PHYSICAL REVIEW
LETTERS

VOLUME 70 | 15 FEBRUARY 1993 | NUMBER 7

Cluster Algorithm for Vertex Models

Hans Gerd Evertz,[1],[a] Gideon Lana,[2],[b] and Mihai Marcu[2],[c]
[1] *Supercomputer Computations Research Institute, Florida State University, Tallahassee, Florida 32306*
[2] *School of Physics and Astronomy, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, 69978 Tel Aviv, Israel*
(Received 17 November 1992)

We present a new type of cluster algorithm that strongly reduces critical slowing down in simulations of vertex models. Since the clusters are closed paths of bonds, we call it the *loop algorithm*. The basic steps in constructing a cluster are the breakup and the freezing of vertices. We concentrate on the case of the F model, which is a subset of the six-vertex model exhibiting a Kosterlitz-Thouless transition. The loop algorithm is also applicable to simulations of other vertex models and of one- and two-dimensional quantum spin systems.

PACS numbers: 02.70.–c, 05.50.+q, 68.35.Rh, 75.10.Jm

Cluster algorithms [1, 2] are one of the few known ways to overcome critical slowing down in Monte Carlo simulations. Starting with [3] and continuing with new ideas like [4] and [5], most of the successful algorithms have dealt with spin systems with two-spin interactions (see, however, [6]).

In *vertex models* [7, 8] the dynamical variables are localized on bonds, and the interaction is between all bonds meeting at a vertex. Furthermore, there are constraints on the possible bond variable values around a vertex.

In this paper we present the *loop algorithm*, a new type of cluster algorithm applicable to vertex models. For usual spin systems most cluster algorithms start by "freezing" (also called "activating") or "deleting" bonds. Clusters are then sets of sites connected by frozen bonds. In the case of vertex models our idea is to define clusters as *closed paths of bonds* ("loops"). To construct such clusters, we have to perform operations at vertices that generalize the freeze-delete procedure. In this context we introduce the concept of *breakup of a vertex*.

For the sake of clarity we concentrate on the F model, which is one of the simplest vertex models. We define it on an $L \times L$ square lattice. Vertices are located at lattice sites. The bond variables take the values ± 1. They can be represented by arrows (e.g., $+1$ means arrow up or right, -1 means arrow down or left). At each vertex we have the *constraint* that the number of incoming arrows equals the number of outgoing arrows. Thus there are six different vertex configurations (six "vertices"), as shown in Fig. 1. Their statistical weights $w(i)$, $i = 1, \ldots, 6$, are

$$w(i) = \begin{cases} e^{-K}, & i = 1, 2, 3, 4, \\ 1, & i = 5, 6. \end{cases} \tag{1}$$

The coupling $K \geq 0$ plays the role of inverse temperature. At $K_c = \ln 2$ there is a Kosterlitz-Thouless transition. The correlation length is finite for $K > K_c$ and infinite for $K \leq K_c$.

In what follows we start by presenting our new loop algorithm. It turns out that there is one free parameter in the algorithm. We discuss how to choose an *optimal* value. Then we analyze the exponential autocorrelation times at $K = K_c$ and at $K = K_c/2$. For the optimum algorithm we find a dynamical critical exponent of $z(K_c) = 0.71(5)$ and $z(K_c/2) = 0.19(2)$. No critical slowing down is visible for the total energy. We briefly show how to generalize our algorithm to more general six- and

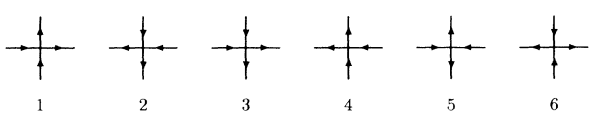

FIG. 1. The six vertex configurations. The labels $1, \ldots, 6$ follow standard conventions [7].

eight-vertex models and how to use it for simulations of quantum spin systems.

The loop algorithm.—If we regard the arrows on bonds as a vector field, the constraint at the vertices is a zero-divergence condition. Therefore every configuration change can be obtained as a sequence of *loop flips*. By "loop" we denote an oriented, closed, nonbranching (but possibly self-intersecting) path of bonds, such that all arrows along the path point in the direction of the path. A loop flip reverses the direction of all arrows along the loop.

Our cluster algorithm performs precisely such operations, with appropriate probabilities. It constructs closed paths consisting of one or several loops without common bonds. All loops in this path are flipped together.

We shall construct the path iteratively, following the direction of the arrows. Let bond b be the latest addition to the path. The arrow on b points to a new vertex v. There are two outgoing arrows at v, and what we need is a unique prescription for continuing the path through v. This is provided by a *breakup* of the vertex v. In addition to the breakup, we have to allow for *freezing* of v. By choosing suitable probabilities for breakup and freezing we shall satisfy detailed balance.

The *breakup* operation is defined by splitting v into two corners, as shown in Fig. 2. At any corner one of the arrows points towards v, while the other one points away from v. Thus we will not allow, e.g., the ul–lr breakup for a vertex in the configuration 3. A "corner flip" is a flip of both arrows. For a given breakup, we only allow the configuration changes resulting from independent corner flips. This preserves the zero divergence condition at v. Notice that a single corner flip transforms a vertex of weight 1 into a vertex of weight e^{-K} and vice versa. Detailed balance is satisfied with the following probabilities for choosing a given breakup:

$$P_{\text{ul-lr}}(i) = \begin{cases} re^K, & i = 1,2, \\ 0, & i = 3,4, \\ r, & i = 5,6, \end{cases} \qquad P_{\text{ll-ur}}(i) = \begin{cases} 0, & i = 1,2, \\ re^K, & i = 3,4, \\ r, & i = 5,6; \end{cases} \tag{2}$$

r is a free parameter for now.

Freezing of a vertex means that its weight must not change. Since there are only two different vertex weights, we introduce two freezing probabilities. They are already determined by the requirement that for a given vertex configuration the sum of freezing and breakup probabilities must be 1:

$$P_{\text{freeze}}(i) = \begin{cases} 1 - re^K, & i = 1,2,3,4, \\ 1 - 2r, & i = 5,6. \end{cases} \tag{3}$$

The range of possible values for r is now obtained by requiring that all probabilities are between 0 and 1:

$$0 \le r \le \min(\tfrac{1}{2}, e^{-K}). \tag{4}$$

Assume now that we have broken or frozen all vertices. Starting from a bond b_0, we proceed to construct a closed path by moving in the arrow direction. As we move from vertex to vertex, we always have a unique way to continue the path. If a vertex is broken, we enter and leave it along the same corner. If the vertex is frozen and of type 1, 2, 3, or 4, we pass through it on a straight line. At such vertices the path may be self-intersecting. Finally, if the latest bond b added to the cluster points to a frozen vertex v of type 5 or 6, the path continues both to the right and to the left of b, i.e., we start a new loop at v. The two loops have to be flipped together. In general, the zero-divergence condition guarantees that all loops will eventually close.

The break-or-freeze decision for all vertices determines a unique partitioning of the lattice into closed paths that can be flipped independently. We choose to perform single cluster updates, i.e., we "grow" a *single path* from a random starting bond b_0, and flip it. The break-or-freeze decision is only needed for the vertices along the path. Thus the computer time for one path is proportional to the length of that path.

It is easy to see that our algorithm is correct. The proof of detailed balance is completely analogous to that for other cluster algorithms [1, 2]. The main ingredient here is that $P_{\text{ul-lr}}$ and $P_{\text{ll-ur}}$ already satisfy detailed balance locally. Furthermore, it is not difficult to see that any two allowed configurations can be connected by a finite number of cluster flips. Thus a finite power of the Markov matrix is ergodic.

How do we choose an optimal value for the parameter r? We have seen that freezing of a vertex of type 5 or 6 forces us to flip two loops together. If we had broken it up instead, we might have been allowed to flip the two loops independently. Thus more freezing leads to larger clusters. We conjecture that the *least possible freezing is optimal.* This is confirmed by numerical tests (see below). From Eq. (3) we then obtain

$$r_{\text{opt}} = \begin{cases} \frac{1}{2}, & K \le K_c, \\ e^{-K}, & K \ge K_c. \end{cases} \tag{5}$$

By maximizing r we also minimize the freezing probability for vertices of type 1, 2, 3, and 4. Notice that if we choose $r = r_{\text{opt}}$, then for $K \le K_c$ vertices of type 5 and 6 are never frozen, so every path consists of a single

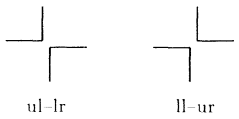

FIG. 2. The two breakups of a vertex: ul–lr (upper left–lower right) and ll–ur (lower left–upper right) .

loop. For $K > K_c$ on the other hand, vertices of type 1, 2, 3, and 4 are never frozen, so we do not continue a path along a straight line through any vertex.

There are some distinct differences between our loop clusters and more conventional spin clusters. For spin clusters, the elementary objects that can be flipped are spins; freezing binds them together into clusters. Our closed loops on the other hand may be viewed as a part of the *boundary* of spin clusters (notice that the boundary of spin clusters may contain loops inside loops). It is reasonable to expect that in typical cases, building a loop cluster will cost less work than for a spin cluster. This is an intrinsic advantage of the loop algorithm.

This can be exemplified nicely for the F model, where a spin-cluster algorithm—the VMR (valleys-to-mountain reflections) algorithm [9]—is also available. At K_c one can see that if we use $r = r_{\rm opt}$, loop clusters are indeed parts of the boundary of VMR spin clusters. Since flipping a loop cluster is not the same as flipping a VMR cluster, we expect the two algorithms to have a different performance. We found (see [9] and below) that in units of clusters, the VMR algorithm is more efficient, but in work units, which are basically units of CPU time, the loop algorithm wins. At $K_c/2$, where the loop clusters are not related [10] to the boundary of VMR clusters, we found the loop algorithm to be more efficient both in units of clusters and in work units, with a larger advantage in the latter.

Performance.—We tested our new algorithm on $L \times L$ square lattices with periodic boundary conditions, both at the transition point K_c and at $\frac{1}{2}K_c$ deep inside the massless phase. We carefully analyzed autocorrelation functions and determined the exponential autocorrelation time τ. At infinite correlation length, *critical slowing down* is quantified by the relation [1]

$$\tau \propto L^z . \tag{6}$$

Local algorithms are slow, with $z \approx 2$. For comparison, we performed runs with a local algorithm that flips arrows around elementary plaquettes with Metropolis probability, and indeed found $z = 2.2(2)$ at $K = K_c$.

In order to make sure that we do observe the slowest mode of the Markov matrix we measured a range of quantities and checked that they exhibit the same τ. As in [9], the slowest mode is strongly coupled to the sublattice energy. The two sublattice energies [9] add up to the total energy. The constraints of the model cause them to be strongly anticorrelated. Within our precision the true value of τ is *not* visible from autocorrelations of the total energy, which decay very quickly. Only for the largest lattices do we see a small hint of a long tail in the autocorrelations. A similar situation occurred in [9], where, when decreasing the statistical errors, the decay governed by the true τ eventually became visible. Note that as a consequence of this situation, the so-called "integrated autocorrelation time" [1] is much smaller than τ, and it would be completely misleading to evaluate the algorithm based only on its values.

We shall quote autocorrelation times τ in units of "sweeps" [1]. We define a sweep such that on average each bond is updated once during a sweep. Thus, if $\tau^{\rm cl}$ is the autocorrelation time in units of clusters, then $\tau = \tau^{\rm cl} \times \langle \text{cluster size} \rangle / 2L^2$. Each of our runs consisted of between 50 000 and 200 000 sweeps. Let us also define $z^{\rm cl}$ by $\tau^{\rm cl} \propto L^{z^{\rm cl}}$, and a cluster size exponent c by $\langle \text{cluster size} \rangle \propto L^c$. We then have

$$z = z^{\rm cl} - (2 - c) . \tag{7}$$

Table I shows the autocorrelation time τ for the optimal choice $r = r_{\rm opt}$. At $K = \frac{1}{2}K_c$, deep inside the massless phase, critical slowing down is almost completely absent. A fit according to Eq. (6) gives $z = 0.19(2)$. The data are also consistent with $z = 0$ and only logarithmic growth. For the cluster size exponent c we obtained $c = 1.446(2)$, which points to the clusters being quite fractal. At the phase transition $K = K_c$ we obtained $z = 0.71(5)$, which is still small. The clusters seem to be less fractal: $c = 1.060(2)$.

We noted above that at $K = K_c$ and for the optimal choice of r, the loop clusters are related to the VMR spin clusters. In [9] we obtained for the VMR algorithm at $K = K_c$ the result $z^{\rm cl} = 1.22(2)$, but we had $c = 1.985(4)$, which left us with $z = 1.20(2)$. In this case it is the smaller dimensionality of the clusters that makes the loop algorithm more efficient.

As mentioned, no critical slowing down is visible for the integrated autocorrelation time of the total energy. At $K = K_c$, $\tau_{\rm int}(E)$ is only 0.80(2) on the largest lattice, and we find $z_{\rm int}(E) \approx 0.20(2)$. At $K = \frac{1}{2}K_c$, $\tau_{\rm int}(E)$ is 1.1(1) on all lattice sizes, so $z_{\rm int}(E)$ is zero.

What happens for nonoptimal values of r? Table II shows our results on the dependence of z on r. z rapidly increases as r moves away from $r_{\rm opt}$. This effect seems to be stronger at $\frac{1}{2}K_c$ than at K_c. We thus see that the optimal value of r indeed produces the best results, as conjectured from our principle of *least possible freezing*.

In the massive phase close to K_c, we expect [10] that $z(K_c)$ will determine the behavior of τ in a similar way as

TABLE I. Exponential autocorrelation time τ at $r = r_{\rm opt}$, and the resulting dynamical critical exponent z.

L	$K = K_c$	$K = \frac{1}{2}K_c$
8	1.8(1)	4.9(4)
16	3.0(2)	5.6(2)
32	4.9(4)	6.2(3)
64	7.2(7)	7.4(3)
128	15.5(1.5)	8.3(2)
256	20.5(2.0)	
z	0.71(5)	0.19(2)

TABLE II. Dependence of the dynamical critical exponent z on the parameter r. We use "$\geq$" where for our lattice sizes τ increases faster than a power of L.

K	r	z
$\frac{1}{2}K_c$	0.500	0.19(2)
$\frac{1}{2}K_c$	0.450	1.90(5)
$\frac{1}{2}K_c$	0.400	$\geq$ 2.6(4)
K_c	0.500	0.71(5)
K_c	0.475	0.77(6)
K_c	0.450	0.99(6)
K_c	0.400	$\geq$ 2.2(1)

in Ref. [9]. To confirm this, a finite size scaling analysis of τ is required.

Generalizations and outlook.—For the sake of clarity we have described our approach in terms of the F model only. It has, however, a much wider range of applicability. We will give a detailed description elsewhere [10]. Here we shall mention only a few highlights.

Our "breakup of vertices" and subsequent path flip *automatically* satisfies the constraints of the F model. General 6- *and* 8-vertex models [7] with arrow flip symmetry have related constraints. By using the framework of Kandel and Domany [2] and the principle of minimal freezing, we can generalize the breakup operation [10] to obtain efficient algorithms for these cases too. Algorithms for more general vertex models can be engineered along the same lines.

Particularly promising is the possibility of *accelerating quantum Monte Carlo simulations* [10, 11]. Quantum spin systems in one and two dimensions can be mapped into vertex models in $1+1$ and $2+1$ dimensions via the Trotter formula and suitable splittings of the Hamiltonian [11]. The simplest example is the spin $\frac{1}{2}$ xxz quantum chain, which is mapped into the 6-vertex model. For higher spins, more complicated vertex models result (e.g., 19-vertex model for spin 1).

For $2+1$ dimensions, different splittings of the Hamiltonian can lead to geometrically quite different situations [10, 11]. We can, e.g., choose between 6-vertex models on a complicated $(2+1)$-dimensional lattice, and models on a bcc lattice with 8 bonds (and a large number of configurations) per vertex. In all these cases, the constraints are of a similar nature as in the F model, and our approach of constructing and updating clusters which are *paths* can be applied in a straightforward way. Notice also that in our approach it is easy to change global properties like the number of world lines or the winding number (see [11]).

Recently we received a paper by Wiese and Ying [12] on a different cluster algorithm for spin $\frac{1}{2}$ quantum spin systems. After mapping to a vertex model similar to the one we refer to, they combine vertices into "block spins" which are then used in a standard spin-cluster construction. This approach restricts the possible updates of the arrows. In our language, their clusters are sets of loops that are frozen together, i.e., that have to be flipped together. For some interesting cases, e.g., the one-dimensional Heisenberg ferromagnet and two-dimensional Heisenberg ferromagnet and antiferromagnet, the additional freezing leads to the problem of *frustration* for the block-spin clusters. We expect our algorithm to perform better in these cases, both because our clusters have less loops and because of the added flexibility offered by the possibility to optimize.

In conclusion, we have presented a new type of cluster algorithm. It flips closed paths of bonds in vertex models. Constraints are automatically satisfied. We have successfully tested our algorithm for the F model and found remarkably small dynamical critical exponents.

There are many promising and straightforward applications of our approach, to other vertex models, and to (1+1)- and (2+1)-dimensional quantum spin systems. Investigations of such systems are in progress.

This work was supported in part by the German–Israeli Foundation for Research and Development (GIF) and by the Basic Research Foundation of the Israel Academy of Sciences and Humanities. We used about 50 hours of Cray YMP processor time. We would like to express our gratitude to the HLRZ at KFA Jülich and to the DOE for providing the necessary computer time.

(a) Electronic address: evertz@scri.fsu.edu
(b) Electronic address: gidi@albert.tau.ac.il
(c) Electronic address: marcu@taunivm.bitnet

[1] For reviews, see, e.g., U. Wolff, in *Lattice '89, Capri, 1989*, edited by N. Cabbibo *et al.* [Nucl. Phys. B (Proc. Suppl.) **17**, 93 (1990)]; A. D. Sokal, in *Lattice '90, Tallahassee, 1990*, edited by U. M. Heller *et al.* [Nucl. Phys. B (Proc. Suppl.) **20**, 55 (1991)].
[2] D. Kandel and E. Domany, Phys. Rev. B **43**, 8539 (1991).
[3] R. H. Swendsen and J. S. Wang, Phys. Rev. Lett. **58**, 86 (1987).
[4] R. C. Brower and P. Tamayo, Phys. Rev. Lett. **62**, 1087 (1989); U. Wolff, Phys. Rev. Lett. **62**, 361 (1989); Nucl. Phys. **B322**, 759 (1989); Phys. Lett. B **228**, 379 (1989).
[5] D. Kandel, R. Ben-Av, and E. Domany, Phys. Rev. Lett. **65**, 941 (1990).
[6] R. Ben-Av, D. Kandel, E. Katznelson, P. Lauwers, and S. Solomon, J. Stat. Phys. **58**, 125 (1990). This cluster algorithm for the three-dimensional Z_2 gauge theory may at first glance bear some resemblance to our loop algorithm, but the similarities are only superficial.
[7] R. J. Baxter, *Exactly Solved Models in Statistical Mechanics* (Academic, New York, 1989).
[8] E. H. Lieb and F. Y. Wu, in *Phase Transitions and Critical Phenomena*, edited by C. Domb and M. S. Green (Academic, New York, 1972), Vol. 1, p. 331.

[9] M. Hasenbusch, G. Lana, M. Marcu, and K. Pinn, Phys. Rev. B **46**, 10472 (1992).

[10] H. G. Evertz and M. Marcu, in Lattice '92, Amsterdam, 1992, edited by J. Smit *et al.* [Nucl. Phys. B (Proc. Suppl.) (to be published)] (Report No. FSU-SCRI-92-165); (to be published).

[11] *Quantum Monte Carlo Methods in Equilibrium and Nonequilibrium Systems*, Proceedings of the Taniguchi Symposium, edited by M. Suzuki (Springer, Berlin, 1987).

[12] U. J. Wiese and H.P. Ying, Phys. Lett. A **168**, 143 (1992).