Journal of Statistical Physics, Vol. 21, No. 5, 1979

Percolation and Cluster Distribution. III. Algorithms for the Site–Bond Problem

J. Hoshen,[1] P. Klymko,[1] and R. Kopelman[1]

Received March 26, 1979; revised May 26, 1979

Algorithms for estimating the percolation probabilities and cluster size distribution are given in the framework of a Monte Carlo simulation for disordered lattices for the generalized site–bond problem. The site–bond approach is useful when a percolation process cannot be exclusively described in the context of pure site or pure bond percolation. An extended multiple labeling technique (ECMLT) is introduced for the generalized problem. The ECMLT is applied to the site–bond percolation problem for square and triangular lattices. Numerical data are given for lattices containing up to 16 million sites. An application to polymer gelation is suggested.

KEY WORDS: Percolation zone; Monte Carlo; site–bond; tree; UNION operation; FIND operation.

1. INTRODUCTION

The concept of percolation has been useful in describing a variety of physical, chemical, and biological phenomena.[1–3] Two distinct types of percolation processes are recognized. These are bond percolation and site percolation. Permeation of fluids through porous media[4] and gel formation by polymers via cross-linking[5] can be explained in terms of the bond percolation theory,[1] whereas crystal phenomena, such as spontaneous magnetization of dilute ferromagnets,[6] diffusion in binary alloys,[7,8] and exciton percolation in molecular crystals,[9,10] are described within the framework of site percolation. Site and bond percolation processes have both been suggested for electrical conductivity models of disordered materials.[11–14]

The calculation of real lattice percolation parameters, such as percolation threshold, percolation probabilities, and cluster size distribution, proved not

Supported by NSF Grant DMR76-07832 and NIH Grant NS08116-9.

[1] Department of Chemistry, University of Michigan, Ann Arbor, Michigan.

0022-4715/79/1100-0583$03.00/0

to be a trivial task.[15,16] Percolation problems of tree lattices have been solved analytically,[17] so that exact results are available for these lattices. Real lattices contain clusters with cyclic components[3,5] which are absent from the tree lattices. The existence of irregular ring structures in real lattices precludes an analytical solution for the percolation problems of real lattices.

Real lattices are approached by two basically different methods. (a) Series expansion. This method is based on series expansion in terms of site or bond probabilities[18] for clusters containing up to 20 sites. The results of the series expansion method can be extrapolated to determine the critical percolation threshold. (b) Monte Carlo simulation. In this method a finite crystal is simulated on the computer. The cluster size distribution, the percolation probability, and the percolation threshold (critical site or bond occupation probabilities) are estimated[19–25] for the simulated lattice.

Both the series expansion method and the Monte Carlo simulation have shortcomings. The series expansion method is not applicable for large clusters, because of the rapid increase in the number of geometrical cluster shapes as the cluster size increases. The series expansion method is used for short-range interactions and for regular lattices. An extension of the method to irregular structures and long-range interactions would not be a simple task and probably not practical. The Monte Carlo simulation, on the other hand, is relatively easy to apply to regular lattice structures, and can be also used on irregular structures which are described in terms of the continuous percolation theory.[26,27] The Monte Carlo simulation is applicable on either side of the percolation threshold. The major deficiency of the Monte Carlo approach is related to the very nature of the procedure, which can only provide an estimate for the percolation probabilities and the cluster size distribution rather than actually calculate them. Increasing the sample size may improve the accuracy and the statistics of the simulation results, but may also be very costly in terms of computer time and space. A careful construction of efficient computer algorithms is of prime importance for treating large samples.[25,28] An ill-chosen algorithm may be impractical for application to a large system because of inadequate time and space complexities.[29]

In this paper we shall introduce algorithms for the generalized site–bond problem.[26,30] This generalization is useful when a percolation process cannot be exclusively described as a pure bond or a pure site percolation process.[30] The site–bond problem can be effectively treated by utilizing an extension of the cluster multiple labeling technique introduced in a recent paper[25] (I). The extended cluster multiple labeling technique (ECMLT) is applied in conjunction with a Monte Carlo simulation of a random lattice. The ECMLT determines the cluster size distribution from which the pertinent percolation parameters can be estimated.[25]

Dean[20] proposed a scheme where the bond problem for a crystal structure could be transformed to a site problem of another crystal structure (usually more complex). This seemed to be a convenient method, as it appeared that it would be simpler to simulate the site percolation problem rather than the bond problem on a computer. The ECMLT can handle the site, the bond, and the site–bond percolation problems on the same basis.

In Section 2 some definitions pertaining to cluster formation and percolation phenomena are given. The basic features of the ECMLT are described in Section 3. The algorithms pertaining to the ECMLT are outlined in Section 4, and a simple example illustrating the implementation and graph-theoretical representation of the algorithms is presented in Section 5. Numerical data are given in Section 6 for the site–bond problem for square and triangular lattices. These data are given for large simulated lattices containing up to 16 million sites, as it has been observed[31] (paper II) that the fluctuations of the Monte Carlo result are relatively large for two-dimensional lattices.[32]

2. THE SITE, THE BOND, AND THE SITE–BOND PROBLEMS

Since the site–bond problem has been vaguely addressed in the literature, it would be instructive to review the problem. For this purpose let us consider a planar square lattice with nearest neighbor interactions only. The site percolation problem for this lattice is that sites on the lattice are occupied with a probability c (or unoccupied with a probability $1 - c$). In the site case once two adjacent sites are occupied, we assume that there exists a bond connecting these sites; hence, they are members of the same cluster.

In the bond percolation problem we assume that all sites are occupied; however, the connecting bond is either open with a probability p or blocked with a probability $1 - p$. Thus, the formation of clusters in the bond case depends on the existence (or absence) of bonds connecting sites.

The site–bond problem is a logical extension of the pure bond and the pure site problems. In the site–bond case for a square lattice a site is occupied with a probability c and is connected to its occupied neighbors with a bond probability p. The square lattice is isotropic, so all bonds are equivalent. If we were concerned with a rectangular lattice, we would have two bond probability parameters $p\{\mathbf{a}_1\}$ and $p\{\mathbf{a}_2\}$ corresponding to the $\mathbf{a}_1$ and $\mathbf{a}_2$ primitive lattice vectors.

This reasoning could be extended to more complex lattices and to neighbors other than nearest neighbors. In the site–bond problem the percolation probability P is

$$P = P(c, p\{\mathbf{R}_1\}, p\{\mathbf{R}_2\}, \ldots, p\{\mathbf{R}_h\}) \tag{1}$$

where $p\{\mathbf{R}_h\}$ are the bond probabilities that two sites $\mathbf{S}_i$ and $\mathbf{S}_j$ separated by a vector $\mathbf{R}_h = \mathbf{S}_i - \mathbf{S}_j$ are connected. The pure site and the pure bond percolations can be given as limiting cases of Eq. (1). The site percolation probability is now given by

$$P = P(c, p\{\mathbf{R}_1\} = 1, p\{\mathbf{R}_2\} = \ ,..., p\{\mathbf{R}_h\} = 1) \tag{2}$$

and the bond percolation probability is given by

$$P = P(c = 1, p\{\mathbf{R}_1\}, p\{\mathbf{R}_2\},..., p\{\mathbf{R}_h\}) \tag{3}$$

Equation (1) defines a percolation zone in an $(h + 1)$-dimensional space. This zone is bounded by the critical percolation surface P_s, where

$$P_s(c, p\{\mathbf{R}_1\}, p\{\mathbf{R}_2\},..., p\{\mathbf{R}_h\}) = 0 \tag{4}$$

The critical percolation surface shrinks to a single point for the special cases of the site and the bond (with a single p parameter) problems.

3. THE EXTENDED CLUSTER MULTIPLE LABELING TECHNIQUE

The extended cluster multiple labeling technique shares many common features with its predecessor the cluster multiple labeling technique, which was described in I.[25] Thus, this discussion will mainly be focused on the special characteristics of the ECMLT. The goals of the ECMLT are to classify clusters according to their sizes and to determine the cluster size distributions for the site–bond problem. A simulated crystal is considered for which the site occupation probability is c and the bond connectivity probabilities are $p\{\mathbf{R}_1\}, p\{\mathbf{R}_2\},..., p\{\mathbf{R}_h\}$. A site $\mathbf{S}_i$ in the crystal and bonds originating from this site are assigned random numbers σ_i and β_{ij}, respectively, where $0 < \sigma_i < 1$ and $0 < \beta_{ij} < 1$. The number of β_{ij} assigned to a site $\mathbf{S}_i$ is equal to the number of connecting bonds extending from $\mathbf{S}_i$ to sites $\mathbf{S}_j$. These $\mathbf{S}_j$ sites are defined as *neighbors* of $\mathbf{S}_i$. It should be noted that $\beta_{ij} = \beta_{ji}$. Crystal sites are inspected sequentially to determine the cluster classification of the sites. If $\sigma_i > c$, site $\mathbf{S}_i$ is vacant, and $L(i)$ is set to 0, where L is the site occupation vector. When $\sigma_i < c$, site $\mathbf{S}_i$ is occupied, and $L(i)$ is assigned a cluster label m_t^α, where α is a symbolic name for the cluster containing $\mathbf{S}_i$. A cluster α may be assigned several cluster labels. These are given as a set of natural numbers:

$$\{m_1^\alpha, m_2^\alpha,..., m_s^\alpha,..., m_t^\alpha,...\} \tag{5}$$

In this set only one number is regarded as the *proper* cluster label, which we shall designate as m_s^α. This is the smallest number of set (5). The following set of integers provides the connections between the m_t^α labels:

$$\{N(m_1^\alpha), N(m_2^\alpha),..., N(m_s^\alpha),..., N(m_t^\alpha),...\} \tag{6}$$

In (6), $N(m_s^{\alpha})$ is the only positive integer member of the set, and denotes the number of occupied sites belonging to the α cluster. The remaining members of (6) are negative integers, providing links between the other m_t^{α} labels and the *proper* label m_s^{α}. The m_t^{α} labels are related to the m_s^{α} label by

$$m_s^{\alpha} = -N(-N(\cdots -N(-N(m_t^{\alpha}))\cdots)) \tag{7}$$

The assignment of a cluster label to an occupied site $\mathbf{S}_i$ depends on the cluster assignments of its previously labeled *neighbors* $\mathbf{S}_j$. The possibilities for the cluster label assignment for site $\mathbf{S}_i$ are as follows:

(a) If the following inequalities hold for all the previously labeled *neighbor* sites $\mathbf{S}_j$

$$\beta_{ij} > p\{\mathbf{S}_i - \mathbf{S}_j\} \qquad \text{or} \qquad L(j) = 0 \tag{8}$$

then a new cluster label m_s^{μ} is assigned to site $\mathbf{S}_i$ and also $N(m_s^{\mu}) = 1$ and $L(i) = m_s^{\mu}$ are set.

(b) If the following inequalities hold for some or all previously labeled *neighbor* sites $\mathbf{S}_j$

$$L(j) > 0 \qquad \text{and} \qquad \beta_{ij} < p\{\mathbf{S}_i - \mathbf{S}_j\} \tag{9}$$

then sites $\mathbf{S}_j$ that obey inequalities (9) and site $\mathbf{S}_i$ are members of the same cluster. Let us assume now that site $\mathbf{S}_i$ links q distinct fragments, where each fragment is denoted by a different *proper* cluster label m_s^{ν}. Also, a cluster fragment α has the smallest *proper* cluster label of the q cluster fragments. This proper label is m_s^{α}, so $L(i)$ is set $L(i) = m_s^{\alpha}$. Now $N(m_s^{\alpha})$ would denote the total number of sites in the linked cluster. The other $q - 1$ of the $N(m_s^{\nu})$ that correspond to the $q - 1$ cluster fragments are reset to $N(m_s^{\nu}) = -m_s^{\alpha}$. It should be noted that the readjustments of the N's are temporary and that only after the entire crystal is scanned and labeled can the cluster sizes be determined from the positive members of the N set.

The probability P_n that a site is occupied *and* belongs to a cluster of size n given for a set of parameters $c, p\{\mathbf{R}_1\}, p\{\mathbf{R}_2\},\ldots, p\{\mathbf{R}_h\}$ can be estimated from

$$P_n = i_n n/T \tag{10}$$

where i_n denotes the number of clusters of size n and T is the total number of sites in the simulated lattice. The percolation probability (1) can be determined from Eq. (10) for $n = n_{\max}$, where $n_{\max}$ denotes the size of the largest cluster. The evaluation of the percolation probability P is performed within the percolation zone. The critical percolation surface (4) can be determined for the simulated lattice by extending the I'_{av} criterion introduced in I[25] for the site problem to the general site–bond problem.

4. THE ALGORITHMS

The multiple labeling process described in Section 3 can conveniently be presented in terms of graph theory. Each cluster label set (5) can be represented by a *tree* graph[33] (see Fig. 5). The *labels* are represented by the tree *vertices*, and the cluster *proper* label m_s^{α} is denoted by the *root* of the tree. The length of the *path* from a given vertex to the root is equal to the number of *edges* connecting the *vertex* to the *root*.

The algorithms to be given in this section involve the following procedures: (a) generation of a random crystal; (b) UNION operations on *disjoint* sets [labeled sets (5)]; and (c) FIND operations, to determine the *proper* label (*root*) of a given labeled site.

UNION–FIND algorithms analogous to the ones presented here have been proposed for data set processing.[34] Specifically, UNION–FIND algorithms have been applied to perform "equivalencing" operations[35] on identifiers in computer languages such as Assembler and Fortran.

For the sake of clarity and compactness, we have chosen to present the algorithms developed for the ECMLT in Pigin Algol format.[34] The lattice generation and labeling process is illustrated in Fig. 1. A vector L, whose elements represent the lattice sites, provides the data base for the problem. Line 1 of Fig. 1 specifies that lattice sites are scanned sequentially from the first site to the last site. If a site is not occupied (line 2) L[SITE] is set to 0; otherwise, the labeling process of L[SITE] begins, and all previously *neighboring* labeled sites are searched (line 3). Forward *neighboring* sites are not searched, as they have not yet been created or labeled. Thus, for nearest neighbors in square and triangular lattices, two and three *neighbors* are scanned, respectively. It should be noted that a simple relationship exists between the index SITE and the indices NEIGHBOR of L because of the translational symmetry of regular lattices. Line 4 of Fig. 1 corresponds to the conditional expressions given by (9). If SITE and NEIGHBOR are connected, then routine CLASSIFY is envoked to determine the *proper* label NEIGHBOR__LABEL of L[NEIGHBOR] (line 5). Initially, the parameter LABEL is determined by the *proper* label of the first encountered labeled L[NEIGHBOR]. If there is more than one occupied *neighboring* site, a UNION operation is performed on the labeled sets, provided that the roots of the *neighboring* sites are different from LABEL (line 6). In line 7, LABEL becomes the combined root of LABEL and NEIGHBOR__LABEL trees if LABEL < NEIGHBOR__LABEL; otherwise, LABEL is set to NEIGHBOR__LABEL. If no *neighboring* sites are occupied (line 8), then a new label is generated by the variable COUNT. L[SITE] is labeled with LABEL (line 9), and N[LABEL] is incremented by 1 as the cluster size increases when L[SITE] is labeled.

```
        LATTICE GENERATION AND LABELING ROUTINE

        BEGIN
        COUNT ← 0;
1.      FOR SITE ← FIRST UNTIL LAST DO
2.      IF SRAND[SITE] > c  THEN L[SITE] ← 0
        ELSE
            BEGIN
            LABEL ← 0;
3.          FOR all previously labeled neighbors of SITE DO
4.          IF L[NEIGHBOR] > 0 AND BRAND[SITE,NEIGHBOR] ≤ p THEN
                BEGIN
5.              CLASSIFY(L[NEIGHBOR],NEIGHBOR_LABEL);
                IF LABEL = 0 THEN LABEL ← NEIGHBOR_LABEL;
6.              IF NEIGHBOR_LABEL ≠ LABEL THEN
                WITHOUT LOSS OF GENERALITY assume
                LABEL < NEIGHBOR_LABEL  OTHERWISE interchange
                the role of LABEL with NEIGHBOR_LABEL
                    BEGIN
7.                  N[LABEL] ← N[LABEL] + N[NEIGHBOR_LABEL];
                    N[NEIGHBOR_LABEL] ← -LABEL;
                    NEIGHBOR_LABEL ← LABEL;
                    END
                END
8.          IF LABEL = 0 THEN
                BEGIN
                COUNT ← COUNT + 1;
                LABEL ← COUNT;
                N[LABEL] ← 0;
                END
9.          L[SITE] ← LABEL;
            N[LABEL] ← N[LABEL] + 1;
            END
        END
```

Fig. 1. Lattice generation and labeling routine for the site–bond case. The vector SRAND and the matrix BRAND correspond to the site and bond random number sets, respectively [see (8) and (9)]. c and p are the site and bond occupation probabilities as defined in Section 2. L is the lattice vector, where the elements of L denote the lattice sites. The N vector is defined by (6).

The lattice generation and site labeling algorithm given in Fig 1 can be applied to the pure bond and the pure site problems. This is accomplished by setting $c = 1$ (line 2) and $p = 1$ for all *neighbors* (line 4) for the bond and site problems, respectively.

The procedure CLASSIFY,[25] which represents a FIND algorithm,[34] is displayed in Fig. 2. The parameter M (line 1) is a site label, whereas ROOT

```
DETERMINATION OF THE PROPER CLUSTER LABEL (ROOT)
                    ROUTINE

1.   PROCEDURE CLASSIFY(M,ROOT):
     BEGIN
     ROOT ← M;
2.   IF N[ROOT] < 0 THEN
          BEGIN
          ROOT ← -N[ROOT];
3.        IF N[ROOT] <0 THEN
               BEGIN
4.             REPEAT ROOT ← -N[ROOT] UNTIL N[ROOT] > 0;
5.             N[M] ← -ROOT;
               END
          END
     END
```

Fig. 2. Procedure CLASSIFY[25] determines the *proper* label ROOT of a given site label M. The N vector is defined by (6).

represents the *proper* label for that site. The procedure initially investigates the tree vertex corresponding to M (line 2). If the condition given in line 2 is true, then M is the ROOT; otherwise, the procedure moves to a higher vertex (line 3). The search for the root continues until the ROOT is found (line 4). Line 5 denotes a partial path compression for M, when the path length from M to ROOT is greater than one edge. Following the path compression, M is attached directly to ROOT through a single edge. The reason for performing path compression on the tree vertices is to speed up CLASSIFY for successive encounters with the label M.

The pertinent feature of the ECMLT and the algorithms associated with it is that only a single scan of the crystal is required for a given set of parameters $c, p\{\mathbf{R}_1\},\ldots, p\{\mathbf{R}_h\}$. As crystal scanning is performed sequentially, the random number sets SRAND[SITE] and BRAND[SITE,NEIGHBOR] need not be stored in computer memory; they are generated as a particular site is inspected and labeled. Furthermore, only a small fraction of the elements of the L vector has to be concurrently stored in computer memory.

The algorithms given in Figs. 1 and 2 provide the basic approach to the ECMLT. However, there is still room for improvements in terms of computer

space and time. This can be achieved, for example, by introducing the following modifications in the algorithms:

(a) Cluster labels can be recycled[25] because the labeling process involves only a single scan of the lattice. As the scanning progresses, many clusters would have been completely scanned and their labels could be reused. This recycling process of labels would lead to a reduction in the size of the vector N.

(b) Time could be saved for a sequence of simulations, where each simulation run[2] would correspond to an increased value of c (while the p parameters are held constant). Here, the entire vectors L and N would be saved for successive runs. In each successive simulation run, when c is incremented by $Dc > 0$ to $c + Dc$, only DcT sites would be labeled, as opposed to $(c + Dc)T$ sites in the original algorithm. In this approach all *neighbors* of a given site are searched in each run (except for the first run), because all *neighbors* might have been labeled in previous runs. The approach suggested here is especially useful for the determination of the critical percolation surface (4) when series of simulations for small increments in c are called for.

The limiting factor for the above modifications is that they cannot be applied simultaneously. In a multiscan approach labels cannot be recycled. Thus, by saving time, computer space is lost.

5. GRAPH-THEORETICAL REPRESENTATION OF THE LABELED SETS

As indicated in previous sections, a graph-theoretical approach can provide a convenient representation for the labeled sets (5). This approach is illustrated here by following a simple example. We shall consider a square lattice shown in Fig. 3. The lattice contains 29×29 sites and is occupied with probabilities $c = 0.57$ and $p = 1$ (site case[25]). We shall focus our attention now on a single cluster which is encountered for the first time during the lattice scan at the site denoted by a circle in Fig. 3. This site is given a *proper* label 47, as shown in Fig. 4. Whenever sites with *proper* labels other than 47 encounter a site belonging to the 47 cluster, a UNION operation is performed. The growth of the 47 cluster through UNION operations on directed trees is displayed in Fig. 5. In phase (c) of the growth, a path compression operation is performed on the label 63. This label is now attached directly to the root, as illustrated in phase (d).

[2] A simulation run is a run for which the cluster size distribution is determined for a single set of c and p parameters.

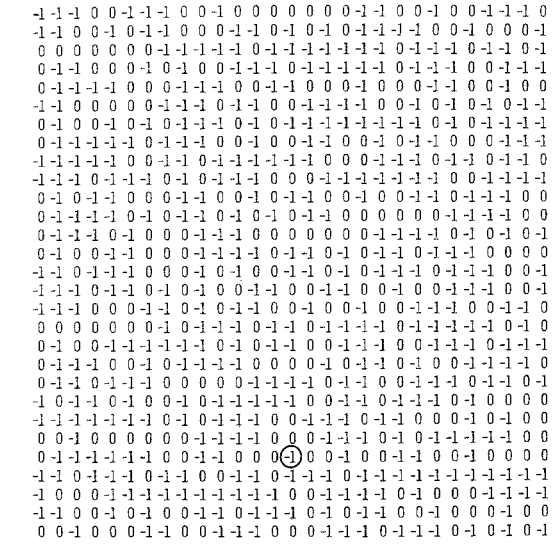

Fig. 3. A square lattice (site case) containing 29 × 29 sites, where $c = 0.57$. Here -1 denotes occupied sites and 0 denotes vacant sites. The circled -1 denotes the first member of the cluster with a *proper* label 47.

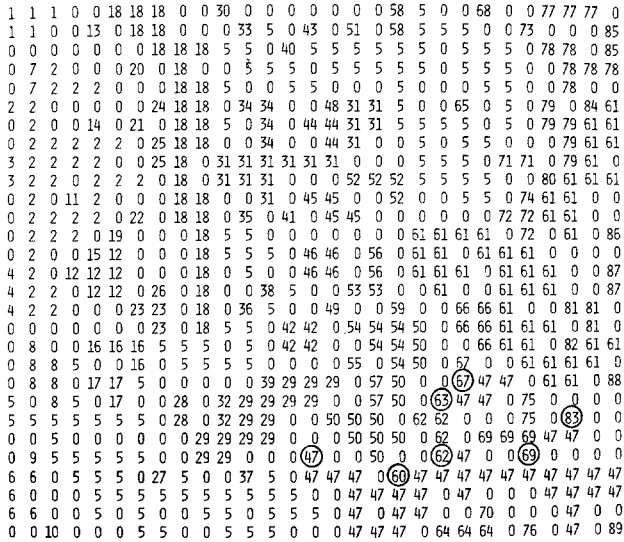

Fig. 4. The occupied sites of Fig. 3 are assigned labels. The sites where the UNION operations are performed are circled.

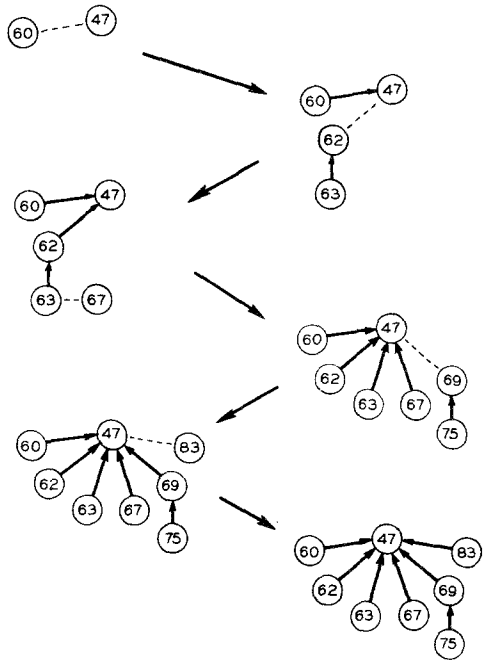

Fig. 5. Graph-theoretical representation for the labeled sets corresponding to the cluster denoted by the *proper* label 47 in Fig. 4. (Note that edges are entering the *roots* but none are pointing out.) The *union* of the trees is denoted by the dashed lines.

By inspecting the various phases of the growth of the 47 tree in Fig. 5, it can be observed that the rightmost vertices, as well as the root, participate actively in the UNION operations. The reason for the inactivity of the other labels can be attributed to the application of a single scanning sequential process. This feature of the labeling process permits the recycling of inactive labels even if the clusters to which they belong are not completely scanned.

6. NUMERICAL RESULTS

The algorithms given in this paper were applied to square and triangular lattices. Numerical data will be given for these structures.

In the first example a square lattice is studied for which only four nearest sites are considered to be *neighbors*. The critical percolation curve $P_s(c, p) = 0$ of the square lattice is determined for various values of c and p, where p denotes the bond probability of the four equivalent *neighbors*. Results for a lattice containing 100×100 sites are displayed in Figs. 6 and 7. In Fig. 6 the percolation probability P is given as function of c for some values of p. The percolation zone bounded by the critical percolation curve (4) is

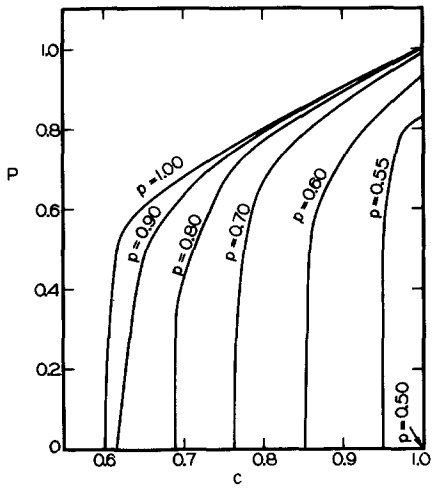

Fig. 6. Percolation probability P vs. c for some values of p for a square lattice containing 100 × 100 sites, with free boundaries.

shown in Fig. 7. It should be noted that for a lattice of a given size and structure the same site and bond random number sets are used for all runs in order to minimize fluctuations. The waviness of the critical percolation curve can be attributed to the small lattice studied in this example. We have found[31] 1.5% variations in the percolation threshold values for the site problem of 100 × 100 site lattices.

The change in the percolation threshold in the transition from a triangular lattice topology to a square lattice topology is given in Fig. 8. In this example two bonds of the triangular lattice are assigned a varying bond probability p, where p is varied from 1 to 0 corresponding to the transition from triangular to square lattice. As lattices containing 2000 × 2000 sites are considered, the fluctuations are much smaller than in the previous example. Here the labels are recycled to reduce the storage requirements. The triangular to square lattice transition seems to follow a straight line, as can be observed from Fig. 8.

The transition from a triangular lattice topology to a square lattice topology is further illustrated in Table I, where data on the cluster size distribution are given for some values of c and p in the vicinity of the critical percolation curve. In Table II the cluster size distribution is given for the site problem of large triangular and square lattices. The pertinent feature of these distributions is that a number of large and intermediate clusters exist in the vicinity of the critical percolation curve. The CPU (central processor unit) time for the 4,000,000-site and 16,000,000-site lattices is approximately 10 and 40 sec, respectively. We ran all the simulations on the University of Michigan Ahmdal 470 computer.

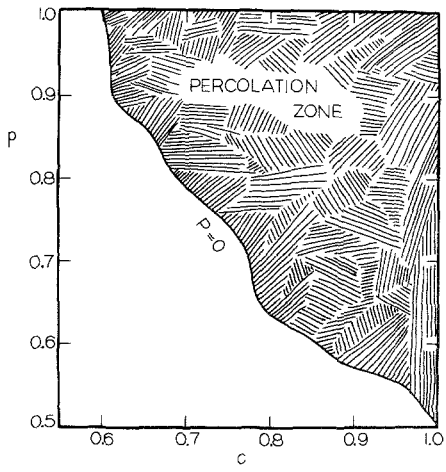

Fig. 7. Critical percolation curve, Eq. (4), for a square lattice containing 100 × 100 sites, with free boundaries.

7. DISCUSSION

In I[25] and also in Section 6 of this paper, the efficiency of the multiple labeling process has been demonstrated. It was shown that the CPU time required to perform a simulation run was essentially linear in the number of lattice sites.[25] Introducing bond probabilities does not alter the picture significantly, as the number of algorithm instructions associated with bonds is proportional to the lattice size. Since for each run only a single scan of the

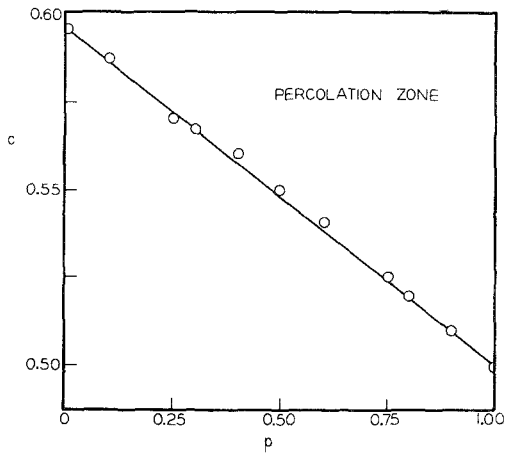

Fig. 8. Critical percolation curve $P_s(c, p) = 0$ for a *triangular* to *square* lattice transition. Here p denotes the bond probability for *two* bonds of the triangular crystal. Free boundaries were used.

Table I. Cluster Size Distribution for 2000 × 2000 Triangular Lattice (p Is Variable for Two Bonds[a])

	Number of clusters within a specified cluster size range[b]									
c/p	1 to 1000	1001 to 2000	2001 to 6000	6001 to 15,000	15,001 to 50,000	50,001 to 100,000	100,001 to 200,000	200,001 to 500,000	500,001 to 1,000,000	1,000,001 to 4,000,000
0.51000/0.90	75,150	38	40	9	4	1	1	3	0	0
	574,594	54,554	133,087	79,213	82,337	64,407	180,233	871,939		
0.51250/0.90	72,602	32	29	5	2	0	0	1	0	1
	516,228	47,587	97,449	45,843	47,209			242,225		1,053,846
0.51500/0.90	70,269	19	21	3	0	0	0	0	0	1
	472,174	28,035	71,206	29,817						1,458,960
0.52000/0.90	65,653	11	8	3	0	0	0	0	0	1
	390,268	15,530	27,735	26,162						1,620,520
0.52000/0.75	86,911	68	75	22	17	2	1	0	0	0
	715,805	96,844	244,773	217,677	499,650	188,801	116,665			
0.52500/0.75	81,527	43	41	12	7	2	2	0	1	0
	593,719	60,233	138,051	117,907	212,555	114,568	269,810		593,296	
0.53000/0.75	76,495	27	23	0	1	0	0	0	0	1
	492,667	37,926	84,269		32,064					1,473,287
0.55000/0.75	59,087	1	0	0	0	0	0	0	0	1
	253,095	1365								1,945,466

[a] Here p denotes the bond probability for two bonds of the triangular crystal.

[b] The first line for each parameter set c/p denotes the number of clusters within a specified cluster size range; the underlined numbers denote the total number of sites belonging to the clusters of the specified size range.

Table II. Cluster Size Distribution for 4000 × 4000 Triangular ($p = 1$) and Square ($p = 0$) Lattices[a]

	Number of clusters within a specified cluster size range									
c/p	1 to 1000	1001 to 2000	2001 to 6000	6001 to 15,000	15,001 to 50,000	50,001 to 100,000	100,001 to 200,000	200,001 to 500,000	500,001 to 1,000,000	1,000,001 to 16,000,000
0.49990/1.0	283,338	135	115	35	15	2	1	1	1	1
	2,188,107	186,449	388,038	322,937	428,368	118,364	122,806	202,380	637,294	3,404,119
0.50005/1.0	282,742	135	110	33	14	2	1	1	1	1
	2,175,051	185,445	368,272	295,531	402,939	122,871	122,894	214,575	638,243	3,475,391
0.50020/1.0	282,147	133	106	31	14	2	1	1	0	1
	2,162,836	183,541	358,348	281,153	410,638	123,111	123,053	215,484		4,145,497
0.50100/1.0	278,956	120	90	25	11	3	1	1	0	1
	2,088,840	163,068	301,983	216,063	317,304	185,026	178,269	237,837		4,328,256
0.59460/0.0	433,360	146	101	28	14	4	0	0	0	2
	2,492,744	205,304	337,011	277,123	379,381	293,010				5,531,237
0.59500/0.0	431,303	134	94	26	10	4	0	0	0	2
	2,461,371	188,139	309,132	258,453	248,086	314,694				5,742,366
0.59650/0.0	423,767	103	70	17	4	0	0	0	0	1
	2,334,378	141,594	215,668	155,333	111,337					6,588,219
0.59750/0.0	419,000	97	59	11	4	0	0	0	0	1
	2,246,226	131,732	186,539	103,887	113,941					6,780,241

[a] See footnotes to Table I.

lattice is performed, the contribution to the time complexity[29] of the bond instructions is linear in the size of the problem. The only nonlinearity associated with the algorithms is related to procedure CLASSIFY (see Section 5).

A useful feature of the ECMLT is the flexibility of the method. Although only regular lattices are considered in this paper, it is possible to extend the method quite readily to irregular structures encountered in continuous percolation processes.[26,27,38] A variety of data bases can be used in conjunction with the ECMLT; however, costly linked lists would not be required in most cases.

In our introductory remarks we emphasized the problems inherent in simulating large lattices. A problem of a different nature requiring further investigation is that of the random number generator associated with the Monte Carlo Simulation. We have used a congruential generator of the IBM RANDU type; however, it has been suggested[37] that pseudo-random-number sets based on congruential generators suffer from various correlations and fail to satisfy some statistical tests for randomness.[38] It is not known to what extent the faults in the generators affect the outcome of the percolation simulations. It would be useful to apply other generators for the Monte Carlo simulations in order to establish whether the results depend on the type of generator used. Feedback shift register generators[39] might be suitable candidates for such a test.

This paper has been devoted mainly to the description of percolation algorithms; however, it is instructive to review a situation where the site–bond algorithm could be applied for polymer gelation. Here we consider the copolymerization of two types of monomeric units A and B. In this model, c corresponds to the concentration of type A monomer, while the p parameters correspond to the fraction of functional groups reacted[40,41] in the condensation reaction. It would be necessary to specify a set of p parameters p(A, A), p(B, B), and p(A, B), corresponding to A–A, B–B, and A–B bonds, respectively. An interesting limiting case arises by setting p(A, A) = 0 and p(B, B) = 0. This limiting case may apply to immunization reactions between antibodies and antigens.[42,43] It should be noted that the algorithms given here should be modified for the copolymerization problem because copolymers contain clusters of both A and B species.[3]

REFERENCES

1. H. L. Frisch and J. M. Hammersley, *J. Soc. Ind. Appl. Math. B* **11**:894 (1963).
2. V. K. S. Shante and S. Kirkpatrick, *Adv. Phys.* **20**:325 (1971).

[3] We note that the above work was first presented by one of us (JH) at a Statistical Mechanics Meeting at Yeshiva University in Spring 1977. Since this manuscript was completed, a number of papers relevant to the site–bond problem have appeared.[44–48]

3. J. W. Essam, *Phase Transition and Critical Phenomena*, C. Domb and M. S. Green, eds. (Academic Press, New York, 1973), Vol. 2, p. 197.
4. (a) L. Torelli and A. E. Scheidegger, *Pure Appl. Geophys.* **89**:32 (1971); (b) P. G. de Gennes and E. Guyon, *J. Mecanique* **17**:1 (1978).
5. D. Stauffer, *J. Chem. Soc. Faraday Trans. II* **72**:1354 (1976).
6. R. J. Elliot, B. R. Heap, D. J. Morgan, and G. S. Rushbrooke, *Phys. Rev. Lett.* **5**:366 (1960).
7. G. E. Pike, W. J. Camp, C. H. Seager, and G. L. McVay, *Phys. Rev. B* **10**:4909 (1974).
8. R. Kikuchi and H. Sato, *J. Chem. Phys.* **53**:2702 (1970).
9. J. Hoshen and R. Kopelman, *J. Chem. Phys.* **65**:2817 (1976).
10. R. Kopelman, E. M. Monberg, F. W. Ochs, and P. N. Prasad, *Phys. Rev. Lett.* **34**:1506 (1975).
11. S. Kirkpatrick, *Phys. Rev. Lett.* **27**:1722 (1971).
12. C. H. Seager and G. E. Pike, *Phys. Rev. B* **10**:1435 (1974).
13. M. H. Cohen, I. Webman, and J. Jortner, *J. Chem. Phys.* **64**:2013 (1976).
14. M. Pollak and I. Reiss, *J. Phys. C* **9**:2339 (1976).
15. M. F. Sykes, D. S. Gaunt, and J. W. Essam, *J. Phys. A* **9**:L43 (1976).
16. J. M. Hammersley, *Methods Comput Phys.* **1**:281 (1963).
17. M. E. Fisher and J. W. Essam, *J. Math. Phys.* **2**:609 (1961).
18. M. F. Sykes, D. S. Gaunt, and M. Glen, *J. Phys. A* **9**:87, 97, 715, 725 (1976).
19. H. L. Frisch, S. B. Gordon, V. A. Vyssotsky, and J. M. Hammersley, *Bell. Syst. Tech. J.* **41**:909 (1962).
20. P. Dean, *Proc. Camb. Phil. Soc.* **5**:397 (1963).
21. A. S. Skal, B. I. Shklowski, and A. L. Efros, *Sov. Phys.—Solid State* **15**:961 (1973).
22. G. D. Quinn, G. H. Bishop, and R. Harrison, *J. Phys. A* **9**:L9 (1976).
23. S. Kirkpatrick, *Phys. Rev. Lett.* **36**:69 (1976).
24. P. L. Leath, *Phys. Rev. Lett.* **36**:921 (1976).
25. J. Hoshen and R. Kopelman, *Phys. Rev. B* **14**:3438 (1976).
26. G. E. Pike and C. H. Seager, *Phys. Rev. B* **10**:1421 (1974).
27. R. Zallen and H. Scher, *Phys. Rev. B* **4**:4471 (1971).
28. (a) J. M. Hammersley and D. C. Handscomb, in *Monte Carlo Methods* (Methuen, London, 1964), p. 135; (b) K. Binder, ed., *Monte Carlo Methods in Statistical Physics* (Springer, Heidelberg, 1979).
29. J. Hartmanis and J. E. Hopcroft, *J. ACM* **18**:444 (1971).
30. S. Kirkpatrik, *Rev. Mod. Phys.* **45**:574 (1973).
31. J. Hoshen, R. Kopelman, and E. M. Monberg, *J. Stat. Phys.* **19**:219 (1978).
32. H. L. Frisch, J. M. Hammersley, and D. J. A. Welsh, *Phys. Rev.* **126**:949 (1962).
33. F. Harary, *Graph Theory* (Addison-Wesley, Reading, Massachusetts, 1969), p. 32.
34. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, Massachusetts, 1974), p. 129.
35. B. A. Galler and M. J. Fischer, *Comm. ACM* **7**:301 (1964).
36. H. Scher and R. Zallen, *J. Chem. Phys.* **53**:3759 (1970).
37. J. R. B. Whittlesey, *Comm. ACM* **11**:641 (1968).
38. G. Masaglia, in *Application of Number Theory to Numerical Analysis*, S. K. Zaremba, ed. (Academic Press, New York, 1972).
39. R. C. Tausworthe, *Math. Comput.* **19**:201 (1965).
40. C. Tanford, *Physical Chemistry of Macromolecules* (Wiley, New York, 1961), p. 140.
41. P. J. Flory, *J. Am. Chem. Soc.* **63**:3083, 3091, 3096 (1941).
42. R. J. Goldberg, *J. Am. Chem. Soc.* **74**:5715 (1952).

43. (a) C. DeLisi, *J. Theor. Biol.* **45**:555 (1974); (b) C. DeLisi and A. Perelson, *J. Theor. Biol.* **62**:159 (1976).
44. H. Ottavi, J. Clerc, G. Giraud, J. Roussenq, E. Guyon, and C. D. Mitescu, *J. Phys. C* **11**:1311 (1978).
45. H. J. Wintle and P. H. Puhach, *J. Stat. Phys.* **18**:557 (1978).
46. A. Coniglio, H. E. Stanley, and W. Klein, *Phys. Rev. Lett.* **42**:518 (1979).
47. P. Agrawal, S. Redner, P. J. Reynolds, and H. E. Stanley, Preprint (1979).
48. H. Nakanishi and P. J. Reynolds, preprint (1979).