




1-1993

A Fast Serial Algorithm for the Finite Temperature Quenched Potts Model

Gregory N. Hassold
Kettering University

Elizabeth A. Holm
University of Michigan-Ann Arbor

Follow this and additional works at: https://digitalcommons.kettering.edu/physics_facultypubs

 Part of the [Physics Commons](#), and the [Theory and Algorithms Commons](#)

Recommended Citation

Hassold, Gregory N. and Holm, Elizabeth A., "A Fast Serial Algorithm for the Finite Temperature Quenched Potts Model" (1993).
Physics Publications. 10.
https://digitalcommons.kettering.edu/physics_facultypubs/10

This Article is brought to you for free and open access by the Physics at Digital Commons @ Kettering University. It has been accepted for inclusion in Physics Publications by an authorized administrator of Digital Commons @ Kettering University. For more information, please contact digitalcommons@kettering.edu.

A fast serial algorithm for the finite temperature quenched Potts model

G. N. Hassold

Department of Science and Mathematics, GMI Engineering and Management Institute, Flint, Michigan 48504

Elizabeth A. Holm

Department of Materials Science and Engineering, The University of Michigan, Ann Arbor, Michigan 48109

(Received 23 June 1992; accepted 18 August 1992)

An efficient serial algorithm for finite temperature, quenched Potts model simulations of domain evolution has been developed. This “ n -fold way” algorithm eliminates unsuccessful spin flip attempts *a priori* by flipping sites with a frequency proportional to their site activity, defined as the sum of the probability of success for every possible spin flip at that site. Finite temperature efficiency for high-spin degeneracy systems is achieved by utilizing a new, analytical expression for the portion of the site activity due to flips to non-neighbor spin values. Hence, to determine the activity of a site, only flips to the nearest neighbor spin values need be considered individually; all other flips are evaluated in a single expression. A complexity analysis of this algorithm gives the dependence of computing time on system parameters and on simulation progress. While a conventional Potts model algorithm has a constant computing time per simulation timestep, the n -fold way algorithm increases in efficiency as domain coarsening progresses. Computer experiments confirm the complexity analysis results and indicate that the n -fold way algorithm is much more efficient than the conventional algorithm even at high fractions of the critical temperature.

I. THE MONTE CARLO POTTS MODEL

Curvature-driven diffusive coarsening of domains governs domain growth in a variety of physical processes, including magnetic domain evolution, grain growth, and soap froth evolution. In such systems, domains arrange in a space-filling, cellular structure. The surface tension balance at domain edges and vertices gives rise to angular boundary conditions which necessitate that some domain boundaries must be curved. In order to decrease the total boundary area (hence the energy) of the system, boundaries move toward their centers of curvature. Thus, the average domain size tends to increase with time.

Since the diffusive time and length scales for the domain growth process are very large on an atomic scale, domain growth is computationally unsuitable for atomistic simulation. Conversely, while continuum models such as vertex-motion models¹⁻⁴ are computationally tractable, they are not easily extended beyond pure, single-phase systems. The high-spin degeneracy, order parameter nonconserved Potts model provides the best of both worlds, with the flexibility of a discrete simulation and the tractability of superatomic length and time scales. In fact, computer simulations utilizing the Potts model have become the *de facto* standard simulation technique and have successfully modeled a variety of domain growth systems, including grain

growth in single phase,⁵⁻⁷ two-phase,^{8,9} and composite systems,¹⁰ soap froth evolution,^{11,12} recrystallization,¹³ and late-stage sintering.¹⁴

The theory and computer implementation of the Potts model have been discussed in detail in a number of publications.^{5-7,15} In essence, a continuum domain structure is mapped onto a two- or three-dimensional lattice by assigning each lattice site i an index s_i corresponding to the domain in which the site is embedded, as shown in Fig. 1; the number of degenerate index values is Q . (Due to the historical use of such models for magnetic domain evolution, the indices are referred to as “spins” and a change in index is a “spin flip.”) The mapping procedure is analogous to color bitmapping the domain structure; domains are clusters of pixels (sites) of the same color (spin). The total system energy is given by the Potts Hamiltonian

$$H = E_0 \sum_{i=1}^N \sum_{j=1}^{z(i)} 1 - \delta(s_i, s_j), \quad (1)$$

where E_0 is the positive energy associated with adjacent unlike spins, N is the total number of lattice sites, $z(i)$ is the number of nearest neighbors j of site i [in this paper, we assume $z(i)$ equals some constant z for all sites in a given system], and δ is the Kronecker delta function with $\delta(s_i, s_j) = 1$ if $s_i = s_j$ and 0, otherwise. In essence, this

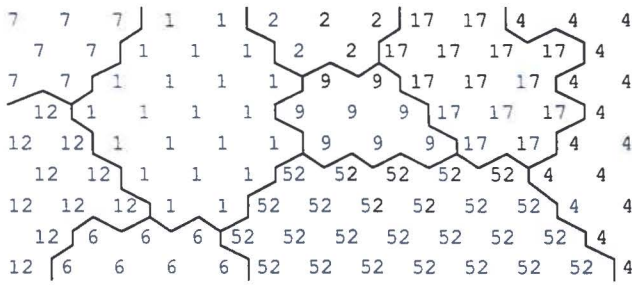


FIG. 1. A continuum domain structure mapped onto a triangular lattice. Each lattice site represents a unit area of the continuum system; the spin assigned to each site corresponds to the domain in which that site is embedded. Thus the cluster of sites of spin 9 correspond to one domain. Domain boundaries occur between sites of unlike spin values.

Hamiltonian counts unlike neighbors of site i ; the energy of site i is simply E_0 times the number of neighbors of i which have spins different from s_i .

In terms of magnetic domains, this Hamiltonian describes a ferromagnetic system in which the perfectly ordered (single domain) state has zero energy; all other states have positive energy which scales with the total domain boundary area of the system. Alternatively, under a grain growth paradigm, E_0 scales with the interfacial energy between unlike orientation grains, and the Hamiltonian describes a system in which the single crystal state has zero energy; all polycrystalline states have positive energy which scales with the total grain boundary area.

Domain growth kinetics are determined through a Monte Carlo technique with a nonconserved order parameter (Glauber dynamics). First, a lattice site and a spin value are chosen at random. The energy change ΔE associated with flipping the site to the random spin value is computed using the Potts Hamiltonian, and the flip is performed with a probability $P(\Delta E)$. The requirement of detailed balance on the transition probabilities is insufficient to uniquely specify $P(\Delta E)$; two common choices are the Metropolis method

$$P(\Delta E) = \begin{cases} 1 & \text{if } \Delta E \leq 0, \\ \exp(-\Delta E/kT) & \text{if } \Delta E > 0, \end{cases} \quad (2)$$

which, for $T = 0$, reduces to

$$P(\Delta E) = \begin{cases} 1 & \text{if } \Delta E \leq 0, \\ 0 & \text{if } \Delta E > 0, \end{cases} \quad (3)$$

and the symmetric method

$$P(\Delta E) = \frac{1}{2}[1 - \tanh(\Delta E/2kT)], \quad (4)$$

which for $T = 0$ reduces to

$$P(\Delta E) = \begin{cases} 1 & \text{if } \Delta E < 0, \\ 0.5 & \text{if } \Delta E = 0, \\ 0 & \text{if } \Delta E > 0. \end{cases} \quad (5)$$

The choice of probability function has no effect upon the geometric and topological characteristics of the evolving domain structure. However, the functional form of $P(\Delta E)$ does affect the kinetics of domain evolution slightly.

In the Monte Carlo simulation, time is incremented after each attempted spin flip by $1/N$ Monte Carlo steps

(MCS), where N is the total number of lattice sites in the system.

Figure 2 depicts a typical series of domain structures for a $Q = 100$ state Potts model with symmetric probability function [Eq. (4)] on an $N = 40\,000$ site triangular lattice which has been quenched from $kT/E_0 = \infty$ (i.e., from a completely random initial domain structure) to $kT/E_0 = 0.1$. The increase in the mean domain size with time continuously reduces the domain perimeter length and thus the system energy. Figure 3 shows the time dependence of mean domain radius r . The slope of this log-log plot increases monotonically in time to a value consistent with the large-system asymptotic exponent of $1/2$ expected for domain growth in an infinite, ideal system.⁵

II. CONVENTIONAL POTTS MODEL ALGORITHM

The conventional Potts model algorithm (CPM) is a straightforward implementation of the sequence of steps given above. For each Monte Carlo step of domain growth, the following operations are performed:

CPM

- 1 for $j = 1$ to N do,
- 2 $E_i = 0$,
- 3 $E_f = 0$,
- 4 pick a site $i: i \in \{1, \dots, N\}$ at random,
- 5 pick a spin value $s_i^*: s_i^* \in \{1, \dots, Q\}$ at random,
- 6 for all neighbors k of site i do,
- 7 if $s_k \neq s_i$ then,
- 8 $E_i = E_i + E_0$,
- 9 if $s_k \neq s_i^*$ then,
- 10 $E_f = E_f + E_0$,
- 11 $\Delta E = E_f - E_i$,
- 12 flip site i to spin s_i^* with probability $P(\Delta E)$
- 13 increment time by $1/N$ Monte Carlo steps.

Therefore, each Monte Carlo step requires N iterations of a z -iteration loop (lines 6–10) plus N iterations of some constant time operations (lines 1–5 and 11–13), so the characteristic computing time per MCS

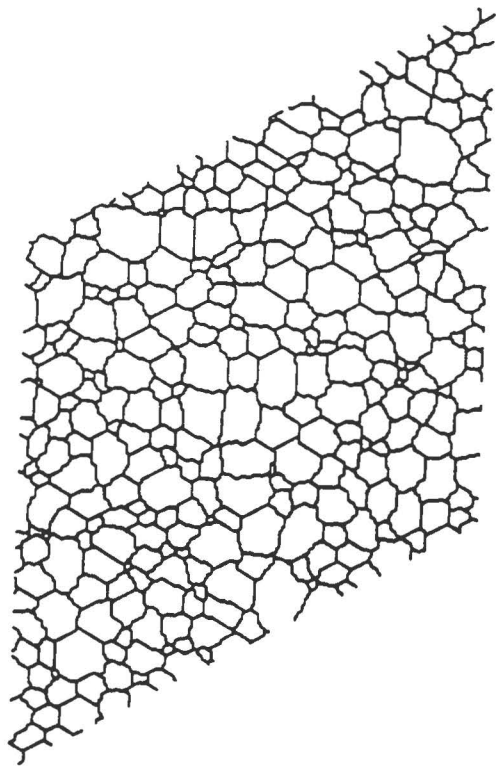
$$t_{\text{cpu}} \propto N(z + \text{constants}), \quad (6)$$

where the constant of proportionality depends on machine type and coding details. In this paper, we will follow the practice of ignoring all but asymptotically significant terms in discussions of computing times. Hence, since z is a variable that may be made larger than any constant term, we write

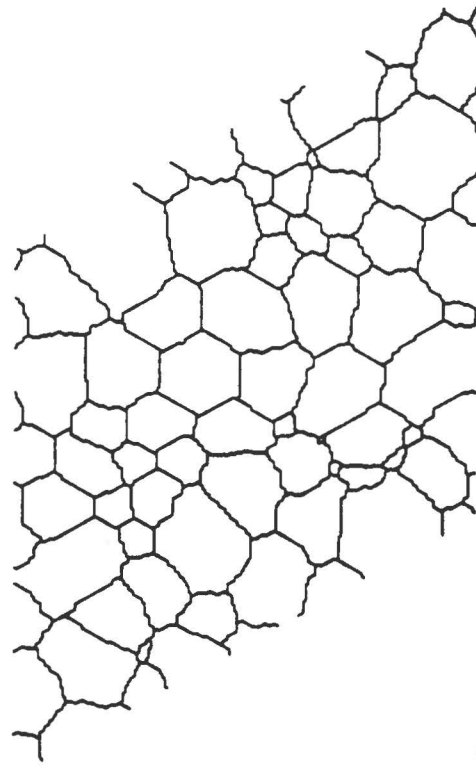
$$t_{\text{cpu}} \propto Nz. \quad (7)$$

Note that for a given N and z , the computing time per MCS does not change as the simulation progresses.

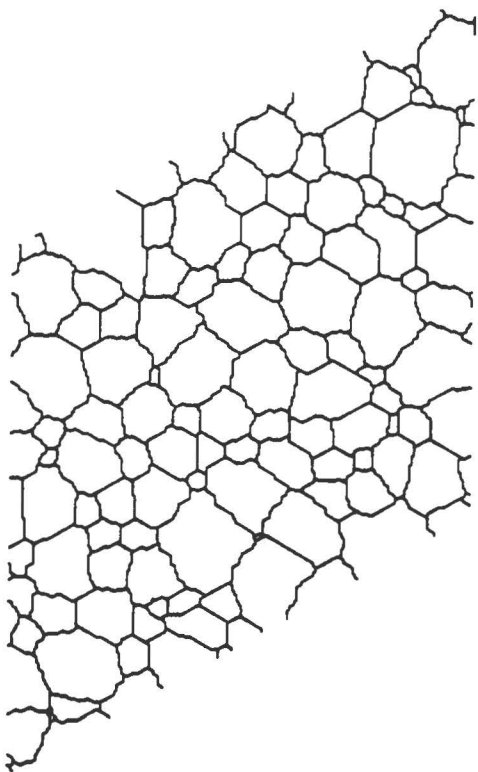
Since typical Potts model applications may require as many as 10^6 – 10^8 MCS per simulation, the linear dependence of computing time per MCS on lattice size may be too slow for large lattice simulations. However, there is an obvious inefficiency in the conventional Monte Carlo algorithm: In the evolved domain structure of Fig. 2, it is evident that for low temperatures relative to the critical temperature, sites within the interior of a domain are unlikely to flip to any other spin value, since a flip would entail a large positive energy cost $\Delta E = z \cdot E_0$. Only those sites at domain boundaries will have a significant chance of



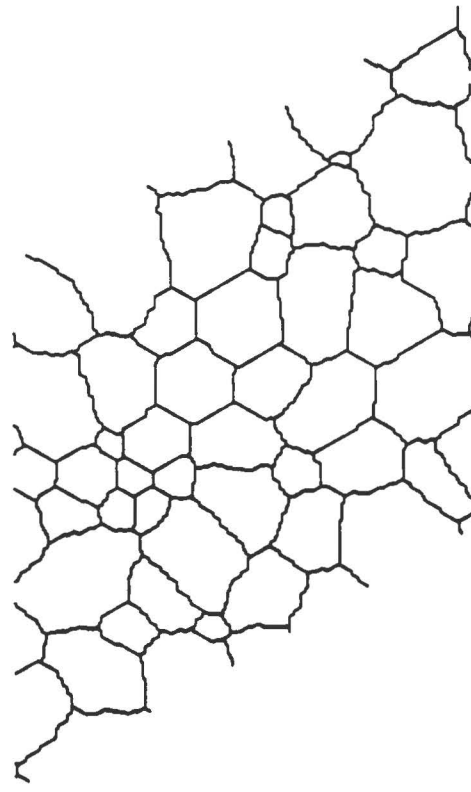
(a)



(c)



(b)



(d)

FIG. 2. Domain evolution in a $Q = 100$ state Potts model on an $N = 40\,000$ site triangular lattice evolving from a completely random initial domain structure at a temperature such that $kT/E_0 = 0.1$. (a) Simulation time $t = 3000$ Monte Carlo steps (MCS). (b) $t = 10\,000$ MCS. (c) $30\,000$ MCS. (d) $t = 100\,000$ MCS. The increase in the mean domain size with time continuously reduces the domain boundary length and thus the system energy.

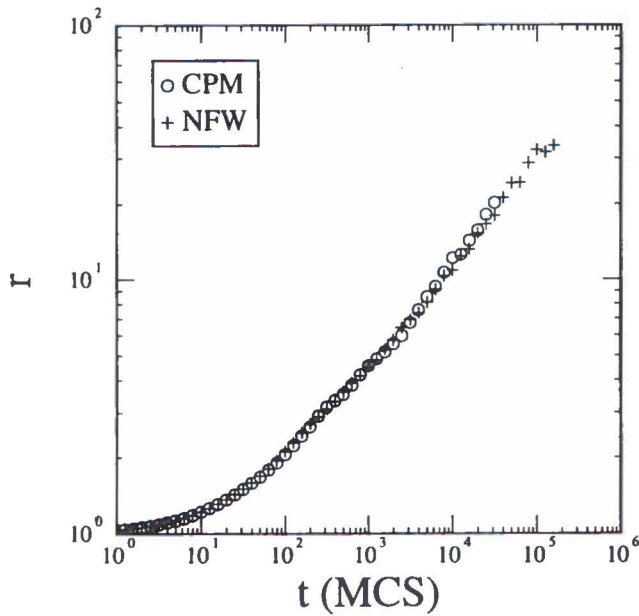


FIG. 3. The evolution of mean domain radius r with simulation time t in the $Q = 100$ state Potts model on an $N = 40\,000$ site triangular lattice at a temperature such that $kT/E_0 = 0.1$. The CPM and NFW simulation algorithms produce statistically identical evolution kinetics. The growth exponent is consistent with a large-system asymptotic exponent of $1/2$.

flipping, and then only to a spin value represented among the site's nearest neighbors. Hence, as domains increase in size, fewer and fewer iterations of the conventional Monte Carlo loop will result in spin flips actually occurring. In this paper, we present an algorithm based on the approach of Bortz *et al.*,¹⁶ which eliminates unsuccessful flip attempts *a priori*. The characteristic time per MCS of this algorithm decreases as the simulation progresses, allowing the extensive simulations required in many Potts model studies.

III. THE ISING MODEL AND THE N-FOLD WAY

Consider the Ising model, which is equivalent to a Potts model with only two degenerate spin states ($Q = 2$). We define an *effective* spin flip as a flip which alters the domain structure of the system. In the Ising model, a given site has but one effective spin flip: a flip to the single spin value different from the original spin of the site. Hence, in the CPM algorithm half of all attempted spin flips will always be ineffective. Moreover, in an evolved Ising system, most sites are located within the interior of a domain where any effective flip entails a large positive energy cost, so most effective flips occur with low probability of success. Bortz *et al.*¹⁶ first proposed an algorithm to eliminate unsuccessful (i.e., ineffective and energetically unfavorable) spin flip attempts *a priori*, so that all flip attempts actually result in domain evolution.

For an Ising model on a z -fold coordinated lattice, only $n = 2z + 2$ different spin environments exist: site i must have either spin $s_i = 1$ with $z, z - 1, \dots, 2, 1$, or 0 like neighbors or $s_i = 2$ with $z, z - 1, \dots, 2, 1$, or 0 like neighbors.

Each site with a given environment has the same transition energy ΔE for an effective flip. The " n -fold way" algorithm entails tabulating the spin environment of each site in the lattice. The activity Π_k of the k th spin environment is defined as

$$\Pi_k \equiv N_k \cdot P(\Delta E_k), \quad (8)$$

where N_k is the number of sites with environment k and ΔE_k is the transition energy of environment k . Each site environment is visited with a probability weighted by its activity. Every time an environment is chosen, an effective spin flip is performed upon a random site with that environment, and the environment of the site and its neighbors are re-evaluated.

In the conventional Monte Carlo algorithm, the simulation time is incremented by a constant amount after each attempted spin flip, whether successful or not. In the n -fold way, every spin flip attempt is successful, so the n -fold way time increment must be scaled by the average time between successful flips in the conventional Monte Carlo scheme. Let τ be the time in which each lattice site is visited on average once in the conventional Monte Carlo algorithm; that is, $\tau = 1$ MCS. Now define the total system activity A such that

$$A \equiv \sum_{k=1}^{2z+2} \Pi_k = \sum_{i=1}^N \pi_i, \quad (9)$$

where π_i is the transition probability [$P(\Delta E)$ for an effective flip] of site i . Then the site average transition probability $\langle \pi \rangle = A/N$, and the average success rate f (i.e., the number of successful flips per MCS) is given by

$$f = \frac{N \langle \pi \rangle}{\tau} = \frac{A}{\tau}. \quad (10)$$

So the probability of any lattice site undergoing a successful flip in time dt is

$$f dt = (A/\tau) dt. \quad (11)$$

The n -fold way time increment should be scaled by the average time between successful spin flips in the conventional Monte Carlo scheme. Define $g(\Delta t)$ as the probability that no successful flip has occurred in the time interval Δt since the last successful flip and $g(\Delta t + dt)$ as the probability that no successful flip has occurred in the time interval $\Delta t + dt$ since the last successful flip. We note that

$$g(\Delta t + dt) = g(\Delta t) \cdot g(dt) \quad (12)$$

and

$$g(dt) = 1 - f dt = 1 - (A/\tau) dt. \quad (13)$$

Combining Eqs. 12 and 13 and taking a Taylor expansion of P about Δt , we find

$$g(\Delta t + dt) = g(\Delta t) \cdot \left(1 - \frac{A}{\tau} dt\right) = g(\Delta t) + \frac{dg(\Delta t)}{dt} dt, \quad (14)$$

which may be rearranged into the differential equation

$$\frac{dg(\Delta t)}{g(\Delta t)} = -\frac{A}{\tau} dt, \quad (15)$$

with solution

$$\ln[g(\Delta t)] = -(A/\tau)\Delta t. \quad (16)$$

Now we note that Δt is a stochastic variable, so $g(\Delta t)$ will

take all of its values with equal probability. Hence, we may replace $g(\Delta t)$ with a random number R on the interval $(0,1)$ so that the n -fold way time increment

$$\Delta t = -(\tau/A) \ln R. \quad (17)$$

This time increment will tend to increase as the total system activity decreases. Hence, as domains grow and more sites acquire low transition probabilities, the n -fold way time increment will grow, reflecting the increased time between successful spin flips in the conventional Monte Carlo algorithm.

It should be noted that this derivation is precisely analogous to determining the free-flight time or distance of an electron or photon from a scattering probability distribution.¹⁷ Here, the site average transition probability $\langle \pi \rangle$ corresponds to the average scattering event probability, and Δt corresponds to a Monte Carlo free-flight time/distance (i.e., the time/distance during which no events happen).

Bortz *et al.* showed that the n -fold way scheme utilizing the above time increment produces results identical to those of the conventional Monte Carlo algorithm with substantially less computation time.¹⁶

IV. THE POTTS MODEL AND THE N-FOLD WAY

A. Zero temperature algorithm

While a site may undergo only one effective spin flip in the $Q = 2$ Ising model, a site in the Q state Potts model has $Q - 1$ different, effective flips available to it. That is, a site may flip to any of the $Q - 1$ spin values different from its own. Hence, the number of different site environments is very large. We can define the activity π_i of site i with spin s_i as

$$\pi_i \equiv \sum_{s_j \neq s_i} P[\Delta E(s_i \rightarrow s_j)], \quad (18)$$

where the sum is taken over the $Q - 1$ spins s_j different from s_i and $\Delta E(s_i \rightarrow s_j)$ is the change in system energy upon flipping site i to spin s_j . While this sum is lengthy to evaluate in the general case, Sahni *et al.* noted that for the $T = 0$ Metropolis algorithm the transition probability for each effective flip is either zero or unity, so that the activity of site i is given by the number of different, effective flips that are allowed energetically.¹⁵ The situation is further simplified at late times, when the majority of sites are located within domains and have only like-spin neighbors. Such "bulk" sites have zero probability of any effective flip; $\pi_i = 0$. Moreover, essentially all of the sites located on domain boundaries have at least one like-spin neighbor and thus have zero probability of flipping to any spin except a neighbor spin, so π_i is a small, easily computed integer.

To take advantage of these late-time simplifications, Sahni *et al.*¹⁵ proposed a $T = 0$ Metropolis algorithm based on the n -fold way Ising model algorithm of Bortz *et al.* For a given site i , let q_i be the number of neighbor spin values different from s_i . (Note: q_i is not necessarily equal to the

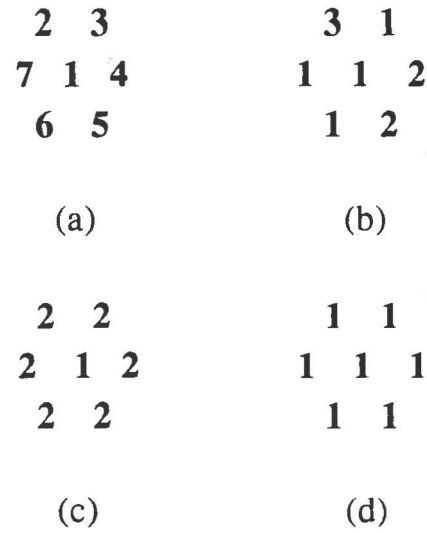


FIG. 4. For a given site i with spin s_i , q_i is the number of neighbor spin values different from s_i . If site i is the center site with spin 1, then (a) $q_i = 6$, (b) $q_i = 2$, (c) $q_i = 1$, and (d) $q_i = 0$.

number of unlike neighbors of site i , as shown in Fig. 4.) In their scheme, sites with $q_i = 0$ (bulk sites) are assigned $\pi_i = 0$; the activity of sites with $1 \leq q_i < z$ is the sum of the probabilities of flipping to each of the q_i unlike neighbor spin values (a sum of 0's and 1's); and the activity of sites with $q_i = z$ is determined by evaluating and summing the probabilities of every possible effective flip. Then, the Potts model n -fold way algorithm proceeds analogously to the Ising model n -fold way algorithm: A site is visited with a frequency proportional to its activity. When a site is visited, one of the energetically favored spin flips (all of which have unit probability) is chosen at random and performed, and the activities of the site and its neighbors are re-evaluated.

The Potts time increment is developed similarly to the Ising increment. Total system activity A is defined as in Eq. (9). The site average transition probability $\langle \pi \rangle = A/N$. Since only $\langle \pi \rangle / (Q - 1)$ attempted flips are successful in the conventional Monte Carlo scheme, the average number of successful flips per MCS f is given by

$$f = \frac{N \langle \pi \rangle}{(Q - 1)\tau} = \frac{A}{(Q - 1)\tau}. \quad (19)$$

The n -fold way time increment Δt is then derived as in Eqs. (10)–(16) with the result

$$\Delta t = -[(Q - 1)\tau/A] \ln R, \quad (20)$$

where R is a random number on $(0,1)$.

While Sahni *et al.* found this algorithm to be significantly faster than the conventional Monte Carlo algorithm at $T = 0$, they found the calculation of the site transition probabilities to be intractably slow at finite temperatures where each of the $Q - 1$ effective spin flips has a finite probability of occurring at every site.¹⁵

B. Finite temperature algorithm

However, an efficient n -fold way algorithm for finite temperature systems may be derived based on the fact that the probabilities of most of the $Q - 1$ effective spin flips at a site may be determined analytically. We define a *tame* spin flip at site i as an effective flip to a nearest neighbor spin value; a *wild* spin flip is an effective spin flip to a spin value not equal to any neighbor spin value.

As in Sahni's algorithm, for a given site i , let q_i be the number of neighbor spin values different from s_i ; these q_i spin values are the tame spin values. The remaining $Q - q_i - 1$ spin values are the wild spin values. Similarly, the activity of site i may be divided into two parts. The tame portion of π_i is the sum of the probabilities of flipping to each of the q_i unlike neighbor spin values. (In the finite temperature system, each of the q_i unlike spin flips has a finite probability of occurring.) The wild portion of π_i is the total probability of all wild flips.

The wild portion of π_i need not be calculated by explicitly summing the $Q - q_i - 1$ wild flip probabilities. During a wild flip, site i goes from having n_i like-spin neighbors to zero like-spin neighbors. Hence, ΔE for any wild flip is $n_i E_0$, and the probability of flipping to any one of the $Q - q_i - 1$ wild spin values is exactly $P(n_i, E_0)$. So the wild part of the activity of site i is given by the analytical expression

$$\pi_i^{\text{wild}} = (Q - q_i - 1)P[(z - n_i)E_0]. \quad (21)$$

Therefore, the activity of any site may be computed by

summing the probabilities of flipping to each of the q_i tame spin values and adding in the wild contribution given in Eq. (21). Since the number of tame spin values is less than or equal to the number of nearest neighbors z , the activity of a site in a finite temperature system has at most $z + 1$ components ($\leq z$ tame flip probabilities and one "total" wild flip probability), not Q components as in Sahni's algorithm.

The finite temperature n -fold way algorithm proceeds analogously to the zero temperature algorithm. A site is visited with a frequency proportional to its activity. When a site is visited, one of its tame spin flips or a wild flip is chosen with probability proportional to its relative flip probability. If a tame flip is chosen, it is performed; if a wild flip is chosen, the site is flipped to one of the $Q - q_i - 1$ wild spin values at random. Finally, the activities of the site and its neighbors are re-evaluated, and the simulation time is incremented.

The time increment is precisely the same as in the zero-temperature algorithm, and is given in Eq. (20).

C. n -fold way time complexity

In order to efficiently select flips based on relative site activities and flip probabilities, the n -fold way algorithm utilizes two tables: $\pi[i]$ is a vector of length N which gives the activity of site i , and $p[i,k]$ is an array with $1 \leq i \leq N$ and $0 \leq k \leq z$ which gives the probability of flipping site i to the spin of its k th neighbor ($p[i,0]$ gives the wild flip probability of site i). The finite temperature n -fold way algorithm NFW is given by the following sequence of steps:

NFW

```

1  while ( $t < t_{\text{maximum}}$ ) do
2    pick an activity  $a \in (0, A]$  at random,
3    find the site  $i$  such that  $\sum_{k=1}^{i-1} \pi[k] < a \leq \sum_{k=1}^i \pi[k]$ ,
4    pick a flip activity  $p \in (0, \pi[i])$  at random,
5
6    find  $i^*$  such that  $\sum_{k=0}^{i^*-1} p[i,k] < p \leq \sum_{k=0}^{i^*} p[i,k]$ ,
7    if  $i^* = 0$  then a wild flip was chosen so
8      find a random spin  $s_{i^*}$ :  $s_{i^*} \neq s_i$  and  $s_{i^*} \neq$  any neighbor spin,
9    else
10     set  $s_{i^*}$  equal to the spin of the  $i^*$ th neighbor of site  $i$ ,
11    flip site  $i$  to spin  $s_{i^*}$ ,
12    for site  $i$  and its  $z$  neighbors do,
13      call UPDATE ( $i$ ),
14    increment time by  $\Delta t$ .
```

Lines 2, 4, 7, 10, 11, and 14 require constant computing time independent of the system parameters N , Q , and z . Thus the computing time for NFW is proportional to the time for finding the site and flip (lines 3 and 6) plus z iterations of the UPDATE subroutine (line 13). In addition, in the relatively rare case when a wild flip is chosen, line 8 will require z operations to find a spin not equal to any neighbor spins.

The routine UPDATE updates the system activity, site activities, and flip probabilities after a flip occurs. (UPDATE also may be used to initialize the tables of activities and probabilities before starting NFW.) This routine creates and utilizes two special arrays: number $[s_k]$ is a vector of length Q which gives the number of neighbors of site i with spin value s_k ; flip $[k]$ is the z -length vector of pointers from neighbor k to its spin value s_k . UPDATE simply uses Eqs. (2)–(5), (18) and (21) to update flip probabilities $p[i,k]$, site activities π_k , and the system activity A .

UPDATE (i)

```

1  old_πi := πi,
2  for all neighbors k of site i do,
3      if number[sk] = 0 then,
4          flip[k] := sk,
5          q' := q + 1,
6          number[sk] := number[sk] + 1,
7  ΔE := E0 * number[si],
8  p[i,0] := (Q - q' - 1) * P(ΔE),
9  πi := p[i,0],
10 for k := 1 to z do,
11     if flip[k] ≠ 0 then
12         ΔE := E0 * (number[si] - number[sk]),
13         p[i,k] := P(ΔE),
14         πi := πi + p[i,k],
15 A := A - old_πi + πi.

```

Lines 1, 7–9, and 15 require constant computing time; lines 2–6 and 10–14 are sequential *z*-iteration loops. Thus the computing time per call to UPDATE is proportional to *z* plus constant terms.

According to line 1 of NFW, the number of NFW iterations to complete one MCS depends upon the *n*-fold way time increment Δt given in Eq. (20), which changes as the simulation progresses. However, note that the average value of Δt during a given time interval is given by

$$\langle \Delta t \rangle = (Q - 1)\tau / \langle A \rangle, \quad (22)$$

since $\langle \ln R \rangle = -1$ for *R* random on (0,1). Recalling that $\tau = 1$ MCS, on average $\langle A \rangle / (Q - 1)$ NFW iterations are required to advance the simulation by one Monte Carlo step. Therefore, in general, the computing time for 1 MCS may be written

$$t_{\text{cpu}} \propto \frac{\langle A \rangle}{Q - 1} \cdot (t_{\text{search}} + z \cdot t_{\text{update}}), \quad (23)$$

where t_{search} is the computing time to search for the site and flip in NFW, t_{update} is the UPDATE computing time, and *z* is the coordination number of the lattice.

Because the number of loop iterations per MCS is proportional to the system activity, some details about the time dependence of the system configuration are necessary to further evaluate the number of operations required for one Monte Carlo step. We choose pure, single-phase domain evolution as the archetypical Potts model system. The *n*-fold way efficiency for other types of systems may be determined analogously with this example.

In the scaling state of pure, single phase domain growth, the mean domain radius *r* scales parabolically with time; that is, in the long-time limit $r \propto t^{1/2}$, where the constant of proportionality includes temperature effects.

In the Potts model at late simulation times, the overwhelmingly most active sites are those on the domain boundaries, even at high fractions of the critical temperature. Thus the mean system activity $\langle A \rangle$ is directly proportional to the number of boundary sites. The number of boundary sites in the system is, in turn, proportional to the number of domains *D* in the system multiplied by the average domain perimeter *P*. Since $D \propto N/r^2$ and $P \propto r$, we find

$$\langle A \rangle \propto N/r \propto N/t^{1/2}. \quad (24)$$

In this case, the *n*-fold way computing time per MCS is given by

$$t_{\text{cpu}} \propto \frac{N}{(Q - 1)t^{1/2}} \cdot (t_{\text{search}} + z \cdot t_{\text{update}}). \quad (25)$$

(Note that we choose to express t_{cpu} in terms of *N*, *Q*, *z*, and *t*; we could equivalently choose *N*, *Q*, *z*, and *r*, since $r \propto t^{1/2}$.)

In the algorithm presented above, the site and flip searches are done by simple summations. Basically, the site activities π_i are added sequentially until the randomly chosen site activity *a* meets the criterion in line 3 of the NFW algorithm. Likewise, the flip activities $p[i,k]$ are added until the randomly chosen flip activity *p* meets the criterion in line 6. Since all sites and all flips are statistically equivalent, we expect to perform, on average, *N*/2 additions per site search and about *z*/2 additions per flip search. In this simple form of the algorithm, the UPDATE subroutine consists of two sequential *z*-iteration loops, so it requires computing time proportional to *z*. Therefore, the expected computing time per MCS is

$$t_{\text{cpu}} \propto \frac{N}{(Q - 1)t^{1/2}} \cdot (N + z + z^2 + \text{const}), \quad (26)$$

where the constant of proportionality depends on machine type and coding details. Since in typical simulations, *z*, *z*², and constants are all much less than *N*, in large systems t_{cpu} is asymptotically bounded such that

$$t_{\text{cpu}} \propto \frac{N^2}{(Q - 1)t^{1/2}}. \quad (27)$$

Note that even this nonintelligent search scheme results in an *n*-fold way algorithm that is faster at late simulation times than the conventional Monte Carlo scheme.

In the above example, the time to search for the site to be flipped governs the computing time of the *n*-fold way. The search time may be decreased in a number of ways. For instance, consider a scheme in which sites which may undergo tame flips (i.e., boundary sites) are placed into one "bin," sites which may undergo only wild flips (i.e., bulk sites) are placed in the other, and the transition probabilities of the sites in each bin are summed to give a "bin activity." At most simulation temperatures, the boundary sites

are almost always the sites chosen to be flipped, so only the boundary site bin need be searched (via the usual summation scheme). Since the number of boundary sites is proportional to $N/t^{1/2}$, t_{search} is almost always proportional to $N/t^{1/2}$. After a flip is performed, only one extra variable (the bin activity) must be updated in the UPDATE subroutine, so t_{update} remains proportional to z . If $z \ll N$ and $z^2 \ll N$, t_{cpu} scales as $N^2/(Q-1)t$ (plus lower-order terms in $z^2N/t^{1/2}$), a significant increase in computing efficiency. However, at very high simulation temperatures the assumption that only boundary sites are routinely flipped breaks down, and the computing time reverts to that of the sequential-search-based algorithm above.

This binning approach may be extended into a multi-bin scheme which is efficient at all temperatures below the system disordering temperature T_c . For instance, suppose sites are placed into \sqrt{N} bins of \sqrt{N} sites each and site activities are summed to give the bin activity of each bin (i.e., bin 1 contains sites 1 through \sqrt{N} and its bin activity is equal to $\pi_1 + \pi_2 + \dots + \pi_{\sqrt{N}}$). Then searching for a site entails finding the correct bin by summing bin activities (on average $\sqrt{N}/2$ bin activities added), then finding the site in the bin (on average $\sqrt{N}/2$ site activities added), then finding the correct flip to perform (on average $z/2$ flip probabilities added). Since only one additional variable (the bin activity) requires updating after a flip, t_{update} remains proportional to z , so

$$t_{\text{cpu}} \propto \frac{N}{(Q-1)t^{1/2}} \cdot (N^{1/2} + z + z^2 + \text{const}). \quad (28)$$

Note that, in practice, z^2 is often the same order of magnitude as $N^{1/2}$, so terms in both N and z^2 may contribute significantly to t_{cpu} , so that at late times

$$t_{\text{cpu}} \propto \frac{N}{(Q-1)t^{1/2}} \cdot (N^{1/2} + z^2). \quad (29)$$

Also, since the n -fold way requires considerably more bookkeeping "overhead" than the very simple CPM algorithm, for the smallest systems, even the constant terms in Eq. (28) may contribute significantly to the total computing time.

Finally, we can determine an optimal binning scheme to minimize the total computing time with respect to N . Consider a system of bins with X "levels" and $N^{1/X}$ bins per level. The search for a given site proceeds by first summing bin activities to find the first-level bin containing the required site. Then, another summation finds the correct second-level bin, and so on, until the last bin found contains only the site to be flipped. The expected number of additions performed Y is given by

$$Y = XN^{1/X}/2, \quad (30)$$

which is minimized when $X = \ln N$. Hence, the optimal binning system is one consisting of $\ln N$ bin levels with $N^{1/\ln N} = e$ elements (i.e., sub-bin or site activities) per bin. Since the number of elements per bin must be an integer, we can approximate this optimal scheme by one with $\log_2 N$ levels of bins with 2 bins at each level, in that case, this scheme is merely a form of binary search, which is an optimal comparison-based search scheme. The search time is directly proportional to the number of additions performed

Y , so with optimal binning, $t_{\text{search}} \propto \log_2 N$. In addition, every time a site is updated, $\log_2 N$ bins must be updated as well, so $t_{\text{update}} \propto z \log_2 N$. Hence, the characteristic computing time for an optimally binned system is

$$t_{\text{cpu}} \propto z^2 N \log_2 N / (Q-1)t^{1/2}, \quad (31)$$

plus constant and lower-order terms. Note that for large systems in which $z^2 \ll \log_2 N$, this binning scheme is asymptotically faster than the two-level binning scheme discussed above. However, in practice, z^2 is often much larger than $\log_2 N$, so the optimally binned system is slower than the two-level binning scheme; in that case, the increase in t_{update} outweighs the optimal decrease in t_{search} .

V. RESULTS

In order to evaluate the advantages and examine the time complexity of the n -fold way algorithm, a number of Potts model domain growth simulations were performed on identically configured SPARCstation 2 workstations. The default simulation system is a 100×100 triangular lattice ($N = 10\,000$ and $z = 6$) with 100 degenerate spin values ($Q = 100$) which is quenched at the beginning of the simulation from a perfectly disordered system with a temperature such that $kT/E_0 = \infty$ to $kT/E_0 = 0.1$. (Note that the critical or disordering temperature T_c is such that $kT_c/E_0 = 0.65$ for the $Q = 100$ Potts model on a triangular lattice.) The n -fold way algorithm implemented here uses the NFW algorithm presented above with a two-level binning scheme (i.e., sites are stored in \sqrt{N} bins of \sqrt{N} sites each).

Domain growth kinetics generated by an $N = 40\,000$ site CPM simulation are compared with those of an $N = 40\,000$ site NFW simulation in Fig. 3. The data sets are statistically identical, and the growth exponent in both systems increases monotonically in time to a value consistent with the large-system asymptotic exponent of $1/2^5$. Numerous experimental results such as these (i.e., detailed geometric and topological analysis of evolving domain structures) confirm that the CPM and NFW algorithms produce statistically identical results. However, note that for a given initial domain structure, the exact structure of the evolved system will differ with the simulation algorithm.

The efficiencies of both algorithms are contrasted in Fig. 5, which displays the computing time per MCS (t_{cpu}) versus the simulation time t (in MCS) for $N = 40\,000$ site simulations. As predicted in Eq. 7, the CPM algorithm has a constant t_{cpu} throughout the simulation. (The very slightly larger initial t_{cpu} results from the proportionally higher fraction of time spent in the external domain measurement algorithm early in the simulation.) In contrast, in the NFW algorithm, t_{cpu} decreases continuously with a late-time slope of about $-1/2$, so t_{cpu} varies inversely with the square root of t , as predicted by Eq. (29).

Note that the NFW algorithm is initially much slower than the CPM algorithm. This early inefficiency occurs because the initial domain structure is very fine grained, so nearly all sites have a large wild flip probability, the total system activity is high, and the average n -fold way time increment is small. It is only after the system has coarsened sufficiently to decrease the net activity (and particularly the total wild flip activity) that NFW overtakes CPM. This indicates that, in practice, it is prudent to commence a

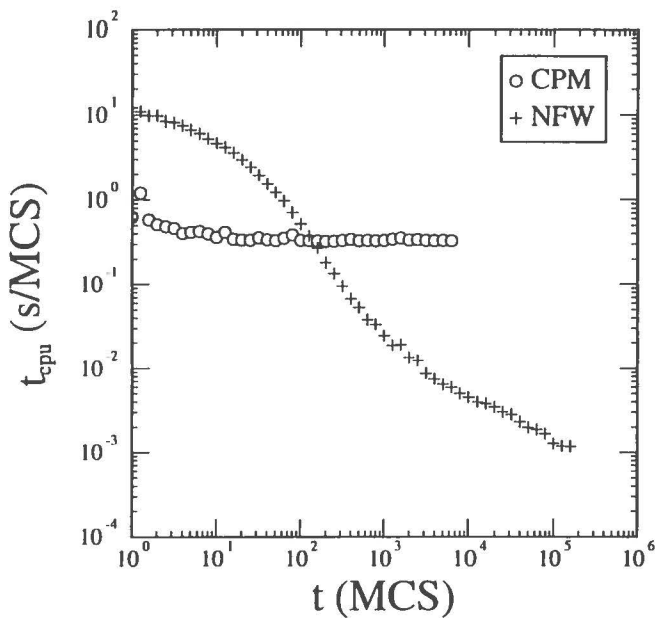


FIG. 5. The computing time per simulation timestep t_{cpu} versus the simulation time t for $Q = 100$ state, $N = 40\,000$ site, $kT/E_0 = 0.1$ simulations. The CPM algorithm has a constant t_{cpu} throughout the simulation. In contrast, in the NFW algorithm, t_{cpu} decreases continuously with a late-time slope of about $-1/2$.

quench with CPM, then switch over to NFW. While the exact timing of the CPM \rightarrow NFW transition depends on the machine type and coding details, the above discussion suggests that some measure of the system coarseness should be a good predictor. We have found that a mean domain diameter between 2 and 3 predicts the transition reasonably well for system sizes and spin degeneracies similar to those studied here.

At any given simulation time in an NFW simulation with two-level binning, t_{cpu} should be proportional to $N(N^{1/2} + z^2)$ [Eq. (29)]. Figure 6 plots $t_{\text{cpu}} / [N(N^{1/2} + z^2)]$ versus simulation time t for a number of different NFW system sizes. The curves collapse reasonably well to a single curve with a late-time slope of $-1/2$, as expected. Despite the generally excellent agreement with the scaling relation in Eq. (29), it does appear that NFW is slightly more efficient for larger systems than for smaller ones. This can be attributed to two system size effects. First, a relatively larger proportion of computing time is spent performing constant-time operations in small systems; that is, the constants in Eq. (28) contribute significantly to the total computing time in small systems. Second, in small systems, the largest domains are truncated from the domain size distribution, so larger systems have more large domains with small perimeter-to-area ratios. Thus large systems are expected to have a slightly smaller activity per site, thus a relatively smaller computation time, than small systems.

Equation (29) indicates that t_{cpu} should vary inversely with the spin state degeneracy (actually, with $Q - 1$). Figure 7 plots $(Q - 1)t_{\text{cpu}}$ versus t for a variety of spin degeneracies. At late times, all the curves collapse to a single curve with a slope of $-1/2$, as expected. It is interest-

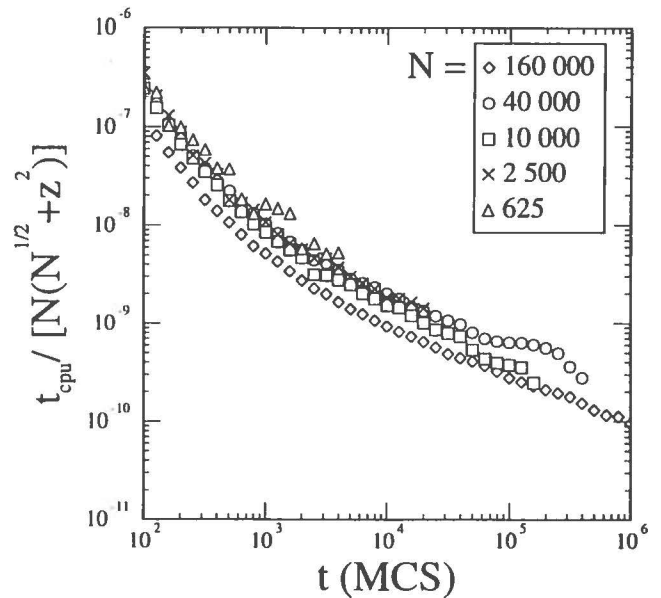


FIG. 6. The NFW computing time per simulation timestep t_{cpu} scaled by the system size factor [Eq. (29)] for a range of N . The curves collapse reasonably well to a single curve with a late-time slope of $-1/2$. NFW is slightly more efficient for larger systems than for smaller ones because constant-time operations are a larger proportion of the total computing time in smaller systems.

ing to note that since the rate of domain growth is inversely proportional to Q , the computing time required to achieve a given domain size is insensitive to Q . This is advantageous in a practical sense, since simulations with large Q more faithfully model certain physical systems.

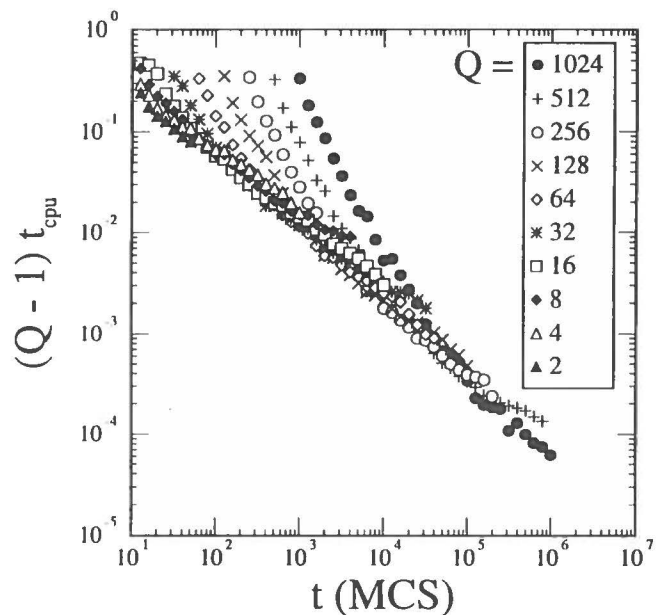


FIG. 7. The NFW computing time per simulation timestep t_{cpu} scaled by the spin state degeneracy factor $(Q - 1)$ for a range of degeneracies Q . At late times, all the curves collapse to a single curve with a slope of $-1/2$. Since the rate of domain growth is generally inversely proportional to Q , the computing time required to achieve a given domain size is insensitive to Q .

Predictions regarding the temperature dependence of the finite temperature n -fold way algorithm require detailed information about the temperature dependence of the nucleation of domains. When the nucleation rate is high, the equilibrium concentration of single-site domains is high, and since single-site domains can undergo any spin flip with high probability, the single-site domain wild flip probabilities dominate and inflate the system activity. In that case, the n -fold way time increment remains small throughout the simulation, and the simulation progresses slowly. In contrast, for temperatures at which the nucleation rate is slow, the total wild flip activity is small compared to the system activity. In that case, the simulation will tend to proceed more quickly. Figure 8 shows t_{cpu} versus t for a number of different simulation temperatures. Although the efficiencies of simulations at low temperatures ($kT/E_0 \leq 0.3$ or $T \leq T_c/2$) are very similar, simulations at higher temperatures are much less efficient. It is nevertheless impressive that even for quenches to relatively high temperatures, the n -fold way algorithm offers a significant advantage over conventional Monte Carlo techniques.

VI. CONCLUSIONS

(1) An efficient serial algorithm for finite temperature, quenched Potts model simulations of curvature-driven domain growth has been developed. This algorithm is based on the n -fold way Ising model algorithm developed by Bortz *et al.*¹⁶ which eliminates unsuccessful spin flip attempts *a priori*, so that all flip attempts actually result in domain evolution. In the Potts model n -fold way algo-

rithm, sites are flipped with a frequency proportional to their site activity, defined as the sum of the probability of success for every possible spin flip at that site. Since every flip attempt is successful, the Monte Carlo time increment is rescaled.

(2) Finite temperature efficiency for high-spin degeneracy systems is achieved by utilizing a new, analytical expression for the portion of the site activity due to flips to non-neighbor spin values. Hence, to determine the activity of a site, only flips to the nearest neighbor spin values need be considered individually; all other flips are evaluated in a single expression.

(3) In the conventional Potts model algorithm, the computing time per Monte Carlo timestep (MCS) is proportional to the system size N and the lattice coordination number z , or $t_{\text{cpu}} \propto Nz$, at all simulation times. In contrast, in the n -fold way algorithm, computing time per MCS is given by

$$t_{\text{cpu}} \propto \frac{\langle A \rangle}{Q-1} \cdot (t_{\text{search}} + z \cdot t_{\text{update}}),$$

where $\langle A \rangle$ is the average site activity in the system, Q is the spin degeneracy, t_{search} is the computing time to search for the site to flip, and t_{update} is the computing time to update the activities of the site and its neighbors after the flip. In systems that undergo domain coarsening, $\langle A \rangle$ decreases as domain growth progresses, so the efficiency of the n -fold way algorithm increases as a simulation progresses. For instance, in normal, single-phase domain evolution discussed in this paper, t_{cpu} decreases as simulation time t increases, such that

$$t_{\text{cpu}} \propto \frac{N}{(Q-1)t^{1/2}} \cdot (t_{\text{search}} + z \cdot t_{\text{update}}).$$

Because of the inverse dependence of t_{cpu} on simulation time, the n -fold way will always be more efficient than the conventional algorithm for coarsening systems at late simulation times, providing the search and update routines are chosen intelligently.

(4) A number of binning schemes may be utilized to minimize t_{search} and t_{update} . If $N^{1/2}$ is much larger than z^2 , then t_{search} and t_{update} are optimized by a binary search scheme, and the computing time per MCS scales as

$$t_{\text{cpu}} \propto \frac{z^2 N \log_2 N}{(Q-1)t^{1/2}}.$$

In contrast, if z^2 is on the order of $N^{1/2}$, a two-level binning system is more efficient. In that case, computing time per MCS scales as

$$t_{\text{cpu}} \propto \frac{N}{(Q-1)t^{1/2}} \cdot (N^{1/2} + z^2).$$

(5) Computer experiments confirm the statistical equivalence of the conventional and n -fold way algorithms for domain evolution. In addition, the computing time per MCS depends on system parameters N , z , t , and Q as predicted by the time complexity analysis.

(6) The n -fold way algorithm retains maximum efficiency for system temperature $T \leq 0.5T_c$. However, even at higher fractions of the critical temperature, the n -fold way is still much more efficient than the conventional algorithm.

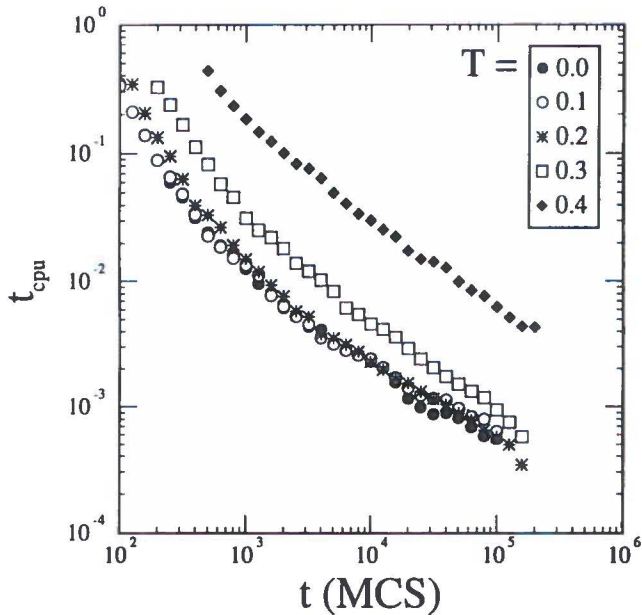


FIG. 8. The NFW computing time per simulation timestep t_{cpu} for different simulation temperatures. Although the computing efficiencies of simulations at low temperatures ($kT/E_0 \leq 0.3$ or $T \leq T_c/2$) are very similar, simulations at higher temperatures are much less efficient. However, the n -fold way algorithm offers a significant advantage over conventional Monte Carlo techniques even at high temperatures.

ACKNOWLEDGMENTS

GNH thanks GMI for support under a research initiation grant. EAH's work has been supported by a National Science Foundation Graduate Research Fellowship and by the IBM Predoctoral Fellowship in Scientific Computing. Both authors wish to thank David J. Srolovitz for valuable discussions.

REFERENCES

1. D. Weiare and J. P. Kermode, *Philos. Mag.* B 47, L29 (1983); 48, 245 (1983); 50, 379 (1984).
2. H. J. Frost and C. V. Thompson, in *Computer Simulation of Microstructural Evolution* (The Metallurgical Society, Warrendale, PA, 1986), p. 33; H. J. Frost, C. V. Thompson, C. L. Howe, and J. Whang, *Scripta Metall.* 22, 65 (1988).
3. K. Kawasaki, T. Nagai, and K. Nakashima, *Philos. Mag.* B 60, 1399 (1989).
4. A. Soares, A. C. Ferro, and M. A. Forres, *Scripta Metall.* 19, 1491 (1985).
5. M. P. Anderson, G. S. Grest, and D. J. Srolovitz, *Philos. Mag.* B 59, 293 (1989).
6. G. S. Grest, D. J. Srolovitz, and M. P. Anderson, *Phys. Rev.* B 38, 4752 (1988).
7. M. P. Anderson, D. J. Srolovitz, G. S. Grest, and P. S. Sahni, *Acta Metall.* 32, 783 (1984); D. J. Srolovitz, M. P. Anderson, P. S. Sahni, and G. S. Grest, *ibid.* 32, 793 (1984); D. J. Srolovitz, M. P. Anderson, G. S. Grest, and P. S. Sahni, *ibid.* 32, 1429 (1984); G. S. Grest, D. J. Srolovitz, and M. P. Anderson, *Acta* 33, 509 (1985); D. J. Srolovitz, G. S. Grest, and M. P. Anderson, *ibid.* 33, 2233 (1985).
8. J. W. Cahn, E. A. Holm, and D. J. Srolovitz, *Proceedings of the International Conference on Grain Growth in Polycrystalline Materials* (Trans Tech Publications, Brookfield, VT, 1991).
9. E. A. Holm, D. J. Srolovitz, and J. W. Cahn, "Microstructural Evolution in Two-Dimensional Two-Phase Polycrystals," to be published.
10. E. A. Holm and D. J. Srolovitz, in *Morris E. Fine Symposium*, edited by P. K. Liaw *et al.* (TMS, Warrendale, PA, 1991), pp. 187-192.
11. E. A. Holm, J. A. Glazier, D. J. Srolovitz, and G. S. Grest, *Phys. Rev.* A 43(6), 2662-2668 (1991).
12. Gary S. Grest, James A. Glazier, Michael P. Anderson, Elizabeth A. Holm, and David J. Srolovitz, *Mat. Res. Soc. Symp. Proc.* (1991).
13. A. D. Rollett, D. J. Srolovitz, and M. J. Luton, in *Morris E. Fine Symposium*, edited by P. K. Liaw *et al.* (TMS, Warrendale, PA, 1991), pp. 111-118.
14. G. N. Hassold, I-W Chen, and D. J. Srolovitz, *J. Am. Ceram. Soc.* 73, 2857 (1990); I-W Chen, G. N. Hassold, and D. J. Srolovitz, *ibid.* 73, 2865 (1990).
15. P. S. Sahni, D. J. Srolovitz, G. S. Grest, M. P. Anderson, and S. A. Safran, *Phys. Rev.* B 28(5), 2705-2716 (1983).
16. A. B. Bortz, M. H. Kalos, and J. L. Liebowitz, *J. Comp. Phys.* 17, 10-18 (1975).
17. M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods Volume I: Basics* (Wiley, New York, 1986), pp. 136-140.