Wang-Landau algorithm as stochastic optimization and its acceleration

Chenguang Dai and Jun S. Liu

Phys. Rev. E **101**, 033301 — Published 6 March 2020

DOI: 10.1103/PhysRevE.101.033301

The Wang-Landau Algorithm as Stochastic Optimization and Its Acceleration

Chenguang Dai* and Jun S. Liu†
Department of Statistics, Harvard University

(Dated: February 3, 2020)

We show that the Wang-Landau algorithm can be formulated as a stochastic gradient descent algorithm minimizing a smooth and convex objective function, of which the gradient is estimated using Markov chain Monte Carlo iterations. The optimization formulation provides us a new way to establish the convergence rate of the Wang-Landau algorithm, by exploiting the fact that almost surely, the density estimates (on the logarithmic scale) remain in a compact set, upon which the objective function is strongly convex. The optimization viewpoint motivates us to improve the efficiency of the Wang-Landau algorithm using popular tools including the momentum method and the adaptive learning rate method. We demonstrate the accelerated Wang-Landau algorithm on a two-dimensional Ising model and a two-dimensional ten-state Potts model.

I. INTRODUCTION

The Wang-Landau (WL) algorithm [1–3] has been proven useful in solving a wide range of computational problems in statistical physics, including spin-glass models [4–15], fluid phase equilibria [16, 17], polymers [18, 19], lattice gauge theory [20], protein folding [21–23], free energy profile [24], and numerical integration [25, 26]. Its successful applications in statistics have also been documented [27–29]. The WL algorithm directly targets the density of states (the number of all possible configurations for an energy level of a system), thus allowing us to calculate thermodynamic quantities over an arbitrary range of temperature within a single run of the algorithm.

Much effort has been made to understand the dynamics of the WL algorithm, along with numerous proposed improvements, of which we highlight three here. (i) Optimizing the modification factor (flatness criterion) [30–33]. Belardinelli and Pereyra [30] proposed that instead of reducing the modification factor exponentially, the log modification factor should be scaled down at the rate of $1/t$ in order to avoid the saturation in the error. (ii) Employing a Parallelization scheme. Wang and Landau [1] suggested that multiple random walkers working simultaneously on the same density of states can accelerate the convergence of the WL algorithm. The efficiency of the parallelization scheme can be further enhanced using the replica-exchange framework [34]. (iii) Incorporating efficient Monte Carlo trial moves [35–37].

In this paper, we consider the WL algorithm from an optimization perspective and formulate it as a first-order method. We derive the corresponding smooth and convex objective function, of which the gradient involves the unknown density of states. Wang and Landau [1] used a random-walk based Metropolis algorithm [38] to estimate the gradient. In general, any suitable Markov chain Monte Carlo (MCMC) strategies [39] can be employed for this purpose. Therefore, the WL algorithm is essentially a stochastic gradient descent algorithm.

The optimization viewpoint enables us to establish the convergence rate of the WL algorithm. Following [40] and using the standard stochastic approximation theory [41], we first show that the density estimates (on the logarithmic scale) almost surely stay in a compact set. Based on this, we exploit the strong convexity of the objective function, restricted on this compact set, to prove the convergence rate. We note that the gradient estimator output from the MCMC iterations is generally biased, thus a critical step is to show that the bias vanishes properly as $t \to \infty$.

The optimization framework also provides us with a new direction for improving the WL algorithm. We explore one possible improvement, by combining the momentum method [42] and the adaptive learning rate method [43, 44]. The general goal is to accelerate the transient phase [45] of the WL algorithm before it enters the fine local convergence regime. The effectiveness of the acceleration method is demonstrated on a two-dimensional Ising model and a two-dimensional ten-state Potts model, in which the learning in the transient phase is considerably demanding.

The rest of the paper is organized as follows. Section II discusses the optimization formulation of the WL algorithm, and establishes the convergence rate from an optimization perspective. Section III introduces possible strategies to accelerate the WL algorithm using optimization tools. Section IV demonstrates the accelerated WL algorithm on two benchmark examples. Finally, Section V concludes with a few remarks.

II. AN OPTIMIZATION FORMULATION

Let the space of all microscopic configurations be X. Suppose there are totally N energy levels, $E_1 < \cdots < E_N$, for the underlying physical model. For a microscopic configuration $x \in \mathsf{X}$, we use $E(x)$ to denote its energy.

* chenguangdai@g.harvard.edu
† jliu@stat.harvard.edu

Let $\{g(E_n)\}_{n=1}^N$ be the normalized density of states, i.e.,

$$g(E_n) \propto \#\{x \in \mathsf{X}, E(x) = E_n\}, \quad \sum_{n=1}^N g(E_n) = 1. \tag{1}$$

After initializing $g_0(E_n)$ as $1/N$, the WL algorithm iterates between the following two steps: (i) Propose a transition configuration and accept it with probability $\min\{1, g_t(E_i)/g_t(E_j)\}$, where E_i and E_j refer to the energy levels before and after this transition, respectively. This is essentially a step of the Metropolis algorithm [38] with the corresponding stationary distribution:

$$\pi_t(x) \propto \sum_{n=1}^N \frac{1}{g_t(E_n)} \mathbb{1}\left(E(x) = E_n\right). \tag{2}$$

(ii) Update the density of states. If $E(x_{t+1}) = E_n$, multiply $g_t(E_n)$ by a modification factor $f_{t+1} > 1$. That is, $g_{t+1}(E_n) \leftarrow g_t(E_n) \times f_{t+1}$.

The modification factor f_t should be properly scaled down in order to guarantee the convergence of the algorithm. There is a rich literature on how to adapt f_t online, including the flat/minimum histogram criterion, and the $1/t$ rule [30] with its various extensions [46, 47]. Under a proper scaling rule, the magnitude of the modification factor f_t is informative of the estimation error [31]. Thus, a commonly used stopping criteria for the WL algorithm is that f_t is small enough (say, below $\exp(10^{-8})$).

In the following, we will work on the logarithmic scale of the density of states. Denote $u_n^{(t)} = \log(g_t(E_n))$ for $n \in [N]$, and let $\boldsymbol{u} = (u_1, \cdots, u_N)$. The density update in the WL algorithm can be rewritten as

$$u_n^{(t+1)} \leftarrow u_n^{(t)} + \eta_{t+1} \mathbb{1}(E(x_{t+1}) = E_n), \tag{3}$$

where $\eta_{t+1} = \log f_{t+1}$, which will be referred to as the learning rate henceforth. The intermediate target distribution $\pi_t(x)$ defined in Equation (2) can also be formulated in terms of $\boldsymbol{u}^{(t)}$. We define

$$\pi_{\boldsymbol{u}}(x) \propto \sum_{n=1}^N \exp(-u_n) \mathbb{1}\left(E(x) = E_n\right), \tag{4}$$

and denote $P_{\boldsymbol{u}}$ as a general transition kernel invariant to $\pi_{\boldsymbol{u}}(x)$. For notational convenience, we use $\pi_t(x)$ to refer to $\pi_{\boldsymbol{u}^{(t)}}(x)$, and use P_t to refer to the transition kernel invariant to $\pi_t(x)$. After each density update, we normalize $\boldsymbol{u}^{(t)}$ to sum to 0, i.e., $u_n^{(t)} \leftarrow u_n^{(t)} - \sum_{i=1}^N u_i^{(t)}/N$, so that $\boldsymbol{u}^{(t)}$ stays in a compact set (see Proposition 1). The WL algorithm can be slightly rephrased as in Algorithm 1.

Let us consider the following optimization problem:

$$\min_{\boldsymbol{u} \in \mathbb{R}^N} h(\boldsymbol{u}) = \log\left(\sum_{n=1}^N \exp(u_n^\star - u_n)\right), \quad \text{subject to} \sum_{n=1}^N u_n = 0, \tag{5}$$

in which $u_n^\star = \log(g(E_n)) - \frac{1}{N}\sum_{i=1}^N \log(g(E_i))$. We write $\boldsymbol{u}^\star = (u_1^\star, \cdots, u_N^\star)$. It is not difficult to see that this is a convex optimization problem because the objective function $h(\boldsymbol{u})$ is a log-sum-exp function and the constraint is linear. It has a unique solution at $u_n = u_n^\star$ for $n \in [N]$, in which $\exp(u_n^\star)$ equals to the density of states $g(E_n)$ up to an multiplicative constant.

Algorithm 1: The Wang-Landau algorithm

```
1. Initialization. $u_n^{(0)} = 0$ for $n \in [N]$.
2. For $t \geq 1$, iterate between the following steps.
    (a) Sample $x_{t+1}$ from $P_t(x_t, \cdot)$.
    (b) Update $\boldsymbol{u}^{(t+1)}$ following Equation (3).
    (c) Normalize $\boldsymbol{u}^{(t+1)}$ to sum to 0.
    (d) Scale down the learning rate $\eta_t$ properly.
3. Stop when the learning rate $\eta_t$ is smaller than a
   prescribed threshold.
```

The projected gradient descent algorithm is a standard approach to solve the constrained optimization problem (5). The gradient of the objective function $h(\boldsymbol{u})$ is

$$\frac{\partial h(\boldsymbol{u})}{\partial u_n} = -\frac{\exp\left(u_n^\star - u_n\right)}{\sum_{i=1}^N \exp\left(u_i^\star - u_i\right)}, \quad n \in [N], \tag{6}$$

which is not directly available because it involves the unknown density of states. However, one can think of approximating the gradient function defined in Equation (6) by one-step or multiple-step Monte Carlo simulations, leading to a stochastic version of the projected gradient descent algorithm.

More precisely, a gradient descent step for minimizing $h(\boldsymbol{u})$ takes the following form:

$$u_n^{(t+1)} \leftarrow u_n^{(t)} + \frac{\eta_{t+1} \exp(u_n^\star - u_n^{(t)})}{\sum_{i=1}^N \exp(u_i^\star - u_i^{(t)})}. \tag{7}$$

Denote the probability of the set $\{x \in \mathsf{X} : E(x) = E_n\}$ with respect to $\pi_t(x)$ as $\pi_t(E_n)$. Since the probability $\pi_t(E_n)$ is proportional to $\exp(u_n^\star - u_n^{(t)})$, the density update in Equation (7) is essentially

$$u_n^{(t+1)} \leftarrow u_n^{(t)} + \eta_{t+1} \pi_t(E_n). \tag{8}$$

A crude approximation to $\pi_t(E_n)$ is the indicator function $\mathbb{1}\left(E(x_{t+1}) = E_i\right)$, given that after several steps of Monte Carlo simulations according to the transition kernel P_t invariant to $\pi_t(x)$, x_{t+1} is approximately a sample from $\pi_t(x)$. This corresponds to the density update in Equation (3).

We note that the projection step to the set $\Pi = \{\boldsymbol{u} \in \mathbb{R}^N, \sum_{n=1}^N u_n = 0\}$ is equivalent to the normalization step (see Algorithm 1 step 2(c)). Thus, we have shown that the stochastic projected gradient descent algorithm solving the constrained optimization problem (5), which estimates the probability $\pi_t(E_n)$ by $\mathbb{1}\left(E(x_{t+1})=E_n\right)$ using the output from Monte Carlo simulations, is equivalent to the WL algorithm.

The above optimization formulation has the following immediate implications. First, the parallel WL algorithm estimates the negative gradient $\pi_t(E_n)$ by $1/m\sum_{k=1}^{m}[\mathbb{1}(E(x_t^{(k)})=E_n)]$, in which m denotes the total number of random walkers, and $x_t^{(k)}$ denotes the kth random walker. Therefore, it reduces the variance of the gradient estimate by a factor m. Second, instead of implementing a single transition step, the separation strategy mentioned in [31] implements multiple transition steps within each iteration, so that the law of the random walker gets closer to the intermediate target distribution $\pi_t(x)$ defined in Equation (4). Therefore, it reduces the bias of the gradient estimate.

The optimization formulation also points out a new approach to establish the convergence rate of the WL algorithm. We first state a required assumption, which assumes that the transition kernels are (uniformly) geometrically ergodic over the space Π.

Assumption 1 *There exists a constant $\rho\in(0,1)$ such that for all $\boldsymbol{u}\in\Pi$, $x\in\mathsf{X}$, $k\in\mathbb{N}$, we have*

$$\sup_{\boldsymbol{u}\in\Pi}\sup_{x\in\mathsf{X}}||P_{\boldsymbol{u}}^k(x,\cdot)-\pi_{\boldsymbol{u}}||_{\mathrm{TV}}\le 2(1-\rho)^k, \tag{9}$$

in which for a signed measure μ, the total variation norm is defined as

$$||\mu||_{\mathrm{TV}}=\sup_{|q|\le 1}\left|\int_{\mathsf{X}} q(x)\mu(dx)\right|. \tag{10}$$

We note that sufficient conditions for Assumption 1 exist in the literature (e.g., condition A2 in [40]), and relaxation of Assumption 1 is also possible [41]. We have the following result.

Proposition 1 *Under Assumption 1, if we scale down the learning rate η_t in the order of $O(1/t)$, the following two statements hold.*

1. *Almost surely convergence.*
   - *(a) There exists a compact set $\mathcal{K}\subseteq\Pi$ such that for any $t\ge 0$, $\boldsymbol{u}^{(t)}\in\mathcal{K}$ almost surely.*
   - *(b) $\mathbb{P}(\lim_{t\to\infty}\boldsymbol{u}^{(t)}=\boldsymbol{u}^\star)=1$.*
2. *Convergence rate. There exists a constant $C>0$ such that*

$$\mathbb{E}||\boldsymbol{u}^{(t)}-\boldsymbol{u}^\star||^2\le C/t. \tag{11}$$

The proof of Proposition 1 is given in the Supplemental Material [48].

The first part of Proposition 1 follows similarly as [40]. The main idea is to rewrite the WL update, including the density update and the normalization step, as

$$\boldsymbol{u}^{(t+1)}\leftarrow\boldsymbol{u}^{(t)}+\eta_{t+1}\boldsymbol{r}(\boldsymbol{u}^{(t)})+\eta_{t+1}(\boldsymbol{R}(x_{t+1})-\boldsymbol{r}(\boldsymbol{u}^{(t)})),$$

in which $R_n(x)=\mathbb{1}(E(x)=E_n)-1/N$, and $r(\boldsymbol{u})$ is the mean-field function defined as

$$\boldsymbol{r}(\boldsymbol{u})=\int_{\mathsf{X}}\boldsymbol{R}(x)\pi_{\boldsymbol{u}}(x)dx=\frac{\exp(\boldsymbol{u}^\star-\boldsymbol{u})}{\sum_{n=1}^N\exp(u_n^\star-u_n)}-\frac{1}{N}.$$

The proof of the almost-sure convergence concludes by applying the standard stochastic approximation theory (Theorem 2.2 and Theorem 2.3 in [49]) after we establish the following two facts. (1) The remainder term $\eta_{t+1}(\boldsymbol{R}(x_{t+1})-\boldsymbol{r}(\boldsymbol{u}^{(t)}))$ vanishes properly as $t\to\infty$. (2) There exists a Lyapunov function $V(\boldsymbol{u})$ specified below,

$$V(\boldsymbol{u})=\frac{1}{N}\sum_{n=1}^N\exp(u_n^\star-u_n)-1, \tag{12}$$

with respect to the mean-field function $r(\boldsymbol{u})$, such that $\langle\nabla V(\boldsymbol{u}),\boldsymbol{r}(\boldsymbol{u})\rangle<0$, $\forall\,\boldsymbol{u}\neq\boldsymbol{u}^\star$, and $\langle\nabla V(\boldsymbol{u}^\star),\boldsymbol{r}(\boldsymbol{u}^\star)\rangle=0$.

The second part of Proposition 1 is our main theoretical contribution. There are two essential ingredients in establishing the convergence rate. (i) Strong convexity. The objective function $h(\boldsymbol{u})$ is only convex but not strongly convex on $\mathbb{R}^N$. However, because $\boldsymbol{u}^{(t)}$ stays in a compact set $\mathcal{K}\subseteq\Pi$ almost surely (see Proposition 1, part 1(a)), we are able to establish the strong convexity of $h(\boldsymbol{u})$ restricted on this compact set $\mathcal{K}$.

Lemma 1 *Under Assumption 1, there exists a constant $\ell>0$ such that for any $t\ge 0$, almost surely, it holds*

$$\langle\nabla h(\boldsymbol{u}^{(t)}),\boldsymbol{u}^{(t)}-\boldsymbol{u}^\star\rangle\ge\ell||\boldsymbol{u}^{(t)}-\boldsymbol{u}^\star||^2. \tag{13}$$

(ii) Vanishing bias. Because x_{t+1} is only an approximate sample from the intermediate target distribution $\pi_t(x)$, the indicator $\mathbb{1}\left(E(x_{t+1})=E_n\right)$ is not an unbiased estimator to the negative gradient $\pi_t(E_n)$. The following Lemma 2 shows that the bias of the gradient estimator vanishes properly, as fast as the learning rate, when $t\to\infty$.

Lemma 2 *Under Assumption 1, there exists a constant $C>0$ such that*

$$\mathbb{E}||\pi_t-P_t(x_t,\cdot)||_{\mathrm{TV}}\le C\eta_{t+1}. \tag{14}$$

The convergence rate of the WL algorithm has been established in different forms in the literature. Zhou and Bhatt [31] show that the discrete probability distribution $\{\pi_t(E_n)\}_{n=1}^N$ will be attracted, in terms of the KL-divergence, to the vicinity of the uniform distribution ($\pi_\infty(E_n)=1/N$) as $t\to\infty$. In addition, they show that the standard deviation of $\exp(u_n^\star-u_n^{(t)})$ roughly scales like $\sqrt{\log f_t}$ when the modification factor f_t is close to 1. Although we are looking at the L^2 error of $\boldsymbol{u}^{(t)}$, which is slightly different from the aforementioned standard deviation, their convergence rate is consistent with our result because $\sqrt{\log f_t}=\sqrt{\eta_t}$ is in the order of $O(1/\sqrt{t})$ if we scale down the learning rate η_t in the order of $O(1/t)$. It is also worthwhile to mention that a corresponding central limit theorem in the original density space is provided in [40].

III. ACCELERATING WANG-LANDAU ALGORITHM

The optimization formulation motivates us to further improve the WL algorithm using optimization tools [50]. Our goal in this paper is to accelerate the convergence in the transient phase. The transient phase [45] generally refers to the initial stage of running a stochastic gradient descent algorithm. For instance, if we scale down the learning rate according to the flat/minimum histogram criterion, we can refer to the transient phase as the running period from the beginning up to the time when the flat/minimum histogram criterion is first satisfied.

When the transient phase appears noticeable, the acceleration tools can be very effective in practice, and have been widely used in large-scale systems such as deep neural networks [51]. In this paper, we restrict ourselves on the first-order acceleration methods, and leave other possibilities for future explorations. In particular, we find that both the momentum method and the adaptive learning rate method are effective in accelerating the WL algorithm. Before we go into details, we note that improvement in the asymptotic convergence rate of the stochastic gradient descent algorithm is hard to achieve (or even impossible) [52, 53] except for some well-structured objective functions such as finite sums.

The momentum method exponentially accumulates a momentum vector, denoted as $\boldsymbol{m}_t$ in the following, to amplify the persistent gradient across iterations. The basic momentum update operates as follows:

$$\begin{aligned} \boldsymbol{m}^{(t)} &\leftarrow \beta \boldsymbol{m}^{(t-1)} + \eta_{t+1} \nabla h(\boldsymbol{u}^{(t)}), \\ \boldsymbol{u}^{(t+1)} &\leftarrow \boldsymbol{u}^{(t)} - \boldsymbol{m}^{(t)}, \end{aligned} \tag{15}$$

where we initialize the momentum vector to be $\boldsymbol{m}^{(0)} = \boldsymbol{0}$. We note that the momentum update essentially adds a fraction β of the previously accumulated gradients $\boldsymbol{m}^{(t-1)}$ into the current update vector $\boldsymbol{m}^{(t)}$. The weighting factor β is a tuning parameter, and is commonly set to be 0.9 or higher.

In the setting of the WL algorithm, the momentum update in Equation (15) becomes

$$\begin{aligned} m_n^{(t)} &\leftarrow \beta m_n^{(t-1)} - \eta_{t+1} \mathbb{1}(E(x_{t+1}) = E_n), \\ u_n^{(t+1)} &\leftarrow u_n^{(t)} - m_n^{(t)}, \qquad \forall n \in [N]. \end{aligned} \tag{16}$$

The intuition behind the momentum acceleration for the WL algorithm can be heuristically described as follows. The event $E(x_{t+1}) = E_n$ suggests that $\pi_t(E_n)$ is likely larger than $1/N$, thus the Markov kernel P_t has a better chance to transit the microscopic configuration x_t into the energy level E_n. Therefore, in order to push $\pi_t(E_n)$ towards $1/N$, that is, downweight the probability mass in the energy level E_n, we increase $u_n^{(t)}$ by η_{t+1}, which corresponds to the density update in Equation (3). In contrast to the WL algorithm, which only increases $u_n^{(t)}$ by η_{t+1} at the current iteration t, we keep increasing $u_n^{(t)}$ for a few more iterations by an exponentially decay momentum $m_n^{(t)}$ to achieve a faster convergence.

The adaptive learning rate method helps standardize the gradient across different coordinates of the parameter $\boldsymbol{u}$, so that they scale in a similar magnitude. Otherwise, it can be challenging to find a suitable global learning rate η_t over different coordinates. Popular algorithms along this research direction include AdaGrad [43], AdaDelta [44], and RMSprop (an unpublished method proposed by Geoffrey Hinton). The RMSprop update operates as follows:

$$\begin{aligned} \boldsymbol{G}^{(t)} &\leftarrow \gamma \boldsymbol{G}^{(t-1)} + (1-\gamma) \nabla h(\boldsymbol{u}^{(t)})^2, \\ \boldsymbol{u}^{(t+1)} &\leftarrow \boldsymbol{u}^{(t)} - \eta_{t+1} [\boldsymbol{G}^{(t)}]^{-1/2} \nabla h(\boldsymbol{u}^{(t)}), \end{aligned} \tag{17}$$

in which both the square and the square root are taken elementwise. $\boldsymbol{G}^{(t)}$ represents the moving average of the squared gradients, so that the current gradient $\nabla h(\boldsymbol{u}^{(t)})$, standardized by $[\boldsymbol{G}^{(t)}]^{1/2}$, is in a similar magnitude across different coordinates. The weighting factor γ is a tuning parameter, which is commonly set to be 0.9 in order to prevent the updates from diminishing too fast. In the setting of the WL algorithm, the RMSprop update in Equation (17) becomes

$$\begin{aligned} G_n^{(t)} &\leftarrow \gamma G_n^{(t-1)} + (1-\gamma) \mathbb{1}(E(x_{t+1}) = E_n), \\ u_n^{(t+1)} &\leftarrow u_n^{(t)} - \eta_{t+1} [G_n^{(t)}]^{-1/2} \mathbb{1}(E(x_{t+1}) = E_n). \end{aligned} \tag{18}$$

The combination of the momentum method and the adaptive learning rate method leads to the Adaptive Moment Estimation (Adam) method [54]. The Adam update operates as follows:

$$\begin{aligned} \boldsymbol{m}^{(t)} &\leftarrow \beta \boldsymbol{m}^{(t-1)} + (1-\beta) \nabla h(\boldsymbol{u}^{(t)}), \\ \boldsymbol{G}^{(t)} &\leftarrow \gamma \boldsymbol{G}^{(t-1)} + (1-\gamma) \nabla h(\boldsymbol{u}^{(t)})^2, \\ \boldsymbol{u}^{(t+1)} &\leftarrow \boldsymbol{u}^{(t)} - \eta_{t+1} [\boldsymbol{G}^{(t)}]^{-1/2} \boldsymbol{m}^{(t)}. \end{aligned} \tag{19}$$

In the setting of the WL algorithm, we note that, although β and γ can be potentially two tuning parameters, if we set $\beta = \gamma$ and initialize $\boldsymbol{m}^{(0)}$ and $\boldsymbol{G}^{(0)}$ to be $\boldsymbol{0}$, we have $\boldsymbol{G}^{(t)} = -\boldsymbol{m}^{(t)}$, since $-\nabla h(\boldsymbol{u}^{(t)})$ is approximated by a one-hot vector, which contains only a single "1" with the remaining elements being 0. This simplification leads to Algorithm 2, which we refer to as the AWL algorithm henceforth.

We remark that for large-scale systems, a naive implementation of Equation (20) can be very inefficient, as we have to loop over every coordinate of $\boldsymbol{m}^{(t)}$ and $\boldsymbol{u}^{(t)}$ in each iteration. A simple solution is to introduce a vector $\boldsymbol{s} = (s_1, \cdots, s_N)$, in which s_n records the last time when m_n and u_n are updated. With the help of s_n, instead of updating m_n and u_n in each iteration, we shall update them only when the energy level E_n is involved in the Monte Carlo simulations.

Algorithm 2: Accelerated Wang-Landau algorithm

```
1. Initialization. $u_n^{(0)} = 0$, $m_n^{(0)} = 0$ for $n \in [N]$.
2. For $t \geq 1$, iterate between the following steps.
    (a) Sample $x_{t+1}$ from $P_t(x_t, \cdot)$.
    (b) Update $\boldsymbol{m}^{(t)}$ and $\boldsymbol{u}^{(t+1)}$ as follows.
        $m_n^{(t)} \leftarrow \beta m_n^{(t-1)} + (1-\beta)\mathbb{1}(E(x_{t+1}) = E_n)$,
        $u_n^{(t+1)} \leftarrow u_n^{(t)} + \eta_{t+1}[m_n^{(t)}]^{1/2}$.        (20)
    (c) Normalize $\boldsymbol{u}^{(t+1)}$ to sum to 0.
    (d) Scale down the learning rate $\eta_t$ properly.
3. Stop when the learning rate $\eta_t$ is smaller than a prescribed threshold.
```

IV. ILLUSTRATIONS

We compare the AWL algorithm with the original WL algorithm on two benchmark examples: (a) a nearest-neighbour Ising model; (b) a nearest-neighbour ten-state Potts model. Both models are defined on a two-dimensional $L \times L$ square lattice equipped with the periodic boundary condition.

For the Ising model, the energy $E(x)$ is given by the Hamiltonian:

$$E(x) = -\sum_{<i,j>} J_{ij} x_i x_j - \psi \sum_j b_j x_j, \tag{21}$$

where $x_i \in \{\pm 1\}$. The subscripts i, j denote the lattice sites, and the notation $< i, j >$ implies that the site i and the site j are nearest neighbors. For the ten-state Potts model, the energy $E(x)$ is given by:

$$E(x) = -\sum_{<i,j>} J_{ij} \mathbb{1}(x_i = x_j) - \psi \sum_j b_j x_j, \tag{22}$$

where $x_i \in \{1, \cdots, 10\}$. For both models, we assume that $J_{ij} \equiv 1$ and $b_j \equiv 0$ (no external magnetic field). If $b_j \equiv 0$, the two-dimensional Ising model exhibits a second-order phase transition. Otherwise, in the presence of an external magnetic field, the two-dimensional Ising model exhibits a first-order phase transition. When $b_j \equiv 0$, the two-dimensional Potts model exhibits a first-order phase transition when the number of states is larger than 4.

Let $\{H_t(E_n)\}_{n=1}^N$ be the histogram of all energy levels at iteration t. We initialize $H_0(E_n) = 0$ for $n \in [N]$. At each iteration t, the AWL algorithm and the WL algorithm update $\boldsymbol{u}^{(t)}$ according to Algorithm 2 and Algorithm 1, respectively. In addition, we update the energy histogram as $H_t(E_n) = H_{t-1}(E_n) + \mathbb{1}(E(x_{t+1}) = E_n)$.

The adaptation of the learning rate η_t follows [30], which is detailed in the following.

1. After every 1,000 MC sweeps, we check $\{H_t(E_n)\}$. If $\min_n H_t(E_n) > 0$, we set $\eta_{t+1} = \eta_t/2$, and reset $H_t(E_n) = 0$ for each energy level E_n. Otherwise if $\min_n H_t(E_n) = 0$, we keep $\eta_{t+1} = \eta_t$.
2. If $\eta_{t+1} \leq N/t$, then $\eta_t = N/t$ for all the subsequent iterations. $H_t(E_n)$ is discarded and the above step is not executed any more.

We note that each MC sweep contains L^2 iterations, in which each iteration refers to a single round of parameter update. That is, step 2(a)–2(c) in Algorithm 1 and Algorithm 2. The energy histogram $\{H_t(E_n)\}$ essentially represents the number of visits to each energy level up to iteration t, since the last update of the learning rate.

We implement one step of the Metropolis algorithm to estimate the gradient, i.e., step 2(a) in Algorithm 1 and Algorithm 2. The proposal schemes for the Ising model and the Potts model are described as follows. Given the current configuration x_t, we randomly pick up a site and change its value. For the Ising model, we filp its sign. For the ten-state Potts model, we set it to be a number uniformly sampled from $\{1, \cdots, 10\}$.

To illustrate the efficiency of the AWL algorithm, we investigate the following four perspectives. (i) The scaling of the first equilibration time, in terms of the number of MC sweeps, with respect to the dimension L. The first equilibration time, which corresponds to the transient phase as we discussed in Section III, is defined to be $\min\{t : \min_n H_t(E_n)\} > 0$. That is, the first time when the energy histogram becomes nonzero everywhere. According to the adaptation rule of the learning rate η_t, the equilibration time is also the first time we decrease the learning rate. (ii) The scaling of the first equilibration time, in terms of the CPU time, with respect to the dimension L. Because the AWL algorithm requires additional computations in updating the momentum vector, the comparison between the two algorithms on the actual CPU time is necessary to see whether the implementation of the acceleration method is indeed worthwhile. (iii) The dynamics of the estimation error $\epsilon(t)$ defined as below following [30] for $L = 80$,

$$\epsilon(t) = \frac{1}{N-1} \sum_{n=1}^N \left| 1 - \frac{\log(g_t(E_n))}{\log(g(E_n))} \right|. \tag{23}$$

For the Ising model, the exact density of states $g(E_n)$ is available, and can be calculated using a publicly available Mathematica program [55]. For the Potts model, no exact solution of $g(E_n)$ is available, thus we pre-run a $1/t$ WL simulation for 5×10^7 MC sweeps, in which the final learning rate is 2×10^{-8}. We then treat the density estimates as an approximation to the exact density of states. (iv) The accuracy in the task of estimating the specific heat for the Ising model with $L = 80$.

We compare the AWL algorithm and the WL algorithm with different initializations of the learning rate, $\eta_0 = 0.05$, 0.10 and 1.00. We test out the two algorithms for different sizes of the two-dimensional square lattice, $L = 50, 60, 70, 80, 90, 100$. The computations in this paper were run on the FASRC Cannon cluster supported by the FAS Division of Science Research Computing Group at Harvard University.

Figure 1 summarizes the computational overheads of the two algorithms for the Ising model. The reported results are based on 50 independent runs of both algorithms, in which the dot represents the empirical mean and the error bar represents the empirical standard deviation. We see that the AWL algorithm takes significantly fewer MC sweeps as well as less CPU time to reach the first equilibration among all settings with different lattice sizes and different initializations of the learning rate. Figure 2 summarizes the computational overheads of the two algorithms on the Potts model. Similar to the case of Ising model, the AWL algorithm is more efficient than the WL algorithm in terms of the first equilibration time measured by the number of MC sweeps and the CPU time.

Figure 3 shows the empirical dynamics of $\epsilon(t)$, averaged over 50 independent runs of both algorithms. The first 100×10^3 MC sweeps for the Ising model and the first 1500×10^3 MC sweeps for the Potts model are representative for the transient phase. We see that in the transient phase, the convergence speed of the AWL algorithm, in terms of the number of MC sweeps, is significantly faster than the convergence speed of the WL algorithm with different initializations of the learning rate.

For the Ising model with $L = 80$, Table I compares the accuracy of the two algorithms in the calculation of the specific heat defined as:

$$C(T) = \frac{\langle E^2\rangle_T - \langle E\rangle_T^2}{T^2}, \tag{24}$$

in which T denotes the temperature. We test out temperatures ranging from 0.4 to 8 incremented by 0.1. The internal energy $\langle E\rangle_T$ is defined as

$$\langle E\rangle_T = \frac{\sum_n E_n g(E_n)\exp(-E_n/T)}{\sum_n g(E_n)\exp(-E_n/T)}. \tag{25}$$

The fluctuation expression $\langle E^2\rangle_T$ is defined similarly. We note that the theoretical value of the specific heat at a given temperature T can be evaluated exactly when the exact density of states is available, which is the case for the two-dimensional Ising model. We independently run each algorithm 50 times to obtain 50 independent estimates of the specific heat at each temperature. The relative error at each temperature is calculated based on the mean of the 50 independent estimates. Table I summarizes the quantiles of the relative errors for $T \in [0.4, 8]$, by running each algorithm for 100×10^3, 150×10^3, and 200×10^3 MC sweeps, respectively. Compared to the WL algorithm, the AWL algorithm yields significantly more accurate estimates of the specific heat especially in the transient phase.

More details of this numerical study can be found in the Supplemental Material. First, within the first 2×10^5 MC sweeps and 2×10^6 MC sweeps for the Ising model and the Potts model, respectively, we report the number of equilibrations that the AWL algorithm and the WL algorithm have reached (equivalently, the number of changes of the learning rate η_t), for different lattice sizes L and different initializations of the learning rate η_0 [56]. We also report the corresponding first 8 equilibration time in terms of the number of MC sweeps [57]. Second, for the Ising model with $L = 80$, we provide a graphical comparison of the estimated specific heat obtained by the AWL algorithm and the WL algorithm, over the temperature region $T \in [0.4, 8]$ [58].

V. CONCLUSION

To summarize, in this paper we present a new interpretation of the WL algorithm from the optimization perspective. We show that the WL algorithm is essentially a stochastic (projected) gradient descent algorithm minimizing a smooth and convex function, in which MCMC steps are used to estimate the unknown gradient. The optimization formulation intuitively explains that because of using more accurate gradient estimates, some notable modifications of the algorithm, such as utilizing multiple random walkers, can improve the WL algorithm. In addition, using the (strong) convexity of the objective function, we provide a new approach to establish the convergence rate of the WL algorithm, which is more explicit compared to the existing results [31, 40]. We expect that our contributions are useful for further theoretical investigations of the WL algorithm.

The optimization interpretation also opens a new way to improve the efficiency of the WL algorithm. There are rich tools in the optimization literature to accelerate the stochastic gradient descent algorithm, including but not restricted to the methods we mentioned in Section III. Different methods can be favorable for different applications. In the presence of noisy gradients, it usually requires some careful tuning to successfully apply the acceleration tools. We demonstrate one possible acceleration approach, using the momentum method and the adaptive learning rate strategy, on a two-dimensional Ising model and a two-dimensional ten-state Potts model.

Acknowledgments. This work was partially supported by NSF DMS-1613035 and DMS-1712714.

[1] F. Wang and D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical Review Letters*, 86(10):2050, 2001.

[2] F. Wang and D. P. Landau. Determining the density

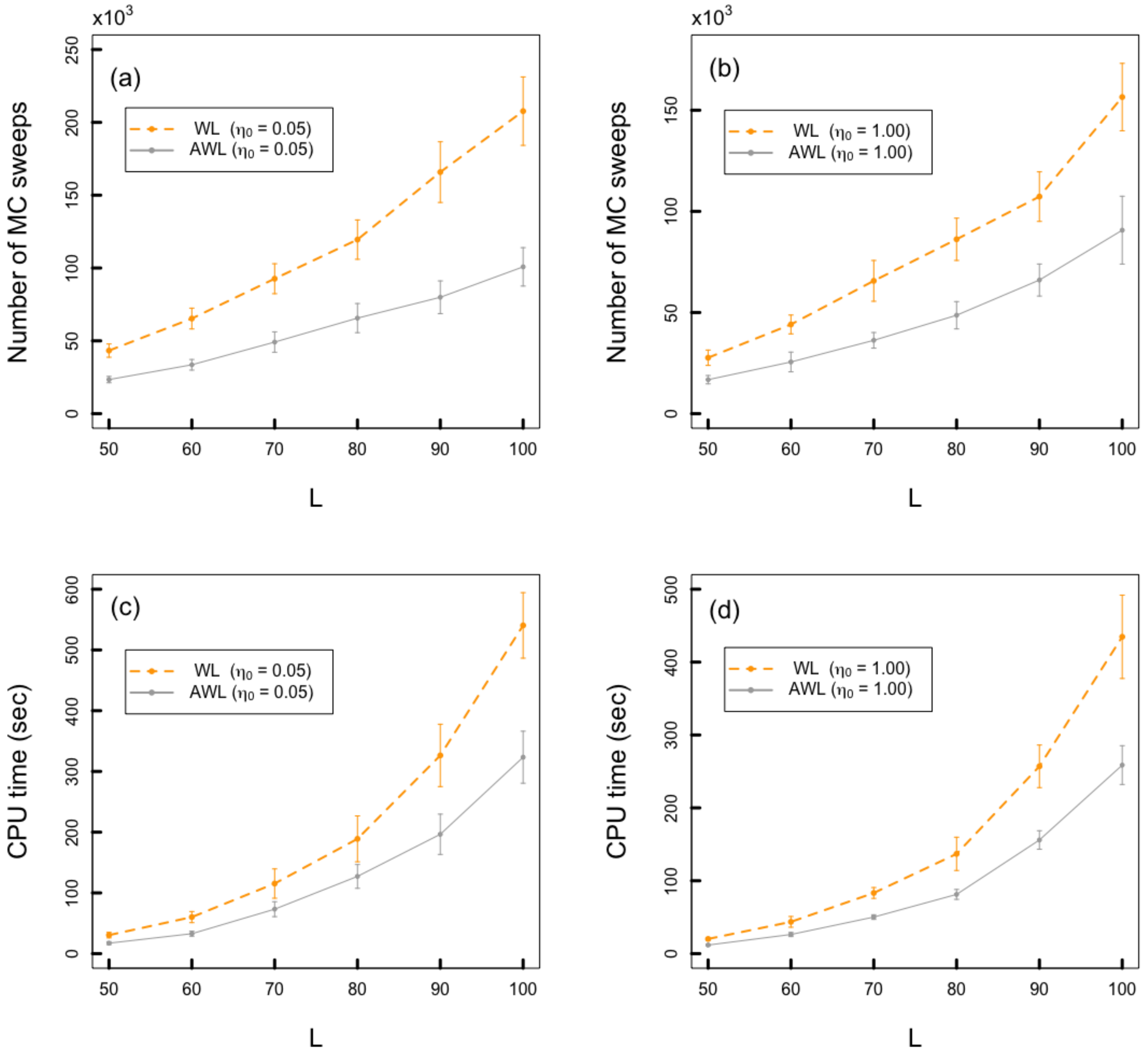

FIG. 1. The computational overheads, in terms of the number of MC sweeps and the CPU time, that the AWL algorithm and the WL algorithm takes to reach the first equilibration on the Ising model. Two initializations of the learning rate are tested out, including $\eta_0 = 0.05$ and $\eta_0 = 1.00$. The reported results are based on 50 independent runs of both algorithms. The dot represents the empirical mean and the error bar represents the empirical standard deviation.

	100×10^3 MC sweeps			150×10^3 MC sweeps			200×10^3 MC sweeps		
Quantiles	25%	50%	75%	25%	50%	75%	25%	50%	75%
AWL ($\eta_0 = 0.05$)	2.9%	6.3%	17.7%	0.9%	2.0%	4.6%	0.5%	1.2%	2.9%
WL ($\eta_0 = 0.05$)	10.5%	18.9%	41.4%	4.6%	9.1%	17.7%	1.1%	2.0%	4.4%
WL ($\eta_0 = 0.10$)	12.2%	24.0%	44.0%	2.4%	4.6%	10.9%	0.7%	2.4%	5.1%
WL ($\eta_0 = 1.00$)	47.1%	57.6%	74.4%	8.0%	16.0%	27.5%	2.8%	4.6%	8.4%

TABLE I. The relative errors of the AWL algorithm and the WL algorithm in the calculation of the specific heat for the Ising model with $L = 80$. The relative errors are calculated based on the mean of 50 independent estimates produced by each algorithm. The quantiles of the relative errors are over the temperature interval $T \in [0.4, 8]$. η_0 denotes the initialization of the learning rate.

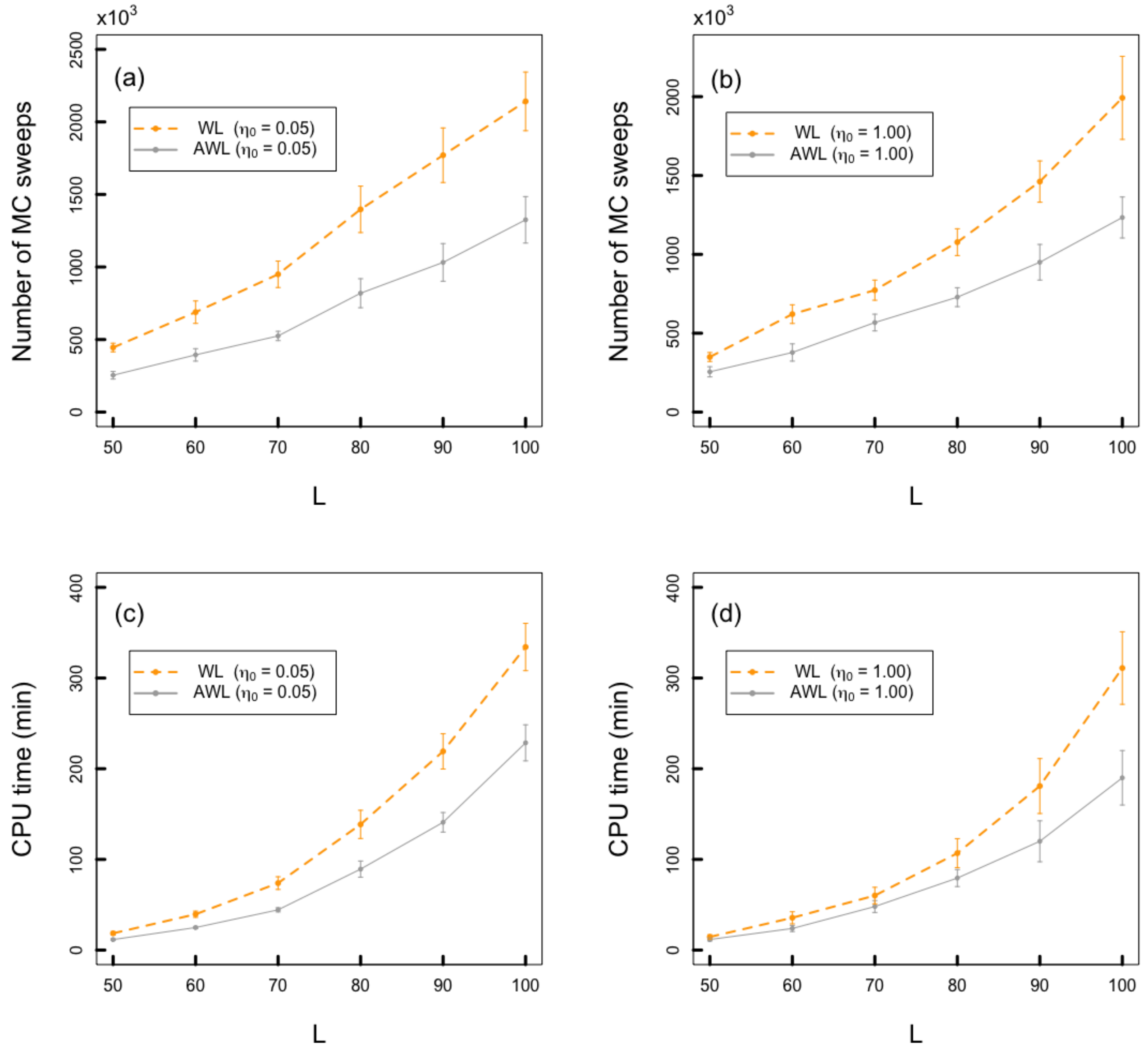

FIG. 2. The computational overheads, in terms of the number of MC sweeps and the CPU time, that the AWL algorithm and the WL algorithm takes to reach the first equilibration on the Potts model. Two initializations of the learning rate are tested out, including $\eta_0 = 0.05$ and $\eta_0 = 1.00$. The reported results are based on 50 independent runs of both algorithms. The dot represents the empirical mean and the error bar represents the empirical standard deviation.

of states for classical statistical models: A random walk algorithm to produce a flat histogram. *Physical Review E*, 64(5):056101, 2001.

[3] D. P. Landau, S. Tsai, and M. Exler. A new approach to Monte Carlo simulations in statistical physics: Wang-Landau sampling. *American Journal of Physics*, 72(10):1294–1302, 2004.

[4] G. Brown and T. C. Schulthess. Wang-Landau estimation of magnetic properties for the Heisenberg model. *Journal of Applied Physics*, 97(10):477, 2005.

[5] S. Torbrüegge and J. Schnack. Sampling the two-dimensional density of states g(E, M) of a giant magnetic molecule using the Wang-Landau method. *Physical Review B*, 75(5):054403, 2007.

[6] S. Alder, S. Trebst, A. K. Hartmann, and M. Troyer. Dynamics of the Wang-Landau algorithm and complexity of rare events for the three-dimensional bimodal Ising spin glass. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(07):P07008, 2004.

[7] J. Snider and C. Y. Clare. Absence of dipole glass transition for randomly dilute classical Ising dipoles. *Physical Review B*, 72(21):214203, 2005.

[8] Y. Okabe, Y. Tomita, and C. Yamaguchi. Application of new Monte Carlo algorithms to random spin systems. *Computer Physics Communications*, 146(1):63–68, 2002.

[9] C. Zhou, T. C. Schulthess, S. Torbrügge, and D. P. Landau. Wang-Landau algorithm for continuous models and joint density of states. *Physical Review Letters*,

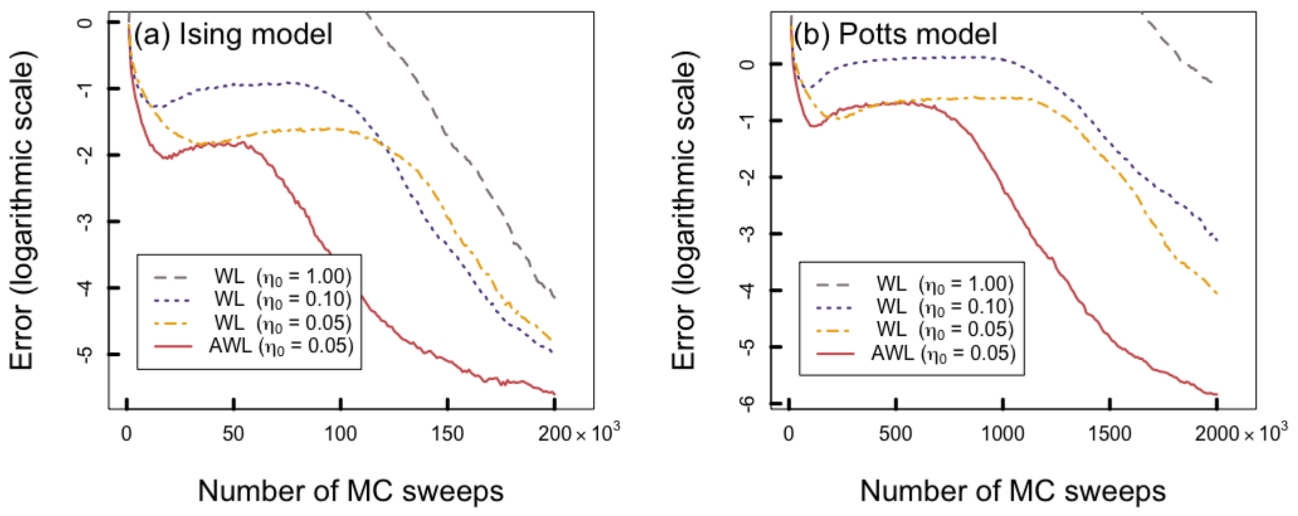

FIG. 3. The dynamics of the estimation error $\epsilon(t)$ (in the logarithmic scale), averaging over 50 independent runs, of the AWL algorithm and the WL algorithm. Panel (a) shows the result for the Ising model, and panel (b) shows the result for the Potts model. η_0 denotes the initialization of the learning rate.

96(12):120201, 2006.

[10] Y. Wu and J. Machta. Ground states and thermal states of the random field Ising model. *Physical Review Letters*, 95(13):137208, 2005.

[11] A. Malakis and N. G. Fytas. Lack of self-averaging of the specific heat in the three-dimensional random-field Ising model. *Physical Review E*, 73(1):016109, 2006.

[12] L. Hernández and H. Ceva. Wang-Landau study of the critical behavior of the bimodal 3D random field Ising model. *Physica A: Statistical Mechanics and its Applications*, 387(12):2793–2801, 2008.

[13] N. G. Fytas and A. Malakis. Phase diagram of the 3D bimodal random-field Ising model. *The European Physical Journal B*, 61(1):111–120, 2008.

[14] S. Tsai, F. Wang, and D. P. Landau. Critical endpoint behavior in an asymmetric Ising model: Application of Wang-Landau sampling to calculate the density of states. *Physical Review E*, 75(6):061108, 2007.

[15] C. Yamaguchi and Y. Okabe. Three-dimensional antiferromagnetic q-state Potts models: application of the Wang-Landau algorithm. *Journal of Physics A: Mathematical and General*, 34(42):8781, 2001.

[16] E. A. Mastny and J. J. de Pablo. Direct calculation of solid-liquid equilibria from density-of-states Monte Carlo simulations. *The Journal of Chemical Physics*, 122(12):124109, 2005.

[17] M. S. Shell, P. G. Debenedetti, and A. Z. Panagiotopoulos. Generalization of the Wang-Landau method for off-lattice simulations. *Physical Review E*, 66(5):056703, 2002.

[18] M. P. Taylor, W. Paul, and K. Binder. Phase transitions of a single polymer chain: A Wang-Landau simulation study. *The Journal of Chemical Physics*, 131(11):114907, 2009.

[19] J. L. Strathmann, F. Rampf, W. Paul, and K. Binder. Transitions of tethered polymer chains. *The Journal of Chemical Physics*, 128:064903, 2008.

[20] K. Langfeld, B. Lucini, and A. Rago. Density of states in gauge theories. *Physical Review Letters*, 109(11):111601, 2012.

[21] N. Rathore and J. J. de Pablo. Monte Carlo simulation of proteins through a random walk in energy space. *The Journal of Chemical Physics*, 116(16):7225–7230, 2002.

[22] N. Rathore, T. A. Knotts IV, and J. J. de Pablo. Density of states simulations of proteins. *The Journal of Chemical Physics*, 118(9):4285–4290, 2003.

[23] N. Rathore, Q. Yan, and J. J. de Pablo. Molecular simulation of the reversible mechanical unfolding of proteins. *The Journal of Chemical Physics*, 120(12):5781–5788, 2004.

[24] F. Calvo. Sampling along reaction coordinates with the Wang-Landau method. *Molecular Physics*, 100(21):3421–3427, 2002.

[25] A. Tröster and C. Dellago. Wang-Landau sampling with self-adaptive range. *Physical Review E*, 71(6):066705, 2005.

[26] Y. W. Li, T. Wüst, D. P. Landau, and H. Q. Lin. Numerical integration using Wang-Landau sampling. *Computer Physics Communications*, 177(6):524–529, 2007.

[27] F. Liang. A generalized Wang-Landau algorithm for Monte Carlo computation. *Journal of the American Statistical Association*, 100(472):1311–1327, 2005.

[28] Y. F. Atchadé and J. S. Liu. The Wang-Landau algorithm in general state spaces: applications and convergence analysis. *Statistica Sinica*, pages 209–233, 2010.

[29] L. Bornn, P. E. Jacob, P. Del Moral, and A. Doucet. An adaptive interacting Wang-Landau algorithm for automatic density exploration. *Journal of Computational and Graphical Statistics*, 22(3):749–773, 2013.

[30] R. E. Belardinelli and V. D. Pereyra. Fast algorithm to calculate density of states. *Physical Review E*, 75(4):046701, 2007.

[31] C. Zhou and R. N. Bhatt. Understanding and improving the Wang-Landau algorithm. *Physical Review E*,

72(2):025701, 2005.

[32] C. Zhou and J. Su. Optimal modification factor and convergence of the Wang-Landau algorithm. *Physical Review E*, 78(4):046705, 2008.

[33] P. Dayal, S. Trebst, S. Wessel, D. Wuertz, M. Troyer, S. Sabhapandit, and S. N. Coppersmith. Performance limitations of flat-histogram methods. *Physical Review Letters*, 92(9):097201, 2004.

[34] T. Vogel, Y. W. Li, T. Wüst, and D. P. Landau. Generic, hierarchical framework for massively parallel Wang-Landau sampling. *Physical Review Letters*, 110(21):210603, 2013.

[35] T. Wüst and D. P. Landau. Versatile approach to access the low temperature thermodynamics of lattice polymers and proteins. *Physical Review Letters*, 102(17):178101, 2009.

[36] C. Yamaguchi and N. Kawashima. Combination of improved multibondic method and the Wang-Landau method. *Physical Review E*, 65(5):056710, 2002.

[37] Y. Wu, M. Körner, L. Colonna-Romano, S. Trebst, H. Gould, J. Machta, and M. Troyer. Overcoming the slowing down of flat-histogram Monte Carlo simulations: Cluster updates and optimized broad-histogram ensembles. *Physical Review E*, 72(4):046704, 2005.

[38] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[39] J. S. Liu. *Monte Carlo strategies in scientific computing.* Springer Science & Business Media, 2008.

[40] G. Fort, B. Jourdain, E. Kuhn, T. Lelièvre, and G. Stoltz. Convergence of the Wang-Landau algorithm. *Mathematics of Computation*, 84(295):2297–2327, 2015.

[41] G. Fort, E. Moulines, and P. Priouret. Convergence of adaptive and interacting Markov chain Monte Carlo algorithms. *The Annals of Statistics*, 39(6):3262–3289, 2011.

[42] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[43] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[44] M. D. Zeiler. AdaDelta: an adaptive learning rate method. *arXiv:1212.5701*, 2012.

[45] C. Darken and J. Moody. Towards faster stochastic gradient search. In *Advances in Neural Information Processing Systems*, pages 1009–1016, 1992.

[46] D. Jayasri, V. S. S. Sastry, and K. P. N. Murthy. Wang-Landau Monte Carlo simulation of isotropic-nematic transition in liquid crystals. *Physical Review E*, 72(3):036702, 2005.

[47] P. Poulain, F. Calvo, R. Antoine, M. Broyer, and P. Dugourd. Performances of Wang-Landau algorithms for continuous systems. *Physical Review E*, 73(5):056704, 2006.

[48] See Supplemental Material at for the proof of Proposition 1.

[49] C. Andrieu, E. Moulines, and P. Priouret. Stability of stochastic approximation under verifiable conditions. *SIAM Journal on Control and Optimization*, 44(1):283–312, 2005.

[50] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.

[51] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013.

[52] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[53] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. Accelerating stochastic gradient descent. *stat*, 1050:26, 2017.

[54] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[55] P. D. Beale. Exact distribution of energies in the two-dimensional Ising model. *Physical Review Letters*, 76(1):78, 1996.

[56] See Supplemental Material at for the number of equilibrations that the AWL algorithm and the WL algorithm have reached.

[57] See Supplemental Material at for the first 8 equilibration time of the AWL algorithm and the WL algorithm.

[58] See Supplemental Material at for a graphical comparison of the estimated specific heat obtained by the AWL algorithm and the WL algorithm.