Vertex Models and Quantum Spin Systems:
A nonlocal approach[*]

Hans Gerd Evertz[1] and Mihai Marcu[2]

[1] Supercomputer Computations Research Institute,
Florida State University, Tallahassee, FL 32306
evertz@scri.fsu.edu

[2] School of Physics and Astronomy,
Raymond and Beverly Sackler Faculty of Exact Sciences,
Tel Aviv University, 69978 Tel Aviv, Israel
marcu@taunivm.bitnet

Abstract

Within a general cluster framework, we discuss the loop-algorithm, a new type of cluster algorithm that reduces critical slowing down in vertex models and in quantum spin systems. We cover the example of the 6-vertex model in detail. For the F-model, we present numerical results that demonstrate the effectiveness of the loop algorithm. We discuss how to modify the original algorithm for some more complicated situations, especially for quantum spin systems in one and two dimensions.

1 Introduction

For Monte Carlo simulations of many interesting physical situations, critical slowing down is a major problem. Standard simulation algorithms employ *local* update procedures like the Metropolis and the heat bath algorithm. With local updates, "information" travels slowly, like a random walk. If the relevant length scale is the correlation length ξ, the number of updates necessary to decorrelate large regions, i.e. the autocorrelation time τ, grows like

$$\tau \propto \xi^z, \tag{1}$$

where $z \approx 2$ for local updates, as suggested by the random walk analogy.

The way out of this problem is to employ *nonlocal* updates. The challenge is to devise algorithms that are nonlocal and still satisfy detailed balance. Multigrid algorithms are one possible approach. In this paper we shall focus on *cluster algorithms*; for a nonexhaustive selection of references see [1, 2, 3, 4, 5, 6].

The first cluster algorithm was invented by Swendsen and Wang (SW) [1] for the case of the Ising spin model. The basic idea is to perform moves that significantly change the Peierls contours characterizing a configuration. As the size of Peierls contours is, typically, anything up to the order of the correlation length, critical slowing down may be eliminated completely or at least partially by this approach. The SW algorithm has been modified and generalized for other spin systems, mostly with two spin interactions [2, 3, 5]. Notice that for these systems clusters are connected regions of spins, with the same dimensionality as the underlying lattice. A few generalizations along different lines were also done [4, 6].

Recently [7, 8, 9] we introduced *cluster algorithms for vertex models and quantum spin systems*, which are the first cluster algorithms for models with constraints. While [7] is an adaptation of the valleys-to mountains-reflections (VMR) algorithm [5], originally devised for solid-on-solid models, the *loop algorithm* introduced in [8, 9] does not resemble any existing scheme.

In this paper we discuss the loop algorithm in detail. In vertex models [10] 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. Our scheme is devised such as to take into account the constraints automatically, and to allow a simple way to construct the clusters. The clusters here are not connected regions of spins, but instead *closed paths* (loops) of bonds.

In what follows we first comment on the SW algorithm for spin systems. Then we discuss the general cluster formalism of Kandel and Domany [4], treating the SW algorithm as an example. Next we define the 6-vertex model, and, as a special case, the F model. We then introduce the loop algorithm, and show how to formulate it for the complete 6-vertex model. The optimization of the algorithm is discussed in a separate section. For the special case of the F-model, we describe how the algorithm particularizes, and then we present our very successful numerical tests of the loop algorithm. We also show how to apply the loop algorithm to simulations of one and two dimensional quantum spin systems [11], like e.g. the xxz quantum spin chain, and the spin $\frac{1}{2}$ Heisenberg antiferromagnet and ferromagnet in two dimensions. Finally we comment on further generalizations and applications.

2 Some comments on the Swendsen-Wang algorithm

Here we present a somewhat unusual viewpoint on the SW algorithm, in order to better understand what is new in the loop algorithm. Let us look at a spin system, like the Ising model, with variables s_x living on sites x, and a nearest neighbor Hamiltonian $H(s_x, s_y)$. The partition function of the model to be simulated is $Z = \sum_s \exp\left(-\sum_{<xy>} H(s_x, s_y)\right)$, where $< xy >$ is a pair of sites.

Consider update proposals (flips) $s_x \rightarrow s'_x$, such that $H(s_x, s_y) = H(s'_x, s'_y)$. Then it follows that updating at the same time all spins in some "cluster" of spins, we will only change the value of the Hamiltonian at the *boundary* of that cluster, not inside it.

In order to satisfy detailed balance, we have to choose clusters with an appropriate probability. The SW algorithm amounts to making a *Metropolis* decision for each bond $< xy >$, namely whether to accept the change in H from $H(s_x, s_y)$ to $H(s'_x, s_y)$. If accepted, a bond is called "deleted", otherwise it is called "frozen". Clusters are then sets of sites connected by frozen bonds. Note that if deleted, a bond *may* be at the boundary of a cluster, but need not.

Finally, an update is performed by finding all the clusters in a given configuration and then flipping each cluster with 50% probability [1]. In Wolff's single cluster variant [3, 2], which is dynamically more favourable, we construct one cluster starting from a randomly chosen initial site, and then flip it with 100% probability.

A technical remark: The Swendsen-Wang algorithm can be vectorized and parallelized [12]. The difficult task is to identify the clusters, which is the same task as e.g. image component labeling. For the single cluster variant, vectorization is the most efficient approach [13].

3 The Kandel - Domany framework.

Cluster algorithms are conveniently described in the general framework of Kandel and Domany [4]. Let us consider the partition function

$$Z = \sum_u \exp\left(-V(u)\right), \tag{2}$$

where u are the configurations to be summed over. We shall call the function V the "interaction". Let us also define a *set* of new interactions $\overline{V}_i$ (the index i numbers the modified interactions). Assume that during a Monte Carlo simulation we arrived at a given configuration u. We choose a new configuration in a two step procedure. The first step is to replace V with one of the $\overline{V}_i$. For a given i, $\overline{V}_i$ is chosen with probability $p_i(u)$. The $p_i(u)$ satisfy:

$$\begin{aligned} p_i(u) &= exp(V(u) - \overline{V}_i(u) + c_i), \\ \sum_i p_i(u) &= 1 \,, \end{aligned} \tag{3}$$

where c_i are constants independent of u. The second step is to update the configuration by employing a procedure that satisfies detailed balance for the chosen $\overline{V}_i$. The combined procedure satisfies detailed balance for the original interaction V [4].

In many cases, the interaction V is a sum over local functions H^c, where c typically are cells of the lattice (like bonds in the spin system case, sites for vertex models, plaquettes for gauge theories). More generally, H^c can contain part (or all) of the interactions in some neighborhood of the cell c. We can choose the modified interactions $\overline{H^c}_{i^c}$ for each H^c independently (i^c numbers the possible new interactions for the cell c). The probabilities for this choice obey eq. (3), with V replaced by H^c, and i by i^c. The total modified interaction $\overline{V}_i$ will then be the sum of all the $\overline{H^c}_{i^c}$ (i is now a multiindex). When clear from the context, we shall drop the index c.

Take the Ising model as an example. The configuration u is comprised of spins $s_x = \pm 1$, the cells where the interaction is localized are bonds $< xy >$, and $V(u) = -J \sum_{<xy>} s_x s_y$. We choose the bonds as our cells c, so we can perform the first step of the Kandel-Domany procedure separately for each bond. The original cell interaction is

$$H^{<xy>}(s) = -J s_x s_y . \tag{4}$$

Now, let us define two new bond interactions; the first ($i^{<xy>} = 1$) is called "freeze", the second ($i^{<xy>} = 2$) is called "delete":

$$\overline{H^{<xy>}}_{\text{freeze}}(s) = \begin{cases} 0, & s_x = s_y \\ \infty, & else \end{cases} ; \qquad \overline{H^{<xy>}}_{\text{delete}}(s) = 0 . \tag{5}$$

Then from (3) we obtain the Swendsen-Wang probabilities if we choose the constants c_i that minimize freezing:

$$\begin{aligned} p^{<xy>}_{\text{delete}}(s) &= \exp\left(-J(s_x s_y + 1)\right) = \min\left(1, \exp\left(-2J s_x s_y\right)\right) , \\ p^{<xy>}_{\text{freeze}}(s) &= 1 - p^{<xy>}_{\text{delete}}(s) , \end{aligned} \tag{6}$$

which is just the Swendsen-Wang probability.

4 The 6-Vertex Model

The 6-vertex model [10] is defined on a square lattice. On each bond there is an Ising-like variable that is usually represented as an arrow. For example, arrow up or right means $+1$, arrow down or left means -1. At each vertex we have the constraint that there are two incoming and two outgoing arrows. In fig. 1 we show the six possible configurations at a vertex, numbered as in [10]. The statistical weight of a configuration is given by the product over all vertices of the vertex weights $\rho(u)$. Thus a priori, for each vertex there are 6 possible weights $\rho(u)$, $u = 1, ..., 6$. We take the weights to be symmetric under reversal of all arrows. Thus, in standard notation [10], we have:

$$\begin{aligned} \rho(1) &= \rho(2) = a , \\ \rho(3) &= \rho(4) = b , \\ \rho(5) &= \rho(6) = c . \end{aligned} \tag{7}$$

The 6-vertex model has two types of phase transitions: of Kosterlitz-Thouless type and of KDP type [10]. A submodel exhibiting the former is the F model, defined by $c = 1$, $a = b = \exp(-K)$, $K \geq 0$. For the latter transition an example is the KDP model itself, defined by $a = 1$, $b = c = \exp(-K)$, $K \geq 0$.
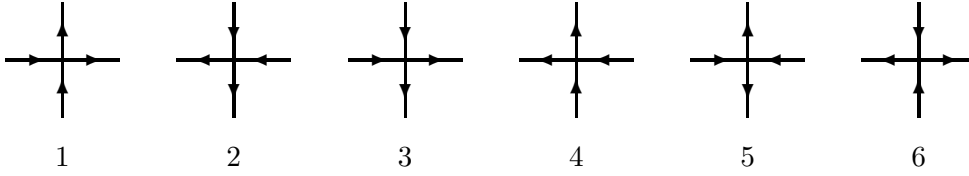
3

Figure 1: The six vertex configurations, $u = 1, ..., 6$ (using the standard conventions of [10]).

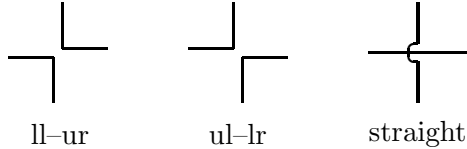ll–ur   ul–lr   straight

Figure 2: The three break-ups of a vertex: ll–ur (lower-left–upper-right), ul–lr (upper-left–lower-right), and straight.

5   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, non-branching (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 the 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 *break-up* of the vertex v. In addition to the break-up, we have to allow for *freezing* of v. By choosing suitable probabilities for break-up and freezing we shall satisfy detailed balance.

The *break-up* operation is defined by splitting v into two pieces, as shown in fig. 2. The two pieces are either two corners or two straight lines. On each piece, 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 break-up for a vertex in the configuration 3. If we break up v, the possible new configurations are obtained by flipping (i.e., reversing both arrows of) the two pieces independently. On the other hand, if we freeze v, the only possible configuration change is to flip all four arrows.

The break-up and freeze probabilities are conveniently described within the Kandel Domany framework. It is sufficient to give them for one vertex, which is in the current configuration u. We define 6 new interactions (weight functions) ρ_i, $i = 1, ..., 6$, corre-

i	action	$\rho_i(\tilde{u})$	$p_i(u)$
1	freeze 1,2	1, $\rho(\tilde{u})=a$ 0, else	q_1/a, $\rho(u)=a$ 0, else
2	freeze 3,4	1, $\rho(\tilde{u})=b$ 0, else	q_2/b, $\rho(u)=b$ 0, else
3	freeze 5,6	1, $\rho(\tilde{u})=c$ 0, else	q_3/c, $\rho(u)=c$ 0, else
4	ll–ur	0, $\rho(\tilde{u})=a$ 1, else	0, $\rho(u)=a$ $q_4/\rho(u)$, else
5	ul–lr	0, $\rho(\tilde{u})=b$ 1, else	0, $\rho(u)=b$ $q_5/\rho(u)$, else
6	straight	0, $\rho(\tilde{u})=c$ 1, else	0, $\rho(u)=c$ $q_6/\rho(u)$, else

Table 1: The new interactions $\rho_i(\tilde{u})$ and the probabilities $p_i(u)$ to choose them at a vertex in current configuration u. See eq. (8).

sponding to specific break-up and freeze operations (the labelling of the new interactions is completely arbitrary, and the fact that we have six of them is just a coincidence). For each vertex in configuration u, we replace with probability $p_i(u)$ the original interaction ρ by the new interaction ρ_i. Equation 3 , i.e. detailed balance and the proper normalization of probabilities, require that for every u

$$p_i(u) = q_i \frac{\rho_i(u)}{\rho(u)} , \quad \sum_i p_i(u) = 1 , \tag{8}$$

where $q_i = \exp{(-c_i)} \geq 0$ are parameters.

As discussed in [4] (see also table 1), *freezing* is described by introducing one new interaction for each different value of $\rho(u)$. For example, to freeze the value a, we choose the interaction ρ_1 to be $\rho_1(\tilde{u}) = 1$ if $\rho(\tilde{u}) = a$, and $\rho_1(\tilde{u}) = 0$ otherwise. In other words, when ρ_1 is chosen, transitions between $\tilde{u} = 1$ and $\tilde{u} = 2$ cost nothing, whereas the vertex configurations 3, 4, 5, and 6 are then not allowed. Notice that we denote by u the current configuration, and by $\tilde{u}$ the argument of the function ρ_i.

Each *break-up* is also described by one new interaction. As an example take the ul–lr break-up. It is given by the new interaction number five, with $\rho_5(\tilde{u}) = 1$ if $\rho(\tilde{u}) = a$ or c, and $\rho_5(\tilde{u}) = 0$ if $\rho(\tilde{u}) = b$. In other words, with the new interaction ρ_5, transitions between 1, 2, 5 and 6 cost nothing, while the vertex configurations 3 and 4 are not allowed. This corresponds precisely to allowing independent corner flips in a ul–lr break-up (see figs. 1,2).

5

Table 1 gives the full list of new weights $\rho_i(\tilde{u})$ and probabilities $p_i(u)$ to choose them in current configuration u. From (8) we also obtain:

$$q_1 + q_5 + q_6 = a \, ,$$
$$q_2 + q_4 + q_6 = b \, , \qquad (9)$$
$$q_3 + q_4 + q_5 = c \, .$$

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. At broken vertices the path enters the vertex through one bond and leaves it through another. If the last bond b added to the cluster points to a frozen vertex v, the path bifurcates in the directions of the two outgoing arrows of v. One of these directions can be considered as belonging to the loop we came from, the other one as belonging to a new loop. Since we also have to flip the second incoming arrow of v, we are assured that this new loop also closes. The two loops have to be flipped together. In general, the zero-divergence condition guarantees that all loops will eventually close.

We have now finished describing the procedure for constructing clusters. In order to specify the algorithm completely, we must choose values for the constants q_i, and decide how the clusters are flipped. The former problem is of utmost importance, and it is the object of the next chapter. For the cluster flips, we may use both the Swendsen-Wang procedure and the Wolff single cluster flip [3]. We choose the latter, 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, so the computer time for one path is proportional to the length of that path.

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.

6   Optimization of free parameters

We have seen that freezing forces loops to be flipped together. Previous experience with cluster algorithms suggests that it is advantageous to be able to flip loops independently, as far as possible. We therefore introduce the principle of *minimal freezing* as a guide for choosing the constants q_i: we shall minimize the freezing probabilities, given the constraints (9) and $q_i \geq 0$. In the next chapter we will show that for the case of the F model, optimization by minimal freezing does indeed minimize critical slowing down. Here we discuss optimization for the 4 phases of the 6-vertex model, usually denoted by capital roman numerals [10].

Let us first look at phase IV, where $c > a + b$. To minimize the freezing of weight c, we have to minimize q_3. From (9), $q_3 = c - a - b + q_1 + q_2 + 2q_6$. With $q_i \geq 0$ this implies $q_{3,\text{min}} = c - a - b$. The minimal value of q_3 can only be chosen if *at the same time* we set $q_1 = q_2 = 0$, i.e. minimize (in this case do not allow for) the freezing of the smaller weights a and b. The optimized parameters for phase IV are then:

$$q_1 = 0, \quad q_2 = 0, \quad q_3 = c - a - b,$$
$$q_4 = b, \quad q_5 = a, \quad q_6 = 0 \,. \tag{10}$$

In phase I the situation is technically similar. Here $a > b + c$, and we minimize freezing with $q_1 = a - b - c$ and $q_2 = q_3 = 0$. The same holds for phase II, $b > a + c$, where we obtain minimal freezing for $q_2 = b - a - c$ and $q_1 = q_3 = 0$.

Phase III (the massless phase) is characterized by $a, b, c < \frac{1}{2}(a + b + c)$. Here we can set all freezing probabilities to zero. Thus,

$$q_1 = 0, \quad 2q_4 = b + c - a \,,$$
$$q_2 = 0, \quad 2q_5 = c + a - b \,, \tag{11}$$
$$q_3 = 0, \quad 2q_6 = a + b - c \,.$$

7 Case of the F model

The F model is obtained from (10) and (11) as the special case $a = b = \exp(-K) \leq 1$, $c = 1$. It has a Kosterlitz-Thouless transition at $K_c = \ln 2$, with a massless phase for $K \leq K_c$.

How do we choose the parameters q_i here ? Symmetry $a = b$ implies $q_1 = q_2$ and $q_4 = q_5$. We can eliminate freezing of vertices $1, 2, 3, 4$, by setting $q_1 = q_2 = 0$. In (9) this leaves one parameter, q_3:

$$2q_4 = 1 - q_3 \,,$$
$$2q_6 = \exp(-K) + q_3 - 1 \,. \tag{12}$$

In the massless phase, we can minimize freezing by also setting $q_3 = 0$. In the massive phase, $q_6 \geq 0$ limits q_3. Thus

$$q_{3,\text{min}} = \begin{cases} 1 - 2\exp(-K), & K \geq K_c, \\ 0, & K \leq K_c. \end{cases} \tag{13}$$

Notice that since $a = b$ in the F model, the straight break-up, the freezing of a, and that of b are operationally the same thing.

If we choose $q_3 = q_{3,\text{min}}$, then for $K \leq K_c$ vertices of type 5 and 6 are never frozen, which has as a consequence that every path consists of a single loop. This loop may intersect itself, like in the drawing of the figure "8". 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. As $K \to \infty$ (temperature goes to zero), most vertices are of type 5 or 6, and they are almost always frozen. Thus the algorithm basically flips between the two degenerate ground states.

For the F model we also have a spin-cluster algorithm – the VMR algorithm [7]. At $K = K_c$ and for $q_3 = q_{3,\mathrm{min}}$, we have a situation where the loop-clusters form 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 different performances. We found (see [7] and the next section) 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 at all related 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.

8 F-model: Performance of the loop algorithm

We tested the loop algorithm on $L \times L$ square lattices with periodic boundary conditions at two values of K: at the transition point K_c, and at $\frac{1}{2}K_c$, which is 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), $\tau \propto L^z$.

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 [7], it turned out that one can use quantities defined on a sublattice in order to couple strongly to the slowest mode. Specifically, we wrote the energy as a sum over two sublattice quantities. We shall present more details of this phenomenon elsewhere. Let us however note here that for the total energy, the true value of τ was not visible within our precision except for a weak hint of a long tail in the autocorrelations on the largest lattices we considered. Note that as a consequence of this situation, the so-called "integrated autocorrelation time" [3] 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" [3], defined such that on the average each bond is updated once during a sweep. Thus, if τ^{cl} is the autocorrelation time in units of clusters, then $\tau = \tau^{\mathrm{cl}} \times \langle \mathrm{cluster\ size} \rangle / (2L^2)$. Each of our runs consisted of between 50000 and 200000 sweeps. Let us also define z^{cl} by $\tau^{\mathrm{cl}} \propto L^{z^{\mathrm{cl}}}$, and a cluster size exponent c by $\langle \mathrm{cluster\ size} \rangle \propto L^c$. We then have:

$$z = z^{\mathrm{cl}} - (2 - c). \tag{14}$$

Table 2 shows the autocorrelation time τ for the optimal choice $q_3 = q_{3,\mathrm{min}}$. At $K = \frac{1}{2}K_c$, deep inside the massless phase, critical slowing down is almost completely absent. A fit according to eq. 1 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 (notice that $z^{\mathrm{cl}} = 0.74(2)$). At the phase

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 2: Exponential autocorrelation time τ at $q_3 = q_{3,\mathrm{min}}$, and the resulting dynamical critical exponent z.

K	q_3	z
$\frac{1}{2}K_c$	0	0.19(2)
$\frac{1}{2}K_c$	0.10	1.90(5)
$\frac{1}{2}K_c$	0.20	$\geq 2.6(4)$
K_c	0	0.71(5)
K_c	0.05	0.77(6)
K_c	0.10	0.99(6)
K_c	0.20	$\geq 2.2(1)$

Table 3: Dependence of the dynamical critical exponent z on the parameter q_3. We use "$\geq$" where for our lattice sizes τ increases faster than a power of L.

transition $K = K_c$ we obtained $z = 0.71(5)$, which is still small. The clusters seem to be less fractal: $c = 1.060(2)$, so that $z^{\mathrm{cl}} = 1.65(5)$.

We noted above that at $K = K_c$ and for the optimal choice of q_3, the loop-clusters are related to the VMR spin-clusters. In [7] we obtained for the VMR algorithm at $K = K_c$ the result $z^{\mathrm{cl}} = 1.22(2)$, but we had $c = 1.985(4)$, which left us with $z = 1.20(2)$. In this case, although in units of clusters the VMR algorithm is more efficient, the smaller dimensionality of the loop-clusters more than make up for this, and in CPU time the loop algorithm is more efficient.

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

What happens for non-minimal values of q_3? Table 3 shows our results on the dependence of z on q_3. z rapidly increases as q_3 moves away from $q_{3,\mathrm{min}}$. This effect seems to be stronger at $\frac{1}{2}K_c$ than at K_c. We thus see that the optimal value of q_3 indeed produces the best results, as conjectured from our principle of *least possible freezing*.

In the massive phase close to K_c, we expect that $z(K_c)$ will determine the behaviour of τ in a similar way as in ref. [7]. To confirm this, a finite size scaling analysis of τ is required. In order to complete the study of the loop algorithm's performance, we should also investigate it at the KDV transition.

In summary, the loop algorithm strongly reduces critical slowing down, from $z = 2.2(2)$ for the Metropolis algorithm, down to $z = 0.71(5)$ at K_c and $z = 0.19(2)$ at $\frac{1}{2}K_c$ deep inside the massless phase. For the integrated autocorrelation time of the total energy, no critical slowing down is visible in either case.

9 Quantum Spin Systems

Particularly promising is the possibility of *accelerating Quantum Monte Carlo simulations*, since 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 directly into the 6-vertex model. For higher spins, more complicated vertex models result (e.g. 19-vertex model for spin one).

For $(2 + 1)$ dimensions, different splittings of the Hamiltonian lead to quite different vertex models, in particular on quite different lattice types [11]. For example, in the case of spin $\frac{1}{2}$ we can choose between 6-vertex models on a quite complicated $2 + 1$ dimensional lattice, and models on a *bcc* lattice, with 8 bonds and a large number of configurations per vertex.

For the simulation of the *2-dimensional Heisenberg antiferromagnet and ferromagnet* using the former splitting, all relevant formulas have been worked out in the present paper. Actually, the low temperature properties of the antiferromagnet have recently been investigated by Wiese and Ying [14] *using our algorithm*. Their calculation is, in our opinion, the first high quality verification of the magnon picture for the low lying excitations. In particular, this excludes to a much higher degree of confidence than before the speculation (some years ago widespread) that the model had a nonzero mass gap.

Notice that, similarly to other cluster algorithms [3], it is straightforward to define improved observables. The investigation [14] in fact uses them.

Let us also remark that the loop algorithm can easily change global properties like the number of world lines or the winding number (see [11]). Thus it is well suited for simulations in the grand canonical ensemble. Last, but not least, the loop algorithm also opens up a new avenue for taming the notorious fermion sign problem [15].

10 Conclusions

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. In particular, we believe that our generalizations of the freeze-delete scheme can be adapted for other models like the 8-vertex model.

Already, the loop algorithm has found important applications in the study of the 2-dimensional Heisenberg antiferromagnet.

Acknowledgements

References

[1] R. H. Swendsen and J. S. Wang, Phys. Rev. Lett. 58 (1987) 86.

[2] R. C. Brower and P. Tamayo, Phys. Rev. Lett. 62 (1989) 1087;
U. Wolff, Phys. Rev. Lett. 62 (1989) 361, Nucl. Phys. B322 (1989) 759, and Phys. Lett. 228B (1989) 379.

[3] For reviews, see e.g.:
U. Wolff, in *Lattice '89*, Capri 1989, N. Cabbibo et al., editors, Nucl. Phys. B (Proc. Suppl.) 17 (1990) 93;
A. D. Sokal, in *Lattice '90*, Tallahassee 1990, U. M. Heller et al., editors, Nucl. Phys. B (Proc. Suppl.) 20 (1991) 55.

[4] D. Kandel and E. Domany, Phys. Rev. B43 (1991) 8539.

[5] H. G. Evertz, M. Hasenbusch, M. Marcu, K. Pinn and S. Solomon, Phys. Lett. 254B (1991) 185, and in *Workshop on Fermion Algorithms*, Jülich 1991, H. J. Herrmann and F. Karsch editors, Int. J. Mod. Phys. C3 (1992) 235.

[6] R. Ben-Av, D. Kandel, E. Katznelson, P. Lauwers and S. Solomon, J. Stat. Phys. 58 (1990) 125. This cluster algorithm for the 3-dimensional Z_2 gauge theory may at first glance bear some resemblance to our loop algorithm, but the similarities are only superficial.

[7] M. Hasenbusch, G. Lana, M. Marcu and K. Pinn, *Cluster algorithm for a solid-on-solid model with constraints,* Phys. Rev. B46 (1992) 10472.

[8] H.G. Evertz, G. Lana and M. Marcu, Phys. Rev. Lett. 70 (1993) 875.

[9] H.G. Evertz and M. Marcu, in *Lattice 92*, Amsterdam 1992, ed. J. Smit et al., Nucl. Phys. B (Proc. Suppl.) 30 (1993) 277.

[10] E. H. Lieb, Phys. Rev. Lett. 18 (1967) 1046;
E. H. Lieb and F. Y. Wu, *Two-dimensional Ferroelectric Models*, in *Phase Transitions and Critical Phenomena* Vol. 1, C. Domb and M. S. Green, editors, (Academic, 1972) p. 331;
R. J. Baxter, *Exactly Solved Models in Statistical Mechanics* (Academic, 1989).

[11] For Quantum Monte Carlo simulations see:
M. Suzuki editor, *Quantum Monte Carlo methods in equilibrium and nonequilibrium systems*, Taniguchi symposium, Springer (1987).

[12] See the overview in [13].

[13] H.G. Evertz, J. Stat. Phys. 70 (1993) 1075.

[14] U.J. Wiese and H.P. Ying, Bern preprint;
bulletin board cond-mat/9212006.

[15] H.G. Evertz, in preparation.