Computer Physics Communications 194 (2015) 54–58

Contents lists available at ScienceDirect

Computer Physics Communications

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

GPU-based cluster-labeling algorithm without the use of conventional iteration: Application to the Swendsen–Wang multi-cluster spin flip algorithm

Yukihiro Komura *

RIKEN, Advanced Institute for Computational Science, 7-1-26 Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo 650-0047, Japan
Department of Physics, Tokyo Metropolitan University, Hachioji, Tokyo 192-0397, Japan

ARTICLE INFO

Article history:
Received 2 December 2014
Received in revised form
23 April 2015
Accepted 23 April 2015
Available online 30 April 2015

Keywords:
Monte Carlo simulation
Cluster algorithm
Ising model
Parallel computing
GPU

ABSTRACT

Cluster-labeling algorithms that use a single GPU can be roughly divided into direct and two-stage approaches. To date, both types use an iterative method to compare the labels of nearest-neighbor sites. In this paper, I present a GPU-based cluster-labeling algorithm that does not use conventional iteration. The proposed method is applicable to both direct algorithms and two-stage approaches. Under the proposed approach, only one comparison with the nearest-neighbor site is needed for a two-dimensional (2D) system, and just two comparisons are needed for three-dimensional (3D) systems. As an application of the new cluster-labeling algorithm, I consider the Swendsen–Wang (SW) multi-cluster spin flip algorithm. The performance of the proposed method is compared with that of other cluster-labeling algorithms for the SW multi-cluster spin flip problem using the 2D and 3D Ising models. As a result, the computation time of the new algorithm is shown to be 40% faster than that of the previous algorithm for the 2D Ising model, and 20% faster than that of the previous algorithm for the 3D Ising model at the critical temperature.

1. Introduction

Recently, computational methods using accelerators such as GPUs have become more common in the field of computer science. Accelerators have evolved into highly parallel many-core processors with tremendous computational horsepower and very high memory bandwidth. Many researchers have shown that computing performance can be dramatically improved by the use of an accelerator. The Compute Unified Device Architecture (CUDA) is the most widely used programming model for accelerators [1]. CUDA is a parallel computing platform and programming model developed by NVIDIA, and is essentially C/C++ with a few extensions that allow functions to be executed on the NVIDIA GPU.

Monte Carlo simulations are often used to analyze the statistical mechanics of complex physical systems. The Metropolis algorithm [2] is widely used in Monte Carlo simulations. However, single spin flip methods such as the Metropolis algorithm often suffer from slow dynamics near the critical temperature of the phase transition. To overcome this problem, cluster spin flip algorithms have been proposed for Monte Carlo simulations. Typical cluster spin flip algorithms include the multi-cluster spin flip algorithm [3] and the single-cluster spin flip algorithm [4].

The Swendsen–Wang (SW) multi-cluster spin flip algorithm is realized using the cluster-labeling algorithm. The cluster-labeling algorithm assigns proper cluster labels to each site on the basis of local connection information. Cluster-labeling algorithms that use a single GPU can be roughly classified into two types: direct and two-stage approaches. In direct cluster-labeling algorithms, the whole lattice is directly calculated. In contrast, two-stage cluster-labeling algorithms divide the whole lattice into sublattices that are independently treated and then merged. Here, I describe several studies on SW multi-cluster spin flip algorithms with a single GPU. In a study of direct-type algorithms, the present author and Okabe [5] proposed a GPU method for the SW multi-cluster spin flip algorithm that used the idea of Hawick et al. [6], which is called the label equivalence method, and the improved idea proposed by Kalentev et al. [7]. They also extended the GPU-based SW multi-cluster spin flip algorithm to allow multiple GPUs to perform the computation [8]. More recently, they publicized sample CUDA programs for the SW multi-cluster spin flip algorithm [9]. As for two-stage methods, Weigel [10] proposed an SW multi-cluster

* Correspondence to: RIKEN, Advanced Institute for Computational Science, 7-1-26 Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo 650-0047, Japan. Tel.: +81 90 3597 0297.
E-mail address: yukihiro.komura@riken.jp.

http://dx.doi.org/10.1016/j.cpc.2015.04.015
0010-4655/

spin flip algorithm that combined a self-labeling algorithm [11] with a label relaxation algorithm or hierarchical sewing algorithm. Similar to the breadth-first search and tree-based union-and-find approach, the self-labeling algorithm is used to partition a set of elements into disjoint subsets. In a recent study of two-stage approaches, Wende et al. [12] proposed an SW multi-cluster spin flip algorithm using a combination of a local cluster search and global label reduction by means of atomic hardware primitives. They calculated each sublattice using a self-labeling method [11], and then applied a novel label reduction across the subclusters using atomic hardware primitives to resolve the label equivalences. Their numerical results showed that their code could achieve a performance gain of more than a factor of two over two previous methods [5,10].

Both the label equivalence method in the direct algorithms and the self-labeling method in the two-stage type are realized by an iterative method of comparison with the nearest-neighbor sites. In the label equivalence method, each site on the whole lattice compares the label of each site with the labels of the nearest-neighbor sites. The label of each site is then updated according to the chain of labels, which is itself updated in the comparison with the nearest-neighbor sites. The calculations are iterated until the labels at all sites no longer need to be compared with the nearest-neighbor sites. In the self-labeling method, each site within a sublattice compares the label of each site with the labels of the nearest neighbor-sites, and updates the label of each site. These calculations continue until the labels at all sites within a sublattice remain unchanged. In this paper, I propose a new cluster-labeling algorithm for a single GPU that does not use a conventional iterative technique. The proposed algorithm uses the label reduction method developed by Wende et al. [12]. The number of comparisons with the nearest-neighbor site in the proposed method is one for a two-dimensional (2D) system and two for a three-dimensional (3D) system. This new cluster-labeling algorithm uses atomic operations, which are performed without interference from any other threads. The proposed approach is not only an alternative to the self-labeling method in two-stage algorithms, but can also replace the label equivalence method in direct algorithms. The remainder of this paper is organized as follows. In Section 2, I briefly describe the SW multi-cluster spin flip algorithm for the Ising model, and explain the label reduction method proposed by Wende et al. [12]. In Section 3, I describe the new cluster-labeling algorithm on a single GPU. In Section 4, I compare the performance of the new cluster-labeling algorithm with that of other methods for the 2D and 3D Ising models. Finally, I present a summary and discussion in Section 5.

2. Swendsen–Wang cluster algorithm

The Hamiltonian of the Ising model is given by

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} s_i s_j, \quad s_i = \pm 1, \tag{1}$$

where J is the coupling and s_i is the spin at lattice site i. The summation is taken over the nearest-neighbor pairs $\langle i, j \rangle$. Periodic boundary conditions are employed.

In the SW multi-cluster spin flip algorithm [3], we separate the spins into clusters, and then flip the spins belonging to the same cluster. The SW multi-cluster spin flip algorithm for the Ising model consists of three main steps:

(1) *Active bond generation*: Construct a bond lattice of active or non-active bonds with probability $p = 1 - e^{2J/T}$, where T is the temperature.
(2) *Cluster labeling*: The active bonds partition the spins into clusters that are identified and labeled using a cluster-labeling algorithm.
(3) *Spin flip*: All spins in each cluster are set randomly to $+1$ or -1.

For cluster labeling, the most widely used technique in CPU implementations is the Hoshen–Kopelman algorithm [13], which assigns the proper cluster label to each site sequentially.

Because the active bond generation and spin flip calculations are performed independently at each site, these steps are well suited to GPU computation. However, the cluster labeling is conducted sequentially for each site; hence, the Hoshen–Kopelman cluster labeling algorithm cannot be directly applied to the GPU computation. Thus, to enable cluster labeling on a GPU, a new algorithm suitable for massive parallel computations is needed.

Here, I briefly explain the cluster-labeling algorithm proposed by Wende et al. [12]. Wende et al. developed a two-stage cluster-labeling algorithm for a single GPU. First, they partitioned the lattice into sublattices of equal size, and then independently identified subclusters within the sublattices. Second, they merged the subclusters together so that, finally, all of those belonging to the same cluster are assigned the same label. They calculated each sublattice using a self-labeling method [11]. In this self-labeling method, each site within a sublattice compares the label for each site with the labels of the nearest-neighbor sites, and updates the label of each site. The calculations continue until the labels at all sites within a sublattice remain unchanged. To merge the subclusters, they proposed a label reduction method. The pseudo-code for this label reduction method is given in Listing 4 of their paper [12]. Using the label reduction method, the subclusters merge at once. Some details of the label reduction method are given in the next section.

3. A new GPU computation for the cluster-labeling algorithm

In this section, I explain the new cluster-labeling algorithm for a single GPU. The algorithm is applicable to both the direct and two-stage approaches. The method of label reduction proposed by Wende et al. [12] is the cornerstone of the new cluster-labeling algorithm for a single GPU. The algorithm proposed in this paper consists of only four steps: (i) initialization; (ii) analysis; (iii) label reduction; (iv) analysis. The procedure of this algorithm is illustrated in Fig. 1. I now describe the detail of each function for a 2D square lattice.

(i) Initialization function

Usually, we assign unique labels of the lattice points to each site in the initial state, that is, *label*[i] $=$ *i*. In the proposed method, I assign the label from the minimum lattice point of connected sites to each site in the initial state. This resolves the problem of the propagation of label information in one direction to the next function. Each site has the label *label*[i] $=$ *min*, where *min* is the minimum site number of the connected sites.

(ii) Analysis function

The analysis function tracks the label from a given site to a new site that is determined by the number of the label at the given site. All sites calculate *label*[i] $=$ *label*[*label*[i]] until the *label* remains unchanged.

(iii) Label reduction function

The label reduction method in this paper is described in Algorithm 1. In Algorithm 1, *i* is the given site, *j* is the nearest-neighbor site of *i*, and *label* is the label at each site. To resolve conflicts in the label update process, the *atomicMin* function is used. Atomic operations provided by CUDA are performed without interference from any other threads. Moreover, I have

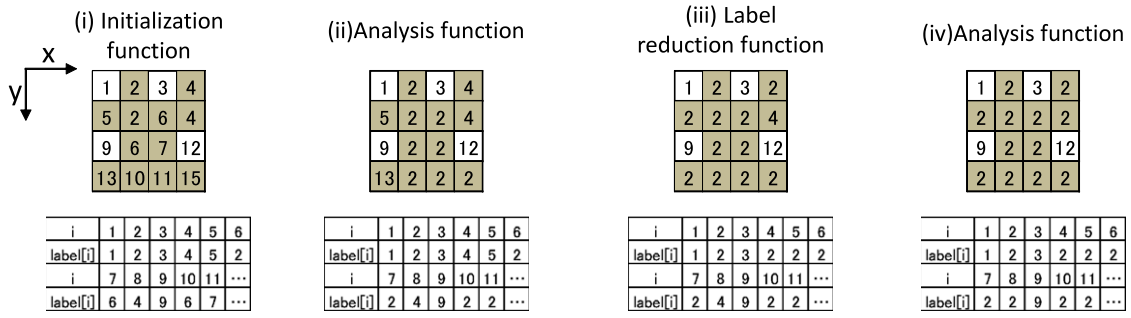

Fig. 1. Procedure of the cluster-labeling algorithm proposed in this paper. The connection of sites in the same cluster is represented by the green color. The thread number, which gives the site number, is denoted by i, and the variable for saving each label is represented by label. At step (i) of the initial function, each thread sets *label*[i] = *min*, where min is the minimum site number of the connected sites. At step (ii) of the analysis function, each thread calculates *label*[i] = *label*[*label*[i]] until the label remains unchanged. At step (iii) of the label reduction function, each thread uses the label reduction method given in Algorithm 1. In the 2D square lattice case, each thread calculates the label reduction method against the negative *x*-direction site. In this example, the threads for $i = 6, 8, 14$ have the state *flag* = *false* in Algorithm 1, and those threads execute the while loop in the label reduction method until the condition *flag* = *true* is satisfied. Step (iv) of the analysis function is the same as step (ii). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

```
label_1 ⇐ label[i] ;
while label_1 ≠ label[label_1] do
  | label_1 ⇐ label[label_1];
end

label_2 ⇐ label[j] ;
while label_2 ≠ label[label_2] do
  | label_2 ⇐ label[label_2];
end

flag ⇐ true;
if the bond between i and j is active then
  | if label_1 != label_2 then flag ⇐ false;
end

if label_1 < label_2 then
  | tmp ⇐ label_1;
  | label_1 ⇐ label_2;
  | label_2 ⇐ tmp;
end

while flag = false do
  | label_3 ⇐ atomicMin(&label[label_1], label_2);
  | if label_3 = label_2 then flag ⇐ true;
  | else if label_3 > label_2 then label_1 ⇐ label_3;
  | else if label_3 < label_2 then
  | label_1 ⇐ label_2; label_2 ⇐ label_3;
end
```

Algorithm 1: Pseudo-code of the label reduction method used in this paper. Here, *i* is a site, *j* is the nearest-neighbor site to *i*, and *label* is the label at each site. The *atomicMin* function proceeds without interference from any other threads. *atomicMin* ensures that *label*_3 ⇐ *label*[*label*_1] and then *label*[*label*_1] is updated to the minimum of either *label*[*label*_1] or *label*_2.

added a procedure for the exchange of *label*_1 and *label*_2 to the original label reduction method [12]. Wende et al. [12] used label reduction as a means of merging subclusters, that is, the pair *i* and *j* are confined to the boundary sites between each sublattice. In this paper, I apply label reduction at all sites. However, the computational cost of applying label reduction in all directions is high because of the atomic operation. In the initialization function and the analysis function, the propagation of label information is resolved in one direction for each site. However, if the site has two active bonds connected in the negative *y*-direction and the negative *x*-direction, label information will not be propagated to those sites. In this case, the label information for the negative *y*-direction is propagated on a priority basis by the above two functions, and label information will not be propagated in the negative *x*-direction. Thus, I use the label reduction method in only one direction (negative *x*-direction) for the 2D square lattice and only two directions (negative *x*-direction and negative *y*-direction) for the 3D simple cube lattice.

(iv) Analysis function

This function is the same as in step (ii).

This new cluster-labeling algorithm is not only an alternative for the self-labeling method in two-stage approaches, but can also be used as a direct-type method. If periodic boundary conditions are employed, their effect is eliminated in the initialization function, and calculations for the negative *y*-direction in the top sites of the 2D lattice and negative *z*-direction in the top layer sites of the 3D lattice are added to the label reduction function.

The proposed cluster-labeling algorithm does not require a conventional iterative approach because of the label reduction method. In the label reduction method, sites with different labels, e.g., *label*_1 and *label*_2 in Algorithm 1, adjust the chain of the labels, i.e., *label*[*label*]. The atomic function in Algorithm 1 updates *label*[*label*_1] if the value of *label*[*label*_1] is bigger than that of *label*_2, and then updates *label*_3 to the previous value of *label*[*label*_1]. The values of *label*_3 and *label*_2 are used in the next while loop iteration, and the while loop continues until the condition *label*_3 = *label*_2 is satisfied. The atomic function is used to prevent the label updates from overlapping. Thus, each chain of labels, i.e., *label*[*label*], is automatically constructed in the label reduction method. In the next analysis function, all sites update their labels by calculating *label*[i] = *label*[*label*[i]] until *label* remains unchanged. Therefore, the new cluster labeling algorithm has an iterative method within the label reduction method. However, in the proposed algorithm, there is only one comparison with the nearest-neighbor sites for the 2D system, and only two comparisons for the 3D system. Additionally, the number of sites at which the while loop is executed in the label reduction method is kept to a minimum.

4. Results

I have tested the performance of the proposed code on an NVIDIA Tesla K20X machine with the CUDA version 6.0 compiler. I compared the performance of the SW multi-cluster spin flip algorithm using four cluster-labeling algorithms: (1) the label equivalence method proposed by Kalentev et al. [7] as a direct approach; (2) the proposed method as a direct-type algorithm; (3) the self-labeling method and label reduction method proposed by Wende

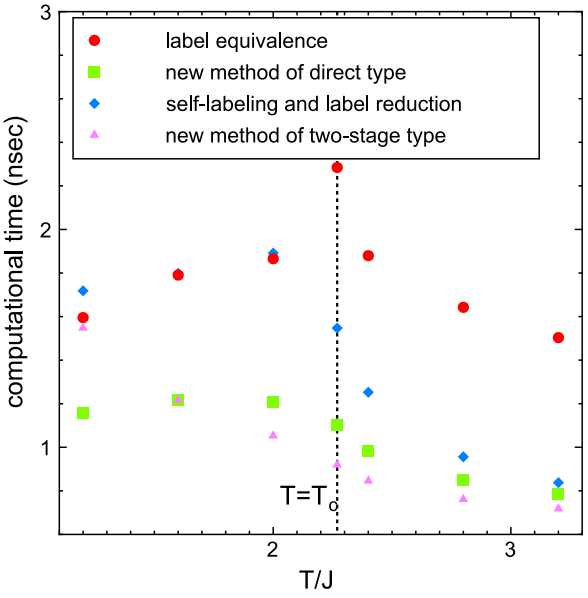

Fig. 2. Temperature dependence of the computation time (ns) for the 2D Ising model with $L = 1024$.

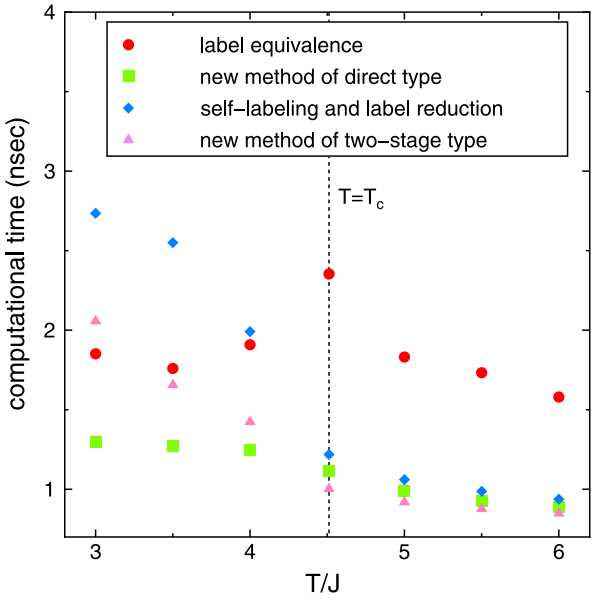

Fig. 3. Temperature dependence of the computation time (ns) for the 3D Ising model with $L = 128$.

et al. [12] as a two-stage approach; and (4) the proposed method in two-stage form, that is, the new method and a label reduction method. For the new direct-type form, I employed the present method at all sites. For the new two-stage form, I used the present method at all sites on each sublattice, and then applied the label reduction method to merge the subclusters. All algorithms used the label equivalence framework described in Ref. [9], except for the cluster-labeling algorithm. To realize a fair comparison, each code was optimized. In the self-labeling method of the two-stage approaches, the sublattices were of size 16×32 in the 2D case, and $4 \times 4 \times 32$ in 3D. The number of sites per sublattice was twice the number of threads per block, as the proposed implementation uses a checkerboard decomposition of the sublattices with one thread per even site. The register usage per thread was restricted to 32 in the self-labeling method. The new two-stage type method used the same sublattice sizes as the self-labeling method. However, the number of sites per sublattice was equal to the number of threads per block. To realize fast calculations, the shared memory on the GPU was used in the self-labeling method and the new two-stage method. To merge the subclusters in the two-stage approach, the label reduction method was applied to all boundary sites on each sublattice. In the two direct methods, the total number of threads was set to the number of sites. Moreover, the direct methods did not use the shared memory on the GPU, because there are no shared data in any blocks. I used the linear congruential random generator [14] to produce random numbers. The code was checked by comparing the results in Ref. [9] with the results produced by the other algorithms. A comparison between the GPU calculation with the label equivalence method and the CPU calculation with the Hoshen–Kopelman algorithm [13] is given in Ref. [5].

I first examine the computation time of the SW multi-cluster spin flip algorithm with each cluster-labeling algorithm for the 2D Ising model at the critical temperature, $T_c/J = 2/\ln(1+\sqrt{2}) = 2.2691$ [15]. Table 1 lists the time required for a spin update calculation. The linear system sizes are $L = 512$, 1024, 2048, and 4096. From Table 1, it can be seen that the new method gives faster computation times for both the direct and two-stage approaches. The computation time of the new direct method is about half that of the label equivalence method for all system sizes. The computation time of the new two-stage approach is 40% faster than using the self-labeling method with label reduction. Additionally, the computation time of the new two-stage method is about 10% faster than that of the new direct method.

Second, I refer to the temperature dependence of the SW multi-cluster spin flip algorithm with the four cluster-labeling algorithms for the 2D Ising model. The temperature dependence of the computation times of each cluster-labeling algorithm for the 2D Ising model with $L = 1024$ is plotted in Fig. 2. From Fig. 2, it can be seen that the behavior of the label equivalence method is similar to that reported in Ref. [5]. The increment in computation time for the self-labeling method with label reduction near the critical temperature is due to the self-labeling method. The new two-stage method is the fastest of the four cluster-labeling algorithms, except at very low temperatures. The new direct method is less affected by the temperature than the other methods.

Third, I examined the computation times of the SW multi-cluster spin flip algorithm with four cluster-labeling algorithms for the 3D Ising model at the critical temperature, $T_c/J = 4.5115$ [16,17]. Table 2 lists the average time required for one spin update calculation. The linear system sizes are $L = 96$, 128, 192, and 256. Table 2 indicates that the computation time of the new direct method is about half that of the label equivalence method for all system sizes, as for the 2D Ising model. Unlike the 2D case, the computation time of the new two-stage method is about only 20% faster than that of the self-labeling and label reduction method. For the same total system size, the computation times of the two new methods are greater than for the 2D Ising model. The increments are due to the increase in the number of search directions in the label reduction function. In contrast, the computation time of the self-labeling and label reduction method is reduced. This decrement is caused by the decrease in the number of iterations in the self-labeling method. The computation time of the new two-stage approach is about 10% faster than that of the new direct method.

Finally, I consider the temperature dependence of the SW multi-cluster spin flip algorithm with each cluster-labeling algorithm for the 3D Ising model. The temperature dependence of the computation time of the four cluster-labeling algorithms for the 3D Ising model with $L = 128$ is plotted in Fig. 3. From Fig. 3, it can be seen that the computation times at low temperatures are greater than those at high temperatures for all methods except label equivalence. The increment in computation times at low temperatures is caused by the atomic function. In the label equivalence method,

Table 1
Average computation time (ns) per spin flip for the 2D Ising model at $T/J = 2.2691$. The time refers to a single spin update.

	$L = 512$	$L = 1024$	$L = 2048$	$L = 4096$
Label equivalence	2.99 ns	2.28 ns	2.09 ns	2.07 ns
New method of the direct type	1.29 ns	1.10 ns	1.00 ns	0.93 ns
Self-labeling method and label reduction method	1.79 ns	1.54 ns	1.47 ns	1.44 ns
New method of two-stage type	1.06 ns	0.92 ns	0.87 ns	0.85 ns

Table 2
Average computation time (ns) per spin flip for the 3D Ising model at $T/J = 4.5115$. The time refers to a single spin update.

	$L = 96$	$L = 128$	$L = 192$	$L = 256$
Label equivalence	2.91 ns	2.35 ns	2.82 ns	2.37 ns
New method of the direct type	1.16 ns	1.11 ns	1.09 ns	1.10 ns
Self-labeling method and label reduction method	1.26 ns	1.21 ns	1.18 ns	1.18 ns
New method of two-stage type	1.04 ns	1.00 ns	0.98 ns	0.98 ns

the computation time near the critical temperature increases because of the increment in the number of iterations. As for the 2D Ising model, the new two-stage method is the fastest of the four cluster-labeling algorithms, except at low temperatures. The new direct method is less affected by the temperature than the other methods.

5. Summary and discussion

I have proposed a new cluster-labeling algorithm for a single GPU that uses the label reduction method proposed by Wende et al. [12]. To realize this new cluster-labeling algorithm, I use atomic operations, which are performed without interference from any other threads. Until now, both the direct and two-stage types of cluster-labeling algorithm have required an iterative method to compare the labels of nearest-neighbor sites. In this paper, I have proposed a cluster-labeling algorithm for a single GPU that does not employ conventional iteration and is applicable to both the direct approach and one stage of the two-stage method. The proposed method requires only one comparison with the nearest-neighbor site for a 2D system, or two comparisons for a 3D system. I examined the proposed cluster-labeling algorithm by implementing it as part of the SW multi-cluster spin flip algorithm. This allowed the performance of the new cluster-labeling algorithm to be evaluated.

The computation time of the SW multi-cluster spin flip algorithm was recorded with four different cluster-labeling algorithms for the 2D and 3D Ising models. It was found that the new two-stage method proposed in this paper was the fastest of the four cluster-labeling algorithms for both the 2D and 3D Ising models at the critical temperature. This suggests that the two-stage method, in which label information in each sublattice is adjusted before that between different sublattices is adjusted, has an advantage over direct-type approaches, in which label information is directly modified across the whole lattice. However, there is no significant difference between the computational performance of the new method in two-stage or direct-type approaches. The temperature dependence of the SW multi-cluster spin flip algorithm was also examined with four different cluster-labeling algorithms. This experiment confirmed that the new method is faster than the previous approaches at all temperatures. Additionally, the new direct method was found to be less affected by changes in temperature than the other methods.

More recently, OpenACC has been proposed as a standard programming model for accelerators [18]. OpenACC allows programmers to use simple compiler directives to identify which areas of code to accelerate, without requiring any modification of the underlying CPU code. However, the use of shared memory can cause difficulties in OpenACC. The high-speed calculations for the two-stage SW multi-cluster spin flip algorithm use the shared memory on the GPU. Thus, the performance of the new two-stage method with OpenACC is likely to be worse than that with CUDA. However, the new direct method proposed in this paper has the potential to be used as a more general cluster-labeling algorithm with OpenACC. I hope that more researchers will take an interest in the GPU calculations of clustering algorithms.

Acknowledgments

I would like to thank Yutaka Okabe for valuable discussions. The numerical calculations were carried out on the TSUBAME2.5 supercomputer at the Tokyo Institute of Technology. This work was supported by JSPS KAKENHI Grant Nos. 15K21623, 25400406.

References

[1] NVIDIA CUDA ZONE https://developer.nvidia.com/cuda-zone.
[2] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (1953) 1087–1092.
[3] R.H. Swendsen, J.S. Wang, Nonuniversal critical dynamics in Monte Carlo simulations, Phys. Rev. Lett. 58 (1987) 86–88.
[4] U. Wolff, Collective Monte Carlo updating for spin systems, Phys. Rev. Lett. 62 (1989) 361–364.
[5] Y. Komura, Y. Okabe, GPU-based Swendsen–Wang multi-cluster algorithm for the simulation of two-dimensional classical spin systems, Comput. Phys. Comm. 183 (2012) 1155–1161.
[6] K.A. Hawick, A. Leist, D.P. Playne, Parallel graph component labelling with GPUs and CUDA, Parallel Comput. 36 (2010) 655–678.
[7] O. Kalentev, A. Rai, S. Kemnitzb, R. Schneider, Connected component labeling on a 2D grid using CUDA, J. Parallel Distrib. Comput. 71 (2011) 615–620.
[8] Y. Komura, Y. Okabe, Multi-GPU-based Swendsen–Wang multi-cluster algorithm for the simulation of two-dimensional q-state Potts model, Comput. Phys. Comm. 184 (2013) 40–44.
[9] Y. Komura, Y. Okabe, GPU-based Swendsen–Wang multi-cluster algorithm for the simulation of two-dimensional classical spin systems, Comput. Phys. Comm. 185 (2014) 1038–1043.
[10] M. Weigel, Connected component identification and cluster update on GPU, Phys. Rev. E 84 (2011) 036709.
[11] C.F. Baillie, P.D. Coddington, Cluster identification algorithms for spin models—sequential and paralle, Concurrency: Pract. Exper. 3 (1991) 129–144.
[12] F. Wende, T. Steinke, Swendsen–Wang multi-cluster algorithm for the 2D/3D Ising model on Xeon Phi and GPU, in: Proceeding SC'13 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis Article No. 83, 2013.
[13] J. Hoshen, R. Kopelman, Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm, Phys. Rev. B 14 (1976) 3438–3445.
[14] T. Preis, P. Virnau, W. Paul, J.J. Schneider, GPU accelerated Monte Carlo simulation of the 2D and 3D Ising model, J. Comput. Phys. 228 (2009) 4468–4477.
[15] L. Onsager, Crystal statistics. I. A two-dimensional model with an order–disorder transition, Phys. Rev. 65 (1944) 117–149.
[16] A.M. Ferrenberg, D.P. Landau, Critical behavior of the three-dimensional Ising model: A high-resolution Monte Carlo study, Phys. Rev. B 44 (1991) 5081–5091.
[17] H.W.J. Blöte, E. Luijten, J.R. Heringa, Ising universality in three dimensions: a Monte Carlo study, J. Phys. A: Math. Gen. 28 (1995) 6289–6314.
[18] OpenACC http://www.openacc-standard.org/.