OVERCOMING CRITICAL SLOWING DOWN IN QUANTUM MONTE CARLO SIMULATIONS[1]

Hans Gerd Evertz[1,2] and Mihai Marcu[3]

[1] Supercomputer Computations Research Institute,
Florida State University,
Tallahassee, FL 32306-4052, USA
evertz@scri.fsu.edu

[2] Department of Physics and Astronomy
and Center for Simulational Physics
University of Georgia,
Athens, GA 30602, USA

[3] Racah Institute of Physics,
Hebrew University,
91904 Jerusalem, Israel
marcu@vax.huji.ac.il

Abstract

The classical d+1-dimensional spin systems used for the simulation of quantum spin systems in d dimensions are, quite generally, vertex models. Standard simulation methods for such models strongly suffer from critical slowing down. Recently, we developed the *loop algorithm*, a new type of cluster algorithm that to a large extent overcomes critical slowing down for vertex models. We present the basic ideas on the example of the F model, a special case of the 6-vertex model. Numerical results clearly demonstrate the effectiveness of the loop algorithm. Then, using the framework for cluster algorithms developed by Kandel and Domany, we explain how to adapt our algorithm to the cases of the 6-vertex model and the 8-vertex model, which are relevant for spin $\frac{1}{2}$ systems. The techniqes presented here can be applied without modification to 2-dimensional spin $\frac{1}{2}$ systems, provided that in the Suzuki-Trotter formula the Hamiltonian is broken up into 4 sums of link terms. Generalizations to more complicated situations (higher spins, different uses of the Suzuki-Trotter formula) are, at least in principle, straightforward.

[1]PUBLISHED IN "QUANTUM MONTE CARLO METHODS IN CONDENSED MATTER PHYSICS", ED. M. SUZUKI, WORLD SCIENTIFIC, 1993, P. 65.

1 Introduction

Simulations of quantum spin systems are based on mapping the problem to a sequence of simulations of an increasingly anisotropic classical spin system in one more dimension [1]. For example, the spin $\frac{1}{2}$ xxz chain is mapped onto a model with Ising-like spin variables, 4-spin interaction, and constraints. This model is identical to the 6-vertex model, when using eigenstates of S^z as complete sets of intermediate states, or to the 8-vertex model, when using eigenstates of S^x.

For simulations of many interesting physical situations, critical slowing down (CSD) is a major problem. Standard simulation algorithms employ *local* update procedures like e.g. the Metropolis and the heat bath algorithm. With local updates, "information travels slowly", like a random walk [2]. 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 dynamical critical exponent z is the quantitative measure of CSD.

A possible way out of this difficulty is to employ *nonlocal* updates, which decorrelate a configuration much more quickly. This is a constructive approach, since up to now no general recipe is known for devising efficient nonlocal moves. The hope that multigrid algorithms would be such a general procedure has unfortunately not been fulfilled. However, in recent years *cluster algorithms* have been successful in a variety of instances [3, 4, 5, 6, 7, 8, 9] (this is a nonexhaustive list of references).

The first cluster algorithm was invented by Swendsen and Wang (SW) [3] for the case of the Ising 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 strongly reduced or even completely eliminated by this approach. The SW algorithm has been modified and generalized for other spin systems, mostly with two-spin interactions [4, 5, 7]. 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 [6, 8, 9].

Recently [10, 11, 12] we introduced *cluster algorithms for vertex models and quantum spin systems*, which are the first cluster algorithms for models with constraints. While [10] is an adaptation of the valleys-to-mountains-reflections (VMR) algorithm [7], originally devised for solid-on-solid models, the *loop algorithm* introduced in [11, 12] does not resemble any existing scheme.

In vertex models the dynamical variables are localized on bonds, and the interaction, usually defined by giving the Boltzmann weights, is between all bonds meeting at a vertex. The possible bond variable values at a vertex are subject to constraints. The statistical weight of a configuration is given by the product over all vertices of the vertex weights.

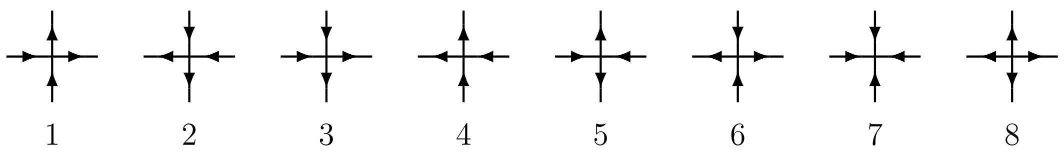
Figure 1: The 8 vertex configurations, $u = 1, ..., 8$, using the notations of [13, 14].

Our scheme is devised such as to take into account the constraints automatically, and to allow a simple way to construct clusters. 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 *break-up of a vertex.*

In this paper we discuss the loop algorithm in detail for the cases of the 6-vertex and the 8-vertex model [13, 14]. The 8-vertex model 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 is an even number of incoming arrows (and consequenly an even number of outgoing ones). In fig. 1 we show the 8 possible configurations u at a vertex, numbered as in [13, 14]. The vertex weights $\rho(u)$ are symmetric under reversal of all arrows; in standard notation [13, 14], we have:

$$\begin{aligned} \rho(1) = \rho(2) = a \ , \\ \rho(3) = \rho(4) = b \ , \\ \rho(5) = \rho(6) = c \ , \\ \rho(7) = \rho(8) = d \ . \end{aligned} \tag{2}$$

The 6-vertex model is defined by $d = 0$. That means that only those 6 vertex configurations are allowed, for which there are exactly two incoming and two outgoing arrows.

The 6-vertex model has two types of phase transitions: Kosterlitz-Thouless (KT) and KDP [13]. For the xxz chain at zero temperature, the former corresponds to the transition at the Heisenberg antiferromagnet point, while the latter corresponds to the transition at the Heisenberg ferromagnet. A submodel exhibiting the KT transition is the F model, defined by $c = 1$, $a = b = \exp(-K)$, $K \geq 0$. The coupling K plays the role of inverse temperature. The KT transition is at $K_c = \ln 2$. The correlation length is finite for $K > K_c$ and infinite for $K \leq K_c$. For the KDP transition, an example is the KDP model itself, defined by $a = 1$, $b = c = \exp(-K)$, $K \geq 0$.

In section 2 we shall describe the loop algorithm on the example of the F model. Besides describing the break-up of a vertex operation and explaining how clusters are constructed, we shall employ the principle of *minimal freezing* in order to optimize the algorithm with respect to a free parameter. In section 3 we analyze the performance of the loop algorithm for the F model. We investigate the exponential autocorrelation times at $K = K_c$ and at $K = K_c/2$. It turns out that the above mentioned optimization is crucial for reducing CSD. In the case of the optimal algorithm, we find a dynamical critical exponent of $z(K_c) = 0.71(5)$ and $z(K_c/2) = 0.19(2)$. In section 4 we review the general formalism for cluster algorithms developed by Kandel and Domany [6], which we then use in section 5 to formulate the loop algorithm for the general arrow flip symmetric 6-vertex model. For this case too, we show how to find the optimal algorithm by minimizing freezing. In section 6 we develop the loop algorithm for the 8-vertex model. As opposed to the 6-vertex model, it is no longer possible to obtain a unique algorithm by minimizing freezing. We propose a way to overcome this problem. For further generalizations of our algorithm, this is an essential issue. While the discussion of the F model and the 6-vertex model was published before [11, 12], the algorithm for the 8-vertex model is published here for the first time. We present our conclusions in section 7. In particular, we briefly discuss the successful use of the loop algorithm for the study of the the spin $\frac{1}{2}$ Heisenberg antiferromagnet in two dimensions [15].

2 The Loop Algorithm for the F Model

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 loop or of 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 *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 corners, as shown in fig. 2. We shall label the two possible break-ups of a vertex by ul–lr (upper-left–lower-right) and ll–ur (lower-left–upper-right). 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 break-up for a vertex in the configuration 3. A "corner flip" is a flip of both arrows. For a given break-up, 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

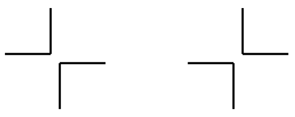

Figure 2: The two break-ups of a vertex used for the F model algorithm.

given break-up (these probabilities depend on the current vertex configuration u):

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

Here 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 break-up probabilities must be one:

$$p_{\text{freeze}}(u) = \begin{cases} 1 - r\mathrm{e}^K & u = 1,2,3,4 \\ 1 - 2r & u = 5,6 \end{cases} . \tag{4}$$

Notice that if we freeze a vertex of type 1, 2, 3 or 4, and we choose to flip the incoming arrow on the bond b, then we must also flip the outgoing arrow that lies on the straight-line continuation of b. We may in addition flip the remaining two arrows, but we do not have to. If on the other hand we freeze a vertex of type 5 or 6, and we choose to flip one of the incoming arrows, then we have to flip all remaining three arrows.

The range of possible values for r is obtained by requiring that all probabilities are between zero and one:

$$0 \le r \le \min(\frac{1}{2}, \mathrm{e}^{-K}) . \tag{5}$$

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 (like e.g. at the center of an "8"). 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. One of these directions can be considered as belonging to the loop we came from, the other one as belonging to a new loop, which also contains the second incoming arrow of v. The two loops have to be flipped together. Actually, 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 [5, 6]. The simplest formal proof is to show that the algorithm fits into the framework of Kandel and Domany [6]; we shall do this later, for the more general case of the 6-vertex model. 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 up the vertex 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. (4) we then obtain

$$r_{\text{opt}} = \begin{cases} \frac{1}{2} & K \leq K_c \\ \mathrm{e}^{-K} & K \geq K_c \end{cases} . \tag{6}$$

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 $K \leq K_c$ vertices of type 5 and 6 are never frozen, so every path consists of a single 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.

The last remark can be exemplified nicely for the F model, where a spin-cluster algorithm – the VMR algorithm [10] – is also available. At K_c one can see that if we use $r = r_{\text{opt}}$, 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 different performance. We found (see [10] 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 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. More details on the comparison between the VMR algorithm and the loop algorithm will be published elsewhere.

3 Numerical Study of the Algorithm's Performance

We tested our new algorithm on $L \times L$ square lattices with periodic boundary conditions[2], 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 [5]

$$\tau \propto L^z\,. \tag{7}$$

Local algorithms are slow, with $z \approx 2$. To be on the safe side, 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 whole range of quantities and checked that they exhibit the same τ. As in [10], the slowest mode is strongly coupled to the sublattice energy[3]. The two sublattice energies [10] 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 [10], 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" [5] 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" [5]. 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

$$\tau = \tau^{\rm cl}\, \frac{<\text{cluster size}>}{2L^2}\,. \tag{8}$$

Each of our runs consisted of between 50000 and 200000 sweeps. Let us also define the exponent $z^{\rm cl}$ by $\tau^{\rm cl} \propto L^{z^{\rm cl}}$, and a cluster size exponent c by $<\text{cluster size}> \propto L^c$. We then have:

$$z = z^{\rm cl} - (2 - c)\,. \tag{9}$$

[2]In order to make contact with studies of the related BCSOS model [10], we did not allow loops that wind around the lattice. In [16] we allowed such loops and found somewhat smaller autocorrelation times.

[3]The sublattice energy is easily defined in the equivalent BCSOS model [10]. In vertex language, it is defined, up to irrelevant additive and multiplicative constants, as follows. Assign value $+1$ to arrows pointing in positive Cartesian directions, -1 otherwise. Label all vertices black or red in checkerboard fashion. Break black vertices ul–lr and red ones ll–ur (see fig. 2). Multiply the two arrow values at each resulting corner with each other, and, for black vertices only, with -1. Add all products. The other sublattice energy is obtained by interchanging black with red in the checkerboard assignment.

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 1: Autocorrelation time τ at $r = r_{\text{opt}}$, and the resulting dynamical exponent z.

Table 1 shows the autocorrelation time τ for the optimal choice $r = r_{\text{opt}}$. At $K = \frac{1}{2}K_c$, deep inside the massless phase, critical slowing down is almost completely absent. A fit according to eq. 7 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 [10] we obtained for the VMR algorithm at $K = K_c$ the result $z^{\text{cl}} = 1.22(2)$, 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 make the loop algorithm more efficient.

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

What happens for non-optimal values of r? Table 2 shows our results on the dependence of z on r. z rapidly increases as r moves away from r_{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 that $z(K_c)$ will determine the behaviour of τ in a similar way as in ref. [10]. To confirm this, a finite size scaling analysis of τ for $K < K_c$ is required.

4 The Kandel-Domany framework

Cluster algorithms are conveniently described in the general framework of Kandel and Domany [6], which guarantees that we are using a correct Markov process. More important than convenience is the fact that this framework allows for a lot of flexibility

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)

Table 2: 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.

in defining new algorithms.

Let us consider the system defined by the partition function

$$Z = \sum_u \rho(u)\,, \tag{10}$$

where u are the configurations to be summed over, and $\rho(u)$ is the Boltzmann weight of u. Let us define a *set* of new Boltzmann weight functions $\rho_i(\tilde{u})$. The index i numbers these "modified interactions". Usually $\rho(u)$ is a product over local Boltzmann weights $\rho^c(u)$, where c are cells in the lattice. In this case the index i will be a multiindex over the relevant set of cells.

Assume now that during a Monte Carlo simulation we arrived at a particular configuration u. We choose a new configuration in a two step procedure. The first step is to replace the weight function ρ with one of the functions ρ_i. The probability $p_i(u)$ of choosing a specific i only depends on the current configuration u. It has to satisfy the following equations:

$$p_i(u) = q_i\,\frac{\rho_i(u)}{\rho(u)}\,, \qquad \sum_i p_i(u) = 1\,, \tag{11}$$

where $q_i \geq 0$ are constants independent of u. The second step is to update the configuration by employing a procedure that satisfies detailed balance for the chosen Boltzmann weight ρ_i. Kandel and Domany have shown that this two-step procedure satisfies detailed balance for the original Boltzmann weight function ρ.

If ρ is a product over some set of lattice cells as described above, and we decide upon ρ_i's with a similar product structure, it is sufficient to choose for each component of the multiindex i (i.e. for each cell) the modified interaction independently. In this case we shall have to fulfill (11) independently for each cell. For vertex models, the cells will be the vertices themselves.

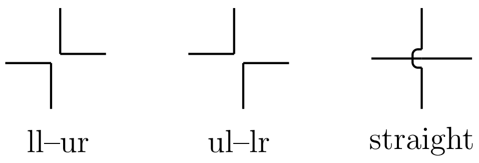

Figure 3: All three possible break-ups of a vertex.

5 The Loop Algorithm for the Full 6-Vertex Model

For the purpose of the loop algorithm, the main difference between the F model and the full arrow flip symmetric 6-vertex model is that, for the Boltzmann weights of (2), $a \neq b$. Consequently, we have to replace the freezing operation for the vertex configurations 1, 2, 3, 4, by three different operations. Firstly there is the freezing of 1, 2 (i.e. the freezing of the weight a), secondly the freezing of 3, 4 (i.e. the freezing of the weight b). Thirdly there is a new break-up operation, that allows transitions between the groups of weights 1,2 and 3, 4. This new break-up is not a break-up into two corners, but into two straight lines, as shown in fig. 3. As with the ul–lr and ll–ur break-ups, the straight break-up splits a vertex into two pieces, each of which contains an incoming and an outgoing arrow. During a cluster flip, each piece may be flipped independently.

Clusters are contructed in a very similar fashion to the case of the F model. Starting from a random bond b_0, we go from vertex to vertex on a path in the direction of the arrows. If we encounter a vertex v for which one of the break-ups was chosen, there is a unique way to pass through it. If on the other hand v is frozen, we continue the path in the direction of one of the outgoing arrows, and mark the other two arrows as also belonging to the cluster. Two things can now happen. If the path returns at a later stage to the vertex v, this will automatically be along the two marked arrows. If the path closes without returning to v, we have to append to the cluster a new "elementary loop" starting with the two marked arrows. The constraint guarantees that all such loops will eventually close.

After having constructed the cluster, we flip it, as in the case of the F model. Thus we have completely characterized our algorithm, up to the definition of the probabilities to choose a break-up or freeze for a vertex. These probabilities are most conveniently described by using the framework of Kandel and Domany [6].

For a given vertex, which is in the current configuration u with the Boltzmann weight $\rho(u)$, we define 6 modified Boltzmann weights ρ_i, $i = 1, ..., 6$, corresponding to specific break-up and freeze operations. These definitions are contained in table 3. The labelling of the modified Boltzmann weights is completely arbitrary, and the fact that their number – six – is the same as the number of allowed vertex configurations is just a coincidence. As discussed in [6] (see also table 3), *freezing* is described by introducing one modified weight 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 $\tilde{u} = 1$

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 3: The modified Boltzmann weight functions ρ_i and the probabilities $p_i(u)$ to choose them at a vertex in current configuration u.

and $\tilde{u} = 2$ cost nothing, whereas the vertex configurations 3, 4, 5, and 6 are not allowed. Each *break-up* is also described by one modified weight. As an example take the ul–lr break-up. It is given by the modified weight number 5, 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.

Notice that, in general, the break-ups correspond to allowing transitions between two groups of two configurations, each group being defined as the set of all configurations having the same given weight. By a slight abuse of language, we shall talk about the transition between the two weights. This point of view will be useful for the 8-vertex model.

In table 3 we also give the probabilities $p_i(u)$ to replace the original Boltzmann weight ρ by the modofied one ρ_i. Fulfillment of (11) ensures detailed balance and the proper normalization of probabilities. With the definitions in table 3, the first part of (11) is automatically fulfilled. Normalization of probabilities (i.e. the second part of (11)) implies constraints on the q_i:

$$\begin{aligned} q_1 + q_5 + q_6 &= a\ , \\ q_2 + q_4 + q_6 &= b\ , \\ q_3 + q_4 + q_5 &= c\ . \end{aligned} \tag{12}$$

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 optimal values for the constants q_i: we shall minimize the freezing probabilities, given the constraints (12) and $q_i \geq 0$. From (12) we immediately see that it is possible to minimize q_1, q_2 and q_3 simultaneously (by increasing q_4, q_5 and q_6). But let us discuss in detail the optimal values of the q_i for all 4 phases of the model [13, 14].

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 (12), $q_3 = c - a - b + q_1 + q_2 + 2q_6$. With $q_i \geq 0$ this implies $q_{3,\min} = c - a - b$. 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:

$$\begin{aligned} q_1 = 0, \quad q_2 = 0, \quad q_3 = c - a - b, \\ q_4 = b, \quad q_5 = a, \quad q_6 = 0\ . \end{aligned} \tag{13}$$

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,

$$\begin{aligned} q_1 = 0, \quad 2q_4 = b + c - a\ , \\ q_2 = 0, \quad 2q_5 = c + a - b\ , \\ q_3 = 0, \quad 2q_6 = a + b - c\ . \end{aligned} \tag{14}$$

We finish this section by showing how to obtain the F model algorithm described in section 2 from the more general 6-vertex algorithm. For the particular case of the F model, we only have phases III and IV, so we can set the freezing probabilities $q_1 = q_2 = 0$, i.e. we completely renounce these two freezing operations. Since $a = b$, (12) then implies $q_4 = q_5$. It is now straightforward to see that, with the identification of the straight break-up to the freezing of 1, 2, 3, 4, we have recovered the algorithm of section 2.

6 A Proposal for the 8-Vertex Model

For the 8-vertex model, the constraint at the vertices is no longer a zero-divergence condition. Therefore we cannot expect any more to have clusters that are made out of loops for which all the arrows point in the same direction. In the 6-vertex case this requirement was needed in order to preserve the constraints. The 8-vertex constraints on the other hand are already preserved if at each vertex the number of flipped arrows is even. This will allow us to devise an algorithm with clusters that are again collections of loops, but for which the orientation plays no role.

Let us consider the freeze and break-up operations for a given vertex v. Since we are dealing with the arrow flip symmetric 8-vertex model (see section 1), freezing would mean that the only allowed configuration change is the flip of all four arrows around v. This is similar to the 6-vertex case. Of course, we have here one extra weight that can be frozen, namely $d = \rho(7) = \rho(8)$. For the break-up the situation is only slightly more complicated. Consider as an example the ul–lr break-up. A corner flip performs transitions not only between the weights a and c, but also between the weights b and d. Within the framework of Kandel and Domany we can thus define two modified Boltzmann weights. The first, which we shall denote $\rho_{ac}(\tilde{u})$, obeys $\rho_{ac}(\tilde{u}) = 1$ for $\rho(\tilde{u}) = a$ or c, and $\rho_{ac}(\tilde{u}) = 0$ otherwise; it allows transitions without any cost between the vertex configurations 1, 2, 5, 6, and it prohibits the configurations 3, 4, 7, 8, from ever occuring. The second Boltzmann weight, which we shall denote $\rho_{bd}(\tilde{u})$, is obtained in the same way, by interchanging $a \leftrightarrow b$, $c \leftrightarrow d$, and $\{1, 2, 5, 6\} \leftrightarrow \{3, 4, 7, 8\}$. For the ll–ur and the straight break-ups we can also define two modified Boltzmann weights each in a completely analogous fashion.

The 10 modified Boltzmann weights can be described very elegantly using a symmetric matrix. To denote a weight, we use greek letters: α, β take values in the set $\{a, b, c, d\}$. The modified weights associated with freezing are denoted by $\rho_{\alpha\alpha}$, while those associated with the break-ups (two for each break-up) are denoted by $\rho_{\alpha\beta} \equiv \rho_{\beta\alpha}$, $\alpha \neq \beta$. They are fully characterized by:

$$\rho_{\alpha\beta}(\tilde{u}) = \begin{cases} 1 & \rho(\tilde{u}) = \alpha \quad \text{or} \quad \rho(\tilde{u}) = \beta \\ 0 & \text{else} \end{cases} . \tag{15}$$

The probabilities $p_{\alpha\beta}$ to choose one of these modified weights are then given by the first part of (11), while the second part of (11) implies the following 4 relations for the 10 constants $q_{\alpha\beta}$ ($\equiv q_{\beta\alpha}$):

$$\sum_{\beta} q_{\alpha\beta} = \alpha \, . \tag{16}$$

At this stage we again employ the principle of minimal freezing. In the case of the 6-vertex model, there were 3 freezing operations, hence 3 constants to minimize. Together with the 3 relations (12) this was enough for obtaining a unique optimal choice for the 6 q's. Here we have 4 freezing operations and 4 relations in (16). Thus even after minimizing freezing, we are left with two free parameters. We do not have any additional physical principle that allows us to fix them. Minimizing something else than freezing is certainly not good. It is also easy to check that no miraculous cancellations happen. The two undesired parameters are here to stay.

Let us indicate one possible solution to this problem. It is straightforward to prove that detailed balance is still fulfilled if the q's are chosen stochastically at each step, provided that this choice is independent of the current configuration u. Thus we propose the following procedure. Parametrize the q's such that freezing is minimal and (16) is fulfilled. For each vertex, then choose the two free parameters randomly, according to some fixed distribution.

After choosing a modified weight at each vertex, the clusters can be constructed in exactly the same way as for the 6-vertex model. If we start at a link b_0, "grow" a cluster and flip it, the amount of work will again be proportional to the cluster size. This completes the description of the loop algorithm for the 8-vertex model.

7 Conclusions and Outlook

We have presented a new type of cluster algorithm that considerably accelerates the simulation of vertex models. The clusters are closed paths of bonds, and the constraints at the vertices are automatically satisfied. We have successfully tested our algorithm for the F model and found remarkably small dynamical critical exponents.

The most important application of our algorithm seems to be the *critical acceleration of Quantum Monte Carlo simulations* This application is based on the fact that quantum spin systems in one and two dimensions can be mapped into vertex models in $1+1$ and $2+1$ dimensions via the Suzuki-Trotter formula and suitable splittings of the Hamiltonian [1].

The simplest example is the spin $\frac{1}{2}$ xxz quantum chain, which is mapped directly into the 6-vertex model or the 8-vertex model. For higher spins, more complicated vertex models result (e.g. 19-vertex model for spin one) [1].

For (2+1) dimensions, different splittings of the Hamiltonian lead to quite different vertex models, in particular on quite different lattice types [1]. For example, in the case of spin $\frac{1}{2}$ we can choose between simple 6-vertex or 8-vertex models on a quite complicated $2+1$ dimensional lattice (if the Hamiltonian is split into 4 sums of link terms), and models on a *bcc* lattice, with 8 bonds and a large number of configurations per vertex (if the Hamiltonian is split into 2 sums of plaquette terms).

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 [15] *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, similar to other cluster algorithms [5], it is straightforward to define improved observables. The investigation [15] 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 [1]). Thus it is well suited for simulations in the grand canonical ensemble.

Last, but not least, the loop algorithm just might open up a new avenue for taming the notorious fermion sign problem.

References

1. For basics of the Quantum Monte Carlo method see this volume, and, especially, its precursor (from which references to the original articles may be taken):
   M. Suzuki editor, *Quantum Monte Carlo methods in equilibrium and nonequilibrium systems*, Taniguchi symposium, Springer series in Solid State Physics ; 74 (1987).
2. P. C. Hohenberg and B. I. Halperin, Rev. Mod. Phys. **49** (1977) 435.
3. R. H. Swendsen and J. S. Wang, Phys. Rev. Lett. **58** (1987) 86.
4. 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.
5. 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.
6. D. Kandel and E. Domany, Phys. Rev. **B43** (1991) 8539.
7. 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.
8. R. Ben-Av, D. Kandel, E. Katznelson, P. Lauwers and S. Solomon, J. Stat. Phys. **58** (1990) 125.
9. D. Kandel, R. Ben-Av and E. Domany, Phys. Rev. Lett. **65** (1990) 941.
10. M. Hasenbusch, G. Lana, M. Marcu and K. Pinn, *Cluster algorithm for a solid-on-solid model with constraints,* Phys. Rev. **B46** (1992) 10472.
11. H.G. Evertz, G. Lana and M. Marcu, Phys. Rev. Lett. **70** (1993) 875.
12. H.G. Evertz and M. Marcu, in *Lattice 92*, Amsterdam 1992, ed. J. Smit et al., Nucl. Phys. B (Proc. Suppl.) **30** (1993) 277.
13. 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.
14. R. J. Baxter, *Exactly Solved Models in Statistical Mechanics* (Academic, 1989).
15. U.J. Wiese and H.P. Ying, Bern preprint, bulletin board cond-mat/9212006.
16. M. A. Novotny and H. G. Evertz, in *Quantum Monte Carlo Methods in Condensed Matter Physics*, ed. M. Suzuki (World Scientific 1993).