Improved Rosenbluth Monte Carlo scheme for cluster counting and lattice animal enumeration

C. M. Care[1] and R. Ettelaie[2]
[1]*Materials Research Institute, Sheffield Hallam University, Pond Street, Sheffield, S1 1WB, United Kingdom*
[2]*Colloids and Rheology Unit, ICI Wilton, P.O. Box 90, Wilton, Middlesbrough, Cleveland, TS90 8JE, United Kingdom*
(Received 17 November 1999)

We describe an algorithm for the Rosenbluth Monte Carlo enumeration of clusters and lattice animals. The method may also be used to calculate associated properties such as moments or perimeter multiplicities of the clusters. The scheme is an extension of the Rosenbluth method for growing polymer chains and is a simplification of a scheme reported earlier by one of the authors. The algorithm may be used to obtain a Monte Carlo estimate of the number of distinct lattice animals on any lattice topology. The method is validated against exact and Monte Carlo enumerations for clusters up to size 50, on a two dimensional square lattice and three dimensional simple cubic lattice. The method may be readily adapted to yield Boltzmann weighted averages over clusters.

PACS number(s): 02.70.Lq, 05.50.+q

I. INTRODUCTION

The enumeration of lattice animals is an important problem in a variety of physical problems including nucleation [1], percolation [2], and branched polymers [3]. A lattice animal is a cluster of N connected sites on a lattice with given symmetry and dimensionality, and we seek to enumerate all distinct animals with a given number of sites. Exact enumeration has been carried out for small lattice animals using a variety of methods [2,4,5] but the methods become computationally prohibitive for large animals. Many techniques have been used to enumerate larger lattice animals including various Monte Carlo growth schemes [2,6–8], a constant fugacity Monte Carlo method [9,10], an incomplete enumeration method [11], and reaction limited cluster-cluster aggregation [3].

In this paper we describe an improvement of a method proposed by one of the authors [12], which was based on an extension of the scheme proposed by Rosenbluth and Rosenbluth [13] for enumerating self-avoiding polymer chains. The central problem in using the Rosenbluth scheme for lattice animal enumeration is calculating the degeneracy of the clusters that are generated. In the method proposed by Care, the cluster growth was modified in a way that forced the degeneracy to be $N!$ where N is the number of sites occupied by the lattice animal. However, the resulting algorithm was fairly complicated to implement. An alternative method of correcting for the degeneracy had been proposed by Pratt [14]. In this latter scheme the correcting weight is more complicated to determine and must be recalculated at each stage of the cluster growth if results are sought at each cluster size. However, the Pratt scheme does not require any restriction on the growth of the cluster.

We show here that there is a class of Rosenbluth-like algorithms that yield a degeneracy of N and are straightforward to implement. The method provides an estimate of the number of lattice animals and can also yield estimates of any other desired properties of the animals, such as their radius of gyration or perimeter multiplicities [2]. We describe and justify the algorithm in Sec. II and present results to illustrate the use of the method in Sec. III. Conclusions are given in Sec. IV.

An alternative, and commonly used, method of estimating the number of lattice animals of a given size is the *incomplete enumeration* method (e.g., [11]), which is based on an exact enumeration scheme for which an algorithm was proposed by Martin [15] and Redner [16]. Although effective, the incomplete enumeration scheme has the problem that configurations produced in a single trial are correlated and therefore averages must be taken over many trials to mitigate this effect. In the algorithm presented below, all the clusters of a given size are generated completely independently, with no reference to the clusters that have been generated previously. This property of the algorithm may have advantages for calculating some classes of cluster properties.

II. ALGORITHM

Any algorithm suitable for the purpose of enumeration of lattice animals using the Rosenbluth Monte Carlo approach must satisfy two important criteria. First of all it has to be ergodic. That is to say, the algorithm should have a nonzero probability of sampling any given cluster shape. The second criterion relates to the degeneracy that is associated with each cluster and requires this to be determinable. This degeneracy arises from the number of different ways that the same cluster shape can be constructed by the algorithm. While it is easy to devise methods of growing clusters that meet the first requirement, the second condition is more difficult to satisfy. For many simple algorithms the calculation of the degeneracy, for every cluster, can be a more complex problem than the original task of enumerating the number of lattice animals.

In the original Rosenbluth Monte Carlo approach of Care [12], this difficulty was overcome by ensuring that the degeneracy for all clusters of size N was the same and equal to $N!$. However, to achieve this result the algorithm had to employ a somewhat elaborate procedure. This made the implementation of the method rather complicated, as well as limiting its possible extension to enumeration of other types of clusters. Here we shall consider an alternative algorithm,

which, while satisfying both of the above criteria, is considerably simpler than the algorithm proposed by Care. In Sec. II A we describe the algorithm in its most basic form, before proving in Sec. II B that the ergodicity and degeneracy requirements are both met. In Sec. II C we demonstrate how the basic algorithm can be further refined to improve its efficiency.

A. Basic algorithm

Having chosen a suitable lattice on which the clusters are to be grown (square and simple cubic lattices were used in this study for two (2D) and three dimensional (3D) systems, respectively), a probability p of acceptance and $q=(1-p)$ of rejecting sites is specified. Although in principle any value of p between 0 and 1 can be selected, the efficiency of the sampling process is largely dependent on a careful choice of this value, as will be discussed later. In addition, an ordered list of all neighbors of a site on the lattice is made. For example, for a 2D square lattice this might read (right, down, left, up). While the order initially chosen is arbitrary, it is essential that this remains the same throughout a given run. In the basic algorithm, once chosen, the probability p remains fixed during the Monte Carlo sampling procedure. However, in Sec. II C the effect of relaxing this requirement is discussed.

We construct an ensemble of N_E clusters and for each of these calculate a weight factor which we subsequently use to calculate weighted averages of various cluster properties. For a property O of the clusters, the weighted average is defined as

$$\langle O\rangle_W=\frac{1}{N_E}\sum_{\alpha=1}^{N_E} W_\alpha O_\alpha . \tag{1}$$

The weight associated with cluster α with N sites is defined to be $W_\alpha=1/(d_N P_\alpha)$, where P_α is the normalized probability of growing the cluster and d_N is a degeneracy equal to the number of ways of growing a particular cluster shape. It can be shown [12] that the weighted average can be used to estimate the number c_N of lattice animals of size N and other properties such as the average radius of gyration $\overline{R_N^2}$:

$$E[\langle 1\rangle_W]=c_N , \tag{2}$$

$$E[\langle R_\nu^2\rangle_W]=\sum_{\{\nu=1\}}^{c_N} R_{N\nu}^2=c_N\overline{R_N^2}. \tag{3}$$

During the growth of each cluster we maintain a record of the sites that have been occupied, the sites that have been rejected, and a ''last-in-first-out stack'' of sites that is maintained according to the rules described below. Each cluster is grown as follows.

(i) Starting from an initial position, the neighbors of this site are examined one at a time according to the list specified above. An adjacent site is accepted with a probability p or else is rejected.

(ii) If the adjacent site is rejected, a note of this is made and the next neighbor in the list is considered.

(iii) If, on the other hand, it is accepted, then this becomes the current site and its position is added to the top of a stack, as well as to a list of accepted sites. The examination of the sites is now resumed for the neighbors of this newly accepted site. Once again this is done in the strict order that was agreed at the start of the algorithm.

(iv) Sites that have already been accepted or rejected are no longer available for examination. Thus, if such a site is encountered, it is ignored and the examination is moved on to the next eligible neighbor in the list.

(v) If at any stage the current site has no more neighbors left, that is, all its adjacent sites are already accepted or rejected, then the current position is moved back by one to the previous location. This will be the position below the current one in the stack. The current position is removed from the top of the stack, though not from the list of accepted sites.

(vi) The algorithm stops for one of the following two reasons. If ever the number of accepted sites reaches N, then the algorithm is immediately terminated. In this case a cluster of size N is successfully produced. Note that, unlike some of the other common cluster growth algorithms [8], it is not necessary here for every neighbor of the generated cluster to be rejected. Some of these might still be unexamined before the algorithm terminates. The second way in which the algorithm stops is when it fails to produce a cluster of size N. In this case, the number of accepted sites will be $M<N$, with all the neighbors of these M sites already having been rejected, leaving no eligible sites left for further examination. From step (v), it is clear that in cases such as this, the current position would have returned to the starting location.

(vii) The probability of producing a cluster of size N, in a manner involving r rejections, is simply $p^{(N-1)}q^r$. Hence the weight W_α associated with the growth of the cluster is given by

$$W_\alpha=1/[d_N p^{(N-1)}(1-p)^r], \tag{4}$$

where the degeneracy d_N is shown below to be exactly N. Failed attempts have a zero weight associated with them. However, they must be included in the weighted average of Eq. (1).

(viii) During the growth of a cluster of size N, we may also collect data for all the clusters of size M where $M\leqslant N$. It must be remembered that the weights for these smaller clusters must be calculated with a degeneracy of M.

A specific example is helpful in demonstrating the algorithm. Figure 1 displays a successful attempt at forming a cluster of size $N=4$, on a square lattice. The order in which the neighbors were examined was chosen to be right, down, left, and up. Let us now consider various steps involved in construction of this cluster in detail. Beginning from the initial position labeled cell 1, the adjacent site to the right of this position is examined. In this case the site is rejected and the current position remains on the cell 1. Such rejected cells are indicated by the letter X. The next neighbor in the list is the one below, labeled cell 2. As it happens, this is accepted. Thus, the current position moves to this site and its position is added to the top of the stack, ahead of the position of cell 1. The process of examining the neighbors is resumed for sites adjacent to cell 2. Once again, following the strict order in the list, the site labeled 3 to the right of the current position is considered first. This is also accepted and as before is

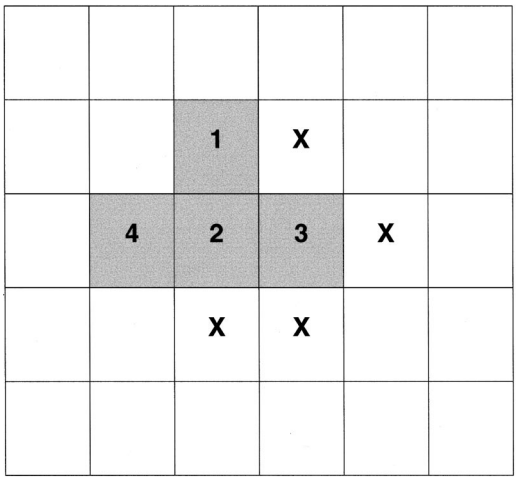

FIG. 1. Sequence of accepted sites leading to a cluster of size $N=4$. The sites examined but rejected along the way are indicated by *X*. In our notation this sequence can be represented by $\{0,1,1,0,0,0,1\}$.

placed at the top of the stack. At this stage the stack contains the positions of cells 3, 2, and 1, in that order. The current position is now cell 3.

The site to the right of this, followed by the one below, are tested and both rejected in succession. Since both the neighbors to the left (i.e., cell 1) and the one above have already been considered, the current position has no more eligible neighbors left to test. Therefore, following rule (v) above, site 3 is removed from the stack. This leaves the position of cell 2 at the top of the stack, making this the current position again. The cell 2 has two neighbors, the adjacent sites below and to the left, which are still unexamined. Of these, according to our agreed list, the site below takes precedence, but as shown in Fig. 1 this is rejected. The current position remains on the cell 2 and the neighboring site (cell labeled 4) to the left of this position is tested. As it happens, this is accepted. A cluster of the desired size $N=4$ is achieved, bringing this particular attempt to a successful end.

For the subsequent discussion, it is useful to represent a sequence of acceptance and rejections by a series of 1 and 0. Thus, for the case shown in Fig. 1 we have $\{0,1,1,0,0,0,1\}$. Note that at any stage throughout a series, the position of the current site and that of the neighbor to be examined, relative to the starting cell, are entirely specified by the decisions that have been made so far. In other words, given a sequence of 1s and zeros we can determine precisely the shape of the cluster that was constructed. This is only possible because of the manner in which the neighbors of the current position are always tested in a strict predefined order. For an algorithm that considers the neighboring sites at random, the same will clearly not be true.

The procedure described above needs to be repeated a large number of times, to obtain the weights for the ensemble average defined in Eq. (1). In particular, using Eq. (2), the number of lattice animals of size *N* can now be determined.

B. Ergodicity and degeneracy of the algorithm

Let us now discuss the issue of the ergodicity of the algorithm. We wish to see whether, starting from any particular site on a given cluster, a series of acceptances and rejections (1 and 0) can always be determined that leads to that cluster shape. We stress that we are not concerned about how probable such a sequence is likely to be, but merely that it exists. We can attempt to construct such a sequence by following the same rules as our algorithm described above, with one exception; we accept and reject each examined site according to whether it forms part of the target cluster shape or not. Obviously, in the original algorithm, each such move has a nonzero chance of occurring, provided *p* is not set to zero or 1. Since we accept only sites that belong to the cluster in question, it follows that if the sequence is successful we will achieve the desired cluster shape. However, we might argue that for some choice of target cluster and starting position, a series started in this manner will always terminate prematurely. That is to say, it will inevitably lead to a failure, with only part of the required cluster having been constructed. Now, it is easy to see that this cannot be true. If the series fails, it implies that all the neighboring sites of the subcluster formed so far are rejected. However, the rest of the cluster must be connected to this subcluster at some point. Hence, at the very least, one neighboring site of the subcluster must be part of the full cluster and could not have been rejected. Starting from any of the sites belonging to a cluster, then, it is always possible to write down a sequence of 1s and zeros that will result in the formation of that cluster. Similarly, considering every starting point on a cluster of size *N*, another implication of the above result is that the corresponding cluster shape can be generated in a minimum of at least *N* distinct ways.

Next, we shall show that the degeneracy of a cluster of size *N* in our algorithm is in fact exactly *N* (unlike the original algorithm of Care [12] which has a degeneracy of $N!$). Let us suppose that, starting from a particular site on a given target cluster shape, our algorithm has two distinct ways of forming this cluster. Associated with each of these, a series of 1s and zeros can be written down, in the same manner as that indicated above. The two ways of constructing the cluster must necessarily begin to differ from each other at some stage along the sequence, where we will have a 1 in one case and a 0 in the other. Now since up to this point the two series are identical, the site being examined at this stage will be the same for both cases. This is rejected in one sequence (hence 0) whereas it is accepted in the other (hence 1). It immediately follows that these two differing ways of constructing the cluster cannot result in the same shape. Using this result, together with the previous one regarding the ergodicity of the algorithm, we are led to conclude that, starting from a given site on a cluster, the algorithm has one and only one way of constructing the cluster. Hence, for a cluster of size *N*, the degeneracy is simply *N*.

C. Refined algorithms

1. Adjacent site stack

During the growth of the cluster a stack can be constructed of all the sites that are adjacent to the cluster and still available for growth. When a new site is added to the cluster, its neighbors are inspected in the predetermined sequence and any available ones are added to the top of this stack. (Note that this stack differs from that discussed in Sec.

II A.) The choice of site to be occupied can be made from all the adjacent sites in a single Monte Carlo decision. Thus, if we consider the underlying process in the method described above, at each step there is a probability p of the site being accepted and a probability $q=1-p$ of the site being rejected. We therefore need to generate a random number with the same distribution as the number of attempts needed to obtain an acceptance. The probability of making k attempts of which only the last is successful is

$$p_k = q^{k-1}p \tag{5}$$

where $1 \leqslant k < \infty$ and $\Sigma_{k=1}^{\infty} p_k = 1$. In order to sample from this distribution we note that the associated cumulative distribution C_m is given by

$$C_m = \sum_{k=1}^{m} q^{k-1}(1-q) = 1 - q^m. \tag{6}$$

Hence, if we generate a random number η, uniformly distributed in the range $0 < \eta < 1$, then a number m given by

$$m = \mathrm{Int}\left(\frac{\ln(\eta)}{\ln(q)} + 1 \right) \tag{7}$$

will have been drawn from the required distribution. Thus we generate the number m according to Eq. (7) and use this to determine which site on the stack is selected, with $m=1$ corresponding to the site at the top of the stack. If $m > N_{adj}$, where N_{adj} is the number of available adjacent sites, the cluster growth is terminated as explained in step (vi) in Sec. II A. All the adjacent sites lying above the chosen site in the stack are transferred into the list of rejected sites. The list of adjacent sites is then adjusted to include the new available sites adjacent to the recently accepted site. As before, it is crucial that these are added to the top of the list in the strict predefined order.

2. Variable probability

An apparent disadvantage of the methods so far described is that, with fixed choice of probability p, occasions arise when a cluster growth will terminate before reaching a cluster of size N, simply because the Monte Carlo choice rejected all the neighboring sites. This problem can be overcome if the value of p is allowed to vary as the cluster grows. The simplest method is to determine the number N_{adj} of available adjacent sites at each point in the cluster growth and select one of these sites with uniform probability. This effectively makes $p = 1/N_{adj}$ and thereby increases the chances of growing a cluster of size N. Note that it is still possible for a cluster growth to become blocked. This happens when the chosen site is the one at the bottom of the current eligible neighbor list, thus causing all the other neighboring sites in the list to be rejected in one step. If the newly accepted site has itself no unexamined neighbors to add to the list, the algorithm is terminated prematurely. Modified in the manner described above the weight associated with a cluster is now

$$W_\alpha = \frac{\Pi_{i=1}^{N} N_{adj}^{i}}{N} \tag{8}$$

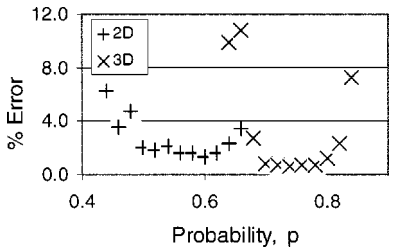

FIG. 2. Percentage errors for clusters of size 50.

rather than the expression given in Eq. (4).

However, when this variable probability method was tested it was found that, although it reduced the number of rejected clusters, it was inefficient at sampling the space of possible clusters when compared with the method described in Sec. II C 1. This inefficiency was measured by comparison of the standard deviation in the estimated cluster number for any given number of clusters in the sampling ensemble. It is thought that the inefficiency of the variable probability method arises because it gives too much weight to sites lower in the stack, yielding many nonrepresentative clusters. It is possible that this problem could be overcome by using a nonuniform sampling distribution (cf. [12]) but this was not tested in this work and the method described in Sec. II C 1 was used to obtain the results described in Sec. III.

III. RESULTS

In order to test the algorithm, the procedure described in Sec. II was used to estimate the number of lattice animals on a square 2D lattice and a simple cubic 3D lattice for which exact results are known up to certain sizes [5]. Before collecting data it was necessary to determine the optimum value of the probability p with which an adjacent site is accepted during the cluster growth. The effect of changing p on the estimated error in the number of clusters of size 50 on the 2D and 3D lattices can be seen in Fig. 2. It can be seen that there is a fairly broad range of values of p for which the error is a minimum and a value of $p=0.6$ was used to obtain the results described below for the 2D lattice and 0.72 for the 3D lattice. The distribution of weights is log normal [12] and becomes highly skewed for large cluster sizes; this is a standard problem with Rosenbluth methods [17]. The minimum in the error achieved by the choice of the value of the probability p has the effect of minimizing the variance of the distribution of the weights W_α.

It is worth noting that the algorithm does not sample the space of cluster shapes uniformly. Rather, cluster shapes have different probabilities of being sampled. However, the probability is known for each cluster shape and is corrected for through the weight factors associated with each cluster shape, as in Eq. (1). The likelihood of occurrence of a particular cluster shape is crucially dependent on the strict order in which the neighbors are examined, as described in first paragraph in Sec. II A. This may cause certain shapes to have small probabilities, or effectively become ''bottlenecks'' in the algorithm. The choice of the probability p will influence the type of cluster that becomes a bottleneck and it is likely that the values for p identified above minimize the effect of these bottlenecks. Alternatively, if the order of

TABLE I. Degenerate Rosenbluth estimate of the number of lattice animals of size N on a three dimensional square lattice using 2.5×10^7 sample clusters, each grown to $N=50$ with $p=0.72$; exact values from [6]; estimated values and associated errors from incomplete enumeration method of Lam and Family [6]; calculation of error estimate described in text; ''true'' error is fractional difference of Rosenbluth estimate and exact value; χ and ξ are defined in the text.

N	Rosenbluth estimate	Exact value	Ref. [6] estimate	e^{est} (% error)	True (% error)	Ref. [6] (% error)	χ	ξ
2	3.000×10^{0}	3						
3	1.499×10^{1}	15						
4	8.600×10^{1}	86	8.594×10^{1}	0.03	0.00	0.51	0.18	0.07
5	5.339×10^{2}	534	5.321×10^{2}	0.03	0.02	0.54	0.77	0.00
6	3.483×10^{3}	3 481	3.475×10^{3}	0.04	0.05	0.58	1.30	0.14
7	2.351×10^{4}	23 502	2.353×10^{4}	0.05	0.02	0.63	0.42	0.14
8	1.630×10^{5}	162 913	1.631×10^{5}	0.05	0.03	0.65	0.58	0.73
9	1.153×10^{6}	1 152 870	1.155×10^{6}	0.06	0.03	0.73	0.50	0.62
10	8.302×10^{6}	8 294 738	8.291×10^{6}	0.06	0.09	0.86	1.40	0.16
11	6.054×10^{7}	60 494 540	6.042×10^{7}	0.06	0.08	0.87	1.29	0.50
12	4.464×10^{8}	446 205 905	4.442×10^{8}	0.07	0.05	0.87	0.70	0.12
13	3.326×10^{9}	3 322 769 129	3.291×10^{9}	0.08	0.11	0.97	1.34	0.48
14	2.496×10^{10}		2.461×10^{10}	0.07		1.09		0.35
15	1.887×10^{11}		1.862×10^{11}	0.07		1.16		-0.10
16	1.436×10^{12}		1.416×10^{12}	0.10		1.22		0.25
17	1.098×10^{13}		1.082×10^{13}	0.10		1.27		-0.03
18	8.448×10^{13}		8.329×10^{13}	0.09		1.37		0.12
19	6.520×10^{14}		6.446×10^{14}	0.11		1.38		0.20
20	5.048×10^{15}		5.002×10^{15}	0.13		1.41		-0.07
21	3.929×10^{16}		3.897×10^{16}	0.14		1.47		-0.21
22	3.063×10^{17}		3.052×10^{17}	0.14		1.49		-0.42
23	2.399×10^{18}		2.391×10^{18}	0.16		1.61		-0.11
24	1.882×10^{19}		1.877×10^{19}	0.19		1.68		0.16
25	1.485×10^{20}		1.480×10^{20}	0.21		1.70		-0.02
26	1.169×10^{21}		1.168×10^{21}	0.21		1.75		-0.11
27	9.214×10^{21}		9.209×10^{21}	0.20		1.81		0.06
28	7.316×10^{22}		7.290×10^{22}	0.21		1.88		0.18
29	5.790×10^{23}		5.786×10^{23}	0.24		1.96		-0.12
30	4.600×10^{24}		4.610×10^{24}	0.25		2.01		0.44
31	3.674×10^{25}			0.26				-0.28
32	2.929×10^{26}			0.25				0.26
33	2.342×10^{27}			0.27				0.54
34	1.872×10^{28}			0.31				0.46
35	1.501×10^{29}			0.31				-0.32
36	1.199×10^{30}			0.32				0.33
37	9.631×10^{30}			0.39				1.08
38	7.691×10^{31}			0.35				0.18
39	6.203×10^{32}			0.40				0.27
40	4.984×10^{33}			0.45				0.54
41	3.999×10^{34}			0.43				0.35
42	3.205×10^{35}			0.46				0.23
43	2.605×10^{36}			0.49				0.35
44	2.100×10^{37}			0.62				2.32
45	1.684×10^{38}			0.71				0.43
46	1.353×10^{39}			0.69				0.65
47	1.087×10^{40}			0.58				0.36
48	8.892×10^{40}			0.68				0.53
49	7.223×10^{41}			0.79				0.02
50	5.789×10^{42}			0.75				0.78

TABLE II. Degenerate Rosenbluth estimate of the number of lattice animals of size N on a two dimensional square lattice using 2.5×10^7 sample clusters, each grown to $N=50$ with $p=0.60$; exact results from [5]; calculation of error estimate described in text; ''true'' error is fractional difference of Rosenbluth estimate and true value; χ and ξ are defined in the text.

N	Rosenbluth estimate	Exact value	e^{est} (% error)	True (% error)	χ	ξ
2	1.999×10^0	2				
3	6.000×10^0	6	0.02	0.01	0.22	-0.48
4	1.900×10^1	19	0.03	0.00	0.00	-0.65
5	6.300×10^1	63	0.03	0.01	0.31	0.14
6	2.160×10^2	216	0.03	0.00	0.00	0.36
7	7.601×10^2	760	0.04	0.02	0.43	-0.27
8	2.724×10^3	2 725	0.04	0.03	0.60	0.08
9	9.903×10^3	9 910	0.05	0.07	1.48	-0.14
10	3.644×10^4	36 446	0.05	0.01	0.21	0.10
11	1.352×10^5	135 268	0.06	0.04	0.69	0.09
12	5.056×10^5	505 861	0.07	0.04	0.66	-0.04
13	1.903×10^6	1 903 890	0.08	0.04	0.51	-0.24
14	7.205×10^6	7 204 874	0.09	0.01	0.06	-0.13
15	2.741×10^7	27 394 666	0.09	0.05	0.49	-0.33
16	1.046×10^8	104 592 937	0.09	0.01	0.07	-0.09
17	4.009×10^8	400 795 844	0.11	0.03	0.29	0.74
18	1.543×10^9	1 540 820 542	0.12	0.13	1.09	0.44
19	5.942×10^9	5 940 738 676	0.10	0.01	0.15	0.26
20	2.298×10^{10}		0.13			-0.42
21	8.895×10^{10}		0.15			-0.02
22	3.451×10^{11}		0.17			0.62
23	1.341×10^{12}		0.18			0.61
24	5.228×10^{12}		0.20			1.61
25	2.039×10^{13}		0.19			-0.04
26	7.970×10^{13}		0.26			-0.05
27	3.122×10^{14}		0.25			0.00
28	1.225×10^{15}		0.24			0.33
29	4.831×10^{15}		0.28			0.20
30	1.883×10^{16}		0.30			-0.13
31	7.426×10^{16}		0.33			0.97
32	2.945×10^{17}		0.45			0.59
33	1.160×10^{18}		0.34			0.19
34	4.561×10^{18}		0.47			0.44
35	1.800×10^{19}		0.40			0.23
36	7.121×10^{19}		0.52			0.29
37	2.823×10^{20}		0.57			0.67
38	1.122×10^{21}		0.67			-0.03
39	4.417×10^{21}		0.65			0.71
40	1.763×10^{22}		0.83			1.30
41	6.979×10^{22}		0.84			1.02
42	2.738×10^{23}		0.78			0.37
43	1.088×10^{24}		0.82			-0.16
44	4.341×10^{24}		0.93			2.12
45	1.704×10^{25}		0.97			0.52
46	6.802×10^{25}		1.10			0.73
47	2.673×10^{26}		1.07			0.41
48	1.058×10^{27}		1.02			0.60
49	4.209×10^{27}		1.14			0.26
50	1.664×10^{28}		1.28			0.29

examining the neighbors is altered, other shapes will become bottlenecks, but the original shapes with low probability now become much more likely. Hence the presence of bottlenecks in the algorithm could also be reduced by breaking up a large run into a number of smaller runs each with a different order of neighbor lists. The problem of bottlenecks does not, however, appear to have been significant for the cluster enumeration results presented below, which were obtained with a single order of examining the neighboring sites.

In Table I we present results obtained using the algorithm defined in Sec. II using the adjacent site stack method of Sec. II C to enumerate clusters on a simple cubic 3D lattice for clusters up to size 50. The results were obtained from an ensemble of 2.5×10^7 clusters. The data took 3.3 h to collect on an R5000 Silicon Graphics workstation using code written in the language C but with no attempt to optimize the code. Only 30% of the clusters achieved a size of 50. The results are quoted together with a standard error e^{est} calculated by breaking the data into 50 blocks and determining the variance of the block means for each cluster size. If the number of samples in each block is sufficient, it follows from the central limit theorem that the sampling distribution of the means should become reasonably symmetrical. We therefore also quote a *skewness* ξ defined by [18]

$$\xi = m_3/m_2^{3/2}, \tag{9}$$

where m_i is the ith moment about the mean of the sampling distribution. It is expected that $\xi \lesssim 0.5$ for a symmetrical distribution and $\xi > 1$ for a highly skew distribution. The statistic ξ should be treated with some caution, since it is likely to be subject to considerable error because it involves the calculation of a third moment from a limited number of data points.

Exact results are known for clusters up to size 13 [6]. In Table I we quote the values for the quantity χ defined by

$$\chi_M = \left| \frac{c_M^{exact} - c_M^{est}}{c_M^{exact} e_M^{est}} \right|, \tag{10}$$

where c_M is the number of clusters of size M. It can be seen that all the values of χ are $O(1)$. Hence we assume that e^{est} is an acceptable method of estimating the error in the method. However, it is likely that e^{est} will underestimate the true error if the distribution becomes more skew. We also quote in Table I the values of c_N calculated by Lam and Family [6] using a Monte Carlo incomplete enumeration method together with the error estimates reported for this method.

In Table II we quote data collected from a square two dimensional lattice by collecting data from 2.5×10^7 clusters up to size 50. This data took only 1.45 h to collect but only 2% of the clusters achieved a size of 50. Comparison is given with exact results [5] up to clusters of size 19. The rate of growth of errors for the two and three dimensional data is shown in Fig. 3, and it can be seen that the errors associated with the method are beginning to diverge quite rapidly above clusters of size 50. This behavior is to be expected with a technique that is based on sampling from a log normal distribution. In a previous paper [12], equivalent results were obtained for clusters up to size 30 with approximately the same sample size. The improvement up to clusters of size 50 obtained by this method arises because the weight associated with clusters of a certain size is generated from roughly half as many random numbers. This effectively halves the standard deviation of the log normal distribution of the weights and allows larger clusters to be sampled before the method becomes unusable.

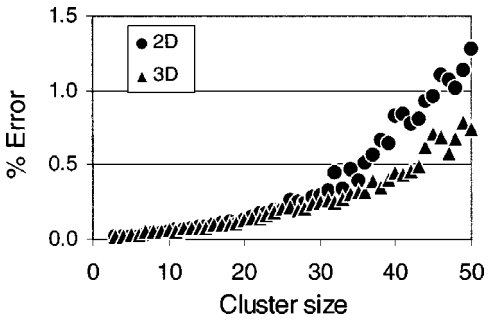

FIG. 3. Variation of percentage error with cluster size.

IV. CONCLUSIONS

We have described a simple Rosenbluth algorithm for the Monte Carlo enumeration of lattice animals and clusters, which can be applied to any lattice topology. A merit of the scheme is that for thermal systems it may be easily adapted to include Boltzmann weightings following, for example, the arguments used by Siepmann and Frenkel [19] in the development of the configurational bias technique. Similarly, the method can be applied to calculation of the averaged properties of a cluster of a given size, in the site percolation problem. In this case we have

$$\langle O \rangle = \frac{\langle (1-P)^t O \rangle_W}{\langle (1-P)^t \rangle_W} = \frac{\sum_{\alpha=1}^{N_E} W_\alpha (1-P)^{t_\alpha} O_\alpha}{\sum_{\alpha=1}^{N_E} W_\alpha (1-P)^{t_\alpha}}, \tag{11}$$

where P is the probability of site occupation in the percolation problem of interest and t_α the number of perimeter sites [20] of the cluster α. Preliminary results also indicate that the method may be useful in the study of the adsorption of clusters onto solid surfaces.

A possible numerical limitation of the method arises from the highly skew probability distribution of Rosenbluth weights that occurs for large cluster sizes. However, the method presented in this work is able to work to considerably higher cluster sizes than the one described in [12] before this becomes a problem.

[1] G. Jacucci, A. Perini, and G. Martin, J. Phys. A **16**, 369 (1983).
[2] B.F. Edwards, M.F. Gyure, and M. Ferer, Phys. Rev. A **46**, 6252 (1992).
[3] R.C. Ball and J.R. Lee, J. Phys. I **6**, 357 (1996).
[4] H.P. Peters, D. Stauffer, H.P. Hölters, and K. Loewenich, Z. Phys. B: Condens. Matter **34**, 339 (1979).
[5] M.F. Sykes and M. Glen, J. Phys. A **9**, 87 (1976).
[6] P.M. Lam and F. Family, Physica A **231**, 369 (1996).
[7] D. Stauffer, Phys. Rev. Lett. **41**, 1333 (1978).
[8] P.L. Leath, Phys. Rev. Lett. **36**, 921 (1976).
[9] S. Redner and P.J. Reynolds, J. Phys. A **14**, 2679 (1981).
[10] S. Redner and V. Privman, J. Phys. A **20**, L857 (1987).
[11] P.M. Lam, Phys. Rev. A **34**, 2339 (1986).
[12] C.M. Care, Phys. Rev. E **57**, 1181 (1997).
[13] M.N. Rosenbluth and A.W. Rosenbluth, J. Chem. Phys. **23**, 356 (1955).
[14] L. Pratt, J. Chem. Phys. **77**, 979 (1982).
[15] J. L. Martin, in *Phase Transitions and Critical Phenomena* edited by C. Domb and M. S. Green (Academic, New York, 1974), Vol. 3, p. 97.
[16] S. Redner, J. Stat. Phys. **29**, 309 (1982).
[17] J. Batoulis and K. Kremer, J. Phys. A **21**, 127 (1988).
[18] M. G. Bulmer, *Principles of Statistics* (Oliver and Boyd, London, 1965).
[19] J.I. Siepmann and D. Frenkel, Mol. Phys. **75**, 59 (1992).
[20] D. Stauffer and A. Aharony, *Introduction to Percolation Theory* (Taylor and Francis, London, 1992).