta_2(t)$ and $\Delta_3(t)$, respectively. For $T_s =$ 1000 and 10000, compared with using Δ^*, using $\Delta_1(t)$ obtains a larger Max, Std and a smaller Min. It indicates that using $\Delta_1(t)$ allows the system to transverse more energy states, especially for a large number of time steps. For $T_s = 100$, the tSB with $\Delta_1(t)$, denoted by tSB1, can find a better solution than bSB. Different from bSB, with the increase of T_s, tSB1 can achieve a competitive cut value with dSB. For the multiple-value qSB, when $n \geq 4$, both uSB and lSB obtain a better solution quality than dSB and bSB. For the uSB, as indicated by the evolutions of cut values in Fig. 3, the solution quality is improved with an increased n. Compared with lSB, uSB performs better when T_s is small. The uSB with $n = 5$, namely uSB5, produces improvements over bSB for $T_s = 100$. The lSB shows its potential of finding a better solution when using a large T_s. Unlike uSB, the increase of n in lSB does not always lead to a quality improvement. The lSB with $n = 4$, namely lSB4, outperforms the lSB with $n = 5$.

Table I summarizes the characteristics of different SB algorithms. The proposed qSB can compress the data by quantizing a group of position values to zeros. The position values are highly compressed in the tSB, but the dynamic threshold will incur additional overhead in implementation. The tSB, dSB, and lSB significantly simplify the multiplication in $\boldsymbol{JX}$. Compared with bSB, the design of multipliers can be customized in uSB for the quantized position values to improve hardware efficiency. Moreover, the proposed qSB can obtain a good solution in a short search, and also jump out of the local minima for energy convergence in a long search.

The n in lSB plays a similar role as Δ in tSB, which determines the numerical range of position values to be quantized to 0. Since the position values evolve with time, the precision requires to be higher at the beginning. A small value of n may lead to a large number of x_j being quantized to 0 at the beginning, thus it is difficult to accurately evolve

TABLE II: The Values of P_g and S_g for the Max-cut Problems on 2000-node Fully and Sparsely Connected Graphs

Vaules of P_g and S_g with T_s		$G22$					K_{2000}				
		bSB	**dSB**	**tSB1**	**uSB5**	**lSB4**	**bSB**	**dSB**	**tSB1**	**uSB5**	**lSB4**
$\boldsymbol{T_S = 100}$	$P_{99\%}$	33%	2%	7%	25%	13%	0	0	3%	12%	6%
	$S_{99\%}$	1150	22794	6345	1600	3306	-	-	15119	3602	7442
$\boldsymbol{T_S = 1000}$	$P_{99.5\%}$	86%	82%	85%	90%	86%	75%	0	55%	83%	70%
	$S_{99.5\%}$	2342	2685	2427	2000	2342	3321	-	5767	2598	3824
$\boldsymbol{T_S = 10000}$	$P_{99.9\%}$	0	64%	84%	63%	71%	0	0	15%	4%	12%
	$S_{99.9\%}$	-	45075	25129	46317	37202	-	-	283362	1128110	360247

* G22: 2000 nodes, 19990 edges, a random graph, edge weight $w_{ij} \in \{0, +1\}$, best-known cut value: 13359;
K_{2000}: 2000 nodes, 1999000 edges, a complete graph, edge weight $w_{ij} \in \{-1, +1\}$, best-known cut value: 33337.

the position values with time. Moreover, in lSB, no matter what n is, as long as the position value becomes larger than 0.5, it is considered as 1. Therefore, dynamic quantization is considered that uses a high-level (9-level) lSB at the beginning and the simple 3-level lSB near the end. We use η to indicate the percentage of time steps that adopts the 3-level lSB. When $\eta \geq 0.5$, the solution quality is similar to or even higher than only using lSB4. It indicates that using a dynamic quantization scheme can accelerate search and improve solution quality.

C. Performance Comparison

We further consider the metrics of probability-to-target (P_g) and step-to-target (S_g) for the evaluation of SB algorithms [15]. P_g is computed by dividing the number of trials required to obtain the target solution by the total number of trials, where g indicates the quality of the target solution. For example, $P_{99.9\%}$ gives the probability of obtained solutions reaching 99.9% of the best-known value. S_g estimates the number of time steps required to find the target solution with a probability of 99%, given by $S_g = T_s \frac{log0.01}{log(1-P_g)}$, where T_s is the total number of time steps.

Table II compares qSB (tSB1, uSB5, lSB4) with the bSB and dSB in terms of P_g and S_g on MCPs with 2000-node graphs, K_{2000} and $G22$ dataset from the $Gset$ benchmark [16] in 10^3 trials. For the $G22$ benchmark, bSB can quickly find a good solution, and dSB is better suited for reaching a high solution quality by a long search. Compared with dSB, the proposed qSB algorithms perform better for both long and short searches in most cases. Moreover, they also perform similarly as bSB for a short search. Due to the massive node connectivity in the K_{2000}, SB requires a larger number of time steps to obtain a good solution. For $T_s = 10000$, it is difficult for bSB and dSB to reach 99.9% of the best-known cut value, whereas the proposed qSB can find a better solution due to their ability to jump out of the local minima. It indicates that the proposed qSB can obtain 99.9% of the best-known cut value with fewer time steps by up to an order of magnitude.

V. Conclusion

This paper investigates efficient quantization schemes in simulated bifurcation (SB) Ising machines for fast and accurate large-scale combinatorial optimization. Specifically, various quantized SB (qSB) algorithms are proposed. A ternary qSB algorithm dynamically ternarizes the position values for the MAC by introducing a linearly increasing threshold. It does not only compress data by using zero elements to represent a group of position values, but also reduces computational complexity by converting MAC into addition and subtraction. Multiple-value qSB algorithms are further developed by using uniform and logarithmic quantizations to improve the precision. The qSB algorithms realize fast energy convergence and increase the probability of jumping out of the local minima. The experiments on 2000-spin Ising problems show that they are up to an order of magnitude faster than a recently developed SB algorithm to obtain a similar solution quality. This study also shows the feasibility of dynamic quantization in the development of energy-efficient fully connected parallel Ising machines.

References

[1] K. Tanahashi, S. Takayanagi, T. Motohashi, and S. Tanaka, "Application of Ising machines and a software development for Ising machines," *JPSJ*, vol. 88, no. 6, p. 061010, 2019.

[2] A. Lucas, "Ising formulations of many NP problems," *Front. Phys.*, 2014.

[3] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[4] Z. Wang, A. Marandi, K. Wen, R. L. Byer, and Y. Yamamoto, "Coherent Ising machine based on degenerate optical parametric oscillators," *Phys. Rev. A*, 2013.

[5] J. Chou, S. Bramhavar, S. Ghosh, and W. Herzog, "Analog coupled oscillator based weighted Ising machine," *Sci. Rep.*, vol. 9, no. 1, pp. 1–10, 2019.

[6] R. A. Rutenbar, "Simulated annealing algorithms: An overview," *IEEE Circuits Syst. Mag.*, 1989.

[7] K. Yamamoto, K. Kawamura, K. Ando, N. Mertig, T. Takemoto, M. Yamaoka *et al.*, "STATICA: A 512-spin 0.25 m-weight annealing processor with an all-spin-updates-at-once architecture for combinatorial optimization with complete spin–spin interactions," *JSSC*, 2020.

[8] H. Goto, K. Tatsumura, and A. R. Dixon, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems," *Sci. Adv.*, vol. 5, no. 4, p. eaav2372, 2019.

[9] H. Goto, "Bifurcation-based adiabatic quantum computation with a nonlinear oscillator network," *Sci. Rep.*, vol. 6, no. 1, pp. 1–8, 2016.

[10] T. Zhang and J. Han, "Efficient traveling salesman problem solvers using the Ising model with simulated bifurcation," in *DATE*. IEEE, 2022, pp. 548–551.

[11] H. Goto, K. Endo, M. Suzuki, Y. Sakai, T. Kanao, Y. Hamakawa *et al.*, "High-performance combinatorial optimization based on classical mechanics," *Sci. Adv.*, 2021.

[12] E. S. Tiunov, A. E. Ulanov, and A. Lvovsky, "Annealing by simulating the coherent Ising machine," *Optics express*, vol. 27, no. 7, pp. 10 288–10 295, 2019.

[13] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[14] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "Lognet: Energy-efficient neural networks using logarithmic computation," in *ICASSP*. IEEE, 2017, pp. 5900–5904.

[15] T. Kanao and H. Goto, "Simulated bifurcation assisted by thermal fluctuation," *Commun. Phys.*, vol. 5, no. 1, p. 153, 2022.

[16] C. Helmberg and F. Rendl, "A spectral bundle method for semidefinite programming," *SIAM J. Optim.*, vol. 10, no. 3, pp. 673–696, 2000.