Computer Physics Communications 184 (2013) 1155–1160

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Scaling properties of a parallel implementation of the multicanonical algorithm

Johannes Zierenberg*, Martin Marenz, Wolfhard Janke

Institut für Theoretische Physik, Universität Leipzig, Postfach 100920, D-04009 Leipzig, Germany

ARTICLE INFO

Article history:
Received 18 July 2012
Received in revised form
6 December 2012
Accepted 7 December 2012
Available online 14 December 2012

Keywords:
Parallel
Multicanonical
Ising
Potts

ABSTRACT

The multicanonical method has been proven powerful for statistical investigations of lattice and off-lattice systems throughout the last two decades. We discuss an intuitive but very efficient parallel implementation of this algorithm and analyze its scaling properties for discrete energy systems, namely the Ising model and the 8-state Potts model. The parallelization relies on independent equilibrium simulations in each iteration with identical weights, merging their statistics in order to obtain estimates for the successive weights. With good care, this allows faster investigations of large systems, because it distributes the time-consuming weight-iteration procedure and allows parallel production runs. We show that the parallel implementation scales very well for the simple Ising model, while the performance of the 8-state Potts model, which exhibits a first-order phase transition, is limited due to emerging barriers and the resulting large integrated autocorrelation times. The quality of estimates in parallel production runs remains of the same order at the same statistical cost.

1. Introduction

Monte Carlo simulations are an important tool to investigate a wide range of theoretical models with respect to their statistical properties such as phase transitions, structure formation and more. Throughout the last two decades, umbrella sampling algorithms like the multicanonical [1,2] or the Wang–Landau [3] algorithm have been proven to be very powerful for investigations of statistical phenomena, especially first-order phase transitions, for lattice and off-lattice models. They have been applied to a variety of systems with rugged free-energy landscapes in physics, chemistry and structural biology [4].

Due to the fact that computer performance increases mainly in terms of parallel processing on multi-core architectures, a parallel implementation is of great interest, if the additional cores bring a benefit to the required simulation time. We present the scaling properties of a simple and straightforward parallelization of the multicanonical method, which has been reported in a similar way in [5] without much detail to the performance. This parallelization considers independent Markov chains, keeping communication to a minimum. Thus, it can be added on top of the multicanonical algorithm without much modification or system-dependent considerations and is also suitable for systems with simple energy calculations. Similar to this parallelization, there have been previous reports for the Wang–Landau algorithm [6,7], which needed a little more adaptation to the algorithm.

2. Multicanonical algorithm

The multicanonical method allows us to sample a system over a range of canonical ensembles at the same time. This is possible, because the statistical weights are modified in such a way that the simulation reaches each configuration energy of a chosen interval with equal probability, resulting in a flat energy histogram. To this end, the canonical partition function, in terms of the density of states $\Omega(E)$, is modified in the following way:

$$Z_{\text{can}} = \sum_{\{x_i\}} e^{-\beta E(\{x_i\})} = \sum_E \Omega(E) e^{-\beta E}$$

$$\rightarrow Z_{\text{MUCA}} = \sum_{\{x_i\}} W\left(E\left(\{x_i\}\right)\right) = \sum_E \Omega(E) W(E). \tag{1}$$

In order that each energy state occurs with the same probability, as requested above, the statistical weights have to equal the inverse density of states $W(E) = \Omega^{-1}(E)$. After an equilibrium simulation with those weights, it is possible to reweight to all canonical ensembles with a Boltzmann energy distribution covered by the flat histogram. This can be done for example by time-series reweighting, where in the average each measured observable is multiplied with its desired weight and divided by the weight with which it was measured:

$$\langle O \rangle_\beta = \frac{\langle O_i e^{-\beta E_i} W^{-1}(E_i) \rangle_{\text{MUCA}}}{\langle e^{-\beta E_i} W^{-1}(E_i) \rangle_{\text{MUCA}}}. \tag{2}$$

Of course, the density of states and consequently the weights that yield a flat energy histogram are usually not known in

* Corresponding author.
E-mail address: zierenberg@itp.uni-leipzig.de (J. Zierenberg).

0010-4655/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2012.12.006

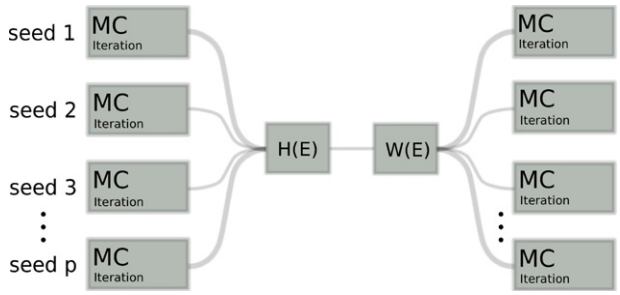

Fig. 1. Scheme of the parallel implementation of the multicanonical algorithm on p cores. After each iteration with independent Markov chains but identical weights, the histograms are merged, the new weights are estimated and the weights are distributed onto all processes again.

advance. Therefore the weights have to be obtained iteratively. In the most simple way consecutive weights are obtained from the last weights and the current energy histogram, $W^{(n+1)}(E) = W^{(n)}(E)/H^{(n)}(E)$. More sophisticated methods exist, where the full statistics of previous iterations is used for a stable and efficient approximation of the density of states [2]. All our simulations use this recursive version with logarithmic weights in order to avoid numerical problems.

Parallel version

The idea of this parallel implementation, similar to [5], is to distribute the time consuming generation of statistics on p independent processes. All processes perform equilibrium simulations with identical weights $W_i^{(n)} = W^{(n)}$, $i = 1, \ldots, p$, but with different random number seeds, resulting in similar but independent energy histograms $H_i^{(n)}(E)$. The histograms are merged after each iteration and one ends up with $H^{(n)}(E) = \sum_i H_i^{(n)}(E)$. According to the weight modification of choice, the collected histogram is processed together with the previous weights in order to estimate the consecutive weights $W^{(n+1)}$. The new weights are distributed onto all processes, which run equilibrium simulations again. That way, the computational effort may be distributed on several cores, allowing us to generate the same amount of statistics in a fraction of the time. It is important to notice that a modification of the program only influences the histogram merging and the distribution of the new weights, see Fig. 1. The iterations are independent copies run in parallel and the weight modification is performed on the master process as in the non-parallelized case.

3. Systems and implementation issues

We consider two discrete two-dimensional spin systems, namely the well known Ising model and the q-state Potts model with $q = 8$, where the Ising model can be mapped onto the $q = 2$ Potts model. The Ising model exhibits a second-order phase transition at $\beta_0 = \ln\left(1 + \sqrt{2}\right)/2$ and the 8-state Potts model exhibits a first-order phase transition at $\beta_0 = \ln\left(1 + \sqrt{8}\right)$. The spins are located on a square lattice with side length L and interact only between nearest neighbors. In the case of the Ising model, the interaction is described by the Hamiltonian

$$\mathcal{H}^{(\text{Ising})} = -J \sum_{\langle i,j \rangle} s_i s_j, \tag{3}$$

where J is the coupling constant and s_i,s_j can take the values $\{-1, 1\}$. For the q-state Potts model, where each site assumes values from $\{0, \ldots, q-1\}$, the nearest-neighbor interaction is described by

$$\mathcal{H}^{(\text{Potts})} = -J \sum_{\langle i,j \rangle} \delta(s_i, s_j), \tag{4}$$

where $\delta(s_i, s_j)$ is the Kronecker-Delta function which is only non-zero in the case $s_i = s_j$.

In both cases the number of discrete energy states is proportional to the number of lattice sites $V = L^2$, such that the width of the energy range increases quickly with system size. The simulations in this study start at infinite temperature, i.e., $\beta = 1/T = 0$ with quite narrow energy histograms. Because an estimation of successive weights is only possible for energies with non-zero histogram entries, the number of iterations may be very large for wide energy ranges. In order to ensure faster convergence, our implementation includes after each estimation of weights a correction function, which linearly interpolates the logarithmic weights at the boundaries of the sampled region (with a range of L bins), allowing the next iteration to sample a larger energy region. The MUCA weights are converged if the last iteration covered the full energy range and all histogram entries are within half and twice the average histogram entry. Between convergence of the weights and the final production run, the systems are thermalized again in order to yield correct estimates of the observables. In both cases, each sweep includes V numbers of spin updates.

4. Performance and scaling

In order to estimate the performance and the speedup of the parallel algorithm appropriately, we performed the analysis in two steps. First, we estimated the optimal number of sweeps per iteration and core, which we will refer to as M. To this end, we performed parallel MUCA simulations over a wide range of M for different lattice sizes L and number of cores p. The simulations were thermalized once in the beginning on every core, continuing the next iteration with the last state of the previous iteration on that core. This violates the equilibrium condition a little, as no intermediate thermalization phase was applied and part of the iteration was needed to reach equilibrium. This is accepted in order to compare the performance equally without an additional parameter to optimize next to M. Furthermore, the physical results were not influenced, because each Markov chain was thermalized before the final production run. We determined the mean number of iterations until convergence to a flat histogram $\bar{N}_{\text{iter}}$ as the average over 10 simulations at different initial seeds. Plotting the total number of sweeps $\bar{N}_{\text{iter}} M p$ versus M, we can estimate the optimal number of sweeps per iteration and core $\tilde{M}_{\text{opt}}$ as the minimum of this function (see Fig. 2(left)). For a small number of cores, this curve has a rather broad minimum, introducing a rough estimate. If, on the other hand, we stretch the curves along the x-axis with the number of cores, the outcomes look quite similar.

Selected results of the estimation of $\tilde{M}_{\text{opt}}$ are shown in Fig. 2(right). We see that for different lattice sizes and spin models the dependence on the number of cores may be described by a $1/p$ power law, where the amplitudes seem to depend on the system size and the number of states (notice that the Potts model curves nearly coincide with those of the Ising model with 4 times system size). In order to measure the performance equally, it is convenient to describe $\tilde{M}_{\text{opt}}$ by a function of system size L and number of cores p, $\tilde{M}_{\text{opt}}(L, p) \approx M_{\text{opt}}(L, p) = M_1(L)/p$, where M_1 is the interpolated optimal M for one core. Therefore, we estimated $\tilde{M}_{\text{opt}}$ for the square lattice sizes 8, 16, 24, 32, 48, 64, and 96 (the latter two only for Ising) with $p \leq 32$ and fitted for fixed size $M_{\text{opt}}(L, p) = M_1(L)/p$. The obtained $M_1(L)$ were plotted over L and fitted with a power law (see Fig. 3). In the end, the optimal number of sweeps per core and iteration were systematically described by the functions

$$\begin{aligned} M_{\text{opt}}^{(\text{Ising})}(L, p) &= 5.7(5) \times L^{2+0.51(4)} \frac{1}{p} \\ M_{\text{opt}}^{(8\text{Potts})}(L, p) &= 24(4) \times L^{2+0.67(6)} \frac{1}{p}, \end{aligned} \tag{5}$$