J. Phys. A: Math. Gen. 17 (1984) 3009–3017. Printed in Great Britain

Generation of percolation cluster perimeters by a random walk

Robert M Ziff†, Peter T Cummings‡ and G Stell§

† Department of Chemical Engineering, The University of Michigan, Ann Arbor, Michigan 48109, USA
‡ Department of Chemical Engineering, Thornton Hall, The University of Virginia, Charlottesville, Virginia 22901, USA
§ Department of Mechanical Engineering and Department of Chemistry, The State University of New York, Stony Brook, New York 11794, USA

Received 15 December 1982, in final form 11 May 1984

Abstract. A type of self-avoiding random walk which generates the perimeter of two-dimensional lattice-percolation clusters is given. The algorithm has been simulated on a computer, yielding the mean perimeter length as a function of occupation probability.

1. Introduction

Random walks have frequently been used to model the conformation of *linear* polymeric molecules for physical systems in which the effective pair interaction between segments, or the correlation of bond angles, vanishes (see, e.g., Hammersley and Morton 1954, de Gennes 1975, 1979, Merajver *et al* 1981, Marqusee and Deutch 1981, Tobochnik *et al* 1982, Redner and Reynolds 1981, Alexandrowicz 1980). Various types of self-avoiding random walks on various lattices have been considered, and such properties as the relation of the radius of gyration to the mass have been thus studied.

More complicated geometrically than linear polymers are the branched or cross-linked polymers, in which some segments can have more than two bonds. In this case, the polymerising system may go through a distinct gelation transition, past which an infinite network or gel is formed. To model clusters of branched polymers, both random growth models (e.g. Broadbent and Hammersley 1957, Shante and Kirkpatrick, 1971, Lubensky and Issacson 1979, Ord and Whittington 1982, Hoshen and Kopelman 1976, and Seitz and Klein 1981) and various kinds of diffusive growth mechanisms (e.g. Rosenstock and Marquardt 1980, Rikvold 1982, Witten and Sander 1981 and Herrmann *et al* 1982) have been used. In this paper, we consider the 'percolation' clusters formed by random site placement on a two-dimensional lattice, and show that the perimeters of such clusters can be generated by a type of self-avoiding random walk whose properties are described below. The generation algorithm allows the simple simulation of cluster perimeters on a computer. Note that random walks have been considered on percolation clusters in other contexts: Stanley *et al* (1976) have argued that the *backbone* of a percolation cluster should act like a random walk, and of course there is the random walk of a particle diffusing over a percolation cluster discussed by de Gennes (1976).

In the site percolation model, points ('sites') on a lattice are randomly made 'occupied' with a probability, p, and left vacant with a probability $(1-p)$. Bonds are drawn between adjacent occupied, forming clusters. When p is above the percolation point, p_c, an infinite cluster or gel forms. An example of a two-dimensional cluster with $n=11$ sites is given in figure 1(a), where occupied sites are represented by a full circle, and vacant sites by a open circle.

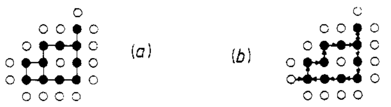

Figure 1. (a) An external percolation cluster of $n=11$ occupied sites. (b) A path around the perimeter of the cluster.

The percolation model should be especially useful to describe the behaviour of polymers near the gel point or critical point, where the clusters are large, because geometric and steric relations are taken into account, in contrast to mean field models, such as the Flory (1953)–Stockmayer (1943) model of polymerisation. The two-dimensional percolation on a square lattice that is considered here is a model for the reaction of branched polymers in which the basic monomeric unit has four functional groups. The percolation model can also be used to describe general agglomeration processes, such as the coagulation of colloidal suspensions, and nucleation.

Bounding each cluster is an external boundary surface and possibly one or more internal surfaces, if holes exist within the cluster. In the gel (for $p > p_c$), only internal surfaces can exist. For each value of p, $0 < p < 1$, a large lattice populated with occupied and unoccupied sites will contain a collection of both internal and external surfaces, with a surface or perimeter associated with each of these surfaces. The perimeter of a given surface can be characterised by either the number of occupied sites of the clusters which border the surface, or the vacant sites surrounding the surface (or within it, for an internal surface). Conventionally, these vacant sites are called 'perimeter sites'. Here we will be concerned with both the occupied sites and the vacant sites at the surface. In an external surface, the vacant sites are on the outside, while for an internal surface they are on the inside. In the example of figure 1(a), there is an internal surface with $n_o = 8$ occupied sites and $n_v = 1$ vacant sites, and an external surface with $n_o = 11$ and $n_v = 14$.

It has been established (Leath 1976a, b, Reich and Leath 1978, Domb *et al* 1975) that the total number of perimeter sites (the vacant sites on both external and internal surfaces) is proportional to the total number of occupied sites for large clusters, so in this sense clusters are ramified; however, little is known for the external surface alone. The nature of cluster surfaces enters in theories of polymer kinetics (Ziff *et al* 1982a,b, 1983, Leyvraz and Tschudi 1981, 1982) and of nucleation (Fisher 1967). The algorithm which we discuss here generates the surfaces of two-dimensional clusters, and allows by simulation the determination of the *distribution of perimeter length* and *mean perimeter length* of all the surfaces of the clusters on a lattice. This algorithm is derived by considering first an algorithm that measures the perimeter on an *existing* cluster, and observing that the cluster does not have to exist beforehand but can be generated on a blank lattice during the counting process. The new algorithm produces a sort of random walk which generates the perimeter independent of the cluster itself—the interior is not generated. In a computer simulation of the perimeter algorithm, in which perimeters with as many as 500 000 sites were generated, it was found that the mean perimeter length diverges *slower* than the mean cluster size, as the percolation point is approached.

2. The perimeter random walk

First consider the problem of measuring the perimeter of a single surface (either internal or external) of a given cluster on a lattice. Following is an algorithm (for a square lattice) that makes a path around the surface, visiting all occupied and vacant sites at the boundary at least once.

(1) Pick an adjacent occupied site–vacant site pair at one point on the surface. Define a direction by drawing an arrow from the vacant site to the occupied site and move to the occupied site.

(2) Facing in the direction of the arrow, look at the adjacent site to the left. If the site is . . .

(b) *vacant*, then go back to 2, now looking in the *next* direction, in the order left, front, right, back.

(c) *occupied*, then draw an arrow from the old occupied site to the new occupied site, and 'move' to the new site. Go back to 2, now looking in the new left direction.

(3) Continue this procedure until the path passes the starting point in the same direction as it had first been passed.

(The step 2a will be provided in a modification below.) This algorithm brings one on a 'walk' around a boundary surface, allowing n_v and n_o to be determined by counting the vacant sites as they are first encountered in step b, and counting the occupied sites as they are first encountered in step c. Since either site can be encountered more than once, it is necessary to remember whether a site has been previously counted, and so a site can have four possible states: occupied and counted, occupied and uncounted, vacant and counted, and vacant and uncounted. For example, the walk around the external surface of the cluster in figure 1(*a*) is shown in figure 1(*b*). Repeating this algorithm many times, one could find the distribution of n_o and n_v for all the surfaces in the system. This distribution would be characterised by a function $P(n_o, n_v; p)$ which gives the probability that a randomly chosen vacant site–occupied site pair belongs to a surface containing n_o occupied sites and n_v vacant sites. The distribution can be found equally well by measuring all the surfaces of a single (large) lattice populated with clusters, or by measuring one surface of a single cluster, then clearing off the lattice, and repopulating it with clusters (with the same p) and repeating.

Now observe that one could also start with a *blank* lattice and make the blank sites either occupied or vacant *as they are encountered* during the perimeter measuring walk. That is, now allow three possible states for a site: blank, occupied, or vacant. Start with a lattice full of blank sites, except for a vacant-site occupied-site pair at the centre, and proceed with the walk algorithm given above, adding the following step after step 2:

(a) *blank*, then make the site occupied (with a probability p) or vacant (with probability $1-p$), and go to b or c, respectively.

This procedure will *generate* cluster perimeters on a blank lattice as a kind of random walk. Because the state of a site (whether occupied or vacant) is determined by a independent process, and the state does not change once it has been chosen, it can be seen that the sites will be populated with the same probability as if they had been chosen beforehand. Therefore, the probability that a given perimeter is generated by this algorithm is the same as the probability that it would be found on an already populated lattice. The advantages of generating, rather than measuring, are that only one cluster is generated at a time, only those sites along the surface are generated, and there are no boundary effects caused by clusters touching the edge of the lattice (until, of course, the available computer space for growing the cluster runs out). In counting n_o and n_v, one need no longer keep track of counted versus uncounted sites as the sites can be counted as they are produced (in step a), and the three states (blank, occupied, vacant) are sufficient for both generating and counting boundary surfaces. The distribution $P(n_o, n_v; p)$ then follows by repeating the generating algorithm many times, each time on a new blank lattice.

A step-by-step example of the generation of a percolation cluster by the above algorithm is shown in figure 2. Occupied sites are indicated by a full circle, vacant sites by a open circle, and blank sites are not shown. In step 1, the initial surface pair is shown, with an arrow drawn from the vacant site to the occupied site. According to the algorithm, the first site to be 'looked at' is the blank site to the left; this site is labelled with a question mark. The state of this site is determined by a random process which generates a '1' with the given probability p and a '0' with probability $(1-p)$. Say a '1' is found, as indicated in the figure at step 1, then the site is made occupied, a new arrow is drawn, and a new blank site is looked at, as shown in step 2. That blank site is made vacant, so in step 3, the next blank site is 'looked at'. And so on, the randomly chosen string 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0 causes the simple perimeter of step 16 to be generated. Note in step 9 a site that has previously been made vacant is encountered; in 12, the path is retraced; in 14, an occupied site is encountered, and in 16, the original vacant site is encountered. Since the next step would duplicate step 2, the walk is terminated. An external perimeter with $n_o = 5$ and $n_v = 9$ is thus produced.

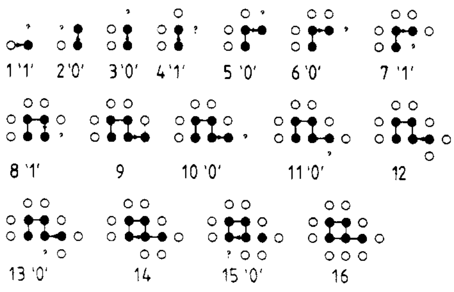

Figure 2. Generation of a perimeter by the random walk algorithm.

The algorithm produces a kind of two-sided self-avoiding random walk, one side being composed of vacant sites and the other of occupied sites. The walk can never cross the vacant side, while the occupied site is joined whenever it is encountered. When $p < p_c$, the walk has a strong tendency to turn right and close to make an external surface, as in figure 3. For all p, $0 < p < 1$, there is a finite probability that either kind of surface can be generated. Near $p = p_c$ large perimeters are generated as the walk turns left and right, trying to decide whether to be an external or an internal surface. In figure 3 is an example of a large internal cluster of $n_o = 1837$ and $n_v = 1201$ sites,

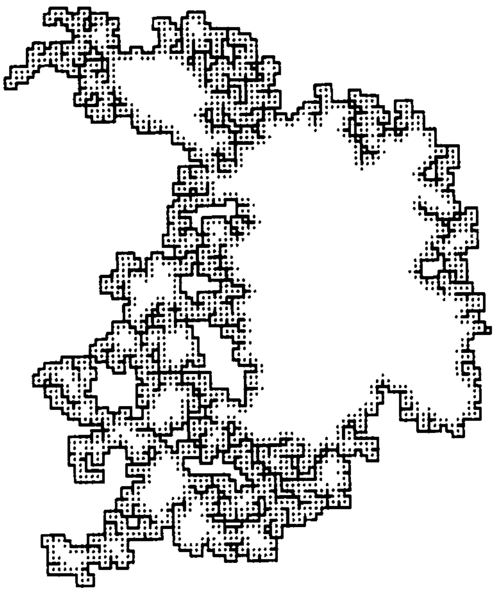

Figure 3. A computer-generated internal perimeter containing 1837 occupied sites and 1201 vacant sites.

generated at $p = 0.593 \approx p_c$ in a continuous path by the algorithm. Here the occupied sites show up as connected lines, and the vacant sites show up as isolated dots. The particularly large area of internal blank sites in this cluster is somewhat unusual.

An internal surface within a cluster or the gel can be thought of as an external surface of a 'cluster' of vacant sites, and an external surface can be thought of as an internal surface of a hole within a vacant-site cluster, or the vacant-site 'gel', if the vacant site system is considered to have a next-nearest-neighbour (NNN) interaction (vacant-bonds can be drawn between a vacant site and any of its eight closest neighbours). An algorithm to generate perimeters in the vacant-site system can be found by modifying our perimeter algorithm by interchanging the role of vacant sites and occupied sites, and allowing NNN interaction for vacant sites. This relationship reflects the duality between the nearest neighbour and NNN square lattices (Sykes and Essam 1963, 1964).

3. Computer simulation

A computer simulation of the perimeter generating algorithm was carried out, first on a desktop computer whose graphics capabilities proved very useful to visualise the surfaces, and then on a larger computer (a Univac 1100) with sufficient speed and memory size for quantitive results. Two bits of memory were used to represent the three possible states of each site, and a type of data banking scheme which assigns blocks of memory to sections of the lattice as they are entered by the random walk was used, allowing a virtual lattice of 4096×4096 sites and cluster surfaces with n_0 as large as 500 000 to be generated, using a memory space of 256 K, 32-bit words. In contrast, if one memory word were assigned to each lattice point, then a lattice of only

512×512 sites could be considered, and perimeters with ≈ 5000 would already run into the boundary. During these simulations, the value of n_o was monitored and averaged over many runs to yield $\langle n_o \rangle$, which is related to P by

$$\langle n_o \rangle = \sum_{n_o=1}^{\infty} \sum_{n_v=1}^{\infty} n_o P(n_o, n_v; p). \qquad (1)$$

The values for internal and external perimeters were averaged separately, yielding $\langle n_o \rangle^{\text{ext}}$ and $\langle n_o \rangle^{\text{int}}$. The results of the simulation of $\simeq 180\,000$ clusters are given in figure 4, where $\ln \langle n_o \rangle^{\text{ext}}$ and $\ln \langle n_o \rangle^{\text{int}}$ are plotted as a function of $\ln |p - p_c|$, using $p_c = 0.593$.

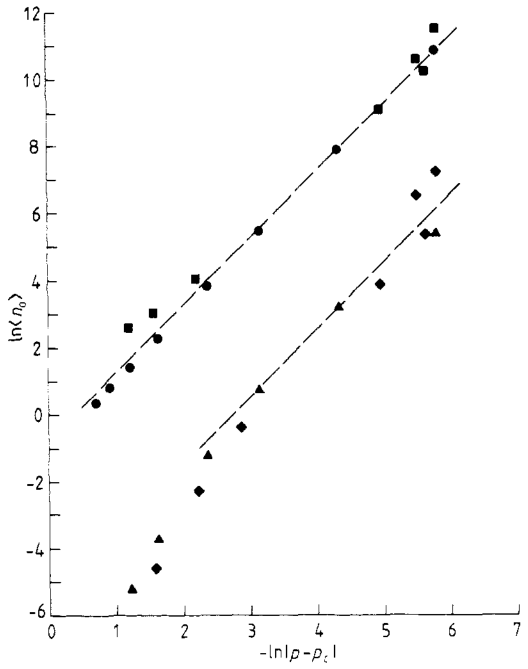

Figure 4. Results of 180 000 computer simulations of perimeters, plotted with $p_c = 0.593$. The broken lines have a slope of 2 and imply the behaviour of equation (2). ● = $\langle n_o \rangle^{\text{ext}} p < p_c$; ■ = $\langle n_o \rangle^{\text{int}} p > p_c$; ▲ = $\langle n_o \rangle^{\text{ext}} p < p_c$; ◆ = $\langle n_o \rangle^{\text{int}} p > p_c$.

The number of clusters generated for each data point varied from 50 000 at $p = 0.1$ and 0.9 to only 64 clusters at $p = 0.590$ and 0.5965. These last two were the closest values to p_c before the maximum size ($n_o \simeq 500\,000$) was reached. Since, at these values of p, $\langle n_o \rangle$ was about 100 000, only a relatively small number of closed perimeters could be generated in a reasonable amount of time.

Clearly, as $p \to p_c$, the average perimeter size diverges. Past p_c the average size again decreases as mainly the holes in the gel contribute to the total perimeter. This peaked behaviour is in contrast to the average cluster size, which diverges as $p \to p_c$ but then remains infinite for all $p > p_c$.

For $p < p_c$, $\langle n_o \rangle^{\text{int}} < \langle n_o \rangle^{\text{ext}}$—there is always more external surface than internal surface. Evidently, even though a single large cluster can have many internal surfaces, the overall external perimeter dominates. The opposite is true for $p > p_c$, as can be seen in figure 4.

The plots in figure 4 suggest that $\langle n_o \rangle^{\text{int}}$ and $\langle n_o \rangle^{\text{ext}}$ on either size of p_c have the following symmetric critical behaviour near p_c:

$$\begin{aligned} \langle n_o \rangle_-^{\text{ext}} = \langle n_o \rangle_+^{\text{int}} = A|p-p_c|^{-\alpha} \\ \langle n_o \rangle_-^{\text{int}} = \langle n_o \rangle_+^{\text{ext}} = B|p-p_c|^{-\alpha} \end{aligned} \qquad (2)$$

where the + refers to $p > p_c$ and − to $p < p_c$, with $\alpha \simeq 2.0 \pm 0.1$, $A \simeq 0.5$, $B \simeq 0.004$. The internal surface and external surface diverge with the same exponent and symmetric amplitudes. Alternately, *assuming* this symmetry, one would conclude from the data $p_c = 0.593 \pm 0.001$, consistent with other findings (Djordjevic *et al* 1982). The plot is very sensitive to small changes in p_c.

Note that the average cluster size diverges as $p \to p_c$ like

$$n \simeq c|p-p_c|^{-\gamma} \qquad (3)$$

with $\gamma \simeq 2.4$ (see Stauffer *et al* 1982). Thus, the average perimeter diverges slower than the average cluster size. Note, however, that these two quantitities are averaged in two different senses: (2) gives the mean length of a perimeter connected to an arbitrarily chosen perimeter site, while (3) gives the mean size of a cluster connected to an arbitrarily chosen occupied site. Thus, the relation of boundary perimeter to area for a given cluster does not follow directly. (Note that the relation of Leath (1976) for perimeter to area concerns the total perimeter of all the boundaries of a cluster.)

4. Further remarks

The proof that the random walks always close on themselves to form closed surfaces comes from the idea of the underlying lattice: For every random walk generated, there is a lattice populated with occupied and vacant sites and which contains the perimeter. It follows that all random walks close because all surfaces on an infinite lattice are closed.

Although the perimeter is generated, the 'inside' is not. In larger clusters (such as in figure 3), there is generally some open space of blank sites, and one does not know if they are occupied or vacant—thus, the size of the cluster (n) is not known. For the perimeter distribution function this size is not needed.

The numbers n_o and n_v are closely related for large surfaces. For a surface of n_o occupied sites and n_v vacant sites, a total of $n_o + n_v$ blank sites are therefore visited in the random walk which generates that surface. Since the blank sites are made occupied with probability p and left vacant with probability $1-p$, it follows that $n_o \simeq p\,(n_o + n_v)$ and $n_v \simeq (1-p)(n_o + n_v)$, or

$$n_v \simeq n_o\, p/(1-p) \qquad (4)$$

for large clusters. This relation was observed in the computer simulations. Thus, the behavior of $\langle n_v \rangle$ is the same as that of $\langle n_o \rangle$ described in figure 4 and equation (2), with the amplitudes rescaled according to (4).

The idea of generating individual perimeters on blank lattices is similar to the idea of generating individual *clusters* on a blank lattice by the basic percolation mechanism, where a cluster is generated by a 'flowing fluid' which emanates from the origin and, when encountering a blank site, either whets it (with probability p), creating an occupied site, or permanently blocks it (with probability $1-p$), creating a vacant site (Broadbent and Hammersley 1957). This algorithm can be used in a computer simulation to generate clusters and find, for example, the average size. Recently, Havlin *et al* (1983) use a similar idea to study diffusion on a percolation cluster, generating the cluster only as the diffusing particle contacts an unvisited site. One can use the cluster-growing algorithm to derive the relation between the total number of perimeter sites (b) and occupied sites (n) of a large cluster given by Leath (1976):

$$b \simeq np/(1-p). \tag{5}$$

For in generating a cluster of n sites and b perimeter sites, a total of $n+b$ blank sites are encountered, of which $np(n+b)$ are made occupied and $b(1-p)(n+b)$ are made vacant. The similarity of this argument to the argument used in deriving (4) explains the similarity of (4) and (5). Thus the ratio of total occupied sites to vacant sites of a large cluster is the same as the ratio of occupied sites to vacant sites of an individual large surface (for a given p). Once again, because an individual site can belong to many surfaces, these results do not say anything about the ratio of external perimeter to total size of a cluster.

The perimeter generating algorithm given here can be generalised easily to any two-dimensional lattice. It may be useful in finding p_c accurately for those lattices in which the exact value is not known. An interesting question for further study would be to find the exponent which describes how the radius of gyration grows with the size, for this random walk.

Acknowledgments

The authors acknowledge the National Science Foundation for support of this research carried out at the State University of New York at Stony Brook, and the Shell Career Initiation Grant for the support of RMZ at the University of Michigan as a Shell Faculty Fellow.

References

Alexandrowicz Z 1980 *Phys. Lett.* **80A** 284
Broadbent S R and Hammersley J M 1957 *Proc. Camb. Phil. Soc.* **53** 629
de Gennes P G 1975 *J. Physique Lett.* **36** L55
—— 1976 *Recherche* 7 919
—— 1979 *Scaling Concepts in Polymer Physics* (Ithaca, New York: Cornell University Press)
Djordjevic Z V, Stanley H E and Margolina A 1982 *J. Phys. A: Math. Gen.* **15** L405
Domb C, Schneider T and Stoll E 1975 *J. Phys. A: Math. Gen.* **8** L90
Fisher M E 1967 *Physics* **3** 225
Flory P J 1953 *Principles of Polymer Chemistry* (Ithaca, New York: Cornell University Press) ch 9
Hammersley J M and Morton K W 1954 *J. R. Stat. Soc.* B**16** 23
Havlin S, Ben-Avraham D, and Sompolinsky H 1983 *Phys. Rev.* A **27** 1730
Herrmann H J, Stauffer D and Landau D P 1982 *Phys. Rev. Lett.* **49** 412
Hoshen J and Kopelman R 1976 *Phys. Rev.* B **14** 3438
Leath P L 1976a *Phys. Rev. Lett.* **36** 921
—— 1976b *Phys. Rev.* B **14** 5046
Leyvraz F and Tschudi H R 1981 *J. Phys. A: Math. Gen.* **14** 3389
—— 1982 *J. Phys. A: Math. Gen.* **15** 1951
Lubensky T C and Issacson J 1979 *Phys. Rev.* A **20** 2130
Marqusee J A and Deutch J M 1981 *J. Chem. Phys.* **75** 5179

Merajver S D, Yorke E D and DeRocco A G 1981 *Phys. Rev.* A **23** 897
Redner S and Reynolds P J 1981 *J. Phys. A: Math. Gen.* **14** 2679
Reich G R and Leath P L 1978 *J. Phys. C: Solid State Phys.* **11** 1155, 4017
Ord G and Whittington S G 1982 *J. Phys. A: Math. Gen.* **15** L29
Rikvold P A 1982 *Phys. Rev.* A **26** 647
Rosenstock H R and Marquardt C L 1980 *Phys. Rev.* B **22** 5797
Seitz W A and Klein D J 1981 *J. Chem. Phys.* **75** 5190
Shante V K S and Kirkpatrick S 1971 *Adv. Phys.* **20** 325
Stanley H E, Birgeneau R J, Reynolds P J and Nicoll J F 1976 *J. Phys. C: Solid State Phys.* **9** L533
Stauffer D, Coniglio A and Adam M 1982 *Adv. Polym. Sci.* **44**
Stockmayer W H 1943 *J. Chem. Phys.* **11** 45
Sykes M F and Essam J W 1963 *Phys. Rev. Lett.* **10** 3
—— 1964 *J. Math. Phys.* **5** 1117
Tobochnik J, Webman I, Lebowitz J L and Kalos M H 1982 *Macromolecules* **15** 549
Witten T A and Sander L M 1981 *Phys. Rev. Lett.* **47** 1400
Ziff R M, Hendriks E M and Ernst M H 1982a *Phys. Rev. Lett.* **49** 593
—— 1982b *J. Stat. Phys.* **31** 519
—— 1983 *J. Phys. A: Math. Gen.* **16** 2293