Journal of Statistical Physics, Vol. 63, Nos. 1/2, 1991

A Parallel Multigrid Algorithm for Percolation Clusters

R. C. Brower,[1] Pablo Tamayo,[2] and Bryant York[3]

Received March 6, 1990, final November 7, 1990

A new parallel cluster-finding algorithm is formulated by using multigrid relaxation methods very similar to those used for differential equation solvers. For percolation clusters, this approach drastically reduces critical slowing down relative to local or scan relaxation methods. Numerical studies of scaling properties with system size are presented in the case of the 2D percolation clusters of the Swendsen–Wang Ising dynamics running on the Connection Machine.

KEY WORDS: Multigrid; Monte Carlo method; percolation; cluster labeling; accelerated dynamics.

1. INTRODUCTION

The rapid identification of geometrical or topological structure is important in many problems of computational physics and image processing. Here we consider the identification of percolation clusters(1) (connected components of an undirected graph), which are the appropriate extended geometrical structures (or collective coordinates) being exploited in a new generation of very efficient Monte Carlo algorithms.(2–5)

In particular, we want to develop cluster-finding algorithms which are efficient on massively parallel computers such as the Connection Machine.

We present a new approach to parallel cluster finding based on combining traditional relaxation techniques for optimization problems

[1] Department of Electrical, Computer and Systems Engineering, and Physics Department, Boston University, Boston, Massachusetts 02215.

[2] Thinking Machines Corporation, Cambridge, Massachusetts 02142, and Physics Department, Boston University, Boston, Massachusetts 02215.

[3] Computer Science Department, Boston University, Boston, Massachusetts 02215.

and nonlocal multigrid methods, which often circumvent critical slowing down. Our approach utilizes distance doubling to construct a series of progressively nonlocal operators to approximate portions of the connectivity matrix. The operators are applied in a nontelescoping multigrid fashion to derive cluster connectivity information. In order to give a specific physical example, much of our discussion and numerical analysis is based on the Swendsen–Wang(2) algorithm for the two-dimensional Ising model. We provide a formalism for describing algorithms of this type, and some empirical results for our Connection Machine implementation.

2. ISING PERCOLATION CLUSTER PROBLEM

Our cluster methods are designed to have good mean performance for a class of random clusters embedded in a regular grid. For example, we can consider graphs G generated by the random (Bernoulli) percolation of links (bonds) with probability p, which gives rise to the distribution

$$P_1(G) = (1/Z_1)[p/(1-p)]^{N(G)}(1-p)^{dN} \tag{1}$$

for a graph G with $N(G)$ percolated links out of the total of dN links on a d-dimensional lattice.(1) (Of course, the graph G is generally a collection of many connected components.) Our algorithms will attempt to label the connected components of the random graph in a small mean time,

$$\langle T(G)\rangle = \sum_G P(G)\, T(G) \tag{2}$$

The mean performance, rather than worst case performance, is the appropriate measure of efficiency for simulations in statistical mechanics.

To study a concrete example, which requires labeling random clusters as an inner loop, we consider the Swendsen–Wang algorithm for the Ising model (or q-state Potts model with $q=2$). The Hamiltonian for the Ising model is

$$H_{\text{Ising}} = \tfrac{1}{2} \sum_{\langle i,j\rangle} (1 - s_i s_j) \tag{3}$$

where the sum is taken over the bonds $\langle i,j\rangle$ on a finite periodic d-dimensional hypercubic lattice with $N = L^d$ sites and spins $s_i = \pm 1$. The Swendsen–Wang algorithm(2–4) for sampling the equilibrium distribution, $P(s_i) = Z^{-1}\exp(-\beta H_{\text{Ising}})$ consists of two stages: a percolation process on each bond with probability $p_{ij} = 1 - \exp[-\beta(1 + s_i s_j)]$, followed by a random flip of all spins subject to the constraint that $s_i = s_j$ on all percolated (or "occupied") bonds.

Moreover, it can be shown in general that the Swendsen–Wang dynamics for the q-state Potts model gives rise to percolation clusters[5–7] on graphs with the following probability distribution:

$$P_q(G) = (1/Z_q)\, q^{N_c} [p/(1-p)]^{N(G)} (1-p)^{dN} \tag{4}$$

where N_c is the total number of connected components. Thus, our two examples, percolation ($q = 1$) and Ising ($q = 2$), are special cases of the Fortuin–Kasteleyn random-cluster model.[6] Clearly, the way to satisfy the above percolation constraint ($s_i = s_j$) is to identify each cluster (i.e., each set of sites which are connected by one path of occupied bonds) and set all the spins s_i on each cluster randomly to $+1$ or -1. Simulations have shown that if there is a rapid, $O(N)$ method for flipping the cluster spins, the resulting dynamics has very short relaxation times: $\tau \sim 12$ for Swendsen–Wang[2] relative to $\tau \sim 8 \times 10^6$ for heat bath on a 1024×1024 lattice.[16]

On a serial machine it is easy to implement a cluster-finding algorithm with a running time of order N. This can be done with either a breadth-first or depth-first search.[21] For example, consider a given enumeration of sites, $i = 1, 2, \ldots, N$, and randomly chosen integer labels $c(i)$. Select the first site with label $c(1)$ as the root of its cluster (tree), then relabel all the connected sites with the root label $[c(i) = c(1)]$, and continue relabeling other sites in the cluster (descending down the tree) until exhaustion. Every relabeled site in the cluster is deleted (marked "off") from the site enumeration list. Then a new "on" site is selected and the process is repeated until a new cluster is labeled and so on. The process stops when all clusters have been labeled. The standard Hoshen–Kopelman algorithm[8] has an empirical running time of order N for percolation on regular lattices; its worst-case time in this problem is unknown (but can be of order N^2 for more pathological graphs). Variants of the Hoshen–Kopelman algorithm can achieve a worst-case run-time of order $N \log N$ or even smaller.

3. THE RELAXATION APPROACH TO CLUSTER LABELING

An alternative approach to cluster labeling can be based on finding the fixed point of a parallel iterative process. Let us introduce a connectivity matrix Δ_{ij}, which is 1 if the percolation bond between sites i and j is occupied and zero otherwise. The parallel iterative process is

$$c'(i) = \mathrm{MIN}\{c(j) \mid (1 + \Delta)_{ij} \neq 0\} \tag{5}$$

where the labels $c(i)$ are initialized to distinct values [e.g., $c(i) = i$]. Note that on the right-hand side, we have the sum of the connectivity matrix Δ_{ij} and the unit matrix (δ_{ij}). Thus, in each iteration the label $c(i)$ on each site

i is replaced by the minimum of itself and all the labels $c(j)$ of its connected neighbors. We will refer to this as the "local diffusion" algorithm. The local diffusion algorithm is relatively fast, easy to implement, and could be the best choice for certain applications. Unfortunately, as we will see in Section 4, it is not very efficient to label fractal structures such as the Ising critical clusters. The goal of our multigrid algorithm will be to introduce a nonlocal mechanism to improve on the local relaxation of Eq. (5).

The virtue of the relaxation formulation of the cluster problem is that we immediately see how it is analogous to many other iterative matrix problems. In fact, if we interpret addition and multiplication in the Boolean sense ($+ = \mathrm{OR}$, $* = \mathrm{AND}$), repeated iterations of Eq. (5) correspond to matrix products on the Boolean matrices $\Delta_{ij} = 0, 1$. Namely, given two successive iterations, $c'(i) = \mathrm{MIN}\{c(j) \mid (1+\Delta)_{ij} \neq 0\}$, followed by $c''(i) = \mathrm{MIN}\{c'(j) \mid (1+\Delta')_{ij} \neq 0\}$, we obtain

$$c''(i) = \mathrm{MIN}\{c(j) \mid ((1+\Delta') * (1+\Delta))_{ij} \neq 0\} \tag{6}$$

Thus the cluster labeling problem may be viewed as a problem in linear algebra on the integer function $c(i)$. It is just the problem of finding the transitive closure of a random undirected graph embedded in a regular lattice, alternatively referred to as finding the connected components of an undirected graph.(21) Both multigrid concepts and the Boolean formulation have been considered for other closely related problems.(12 14)

Before describing our algorithm, let us remark that for the application to the Swendsen–Wang spin update, we do not necessarily have to solve the cluster-finding problem. It only requires solving a closely related relaxation problem for the Ising spins s_i themselves. Namely, finding the new spin state is equivalent to finding a random state among all of those that minimize the "percolation energy" for a dilute Ising system,

$$H_{\mathrm{perc}} = \tfrac{1}{2} \sum_{\langle i,j \rangle} \Delta_{ij}(1 - s_i s_j) \tag{7}$$

So, in principle, it may be possible to bypass the cluster-labeling problem altogether and to find directly (by iteration) a random member of the degenerate ground states of H_{perc}. We are investigating whether such relaxation shemes can also lead to efficient algorithms for parallel Monte Carlo simulations.

4. THE MULTIGRID CLUSTER-FINDING METHOD

To motivate our approach, note that the full connectivity matrix

$$M_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ lie in the same cluster} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

can be written as

$$\begin{aligned} M &= \underbrace{(1+\Delta) * \cdots * (1+\Delta)}_{F(G)\text{ times}} \\ &\equiv (1+\Delta)^{F(G)} \end{aligned} \tag{9}$$

where $F(G)$ is the smallest integer such that $M * (1+\Delta) = M$ (transitive closure). Clearly, $F(G)$ is the maximum distance between two points in the same percolation cluster (the distance between two sites is defined as the maximum length of a path connecting them and consisting of percolated bonds). In graph language, $F(G)$ is the maximum depth of the embedded trees in all the clusters. This is related to the percolation theory exponent $\delta_{\min}$, which is defined as the fractal dimension of the minimum distance between two points on the same percolation cluster. For random percolation ($q=1$) the value of $\delta_{\min}$ is reported by Herrmann and Stanley[15] to be 1.130 ± 0.002 for $d=2$ and 1.34 ± 0.01 for $d=3$.

The "local diffusion" algorithm described in Section 3 forms the connectivity matrix M by repeated multiplication with the matrix $(1+\Delta)$; obviously, this takes $F(G)$ iterations. In order to reduce the number of iterations required, we can consider a "nonlocal diffusion" algorithm of the following form: let $\Delta^{(0)} = \Delta$, and let $\Delta^{(1)}, \Delta^{(2)},..., \Delta^{(n)}$ be *arbitrary* Boolean matrices satisfying $0 \leqslant \Delta^{(i)} \leqslant M$ (we will discuss later how they are to be chosen). Then it is easy to see that

$$M = [(1+\Delta^{(n)}) * \cdots * (1+\Delta^{(1)}) * (1+\Delta^{(0)})]^K \tag{10}$$

for some integer K: certainly this holds for $K = F(G)$, but if we are lucky it may hold for much smaller values of K as well. It follows that the desired cluster labeling can be obtained by an iterative algorithm similar to (5), in which the matrix Δ is replaced by the cyclic sequence $\Delta^{(0)}, \Delta^{(1)},..., \Delta^{(n)}, \Delta^{(0)}, \Delta^{(1)},..., \Delta^{(n)},...$. This algorithm must run K full cycles, or $K(n+1)$ steps in all. The hope is that with a suitable choice of the $\Delta^{(1)}, \Delta^{(2)}, \Delta^{(3)},..., \Delta^{(n)}$ we may achieve $K(n+1) \ll F(G)$.

For graphs which are subgraphs of a regular lattice, it is convenient to choose the matrices $\Delta^{(1)}, \Delta^{(2)},..., \Delta^{(n)}$ to have a "multigrid" structure: namely in $(1+\Delta^{(l)})$ only those matrix elements which reach out a distance 2^l in one of the coordinate directions are allowed to be nonzero. These matrix elements correspond to "nearest neighbors" on the lth coarse grid. (Such a regular truncation is particularly attractive on the Connection Machine, since it allows us to use the fast "power-of-two" NEWS communication primitives.) The number of multigrid levels is then $l_{\max} = \log_2 L - 1$. Figure 1a shows some of the multigrid $\Delta^{(l)}$'s. The matrix

$$D = (1+\Delta^{(l_{\max})}) * \cdots * (1+\Delta^{(1)}) * (1+\Delta^{(0)}) \tag{11}$$

corresponds to a single downward (fine-to-coarse) traversal of the levels, while the matrix

$$U = (1 + \Delta^{(0)}) * (1 + \Delta^{(1)}) * \cdots * (1 + \Delta^{(l_{\max})}) \tag{12}$$

corresponds to a single upward (coarse-to-fine) traversal. The coarsest level is $l_{\max} = \log_2 L - 1$. A single "V-cycle" $V = U * D$ is composed of a downward traversal followed by an upward traversal.

The procedure for choosing the matrices $\Delta^{(l)}$ must now be specified. In principle they can be any Boolean matrices satisfying $0 \leqslant \Delta^{(l)} \leqslant M$ along with the sparsity constraints. Thus the *marginal* choice of $\Delta^{(l)}$ is the truncation of M to bonds of length 2^l in the coordinate directions:

$$\Delta_{ij}^{(l)} = \begin{cases} M_{ij} & \text{if } i-j \text{ is a distance } \pm 2^l \text{ in a coordinate direction} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

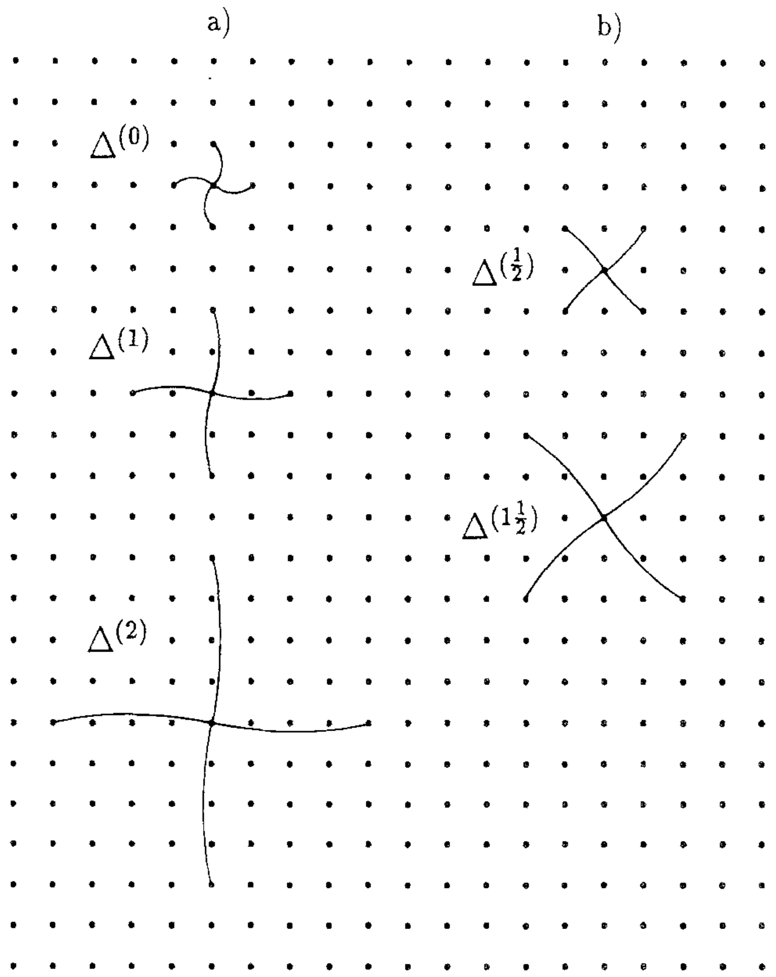


Fig. 1. (a) Multigrid connectivity matrices $\Delta^{(l)}$ connect neighbors at powers-of-two distances along coordinate axes. (b) The connectivity matrices for half-integer levels improve convergence by connecting neighbors along diagonals.

This choice of $\Delta^{(l)}$ is obviously "optimal" in the sense that it minimizes the required number K of iterations. On the other hand, it seems impossible to compute these "ideal" $\Delta^{(l)}$ without first solving the full cluster-finding problem (i.e., computing M). The other choice is to define the $\Delta^{(l)}$ by repeated squaring:

$$\Delta_{ij}^{(l)} = \begin{cases} (\Delta^{(l-1)} * \Delta^{(l-1)})_{ij} & \text{if } i-j \text{ is a distance } \pm 2^l \\ & \text{in a coordinate direction} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

This is simply to define, but the algorithm may converge rather slowly. We shall make an intermediate choice as a compromise between the "ideal" choice (13) and the "repeated squaring" choice (14). Before describing our choice it is important to mention that we found it convenient to improve convergence by defining half-integer levels ($\sqrt{2}$ blocking, corresponding to next to nearest neighbors), in addition to the original integer multigrid levels. Some of these half-integer $\Delta^{(l)}$'s are shown in Fig. 1b. In this way, the hierarchy of connectivity matrices consists of levels: 0, $\frac{1}{2}$, 1, $1\frac{1}{2}$, 2,..., $l_{\max}-\frac{1}{2}$, $l_{\max}$.

We define our truncated matrices $\Delta^{(l)}$ by the iteration schemes shown diagrammatically in Fig. 2a. The diagram establishes that the connectivity bond between two sites (on axes at distance 2^l) is occupied if either the two bonds at the lower level connecting the two sites in a straight line are occupied (level $l-1$), or there is a path of occupied bonds going along

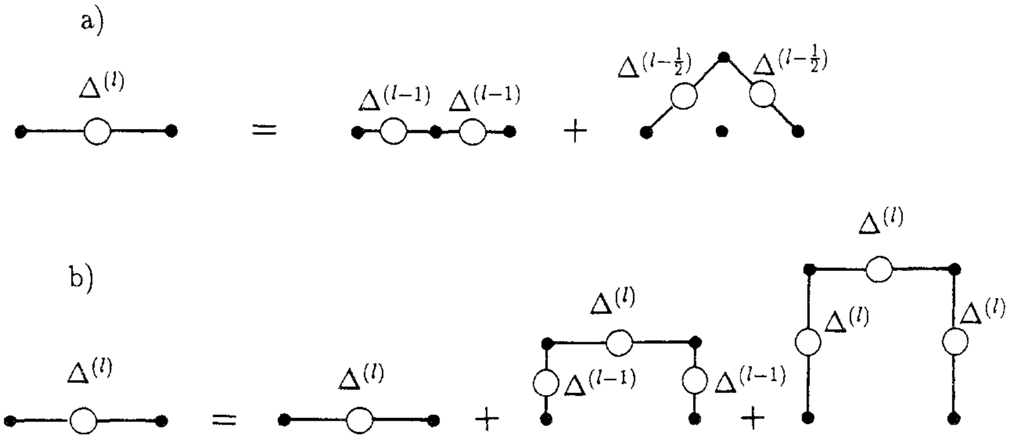

a)
$\Delta^{(l)}$
=
$\Delta^{(l-1)}$ $\Delta^{(l-1)}$
+
$\Delta^{(l-\frac{1}{2})}$ $\Delta^{(l-\frac{1}{2})}$
b)
$\Delta^{(l)}$
=
$\Delta^{(l)}$
+
$\Delta^{(l)}$
$\Delta^{(l-1)}$ $\Delta^{(l-1)}$
+
$\Delta^{(l)}$
$\Delta^{(l)}$ $\Delta^{(l)}$

Fig. 2. Recursive definition of the $\Delta^{(l)}$ connectivity operators, where each higher level $\Delta^{(l)}$ is computed by a single iteration of these relations, starting with the nearest neighbor percolation to define $\Delta^{(0)}=\Delta$ and the boundary conditions, $\Delta^{(1/2)}=\Delta^{(-1/2)}=\Delta^{(-1)}=0$. The last term in the first diagram should be interpreted to include the reflected term corresponding to a path going "down–up." Iterations of the second diagram improve the connectivity by or-ing additional paths to the ones defined by the first diagram. Rotations by multiples of 45 deg generate the diagrams for connectivity along horizontal and diagonal (half-integer) levels.

diagonals (level $l-\frac{1}{2}$). The diagram must be interpreted to include the reflection of the last term, which provides the path going "down–up." Rotations of this diagram by multiples of 45 deg generate all the other diagrams for connectivity along horizontal and diagonal axes (half levels). One way to look at this is to consider that each successive connectivity operator is formed from next to nearest neighbor bonds giving rise to a operator on the next level $l+\frac{1}{2}$ with a lattice rotated by 45 deg. Thus, the recurrence relation for $\Delta^{(l+1)}$ relates it to itself and the two finer levels $\Delta^{(l+1/2)}$ and $\Delta^{(l)}$. This is equivalent to applying an eight-nearest-neighbor connectivity operator at each integer level. In terms of our idealized V-cycle this operator produces L^2-3 factors, which we know are a necessary precondition for identifying fractal clusters in logarithmic time.

Once the $\Delta_{ij}^{(l)}$ are defined, they can be improved by a few iterations of the diagram of Fig. 2b. This diagram improves the already defined $\Delta_{ij}^{(l)}$ by or-ing to it the contributions of additional paths; in this case the "staplelike" paths at levels $l-1$ and l. The more iterations of the improving scheme that are performed, the greater is the number of paths represented by $\Delta_{ij}^{(l)}$. As one would expect, these improvement iterations take considerable time and therefore in practice only a few of them are performed.

Our full multigrid cluster-finding algorithm consists, therefore, of a setup phase in which the matrices $\Delta^{(1)}$, $\Delta^{(1/2)}$, $\Delta^{(2)}$,... are successively constructed, followed by a solution phase in which these matrices are applied iteratively in a V-cycle. The iteration is terminated when a fixed point is found, i.e., when the cluster labels $c(i)$ are completely unchanged in the most recent V-cycle.

To summarize, the multigrid algorithm consists of the following steps:

1. Obtain $\Delta^{(0)}$, the connectivity of the problem at the original scale (e.g., in the Swendsen–Wang algorithm this is the percolation process).
2. For all levels l $(0<l\leqslant l_{\max}=\log_2 L-1)$, compute truncated connectivity matrices $\Delta^{(l)}$ (restricted to distances of 2^l in axes directions) by:
   (a) Defining $\Delta^{(l)}$ based on $\Delta^{(l-1)}$ and $\Delta^{(l-1/2)}$ using the recursive diagram of Fig. 2a.
   (b) Improving $\Delta^{(l)}$ by repeated iterations of the diagram of Fig. 2b.
3. Initialize cluster labels to distinct values: $c(i)=i$.
4. Iterate the following relaxation scheme:

$$c'(i)=\mathrm{MIN}\{c(j)\,|\,(1+\Delta^{(l)})_{ij}\neq 0\} \qquad (15)$$

at each level of the multigrid hierarchy according to a V-cycle schedule:

$$\underbrace{(1+\Delta^{(0)}),\ (1+\Delta^{(1/2)}),\ldots,\ (1+\Delta^{(l_{\max})}),}_{U}$$

$$\underbrace{(1+\Delta^{(l_{\max})}),\ (1+\Delta^{(l_{\max}-1/2)}),\ldots,\ (1+\Delta^{(0)})}_{D}$$

until a fixed point is found.

5. NUMERICAL RESULTS FOR THE MULTIGRID CLUSTER ALGORITHM

Here we report our numerical results and give additional details about the parallel implementation of the algorithm.

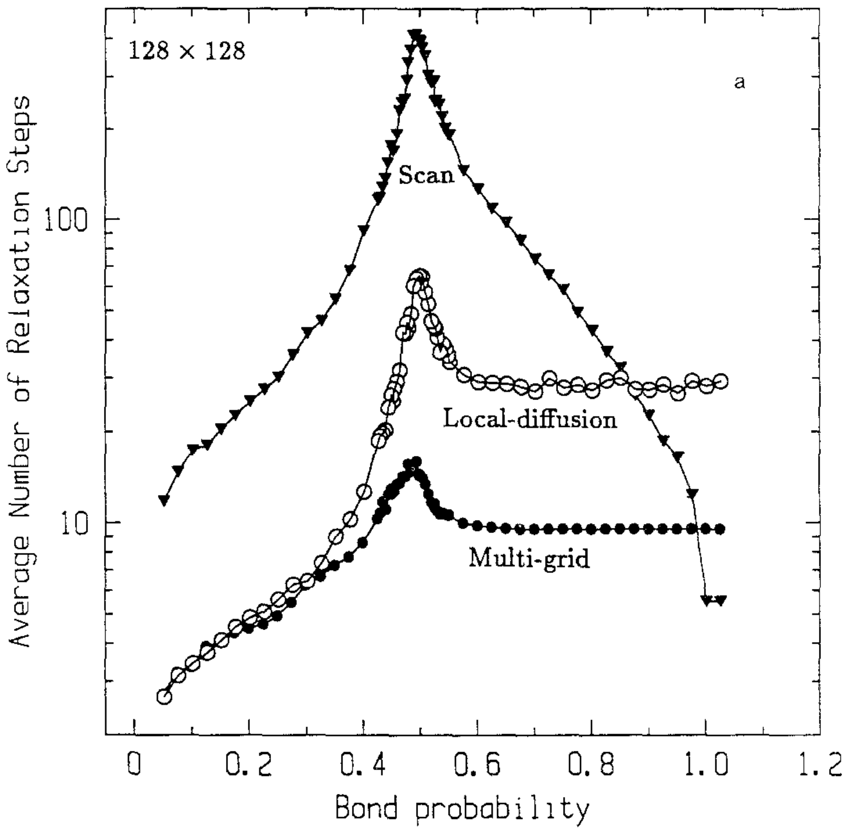
There is a very large parameter space of multigrid methods, which might be tuned for each distribution of graphs $P(G)$ encountered. We have not yet made a very systematic exploration of this space. Instead, we have used intuition and experience to guide our choices.

We implemented our algorithm on the Connection Machine 2 (the CM-2 is a fine-grain SIMD computer). Here each site is assigned to a (real or virtual) processor. Since the multigrid is nontelescoping—that is, all lattice sites belong to all grids—all processors are utilized in every cycle. This contrasts with other parallel schemes on coarse-grain machines in which the processors are assigned to regions of the lattice, stripes, or multisite blocks.[11] Such coarser grain mappings are also possible on the CM-2 by using the "sprint nodes," but we do not explore this option in this paper.[18]

The scaling of our algorithm, as the lattice volume N increases, will be expressed in terms of the number of iterations per processor, where the processor number is assumed to be equal to N. In fact, the Connection Machine has a maximum of $N_p = 65{,}536$ physical processors and it runs more efficiently when the virtual processor ratio N/N_p is on the order of 16 to 32.

Our multigrid code written in C/Paris exploits special Connection Machine instructions (power-of-two NEWS) that provide fast communication with neighbors at a power-of-two distance along the axes. We ran most of our simulation on 8 and 16K processor configurations with the use of the 32K configuration for the largest lattice because of the need for additional memory.

In the multigrid data reported in Figs. 3–5, we used a modified V-cycle, denoted as D' (down + top), which consists of a D downward



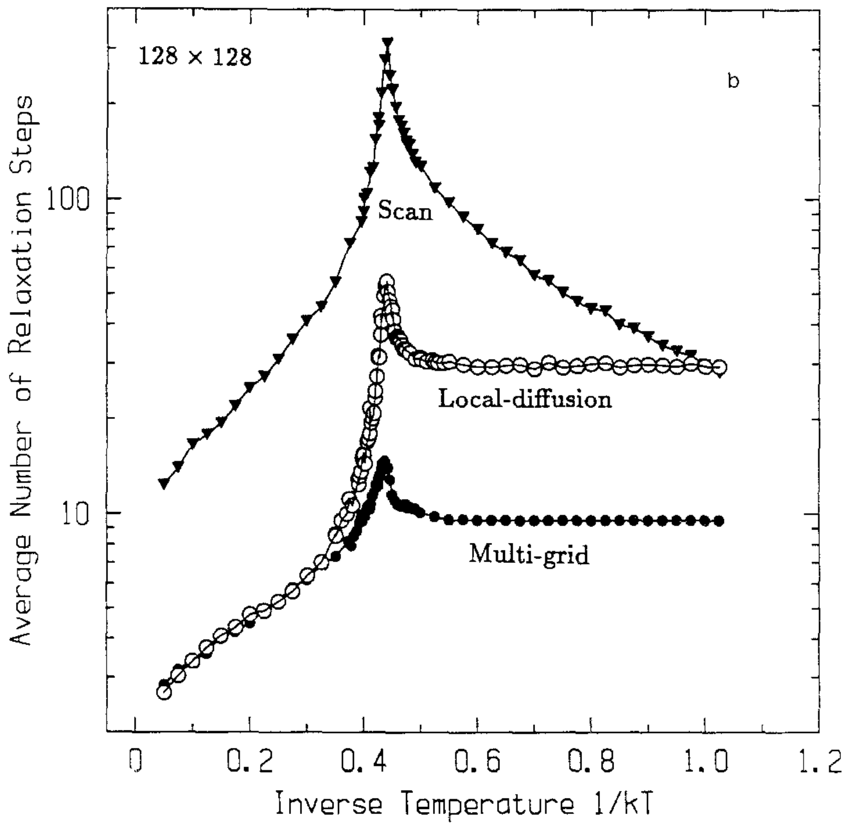


Fig. 3. Average number of relaxation steps for the three algorithms: multigrid, scan, and local diffusion for a 128 × 128 system. (a) The results for $d = 2$ random bond percolation. (b) The results for the $d = 2$ Swendsen–Wang Ising dynamics. In both cases, at $p = 0.5$ (random percolation) or $1/kT = 0.4406868$ (Ising), the clusters are fractal and the algorithms exhibit critical slowing down.

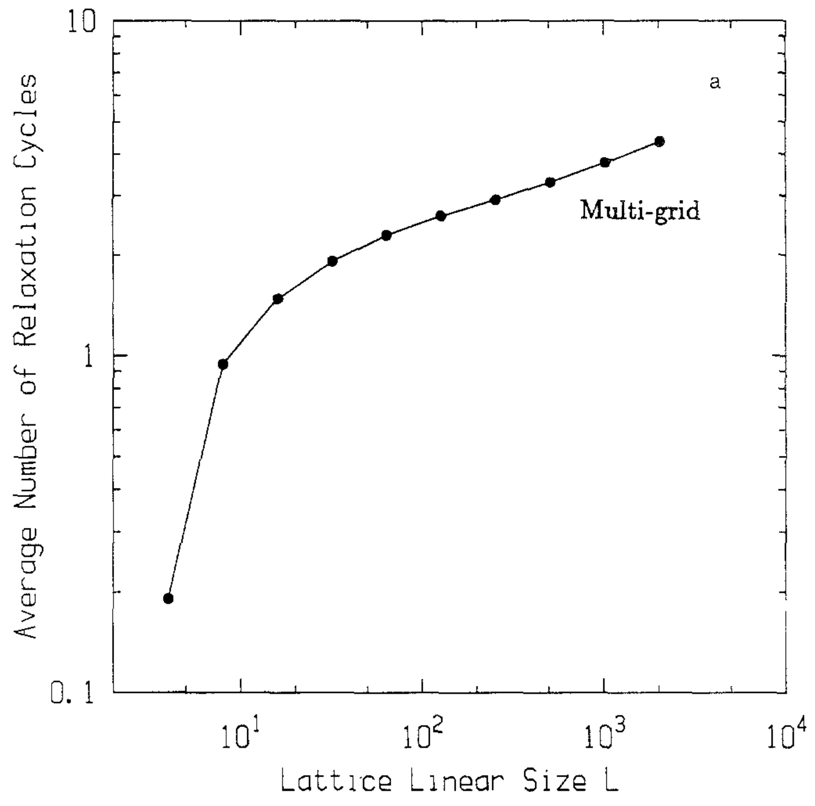
traversal of the levels in which each relaxation step is followed with a hit at the top (level 0). This updating scheme is a little faster than the canonical V-cycle. Additional speedup comes from the fact that we test for completion after each hit at the top instead of waiting for the end of the cycle. For that reason the data of Fig. 5 show fractional cycles.

We study how the number of iterations of Fig. 1b (improvement) affected the overall performance and found that it increased significantly from 0 to 1 iterations and from 1 to 2 iterations, but increased slowly for more than two. Consequently, we choose two iterations as a good compromise.

For comparison purposes we obtained data for the local-diffusion algorithm and for a "scan" algorithm. The scan algorithm is based on the "scan-with-min" operation. It is implemented by a binary tree that moves the minimal cluster label along a single axis of the lattice. In two dimensions the scan algorithm is analogous to the operator splitting technique(17) used in differential equations whereby the operator $1+\Delta$ is split: $1+\Delta = 1+\Delta_x+\Delta_y$, and each iteration alternates between the exact solution of the 1D operators $1+\Delta_x$ and $1+\Delta_y$ along the X and Y axes, respectively. We will contrast this 2D scan with our 2D multigrid, which "interleaves" distance doubling on both axes, and we shall show that the multigrid is much more efficient for highly fractal clusters.

Different schemes will be more or less efficient depending on the cluster statistics actually encountered in a given problem. We study the temperature dependence of the three algorithms: multigrid, local-diffusion, and scan. In the Swendsen–Wang algorithm the cluster distribution is effected by one parameter, the inverse temperature $\beta = 1/kT$. At small β the clusters are small and compact, so that any local algorithm will suffice. But at large β a single large cluster dominates and, as can see in Fig. 3, the multigrid algorithm outperforms local diffusion. In fact, at $\beta = \infty$, it can be proved that one multigrid down D or up U cycle is exact. For intermediate β there is a critical point where the clusters are fractal and all our algorithms experience some critical slowing down. Figure 5 shows the performance of the multigrid algorithm for two different lattice sizes (128^2 and 1024^2). As we see in Figs. 4b and 4c, the multigrid algorithm is much more efficient than the 2D scan algorithm as the system size increases. From the local diffusion data of Fig. 4b we can roughly estimate $b_{\min}$ to be 1.10 ± 0.10, consistent with ref. 15.

At the critical point, all our algorithms show some critical slowing down.(19) However, as is seen in Fig. 4, the multigrid algorithm has much better performance than the purely local algorithm or the scan approach. In spite of taking numerical measurements up to quite large system sizes (2048^2), there is still no clear indication of a simple asymptotic behavior for



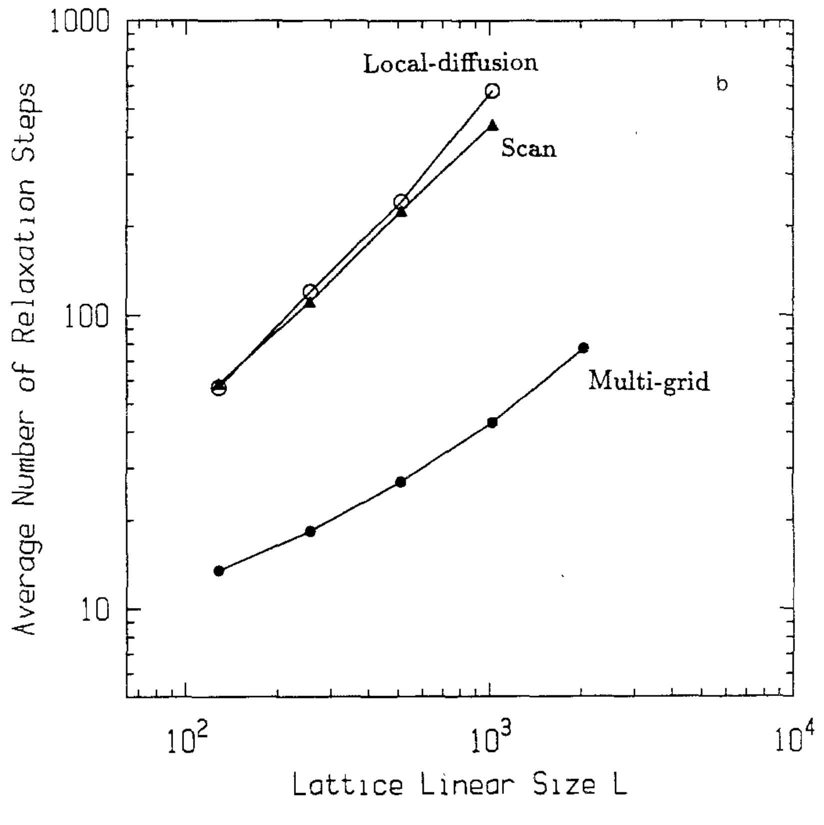


Fig. 4. (a) Average number of relaxation cycles (D') as a function of lattice size for the multigrid algorithm (log-log plot). (b, c) Average number of relaxation steps as a function of system size for the three algorithms: multigrid, scan, and local diffusion. They show the same data, but graph (c) is a semilog plot.

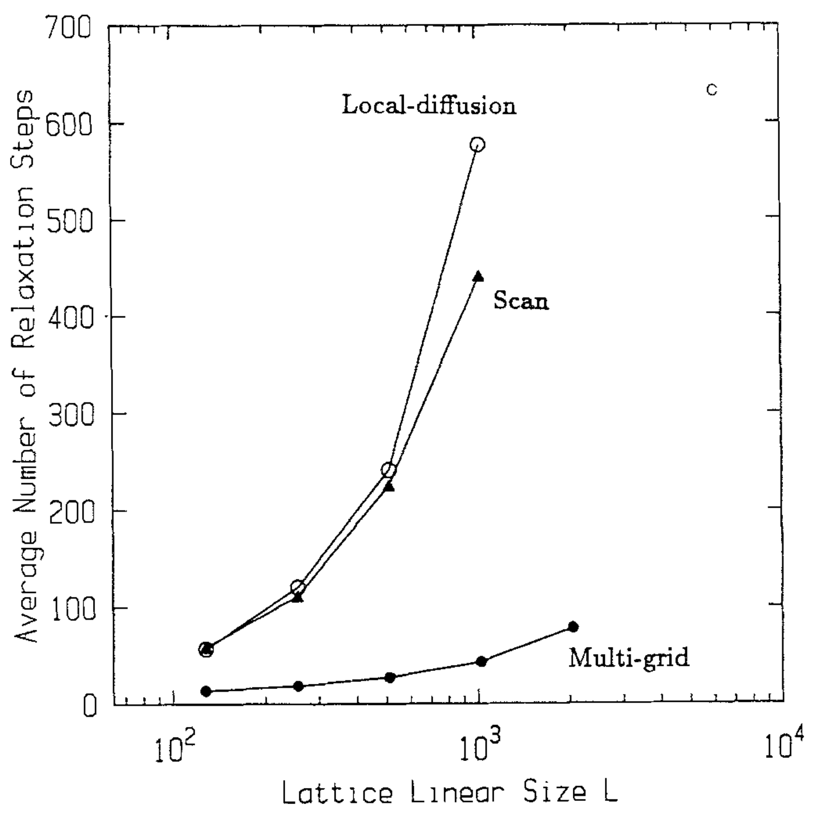


Fig. 4. (*Continued*)

our algorithm. We may be seeing a long intermediate scaling region nearly approximated by $\log N$ followed by a resurgence of a small power behavior. Since our multigrid algorithm is nontelescoping, a single cycle is $\log N$ deep and with logarithmic scaling at best our full multigrid algorithm is $O(N \log^2 N)$.

6. CONCLUSIONS

The main result of our present investigation is to show that the mean performance of the multigrid cluster finder has good scaling properties. Karp[9] has developed a sequential algorithm with $O(N)$ expected performance for the related problem of finding the transitive closure of a random digraph; however, he experiences critical slowing down—i.e., $O(w(N)(N \log N)^{4/3})$ expected time in the critical region [where $w(N)$ is an arbitrary, nondecreasing, unbounded function]. Our algorithm is probably $O(N \log N)$ away from the critical point and somewhat worse at the critical point. For 2D, there are known parallel algorithms with worst case performance matching $O(N \log^2 N)$. It is easy to see that our worst case

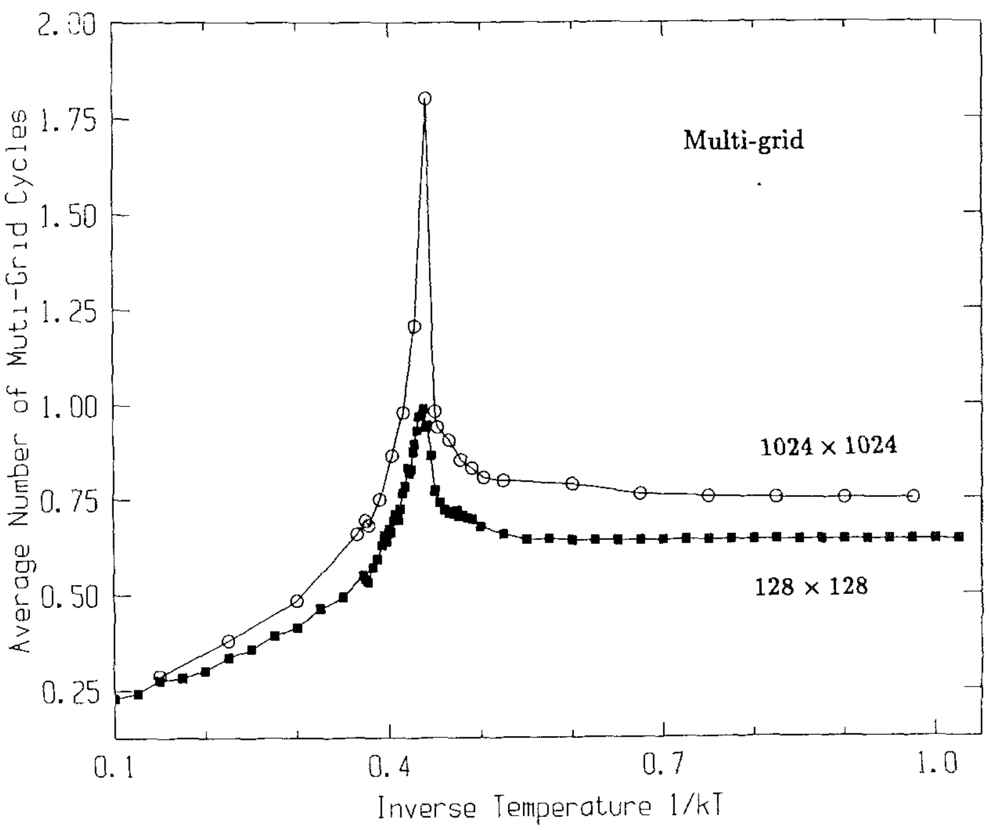


Fig. 5. Average number of relaxation cycles (D') as a function of inverse temperature $1/kT$ for the multigrid algorithm. Two different lattice sizes are shown: 128×128 and 1024×1024. The number of cycles is fractional because we test for completion inside the cycle.

performance for pathological graps is $O(N^2)$, but our purpose was to find good mean performance for realistic systems on real computers.

In terms of absolute performance on the Connection Machine, the update time per spin for our present multigrid code is of the order of 1.5 μsec per spin on a full 64K processor Connection Machine. This is approximately three times as fast as the standard Hoshen–Kopelman scalar codes running on the IBM 3090.[20] In terms of relaxation rates our overall Monte Carlo Ising code is already far superior to heat bath or Metropolis. For example, due to the much shorter autocorrelation times of the Swendsen–Wang algorithm, for a lattice of 1024^2 sites our code produces statistically independent samples at an iteration rate which is about 667,000 times faster than heat bath. Consequently, although the multigrid update time is about 25 times slower than our implementation of the heat bath algorithm on the Connection Machine, it still gives new statistically independent configurations about 27,000 times faster for these lattices. No practical simulation can ignore factors of four orders of magnitude speedup.

If our cluster methods are applied to implementations of field theories(5,22,23) with floating-point intensive Monte Carlo inner loops, the value of our approach would be more dramatic. The basic difficulty with implementation of the pure Ising model is that the bit manipulation of heat bath is replaced in the Connection Machine with the expensive communication of cluster labels of length $\log_2 N$. Implementations on scalar machines "bit pack" several labels into a single word, thus reducing the corresponding memory access overhead and increasing computational speed by treating bits in parallel.

We are investigating more efficient multigrid relaxation codes on the Connection Machine. Algorithms based on pure spin relaxation which attempt to avoid altogether the communication of cluster labels are being studied. In addition, special stencil operators which exploit the "slicewise" architecture of the Connection Machine 2 could improbe the efficiency of the multigrid cluster-finding code. However, already the multigrid approach is proving to be a practical way to marry these extremely efficient cluster acceleration methods to a massively parallel SIMD architecture.

ACKNOWLEDGMENTS

We would like to acknowledge many useful conversations with R. Edwards, R. Giles, R. Kotiuga, T. Ray, A. Sokal, B. Klein, J. Mesirov, J. Ph. Brunet, B. Boghosian, A. Burkitt, D. Heermann, and D. Stauffer. We are indebted to both referees for their many useful suggestions.

REFERENCES

1. D. Stauffer, *Introduction to Percolation Theory* (Taylor and Francis, London, 1984).
2. R. Swendsen and J. S. Wang, *Phys. Rev. Lett.* **58**:86 (1987).
3. W. Klein, T. Ray, and P. Tamayo, *Phys. Rev. Lett.* **62**:163 (1989).
4. P. Tamayo, R. C. Brower, and W. Klein, *J. Stat. Phys.* **58**:1083 (1990).
5. R. G. Edwards and A. D. Sokal, *Phys. Rev. D* **38**:2009 (1988); A. Sokal, Monte Carlo methods in statistical mechanics: Foundations and new algorithms, lecture notes.
6. C. M. Fortuin and P. W. Kasteleyn, *Physica* **57**:536 (1972); P. W. Kasteleyn and C. M. Fortuin, *J. Phys. Soc. Japan Suppl.* **26s**:11 (1969).
7. A. Coniglio and W. Klein, *J. Phys. A* **13**:2775 (1980).
8. J. Hoshen and R. Kopelman, *Phys. Rev. B* **14**:3438 (1976).
9. R. M. Karp, The transitive closure of a random digraph, TR-89-047, August 1989, International Computer Science Institute, Berkeley, California.
10. J. Woo and S. Sahni, Hypercube computing: Connected components, *J. Supercomputing* **3**:209–234 (1989).
11. A. N. Burkitt and D. W. Heermann, *Comp. Phys. Comm.* **54**:201 (1989).
12. A. Hansen and S. Roux, *J. Phys. A* **20**:L873 (1987).
13. A. Stoll, *J. Phys. Cond. Matter* **1**:6959 (1989).

14. D. Kandel, E. Domany, D. Ron, A. Brandt, and E. Loh, Jr., *Phys. Rev. Lett.* **60**:1591 (1988).
15. H. J. Herrmann and H. E. Stanley, *J. Phys. A* **21**:L829 (1988).
16. N. Ito, M. Taiji, and M. Suzuki, *J. Phys.* (Paris) **49**:C8-1397 (1988).
17. W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes* (Cambridge University Press, 1988).
18. H. D. Simon, ed., *Scientific Applications of the Connection Machine*, (World Scientific, Singapore, 1989).
19. P. C. Hohenberg and B. Halperin, *Rev. Mod. Phys.* **49**:435 (1977).
20. J. S. Wang, Private communication.
21. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms* (MIT Press, 1990); D. Knuth, *The Art of Computer Programming*, Vol. 3 (Addison-Wesley, 1973).
22. R. Brower and P. Tamayo, *Phys. Rev. Lett.* **62**:1087 (1989).
23. U. Wolff, *Phys. Rev. Lett.* **60**:1461 (1988).