

Beyond gradient descent: Adam for analog Ising machinesStijn Van Vooren ^{*}, Guy Van der Sande , and Guy Verschaffelt *Applied Physics Research Group, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium*

(Received 29 May 2026; accepted 27 August 2026; published 13 October 2026)

As Moore's law reaches its limits, Ising machines offer a promising alternative computing approach for difficult optimization problems. However, many analog, time-continuous Ising machines rely on gradient-descent-like dynamics to find solutions, which can limit speed and robustness. We investigate whether momentum and Adam optimization can improve these systems. Since these optimizers are traditionally formulated in discrete time, we derive continuous-time versions suitable for analog, time-continuous Ising-machine dynamics. On Max-Cut benchmarks, we find that Adam-based dynamics substantially reduce time to target and improve solution quality compared with gradient-descent- and momentum-based dynamics. We further introduce a first-order continuous-time approximation of Adam that is intended as a simpler starting point for future physical implementations and while performing better than the full Adam formulation in a continuous-time setting. We also study a purely algorithmic discrete-time setting, where the performance gap is reduced on easier problem instances, while the Adam-based update rule performs best on harder weighted problem instances. These results identify continuous-time Adam dynamics as a powerful design principle for analog Ising machines.

DOI: [10.1103/qg6g-nkf3](https://doi.org/10.1103/qg6g-nkf3)**I. INTRODUCTION**

The miniaturization of transistors is approaching physical limits, making it increasingly more difficult and costly to continue the exponential growth of computing power predicted by Moore's Law [1,2]. As a result, the focus is shifting towards enhancing computational performance through software, algorithms, and innovative hardware architectures [3]. In this evolving landscape, Ising machines have emerged as a promising non-von Neumann computing approach to address NP-hard optimization problems, statistical sampling and machine learning more efficiently than traditional digital computers [4–7].

Ising machines are physical or algorithmic systems based on the Ising spin model from statistical physics. They can be used for optimization, sampling, and machine learning; here we focus on binary optimization, where the goal is to minimize a cost function. In this setting, the problem is mapped [8] onto the Ising Hamiltonian by choosing the couplings J_{ij} and bias terms b_i such that low-energy spin configurations correspond to low-cost solutions:

$$H_{\text{Ising}}(\sigma) = -\frac{1}{2} \sum_{i,j} J_{ij} \sigma_i \sigma_j - \sum_i b_i \sigma_i, \quad (1)$$

where $\sigma = \{\sigma_1, \sigma_2, \dots, \sigma_N\}$ denotes the state of the system, $\sigma_i \in \{-1, +1\}$ are binary spin variables, J_{ij} are the couplings

between spins, b_i are the bias terms, and N is the number of spins.

To situate our work more precisely, it is useful to categorize Ising machines by how the spin state is represented. In analog Ising machines, each spin is stored internally as a continuous variable x_i , and the binary Ising spin $\sigma_i \in \{-1, +1\}$ is obtained only after thresholding, e.g., through $\sigma_i = \text{sign}(x_i)$. In binary Ising machines, the hardware state itself is already binary. Both categories exist in, e.g., electrical and optical hardware [6,9–13].

Our work is aimed at analog machines, where it is also useful to distinguish time-continuous realizations, whose states evolve continuously in time, from time-discrete realizations, where continuous internal variables are updated in clocked steps. This distinction is important in practice: in many time-discrete physical implementations, the evaluation of the coupling becomes a genuine bottleneck. In particular, the analog state must first be measured and converted by an analog-to-digital converter (ADC), then the coupling multiplication is carried out in digital hardware such as a field-programmable gate array (FPGA), and the result must finally be converted by a digital-to-analog converter (DAC) and reinjected into the physical system. This means that the machine is no longer operating at the native bandwidth of the analog substrate; instead, its update rate is constrained by the latency and throughput of the ADC/DAC and FPGA loop. By contrast, time-continuous analog Ising machines implement the spin coupling directly in the physical system itself, making them the natural setting when one wants to exploit the intrinsic parallelism and high bandwidth of analog hardware [5,7].

Representative examples help illustrate the different categories. Among analog, time-continuous Ising machines, electrical oscillator-network implementations relax toward low-energy Ising states through the continuous dynamics

^{*}Contact author: Stijn.Van.Vooren@vub.be

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

of nonlinear oscillators [9], while all-optical coherent Ising machines based on injection-locked multicore-fiber lasers realize the couplings with spatial light modulators, and relax through coupled laser dynamics [10]. Analog, time-discrete Ising machines also use continuous internal variables, but update them in synchronized discrete steps. Examples include simulated bifurcation, typically implemented in digital hardware [11,14], and optical measurement-feedback coherent Ising machines, which use time-multiplexed degenerate optical parametric oscillator (DOPO) pulses and compute the couplings once per cavity round trip using a central processing unit (CPU) or FPGA [6]. Binary Ising machines, by contrast, store the spin directly in bistable electrical or optical states; one large-scale optical example encodes spins in binary optical phases using spatial light modulation [13]. With this classification we want to make clear that our results are not tied to a single device, but to the broader class of analog Ising-machine architectures.

Many of these Ising machines rely heavily on gradient descent for their optimization. While gradient descent is a cornerstone of many optimization methods, it comes with notable challenges. One key limitation is that gradient descent can only move downhill in the energy landscape. This means it is prone to getting stuck in local minima, which are suboptimal solutions that hinder the system from reaching the global minimum—the true optimal solution. To remedy this issue, Ising machines typically incorporate noise into the system. This noise helps the system escape local minima by introducing stochastic fluctuations, allowing the machine to “jump” out of shallow wells and explore other regions of the energy landscape. While this approach can improve the system’s ability to find the global minimum, it also introduces challenges in terms of the required noise amplitude and the tradeoff between exploration and exploitation. Too much noise can cause the system to fluctuate too much, preventing convergence to an optimal solution, while too little noise might not help escape local minima effectively [15]. While gradient descent-based Ising machines have demonstrated success, they remain constrained by slow convergence and a tendency to get trapped in local minima. Momentum and Adam optimizers offer significant advantages over gradient descent. Momentum can be intuitively understood as the inertia of a moving object. When an optimizer uses momentum, it accumulates velocity from previous gradients, allowing it to continue moving in the same direction even when encountering uphill slopes (local minima). This inertia helps the optimizer to escape local minima and maintain its trajectory towards the global minimum. The Adam optimizer combines the benefits of momentum and adaptive learning rates, adjusting the step size based on the first and second moments of the gradients. This results in faster convergence and better handling of noisy gradients, making Adam-based Ising machines potentially more efficient and robust for combinatorial optimization tasks.

Momentum has been introduced in some Ising machine implementations [16]. Incorporating momentum dynamics into simulated bifurcation, the Toshiba Bifurcation Machine achieves improved optimization performance [11,12]. This technique is state of the art for Sherrington-Kirkpatrick (SK) optimization problems, according to recent benchmarks [17], while for other problem types the data are incomplete. While

momentum-based optimizers have been widely successful in combinatorial optimization problems, the Adam optimizer has dominated machine learning and other fields, particularly in deep learning, natural language processing, and computer vision, due to its superior convergence speed and robustness. Despite this widespread success, Adam optimizer had not been applied to Ising machines until Brown *et al.* [18] introduced them in the context of a continuous-variable coherent Ising machine (CIM). In their work, Brown *et al.* demonstrated that momentum and Adam can significantly accelerate the optimization process in CIMs by modifying the feedback with more sophisticated update rules. Their simulations showed that these methods improve convergence speed, sample diversity, and stability over standard gradient descent in CIMs, particularly for poorly conditioned problems. Notably, they found that Adam, with its adaptive learning rate adjustments, enhances robustness against variations in feedback strength, making it more reliable for practical implementations. However, their work is specifically done for the CIM system. Our work is inspired by this, but is substantially broader. Rather than adapting momentum and Adam to this specific CIM architecture, we reformulate them for the general class of analog, time-continuous Ising machines.

This distinction matters because Adam is normally used as a discrete-time optimization algorithm. In purely algorithmic implementations, and also in many time-discrete physical Ising machines where the state is measured, processed digitally, and fed back through electronics such as an FPGA, porting these optimizers is comparatively straightforward: one can directly use the standard discrete momentum or Adam update rules known from the literature. In such cases, the main computation, often dominated by evaluating the coupling, is carried out digitally, so the effective update rate is tied to the speed of the digital control loop rather than to the native bandwidth of the analog substrate. Our focus, instead, is on analog Ising machines whose states evolve directly in continuous time. For this class of systems, the standard discrete formulations of momentum and Adam optimizer are not directly applicable. In this work, we therefore derive continuous-time versions of both optimizers and use them to construct analog, time-continuous Ising-machine dynamics. While continuous-time momentum is well established, continuous-time formulations of Adam have also been developed in the optimization literature [19]. Our contribution here is to adapt this continuous-time perspective to the present Ising-machine setting, and to introduce a simplified first-order variant, which in our benchmarks performs even better than the full Adam-based formulation.

In addition, our work investigates a key aspect that was not explored by Brown *et al.*, namely, the impact of nonlinearity on performance. Ising machines rely on nonlinearity to enforce binary behavior, and different choices of nonlinear functions can drastically influence convergence speed and solution quality [20]. We systematically analyze how different nonlinearities affect the optimization dynamics. This provides insights into the interplay between nonlinearity and optimizer choice, further enhancing the efficiency of Ising machines across different analog, time-continuous implementations.

Other recent approaches act on complementary parts of the search. Simulated bifurcation (SB) evolves the positions and

momenta of a classical nonlinear-oscillator network [14]. Its adiabatic variant (aSB) follows classical adiabatic evolution using a common time-dependent bifurcation parameter [14]. Ballistic SB (bSB) replaces the nonlinear potential walls by perfectly inelastic walls [11], while discrete SB (dSB) replaces the continuous interaction variables of bSB by their signs to improve solution accuracy while retaining parallel updates [11]. Generalized ballistic simulated bifurcation (GbSB) [21] assigns an individual bifurcation parameter to each oscillator and controls these parameters nonlinearly so that oscillators are less likely to stick to the walls and become trapped in local minima. Its large increase in success probability occurs near the edge of chaos.

Chaotic amplitude control (CAC) [22,23] takes a different route. It introduces an auxiliary error variable for each spin that modulates the coupling strength according to the deviation from a target amplitude. This feedback corrects amplitude heterogeneity and destabilizes low-energy trapping states, producing a nonrelaxational chaotic search rather than simple descent toward a fixed point. The structured M -layer construction [24] instead acts on the problem representation: it creates M copies of the interaction graph and reconnects their edges between layers according to a structured mixing rule. This reshapes the global loop structure without changing the local interactions, smoothing the landscape by suppressing many suboptimal metastable states.

These approaches modify different parts of the search, while Adam changes how gradient information is accumulated and scaled. This makes Adam unusually modular: its adaptive update could in principle be combined with SB- or CAC-type dynamics, or applied to an M -layer representation. Adam is therefore attractive both as a standalone optimizer and as a building block for future hybrid Ising machines.

The remainder of this paper is organized as follows. We first place gradient-descent-based Ising machines in a general optimization framework and use this formulation to introduce momentum- and Adam-based dynamics. Since these optimizers are conventionally defined in discrete time, we recast them in continuous time so that they can be applied naturally to analog Ising machines, whose physical dynamics evolve continuously. In doing so, we also introduce a simplified first-order approximation of the continuous-time Adam dynamics, designed to be more suitable for physical implementation. We then benchmark the resulting Ising-machine models across several nonlinearities on standard Max-Cut problem instances, using time to target, solution quality, and success rate as performance measures. The benchmarks show that the first-order Adam-based dynamics, especially with the sigmoid nonlinearity, give large and consistent improvements over gradient-descent- and momentum-based Ising machines, both in convergence speed and in the quality of the solutions obtained. Finally, we also analyze a purely algorithmic discrete-time setting, where the continuous-time dynamics are implemented through numerical discretization and the time step is treated as an additional hyperparameter. The Results section concludes with a cross-study comparison of solution quality against published results for selected state-of-the-art Ising solvers on the common 51-instance GSET benchmark.

II. GRADIENT DESCENT-BASED ISING MACHINES (GD-IMS)

We begin with gradient descent-based Ising machines (GD-IM), which provide the starting point for the momentum- and Adam-based extensions introduced below. As discussed above, the discrete optimization problem is first mapped onto the Ising Hamiltonian of Eq. (1), so that the binary problem can be written as

$$\min_{\sigma \in \{-1, +1\}^N} \{f(\sigma) = H_{\text{Ising}}(\sigma)\}. \quad (2)$$

Solving binary optimization problems directly is computationally prohibitive due to the vast combinatorial solution space, making brute-force methods impractical to impossible. To address this, we leverage the analog nature of Ising machines by replacing the binary spins $\sigma_i \in \{-1, +1\}$ with continuous variables $x_i \in \mathbb{R}$. This turns the binary problem into the continuous optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^N} f(\mathbf{x}), \quad (3)$$

with

$$f(\mathbf{x}) = H_{\text{Ising,cont}}(\mathbf{x}) = -\frac{1}{2} \sum_{i,j} J_{ij} x_i x_j - \sum_i b_i x_i. \quad (4)$$

This continuous formulation allows us to use gradient-based dynamics, while the original binary configuration is recovered only at readout by taking $\sigma_i = \text{sign}(x_i)$. Henceforth, we work with the continuous variables $\mathbf{x} \in \mathbb{R}^N$ and reserve σ for the binary readout. For simplicity, we neglect external biases throughout this paper, setting $b_i = 0$. Continuous-time gradient descent-based Ising machines evolve to their lower energy states according to

$$\frac{dx_i}{dt} = -(\nabla f(\mathbf{x}))_i. \quad (5)$$

From Eq. (3), the gradient becomes

$$-(\nabla f(\mathbf{x}))_i = \sum_j J_{ij} x_j. \quad (6)$$

This formulation ensures that the system follows the negative gradient, decreasing the function value over time in the direction of steepest descent. In numerical simulations of continuous-time Ising machines this continuous evolution is discretized using, e.g., Euler's method.

For analog Ising machines, the continuous dynamics must remain consistent with the original binary problem, in which each spin can take only the values $+1$ or -1 . Since analog spin variables can take a continuous range of values, a nonlinearity is introduced to steer them toward binary states and thereby recover meaningful solutions to the original problem. Böhm *et al.* showed that the choice of nonlinearity can strongly influence the computational performance of analog Ising machines [20]. To examine this dependence systematically, we consider four nonlinear choices, following Böhm *et al.* [20], each of which modifies Eq. (6) so that the dynamics favor binary states. In all cases, α controls the local restoring force, β sets the coupling strength, and $\gamma \zeta_i(t)$ is a Gaussian white-noise term that can help the system escape shallow local minima.

With the *polynomial* choice, corresponding to the nonlinear term arising in coherent Ising machine (CIM) dynamics [20], the gradient becomes

$$-(\nabla f(\mathbf{x}))_i = (\alpha - 1)x_i - x_i^3 + \beta \sum_j J_{ij}x_j + \gamma \zeta_i(t), \quad (7)$$

which introduces a double-well structure in the energy landscape, which effectively deepens the energy landscape near binary-valued amplitudes and raises it for intermediate values. For the *sigmoid* nonlinearity, representative of neuron-inspired dynamics [20], one obtains

$$-(\nabla f(\mathbf{x}))_i = -x_i + \tanh \left(\alpha x_i + \beta \sum_j J_{ij}x_j + \gamma \zeta_i(t) \right), \quad (8)$$

where the hyperbolic tangent bends large inputs toward bounded values instead of letting them grow without limit. In this way, small inputs are treated almost linearly, while larger ones are gradually compressed. The *periodic* nonlinearity, associated with optoelectronic oscillator (OEO) implementations [20], leads to

$$-(\nabla f(\mathbf{x}))_i = -x_i + \cos^2 \left(\alpha x_i - \frac{\pi}{4} + \beta \sum_j J_{ij}x_j + \gamma \zeta_i(t) \right) - \frac{1}{2}, \quad (9)$$

so that the restoring term becomes periodic, creating a repeated pattern of preferred amplitude regions. Finally, for the *clipped* nonlinearity, corresponding to clipped-OEO dynamics [20], the gradient becomes

$$-(\nabla f(\mathbf{x}))_i = \begin{cases} (\alpha - 1)x_i + \beta \sum_j J_{ij}x_j + \gamma \zeta_i(t), & |x_i| \leq 0.4 \\ 0, & |x_i| > 0.4 \end{cases}, \quad (10)$$

which imposes a hard cutoff on the amplitude and thereby tends to homogenize the spin magnitudes.

Together with one of these choices, the gradient-flow dynamics $\frac{d\mathbf{x}}{dt} = -\nabla f(\mathbf{x})$ define the *gradient descent Ising machine* (GD-IM). In the momentum Ising machine (MOM-IM), Adam Ising machine (ADAM-IM), and first-order Adam Ising machine (1-ADAM-IM), introduced below, the nonlinearity is kept fixed and only the optimization dynamics are modified.

III. THEORETICAL RESULTS

A. Momentum-based Ising machines (MOM-IMs)

To enable the physical implementation of the momentum optimizer in Ising machines, we derive its continuous-time formulation starting from the well-known discrete momentum optimizer [16]. A continuous formulation is essential for continuous-time analog Ising machines, where state evolution occurs dynamically rather than through discrete updates. By formulating the optimizer in continuous time, we establish a natural extension of the standard discrete momentum method,

ensuring that it recovers the well-known discrete version under Euler discretization. The discrete momentum optimizer for an objective function $f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$ is defined by the update rules [16]

$$\begin{aligned} v_i^{(j+1)} &= \beta_1 v_i^{(j)} + (1 - \beta_1)(\nabla f(\mathbf{x}^{(j)}))_i, \\ x_i^{(j+1)} &= x_i^{(j)} - \eta v_i^{(j+1)}, \end{aligned} \quad (11)$$

where $\beta_1 \in [0, 1]$ is a damping parameter, η is the learning rate, and j denotes the iteration index. To obtain a continuous-time representation, we first rewrite (11) by expressing the updates relative to their previous values:

$$\begin{aligned} v_i^{(j+1)} &= v_i^{(j)} + (1 - \beta_1)((\nabla f(\mathbf{x}^{(j)}))_i - v_i^{(j)}), \\ x_i^{(j+1)} &= x_i^{(j)} - \eta v_i^{(j+1)}. \end{aligned} \quad (12)$$

We then introduce an explicit time step $\Delta t > 0$ and interpret the iteration index as sampling a trajectory at times $t_j = j \Delta t$. This makes it possible to view the scheme as a forward Euler discretization of an underlying continuous-time system. Recall that for an ODE $\dot{y}(t) = g(y(t), t)$, the forward Euler time step is

$$y(t + \Delta t) = y(t) + \Delta t g(y(t), t). \quad (13)$$

Accordingly, we write

$$\begin{aligned} v_i^{(j+1)} &= v_i^{(j)} + \Delta t (1 - \beta_1)((\nabla f(\mathbf{x}^{(j)}))_i - v_i^{(j)}), \\ x_i^{(j+1)} &= x_i^{(j)} - \Delta t \eta v_i^{(j+1)} \approx x_i^{(j)} - \Delta t \eta v_i^{(j)}, \end{aligned} \quad (14)$$

which reduces to (12) when $\Delta t = 1$. In the second line of Eq. (14), we replace $v_i^{(j+1)}$ by $v_i^{(j)}$ to obtain an explicit Euler form. Since $v_i^{(j+1)} - v_i^{(j)} = O(\Delta t)$, this modifies the x_i update only by $O(\Delta t^2)$. Interpreting $v_i^{(j)} = v_i(t_j)$ and $x_i^{(j)} = x_i(t_j)$, the updates take the Euler form

$$\begin{aligned} v_i(t + \Delta t) &= v_i(t) + \Delta t (1 - \beta_1)((\nabla f(\mathbf{x}(t)))_i - v_i(t)), \\ x_i(t + \Delta t) &= x_i(t) - \Delta t \eta v_i(t). \end{aligned} \quad (15)$$

Taking the limit $\Delta t \rightarrow 0$ yields the continuous-time system, in vector notation

$$\begin{aligned} \frac{d\mathbf{v}}{dt} &= (1 - \beta_1)(\nabla f - \mathbf{v}), \\ \frac{d\mathbf{x}}{dt} &= -\eta \mathbf{v}, \end{aligned} \quad (16)$$

which matches the standard continuous-time momentum formulation found in [11, 12, 19, 25]. Together with a suitable choice of nonlinearity (i.e. using Eqs. (7), (8), (9), or (10) to specify ∇f) to enforce binary spin states, this defines the *momentum Ising machine* (MOM-IM). This system can be rewritten as a second-order differential equation by eliminating the auxiliary velocity variable, yielding

$$\frac{d^2 x_i}{dt^2} + (1 - \beta_1) \frac{dx_i}{dt} + \eta(1 - \beta_1)(\nabla f(\mathbf{x}))_i = 0, \quad (17)$$

which corresponds to the classical heavy ball with friction model introduced by Polyak [16]. In this analogy, the optimization trajectory resembles a massive particle moving through a potential energy landscape defined by $f(\mathbf{x})$, with friction proportional to velocity. This formulation captures the role of momentum as a means of accelerating convergence:

inertia allows the system to maintain direction across shallow regions of the landscape, while the friction term ensures eventual stabilization near a minimum. Compared to gradient descent, which lacks such temporal smoothing, this dynamic can more effectively traverse narrow valleys and escape sub-optimal minima, often leading to faster convergence and less entrapment in local minima.

B. Adam-based Ising machines (ADAM-IMs) and first-order Adam Ising machines (1-ADAM-IMs)

Just as in the momentum case, we start from the discrete Adam update rules [26] and rewrite them into a form suitable for interpretation as a forward Euler discretization. Adam augments momentum by tracking a second moment of the gradient together with bias-corrected estimates. The standard updates read [26]

$$\begin{aligned} g_i^{(j+1)} &= (\nabla f(\mathbf{x}^{(j)}))_i, \\ v_i^{(j+1)} &= \beta_1 v_i^{(j)} + (1 - \beta_1) g_i^{(j+1)}, \\ w_i^{(j+1)} &= \beta_2 w_i^{(j)} + (1 - \beta_2) (g_i^{(j+1)})^2, \\ \hat{v}_i^{(j+1)} &= \frac{v_i^{(j+1)}}{1 - \beta_1^j}, \quad \hat{w}_i^{(j+1)} = \frac{w_i^{(j+1)}}{1 - \beta_2^j}, \\ x_i^{(j+1)} &= x_i^{(j)} - \eta \frac{\hat{v}_i^{(j+1)}}{\sqrt{\hat{w}_i^{(j+1)} + \epsilon}}. \end{aligned} \quad (18)$$

Here $j = 1, 2, \dots$ is the iteration index; β_1 and β_2 are the fixed decay coefficients of the first and second moments, respectively; η is the learning rate; and $\epsilon > 0$ prevents division by zero. The quantities β_1^j and β_2^j are ordinary powers of these fixed coefficients, not additional parameters. They provide the bias correction and evolve deterministically with the iteration, equivalently via $\beta_k^{j+1} = \beta_k \beta_k^j$ for $k \in \{1, 2\}$. Eliminating the auxiliary variables by substitution and rewriting the updates relative to their previous values, as in the momentum case, gives

$$\begin{aligned} v_i^{(j+1)} &= v_i^{(j)} + (1 - \beta_1)((\nabla f(\mathbf{x}^{(j)}))_i - v_i^{(j)}), \\ w_i^{(j+1)} &= w_i^{(j)} + (1 - \beta_2)((\nabla f(\mathbf{x}^{(j)}))_i^2 - w_i^{(j)}), \\ x_i^{(j+1)} &= x_i^{(j)} - \eta \frac{\sqrt{1 - \beta_2^j} v_i^{(j+1)}}{1 - \beta_1^j \sqrt{w_i^{(j+1)} + \epsilon}}. \end{aligned} \quad (19)$$

As in the momentum derivation, we first replace $v_i^{(j+1)}$ and $w_i^{(j+1)}$ in the x_i update by their values at step j , which gives the first-order approximation

$$x_i^{(j+1)} \approx x_i^{(j)} - \eta \frac{\sqrt{1 - \beta_2^j} v_i^{(j)}}{1 - \beta_1^j \sqrt{w_i^{(j)} + \epsilon}}. \quad (20)$$

Since both v_i and w_i change by $O(\Delta t)$ over a single step, this modifies the x_i update only at order $O(\Delta t^2)$ and therefore does not affect the continuous-time limit. We now introduce an explicit time step $\Delta t > 0$ and interpret the iteration index as sampling a trajectory at times $t_j := j \Delta t$. In this

parametrization, the original discrete updates are recovered for $\Delta t = 1$, and the scheme becomes

$$\begin{aligned} v_i^{(j+1)} &= v_i^{(j)} + \Delta t (1 - \beta_1)((\nabla f(\mathbf{x}^{(j)}))_i - v_i^{(j)}), \\ w_i^{(j+1)} &= w_i^{(j)} + \Delta t (1 - \beta_2)((\nabla f(\mathbf{x}^{(j)}))_i^2 - w_i^{(j)}), \\ x_i^{(j+1)} &= x_i^{(j)} - \Delta t \eta \frac{\sqrt{1 - \beta_2^j} v_i^{(j)}}{1 - \beta_1^j \sqrt{w_i^{(j)} + \epsilon}}. \end{aligned} \quad (21)$$

Identifying $v_i^{(j)}$ with $v_i(t_j)$, $w_i^{(j)}$ with $w_i(t_j)$, and $x_i^{(j)}$ with $x_i(t_j)$, and then taking the limit $\Delta t \rightarrow 0$, yields the continuous-time Adam dynamics. For the bias-correction factors, the discrete setup uses $\Delta t = 1$, so $t = j$; under this convention, the discrete powers β_k^j become β_k^t in the continuous-time notation. Powers, square roots, and divisions are understood componentwise:

$$\begin{aligned} \frac{d\mathbf{v}}{dt} &= (1 - \beta_1)(\nabla f - \mathbf{v}), \\ \frac{d\mathbf{w}}{dt} &= (1 - \beta_2)((\nabla f)^2 - \mathbf{w}), \\ \frac{d\mathbf{x}}{dt} &= -\eta \frac{\sqrt{1 - \beta_2^t} \mathbf{v}}{1 - \beta_1^t \sqrt{\mathbf{w} + \epsilon}}. \end{aligned} \quad (22)$$

Together with a suitable nonlinearity to enforce binary spin states, this continuous-time formulation defines the *Adam Ising machine* (ADAM-IM).

The third line of Eq. (22) contains a time-dependent prefactor $\frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t}$ that complicates both physical implementation and analysis, which we refer to as the *adaptive learning rate*. To reduce this complexity, we examine its first-order Puiseux series expansion:

$$\frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} = \frac{\sqrt{-\ln(\beta_2)}}{-\ln(\beta_1)\sqrt{t}} + O(\sqrt{t}). \quad (23)$$

Motivated by this, we define a simplified variant

$$\begin{aligned} \frac{d\mathbf{v}}{dt} &= (1 - \beta_1)(\nabla f - \mathbf{v}) \\ \frac{d\mathbf{w}}{dt} &= (1 - \beta_2)((\nabla f)^2 - \mathbf{w}) \\ \frac{d\mathbf{x}}{dt} &= -\eta' \frac{\mathbf{v}}{\sqrt{\mathbf{w}t + \epsilon}}, \end{aligned} \quad (24)$$

where η' is the effective 1-ADAM-IM learning rate, defined in terms of the ADAM-IM learning-rate scale η as

$$\eta' = \frac{\sqrt{-\ln(\beta_2)}}{-\ln(\beta_1)} \eta.$$

Thus, the constant prefactor from Eq. (23) is absorbed into η' . Together with a suitable choice of nonlinearity to enforce binary spin states, this defines the *first-order Adam Ising machine* (1-ADAM-IM). In Fig. 1 we compare the Adam adaptive learning rate $\frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t}$ (shown in blue) with its first-order Puiseux expansion, $\frac{\sqrt{-\ln(\beta_2)}}{-\ln(\beta_1)\sqrt{t}}$ (shown in orange), for $\beta_1 = \beta_2 = 0.99$. This symmetric choice is the fixed baseline

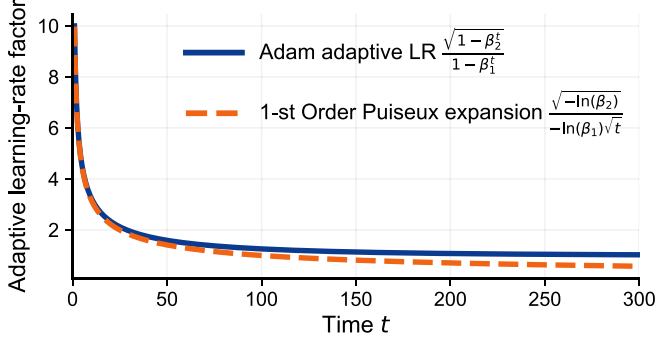


FIG. 1. Comparison of the Adam adaptive learning-rate factor and its first-order Puiseux approximation for $\beta_1 = \beta_2 = 0.99$.

used in our comparisons and isolates the approximation itself; it is not the standard discrete-Adam default, $(\beta_1, \beta_2) = (0.9, 0.999)$ [26]. The two expressions show good agreement, indicating that the simplified form captures the behavior of the original function very well. For any $\beta_1, \beta_2 \in (0, 1)$, the exact Adam factor approaches 1, whereas the first-order approximation decays to 0. With unequal decay coefficients, however, the exact factor need not approach 1 monotonically: it can first decrease and then rise again. Consequently, the first-order formulation gradually reduces the effective step size of the dynamics. This may be advantageous after the initial descent, since the largest energy reductions typically occur early in the evolution, while further improvements near low-energy configurations require more local adjustments that are less likely to overshoot nearby lower-energy states. The first-order formulation provides a simpler alternative to the full Adam-based Ising machine. We do not propose a concrete physical implementation of either continuous-time Adam formulation here. Nevertheless, comparing Eq. (22) with Eq. (24) shows a clear simplification: the full Adam state equation contains the prefactor depending on β_1^t and β_2^t , whereas the first-order system uses the leading algebraic time dependence from Eq. (23), proportional to $t^{-1/2}$. It therefore provides a useful starting point for exploring how Adam-like dynamics could be incorporated into future analog Ising-machine hardware. In addition, β_1 and β_2 no longer appear in the state equation for $d\mathbf{x}/dt$, but only control the relaxation of the internal moment variables $\mathbf{v}$ and $\mathbf{w}$. This removes the direct connection to the discrete-time Adam coefficients and allows these parameters to be tuned more freely. As shown below, this simplified formulation improves both convergence speed and solution quality.

C. Relations between gradient descent, momentum, Adam, and 1-Adam

When comparing the time to target (TTT) of different continuous-time optimizers, one would ideally like the comparison to reflect differences in the optimizer dynamics themselves rather than a trivial rescaling of time. Exact matching of intrinsic timescales is generally not possible, because GD-IM, MOM-IM, ADAM-IM, and 1-ADAM-IM have different moment dynamics. For MOM-IM,

$$\dot{\mathbf{v}} = (1 - \beta_1)(\nabla f(\mathbf{x}) - \mathbf{v}), \quad \dot{\mathbf{x}} = -\eta \mathbf{v}, \quad (25)$$

the adiabatic limit $(1 - \beta_1) \rightarrow \infty$ forces $\mathbf{v} \rightarrow \nabla f(\mathbf{x})$, such that

$$\dot{\mathbf{x}} = -\eta \nabla f(\mathbf{x}), \quad (26)$$

which coincides with GD-IM for $\eta = 1$. Thus MOM-IM reduces cleanly to GD-IM in the adiabatic limit. By contrast, ADAM-IM does not reduce cleanly to MOM-IM. Its x equation contains the adaptive factor

$$\frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \frac{1}{\sqrt{\mathbf{w}} + \epsilon}. \quad (27)$$

For a reduction to MOM-IM, this factor would have to become a nonzero constant. This does not happen in general, since the adaptive factor [Eq. (27)] depends explicitly on both time and the second-moment variable $\mathbf{w}$. Even in the late-time regime $t \rightarrow \infty$, one obtains only

$$\frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \rightarrow 1, \quad (28)$$

so that, if $\mathbf{w}(t) \rightarrow \mathbf{w}_\infty$, the dynamics approach

$$\dot{\mathbf{x}} \approx -\eta \frac{\mathbf{v}}{\sqrt{\mathbf{w}_\infty} + \epsilon}, \quad (29)$$

that is, a rescaled momentum-like step. This is still not the same as MOM-IM. A similar obstruction appears for 1-ADAM-IM,

$$\dot{\mathbf{x}} = -\eta' \frac{\mathbf{v}}{\sqrt{\mathbf{w}}\sqrt{t} + \epsilon}. \quad (30)$$

For this equation to reduce to GD-IM, the right-hand side would have to be proportional to the gradient, as in Eq. (26). Even when $\mathbf{v}$ follows ∇f , this proportionality is prevented by the factor $(\sqrt{\mathbf{w}}\sqrt{t} + \epsilon)^{-1}$, which remains component- and time-dependent. Accordingly, we use $\eta = 1$ as the common GD-IM/MOM-IM baseline in the main comparison, while the ADAM-IM learning rate η and the distinct 1-ADAM-IM learning rate η' are free hyperparameters. The Supplemental Material [27] provides complementary learning-rate checks. It reoptimizes MOM-IM with $\eta \in [1, 200]$ and directly tunes η for GD-IM in the algorithmic comparison. For the continuous-time results, it instead rescales the TTT values by the corresponding learning-rate factors. Multiplying TTT by η is the time rescaling that GD-IM would undergo if it used the same value of η , so this diagnostic attempts to match the effective timescales. These checks show that the Adam advantage is not a trivial consequence of the chosen learning rates or of different effective timescales: they do not eliminate the Adam advantage in TTT, solution quality, or success rate, and do not change the optimizer ranking. Absolute comparisons of TTT should therefore be interpreted with care. In particular, we place greater emphasis on relative trends in TTT across problem instances and on timescale-independent measures such as solution quality and success rate.

IV. BENCHMARK RESULTS

All simulations in this work rely on a numerical Euler-Maruyama discretization of the continuous-time dynamical equations underlying each Ising-machine model. Every grid scan and Bayesian optimization evaluation averages 400

stochastic trajectories of 10 000–15 000 Euler time steps (10 000 for g05 and 15 000 for GSET). The Euler time step is chosen as $\Delta t = 0.01$ for the continuous-time Ising machines and as a tunable parameter for the purely algorithmic comparisons. At the start of each run, all required initial state components are sampled independently from zero-mean Gaussian noise with standard deviation $\sqrt{\Delta t}$. The sampled values are used directly for $x_i(0)$ and, when present, $v_i(0)$, while Adam-based $w_i(0)$ is initialized using the absolute value of its sample. The system then evolves with fixed hyperparameters according to the corresponding dynamical equations; no annealing schedule is applied.

The success rate is defined as the fraction of runs that reach a prescribed target energy E_{target} . We benchmark all Ising-machine variants on two well-established MAX-CUT problem families: the g05 problem instances from the BiqMac library [28], which consist of unweighted problems of sizes $N = 60, 80, 100$ with exactly known ground-state energies, and the GSET problem instances [29], which include significantly larger sparse graphs of sizes $N = 800$ and $N = 2000$, both unweighted and weighted. For g05 problem instances we set E_{target} equal to the ground state, whereas for GSET we choose $E_{\text{target}} = 0.995 E_{\text{best}}$, where E_{best} is the energy of the best cut value reported in the literature.

To quantify optimization speed, we first introduce the time to target (TTT). For each individual run, we record the *first-passage time*, i.e., the earliest time at which the trajectory first reaches a state whose energy satisfies the chosen target criterion. The quantity T_a denotes the average of these first-passage times over all runs that successfully reach the target, while the transient success rate SR_{tr} is the fraction of runs that reach the target at least once during their evolution. Because independent repetitions can be used to increase the overall success probability, we define

$$\text{TTT} = \begin{cases} T_a \frac{\ln(0.01)}{\ln(1-\text{SR}_{\text{tr}})}, & \text{SR}_{\text{tr}} \leq 0.99, \\ T_a, & \text{SR}_{\text{tr}} > 0.99, \end{cases} \quad (31)$$

which represents the time required to reach the specified target with 99% probability. This is the usual time-to-solution (TTS) when the target is the ground state or best-known solution; we retain TTT because the target can instead be a prescribed fraction of the best-known value. The time entering TTT via T_a depends on the setting. For the continuous-time Ising-machine models, time is measured as the simulated Euler time $t = n\Delta t$ obtained from the Euler discretization. Throughout this work, we fix $\Delta t = 0.01$, so that all continuous-time methods are compared on the same numerical timescale and the discretization remains sufficiently fine to resolve the underlying dynamics. Thus, unless stated otherwise, TTT refers to this Euler time. For the purely algorithmic discrete-time implementations discussed in Sec. IV C, by contrast, Δt is a tunable numerical parameter rather than a meaningful time variable, and we therefore report wall-clock CPU time measured internally by the C++ solver, denoted by TTT_{CPU} . All simulations were performed on VSC Tier-1 Hortense using the same type of CPU, namely AMD EPYC 7H12 processors (Rome, 2.6 GHz), ensuring a fair comparison between methods. For Bayesian optimization, we use $-1/\text{TTT}$ rather than the raw TTT as the objective. This is because TTT

diverges when no target is reached, creating a discontinuity that is difficult for Bayesian optimization to model. By contrast, $-1/\text{TTT}$ maps such cases to a finite value 0, preserves the preference for low TTT, gives higher resolution in the low-TTT region, and restores the minimization convention through the minus sign. The same construction is used for TTT_{CPU} in the algorithmic setting.

Each Ising-machine variant has a different number of free hyperparameters. To keep the comparison both fair and computationally tractable, we therefore use a two-stage procedure. First, for every nonlinearity and every Ising-machine variant, we perform a coarse 30×30 grid scan over (α, β) , with all remaining parameters fixed to a controlled baseline. Specifically, we use $\gamma = 0.005$ for all variants; $\eta = 1$ for GD-IM; $\beta_1 = 0.99$ and $\eta = 1$ for MOM-IM; $\beta_1 = \beta_2 = 0.99$ and $\eta = 1$ for ADAM-IM; and $\beta_1 = \beta_2 = 0.99$ with $\eta' = 9.97$ for 1-ADAM-IM. For 1-ADAM-IM, $\eta' = \frac{\sqrt{-\ln(\beta_2)}}{-\ln(\beta_1)} \eta$ follows from the same underlying learning-rate scale $\eta = 1$, so that the two Adam-based formulations are compared under matched constants. The complete search domains and settings are listed in the Supplemental Material [27]. These controlled reference settings are used to isolate the nonlinearity and are not claimed to be universal literature optima. Different fixed settings can change the quantitative coarse-scan bars. Our final conclusions therefore rely on the second stage, in which all applicable optimizer and nonlinearity parameters are jointly tuned by Bayesian optimization. This first stage is restricted to the g05 problem instances and provides an initial comparison of the nonlinearities at the coarse resolution of the grid scan. Each Bayesian-optimization run uses 1000 parameter evaluations, of which the first 300 are random and the remaining 700 are chosen adaptively. For the final optimizer comparisons, the Bayesian optimization was repeated over seven distinct parameter ranges per optimizer. This was done to avoid relying on a single broad search domain: in practice, we found that very large parameter ranges made it difficult for Bayesian optimization to refine the optimum, since the relevant high-performing region could occupy only a small part of the full domain. Splitting the search into several smaller, range-focused runs improved the robustness of the tuning while keeping the total computational cost manageable. The choice of seven ranges was therefore a compromise between search coverage and the available computational budget. All optimizers are compared under the same computational budget, using the same number of Bayesian-optimization evaluations, stochastic trajectories per evaluation, and Euler steps per trajectory. Additional Bayesian-optimization details are provided in the Supplemental Material [27]. Moreover, in our implementation the additional arithmetic required by MOM-IM, ADAM-IM, and 1-ADAM-IM becomes negligible compared to the cost of evaluating the coupling term $\sum_j J_{ij}x_j$ as the system size grows. The optimizer-specific updates apply a constant number of componentwise operations to each of the N state components and their moment variables and therefore scale as $O(N)$; a naive dense coupling evaluation computes N contributions for each of the N local fields and therefore scales as $O(N^2)$. To examine the resulting crossover, we profiled a single dense 1-ADAM-IM trajectory at eight system sizes from $N = 100$ to $N = 10,000$, using

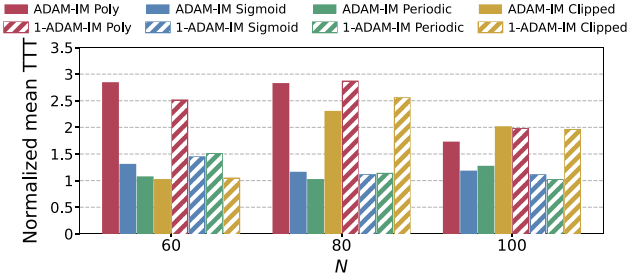


FIG. 2. Grid-scan comparison of ADAM-IM and 1-ADAM-IM across the four nonlinearities on the g05 benchmark set. Bars show the normalized mean TTT over the ten problems at each N . The scan varies (α, β) with $\gamma = 0.005$. For ADAM-IM, $\beta_1 = \beta_2 = 0.99$ and $\eta = 1$; for 1-ADAM-IM, $\beta_1 = \beta_2 = 0.99$, and using $\eta = 1$ in $\eta' = \frac{\sqrt{-\ln(\beta_2)}}{-\ln(\beta_1)} \eta$ gives $\eta' = 9.97$.

100 profiling iterations per timing sample and 25 independent repeats. The coupling share rises from 86.53% at $N = 100$ to 94.33% at $N = 200$, 98.00% at $N = 400$, 99.10% at $N = 800$, 99.68% at $N = 2000$, 99.83% at $N = 5000$, and 99.92% at $N = 10,000$. The increase in the coupling share reflects these different scalings. Sparse, structured, or hardware-accelerated coupling matrices can change both the constants and the coupling scaling. The profiling protocol and complete results are given in the Supplemental Material [27].

A. Selection of nonlinearity

To begin our analysis, we compare the performance of all nonlinearities for Adam-based Ising machines using a systematic grid scan over the nonlinearity parameters. Grid scans are computationally feasible when restricted to a small number of parameters. For this reason, we restrict our first evaluation to a 30×30 grid scan over (α, β) while keeping all other hyperparameters fixed to baseline values. This coarse but exhaustive scan provides a comparison of the four nonlinearities for each Ising-machine variant, without relying on full Bayesian optimization. By varying only (α, β) , we largely suppress the effect of optimizer-specific hyperparameters and isolate the role of the nonlinearity itself. Under Bayesian optimization, by contrast, the internal optimizer parameters are tuned as well, so the resulting performance reflects the combined effect of optimizer and nonlinearity. Accordingly, the coarse scan is used to rank the nonlinearities within each optimizer and to test how closely 1-ADAM-IM reproduces ADAM-IM under matched constants.

Figure 2 summarizes the coarse grid-scan comparison across the four nonlinearities for ADAM-IM and 1-ADAM-IM on the g05 benchmark set. For each g05 problem instance and each optimizer-nonlinearity pair, we first determine the minimum TTT obtained anywhere on the 30×30 grid. This gives one best grid-scan value for every pair on every problem instance. For each problem size N , these per-problem-instance minima are then averaged over the ten corresponding instances. Finally, the resulting average TTT values are divided by the lowest average at the same N , so that the best optimizer-nonlinearity pair has normalized average TTT = 1. The full per-problem-instance grid-scan data are provided

in the Supplemental Material [27]. Two points are immediately clear from this aggregated view. First, under this coarse grid scan, ADAM-IM and 1-ADAM-IM perform essentially identically: when both are evaluated at the same hyperparameter values and the 1-ADAM-IM learning rate is fixed from the first-order expansion, the simplified 1-ADAM-IM formulation reproduces the behavior of the full Adam dynamics. Second, the sigmoid and periodic nonlinearities emerge as the strongest overall choices and perform very similarly in this coarse comparison. By contrast, the polynomial and clipped nonlinearities perform weaker on average. The weaker performance of the clipped nonlinearity is notable, since clipped transfer functions have been shown to perform well in gradient-descent-based analog Ising machines by suppressing amplitude inhomogeneity [20]. This shows that the nonlinearity ranking depends not only on the nonlinearity itself, but also on its interaction with the optimizer dynamics.

To complement the coarse grid scan, we next perform a full Bayesian optimization over all hyperparameters, excluding the time step Δt , for each nonlinearity and compare the resulting time to target, TTT, on the g05 benchmark set. In Fig. 3 TTT is normalized separately for each benchmark problem instance by dividing by the lowest TTT obtained on that problem instance, so that the fastest method is assigned the value 1. The figures therefore show relative performance on each problem instance rather than absolute TTT. They display the Bayesian-optimized results for (a) ADAM-IM and (b) 1-ADAM-IM, and thus reveal how the optimizer and the choice of nonlinearity jointly determine performance once all tunable parameters are refined beyond the (α, β) grid.

The Bayesian-optimized results refine the nonlinearity ranking suggested by the coarse scan. In both Adam-based optimizers, the performance of the sigmoid nonlinearity emerges as the clear overall winner: it achieves the lowest TTT across the g05 problem instances and consistently outperforms the other nonlinearities. The polynomial nonlinearity, although clearly weaker in the coarse grid scan, can be tuned by Bayesian optimization to approach the sigmoid nonlinearity on some easier problem instances, but it does not match its overall robustness. By contrast, the periodic nonlinearity—which was nearly on par with the sigmoid nonlinearity in the grid scan—is now systematically outperformed once all hyperparameters are co-optimized. The clipped nonlinearity performs worst: in some cases it fails to reach the target at all, indicating limited tunability in the enlarged hyperparameter space. Since all nonlinearities are evaluated in the same way, we do not attribute the clipped result to a numerical artifact. This shows that the nonlinearity ranking depends not only on the optimizer-nonlinearity combination, but also on how the corresponding hyperparameters are tuned.

This naturally raises the question of whether the same nonlinearity preference also persists for the simpler optimizers GD-IM and MOM-IM. To address this, we compare all four nonlinearities across the three optimizers Gradient Descent (GD-IM), Momentum (MOM-IM), and First-Order Adam (1-ADAM-IM). As before, we perform a 30×30 scan over (α, β) with all remaining parameters fixed. Figure 4 summarizes this grid scan comparison in a compact form. For each g05 problem instance and each method, we first determine the minimum TTT over the grid scan. For each

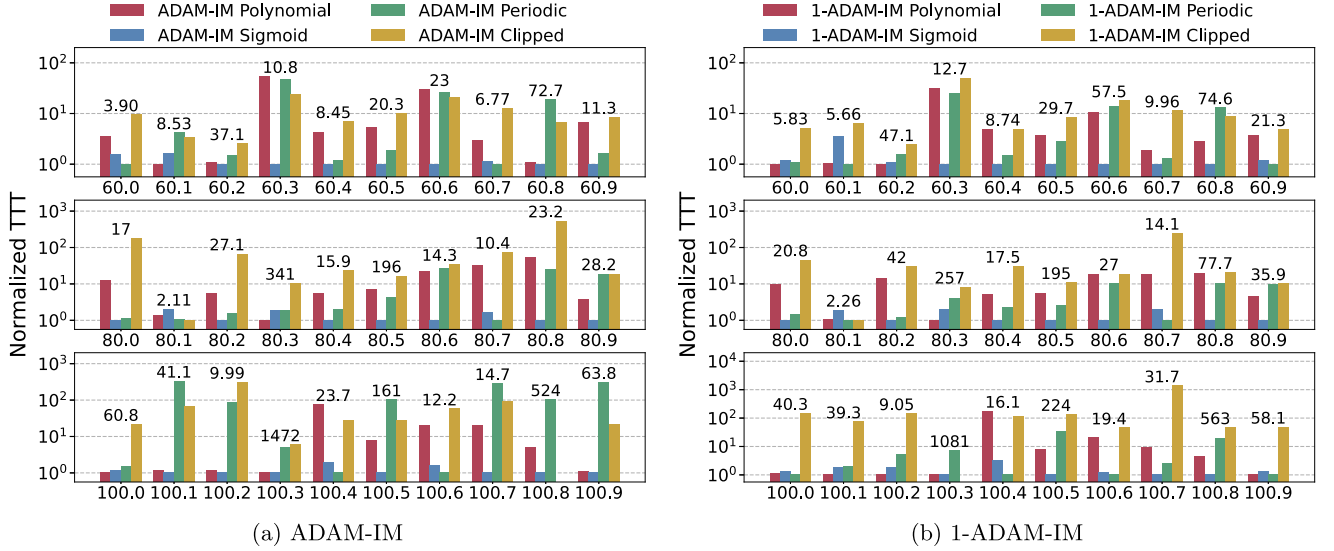


FIG. 3. Bayesian-optimized normalized time to target (TTT) on a logarithmic scale for all nonlinearities on the g05 benchmark set. Results are shown for (a) ADAM-IM and (b) 1-ADAM-IM. For each problem instance, TTT is normalized to the lowest value obtained on that problem instance, so that the fastest method is assigned the value of 1. The numerical labels above the bars indicate the corresponding unnormalized minimum TTT values used for this normalization.

problem size N , these per-problem-instance minima are averaged over the ten corresponding instances, after which the average values are normalized by the lowest average at that N . The full per-problem-instance grid-scan data are provided in the Supplemental Material [27]. Two conclusions stand out clearly from this normalized-average comparison. First, 1-ADAM-IM is by far the fastest optimizer throughout the coarse grid scan, with a speed advantage of roughly one to two orders of magnitude over both GD-IM and MOM-IM across all three problem sizes. Second, the nonlinearity ranking depends on the optimizer, but the sigmoid nonlinearity emerges as the clearest overall winner in this broader comparison. For GD-IM and MOM-IM, sigmoid is unambiguously the best-performing choice. Under 1-ADAM-IM, the periodic nonlinearity remains the closest competitor and is sometimes nearly as fast, but the sigmoid nonlinearity still yields the lowest TTT overall. Thus, while the periodic nonlinearity was also competitive in the Adam-based coarse scan, the present

three-optimizer comparison more clearly singles out sigmoid as the most robust nonlinearity at coarse parameter grid-scan resolution.

To test whether the same picture persists beyond the coarse grid scan, we also carried out the full Bayesian-optimization analysis for GD-IM and MOM-IM on the g05 benchmark set, optimizing all free hyperparameters jointly for each of the four nonlinearities. The detailed results are reported in the Supplemental Material [27]. Qualitatively, however, they lead to the same conclusion as for the Adam-based variants: the sigmoid nonlinearity remains the strongest overall choice after full tuning. The polynomial nonlinearity can partly catch up on some easier problem instances, but is less consistently effective across the full problem set, while the periodic and clipped nonlinearities fall behind once all hyperparameters are co-optimized.

This outcome is consistent with earlier work by Böhm *et al.* [20], who compared the same four nonlinear transfer functions for GD-IM and argued in favor of the sigmoid nonlinearity. Their reasoning is that sigmoid, being a smooth saturating function, effectively suppresses amplitude inhomogeneity—a dominant source of performance degradation in analog Ising machines—and does so without introducing the attractor distortions or multistability issues sometimes seen in clipped and periodic systems. Consistent with this earlier study, clipping remains relatively competitive for GD-IM across all three problem sizes in our grid scan, but falls behind after Bayesian optimization. For MOM-IM and the Adam-based methods, clipping already falls behind in the grid scan and performs worst after Bayesian optimization. Thus, its ranking depends on both the optimizer and the tuning protocol.

B. Optimizer comparison with the sigmoid nonlinearity

Having established that the sigmoid nonlinearity is the most robust and consistently best-performing choice across

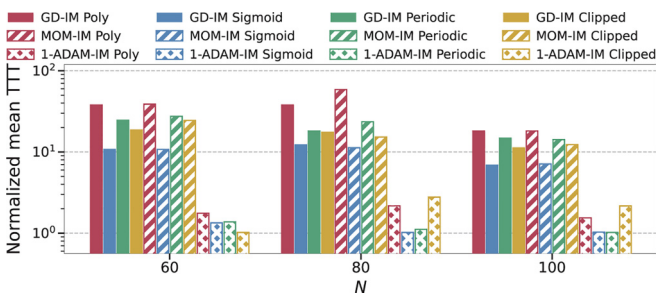


FIG. 4. Grid-scan comparison of GD-IM, MOM-IM and 1-ADAM-IM across the four nonlinearities on the g05 benchmark set. Bars show the normalized mean TTT over the ten problems at each N , on a logarithmic scale. The scan varies (α, β) with $\gamma = 0.005$. We use $\eta = 1$ for GD-IM; $\beta_1 = 0.99$ and $\eta = 1$ for MOM-IM; and $\beta_1 = \beta_2 = 0.99$ for 1-ADAM-IM, where using $\eta = 1$ in $\eta' = \frac{\sqrt{-\ln(\beta_2)}}{-\ln(\beta_1)}$ gives $\eta' = 9.97$.

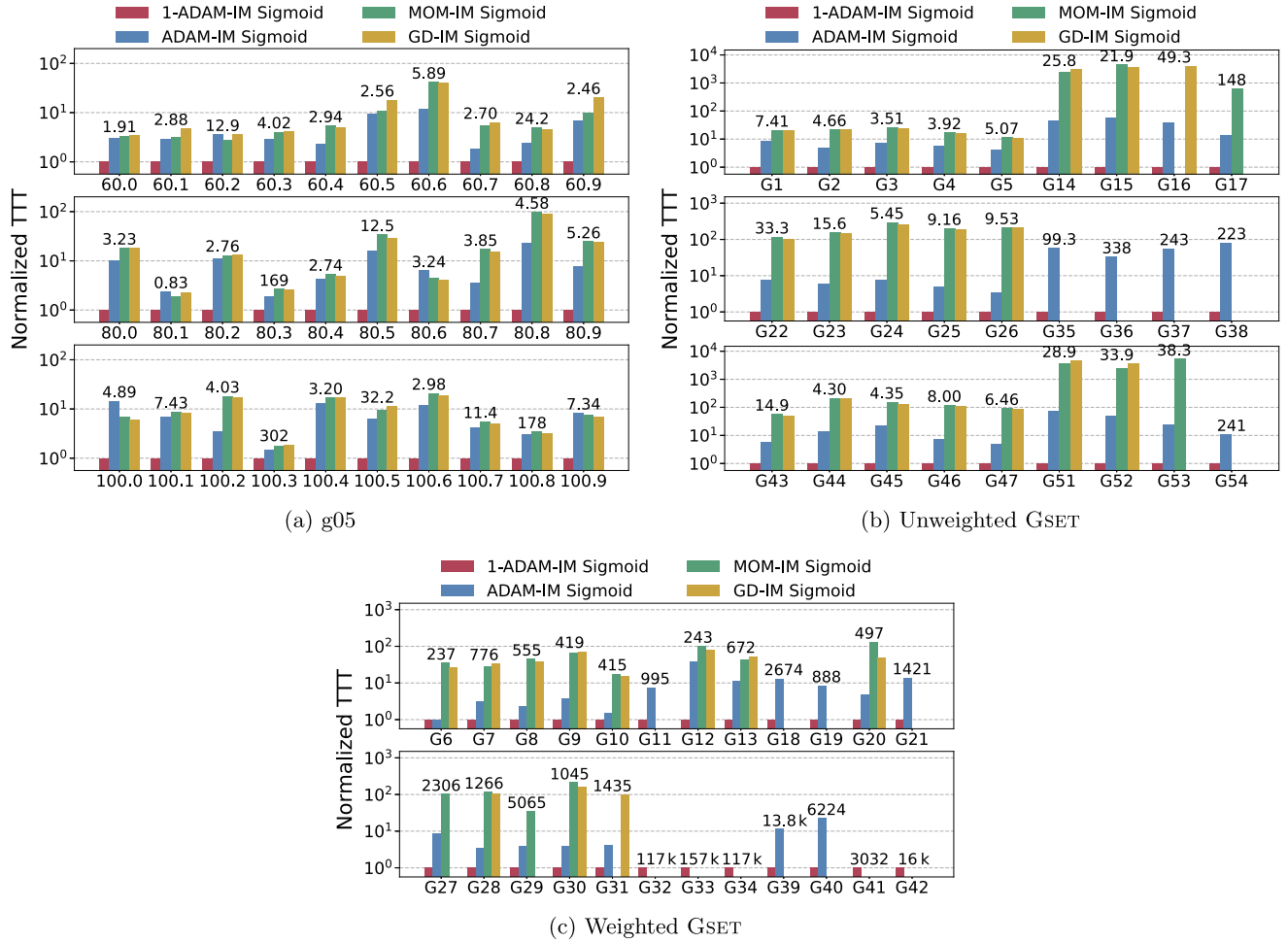


FIG. 5. Bayesian-optimized normalized TTT for the continuous-time implementations with the sigmoid nonlinearity on (a) g05, (b) unweighted GSET, and (c) weighted GSET. Values are normalized per problem instance to the fastest method. In panel (a), all optimizers reach the ground-state target; in panels (b) and (c), bars are omitted when the optimizer does not reach 99.5% of the best-known cut value. Numerical labels indicate the unnormalized minimum TTT used for normalization.

all optimizers, we now restrict our attention to this nonlinearity and compare the optimizers themselves. The goal of this step is to assess how Gradient Descent, Momentum, Adam, and First-Order Adam behave when all their hyperparameters are fully tuned through Bayesian optimization, and when tested on benchmark sets that are more challenging. To obtain a clearer picture, we therefore also consider the GSET benchmarks, which are significantly harder and come in two variants: unweighted and weighted MaxCut problem instances. These graphs are well known to produce more rugged energy landscapes and larger performance spreads, making them ideal for highlighting optimizer-induced differences.

Figure 5 shows the Bayesian-optimized TTT for GD-IM, MOM-IM, ADAM-IM, and 1-ADAM-IM on (a) g05, (b) unweighted, and (c) weighted GSET problems. For every optimizer and every problem instance, the reported value corresponds to the best result obtained across seven independent Bayesian optimizations performed over seven distinct hyperparameter ranges. A striking and universal observation across all problem instances is that 1-ADAM-IM is the fastest optimizer for every single benchmark problem. The magnitude of this advantage depends strongly on problem difficulty,

becoming more pronounced as we move from g05 to unweighted and finally weighted GSET graphs.

For the easier subset (G1–G5), 1-ADAM-IM is roughly one order of magnitude faster than the other methods, which again perform comparably among themselves. For medium-difficulty problem instances (G22–G26 and G43–G47), a stronger performance difference emerges. Here 1-ADAM-IM is typically two orders of magnitude faster than GD-IM and MOM-IM, and about one order of magnitude faster than ADAM-IM; ADAM-IM itself tends to be roughly one order of magnitude faster than GD-IM and MOM-IM. On the hardest unweighted problems (G14–G17, G35–G38, G51–G54), the contrast becomes most pronounced: GD-IM and MOM-IM frequently fail to reach 99.5% of the best-known cut value (in which case no bar is shown), and even when they do succeed, their normalized TTTs are three to four orders of magnitude larger than those of 1-ADAM-IM. Both ADAM-IM and 1-ADAM-IM reliably reach the 99.5% threshold on all these problem instances, with 1-ADAM-IM maintaining a consistent one- to two-order-of-magnitude advantage.

The same trend is amplified in the weighted GSET problems [Fig. 5(c)], which are among the most challenging MaxCut

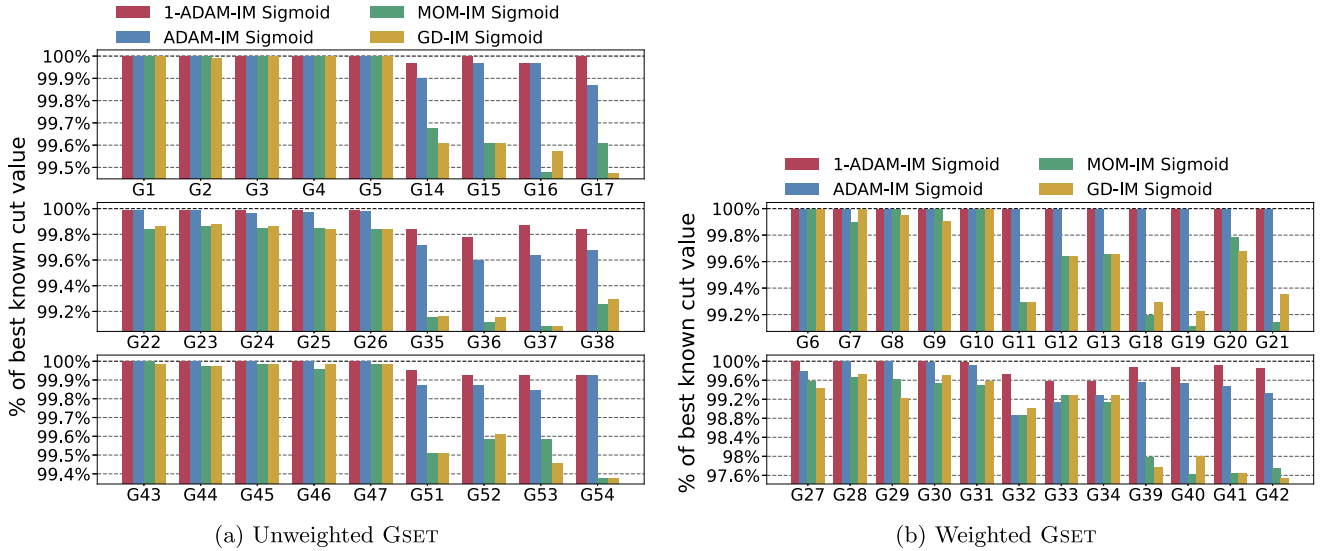


FIG. 6. Best solution quality obtained with the sigmoid nonlinearity on (a) unweighted and (b) weighted GSET. Values show the percentage of the best-known MaxCut value reached by each optimizer, using the best result over seven Bayesian-optimization runs.

benchmarks in our tests. For medium-hard problem instances (G6–G13, G18–G21, G27–G31), GD-IM and MOM-IM occasionally fail to meet the 99.5% success criterion. When they do succeed, 1-ADAM-IM is again around two orders of magnitude faster, and typically maintains about a one order of magnitude lead over ADAM-IM. For the hardest weighted problem instances (G32–G42), the gap becomes decisive: 1-ADAM-IM is the only optimizer that consistently reaches 99.5% of the best-known solution. ADAM-IM succeeds only sporadically, and GD-IM and MOM-IM almost never reach the target accuracy. This indicates that 1-ADAM-IM not only converges faster but also attains higher-quality solutions on the most difficult graphs. This motivates a more detailed investigation of how these optimizers differ in success rates, convergence profiles, and solution quality in the next section.

To complement the TTT results, we now compare the achieved solution quality for each optimizer on the unweighted and weighted GSET MaxCut benchmarks. Figure 6 displays, for each problem instance, the percentage of the best-known cut value reached by GD-IM, MOM-IM, ADAM-IM, and 1-ADAM-IM for the (a) unweighted and (b) weighted GSET problems. To ensure a fair comparison, each optimizer is given the same computational budget. For every optimizer and every problem instance, the reported value corresponds to the best result obtained across seven independent Bayesian optimizations performed over seven distinct hyperparameter ranges, identical to those used in the previous TTT analysis. The corresponding absolute cut values, allowing direct comparison with the best-known values from the literature, are reported in the Supplemental Material [27]. The solution-quality results in Fig. 6 show that the speed advantage of 1-ADAM-IM is not obtained at the expense of final solution quality. On the easier GSET problem instances, all four optimizers often reach or nearly reach the best-known cuts, and the differences between methods are therefore small. On the harder problem instances, however, 1-ADAM-IM clearly emerges as the strongest method. ADAM-IM remains the closest competitor, but its TTT is consistently higher than that

of 1-ADAM-IM, while MOM-IM and GD-IM show substantially weaker performance. This hierarchy is already apparent on the harder unweighted graphs and becomes most striking on the weighted GSET benchmarks, where 1-ADAM-IM most often reaches the best-known cuts and separates itself decisively from the other optimizers. Overall, these results show that 1-ADAM-IM is not only the fastest optimizer considered here, but also gives the strongest final solution quality, especially on the most difficult benchmark problem instances.

To complement the TTT and solution-quality results, we also examine the transient success rates of all four optimizers under the *sigmoid* nonlinearity. Throughout, we reuse exactly the same simulation data as in the previous Bayesian-optimization study. We use 99.5% of the best-known cut value for GSET (unweighted and weighted) as target.

Figure 7 presents the transient success rates for (a) unweighted GSET and (b) weighted GSET. As seen in Fig. 7, 1-ADAM-IM achieves the highest transient success rate throughout all benchmark problem instances. On the unweighted GSET problem instances the difference becomes striking: 1-ADAM-IM remains at or near 100% transient success on all problem instances, while GD-IM, MOM-IM, and ADAM-IM drop substantially. In particular, GD-IM and MOM-IM fall to 0% on roughly half of these problem instances, whereas 1-ADAM-IM remains at 100% or at least very high on those same problems. On the weighted GSET problem instances, which are the most challenging, the contrast becomes even stronger. Although 1-ADAM-IM no longer stays near 100%, it still maintains clearly elevated transient success rates, whereas the transient success rate for GD-IM and MOM-IM often fall to only a few percent or 0%. Note that some problem instances reached the target of 99.5% of the best-known solution, as shown in Fig. 6, yet appear to have a transient success rate close to zero in Fig. 7. In fact, these values are nonzero but barely visible, since they are equal to $1/400$. For example on the harder G32–G34 problems, the transient success rate of 1-ADAM-IM is nonzero but very small, making it barely visible in Fig. 7(b).

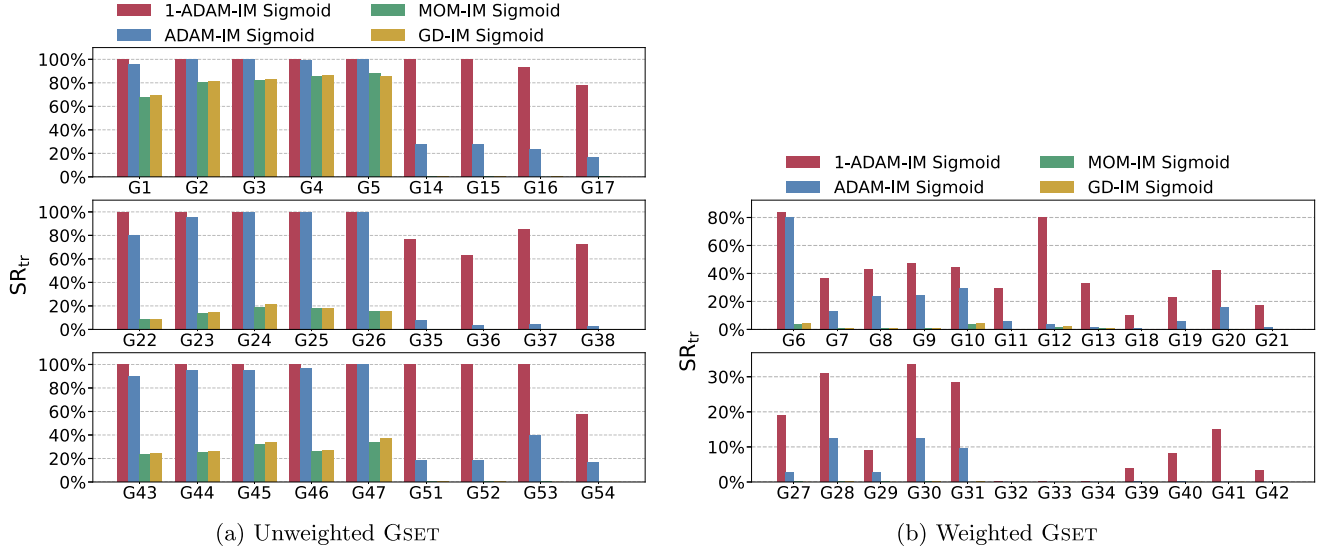


FIG. 7. Transient success rate with the sigmoid nonlinearity on (a) unweighted GSET and (b) weighted GSET. For each optimizer and problem instance, values show the best result over seven Bayesian-optimization runs.

Overall, this consistently superior transient success rate is a major reason why 1-ADAM-IM so clearly outperforms the other optimizers in TTT. Since transient success is not itself a time-based measure, it also provides a timescale-independent confirmation that 1-ADAM-IM is the strongest method.

These findings reinforce the hierarchy already apparent in the TTT and solution quality results and show that 1-ADAM-IM combines both superior speed, superior solution quality and higher success rates, making it the best-performing optimizer across all benchmarks considered in this work.

C. Discrete-time algorithmic implementations

In the previous sections, we investigated the use of ADAM optimizer to improve analog IMs, that evolve toward low-energy states in a continuous manner. However, many IMs evolve in discrete-time, in an iterative way, e.g., the measurement-feedback coherent Ising machines [6]. A natural extension is therefore to ask whether the same optimizer ideas are also useful in such discrete-time, iterative settings.

In the continuous-time simulations discussed above, the Euler-Maruyama time step Δt is only a numerical discretization parameter. It must therefore remain sufficiently small: if Δt is too large, the simulated trajectory no longer approximates the underlying continuous-time analog dynamics. In a purely algorithmic implementation, this restriction disappears. The discrete update rule itself defines the algorithm, so Δt can be treated as an additional tunable parameter rather than as a resolution parameter. Since the resulting time step no longer corresponds to a time increment, we compare algorithmic performance using wall-clock time to target, denoted by TTT_{CPU} .

There are two possible ways to construct the discrete-time algorithms used in this comparison. One option is to start from the continuous-time equations derived above and discretize them with Euler-Maruyama, now allowing the time step to be large and fully tuned. For GD-IM, we also test the corresponding fixed-time-step version with $\Delta t = 1$, so that only

the learning-rate scale is tuned. The other option, available for MOM-IM, ADAM-IM, and 1-ADAM-IM, is to use the standard discrete optimizer update rules from the literature [16,26], given in Eqs. (11) and (18). For each optimizer, we tested both available discrete-time variants and report only the better-performing one in the algorithmic results below.

This comparison leads to different choices for different optimizers. For GD-IM, MOM-IM, and 1-ADAM-IM, the Euler-Maruyama versions of Eqs. (26), (16), and (24) give the best results and are therefore used below. In these cases, Bayesian optimization searches over $\log_{10} \Delta t$, rather than over Δt directly, because useful time step values can span several orders of magnitude. By contrast, the applicable learning rate, η or η' , is optimized directly on a linear scale over its chosen interval. For ADAM-IM, the standard discrete Adam update rule in Eq. (18) performs best and is therefore used below. The full comparison between Euler-Maruyama and standard discrete implementations is given in the Supplemental Material [27].

As in the continuous-time comparison, we restrict the algorithmic analysis to the sigmoid nonlinearity and optimize all free hyperparameters by Bayesian optimization. The detailed Bayesian-optimization settings are given in the Supplemental Material [27]. Figure 8 shows the resulting Bayesian-optimized normalized TTT_{CPU} for GD-IM, MOM-IM, ADAM-IM, and 1-ADAM-IM on the (a) unweighted GSET and (b) weighted GSET benchmarks. The algorithmic TTT_{CPU} results show substantially smaller differences between optimizers than in the continuous-time setting. On the unweighted GSET problem instances, the performance differences generally remain within one order of magnitude, and all four optimizers perform comparably. On the weighted GSET problem instances, however, a clearer hierarchy emerges. ADAM-IM gives the lowest TTT_{CPU} overall, followed by 1-ADAM-IM, while GD-IM and especially MOM-IM are less competitive. This difference is most pronounced on the hardest weighted problem instances, shown in the second row of Fig. 8, where only ADAM-IM and 1-ADAM-IM consistently

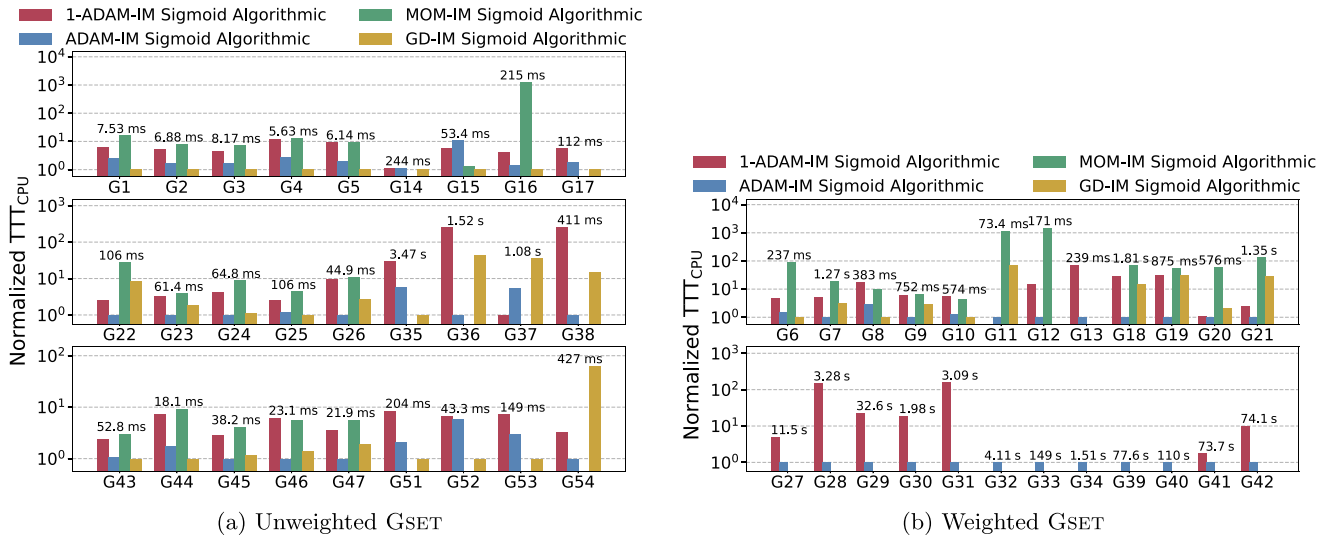


FIG. 8. Algorithmic discrete-time normalized TTT_{CPU} comparison after Bayesian optimization with the sigmoid nonlinearity on (a) unweighted GSET and (b) weighted GSET. Values are normalized per problem instance to the fastest method. Bars are omitted when the optimizer does not reach 99.5% of the best-known cut value. Numerical labels indicate the unnormalized minimum TTT_{CPU} used for normalization.

reach 99.5% of the best-known cut value. Among these two, ADAM-IM is the faster algorithmic method.

To determine whether this speed advantage is accompanied by better final solutions, we next compare the achieved solution quality. Figure 9 shows the percentage of the best-known MaxCut value reached by each optimizer on the unweighted and weighted GSET problem instances. The solution-quality results follow the same overall trend as those observed for the TTT. On the unweighted GSET problem instances, the optimizers perform similarly, although MOM-IM is somewhat weaker than the other methods. On the weighted problem instances, the differences become more pronounced: ADAM-IM reaches the best solutions most consistently, followed by 1-ADAM-IM, while GD-IM and MOM-IM obtain lower-quality solutions on the harder

problems. Thus, the performance gap grows with problem difficulty.

Finally, we examine whether the same trend is visible in the transient success rate. Figure 10 reports the best transient success rate obtained by each optimizer under the same Bayesian-optimization procedure. For the problem instances that reached the target of 99.5% of the best known solution, which can be found in Fig. 9, but the transient success rate appears to be zero on Fig. 10, they are just hardly visible, and have a value of $1/400$. The transient success-rate results reinforce the same picture. On the unweighted GSET problem instances, the performance of the different methods is quite similar, with MOM-IM again showing the weakest performance. On the easier weighted problem instances, ADAM-IM achieves the highest transient success rate, followed by

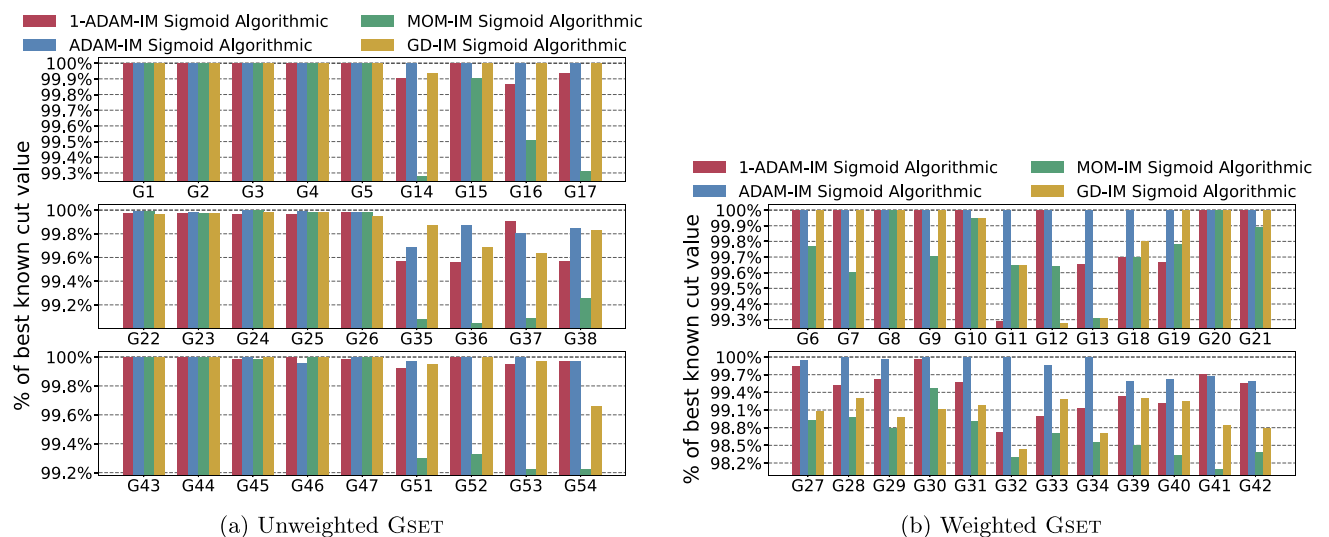


FIG. 9. Algorithmic discrete-time solution quality after Bayesian optimization obtained with the sigmoid nonlinearity on (a) unweighted and (b) weighted GSET. Values show the percentage of the best-known MaxCut value reached by each optimizer.

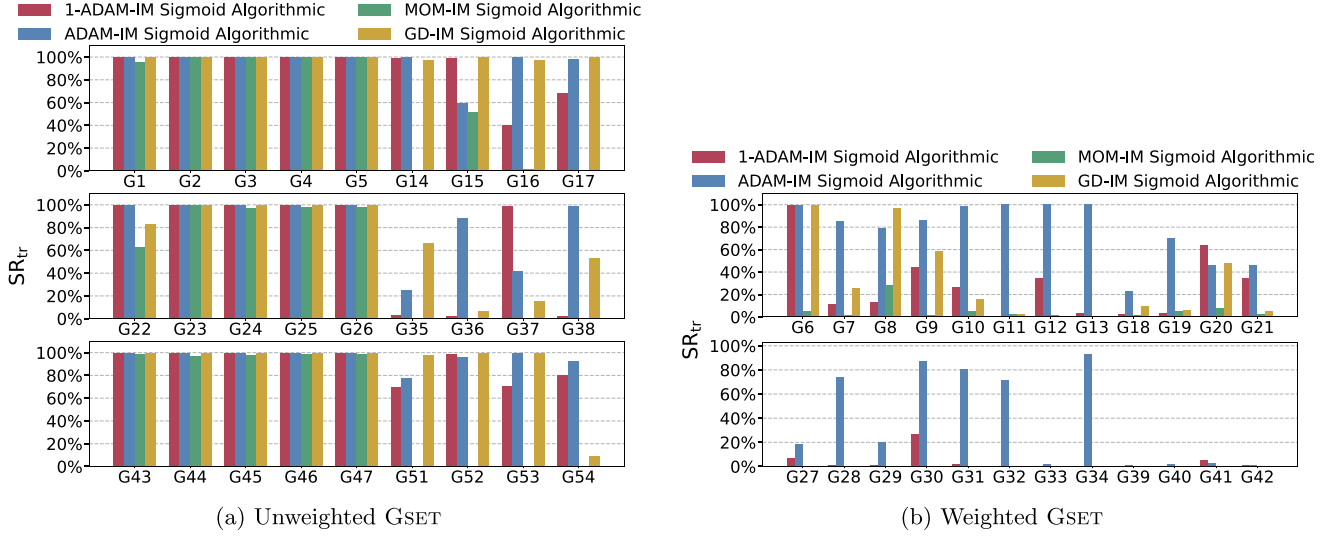


FIG. 10. Algorithmic discrete-time transient success rate after Bayesian optimization with the sigmoid nonlinearity on (a) unweighted GSET and (b) weighted GSET.

1-ADAM-IM, then GD-IM and MOM-IM. On the hardest weighted problem instances, the performance differences become larger: ADAM-IM is clearly the best-performing method, with 1-ADAM-IM as the closest competitor, while GD-IM and MOM-IM rarely reach the target. The behavior on G32 and G34 is particularly noteworthy: these problem instances are difficult for the other optimizers in our algorithmic comparison, yet ADAM-IM reaches the best-known cut value in Fig. 9 and maintains substantial transient success rate in Fig. 10 (around 70% for G32 and close to 100% for G34).

Overall, the algorithmic setting leads to a different conclusion from that of the continuous-time setting. When the target energy is reachable by all methods, as on the unweighted GSET problem instances, GD-IM can be tuned to achieve TTT_{CPU} comparable to the other methods, leaving less room for an adaptive optimizer to improve speed alone. The advantage of ADAM-IM becomes clear on the harder weighted problem instances, where GD-IM and MOM-IM increasingly fail to reach the target.

D. Comparison with selected state-of-the-art Ising solvers

For comparison with selected state-of-the-art Ising solvers, the Supplemental Material [27] reports cut quality on the 51 common GSET instances. For each instance, the cut ratio is the cut obtained by a method divided by the best-known cut, and the mean cut ratio is the unweighted average of these 51 ratios, expressed as a percentage. For this solution-quality comparison, Bayesian optimization directly targets the maximum cut value rather than TTT. The continuous-time 1-ADAM-IM reaches the best-known cut on 33/51 instances and obtains a mean cut ratio of 99.968%, while the discrete-time ADAM-IM reaches the best-known cut on 41/51 instances and obtains a mean cut ratio of 99.988%. The published reference series obtain mean ratios of 99.988% for original chaotic amplitude control (CAC) [22,23], 99.994% for a simulated coherent Ising machine with chaotic amplitude control (SimCIM + CAC) [23], 99.987% for the adiabatic

simulated-bifurcation machine (aSBM) [11], and 99.999% for the discrete simulated-bifurcation machine (dSBM) [11]. Because the studies use different hardware, trajectory counts, tuning budgets, stopping criteria, and success definitions, these best-observed results are solution-quality ceilings rather than a direct TTT, TTS, success-probability, or equal-budget comparison. Our Adam results were obtained with a general-purpose CPU implementation, without acceleration by a graphics processing unit (GPU), an FPGA, or dedicated Ising-machine hardware, which further precludes a meaningful cross-study runtime comparison. Although this comparison shows that the Adam-based methods achieve competitive solution quality, Adam need not be treated only as a standalone alternative. Its adaptive moment updates could in principle be incorporated into the dynamics of these solvers, making hybrid approaches a promising direction.

V. DISCUSSION

Our results show that optimizer performance depends strongly on whether the Ising machine is used as a continuous-time system or as a discrete iterative algorithm, and on how its hyperparameters are tuned. In the continuous-time setting, where the Euler time step Δt is kept sufficiently small to track the underlying dynamics, 1-ADAM-IM consistently gives the lowest time to target, the highest solution quality, and the highest transient success rates. ADAM-IM is the closest competitor, while MOM-IM and GD-IM fall substantially behind on the harder problem instances.

The strong continuous-time performance of 1-ADAM-IM can be understood from the additional freedom introduced by the first-order formulation. In the full ADAM-IM dynamics, the parameters (β_1, β_2) are restricted to the interval $[0, 1]$, inherited from the discrete algorithm where they represent exponential moving-average decay factors. In contrast, the first-order formulation removes both the time-dependent debiasing term and the probabilistic interpretation of (β_1, β_2) , allowing them to take arbitrary real values. This enlarged

parameter domain, with optimal values typically found in the range $\beta_1, \beta_2 \in (-120, 1)$, provides substantially more freedom to accelerate the internal dynamics. More negative values of these parameters, increase the relaxation rates $(1 - \beta_1)$ and $(1 - \beta_2)$. As a result, the moment variables v and w respond more quickly to changes in the gradient, producing faster dynamics and reducing the time needed to reach the target. A further possible reason is the different long-time behavior of the adaptive factor. In 1-ADAM-IM it decays to zero, whereas in ADAM-IM it approaches one. This slows the late-time 1-ADAM-IM dynamics, which may be beneficial near low-energy states where further improvements often require small local changes rather than large moves through the energy landscape.

The picture changes when the same optimizer ideas are used as purely discrete iterative algorithms. The algorithmic discrete-time setting leads to a different hierarchy. Once large discrete updates and standard optimizer rules are allowed, the performance gap between optimizers becomes smaller, especially on the unweighted GSET problem instances. On the harder weighted problem instances, however, ADAM-IM becomes the strongest method overall, followed by 1-ADAM-IM, while GD-IM and MOM-IM remain less reliable. Absolute TTT values should not be compared directly between the continuous-time and algorithmic regimes, since the former are measured in Euler time whereas the latter are measured in CPU time. Even for solution quality and transient success rate, one should be careful: algorithmic runs use a different tuning setup, different parameter ranges, only one Bayesian-optimization run rather than seven, fewer evaluations per run, and fewer stochastic trajectories per evaluation. The continuous-time figures should therefore be used to compare the continuous-time optimizers with each other, while the algorithmic figures should be used to compare the algorithmic optimizers with each other. They should not be read as a quantitative one-to-one comparison between continuous-time and algorithmic implementations.

With this caveat, the qualitative differences between the two regimes are still informative, especially when comparing the solution-quality and transient-success-rate trends in Figs. 6 and 7 with their algorithmic counterparts in Figs. 9 and 10. Keeping these differences in tuning and measurement in mind, 1-ADAM-IM shows similar continuous-time and algorithmic results on the unweighted GSET problem instances, while its algorithmic version is somewhat weaker on the weighted problem instances. ADAM-IM shows the opposite trend: it improves in the algorithmic setting, especially on the weighted benchmarks. This is consistent with the implementation comparison in the Supplemental Material [27], where the standard discrete Adam update, Eq. (18), gives the best algorithmic ADAM-IM results compared to the Euler-Maruyama discretization of the continuous-time Adam dynamics. One possible reason is that the Euler-Maruyama discretization of the continuous-time Adam equations is not identical to the original discrete Adam rule, Eq. (18), once large time steps are allowed. In the derivation, the standard Adam update is first rewritten in terms of Euler updates, but before taking the continuous-time limit the newly updated moment variables $v_i^{(j+1)}$ and $w_i^{(j+1)}$ in the x_i -update are replaced by their

previous-step values $v_i^{(j)}$ and $w_i^{(j)}$, as shown in Eq. (20). This changes the update only at order $O(\Delta t^2)$, so it is negligible in the continuous-time regime where Δt is kept small. In the algorithmic setting, however, Δt is allowed to be large, so this small- Δt equivalence no longer has to hold. The standard discrete Adam rule can therefore behave differently from the Euler-Maruyama discretization of the continuous-time Adam equations.

GD-IM also performs better in the algorithmic setting, especially on the easier problem instances. In the continuous-time setting, changing η would mainly rescale time: to keep the same numerical resolution of the underlying trajectory, Δt would have to be reduced so that $\eta\Delta t$ remains sufficiently small. In the algorithmic setting, this restriction is absent. Here Δt is itself an additional tunable step-size parameter, so the effective update scale can be adjusted through the product $\eta\Delta t$. Because useful time step values can span orders of magnitude, the Bayesian optimization searches over $\log_{10} \Delta t$ rather than Δt directly, while η is tuned linearly over its chosen range. This gives GD-IM a genuinely larger search space than in the continuous-time setting, where Δt is fixed by numerical-resolution requirements. MOM-IM benefits from the same algorithmic freedom, since its Euler-Maruyama version can also use Δt as an additional tunable step-size parameter. Compared with the continuous-time setting, this improves MOM-IM's transient success rate on the unweighted problem instances and its solution quality on some harder weighted problem instances, but this extra tuning freedom is not sufficient to close the gap to the Adam-based methods. Overall, compared with the continuous-time setting, the algorithmic setting reduces the differences between optimizers on easier problem instances. On the harder weighted problem instances, however, the best algorithmic performance comes from the standard discrete Adam update, given by Eq. (18).

A broader practical lesson from these results is that Ising-machine performance can depend strongly on how the hyperparameter search is organized. In our nonlinearity study, the coarse grid scans varied only the nonlinearity parameters (α, β) , with all optimizer-specific parameters fixed. This showed that the periodic nonlinearity can appear competitive with the sigmoid nonlinearity for Adam-based IMs. After full Bayesian optimization, however, the nonlinearity parameters and optimizer parameters are tuned jointly, and the sigmoid nonlinearity consistently emerged as the strongest choice. Thus, the apparent ranking of nonlinearities can change once the optimizer dynamics and nonlinear transfer function are optimized together.

This observation also motivated the seven-range Bayesian-optimization strategy used for the final GSET comparison in the continuous-time setting. Instead of using one very broad search domain, we split the search into several smaller ranges, which made it easier for Bayesian optimization to refine high-performing regions while keeping the total computational cost manageable. More generally, these results show that the choice of optimizer, nonlinearity, and hyperparameter-search domain should be treated together: careful tuning is not just an implementation detail, but an important part of achieving strong Ising-machine performance. In practice, our results support a two-stage tuning protocol: first a light grid

scan to set reasonable ranges, then range-split Bayesian optimization to refine the best-performing regions. Bayesian optimization improves upon the coarse grid-scan baseline for all optimizers, with the largest gain observed for 1-ADAM-IM. This is practically important because the refinement is not more sample-expensive than the grid scan itself: the coarse scan uses $30 \times 30 = 900$ parameter evaluations, whereas each Bayesian-optimization run uses 1000 evaluations. Moreover, the overhead of the Bayesian-optimization procedure itself is negligible compared with the cost of generating these samples. In other words, Bayesian optimization may appear much more sophisticated, but in our setting it operates at essentially the same sampling budget while delivering systematically better performance. This makes the combination of a coarse scan for orientation followed by full Bayesian optimization a particularly effective protocol.

A related practical point is the choice of Bayesian-optimization objective. In this work, the Bayesian optimization is based on a target-reaching metric, namely TTT. This is appropriate for the benchmark setting used here, where the g05 ground states are known and the GSET targets can be defined relative to best-known cut values. In a practical deployment, however, such a reference value is usually not known in advance. In that case, the Bayesian optimization would need a different objective, for example one that balances the best energy reached against the runtime needed to reach it.

Finally, because the continuous-time results are expressed partly through time to target, it is important to separate optimizer effects from simple time-scale effects. As discussed earlier, the different optimizers do not evolve on exactly the same natural timescale, so absolute differences in TTT measured in Euler time should be interpreted with some care. In particular, part of the observed difference between GD-IM, MOM-IM, ADAM-IM, and 1-ADAM-IM could in principle come from how the natural timescale of each optimizer is set, rather than from the optimizer dynamics alone. That caveat, however, cannot explain the main trend in our results. If the observed differences simply reflected different effective timescales, one would expect a simple global rescaling of time, $t \mapsto ct$, for each optimizer. Such a rescaling would multiply all TTT values of that optimizer by the same constant across benchmark problem instances. It would therefore not explain why the advantage of ADAM-IM and especially 1-ADAM-IM becomes progressively larger as the problems become harder, from g05 to GSET and then to the harder GSET problem instances. This conclusion is further supported by the solution-quality and transient-success-rate results, which are not themselves time-based measures: 1-ADAM-IM consistently reaches higher cut values and higher success rates, often by a substantial margin. The Supplemental Material [27] provides two complementary checks. First, MOM-IM is

reoptimized with $\eta \in [1, 200]$, and η is also tuned directly for GD-IM in the algorithmic comparison. Optimizing η gives MOM-IM only modest improvements in transient success rate and cut quality and reduces its raw TTT, but it does not close the one-to-four-order-of-magnitude gap to 1-ADAM-IM; the optimizer ranking is unchanged. Second, for the continuous-time comparison, the TTT values are rescaled by the corresponding learning-rate factors. Multiplying TTT by η is the time rescaling that GD-IM would undergo if it used the same value of η , so this diagnostic attempts to match the effective timescales of the optimizers. This timescale matching does not change the optimizer ranking or any of the conclusions; instead, it further strengthens the evidence that the superior performance of the Adam-based methods arises from their optimizer dynamics rather than from a trivial learning-rate choice or a difference in effective timescale.

Taken together, these results show that Adam-type dynamics do more than rescale gradient descent: they change the optimization behavior in a way that becomes increasingly beneficial on harder problem instances. In the continuous-time setting, the 1-ADAM-IM formulation is the strongest method across all benchmarks, improving not only TTT but also solution quality and transient success rate. In the algorithmic setting, the advantage shifts to the standard discrete Adam update, which gives the best performance on the harder weighted instances. This makes Adam-type optimization a promising design principle both for improving analog, time-continuous Ising machines and for guiding future algorithmic Ising-machine variants.

ACKNOWLEDGMENTS

We thank Toon Sevenants for helpful discussions and for feedback on the figures and their presentation. This research was funded by the Research Foundation Flanders (FWO) under Grants No. G0A6L25N and No. G006020N. Additional funding was provided by the EOS project “Photonic Ising Machines.” This project (EOS No. 40007536) has received funding from the FWO and F.R.S.-FNRS under the Excellence of Science (EOS) program. The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation Flanders (FWO) and the Flemish Government Department WEWIS.

DATA AVAILABILITY

The code and processed data used in this work are available at [31]. The repository contains the continuous-time and discrete-time Ising-machine simulation code, Bayesian-optimization scripts, processed benchmark results, and plotting code.

- [1] T. N. Theis and H.-S. P. Wong, The end of Moore’s law: A new beginning for information technology, *Comput. Sci. Eng.* **19**, 41 (2017).
- [2] J. Shalf, The future of computing beyond Moore’s Law, *Philos. Trans. R. Soc. A* **378**, 20190061 (2020).

- [3] C. E. Leiserson, N. C. Thompson, J. S. Emer, B. C. Kuszmaul, B. W. Lampson, D. Sanchez, and T. B. Schardl, There’s plenty of room at the top: What will drive computer performance after Moore’s law? *Science* **368**, eaam9744 (2020).

- [4] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, *et al.*, Quantum annealing with manufactured spins, *Nature (London)* **473**, 194 (2011).
- [5] Y. Yamamoto, K. Aihara, T. Leleu, K.-i. Kawarabayashi, S. Kako, M. Fejer, K. Inoue, and H. Takesue, Coherent Ising machines—Optical neural networks operating at the quantum limit, *npj Quantum Inf.* **3**, 49 (2017).
- [6] 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.-i. Kawarabayashi, K. Inoue, S. Utsunomiya, and H. Takesue, A coherent Ising machine for 2000-node optimization problems, *Science* **354**, 603 (2016).
- [7] F. Böhm, D. Alonso-Urquijo, G. Verschaffelt, and G. Van der Sande, Noise-injected analog Ising machines enable ultrafast statistical sampling and machine learning, *Nat. Commun.* **13**, 5847 (2022).
- [8] A. Lucas, Ising formulations of many NP problems, *Front. Phys.* **2**, 1 (2014).
- [9] Y. Zhang, Y. Deng, Y. Lin, Y. Jiang, Y. Dong, X. Chen, G. Wang, D. Shang, Q. Wang, H. Yu, and Z. Wang, Oscillator-network-based Ising machine, *Micromachines* **13**, 1016 (2022).
- [10] M. Babaeian, D. T. Nguyen, V. Demir, M. Akbulut, P.-A. Blanche, Y. Kaneda, S. Guha, M. A. Neifeld, and N. Peyghambarian, A single shot coherent Ising machine based on a network of injection-locked multicore fiber lasers, *Nat. Commun.* **10**, 3516 (2019).
- [11] H. Goto, K. Endo, M. Suzuki, Y. Sakai, T. Kanao, Y. Hamakawa, R. Hidaka, M. Yamasaki, and K. Tatsumura, High-performance combinatorial optimization based on classical mechanics, *Sci. Adv.* **7**, eabe7953 (2021).
- [12] T. Okuyama, T. Sonobe, K.-i. Kawarabayashi, and M. Yamaoka, Binary optimization by momentum annealing, *Phys. Rev. E* **100**, 012111 (2019).
- [13] D. Pierangeli, G. Marcucci, and C. Conti, Large-scale photonic Ising machine by spatial light modulation, *Phys. Rev. Lett.* **122**, 213902 (2019).
- [14] H. Goto, K. Tatsumura, and A. R. Dixon, Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems, *Sci. Adv.* **5**, eaav2372 (2019).
- [15] L. Mys, G. Verschaffelt, and G. Van der Sande, Predicting the optimal noise strength for solving optimization problems with analog Ising machines, *Phys. Rev. Applied* **25**, 034068 (2026).
- [16] B. T. Polyak, Some methods of speeding up the convergence of iteration methods, *USSR Comput. Math. Math. Phys.* **4**, 1 (1964).
- [17] N. Mohseni, P. L. McMahon, and T. Byrnes, Ising machines as hardware solvers of combinatorial optimization problems, *Nat. Rev. Phys.* **4**, 363 (2022).
- [18] R. A. Brown, D. Venturelli, M. Pavone, and D. E. Bernal Neira, Accelerating continuous variable coherent Ising machines via momentum, in *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, edited by B. Dilkina (Springer Nature Switzerland, Cham, 2024), pp. 109–126.
- [19] A. Barakat and P. Bianchi, Convergence and dynamical behavior of the ADAM algorithm for nonconvex stochastic optimization, *SIAM J. Optim.* **31**, 244 (2021).
- [20] F. Böhm, T. V. Vaerenbergh, G. Verschaffelt, and G. Van der Sande, Order-of-magnitude differences in computational performance of analog Ising machines induced by the choice of nonlinearity, *Commun. Phys.* **4**, 149 (2021).
- [21] H. Goto, R. Hidaka, and K. Tatsumura, Edge-of-chaos enhanced quantum-inspired algorithm for combinatorial optimization, *Phys. Rev. Appl.* **25**, 044011 (2026).
- [22] T. Leleu, Y. Yamamoto, P. L. McMahon, and K. Aihara, Destabilization of local minima in analog spin systems by correction of amplitude heterogeneity, *Phys. Rev. Lett.* **122**, 040607 (2019).
- [23] T. Leleu, F. Khoiratee, T. Levi, R. Hamerly, T. Kohno, and K. Aihara, Scaling advantage of chaotic amplitude control for high-performance combinatorial optimization, *Commun. Phys.* **4**, 266 (2021).
- [24] T. Leleu, S. Reifenshtein, A. Yamamura, and S. Ganguli, Reshaping global loop structure to accelerate local optimization by smoothing rugged landscapes, *arXiv:2602.01490*.
- [25] A. B. Da Silva and M. Gazeau, A general system of differential equations to model first-order adaptive algorithms, *J. Mach. Learn. Res.* **21**, 5072 (2020).
- [26] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *3rd International Conference on Learning Representations (ICLR, San Diego, CA, USA, 2015)*.
- [27] See Supplemental Material at <http://link.aps.org/supplemental/10.1103/qg6g-nkf3> for Bayesian-optimization details, learning-rate and nonlinearity checks, benchmark cut values, implementation comparisons, and TTT diagnostics, which includes Refs. [11,22,23,30].
- [28] F. Rendl, G. Rinaldi, and A. Wiegale, Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations, *Math. Program.* **121**, 307 (2010).
- [29] Y. Ye, Gset Max-Cut Problem Set (2003), <https://web.stanford.edu/~yyye/yyye/Gset/>.
- [30] R. Martinez-Cantin, BayesOpt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits, *J. Mach. Learn. Res.* **15**, 3915 (2014).
- [31] <https://github.com/StijnVanVooren/Adam-analog-ising-machines>.