Enumerating 2D percolation series by the finite-lattice method: theory

1995 J. Phys. A: Math. Gen. 28 335

(http://iopscience.iop.org/0305-4470/28/2/011)

J. Phys. A: Math. Gen. 28 (1995) 335–349. Printed in the UK

Enumerating 2D percolation series by the finite-lattice method: theory

Andrew Conway†
Department of Mathematics, University of Melbourne, Parkville 3052, Australia

Received 11 May 1994

Abstract. This paper describes a transfer-matrix algorithm for enumeration of series of interest in percolation on the square lattice. It allows efficient generation of both low-temperature and high-temperature expansions, as well as the combinatorially interesting enumeration of undirected animals by area or perimeter, with moments of the other property.

1. Introduction

Percolation is an interesting problem in statistical physics. One common approach is to formulate the problem as sites on a lattice, where each site can be occupied or not according to some probability p. Then several macroscopic properties can be observed as a function of p, such as the probability of the existence of an infinite cluster. These functions are studied analytically, through Monte Carlo simulations, and through series enumeration and analysis. This paper describes a new algorithm for enumerating several series of interest to both the combinatorial and statistical physics communities on the two-dimensional square lattice. No analysis of results is done in this paper, that is left for a subsequent paper [1].

The most obvious combinatorial problem is the enumeration of animals (a connected subset of the lattice, also known as polyonimoes) by area or perimeter. Enumeration by site area has been given by Sykes and Glen [2] on several lattices. They found that on the square lattice, the total number of connected clusters with s sites grows asymptotically like $As^{-1}\lambda^s$ where $\lambda = 4.06 \pm 0.02$. They give enumerations up to $s = 19$. More recently, Redelmeier [3] used ten months of CPU time and a very optimized program to extend the enumeration to $s = 24$. Guttmann analysed this series to estimate $\lambda = 4.0626 \pm 0.0002$ in [4]. This has stretched the exhaustive enumeration algorithm for the two-dimensional square lattice to the practical limit: any significant extensions of this series will have to come from more efficient algorithms, such as the one described in this paper. There have been improvements on other lattices using significantly improved counting algorithms in [5].

Enumeration by both area and perimeter simultaneously is also possible and is very useful: *perimeter polynomials* (described later) can thence be directly obtained. A lesser, but still interesting combinatorial entity is the moment of the perimeter when enumerating by area, and vice versa.

As usual, define $g_{s,t}$ to be the number of connected clusters with area (sites or bonds) s and perimeter (sites or bonds) t.

† E-mail address: arc@mundoe.maths.mu.oz.au

0305-4470/95/020335+15$19.50

The more interesting objects of study are those directly relating to percolation. Suppose that each site (or bond) is occupied with probability p and unoccupied with probability $q = 1 - p$. Then the lattice will split up into various connected clusters. A phase transition occurs here for some critical probability p_c. At this phase transition, an infinite cluster appears for the first time. First consider the low-temperature regime ($p < p_c$).

The phrases *high temperature* and *low temperature* are used throughout this paper. They really mean high density† and low density. The word *temperature* is used, as the density often corresponds to temperature in statistical mechanical models involving percolation.

Let $\langle n_s \rangle(p)$ be the mean number per lattice site of clusters of area s. Then

$$\langle n_s \rangle = p^s \sum_t g_{s,t} q^t . \tag{1}$$

The *perimeter polynomials* $D_s(q)$ are then defined such that

$$D_s(q) = \frac{\langle n_s \rangle}{p^s} = \sum_t g_{s,t} q^t . \tag{2}$$

Enumeration of perimeter polynomials for the square lattice up to $s = 17$ is given in [2], together with a discussion of low-temperature percolation.

At low values of p (called the low-temperature or low density regime $p < p_c$), there is no infinite cluster. Since the probability of belonging to a cluster of size s is $s\langle n_s \rangle$, and every occupied site must belong to exactly one of these types of clusters, there is a formal relation

$$p = \sum_s s \langle n_s \rangle . \tag{3}$$

This is thus not a very interesting thing to compute: it is a useful test of an algorithm, but it is not going to yield any new information. However, that is only the first moment. The second moment is more interesting. Define

$$S(p) = \sum_n b_n p^n = \frac{1}{p} \sum_s s^2 \langle n_s \rangle . \tag{4}$$

Then $S(p)$ is the mean number of occupied sites connected to an occupied site, including that site itself. Thus $S(p) = 1 + O(p)$. This is also discussed in [2] for both site and bond problems. b_n is listed for up to 18 sites and 14 bonds. An analysis of this data in order to obtain an estimate for p_c is given in [6].

For values of $p > p_c$ (high temperature), the problem is significantly different, as there is now a possibility of an infinite cluster, so (3) no longer holds. Instead, it must be modified to

$$p = p_\infty + \sum_s s \langle n_s \rangle \tag{5}$$

where p_∞ is the probability that a given site is in an infinite cluster, and the percolation probability $S(p)$ is the probability that a given occupied site is in an infinite cluster:

$$P(p) = p_\infty / p . \tag{6}$$

So one wishes to calculate $P(p)$, but this cannot be obtained from the $\langle n_s \rangle$ values from before: they will all cancel out and are not valid for $p > p_c$. The solution is to work

† High density means that the average density is above the critical density.

with perimeters rather than areas, which will give a function $P(p)$ in terms of a convergent series in q. This will not converge for $p = 0$ and $q = 1$:

$$P(p) = 1 - \frac{1}{p} \sum_t \sum_s s p^s q^t g_{s,t} \,. \tag{7}$$

Note that in the low-temperature regime this is still identically zero: it becomes $1-(1/p)p$.

The generalization of the mean size of finite clusters, $S(p)$, given for low temperatures by (4) becomes

$$S(p) = \sum_n b_n p^n = \frac{1}{p_f} \sum_s s^2 \langle n_s \rangle \tag{8}$$

where $p_f = p - p_\infty$ is the probability that a given site is in a finite cluster. A discussion of high-temperature percolation in two dimensions, together with enumerations of $P(p)$ and $S(p)$ for sites and bonds, for the triangular, square and honeycomb lattices can be found in [7] and the analysis in [8].

Work in three dimensions can be found in, for example, [9–12], and in higher dimensions in [13]. A detailed perspective more from a computer programmers' viewpoint can be found in [14].

In the next section the algorithm is described in detail, first in general principles and then with specific details for the various series that have been enumerated using this method. In section 3 the computational complexity of the algorithm for each series is analysed, and a brief conclusion is given in section 4. A subsequent paper [1] will present the results and analysis of the series obtained by this algorithm.

2. Algorithm

The algorithm that I give here is based on the transfer-matrix paradigm, and follows a similar pattern to the discussion of the self-avoiding paths [15] algorithm, in that work is performed on a finite rectangular lattice, with the transfer-matrix process being applied to a list of partial generating functions stored according to a signature. Again, the application of the transfer matrix is performed one site at a time, and repeated for all sites of the lattice. When a cluster is completed, the partial generating function is accumulated into a final resultant generating function.

The differences are:

- different information is kept;
- the boundary crosses sites rather than bonds;
- different objects are being enumerated, so that the signature is very different;
- similarly, the starting, stopping and transitions are very different;
- different uniqueness conditions are used, and very little algebraic manipulation is needed after the computation;
- the time/space/length complexity is different.

These issues will now be covered separately.

2.1. Information conserved

Each time one applies the transfer-matrix step, adding an additional site, one can increase the total area by a certain amount δs, and the total perimeter by an amount δt. Note that the perimeter will not always be the total perimeter of the current object; potential perimeter

Table 1. Meaning the variable represented by u and the moment represented by x in $F(u, x)$, together with the amount by which to multiply a partial generating function to add in δs area and δt perimeter.

Variable represented by u	Moment represented by x	Factor to account for δs and δt
Area	Perimeter	$u^{\delta s}(1+x)^{\delta t}$
Perimeter	Area	$u^{\delta t}(1+x)^{\delta s}$
p		$u^{\delta s}(1-u)^{\delta t}(1+x)^{\delta s}$
$q = 1-p$		$u^{\delta t}(1-u)^{\delta s}(1+x)^{\delta s}$

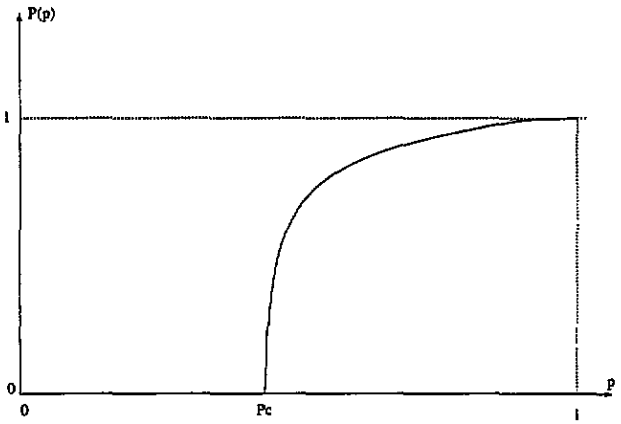

Figure 1. A graph of the percolation probability $P(p)$.

sites to the right and below the current site being added are for the moment ignored as they will be processed later.

The most general method of storing generating functions would be to maintain a two-variable generating function $\sum_{s,t} g_{s,t} u^s v^t$. Then one multiplies the partial generating function being worked with by $u^{\delta s} v^{\delta t}$ and accumulates into the new partial generating function.

However, this consumes a large amount of memory which is the ultimate limitation. It could be more useful just to try to obtain the information relevant to the particular problem being considered. One can obtain some function of one variable, say $F_0(u) = \sum_n f_n u^n$, and some of its moments: $F_w(u) = \sum_{n,m} m^w f_{n,m} u^n$. Then one keeps a two-variable generating function. One variable is u as above: the other is x and is related to m. The way it is related is that $F(u, x) = \sum_{n,m} (1+x)^m f_n u^n$. Then if $F(u, 0) = \sum_i F^{[i]} x^i$, from a binomial expansion of $(1+x)^m$ we obtain $F_0(u) = F^{[0]}(u)$, $F_1(u) = F^{[1]}(u)$, and $F_2(u) = 2F^{[2]}(u) + F^{[1]}(u)$, etc. This turns out to be a convenient representation which can be easily understood. This two-variable generating function has the advantage of growing linearly with the length of the series rather than quadratically.

The meaning of u depends upon the particular quantity being evaluated. See table 1.

Typically one is mainly interested in the zeroth, first and second moments, so one only need keep the generating functions up to order x^2. Also, the transfer-matrix technique only determines the series up to a certain power of u, so one keeps the generating function up to $u^{u_{\max}}$. It is usually worthwhile keeping the generating function to one higher power, however, as the first correction term can often be easily and efficiently calculated.

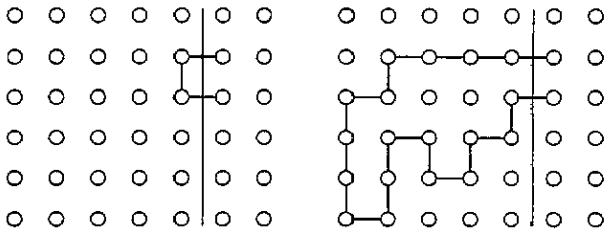

Figure 2. Two examples of a partial path, each with a boundary (vertical line) and the same boundary conditions.

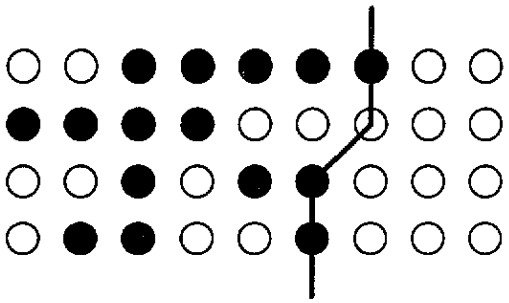

Figure 3. A typical boundary for a percolation calculation.

2.2. *Moving boundary condition labelling*

When using the transfer-matrix technique to enumerate paths in a strip, the moving boundary† cuts across bonds. For percolation, it is more useful for the boundary to cut across sites (see figure 3).

Now we need some method of labelling each boundary condition—to make a 'signature' for it. This will involve one digit for each site on the boundary. Suppose we are dealing with a boundary W sites wide. Then there will be W sites on the boundary, and W digits in the signature.

This can be done with eight different symbols defined in table 2. The symbols used are the digits 0 to 7. Note that 2 and 3 are not used for site percolation; that there must be exactly as many 5 sites as 7 sites; and that a 6 site uniquely specifies which cluster it is connected to as the clusters cannot cross.

Table 2. Meaning of digits used in labelling the boundary.

Digit	Meaning
0	Unoccupied site with no perimeter sites already counted
1	Unoccupied site with one perimeter site already counted
2	Unoccupied site with two perimeter sites already counted
3	Unoccupied site with three perimeter sites already counted
4	Occupied site not attached to any others on boundary
5	The first (of at least two) connected, boundary, occupied sites
6	An intermediate (of at least two) connected, boundary, occupied sites
7	The last (of at least two) connected, boundary, occupied sites

† The line that cuts down through the lattice and moves across one site each step—see figure 2.

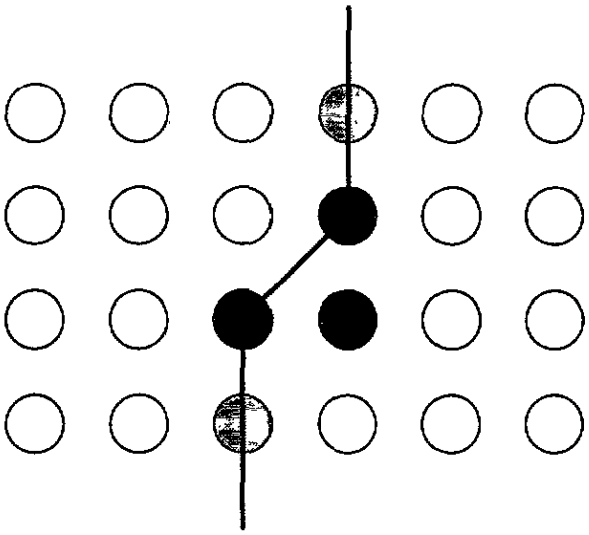

Figure 4. The boundary, showing the next site to be added (black), the above and left sites (dark grey) and the other boundary sites (light grey).

The number of possible boundary conditions for a given width w is then trivially no more than than 8^w, and in fact is significantly less (see section 3 for details).

2.3. *Transitions*

The mechanics of the transfer-matrix technique involve adding one site at a time. This section will discuss the effects of adding one site.

The order used for adding sites will be first all in the leftmost column, starting from the top and working downwards. Then all in the second column from the left, working downwards. One continues to the right until all columns are finished.

Note that one must have one extra row on the bottom and column on the right. However, one is not allowed to do any operation on these extra sites which would increase the area of the animal: it is solely to make sure that the perimeter is correct as one only works on the perimeter above and to the left of the current site.

The site being added will be on the corner of the current boundary, as shown in figure 4. The two most important sites are the *above* site and the *left* site. These are marked in dark grey in figure 4. Basically, the above and left sites are the only sites that can directly influence the current site. If one is on the top edge of the lattice, one can assume an unoccupied, unattached site (type 0) as the above site.

The *left* site is the most important as it is in a sense being extended. The *above* site will strongly affect what is going on, and is likely to be changed in the process. Other sites on the signature may be changed if clusters touch at the new site.

First consider what is going to happen if the site is unoccupied (unattached if one is talking about bonds). This will not be a possible transition if the left site is of type 4 (lone, unattached to any other site on the boundary). The problem is that this would leave the cluster which was attached to the left site as totally unattached, meaning that there would now be more than one cluster, with no hope of ever joining them. Since we only want to count connected clusters, this is not acceptable. The only case when this could

be considered, is when the left site is the *only* occupied/attached site on the boundary, in which case leaving the new site blank would mean that we have finished a cluster, and the partial generating function should be accumulated into the final result. This is the only way of finishing a cluster.

If the site to the left is anything other than a type 4 site, it is perfectly possible to have a blank site as the current site. Note that if the left site is either a type 5 or 7, the rest of the signature is going to have to be relabelled, as the start or end of a connected cluster is vanishing. To do this, one finds the nearest site on the boundary in the connected cluster (down if the site to the left is a type 5; up if it is a type 7). This site then changes number. If it is a type 6 (middle), replace it with the type 5 or 7 which was destroyed. If it is a type 7 or 5 (the other end), replace it with a type 4 (unattached to anything else). The above site will not change directly, although it may of course be changed indirectly due to the procedure just described.

Trivially, no area ($\delta s = 0$) is added as a result of not adding a site (or any bonds). Then one must work out how many perimeter sites are being added. For the site problem, there will be one perimeter site added if either or both of the left or above sites are occupied, otherwise there will be no perimeter added. For the bond problem, there is one perimeter site added for each of the occupied sites to the left or above. Thus there may be 0, 1 or 2 perimeter sites added. The code for the new site will be the number of perimeter sites added.

Next consider (in the bond case) what happens in the next simplest case: there is one bond added from the above site to the current site, but no bond added from the left site. Like the above case, this is only possible if the left site is not a type 4 site. If the left site is a type 5 or 7 site, the rest of the signature must be changed to reflect its burial. The above site itself will probably be altered by adding a bond to it. If the above site was empty, or if it was a type 4 (alone), the above site will become type 5 (start of a cluster), and the new site will be a type 7 (end). If the above site was a type 5 (top) or type 6 (middle), then it will not change, and the new site will be a type 6 (middle). If the above site is a type 7 (end) then the above site will become a type 6 (middle), and the new site will be a type 7. The area being added is one bond. The perimeter being added can come from two places:

- there will be one bond perimeter for the bond from the new site to the left site;
- if the above site was blank, there may be some perimeter sites from that site back to the sites above and left. The number of such perimeter sites will be two minus the number type of the site. This is the whole point of having different types of blank sites.

We have now treated every possible case where the left site is *not* connected, so we now consider the case where it *is* connected.

First, consider the problem in bond percolation when the bond to the left is connected, but the bond to the site above is not connected. The type of the new site will then be exactly the type of the left site, unless the left site was unconnected, in which case the new site will be a type 4 site (unconnected). There will be one unit of area added, and the following sources of additional perimeter:

- there will be one perimeter bond for the bond between the current site and the site above. If the site above is blank, it must have its number increased by 1 to mark the fact that this bond has just been counted;
- if the left site is blank, there may be perimeter sites off it. The actual number of these will be 3 minus the numerical type of the left site.

Now consider the case of adding a site (site percolation) or adding two bonds (bond percolation). These are very similar if the above site is occupied. In either case, the area

Table 3. Transitions in percolation when (*a*) adding a site in site percolation, or (*b*) adding two bonds when doing bond percolation. The row indicates the left site, the column indicates the site above. The two digits are the new left/current site values for the signature.

Above → Left ↓	Blank (site)	Blank (bond)	4 (unattached)	5 Start	6 Middle	7 End
Blank	1, 4	5, 7	5, 7	5, 6	6, 6	6, 7
4	1, 4	5, 7	5, 7	5, 6	6, 6	6, 7
5	1, 5	5, 6	5, 6	5, 6[a]	6, 6[a]	6, 6
6	1, 6	6, 6	6, 6	5, 6	6, 6	6[b], 7
7	1, 7	6, 7	6, 7	5, 7	6, 7	6[b], 7

[a] Indicates that a 5 has been converted into a 6, and the corresponding 7 should be converted to a 6.

[b] Indicates that a 7 has been converted to a 6, and the corresponding 5 should be converted to a 6.

added is 2 for the bond case and 1 for the site case. Also, the contribution to perimeter from the left site will be 0 if that site is occupied/attached, and then

- 1 minus the numerical value† of the left site for site percolation
- 3 minus the numerical value of the left site for bond percolation.

The new values for the above site and the new site can be read from table 3. Note that some of these transitions involve the joining of two clusters and require a change elsewhere in the signature.

If the site above is occupied, there is no extra perimeter due to it, and the total perimeter added will just be the contribution from the left site.

If the site above is not occupied, then for bond percolation it will become occupied as we are adding a bond to it from the current site. In this case the perimeter added due to the above site will be 2 minus the numerical value of the site above. For site percolation, the perimeter added will be 1 minus the numerical value of the site above. If the site above was type 0, it will become type 1.

2.4. Uniqueness

Uniqueness in the vertical direction can be guaranteed by performing the calculations twice with widths differing by 1. For any given length of the lattice being processed, the difference in the generating functions between a strip of width w and width $w - 1$ will only contain those animals that actually use the full width. Thus uniqueness in the vertical direction is obtained.

For site percolation, uniqueness can be guaranteed in the horizontal direction in a similar way to that done for self-avoiding walks. That is, one does not propagate the zero signature past the first column. This forces all animals to have at least one site in the first column, guaranteeing uniqueness in the horizontal direction.

For bond percolation, the situation is a little more complex. In order to maintain symmetry, one does not want to have any bonds going out of the finite lattice. One does not allow any zero signatures past the second column; nor does one allow any signatures not containing a horizontal bond past the second column. This can be implemented with a flag without worrying about a blow-out in memory since it takes a few columns for the bulk of the signatures to appear, by which time the flag is set on all of them.

† The numerical value of a site is the code number assigned to it.

2.5. *Obtaining results*

The above method allows one to get a generating function for all the animals that fit inside a finite rectangular lattice of width w.

Symmetry can be used to increase the area covered. If $f_L(u,x)$ is the generating function for a length L, then $2f_L(u,x) - f_w(u,x)$ is the generating function for all animals which are no more than w in extent in one direction, and no more than L in extent in the other direction. One must then calculate which are the animals which have the lowest power in u that do not fit into this region, as they will cause the first error in the generating functions. One can also then calculate the largest value of L to which it is worth progressing. Note that the computation time goes linearly in L but exponentially in w, and the memory requirement is pretty much independent† of L, so L should be taken as far as is useful for a given w.

2.5.1. Low-temperature site percolation and enumeration by sites. For low-temperature site percolation and site enumeration, the number of sites is the variable that determines the power of u. Thus we want to find the animal with the fewest sites that has a horizontal and vertical extent of at least $w+1$. This is clearly an animal arranged like some permutation of a cross, with $w+1$ sites going 'vertically', and $w+1$ sites going 'horizontally'. Two of these sites overlap, so such an animal has $2w+1$ sites.

However, these animals can be counted efficiently, so it is easy to find the correction term, so one can actually get $f(u,x)$ accurate up to and including u^{2w+1}. For this one needs $L = 2w+1$. Without the correction term, $f(u,x)$ is accurate to u^{2w}.

Two examples of these animals are given in figure 5. They can be reasonably efficiently calculated recursively, especially with dynamic programming techniques. The basic algorithm is to consider a vertical strip at a time, and effectively do a transfer-matrix-type operation to go from one strip to the next. Unlike previous uses of the transfer-matrix technique in this paper, one does a whole strip at a time. As these animals are so restricted,

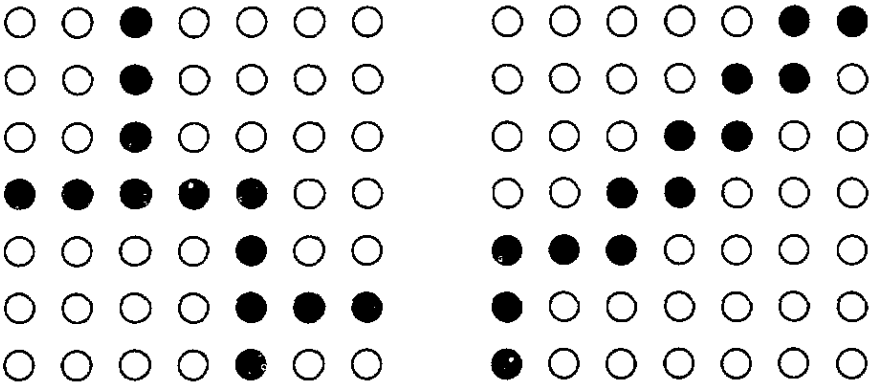

Figure 5. Two examples of the types of animals which cause the first error in the generating functions for low-temperature site and bond percolation calculations and area (site and bond) enumeration.

† Obviously, for $L = 0$, the memory requirement is zero, but once L has reached about w, the memory requirement does not change much with increasing L.

there are very few possible boundary configurations, (a polynomial in the width), so this algorithm is fast. The boundary conditions consist of the top occupied cell t, the bottom occupied cell b†, and knowledge of whether the top or bottom of the $w+1$ square grid have been reached yet. Here is a description of the algorithm suitable for a recursive computer program:

- *if one has reached the last column, then count one animal iff both the top and bottom* have been reached. Otherwise,
- if both the top and bottom have been reached, then there is no more room to manoeuvre vertically, and the rest of the animal must have a horizontal line. This allows $t-b+1$ animals, as the horizontal line may come from any of the $t-b+1$ sites on the current boundary;
- If neither the top nor bottom have been reached, one may either continue straight along, or have a vertical line from either the top or the bottom, subject to the constraint that it must join on to the original animal;
- *if the bottom has been reached, but not the top, one can climb upwards.* That is, the next column will be a vertical line going upwards from the current top site t to any new site between t and $w+1$ inclusive;
- if the top has been reached, but not the bottom, one can go downwards. The next column will be a vertical line going from the current bottom site b to any new site between 1 and b inclusive.

This just counts the number of missed animals of $2w+1$ sites: one also wants to know the sum of perimeters and squared perimeters of these animals. This can be calculated as a straightforward, meticulous modification of the above algorithm, which is not described in this paper. Of course low-temperature site percolation does not need the perimeter, just the number of sites which is $2s+1$.

A list of correction terms is given in table 4.

Table 4. An enumeration of the number of animals of the type shown in figure 5, which thus provide the error terms in enumerations by sites and bonds.

w	Number of animals
0	1
1	4
2	25
3	120
4	497
5	1924
6	7265
7	27288
8	102745
9	388692
10	1477721
11	5643064
12	21632785
13	83204260

2.5.2. *Low-temperature bond percolation and enumeration by bonds.* For low-temperature bond percolation and bond enumeration, the number of bonds is the variable that determines

† All cells in between must be occupied.

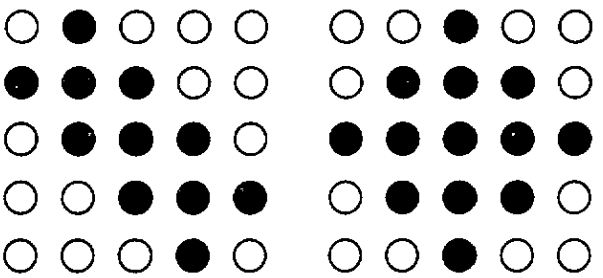

Figure 6. Two examples of the types of animals which cause the first error in the generating functions for high-temperature site percolation calculations and site perimeter enumeration.

the power of u. Thus we want to find the animal with the fewest bonds which has a horizontal and vertical extent of at least $w+1$. This is again clearly an animal arranged like some permutation of a cross, with w bonds going vertically, and w bonds going horizontally. They will cross at some site, but the bonds do not overlap, so such an animal has $2w$ bonds.

However, these animals can also be efficiently counted, so it is easy to find the correction term, so one can actually get $f(u,x)$ accurate up to and including u^{2w}. For this one needs $L=2w+1$. Without the correction term, $f(u,x)$ is accurate to u^{2w-1}.

To get the correction terms, one is looking for exactly the same animals as for low-temperature site percolation (figure 5), so one can use the same algorithm and table 4. The moments are particularly easy to generate, as each animal will have an area of $2w$ bonds and a perimeter of $4w+4$ bonds†.

Thus one can use table 4 to get corrections to all problems described here.

2.5.3. High-temperature site percolation and enumeration by site perimeter. For high-temperature site percolation and site perimeter enumeration, the number of perimeter sites is the variable that determines the power of u. Thus we want to find the animal with the fewest perimeter sites that has a horizontal and vertical extent of at least $w+1$. This set of animals is harder to imagine than the animals for the low-temperature correction terms, but it can be seen that such an animal will have $w+1$ perimeter sites on the right, $w+1$ on the left‡, and two extra perimeter sites, one at the top and one at the bottom. Some examples of such animals are given in figure 6. Such animals have perimeter $2w+4$.

These animals can be counted very efficiently, so it is easy to find the correction term, and one can actually get $f(u,x)$ accurate up to and including u^{2w+4}. For this one needs $L=w+1$. Without the correction term, $f(u,x)$ is accurate to u^{2w+3}.

The number of animals needed for the correction term is easy to calculate: if one looks at figure 6, one notes that the shapes of the animals can be stretched in such a manner that there is one animal for each of the sites in the top row, excluding the leftmost and rightmost sites. Figure 6 shows an extreme and an intermediate such animal. Thus there are $w-1$ animals in the correction term.

There are so few animals that it is easy and fast to enumerate them directly and count the area for each. Table 5 contains these correction terms. These are the correction terms for both high-temperature percolation and the site moments of enumeration by perimeter.

† Each of the $2s+1$ sites which bonds are attached to will have four bonds coming out. Of the resulting $8w+4$ bonds, there is a $2w$ overlap, $2w$ are occupied and $4w+4$ are perimeter bonds.

‡ They must be separated by the animal which has vertical extent $w+1$ sites.

Table 5. Correction terms for high-temperature percolation.

Width w	Number of animals	Total area	Second moment of area
1	0	0	0
2	1	5	25
3	2	16	128
4	3	35	411
5	4	64	1040
6	5	105	2261
7	6	160	4416
8	7	231	7959
9	8	320	13472
10	9	429	21681
11	10	560	33472
12	11	715	49907
13	12	896	72240
14	13	1105	101933
15	14	1344	140672
16	15	1615	190383
17	16	1920	253248
18	17	2261	331721
19	18	2640	428544

2.5.4. High-temperature bond percolation and enumeration by bond perimeter. For high-temperature bond percolation and bond perimeter enumeration, the number of perimeter bonds is the variable that determines the power of u. Thus we want to find the animal with the fewest perimeter sites which has a horizontal and vertical extent of at least $w+1$. This set of animals is easy to describe: there will be $w+1$ sites reachable in each direction, and on either side of these there will be $w+1$ perimeter bonds. Each of the two directions has two sides, so such an animal must have at least $4w+4$ perimeter bonds. One example of such an animal is a cross; another example is a full square of side length $w+1$.

These animals are very difficult to count, so it is not easy to find a correction term. No such term has been found in this paper, so one can get $f(u,x)$ accurate only up to and including u^{4w+3}.

2.6. *Other technicalities*

When writing a program to implement this algorithm, similar sorts of problems exist as in the case of path enumeration algorithms [15], with similar solutions. In particular, a hash table to access the different signatures, containing a pointer to the partial generating function is an efficient method of using memory.

As with self-avoiding paths, modular arithmetic can cut down the memory requirements significantly. A little more care needs to be taken as negative numbers may crop up.

One way to save a few per cent of memory is to notice that for high-temperature percolation, the first four coefficients $1 \ldots u^3$ are zero. Thus there is no point storing them. To actually implement this is difficult as they are not necessarily zero for the partial generating functions. However, a very easy optimization is *not* to add in the final perimeter site when doing an accumulation. This cuts out one of the perimeter sites and saves a little memory with very little effort.

3. Complexity

As usual for the transfer-matrix method, the complexity in terms of both time and memory of this algorithm is primarily determined by the number of different signatures for a given width w. This grows exponentially with w. It is this exponent that determines the overall usefulness of this algorithm. Note that in the rest of this section I will say that the complexity grows like λ^n when I really mean polynomial times λ^n. The polynomial tends to be unimportant as the differences in λ tend to be very large and quickly swamp the polynomial.

As mentioned in section 2.2, there is an upper bound to the number of boundaries of 8^n (for bond percolation) and 6^n for site percolation. The actual performance is much better, as many signature combinations are illegal. For instance, one needs to have exactly as many start (5) sites as end (7) sites, and middle (6) sites cannot appear outside a start/end pair. These do not really change the exponent: they just reduce the complexity by a polynomial. For more significant changes, we must look at the site and bond cases separately.

First consider site problems. In this case neither an unattached (4) nor an empty (0) site can appear next to any non-empty site. Neither can a 5 follow anything other than a 1, nor can a 7 precede anything other than a 1†. In fact, these restrictions make the complexity grow somewhat better than 3.3^w (details later). Since the total number of terms obtainable goes up by 2 for each extra value of w, this means that the computational complexity of the site problem grows like 1.8^n. This can be compared with direct enumeration which grows worse than 5^n for low-temperature-type calculations, and faster than exponentially for high-temperature-type calculations. Alternatives, such as the shadow method of Sykes, as described by Martin in [14] are more efficient than direct enumeration for low values of n and high dimensions, but grow faster than exponentially. However, the transfer-matrix method is more efficient even for the values of n currently practical for present computers for two dimensions.

For bond percolation the situation is worse, as the only restrictions are on the blank sites: 0 or 1 cannot appear between two (not necessarily mutually) attached sites; 0 or 3 cannot appear between one attached and one unattached site, and 2 or 3 cannot appear between any two unattached sites. Thus the complexity will grow like a 6^w. For low-temperature percolation and bond enumeration this means that the complexity grows like $\sqrt{6}^n \approx 2.45^n$, and for perimeter bond counting and high-temperature bond percolation, the complexity grows like $\sqrt[4]{6}^n \approx 1.5^n$, which is significantly better than direct enumeration, for which the growth rates are about 5^n and faster than exponential, respectively.

3.1. Site percolation complexity

The total number of possible boundary conditions for a cross section of w sites can be given by the coefficient of x^w in the generating function M given by the following algebraic language, with all labels (integers) changed to x:

$$M \to Z, Z1M, 41M, 5C71M, \epsilon, 4, 5C7 \tag{9}$$

$$Z \to 0Z, 0 \tag{10}$$

$$C \to \epsilon, 1G, 6C \tag{11}$$

$$G \to \epsilon, 1G, Z1G, 41G, 5C71G, 6C\,. \tag{12}$$

This is actually an unambiguous grammar, though it is not in an obviously unambiguous form. See for instance [16, 17] for usage of algebraic languages in this fashion.

† These restrictions break down at the current site being processed, as this represents a break in the signature.

Table 6. Complexity of the percolation algorithm for a given width.

w	Number of boundary conditions
1	3
2	7
3	18
4	46
5	121
6	323
7	878
8	2422
9	6774
10	19174
11	54852
12	158380
13	461045
14	1351715
15	3987990
16	11831270
17	35273290
18	105625010

An expansion of this series is given in table 6. Note that the actual amount of memory needed will be significantly more than table 6 indicates for two reasons:

- for intermediate positions in the transformation (that is, when there is a diagonal kink in the boundary), there is reduced connectivity at the link, allowing possibilities that would not otherwise be allowed. This will increase the number of possible boundary conditions to a number between the entry for w and the entry for $w+1$ in table 6. It is not straightforward to calculate this number directly;
- there will be two copies of most boundary conditions, since one has an 'in' set, and an 'out' set. This will almost double the total memory requirements, depending upon what order the partial generating functions are processed.

These effects only affect the results by a constant factor.

The generating function $M(x)$ may be analysed for its critical point, which turns out to be roughly 0.309017 (reciprocal 3.23607). Thus the complexity of site percolation using this method grows like a polynomial times $3.23607^{n/2}$.

4. Conclusion

The algorithm just presented can be used to enumerate several series of interest and their moments involving two-dimensional clusters, with algorithmic complexity exponentially better than previous algorithms.

The methods for producing both high- and low-temperature series, and enumeration by area and perimeter (with moments of perimeter and area conversely) have been given. Slight modifications can be used to get more complex moments, containing information on both bonds and sites using this method.

Calculations have been performed on an IBM RS6000, giving series of length 25 (18) for site (bond) low-temperature percolation, and 24(33) for site (bond) high-temperature percolation. These will be given and analysed in a subsequent paper.

Acknowledgments

I would like to thank the University of Melbourne for the award of the A O Capell, Wyselaskie and Daniel Curdie scholarships. I would like to thank Tony Guttmann for his advice and critical reading of this manuscript, and Ian Enting for his suggestion to use the finite lattice method for enumerating percolation series.

References

[1] Conway A R and Guttmann A J 1994 On two-dimensional percolation *J. Phys. A: Math. Gen.* submitted
[2] Sykes M F and Glen Maureen 1976 Percolation processes in two dimensions I. Low-density series expansions *J. Phys. A: Math. Gen.* **9** 87–95
[3] Redelmeier D Hugh 1981 Counting polyonimoes: yet another attack *Discrete Mathematics* **36** 191–203
[4] Guttmann A J 1982 On the number of lattice animals embeddable in the square lattice *J. Phys. A: Math. Gen.* **15** 1987–90
[5] Mertens S 1990 Lattice animals: a fast enumeration algorithm and new perimeter polynomials *J. Stat. Phys.* **58** 1095–108
[6] Sykes M F, Gaunt D S and Glen Maureen 1976 Percolation processes in two dimensions II. Critical concentrations and the mean size index *J. Phys. A: Math. Gen.* **9** 97–103
[7] Sykes M F, Gaunt D S and Glen Maureen 1976 Percolation processes in two dimensions III. High density series expansions. *J. Phys. A: Math. Gen.* **9** 715–24
[8] Sykes M F, Gaunt D S and Glen Maureen 1976 Percolation processes in two dimensions IV. Percolation probability *J. Phys. A: Math. Gen.* **9** 725–30
[9] Sykes M F, Gaunt D S and Glen Maureen 1976 Percolation processes in three dimensions *J. Phys. A: Math. Gen.* **9** 1705–12
[10] Sykes M F 1986 Generating functions for connected embeddings in a lattice: IV. Site percolation *J. Phys. A: Math. Gen.* **19** 2431–37
[11] Sykes M F and Wilkinson M K 1986 Derivation of series expansions for a study of percolation processes *J. Phys. A: Math. Gen.* **19** 3415–24
[12] Gaunt D S and Sykes M F 1983 Series study of random percolation in 3d *J. Phys. A: Math. Gen.* **16** 783–800
[13] Adler Joan, Meir Yigal, Aharony Amron and Harris A B 1990 Series study of percolation moments in general dimension *Phys. Rev.* B **41** 9183
[14] Martin J L 1990 The impact of large scale computing on lattice statistics *J. Stat. Phys.* **58** 749–74
[15] Conway A R and Guttmann A J 1993 Enumeration of self avoiding trails on a square lattice using a transfer-matrix technique *J. Phys. A: Math. Gen.* **26** 1535–52
[16] Bétréma J and Penaud J G 1991 Animaux et arbres guingois *Series formelles et Combinatoire Algebrique* ed M Delest, G Jacob and P Leroux p 85–102
[17] Delest M 1991 Polyominos and animals: some recent results *J. Math. Chem.* **8** 3–18