Dual geometric worm algorithm for two-dimensional discrete classical lattice models

Peter Hitchcock* and Erik S. Sørensen†
Department of Physics and Astronomy, McMaster University, Hamilton, Ontario, Canada L8S 4M1

Fabien Alet‡
Computational Laboratory, ETH Zürich, CH-8092 Zürich, Switzerland
and Theoretische Physik, ETH Zürich, CH-8093 Zürich, Switzerland

(Received 1 November 2003; published 2 July 2004)

We present a dual geometrical worm algorithm for two-dimensional Ising models. The existence of such dual algorithms was first pointed out by Prokof'ev and Svistunov [N. Prokof'ev and B. Svistunov, Phys. Rev. Lett. **87**, 160601 (2001)]. The algorithm is defined on the dual lattice and is formulated in terms of bond variables and can therefore be generalized to other two-dimensional models that can be formulated in terms of bond variables. We also discuss two related algorithms formulated on the direct lattice, applicable in any dimension. These latter algorithms turn out to be less efficient but of considerable intrinsic interest. We show how such algorithms quite generally can be "directed" by minimizing the probability for the worms to erase themselves. Explicit proofs of detailed balance are given for all the algorithms. In terms of computational efficiency the dual geometrical worm algorithm is comparable to well known cluster algorithms such as the Swendsen-Wang and Wolff algorithms, however, it is quite different in structure and allows for a very simple and efficient implementation. The dual algorithm also allows for a very elegant way of calculating the domain wall free energy.

DOI: 10.1103/PhysRevE.70.016702 PACS number(s): 05.10.Ln, 05.20.-y, 02.70.Tt

I. INTRODUCTION

Over recent decades many powerful Monte Carlo (MC) algorithms have been developed, greatly enhancing the scope and applicability of Monte Carlo techniques. In fact, it is quite likely that these algorithmic advances have, and will continue to have, a far greater impact on the predictive power of Monte Carlo simulations than advances in raw computational capacity, a point that is often overlooked. The continued development of such advanced algorithms is therefore very important. Here we shall mainly be concerned with MC algorithms suitable for the study of lattice models described by classical statistical mechanics. Some of the most notable developments in this field have been the development of cluster algorithms by Swendsen and Wang [1] and by Wolff [2]. More recent developments include invaded cluster algorithms [3] that self-adjust to the critical temperature, flat histogram methods [4], focusing on the density of states and techniques performing Monte Carlo sampling of the high temperature series expansion of the partition function [5] using worm algorithms [6]. Two of us recently proposed a very efficient geometrical worm algorithm [7,8] for the bosonic Hubbard model. In this algorithm variables are not updated at random but instead a "worm" is propagated through the lattice, at each step choosing a new site to visit with a probability proportional to the *relative* (among the different sites considered) probability of changing the corresponding variable. The high efficiency of the algorithm stems from the fact that the worm is *always* moved and it is only through the actual movement that the local environment—the local "geometry"—comes into play. The ideas underlying this algorithm are quite generally applicable and the present paper is concerned with their generalization to classical statistical mechanics models.

As the canonical testing ground for algorithms we consider the standard ferromagnetic Ising model in two dimensions defined by

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

Here $\langle i,j\rangle$ denote the summation over nearest neighbor spins. It is well known that the critical temperature for this model in two dimensions is $k_B T_c = 2J/\log(1+\sqrt{2}) = J2.26918\ldots$. When investigating magnetic materials modeled by classical statistical mechanics such as Eq. (1) using Monte Carlo methods one has to take into account the effects of the nonzero autocorrelation time τ (defined later) that is always present in Monte Carlo simulations. The autocorrelation time describes the correlation between observations of an observable $\mathcal{O}(t_0)$ and $\mathcal{O}(t_0+t)$, t Monte Carlo sweeps (MCS) apart. The autocorrelation time, τ, depends on the simulation temperature and the system size and grows dramatically close to the critical temperature, T_c, a phenomenon referred to as critical slowing down. At T_c the autocorrelation time displays a power-law dependence of the system size, $\tau \sim L^{z_{\mathrm{MC}}}$, defining a Monte Carlo dynamical exponent z_{MC}. For the well known Metropolis algorithm one estimates [9,10] $z_{\mathrm{MC}} \sim 2.1-2.2$. If efficient algorithms, with a very small z_{MC}, cannot be found, this scaling renders the Monte

*Electronic address: hitchpa@muss.cis.mcmaster.ca
†Electronic address: sorensen@mcmaster.ca
URL: http://ingwin.physics.mcmaster.ca/~sorensen
‡Electronic address: alet@phys.ethz.ch

Carlo method essentially useless for large lattice sizes since all data will be correlated. It is therefore crucial to develop algorithms with a very small or zero z_{MC}. In order to test the proposed algorithms we shall therefore only consider simulations at the critical point since this is where the critical slowing down is the most pronounced and where z_{MC} is defined.

The geometrical worm algorithm [7,8] has proven to be very efficient for the study of the bosonic Hubbard model when formulated in terms of currents on the links of the space-time lattice. The estimated z_{MC} is very close to zero [8]. It is therefore natural to generalize this algorithm to classical models defined in terms of bond variables. This is done in Sec. III, where a very efficient algorithm formulated directly on the *dual* lattice is developed. A related dual algorithm was initially described by Prokof'ev and Svistunov [5]. Our algorithm can be generalized to Potts, clock, and other discrete lattice models of which the Ising model is the simplest example. The dual algorithm also allows for a very simple and elegant way of calculating the domain wall free energy directly. In Sec. III we outline how this is done. Regretably, only in two dimensions is it possible to define such a dual algorithm. In Sec. III we also present a directed version of this algorithm where the probability for the worms to erase themselves is minimized. However, before presenting the dual algorithm it is instructive to consider geometrical worm algorithms defined on the *direct* lattice. Such an algorithm is described in Sec. II, in both directed and undirected versions. This algorithm is applicable in any dimension but is of less interest due to its poor efficiency. However, from an algorithmic perspective this algorithm is interesting in its own right and it leads naturally to the definition of the dual algorithm. Finally, in Sec. IV we conclude with a number of observations concerning the properties of the algorithms.

II. GEOMETRICAL WORM ALGORITHMS ON THE DIRECT LATTICE

The first algorithm we present we shall refer to as the linear worm algorithm. This algorithm is closely related to the geometrical worm algorithms [7,8], however, the worms do not form closed loops, instead they form linear strings (a worm) of flipped spins. A major advantage of this algorithm is that we can select the length of the linear worm or even the entire distribution of worm lengths. This could be advantageous for the study of frustrated or disordered models where other cluster algorithms fail due to the fact that too "big" or too "small" clusters are being generated, leading to a significant loss of efficiency.

A. Algorithm A (linear worms)

We begin with a number of useful definitions: In order to define a working algorithm we consider two configurations of the spins, μ and ν related by the introduction of a worm. Let μ denote the configurations of the spins without the worm, w, and ν the configuration of the spins with the worm. Furthermore, let $s_1 \ldots s_W$ be the sites (or spins) visited by the worm (of length W) and let E_i denote the energy required

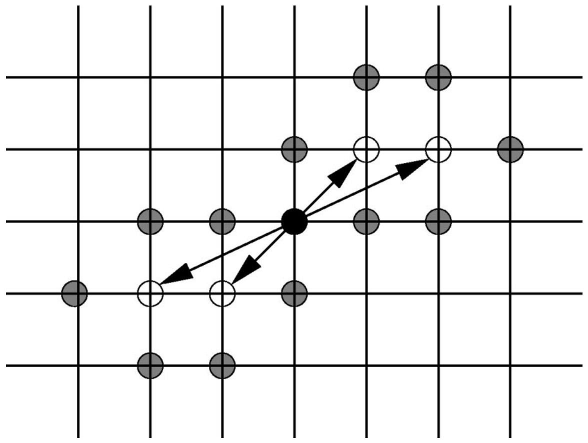
FIG. 1. An example of the configuration of the neighbors for algorithm A,B. The current site, s_i, is indicated by •, the neighbors by ∘. The activation (weight) at the current site will depend on the spins at the nearest neighbor sites (shaded circles).

to flip the spin at site s_i from its position in the configuration μ, with $-E_i$ the energy required to flip spin i from its position in ν to that in μ. Note that E_i is defined relative to the spin configuration μ. In the following we shall make extensive use of the activation or local weight for overturning a spin, $A_{s_i}^{E_i}$. More precisely, $A_{s_i}^{E_i}$ denotes the weight for flipping s_i from its position in μ to that in ν and $A_{s_i}^{-E_i}$ the weight for going in the opposite direction. As we shall see, $A_{s_i}^{E_i}$ is not uniquely determined. Here, we shall use $A_{s_i}^{E_i}=\min[1,\exp(-\Delta E_i/k_B T)]$ although other choices would be equally suitable. When the worm is moving through the lattice it will move from the current site s_i to a set of neighboring sites σ and it becomes necessary to define the normalization $N_{s_i}=\Sigma_\sigma A_{s_i+\sigma}^{E_{i+\sigma}}$. This normalization is used for choosing the next neighbor to visit among the set σ of neighbors.

If one considers the proofs for the geometrical worm algorithms [7,8], it is not difficult to see that a generalization to classical statistical models defined on the direct lattice will depend crucially on the fact that the normalization N_{s_i} does not depend on the position (up or down) of the spin, s_i, at the site itself. For the Ising model, defined in Eq. (1), only nearest neighbor interactions are taken into account and we can satisfy this requirement simply by not allowing the worm to move to any of the four nearest neighbor sites to the current site s_i. In principle, one can consider moving the worm to *any* other sites in the lattice, an aspect of this algorithm of considerable intest. For the case of longer range interactions we would have to restrict the worm to move only to sites that the current spin does *not* interact with. Note that, since we allow sites on the same sublattice to be visited then the $A_{s_i}^{E_i}$ will depend on the *order* that we visit the sites since sites neighboring s_i could have been visited previously by the worm. As an example of a simple choice for the neighbors we show in Fig. 1 an example where the worm can move to four neighbors around the current site, excluding the four nearest neighbors to the site. When defining the set of neighbors σ for the worm to move to from s_i one also has to satisfy the trivial property that if s_{i+1} is a neighbor of s_i then s_i should also be a neighbor of s_{i+1}.

We now propose one possible implementation of the geometrical worm algorithm for the Ising model. A set of suitable neighbors to s_i is defined as outlined earlier and at the beginning of the construction of the worm we *draw* its length, W from a normalized distribution. We refer to this algorithm as linear worms (A). Later we outline the algorithm in pseudocode.

(1) Choose a random starting site, $s_{i=1}$ and with normalized probability a length W.

(2) Calculate the probability for flipping the spin on that site $A_{s_1}^{E_1}$ with $A_{s_i}^{E_i}=\min[1,\exp(-\Delta E_{s_i}/k_BT)]$.

(3) Flip the selected spin, s_i.

(4) For each of the k neighbors in the set of neighbors σ, calculate the weight for flipping, $A_{s_i+\sigma}^{E_{i+\sigma}}=\min[1,\exp(-\Delta E_{s_i+\sigma}/k_BT)]$. Calculate the normalization $N_{s_i}=\Sigma_\sigma A_{s_i+\sigma}^{E_{i+\sigma}}$ and the probabilities $p_{s_i}^{\sigma}=A_{s_i+\sigma}^{E_i}/N_{s_i}$.

(5) According to the probabilities $p_{s_i}^{\sigma}$ select a new site s_{i+1} among the neighbors to go to and increase i by one, $i\rightarrow i+1$.

(6) If $i<W$ go to 3.

(7) Calculate $A_{s_W}^{-E_W}$ and $\overline{N}_{s_W}$ after the worm is constructed and with probability $P_e(w)=1-\min[1,A_{s_1}^{E_1}N_{s_1}/(A_{s_W}^{-E_W}\overline{N}_{s_W})]$ erase the constructed worm. Here $A_{s_1}^{E_1}$ and N_{s_1} are calculated *before* the worm is constructed.

(8) Go to 1.

Note that if we decide to construct a worm of length $W=1$ at a given site s_1 then the earlier algorithm corresponds to attempting a Metropolis spin-flip at that site. See Appendix A for a proof of the earlier algorithm.

B. Algorithm B (directed linear worms)

It is an obvious advantage to have control over the distribution of the length of the worms. However, if we choose the length of the worm at the start of the construction of the worm as in algorithm A then we allow for the worm to backtrack, thereby erasing itself. We can try to eliminate or rather minimize the probability for the worm to do backtracking by constructing a directed algorithm. We closely follow the method outlined in Ref. [8] in order to construct a directed algorithm. See also Ref. [11] for previous work on directed algorithms. Let σ_m and σ_n be among the neighbors, σ, of the site s_i. The earlier proof of algorithm A does not depend directly on the definition of the probabilities $p_{s_i}^{\sigma_m}$ and $p_{s_i}^{\sigma_n}$, but only on their ratio, since they have to satisfy the following relation:

$$\frac{p_{s_i}^{\sigma_m}}{p_{s_i}^{\sigma_n}}=\frac{A_{s_i+\sigma_m}/N_{s_i}}{A_{s_i+\sigma_n}/N_{s_i}}=\frac{A_{s_i+\sigma_m}}{A_{s_i+\sigma_n}}. \tag{2}$$

This leaves us considerable freedom since we can define conditional probabilities, $p_{s_i}(m|n)$, corresponding to the probability to continue in the direction σ_m at the site s_i if we are coming from σ_n. At a given site we then only need to satisfy

$$\frac{p_{s_i}(m|n)}{p_{s_i}(n|m)}=\frac{A_{s_i+\sigma_m}}{A_{s_i+\sigma_n}}. \tag{3}$$

If we have k neighbors this defines a $k\times k$ matrix P where the diagonal elements correspond to the backtracking probabilities, the probabilities for the worm to erase itself. Previously, we had effectively been using $p_{s_i}(m|n)$'s which were independent of the incoming direction, σ_n, since we simply had $p_{s_i}(m|n)=A_{s_i+\sigma_m}/N_{s_i}$, corresponding to a matrix P with identical columns. However, the earlier condition of balance, Eq. (3) leaves sufficient room for choosing very small backtracking probabilities and in most situations the corresponding diagonal elements of P can be chosen to be zero. If we impose the constraint that the sum of the diagonal elements of P should be minimal the problem of finding an optimal matrix P can be formulated as a standard linear programming problem to which conventional techniques can be applied [8].

At a given site s_i we choose four neighbors, as indicated in Fig. 1 by the open circles $\circ$. The activation at a given neighbor will depend on its four nearest neighbors shown as the shaded circles in Fig. 1. Technically, each $p_{s_i}(m|n)$ now depends on the position of all the 16 surrounding spins. and hence, there are in principle 2^{16} possible matrices P at each site. It is therefore not feasible to choose too large a set of neighbors when directing the algorithm. In the absence of disorder, the 2^{16} matrices can be tabulated and minimized at the beginning of the simulation and for the isotropic ferromagnetic model at hand it is easy to see that at most 625 *different* matrices occur.

We can now use the earlier matrices P for an algorithm that performs *directed* linear worms of length varying between 1 and W. We again assume that a set σ of k neighbors has been chosen.

(1) Choose a random starting site, $s_{i=1}$ and with normalized probability a length W.

(2) Calculate the probability for flipping the spin on that site $A_{s_1}^{E_1}$ with $A_{s_i}^{E_i}=\min[1,\exp(-\Delta E_{s_i}/k_BT)]$.

(3) Flip the selected spin, s_i.

(4) If $s_i=s_1$ then for each of the k neighbors in the set $\boldsymbol{\sigma}$, calculate the probability for flipping, $A_{s_i+\sigma}^{E_{i+\sigma}}=\min[1,\exp(-\Delta E_{s_i+\sigma}/k_BT)]$. Calculate the normalization $N_{s_i}=\Sigma_\sigma A_{s_i+\sigma}^{E_{i+\sigma}}$ and the probabilities $p_{s_i}^{\sigma}=A_{s_i+\sigma}^{E_i}/N_{s_i}$. **Else**: Depending on the configuration of the nearest neighbors of the k neighbors select the correct (minimized) matrix P and if the worm arrived from direction σ_n set $p_{s_i}^{\sigma}$ equal to the n'th column of P.

(5) According to the probabilities $p_{s_i}^{\sigma}$ select a new site s_{i+1} to go to and increase i by one, $i\rightarrow i+1$.

(6) If $i<W$ go to 3.

(7) Calculate $A_{s_W}^{-E_W}$ and $\overline{N}_{s_W}$ after the worm is constructed and with probability $P_e(w)=1-\min[1,A_{s_1}^{E_1}N_{s_1}/(A_{s_W}^{-E_W}\overline{N}_{s_W})]$ erase the constructed worm. Here $A_{s_1}^{E_1}$ and N_{s_1} are calculated *before* the worm is constructed.

(8) Go to 1.

The proof of the earlier algorithm can be found in Appendix B.

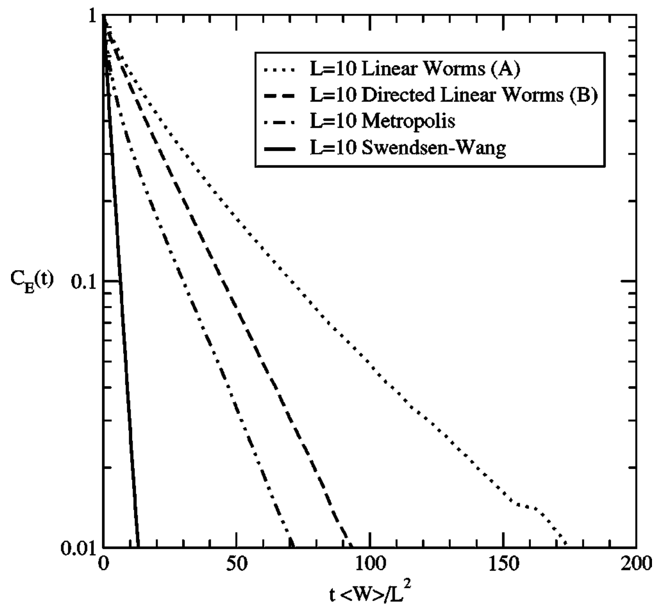

FIG. 2. Autocorrelation functions for the energy $C_E(t)$ as a function of Monte Carlo time, calculated at T_c. Shown are results for $L=10$ for the Metropolis, Swendsen-Wang and for the linear worm (A) and directed linear worm (B) algorithms (with uniform worm length). Note that in order to compare the four algorithms the time axis has been scaled so that in all cases one time step corresponds to an attempted update of all the variables. The autocorrelation functions shown were calculated averaging over 20 million worms (MCS for the metropolis algorithm).

C. Performance: Algorithms A,B

In order to test the performance of algorithms A and B we consider the autocorrelation function $C_{\mathcal{O}}(t)$ of an observable $\mathcal{O}$. This function is defined in the standard way

$$C_{\mathcal{O}}(t) = \frac{\langle \mathcal{O}(t)\mathcal{O}(0)\rangle - \langle \mathcal{O}\rangle^2}{\langle \mathcal{O}^2\rangle - \langle \mathcal{O}\rangle^2}. \tag{4}$$

For a reliable estimate of $C_{\mathcal{O}}(t)$ we typically generate 20 million worms. In Fig. 2 we show results for the autocorrelation function for the energy $C_E(t)$ for the linear worm algorithms A,B for a system of size $L=10$. In both cases the worm length W was chosen from a uniform distribution. This is compared to $C_E(t)$ for the usual single flip Metropolis algorithm and the Swendsen-Wang algorithm. For the latter two algorithms the Monte Carlo time is usually measured in terms of MCS where an attempt to update *all* the L^2 spins in the lattice has been made. However, for algorithms A and B a worm will on average only attempt to update $\langle W\rangle$ of the L^2 spins. Hence, if time is measured in terms of generated worms for the worm algorithms it should be rescaled by a factor of $\langle W\rangle/L^2$ for a fair comparison to be made with algorithms where Monte Carlo time is measured in MCS. This has been done in Fig. 2. From the results in Fig. 2 it is clear that the efficiency of algorithm A and B is fairly poor and worse than the much simpler Metropolis algorithm. We have checked that this remains true for significantly larger lattices sizes.

The main cause of this poor behavior is the necessity to include a rejection probability P_e. If a worm on average attempts to update $\langle W\rangle$ spins, we need on average to generate $L^2/\langle W\rangle$ worms to complete a MCS. As a rough estimate, let us consider that the worm is accepted with the average probability p and that only a fraction x of the $\langle W\rangle$ attempted spin flips result in an updated spin (a spin can be visited several times). It then follows that on average pxL^2 are actually flipped in a MCS with the linear worm algorithm. If the average probability for flipping a spin with the Metropolis algorithm is q and if all the spins selected in a MCS are different, the corresponding number for the Metropolis algorithm is qL^2, presumably of the same order as pxL^2 and likely larger. The effect of the directed linear worm algorithm is to maximize x as much as possible and it therefore seems unlikely that the linear worm algorithm in its present form can perform better than the Metropolis algorithm unless p also is maximized.

It would be very interesting if algorithms A and B could be modified so that $p\equiv 1$ $(P_e\equiv 0)$. We have so far been unable to do so. Such a modified algorithm would be significantly more efficient than the Metropolis algorithm (but presumably less efficient than the Swendsen-Wang algorithm). It would be much more versatile and could be of significant interest for frustrated or disordered systems since it would not require the construction of clusters but only of the much simpler worms, the length of which can be chosen.

Even though it is quite interesting to be able to choose the distribution of the worm lengths at the outset, the question arises which distribution of worm lengths will give the most optimal algorithm. In the present work we have chosen a distribution of worm lengths that is uniform between 1 and $2L$, but on general grounds we expect a power-law form for this distribution to be more optimal at the critical point. For the study of Bose-Hubbard models using geometrical worm algorithms it was noted [8] that the distribution of the size of the worms follows a power-law with an exponent of approximately 1.37 at the critical point. Hence, in the present case it would appear likely that an optimal power-law distribution of the worm lengths at T_c can be found, defining a "dynamical" exponent. The idea of choosing the distribution of the worm size to optimize the algorithm resembles previous work by Barkema and Newman [12,13].

In closing this section, we note that it is quite straightforward to define an algorithm on the direct lattice where the worms form closed loops (rings). In this case the length of the worm is determined by the size of the loop. We have tested such algorithms but their performance is even worse than algorithms A and B since the constraint that the initial spin has to be revisited makes the length of the worms in some cases diverge, or for other variants of such an algorithm, go to zero.

III. DUAL ALGORITHM

It is clear that one problem with both of the earlier algorithms is the fact that the spin on the initial site is treated differently than the remaining spins. This is because the geometrical worm algorithms are more suitable for an imple-

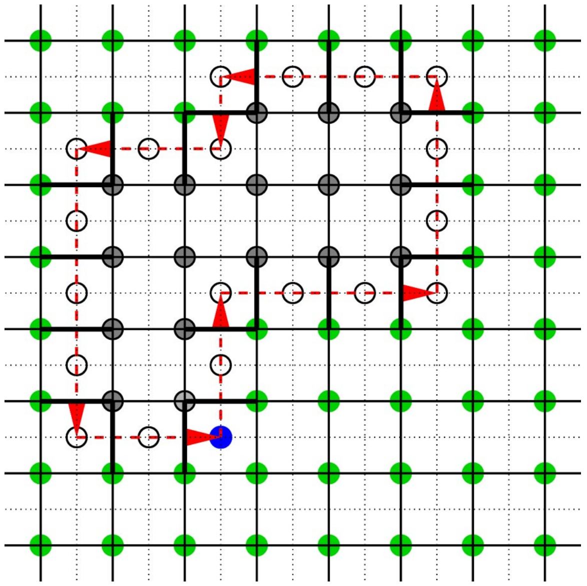
FIG. 3. A worm moving through the dual lattice. The direct lattice is indicated by solid lines and the spins on the direct lattice by solid circles. The dual lattice is indicated by dashed lines and the sites on the dual lattice by open circles. As the worm moves through the lattice the bond variables are flipped as indicated by the thick bonds in the figure.

mentation directly on the dual model. Hence, we will now try to describe an algorithm that moves a worm along the *dual* lattice by updating bond variables on the direct lattice See Fig. 3.

We begin with some definitions analogous to the treatment of Kadanoff [14]. On the direct lattice we define, at each site (j,k) an integer variable $s_{j,k}=\pm 1$. Here j describes the index in the x direction and k the index in the y direction. Then we can define the following bond variables:

$$b^x(j,k) = s_{j+1,k}s_{j,k}, \tag{5}$$

$$b^y(j,k) = s_{j,k+1}s_{j,k}. \tag{6}$$

The Ising model has L^2 variables where as we see that we have $2L^2$ bond variables. However, it is easy to see that the bond variables satisfy a divergence free constraint at each site of the dual lattice

$$b^x(j,k) + b^y(j+1,k) - b^x(j,k+1) - b^y(j,k)(\text{mod } 4) = 0 \tag{7}$$

giving us L^2 constraints. However, if we define the model on a torus the constraint on each dual lattice site is equal to the sum over the constraints on all the other dual sites. Hence, in this case we obtain only L^2-1 independent constraints. In addition we also have to satisfy the boundary conditions, $s_{L+1,k}\equiv s_{1,k}$, $s_{j,L+1}\equiv s_{j,1}$. This implies that for an $L\times L$ lattice with *even* L:

$$\sum_{j=1}^{L} b^x(j,k) - L(\text{mod } 4) = 0 \;\forall\; k, \tag{8}$$

$$\sum_{k=1}^{L} b^y(j,k) - L(\text{mod } 4) = 0 \;\forall\; j. \tag{9}$$

These constraints are not independent of the previously defined constraints Eq. (7). In fact, it is easy to see that if the boundary constraints Eq. (9) are applied at just one row and one column then the divergence free constraints Eq. (7) will enforce the boundary constraints at the remaining rows and columns. Hence, these constraints only give us 2 more independent constraints, in total L^2+1 constraints. The model, written in terms of the bond variables, therefore has L^2-1 free variables and correspondingly half the number of degrees of freedom compared to the formulation in terms of the spin variables. This is a natural consequence of the fact that the bond-variable model does not distinguish between a state and the same state with all the spins reversed.

The partition function for the Ising model on a $L\times L$ torus can now be written in the following manner:

$$Z = \text{Tr}'_{b^x}\text{Tr}'_{b^y}\prod_{j=1}^{L}\prod_{k=1}^{L}\exp\{K^x_{j,k}b^x(j,k) + K^y_{j,k}b^y(j,k)\}. \tag{10}$$

Here Tr′ denotes the trace over bond variables satisfying the earlier constraints and $K^x_{j,k}=J^x_{j,k}/k_BT$, $K^y_{j,k}=J^y_{j,k}/k_BT$.

A. Algorithm C (dual worm algorithm)

We now turn to a discussion of the dual algorithm. From the earlier description in terms of bond variables it is now quite easy to define a geometrical worm algorithm on the dual lattice closely following previous work on such algorithms [7,8]. We denote the ith site on the dual lattice that the worm visits as d_i. A worm is constructed by going through a sequence of neighboring sites on the dual lattice by each time choosing a direction σ to follow according to an appropriately determined probability. When the worm moves from d_i to d_{i+1} the corresponding bond-variable b_i that the worm crosses is flipped. The bond variables can take on only 2 values ±1. Hence, the associated energy cost for flipping b_i is given by $\Delta E=2J^{\alpha}_{j,k}b^{\alpha}(j,k)$, $\alpha=x,y$. We can now define weights for each direction σ, $A^{E_\sigma}=\min[1,\exp(-\Delta E^{\sigma}/k_BT)]$ used for determining the correct probability for choosing a new site. This is not the only choice for the weights. Other equivalent choices should work equally well. The bond variables are updated during the construction of the worm and the worm is finished when the starting site on the dual site is reached again and the worm forms a closed loop. The dual worm algorithm can then be summarized using the following pseudocode.

```
(1) Choose a random initial site $d_1$ on the dual lattice.
(2) For each of the directions $\sigma=\pm x,\pm y$ calculate the weights $A^{E_\sigma}$ associated with flipping the bond variable perpendicular to that direction, $A^{E_\sigma}=\min[1,\exp(-\Delta E^{\sigma}/k_BT)]$.
(3) Calculate the normalization $N_{d_i}=\Sigma_\sigma A^{E_\sigma}$ and the associated probabilities $p^{\sigma}_{d_i}=A^{E_\sigma}/N_{d_i}$.
(4) According to the probabilities, $p^{\sigma}_{d_i}$, choose a direction $\sigma$.
(5) Update the bond-variable $b^{\sigma}_i$ for the direction chosen and move the worm to the new dual lattice site $d_{i+1}$.
```

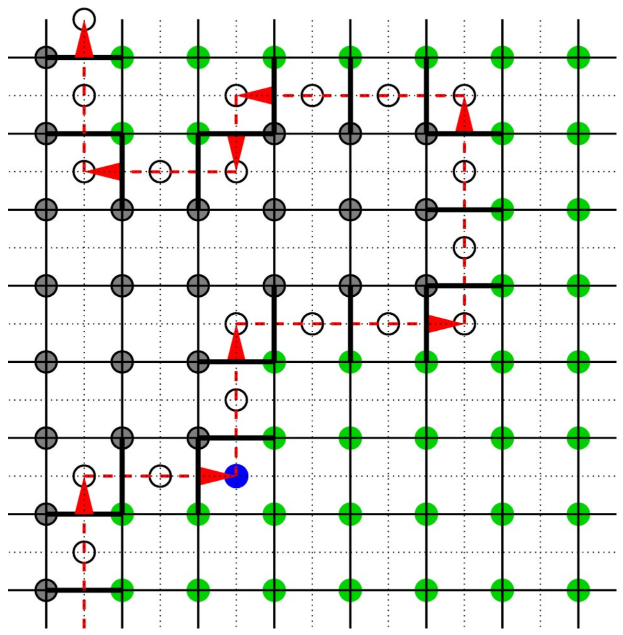

FIG. 4. An illegal worm move. The worm winds around the lattice in the vertical direction and correspondingly *all* rows have an *odd* number of flipped bond variables b. The direct lattice is indicated by solid lines and the spins on the direct lattice by solid circles. The dual lattice is indicated by dashed lines and the sites on the dual lattice by open circles. As the worm moves through the lattice the bond variables are flipped as indicated by the thick bonds in the figure.

(6) If $d_i \neq d_1$ go to 2.

(7) Calculate the normalizations $\overline{N}_{d_1}$ and N_{d_1} of the initial site, s_1, with and without the worm present. If the worm is "legal," i.e., with *even* winding number in both the x and the y direction (O^x_w, O^y_w both 0—see definition later), then erase the worm with probability $P_e = 1 - \min(1, N_{d_1}/\overline{N}_{d_1})$. If the worm is "illegal," that is if either O^x_w or O^y_w calculated when the worm has closed equal 1, then always erase it. Go to 1.

Following the earlier discussion of the boundary constraints it is easy to determine if the winding number in the x or y direction is odd by simply choosing one row k_0 and one column j_0 and calculating the number of frustrated, $b = -1$, bond variables

$$O^x = \sum_j \frac{1 - b^x(j,k_0)}{2} (\text{mod } 2),$$

$$O^y = \sum_k \frac{1 - b^y(j_0,k)}{2} (\text{mod } 2), \tag{11}$$

since a worm with an odd winding number in the y direction will result in O^x being 1 independent of k_0, or equivalently for the x direction and O^y. See Fig. 4. O^x and O^y then take on the values 1 or 0 depending on whether the corresponding winding numbers are odd or even.

Several points are noteworthy about this algorithm. To a great extent it is simply the dual version of the Wolff algorithm [2]. Each worm encloses a cluster of spins on the direct lattice (not necessarily a simply connected cluster). Flipping the bond variables in the worm effectively flips the spin in the cluster. This correspondence is particular to two dimensions which is the only dimension where the present dual worm algorithm can be defined. The intuitive argument for this is that the bond variables updated by the worm have to enclose a finite volume of spins on the direct lattice; this is only possible in two dimensions. In dimensions higher than 2 the one-dimensional worms are not capable of enclosing a finite volume. Mathematically, it can be seen that the divergenceless constraint, Eq. (7), cannot be satisfied under worm moves. Under most conditions the rejection probability, P_e, is very small, however, for small lattice sizes the probability for generating a worm with an *odd* winding number in either the x or y direction can be non-negligible. It is important to note that since the bond variables are updated during the construction of the worm the generated configurations are not valid and the earlier constraints are *not* satisfied. However, once the construction of the worm is finished and the path of the worm closed, the divergenceless constraint is satisfied. Rejecting worms with odd winding numbers then assures that the resulting configurations are valid. Also, when the worm moves through the lattice it may pass many times through the same link and cross itself before it reaches the initial site where the construction terminates. Finally, at each step i in the construction of the worm it is likely that the worm at the site d_i will partially "erase" itself by choosing to go back to the site d_{i-1} visited immediately before, thereby "bouncing" off the site d_i.

Proof of algorithm C (dual worm algorithm)

Now we turn to the proof of detailed balance for the algorithm. As before, let μ denote the configuration of the bond variables without the worm, w, and ν the configuration with the worm. Let us consider the case where the worm, w, visits the sites $\{d_1 \ldots d_W\}$ on the dual lattice. Here d_1 is the initial site. The worm then goes through the corresponding bond variables $\{b_1 \ldots b_W\}$. Note that d_W is the last site visited before the worm reaches d_1. Furthermore, let E_i denote the energy required to flip the bond-variable b_i from its position in the configuration μ, with $-E_i$ the energy required to flip b_i from its position in ν to that in μ. The total probability for constructing the worm w is then given by

$$P(w; d_1 \to d_W) = P(d_1)[1 - P_e(w)] P(\text{legal}|w) \prod_{i=1}^{W} \frac{A^{E_i}}{N_{d_i}}. \tag{12}$$

The index σ denotes the direction needed to go from d_i to d_{i+1}, $(\pm x, \pm y)$, crossing the bond-variable b_i. $P(d_1)$ is the probability for choosing site d_1 as the starting point and $P_e(w)$ is the probability for erasing a legal worm after construction. Finally, $P(\text{legal}|w)$ is the conditional probability that the constructed worm w is legal, with *even* winding numbers in both x and y directions. If the worm w is legal and has been accepted we have to consider the probability for reversing the move. That is, we consider the probability for constructing an anti-worm $\overline{w}$ annihilating the worm w.

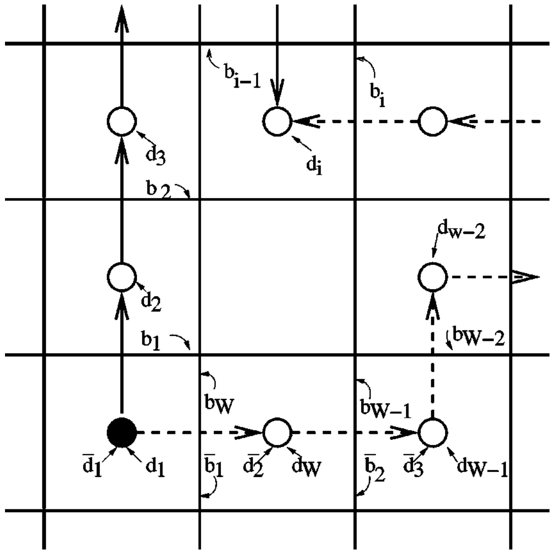

FIG. 5. The notation used for the proof of the dual worm algorithm. The configuration shown corresponds to a worm (solid line) partly erased by its corresponding antiworm (dashed line).

Note that, in this case the sites are visited in the opposite order, $\overline{d}_1=d_1,\overline{d}_2=d_W,\ldots,\overline{d}_W=d_2$, in general $\overline{d}_i=d_{W-i+2}$ $(i\neq 1)$. In the same manner the bond variables are also visited in the reverse order $\overline{b}_1=b_W,\overline{b}_2=b_{W-1},\ldots,\overline{b}_W=b_1$, in general $\overline{b}_i=d_{W-i+1}$. Consequently, when describing $\overline{w}$, we directly use d,b instead of $\overline{d},\overline{b}$. The notation is illustrated in Fig. 5. We therefore have

$$P(\overline{w};\overline{d}_1\rightarrow\overline{d}_W)=P(d_1)[1-P_e(\overline{w})]P(\text{legal}|\overline{w})\times\frac{A^{-E_W}}{\overline{N}_{d_1}}\prod_{i=W}^{2}\frac{A^{-E_{i-1}}}{\overline{N}_{d_i}}. \tag{13}$$

Please note that the index i runs over $W,\ldots,2$. Let us now consider the case where both of the worms w and $\overline{w}$ have reached the site d_i *different* from the starting site d_1. See Fig. 5. Since we are updating the link variables during the construction of the worm we immediately see that $N_{d_i}=\overline{N}_{d_i}$ for $i\neq 1$. Furthermore, A^{E_i} and A^{-E_i} only depend on the bond-variable b_i, and we therefore see that

$$\frac{A^{E_i}}{A^{-E_i}}=\exp(-\Delta E_i/k_BT),\quad i=1\ldots W. \tag{14}$$

Hence, since $P(d_1)$ is uniform throughout the dual lattice, and since w and $\overline{w}$ must wind the lattice in precisely the same manner resulting in $P(\text{legal}|w)=P(\text{legal}|\overline{w})$, we find

$$\frac{P(w)}{P(\overline{w})}=\frac{1-P_e(w)}{1-P_e(\overline{w})}\frac{\overline{N}_{d_1}}{N_{d_1}}\exp(-\Delta E_{\text{Tot}}/k_BT), \tag{15}$$

where ΔE_{Tot} is the total energy difference between a configuration with and without the worm w present. With our definition of P_e we see that $[1-P_e(w)]/[1-P_e(\overline{w})]=N_{d_1}/\overline{N}_{d_1}$. Hence, with this choice of P_e we satisfy detailed balance since

$$\frac{P_w}{P_{\overline{w}}}=\exp(-\Delta E_{\text{Tot}}/k_BT). \tag{16}$$

As was the case for the previous algorithms there are two ways of introducing a worm that will take us from $\mu\rightarrow\nu$, and equivalently two ways of introducing $\overline{w}$. (Strictly speaking, each of the two ways actually represents an infinite sum as discussed under the proof of algorithm A.) In both cases Eq. (16) holds and it follows that $P(\mu\rightarrow\nu)=P(\nu\rightarrow\mu)\exp(-\Delta E_{\text{Tot}}/k_BT)$ showing that detailed balance is satisfied. Ergodicity is simply proven as the worm can perform local loops and wind around the lattice in any direction.

Illegal worms. One should be cautious when treating illegal moves such as the worms with odd winding numbers encountered earlier. Suppose that our Monte Carlo algorithm has generated a number of configurations $C_1,C_2,C_2,C_3,\ldots,C_{i-1},C_i$ where some configurations are repeated since the worm moved has been rejected. From the configuration C_i we now construct a worm that turns out to be illegal with odd winding numbers. From the earlier proof we then see that we *have* to repeat C_i in the sequence of configurations since *not* repeating the configuration would lead to a total acceptance probability different from $[1-P_e(w)]P(\text{legal}|w)$. For instance, imagine we kept generating worms till we found a legal one. This would make the total acceptance ratio a sum over the number of times we try which would be incorrect.

B. Algorithm D (dual directed worm algorithm)

Following the discussion of algorithm B it is straightforward to develop a directed version of algorithm C. As was the case for algorithm B we define conditional probabilities $p_{d_i}(m|n)$, corresponding to the probability for continuing in the direction σ_m if the worm has come from direction σ_n. Since we in algorithm C, at each site on the dual lattice d_i, can go in four directions this leads us to define a 4×4 matrix P at each site on the dual lattice. However, compared to algorithm B, the number of different matrices at a given dual site is now much smaller since the weights A^{E_i} *only* depend on the associated bond variable. In fact there are now only 16 possible matrices at a given site and these can easily be optimized and tabulated at the outset of the calculation. Even in the presence of disorder, a large number of them can be tabulated. This directed dual worm algorithm is now identical to algorithm C except for the fact that if d_i is different from d_1, the $p_{d_i}^{\sigma}$ are selected from these optimized matrices (the same way as it was done for algorithm B). Due to the simplicity of this modification and the similarity with algorithm B, we do not present pseudocode nor a proof for this directed algorithm. It is easy to see that rejection probability remains identical to the one used in algorithm C, $P_e(w)=1-\min(1,N_{d_1}/\overline{N}_{d_1})$.

C. Performance algorithm C,D

In order to compare the two algorithms C and D defined on the dual lattice we have again calculated the autocorrela-

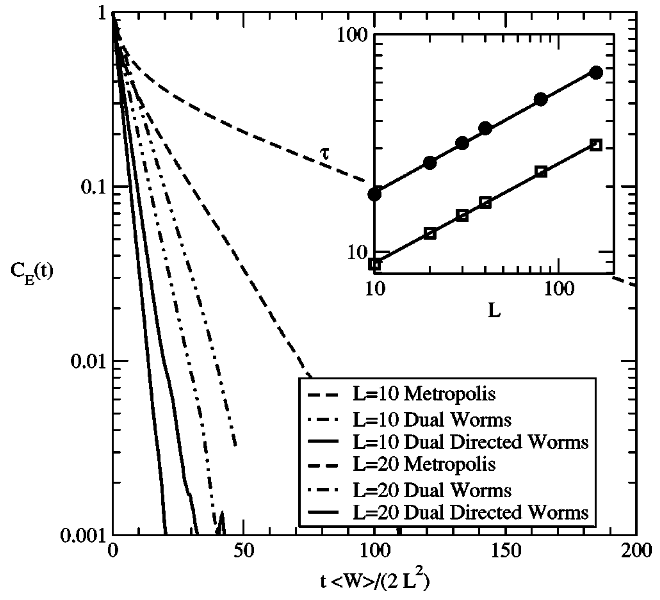

FIG. 6. Autocorrelation functions for the energy $C_E(t)$ as a function of Monte Carlo time, calculated at T_c. Shown are results for $L=10,20$ for the Metropolis, dual worm and dual directed worm algorithms. Note that, in order to compare the three algorithms the time axis has been scaled so that in all cases one time step corresponds to an attempted update of all the variables. The inset shows the scaling of the autocorrelation time τ' as a function of the system size for the dual worm and dual directed worm algorithm. The autocorrelation time is here rescaled: $\tau'=\tau\langle W\rangle/(2L^2)$. In both cases we find roughly a power-law form $\tau'\simeq L^{0.46}$ shown as the solid lines in the inset. The autocorrelation functions shown were calculated averaging over 10–20 million worms (MCS for the metropolis algorithm).

tion function $C_E(t)$. Our results are shown in Fig. 6 and tabulated in Table I. As on the direct lattice we use the energy to calculate the autocorrelation functions. For algorithms C and D we have constructed 10 million worms for the smaller lattices and 20 million for the largest lattice ($L=160$). We compare to the Metropolis algorithm where each time step corresponds to an attempted update of the L^2 spins. Since the worm algorithm in this case on average only attempts to update $\langle W\rangle$ out of the $2L^2$ bond variables we have to rescale the time axis by a factor of $\langle W\rangle/(2L^2)$ when we compare algorithms C and D to the Metropolis algorithm. We can therefore define an autocorrelation time τ in units of worms constructed and a rescaled autocorrelation time $\tau' = \tau\langle W\rangle/(2L^2)$ in units of $2L^2$ attempted updates. In order to simplify the analysis of the data we define the autocorrelation time as $C_E(\tau)\equiv 0.05$. As can be seen in Fig. 6, the dual worm algorithms represent a dramatic improvement over the Metropolis algorithm. We have not plotted results for the Swendsen-Wang algorithm on the direct lattice since for $L=10$ they are indistinguishable from the results obtained for the dual directed worm algorithm. The scaling of τ' with L is a relatively small power-law $\tau'\sim L^{0.46}$. This exponent is small but *larger* than most values quoted for the Swendsen-Wang and Wolff algorithms. This is perhaps not too surprising since the worms are allowed to cross themselves thereby erasing previous updates. The difference in autocorrelation times between algorithm C and its directed version D appears to be a constant factor. Therefore, the direction of algorithm C does not change the exponent z_{MC}.

TABLE I. Average worm size defined as the average number of bond variables attempted to change during the construction of any worm, including worms eventually rejected. Also listed is the estimated autocorrelation time, τ, in units of worms constructed and the rescaled autocorrelation time $\tau'=\tau\langle W\rangle/(2L^2)$ in units of $2L^2$ attempted updates. The autocorrelation time is here defined by $C_E(\tau)=0.05$. Results are listed for the dual and dual directed worm algorithms.

	Dual			Dual directed		
L	$\langle W\rangle$	τ	τ'	$\langle W\rangle$	τ	τ'
10	57	64.6	18.4	60	29.3	8.8
20	188	109.4	25.7	198	49.3	12.2
30	382	149.4	31.7	400	66.6	14.8
40	630	187.9	37.0	660	81.9	16.9
80	2119	303.8	50.3	2204	135.9	23.4
160	7132	478.8	66.7	7408	214.3	31.0

For a 10×10 lattice at T_c roughly 85% of all constructed worms are accepted and about 10% of the constructed worms are illegal when simulations are performed using the dual geometric worm algorithm. For the directed version of the algorithm the corresponding numbers are 75% and 20%. The number of illegal worms drops rapidly with L and for $L=30$, 90% of the worms are accepted with 5% illegal worms for the undirected algorithm compared to 83% and 10% for the directed dual algorithm.

The presented dual algorithms are quite general since they depend only in a relatively limited way on the underlying Hamiltonian. Even though they may not be competitive with the Swendsen-Wang and Wolff algorithms for the Ising model they may be quite interesting to apply to other two-dimensional models that can be formulated in terms of bond variables and where more effective cluster algorithms are difficult to define.

D. Domain wall free energy

A quantity of particular interest in many studies of ordering transitions is the domain wall free energy, the difference in free energy between configurations of the system with periodic (P) and antiperiodic (AP) boundary conditions in one direction, $\Delta F=F_{\mathrm{AP}}-F_P$. The domain wall free energy can be shown to obey simple scaling relations and is a very efficient tool for distinguishing between different ordered phases. Unfortunately, it is usually very difficult to calculate free energies directly in Monte Carlo simulations. A clever trick to do so is the boundary flip MC method, proposed by Hasenbusch [15,16], where the coupling constants at the boundary are considered as dynamical variables, and can be flipped during the course of the MC simulation. If $P_P(T)$ and $P_AP(T)$ describe the probability to obtain MC configurations with periodic and antiperiodic boundary conditions

TABLE II. The domain wall free-energy $\Delta F/kT$ for several system sizes calculated at the critical temperature T_c obtained using our MC method and exact results from Ref. [17].

L	Exact	MC
4	0.9658246670	0.9654(5)
16	0.9853282229	0.985(10)
24	0.9859725679	0.985(10)
32	0.9861972559	0.986(10)
48	0.9863574947	0.986(10)
64	0.9864135295	0.984(20)

and Z_P, Z_{AP} the associated partition functions, then the domain wall free-energy is given by

$$e^{-\beta(F_{\mathrm{AP}}-F_P)} = \frac{Z_{\mathrm{AP}}}{Z_P} = \frac{P_{\mathrm{AP}}(T)}{P_P(T)}. \qquad (17)$$

Effectively, the boundary flip MC method attempts to sample the ratio Z_{AP}/Z_P, and this is usually cumbersome. However, for the dual algorithms it is trivial to obtain this ratio. It follows from Eq. (9), that if we allow the constructed worms to have all different kinds of winding around the torus (making all worms legal) then we are sampling a partition function which is a sum of four terms

$$Z = Z_{P_x,P_y} + Z_{\mathrm{AP}_x,P_y} + Z_{P_x,\mathrm{AP}_y} + Z_{\mathrm{AP}_x,\mathrm{AP}_y}. \qquad (18)$$

Here, $Z_{\mathrm{BC}_x,\mathrm{BC}_y}$ is the partition function with $\mathrm{BC}_x(\mathrm{BC}_y)$ in the $x(y)$ direction. This is so, because we can turn an illegal worm into a legal worm by changing the boundary condition in the direction(s) where the winding of the worm violates Eq. (9). During the MC simulation it is easy to see to which term a given configuration contributes simply by keeping track of the *total* winding number in both the x and y direction. See Eq. (11). When this is even, the boundary condition in the associated direction is periodic, and if it is odd the boundary condition is antiperiodic. In order to calculate $Z_{\mathrm{AP}_x,P_y}/Z_{P_x,P_y}$ we therefore simply have to count how many times the *total* winding number is odd in the x direction and even in the y direction and divide with the number of times it is even in both directions. Due to its simplicity, we expect this approach to be an extremely efficient way of calculating ΔF. In Table II we show preliminary results for $\Delta F/kT$ calculated with 10^8 worms for the two-dimensional Ising model at T_c. For reference we compare to exact results obtained using pfaffians [17].

IV. DISCUSSION

We have presented two generalizations of the geometrical worm algorithm applicable to classical statistical mechanics model. The first algorithm exploits linear worms on the direct lattice, the other closed loops of worms on the dual lattice. In both cases we have developed directed versions of the algorithms. While the first algorithm is applicable in any spatial dimension, it is quite clear that for topological reasons it is only possible to define a worm algorithm of the proposed kind on the dual lattice in two spatial dimensions.

Due to its poor performance the linear worm algorithm (A) is mainly of interest from the perspective of fostering the development of more advanced algorithms. The linear worm algorithm presents a number of very attractive features such as the possibility to choose the distribution of the worm lengths and it seems possible to significantly improve the performance of the algorithm (and most other geometrical worm algorithms) by eliminating the rejection probability. So far, we have been unable to do so, but this would clearly be very interesting since the algorithms then would be quite promising for the study of frustrated or even disordered models where other cluster algorithms fail.

The dual worm algorithm is very efficient and even though the obtained z_{MC} is larger than for the Swendsen-Wang algorithm it is quite competitive since it requires relatively little overhead and is very simple to implement. One of the most interesting features of this algorithm is the topological aspect of the worms, resulting in forbidden worm moves.

The geometrical worm algorithm has in some cases been very successfully applied to the study of models with disorder [7]. We therefore applied both the linear and dual worm algorithm to the bond disordered Ising model. In both cases do the algorithms perform worse than the Metropolis algorithm. For the dual worm algorithm this failure is due to the fact that the average size of worms diverge when glassy phases are approached. The linear worm algorithm would seem more promising for study of models with disorder since the length of the worms can be controlled from the outset. In fact the performance of the algorithm is in this case quite comparable to the Metropolis algorithm (which has a diverging τ close to a glassy phase). One might have expected the linear worm algorithm to perform significantly *better* than the Metropolis algorithm when disorder is present. We assume that the fact that it is only comparable to the Metropolis algorithm is because large parts of the spins are not effectively flipped since worms are mostly rejected in those parts of the lattice. If this is true, a modified version of the linear worm algorithm with $P_e \equiv 0$ or with a much more ingenious choice of the neighbors would be of considerable interest since it would hold the potential to sample such disordered models significantly more effectively than the Metropolis algorithm.

ACKNOWLEDGMENTS

This research is supported by NSERC of Canada as well as SHARCNET. P.H. gratefully acknowledges SHARCNET for a summer scholarship. F.A. acknowledges support from the Swiss National Science Foundation and stimulating discussions with Philippe de Forcrand. The authors are grateful to Philippe de Forcrand for suggesting the earlier way of calculating ΔF.

APPENDIX A: PROOF OF ALGORITHM A

The probability for creating the worm, w, starting from site s_1 is given by

$$P(w;s_1 \to s_W) = P(W)P(s_1)\frac{A_{s_2}^{E_2}}{N_{s_1}}\frac{A_{s_3}^{E_3}}{N_{s_2}}\cdots\frac{A_{s_W}^{E_W}}{N_{s_{W-1}}}[1 - P_e(w)]. \tag{A1}$$

Here $P(W)$ denotes the probability for choosing a worm length of W. The probability for introducing an antiworm, $\overline{w}$, starting at the site s_1, erasing the worm by going in the opposite direction along w becomes

$$P(\overline{w};s_W \to s_1) = P(W)P(s_W)\frac{A_{s_{W-1}}^{-E_{W-1}}}{\overline{N}_{s_W}}\cdots\frac{A_{s_2}^{-E_2}}{\overline{N}_{s_3}}\frac{A_{s_1}^{-E_1}}{\overline{N}_{s_2}}[1 - P_e(\overline{w})]. \tag{A2}$$

Since neither N_{s_i} nor $\overline{N}_{s_i}$ depend on the position of the spin on the site i and since the spins are visited in the precise opposite order for the antiworm with respect to the worm, we then see that $N_{s_i}=\overline{N}_{s_i}$ $\forall i \neq 1, W$, and we find

$$\frac{P(w;s_1 \to s_W)}{P(\overline{w};s_W \to s_1)} = \frac{1 - P_e(w)}{1 - P_e(\overline{w})}\frac{A_{s_2}^{E_2}\ldots A_{s_W}^{E_W}}{A_{s_{W-1}}^{-E_{W-1}}\ldots A_{s_1}^{-E_1}}\frac{\overline{N}_{s_w}}{N_{s_1}}. \tag{A3}$$

Since $A_{s_i}^{E_i}/A_{s_i}^{-E_i}=\exp(-\Delta E_i/k_BT)$ we see, using our definition of P_e, that

$$\frac{P(w;s_1 \to s_W)}{P(\overline{w};s_W \to s_1)} = \exp(-\Delta E/k_BT). \tag{A4}$$

Quite generally, there is more than one way to introduce a worm to go from μ to ν. In fact there are two ways of introducing a worm taking the system from configuration μ to ν, one where the worm traverses the sites from s_1 to s_W and another where the worm traverses the sites from s_W to s_1. Equivalently, there are two ways of introducing the antiworm. Hence, $P(\mu \to \nu)$ is a sum of the two probabilities, $P(\mu \to \nu) = P(w;s_1 \to s_W) + P(w;s_W \to s_1)$. However, employing the result we have just proven we see that

$$\begin{aligned} P(\mu \to \nu) &= [P(\overline{w};s_W \to s_1) + P(\overline{w};s_1 \to s_W)] \\ &\quad \times \exp(-\Delta E/k_BT) \\ &= P(\nu \to \mu)\exp(-\Delta E/k_BT). \end{aligned} \tag{A5}$$

Hence, the proposed algorithm satisfies detailed balance on a bipartite lattice and we see that detailed balance is satisfied no matter how many possible ways one can introduce a worm in order to go from $\mu \to \nu$ as long as the number of possible antiworms matches. Strictly speaking, since the worm can partly erase itself, there *is* in fact an infinite number of worm-moves that will take us from μ to ν, however, in each case there is a matching antiworm so we can replace the earlier sum over two probabilities with an infinite sum, yielding the same conclusion. Ergodicity is proven by noting that single spin flips are incorporated in the algorithm.

APPENDIX B: PROOF OF ALGORITHM B

The proof is quite similar to the proof of algorithm A and we use the same notation. The probability for creating a worm, w, starting from site s_1 is then given by

$$\begin{aligned} P(w;s_1 \to s_W) = P(W)P(s_1)\frac{A_{s_2}^{E_2}}{N_{s_1}}p_{s_2}(s_3|s_1)\ldots p_{s_{W-2}} \\ \times(s_{W-1}|s_{W-3})p_{s_{W-1}}(s_W|s_{W-2})[1 - P_e(w)]. \end{aligned} \tag{B1}$$

Here $P(s_1)$ denotes the probability for choosing site s_1 as the starting site for the worm, $P(W)$ the probability for choosing a worm length W. $p_{s_i}(s_{i+1}|s_{i-1})$ denotes the conditional probability, extracted from the minimized matrix P, at the site s_i for continuing to the site s_{i+1} knowing that the worm is coming from the site s_{i-1}. Starting from the configuration ν we can now calculate the probability for introducing an antiworm, $\overline{w}$, starting at the site s_W, erasing the worm, w. We find

$$\begin{aligned} P(\overline{w};s_W \to s_1) = P(W)P(s_W)\frac{A_{s_{W-1}}^{-E_{W-1}}}{\overline{N}_{s_W}}\overline{p}_{s_{W-1}}(s_{W-2}|s_W)\ldots \overline{p}_{s_3} \\ \times(s_2|s_4)\overline{p}_{s_2}(s_1|s_3)[1 - P_e(\overline{w})]. \end{aligned} \tag{B2}$$

At a given site, s_i, the configurations of the spins during the construction of the worm and the antiworm only differ by the position of the spin at the site itself. Since the set of neighbors σ exclude any nearest neighbors to s_i we see that $P_{s_i}=\overline{P}_{s_i}$. Hence, by construction

$$\frac{p_{s_i}(s_{i+1}|s_{i-1})}{\overline{p}_{s_i}(s_{i-1}|s_{i+1})} = \frac{A_{s_{i+1}}^{E_{i+1}}}{A_{s_{i-1}}^{-E_{i-1}}}. \tag{B3}$$

Using the definition of P_e and the fact that $P(s_1)=P(s_W)$, we then see that

$$\frac{P(w;s_1 \to s_W)}{P(\overline{w};s_W \to s_1)} = \frac{A_{s_1}^{E_1}\ldots A_{s_W}^{E_W}}{A_{s_1}^{-E_1}\ldots A_{s_W}^{-E_W}}. \tag{B4}$$

Since $A_{s_i}^{E_i}/A_{s_i}^{-E_i}=\exp(-\Delta E_i/k_BT)$ we finally see that

$$\frac{P(w;s_1 \to s_W)}{P(\overline{w};s_W \to s_1)} = \exp(-\Delta E/k_BT), \tag{B5}$$

where ΔE is the *total* energy cost in going from configuration μ to configuration ν. As was the case for algorithm A, there are two ways of introducing a worm taking the system from configuration μ to ν (modulo sites visited an even number of times), one where the worm traverses the sites from s_1 to s_W and another where the worm traverses the sites from s_W to s_1. Equivalently, there are two ways of introducing the antiworm. We again see that detailed balance is satisfied. Ergodicity is proven by noting that single spin flips are incorporated in the algorithm, since for $W=1$ the algorithm corresponds to attempting a Metropolis spin flip at the selected site.

[1] R. H. Swendsen and J. S. Wang, Phys. Rev. Lett. **58**, 86 (1987).
[2] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).
[3] J. Machta, Y. S. Choi, A. Lucke, T. Schweizer, and L. V. Chayes, Phys. Rev. Lett. **75**, 2792 (1995).
[4] F. Wang and D. P. Landau, Phys. Rev. Lett. **86**, 2050 (2001).
[5] N. Prokof'ev and B. Svistunov, Phys. Rev. Lett. **87**, 160601 (2001).
[6] N. V. Prokof'ev, B. V. Svistunov, and I. S. Tupitsyn, Phys. Lett. A **238**, 253 (1998).
[7] F. Alet and E. S. Sørensen, Phys. Rev. E **67**, 015701(R) (2003).
[8] F. Alet and E. S. Sørensen, Phys. Rev. E **68**, 026702 (2003).
[9] M. E. J. Newman and G. T. Barkema, *Monte Carlo Methods in Statistical Physics* (Oxford University Press, New York, 1999).
[10] D. P. Landau and K. Binder, *A Guide to Monte Carlo Methods in Statistical Physics* (Cambridge University Press, Cambridge, 2000)
[11] O. F. Syljuåsen and A. W. Sandvik, Phys. Rev. E **66**, 046701 (2002).
[12] G. T. Barkema and J. F. Marko, Phys. Rev. Lett. **71**, 2070 (1993).
[13] M. E. J. Newman and G. T. Barkema, Phys. Rev. E **53**, 393 (1996).
[14] L. P. Kadanoff, *Statistical Physics, Statics, Dynamics and Renormalization* (World Scientific, Singapore, 2000).
[15] M. Hasenbusch, J. Phys. I **3**, 753 (1993).
[16] K. Hukushima, Phys. Rev. E **60**, 3606 (1999).
[17] E. Sørensen, e-print cond-mat/0006233.