

University of Sofia “St. Kliment Ohridski”
Faculty of Physics

Mini Research Project:
**Percolation on a Two-Dimensional
Square Lattice**

Kalin Arsov
2nd year undergraduate student, ID 8861

Adviser: Prof. Dr. Ana Proykova
Department of Atomic Physics

Sofia

Summer, 2003

Contents

1	Introduction	1
1.1	What is percolation?	1
1.2	Applications	2
1.3	Exact solutions	4
2	Labelling algorithms	5
3	Scaling and Renormalisation Group	9
3.1	Quantities of Interest in Percolation	9
3.1.1	Percolation probability (Cluster strength) $P(p)$	9
3.1.2	Cluster size distribution n_s	9
3.1.3	Mean cluster size $S(p)$	10
3.1.4	Correlation length ξ	10
3.2	Scaling	11
3.2.1	Finite-size scaling	11
3.3	Renormalisation Group	12
3.3.1	Square Lattice	12
3.3.2	The Triangular Lattice	14
4	Results	15
4.1	Common Errors	15
4.2	Cluster-Size Distribution	16
4.2.1	Corrections to Scaling	17
4.2.2	Influence of Boundary Conditions	19
4.3	Wrapping Probability	20
4.4	Mean Cluster Size	22
	Acknowledgements	24
	A Programme Layout	25
	References	35

Chapter 1

Introduction

1.1 What is percolation?

The name *percolation* [1] derives from water flow through a layer of ground coffee in every percolator. Hence, the percolation problem is not a new one, however, it is not as narrow as one would think.

Research in percolation started in 1941 with Paul Flory's investigation of the gelation of polymers [2], although the term *percolation* was coined by Broadbent and Hammersley in 1957 [3] in their investigation of gas masks.

The formulation of the percolation problem is quite simple, although in general percolation is a mathematical theory [4] exactly applicable for infinite systems, either discrete or continuous. We will consider only discrete systems. A detailed but clear introduction to percolation theory can be found in [1].

Let us consider a regular lattice. It can be of any dimension: 2, 3, ..., ∞ -dimensional, and any shape: square, triangular, honeycomb, tree, etc. Let every site be independently occupied with a probability p , or not with probability $1-p$. Two or more occupied sites form a *cluster* if they are a group of neighbouring sites, where two sites are neighbours if and only if they are both occupied and they share a side. Percolation theory deals with number of properties of these clusters. At a finite value of $p = p_c$, called *percolation threshold*, a cluster large enough to span the whole system from one side to another, in the limit of infinite systems, appears. We call this cluster *spanning* or *percolating*. In the limit of finite lattices the cluster is only connecting. We call this process *site percolation*. In the infinite lattice limit the system exhibits a continuous second order phase transition at $p = p_c$, from percolating phase, where a spanning cluster runs through the entire system, for $p \geq p_c$, to a fragmented phase, where only finite clusters exist, for $p < p_c$. Many properties of a geometric phase transition associated with percolation

are similar to those of a thermodynamic phase transition [5, 6].

There is another type of percolation - *bond percolation*. In bond percolation the bonds of a regular lattice are occupied randomly with connecting elements with a certain probability p , or not with a probability $1 - p$. Two lattice sites are called connected if there exists a path between them consisting of bonds occupied by connecting elements. A set of connected elements is called *cluster*. As in the site percolation, the system exhibits phase transition at a finite value $p = p_c$.

This universality of the percolation model let us use it as a model for various problems.

1.2 Applications

Percolation is used extensively to model the phase transitions in some physical processes:

- Flow of liquid in a porous media [7] - here we observe a transition between local and extended wetting;
- Communication or resistor networks [8, 9] - disconnected/connected state transition;
- Conductor/insulator transition in composite materials [10];
- Dilute magnets - para/ferromagnetic transition;
- Polymer gelation, vulcanisation - liquid/gel transition.

It is also used outside physics to model:

- Social problems [11, 12];
- Spread of diseases in a population [13];
- Forest fires [14];
- Biological evolution [15].

A clear view of the percolation applications can be seen in the Table 1.1.

A detailed review of the percolation theory applications can be found in Ref. [16].

Phenomenon or System	Transition
Flow of liquid in a porous medium	Local/extended wetting
Spread of disease in a population	Containment/epidemic
Communication or resistor networks	Disconnected/connected
Conductor-insulator composite materials	Insulator/metal
Discontinuous metal film	Insulator/metal
Composite superconductor-metal materials	Normal/superconducting
Stochastic star formation in spiral galaxies	Non propagation/propagation
Quarks in nuclear matter	Confinement/non confinement
Thin helium films on the surfaces	Normal/superfluid
Metal-atom dispersion in insulators	Insulator/metal
Dilute magnets	Para/ferromagnetic
Polymer gelation, vulcanisation	Liquid/gel
Glass transition	Liquid/glass
Mobility edge in amorphous semiconductors	Localised/extended states
Variable-range hopping in amorphous semiconductors	Resistor-network analog

Table 1.1: Applications of the percolation theory

1.3 Exact solutions

There are few examples where the percolation threshold can be computed exactly:

- One dimension - obviously $p_c = 1$;
- Two dimensional triangular lattice - it is shown that the site percolation threshold for this type of lattices is $p_c = \frac{1}{2}$ [4];
- Bond percolation on \mathbb{L}^2 - it is proved ([4] and references therein) that the percolation threshold for bond percolation on a square lattice is simply $p_c = \frac{1}{2}$ (the interesting fact is that it took 20 years and the relentless work of three scientists to prove that theorem), $p_c = 2 \sin(\pi/18)$, for a triangular lattice, $p_c = 1 - 2 \sin(\pi/18)$, for a hexagonal lattice, and $p_c = p_c(\text{bow-tie})$, for a bow-tie lattice (see figure 1.3), where $p_c(\text{bow-tie})$ is the unique root in $(0, 1)$ of the equation

$$1 - p - 6p^2 + 6p^3 - p^5 = 0$$

(this root is $p \approx 0.40452$);

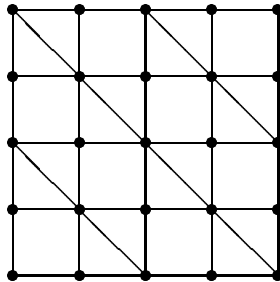


Figure 1.1: A sample 5×5 bow-tie lattice

- Bethe lattice in infinite dimensions - a Bethe lattice is an infinite tree, where every node has exactly the same degree (number of incident edges) z , $z \geq 3$. For a Bethe lattice $p_c = \frac{1}{z-1}$ [17].

In this project we will consider the simplest of percolation models: site percolation on a two dimensional square lattice with open boundaries. Although it is simple to formulate it, in fact, site percolation on a two dimensional square lattice has no exact analytic solution. Therefore, we need a proper algorithm to find if a percolating cluster exists for a given number of occupied cells.

Chapter 2

Labelling algorithms

If one only wants to know if there is a spanning cluster on a square lattice of a given size the following algorithm can be used:

1. Occupy each site with occupation probability p (one can do this easily by drawing a random number between 0 and 1 - if it is smaller than p the site is occupied). Let us write 1 in the cell if the site is occupied and 0 if it is not. Thus, one can obtain the lattice in figure 2.1:

1	0	1	0	1
1	0	1	1	1
1	1	1	1	0
0	1	0	1	1
1	1	1	0	1

Figure 2.1: Randomly generated 5×5 square lattice occupied with occupation probability $p = 0.72$.

2. Replace all the units in the first line by 5s.
3. On the second line, from the left to the right, for every site do the following:
 - If the site is occupied and one (or more) of its neighbours is a 5, write a 5 in this cell. The easiest way to do this is just to check if the sum of the adjacent cell's labels is greater than 4;
 - If the site is occupied but there is no neighbouring 5 continue with the next site;
 - if the site is not occupied continue with the next site.

Since the only case in which we do something with the site is the first one we can simply check if the sum of the labels of the neighbouring sites multiplied by the label of the current site is greater than 4;

4. Continue with the next lines repeating the steps in 3. until the bottom is reached or until there is not a new 5 in the last explored line;
5. Return to the second line and repeat steps 3. and 4.
6. Repeat step 5. until there are no new 5s in the entire lattice;
7. If a 5 occurs in the last line there is a percolating cluster, and all the cells in it are labelled with 5s. Otherwise, there is no a spanning cluster.

For the lattice we investigated (see 2.1) we label almost the entire system on the first run and get the picture in figure 2.2:

5	0	5	0	5
5	0	5	5	5
5	5	5	5	0
0	5	0	5	5
1	5	5	0	5

Figure 2.2: Picture of the analysed lattice after the first run of the algorithm

In the second run we will replace the only “1” left with a “5” and thus in no time obtain the desired result. But for larger lattices, especially near the critical point ($p = p_c$), the number of cycles we must go through would be proportional to the square of the lattice size ($L \times L$) and thus the computer time would be prohibitively large. Besides, we cannot analyse the system’s main features (see 3.1) using this algorithm.

What we would like to have is an algorithm which gives all sites within the same cluster the same label, and gives different labels to sites belonging to different clusters. Thus a spanning cluster exists if a label appears in both the top and the bottom lines, and by counting how many sites have the same label we can get the size of the cluster denoted by the specified label. A fast and efficient algorithm giving us all we want was developed in 1976 by Hoshen and Kopelman [18].

The Hoshen-Kopelman algorithm examines the lattice in the same way as you read this paper - site after site, from left to right within each line, and then from the top line to the bottom line. Let us analyse the sample lattice in 2.1 to understand this algorithm.

1		2		3
1		2	2	•
•	•	•	•	
	•		•	•
•	•	•		•

Figure 2.3: The 5×5 square lattice being analysed by the Hoshen-Kopelman algorithm.

We will give the first occupied site we meet the label “1”. The empty sites need no label (in fact, as you will see later, they will get some special label) so we continue with the next occupied site. Let us label it by a “2”, and the consequent occupied site will be labelled by a “3”. We continue with the first site on the second line (since the entire first line was labelled). It has a neighbour labelled by a “1”, therefore we label it by a “1”, too. The next occupied site receives the label “2” since its top neighbour is labelled by a “2”. The site’s right neighbour, hence, is labelled by a “2” too. Thus we obtain the lattice in the Figure2.3.

The last site in the second line is occupied also but its left and top neighbours have different labels - 2 and 3. In order to keep the label numbers as small as possible we will denote this site by a “2”. One would notice that all sites labelled by 2s and all sites labelled by 3s belong to the same cluster. The easiest way to keep this information is to relabel all the 3s into 2s. But often the easiest way is not the best way. In this case the relabelling would work perfectly on this small lattice but for large lattices the number of relabellings would be prohibitively large, like in our previous algorithm.

The clever idea of Hoshen and Kopelman was to divide the whole set of labels into *good* labels, which characterise the cluster they belong to (in fact, they are the smallest of the labels that belong to a given cluster), and *bad* labels, which in the beginning marked new clusters but later turned out to be a part of other clusters, instead of relabelling these bad clusters. To distinguish easily between good and bad labels Hoshen and Kopelman introduced an additional array N such that $N(M) = M$, if the label M is good, and $N(M) = M'$, where M' is the good label to which the bad label M turned out to be connected.

Thus, in the case of the lattice in the Figure2.1, we have two good labels $N(1) = 1$ and $N(2) = 2$ before the time we got to the current site. And now, according to the Hoshen-Kopelman algorithm, we assign the new site the label 2 and mark 3 as a bad label: $N(3) = 2$. We can continue with the third line and thus get the figure 2.4.

1	0	2	0	3	and	$N(1) = 1$ $N(2) = 1$ $N(3) = 2$
1	0	2	2	2		
1	1	1	1	0		

Figure 2.4: The first 3 lines of the analysed lattice (we denote the empty sites by 0) and the values in the “labels of the labels” array N

Sometimes we need to reclassify the bad labels in order to find the “really good” label a site is connected to, before we assign it one. This means that in our lattice we must reclassify $N(3)$ because $N(3) = 2$ and $N(2) = 1$, and thus we get $N(3) = 1$. It is practical to reclassify each of the preceding neighbours of the analysed site (the top and the left one). In fact, we need no reclassification of the left neighbour because we obtained its label in the previous step and did not have the time to change it.

Let us notice that we will mark the empty sites with the label MAX (where MAX is some large number, greater than all other labels) and thus we will simply give the site the minimal of the adjacent sites’ labels. Furthermore, if one only wants to find if there is a path from the top line to the bottom line we can simply check if one of the labels occurring in the first line appears in the last one. Last, but not least, is the fact that one must keep a single line only but not the whole lattice (because we analyse the lattice line by line), and the fact that as in all Monte Carlo simulations the lattice is created on the fly, thus saving us the time consuming reading from the hard disk.

In our program we will use a modification (see [1]) of the Hoshen-Kopelman algorithm which allows us more easily to obtain clusters’ sizes. We will denote the good labels with positive numbers, meaning the size of the cluster they belong to at this moment, and the bad labels with negative numbers, equal to minus the good label of the cluster the site belongs to. In case of a label conflict we again select as a label of the site the least of the labels of the neighbouring sites. And for N of this label we will take the sum of the values of N for the previously separated clusters, plus 1, for the site we will have just added.

Now, at last, we are ready to explore the features of the percolating system.

Chapter 3

Scaling and Renormalisation Group

3.1 Quantities of Interest in Percolation

3.1.1 Percolation probability (Cluster strength) $P(p)$

The *percolation probability* is the fraction of the entire system that is taken up by the infinite percolating cluster for a given occupation probability p :

$$P(p) = \frac{\text{number of sites in the spanning cluster}}{\text{total number of occupied sites}} \quad (3.1)$$

$P(p)$ provides the principal measure of the growth in volume of the spanning cluster as concentration increases beyond p_c . Notice that $P(p < p_c) = 0$ for infinite systems because there is no percolating cluster. The percolation probability is also known as the first moment of the cluster size distribution.

3.1.2 Cluster size distribution n_s

Cluster size distribution or *normalised cluster number* n_s we call the total number of s -clusters (clusters containing s occupied sites) divided by L^2 . The probability that an arbitrary site (occupied or not) belongs to an s -cluster is sn_s , and the probability that it belongs to any cluster is $\sum_{s=1}^{\infty} sn_s$. Hence, we have the normalized probability ω_s that an occupied site belongs to an s cluster as:

$$\omega_s = \frac{sn_s}{\sum_{s=1}^{\infty} sn_s} \quad (3.2)$$

3.1.3 Mean cluster size $S(p)$

The *mean cluster size* is the second moment of the cluster size distribution. The problem with it is what to mean by mean. The usual sense is to take it as the average cluster size over a uniform distribution. But since our case is different (we consider uniform distribution of occupied sites, not uniform distribution of clusters), we will define mean cluster size as the mean size of the cluster to which a randomly chosen occupied site belongs. Besides, it turns out that the second definition is more useful of the two. So the mean cluster size is (using the definitions in the cluster size distribution section):

$$S = \sum_{s=1}^{\infty} s\omega_s = \sum_{s=1}^{\infty} \frac{s^2 n_s}{\sum_{s=1}^{\infty} s n_s} = \frac{\sum_{s=1}^{\infty} s^2 n_s}{\sum_{s=1}^{\infty} s n_s} \quad (3.3)$$

The denominator of the last is the total number of occupied sites since every site belongs to some cluster.

3.1.4 Correlation length ξ

Let us consider the function $g(r)$ defined as the probability that a site at distance r away from an occupied site belongs to the same cluster. We call this function *correlation function* or *pair connectivity function*. For $r = 0$, the two sites are identical and $g(r) = 1$; for $r = 1$, $g(r)$ is the probability of occupation p .

Using the correlation function we can find the mean number of sites, to which an occupied site at the origin of the lattice is connected, as:

$$\sum_r g(r) \quad (3.4)$$

where the sum is over all r and hence over all sites. On the other hand, this is simply the mean cluster size S :

$$\sum_r g(r) = S = \frac{1}{p} \sum_{s=1}^{\infty} s^2 n_s \quad (3.5)$$

Using the correlation function we also can define the *correlation length* or *connectivity* of the system as:

$$\xi^2 = \frac{\sum_r r^2 g(r)}{\sum_r g(r)} \quad (3.6)$$

The correlation length is some kind of typical distance between two clusters, averaged over all sites. Stauffer [1] describes it as "the main contribution

to the second moment of the cluster size distribution near the percolation threshold.”

3.2 Scaling

It is proved that in the exactly solvable cases (see 1.3) the features mentioned above obey power-law dependences near the critical point. The generalisation of this to arbitrary lattices is the *scaling ansatz*, which cannot be proved but it can be motivated from the fractal behaviour of the percolating cluster (not to be mentioned in this project, for a detailed overview of fractals one can refer to [22]), from renormalisation group arguments and from the good agreement of this ansatz with numerical results. In fact, the scaling theory predicts that the phase transition at $p = p_c$ exhibits scaling properties of power laws [4, 1] (The following ansatz, although not in the same form, was first suggested by Fisher [19]):

$$P \propto (p - p_c)^\beta \tag{3.7}$$

$$n_s \propto s^{-\tau} \tag{3.8}$$

$$S \propto |p - p_c|^{-\gamma} \tag{3.9}$$

$$\xi \propto |p - p_c|^{-\nu} \tag{3.10}$$

where β , γ , τ and ν are called critical exponents. Unlike the percolation threshold, which varies greatly from lattice to lattice, the critical exponents are the same for all lattices of the same dimensionality D .

3.2.1 Finite-size scaling

The idea of the finite-size scaling is to extend scaling so that it also contains the system size L as a parameter, because a finite system of size L cuts off all length scales.

At the critical point we have $\xi \sim L$ and hence:

$$L \sim \xi \propto |p - p_c|^{-\nu} \tag{3.11}$$

$$|p - p_c| \propto L^{-1/\nu} \tag{3.12}$$

$$P(p; L) \simeq (p - p_c)^\beta \propto L^{-\beta/\nu} \quad (3.13)$$

$$S(p; L) \simeq |p - p_c|^{-\gamma} \propto L^{\gamma/\nu} \quad (3.14)$$

3.3 Renormalisation Group

The *renormalisation group theory* [20, 21] is linked to the scaling ansatz and the self-similarity of the percolating cluster at the critical point p_c (for more on self-similarity and fractal behaviour of the percolating cluster see [22, 23]). And, as we said in 3.2, it provides a kind of motivation for the first one.

The idea of this method is to ignore the unimportant microscopic details and to concentrate on the important physics of the large scales. We do this by replacing a $b \times b$ square with a single one, thus coarsening the picture of the lattice. The new square is occupied if there is a spanning cluster in the $b \times b$ square, or empty otherwise.

3.3.1 Square Lattice

The simplest example for renormalisation is when we choose $b = 2$ (in fact, it appears that on the triangular lattice with $b = \sqrt{3}$ the picture is simpler but we will discuss this in the next section). The important moment here is how to define a spanning cluster on the 2×2 square lattice. There are 3 possible approaches:

1. Define a cluster as percolating if it spans the system horizontally or vertically. In this case we have 2 horizontally spanning clusters with 2 occupied sites, 2 vertically spanning clusters with 2 occupied sites, 4 spanning clusters with 3 occupied sites, and one spanning cluster with four occupied sites. Therefore, the total probability $R(p)$ for appearance of a spanning cluster on the 2×2 square lattice is the sum of the probabilities for appearance of each of the possibilities, or:

$$R(p) = 4p^2(1 - p)^2 + 4p^3(1 - p) + p^4 \quad (3.15)$$

Since we want the occupation probability of the coarser lattice to be the same as the original one we must make $R(p) = p$, or:

$$4p^2(1 - p)^2 + 4p^3(1 - p) + p^4 = p \quad (3.16)$$

which has two trivial solutions $p_1 = 0$ and $p_2 = 1$, and 2 non-trivial ones: $p_{3,4} = \frac{3 \pm \sqrt{5}}{2}$. But $p \in [0; 1]$, hence, we have only one non-trivial fixed point $p_2^* = \frac{3 - \sqrt{5}}{2} \approx 0.3820$, which is far from the real value $p_c = 0.5927$ (we will denote the only non-trivial fixed point of the renormalisation group transformation for a $b \times b$ square by p_b^*). Further analysis shows us that for $p < p_2^*$ the occupation probability for the coarser lattice is less than the one of the original, and for $p > p_2^*$ it is larger;

2. Define a cluster as percolating if it spans the 2×2 square vertically. There are 2 vertically spanning clusters with 2 occupied sites, 4 spanning clusters with 3 occupied sites, and one spanning cluster with 4 occupied sites satisfying this condition. Therefore, $R(p)$ is:

$$R(p) = 2p^2(1 - p)^2 + 4p^3(1 - p) + p^4 \quad (3.17)$$

and the renormalisation group transformation gives us the two trivial fixed points: $p_1 = 0$ and $p_2 = 1$, and one non-trivial ($p \in [0; 1]$): $p_2^* = \frac{\sqrt{5}-1}{2} \approx 0.6180$, which is quite close to the real occupation probability $p_c = 0.5927$. Again for $p < p_2^*$ the renormalisation group decreases the occupation probability, and for $p > p_2^*$ it increases it.

Absolutely the same result would be achieved if we had defined a percolating cluster on the 2×2 square lattice as a cluster spanning it horizontally;

3. Define a cluster a percolating if it spans the 2×2 square lattice both horizontally and vertically. There are 5 clusters satisfying these conditions: 4 spanning clusters with 3 occupied sites and one spanning cluster with 4 occupied sites. Here $R(p)$ is equal to:

$$R(p) = 4p^3(1 - p) + p^4 \quad (3.18)$$

and the renormalisation group transformation gives us again the two trivial fixed points $p_1 = 0$ and $p_2 = 1$, and one non-trivial: $p_2^* = \frac{\sqrt{13}+1}{6} \approx 0.7676$, which, like our first case, is far from p_c . The influence of the renormalisation group on the occupation probability is the same as the one in the previous cases.

3.3.2 The Triangular Lattice

In this case we replace 3 sites by one and thus $b^2 = 3$. The renormalisation group transformation we obtain is now:

$$p \leftarrow R(p) = 3p^2(1 - p) + p^3 \quad (3.19)$$

with fixed points 0, $\frac{1}{2}$, and 1. The non-trivial solution here $p_3^* = \frac{1}{2}$ is exact ($p_c = p_3^*$) and thus the renormalisation group does not change the occupation probability for the lattice. This allows us to use this method as a means of finding if the triangular lattice system percolates, or not.

Let us only notice that if we want more precise results for the features of the system after the renormalisation group transformation we must use larger b .

Chapter 4

Results

In our research we explored two of the four main features of the percolating system - the cluster-size distribution and the mean cluster size. Apart from this we investigated a feature characteristic of the type of the system (its shape and size) - its wrapping probability, or the probability that a percolating cluster appears in the system. In all calculations we made we observed site percolation on a two-dimensional square lattice with open boundaries. We say that the system percolates if a cluster spans the lattice vertically (one good reason for making this assumption is the renormalisation group theory which shows (see section 3.3.1) that near the critical point ($p \approx p_c$) occupation probability keeps its value after renormalisation with highest accuracy if we defined a cluster as percolating in that way).

There are some common effects which implement systematic error in the forthcoming results.

4.1 Common Errors

The main factors which contribute errors in our results are two - finite-size effects and statistical fluctuations. The only fact that we can simulate only finite systems but we try to obtain the properties of the infinite ones sheds first light on the reasons because of which we talk of finite-size effects as systematic errors. In general the finite systems can exhibit rather different behaviour than the infinite ones (for example, in a 2 x 2 square lattice the probability for percolation on the first column is $p^2(1 - p)^2 + 2p^3(1 - p) + p^4$ (number significantly greater than zero) but for an infinite lattice this probability is exactly zero. For more on the differences between finite and infinite systems see [24, 25]). However, we are not interested in exploring these curious properties but we want only to mark the main tendencies in the

investigated features' behaviour and thus get some properties of the infinite systems and wherever we can - to give some exact results for them.

Unfortunately, as we said, there is one more contributor of systematic error - the statistical fluctuations. Since they are inversely proportional to L^d (where d is the dimensionality of the system) for the small systems we observe that the deviations would be significant. Therefore, to get each point in the forthcoming plots we made averaging of the results from the maximum possible iterations allowed by the system size L and the length of the random number sequence provided by the pseudo-random number generator we use (which is the linear-congruential `ibm=ibm*16807` with random-number sequence of approximately 500 million different numbers).

4.2 Cluster-Size Distribution

It is evident that for $p \ll p_c$ all clusters are small and isolated and thus n_s has contribution only from small sizes. When p increases clusters merge and start to grow adding the contribution of greater sizes. At $p = p_c$ a random thin web forming the percolating cluster appears and covers the whole space. Finite clusters are now placed in the pores of this cluster. Now there are contributions from all cluster sizes to the cluster size distribution. When p continues to grow the spanning cluster starts to absorb the finite ones making the contribution from the finite clusters smaller.

Since the only interesting case is when $p = p_c$ (because only then we have contributions from all cluster sizes) we will investigate only him. The known value for p_c for a square lattice is $p_c = 0.592746$ (see [26]). Let us notice that in fact we used $p = 0.593046$, which is close enough to p_c , because the system did not percolate for smaller p for the seed number we used (the seed was 5).

We expect n_s (see section 3.1.2) to follow the power law $n_s \propto s^{-\tau}$ (see section 3.2) where the critical exponent τ is known exactly (see [27, 28]): $\tau = 187/91$. Thus in a double logarithmic plot the cluster-size distribution would be presented by a straight line with a slope $-\tau$. In order to reduce the deviations caused by finite-size effects and, to a smaller degree, by statistical fluctuations we do not use averaging of numerous iterations but use a single run on the maximum lattice with a size less than the uncorrelated sequence generated by the pseudo-random number generator we use. This is the lattice 22357×22357 . To make handling of Monte Carlo data easier we do not store each n_s for each s but, instead, we gather this data in bins. The first bin stores n_1 , the second: $n_2 + n_3$, the third: $n_4 + n_5 + n_6 + n_7, \dots$, the k th bin stores the consequent sum of 2^{k-1} numbers. Thus, instead of having an array

of almost 10 million numbers, most of which are zeros, we have a sequence of only 24 meaning numbers. The analysis of the binned data (for $s \geq 2$, for $s = 1$ we have $n'_1 = n_1$) is easy:

$$\begin{aligned}
n'_s &= \sum_{s'=2^{s-1}}^{2^{2(s-1)}-1} n_{s'} \propto \sum_{s'=2^{s-1}}^{2^{2(s-1)}-1} (s')^{-\tau} \simeq \int_{s'=2^{s-1}}^{2^{2(s-1)}-1} (s')^{-\tau} d(s') \\
&= \frac{(s')^{-\tau+1}}{-\tau+1} \Big|_{2^{s-1}}^{2^{2(s-1)}-1} = \frac{(2^{s-1})^{-\tau+1} - (2^{2(s-1)} - 1)^{-\tau+1}}{\tau - 1} \\
&\propto \frac{1}{(2^{s-1})^{\tau-1}} - \frac{1}{(2^{2(s-1)} - 1)^{\tau-1}} \tag{4.1}
\end{aligned}$$

and since $\tau - 1 \approx 1$ for $s \geq 5$ we can neglect the subtrahend in the last expression (it is less than 1 per cent of the minuend). Therefore:

$$n'_s \propto 2^{(s-1)(-\tau+1)} = 2^{s(-\tau+1)} 2^{\tau-1} \propto 2^{s(-\tau+1)} \tag{4.2}$$

Since we use an exponential function fit we must change the dependence in the following way:

$$n'_s \propto 2^{s(-\tau+1)} = e^{\ln 2(-\tau+1)s} \tag{4.3}$$

Besides, with a logarithmic Y axis we would have a straight line for n'_s with slope $\lg 2(-\tau + 1)$. Thus, we obtain the plot in the figure 4.1. In it we made the above mentioned fit for the region of the data that is not influenced by the two effects: corrections to scaling, for small s , and finite-size effects, for large s (see the consequent two sections). Apart from this the first point from the fit region is the fifth since the approximation we made in expression 4.2 is valid for $s \geq 5$. Thus we obtain the fit:

$$n'_s = 1.7749 \cdot 10^7 e^{-0.73846s} \tag{4.4}$$

Therefore $\ln 2(-\tau + 1) = -0.73846$ and, hence, $\tau = 2.06(5)$ which is in excellent agreement with the theoretical result for $L \rightarrow \infty$: $\tau = 187/91 \approx 2.055$.

4.2.1 Corrections to Scaling

The lack of inherent lengths in the percolating system is very important for the power law $n_s \propto s^{-\tau}$ to be fulfilled since every length scale after the renormalisation would change and thus the self-similarity, characteristic of the percolating system, would not be valid. However, there are two inherent lengths in our system - the first is the finite size of the system, and the second

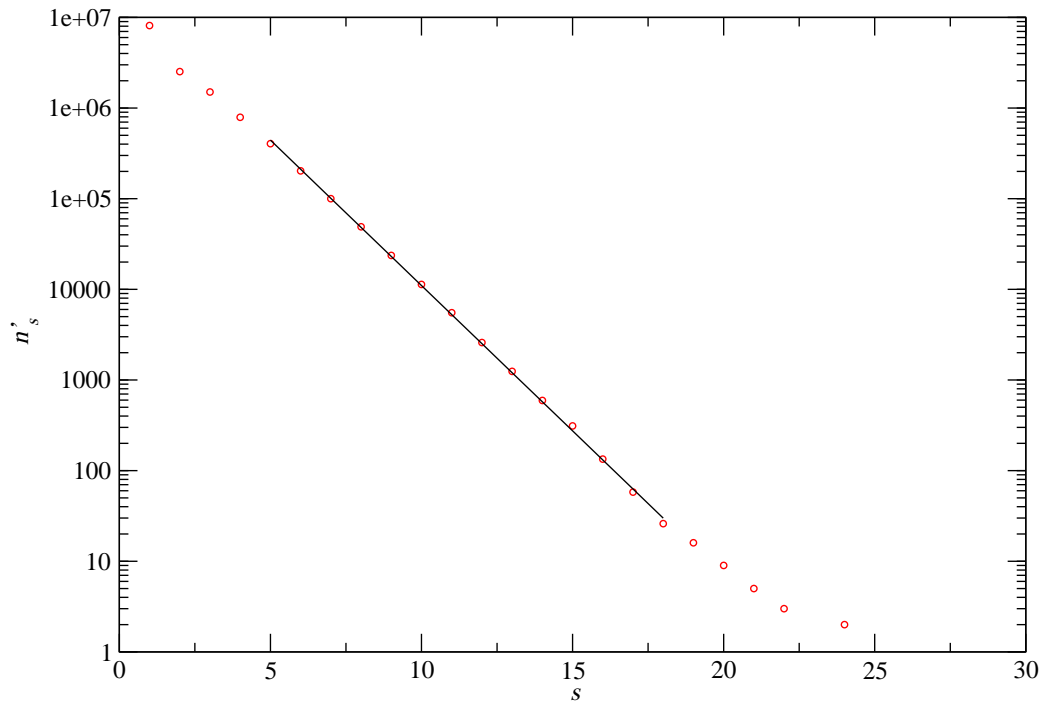


Figure 4.1: Cluster-size distribution in the 22357×22357 square lattice with occupation probability $p = 0.593046$ using exponential binning and a logarithmic Y axis, where n'_s and s are defined in expression 4.1.

is the lattice spacing a (in our case of finite square lattice a is obviously equal to 1). We will discuss the finite-size effect in the next section. The lattice spacing a influences greatly the small clusters' distribution. The reason is that when we renormalise the system the small clusters (of size $s \approx a$) would vanish. Therefore these clusters do not follow the general law of self-similarity near the critical point and hence should deviate from the power law line (as it can be seen clearly in figure 4.1). The expected behaviour for small clusters is (according to [29]):

$$n_s \propto s^{-\tau}(1 - k_1 s^{-\Delta_1}) \quad (4.5)$$

The term $(1 - k_1 s^{-\Delta_1})$, where k_1 and Δ_1 are parameters which can be found by computer simulations, is called corrections to scaling (see [30] for more on this topic).

4.2.2 Influence of Boundary Conditions

As we said, the size of the lattice is an important inherent length. It is easy to understand its influence on the power law $n_s \propto s^{-\tau}$ for systems with open boundaries (namely, the systems we explore).

When a large cluster is placed near a boundary a part of it is cut off and although it would extend beyond we stop the counting of sites and thus get more but smaller clusters. Therefore, we expect a deviation of the cluster-size distribution above the power law. From the plot (the figure 4.1) one can see that these deviations are much more significant for large s . That is true because small clusters in the interior of the lattice do not feel the boundaries and, besides, the probability that a small cluster is near the boundaries is negligible. That is not the case of large clusters, especially of those of linear size close to L . For a large cluster there are not so many places in the interior of the lattice, hence, it is very probable that it is cut off and decreased significantly in size.

Let us notice that in fact the open boundaries implement free surfaces in the system. Since clusters can be placed anywhere in the volume of the lattice but they would feel the finite-size effects if they touch a free surface, the effects are proportional to the surface divided by the volume of the system, namely proportional to $1/L$. If one wants to avoid this deviation one must avoid free surfaces using either fully periodic or helical boundary conditions. However, we cannot avoid completely the finite-size effects because in a system of size L^d (where d is the dimensionality of the system) there cannot occur a cluster of size greater than L^d . And therefore, these effects are proportional to $\frac{1}{L^d}$.

We investigated only systems with open boundaries since the results we got are good enough for the purpose of our research and the implementation of the Hoshen-Kopelman algorithm is easier for open boundaries.

The last thing we would like to notice for finite-size effects is that the aspect ratio of the investigated system (its width divided by its height, for a two-dimensional system) also impacts the results. One can see [31, 32] for more.

4.3 Wrapping Probability

For a specific occupation probability p we use number of runs of the Hoshen-Kopelman algorithm (from 500 to 50000, depending on the lattice's size) and mark how many times the system did percolate. We call the quotient of this number to the total number of runs *wrapping probability of the system* $W(p)$. It is clear that the wrapping probability is the probability for appearance of a spanning cluster. Conducting the upper calculations we obtained the results presented in figure 4.2.

From the plot one can see that the statistical fluctuations for all the lattices were suppressed by the averaging of multiple runs (500 and over for each point in different lattices).

Despite the statistical error the behaviour of $W(p)$ is quite clear from the plot. We observe a monotonous increase of the function which grows with the growth of L . This is in complete agreement with the expected behaviour of the infinite lattice:

- for $p < p_c$** all clusters are finite and system does not percolate
- for $p > p_c$** an infinite percolating cluster exists

Namely, $W(p)$ is a θ -function in the infinite lattice limit:

$$W(p) = \begin{cases} 0 & \text{if } p < p_c \\ 1 & \text{if } p > p_c \end{cases} \quad (4.6)$$

Let us notice that in the plot we use the fit:

$$f(p) = \frac{p^\alpha}{p_c^\alpha + p^\alpha} \quad (4.7)$$

where α is a parameter and p_c is the percolation threshold for the specific lattice. Obviously, it does not exhibit the real law but it shows quite well the tendencies of the function and being symmetric let us notice that the real distribution $W(p)$ is not symmetric and in the critical region the slope for $p > p_c$ is greater than the one for $p < p_c$.

The results for the mean cluster size are quite good, too.

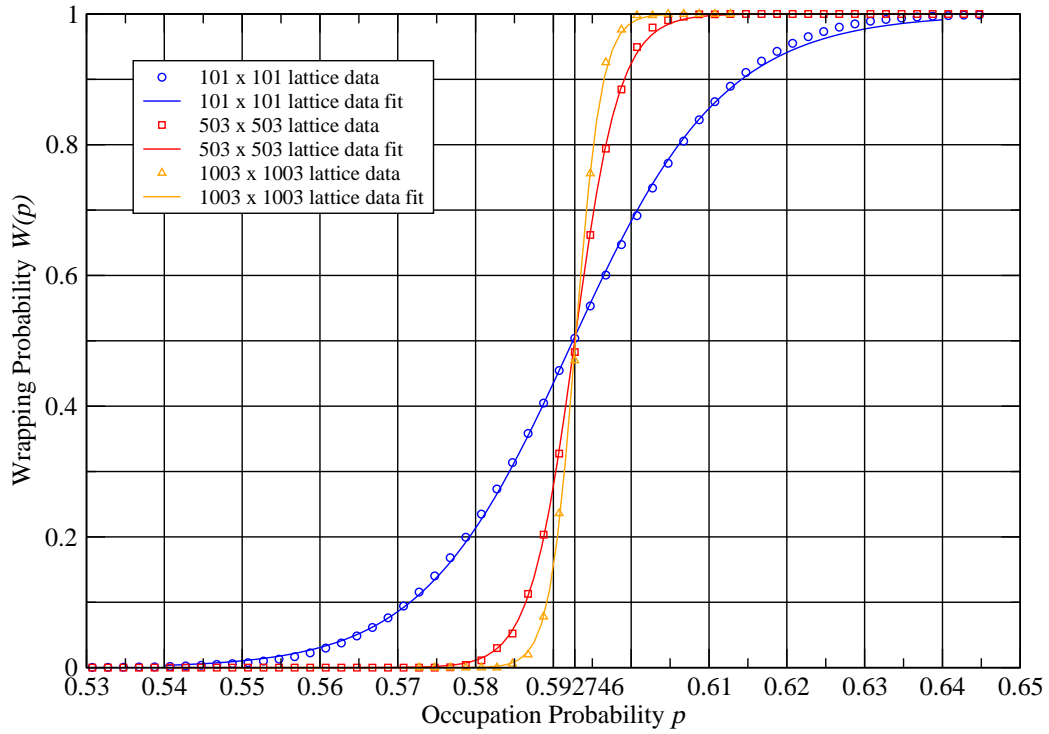


Figure 4.2: Wrapping probability for different system sizes. The results for the 101×101 square lattice are obtained after averaging of 50000 iterations, for the 503×503 lattice - with 1976 iterations, and for the 1003×1003 - with 500 iterations.

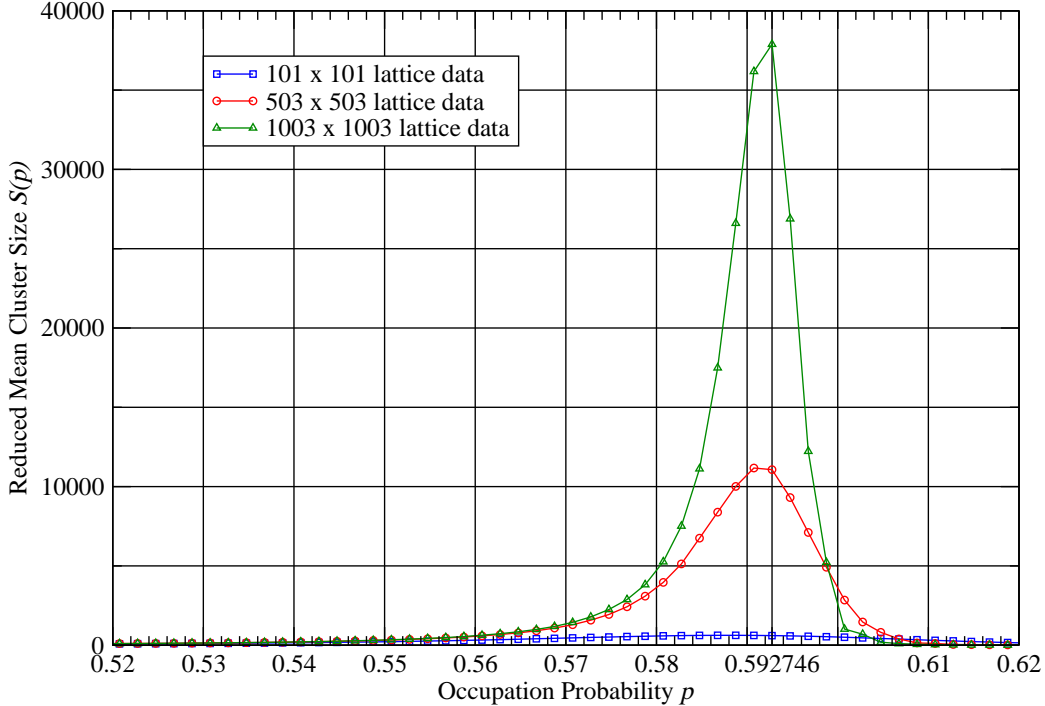


Figure 4.3: Dependence of the mean cluster size from the occupation probability. The results for each point in the 101×101 lattice curve are obtained after averaging 50000 iterations, for the 503×503 lattice curve - after averaging 1976 iterations, and for the 1003×1003 lattice curve - after averaging 500 iterations.

4.4 Mean Cluster Size

Let us first notice that we have explored the reduced mean cluster size, i.e. the mean size as defined in expression (3.3) but upon all clusters except the spanning one (since in the infinite lattice limit the percolating cluster gives infinite contribution to $S(p)$). We have investigated only three lattice sizes with the idea only to figure out the main tendencies in the mean cluster size distribution. The obtained results are presented in the graph in figure 4.3. For the 101×101 lattice we made 50000 runs for each point, for the 503×503 lattice the runs were 1976, and for the 1003×1003 lattice the runs were 500. That is why one can hardly see the statistical fluctuations.

When $p < p_c$ we have only finite clusters which grow in size with the increase of p . At $p = p_c$, there is a peak in $S(p)$ (in fact, the peak is observed just a moment before we reach p_c because when p reaches p_c there is a sudden drop in $S(p)$ caused by the subtraction of the main contribution to the mean

cluster size - the percolating cluster). For $p > p_c$, $S(p)$ decreases since the spanning cluster starts to absorb the larger clusters thus diminishing the reduced mean cluster size. It is worth noticing that this slope is steeper than the one when p inclines to p_c from below. Another result is that when L grows the slope of $S(p)$ near criticality increases. Thus, in the infinite lattice limit, for $p \rightarrow p_c$ we have that $S(p) \rightarrow \infty$. One can see that the site percolation threshold for the finite square lattice 1003×1003 is equal to the known value for $L \rightarrow \infty$: $p_c = 0.592746$ with high accuracy (the discovery of the exact percolation threshold for the lattice 1003×1003 and for $L \rightarrow \infty$ is beyond the scope of our project).

Acknowledgements

First and foremost I would like to thank Prof. Ana Proykova for starting work with me. I would like to thank for her attention and patience with me. Second I would like to thank Prof. Dietrich Stauffer for his valuable advices and for teaching me what he called “scientific computation”. I would like to thank also Hristo Iliev, Stoyan Pisov and Rossen Radev without whose technical help I would not be able to finish this project.

Appendix A

Programme Layout

In this chapter we will introduce you the source of the programmes we created and used in this project. Because of the principle differences between the investigation of n_s , and the investigation of $S(p)$ and $W(p)$ (for n_s we use a single run, while for $S(p)$ and $W(p)$ we use multiple runs) we use different programs for n_s , and for $S(p)$ and $W(p)$. However, the core of the two programmes is identical - the Hoshen-Kopelman algorithm.

First we will present you the source code of the programme for n_s .

```
#include <stdio.h>

#define L    22357
#define p    0.593046
#define MAX  L*L/8
#define MAX2 MAX*2

unsigned int line, mleft, mtop, mnew, level[L+1], numbers1;
unsigned int histogram[25];
int ms, hlp, n[MAX+1];
int nopath;

int main(void)
{
    int index, INDEX, i, k, ip, ibm;
    double log2=1.000001/log(2.0);

    ibm=5;

    printf("L=%d, MAX=%d, p=%f, ibm=%d\n", L, MAX, p, ibm);
```

```

for (i=0; i<=L; i++)
    level[i]=MAX;

n[MAX]=MAX;

index=0;
ip=(int)((2.0*p-1.0)*2147483647.0);

fprintf(stderr, "ns.dat\n");

for (i=1; i<=L;i++)
{
    ibm=ibm*16807;
    if (ibm>ip) goto nonumber1;

    mleft=level[i-1];
    mtop=level[i];
    if ((mleft+mtop)==MAX2) goto newclust1;
    ms=n[mtop];
    if (ms>0) goto noreclas1;

    reclass1:
        hlp=ms;
        ms=n[-ms];
        if (ms<0) goto reclass1;
        n[mtop]=hlp;
        mtop=-hlp;

    noreclas1:
        mnew=(mtop<mleft)?mtop:mleft;
        level[i]=mnew;
        if ((mtop<MAX)&&(mtop!=mnew))
        {
            n[mleft]+=n[mtop];
            n[mtop]=-mnew;
        }
        if ((mleft<MAX)&&(mleft!=mnew))
        {
            n[mtop]+=n[mleft];
            n[mleft]=-mnew;
        }
}

```

```

    }
    n[mnew]++;
    continue;

newclust1:
    index++;
    n[index]=1;
    level[i]=index;
    continue;

nonumber1:
    level[i]=MAX;
}

INDEX=index;

for (k=2; k<=L; k++)
{
    line=MAX;

    for (i=1; i<=L;i++)
    {
        ibm=ibm*16807;
        if (ibm>ip) goto nonumber;

        mleft=level[i-1];
        mtop=level[i];
        if ((mleft+mtop)==MAX2) goto newclust;
        ms=n[mtop];
        if (ms>=0) goto noreclas;

reclas:
        hlp=ms;
        ms=n[-ms];
        if (ms<0) goto reclass;
        n[mtop]=hlp;
        mtop=-hlp;

noreclas:
        mnew=(mtop<mleft)?mtop:mleft;

```

```

    level[i]=mnew;
    if ((mtop<MAX)&&(mtop!=mnew))
    {
        n[mleft]+=n[mtop];
        n[mtop]=-mnew;
    }
    if ((mleft<MAX)&&(mleft!=mnew))
    {
        n[mtop]+=n[mleft];
        n[mleft]=-mnew;
    }
    n[mnew]++;
    goto end;

    newclust:
        index++;
        n[index]=1;
        level[i]=index;
        goto end;

    nonumber:
        level[i]=MAX;

    end:
        line=(line<level[i])?line:level[i];
}

if (line>(unsigned int)INDEX&&!(nopath))
{
    nopath=k;
}
}

hlp=0;
for (i=1; i<=index; i++)
{
    if (n[i]>0)
    {
        histogram[(int)(log((double)n[i])*log2)]++;
        hlp=(hlp>n[i])?hlp:n[i];
        numbers1+=n[i];
    }
}

```

```

    }
}

printf("Index = %d\n", index);

printf("Max cluster size = %d\n", hlp);

printf("Real concentration: %f", (float)numbers1/(float)(L*L));

if (nopath)
{
    printf("\nThere is no path! - Line %d\n", nopath);
}

FILE *out;
out = fopen("ns.dat", "wt");

for (i=1;i<=25;i++)
{
    fprintf(out, "%d %d\n", i, histogram[i]);
}

fclose(out);

return 1;
}

```

Following is the source code for the $S(p) - W(p)$ programme.

```
#include <stdio.h>
#include <stdlib.h>

#define L 503
#define L2 L*L
#define MAX L2/6
#define MAX2 MAX*2
#define NRUN 1976
#define NCON 100

unsigned int line, mleft, mtop, mnew, level[L+1];
int ms, hlp, hlp1, n[MAX+1];
int nopath;

int main(void)
{
    int index, INDEX, i, k, ip, ibm, iter, icount, icon;
    double p, S;
    double Sp[3][NCON+1], Wp[3][NCON+1];
    FILE *out1, *out2;

    p=0.502746;
    ibm=5;

    printf("L=%d, MAX=%d, ibm=%d\n", L, MAX, ibm);

    for (icon=1; icon <= NCON; icon++)
    {
        S=0.0;
        icount=NRUN;
        ibm=5;
        p+=0.002;

        ip=(int)((2.0*p-1.0)*2147483647.0);

        for (iter=1; iter <= NRUN; iter++)
        {
            for (i=0; i<=L; i++)
                level[i]=MAX;
        }
    }
}
```

```

n[MAX]=MAX;

index=0;

for (i=1; i<=L;i++)
{
  nopath=0;
  ibm=ibm*16807;
  if (ibm>ip) goto nonumber1;

  mleft=level[i-1];
  mtop=level[i];
  if ((mleft+mtop)==MAX2) goto newclust1;
  ms=n[mtop];
  if (ms>0) goto noreclas1;

  reclass1:
  hlp=ms;
  ms=n[-ms];
  if (ms<0) goto reclass1;

  n[mtop]=hlp;
  mtop=-hlp;

  noreclas1:
  mnew=(mtop<mleft)?mtop:mleft;
  level[i]=mnew;

  if ((mtop<MAX)&&(mtop!=mnew))
  {
    n[mleft]+=n[mtop];
    n[mtop]=-mnew;
  }
  if ((mleft<MAX)&&(mleft!=mnew))
  {
    n[mtop]+=n[mleft];
    n[mleft]=-mnew;
  }
  n[mnew]++;
  continue;
}

```

```

newclust1:
  index++;
  n[index]=1;
  level[i]=index;
  continue;

  nonumber1:
    level[i]=MAX;
  }

INDEX=index;

for (k=2; k<=L; k++)
{
  line=MAX;

for (i=1; i<=L;i++)
{
  ibm=ibm*16807;
  if (ibm>ip) goto nonumber;

mleft=level[i-1];
  mtop=level[i];
  if ((mleft+mtop)==MAX2) goto newclust;
  ms=n[mtop];
  if (ms>=0) goto noreclas;

  reclass:
hlp=ms;
  ms=n[-ms];
  if (ms<0) goto reclass;

n[mtop]=hlp;
  mtop=-hlp;

  noreclas:
  mnew=(mtop<mleft)?mtop:mleft;
  level[i]=mnew;
  if ((mtop<MAX)&&(mtop!=mnew))
  {

```

```

        n[mleft]+=n[mtop];
        n[mtop]=-mnew;
    }
    if ((mleft<MAX)&&(mleft!=mnew))
    {
        n[mtop]+=n[mleft];
        n[mleft]=-mnew;
    }
    n[mnew]++;
    goto end;

newclust:
    index++;
    n[index]=1;
    level[i]=index;
    goto end;

nonumber:
    level[i]=MAX;

end:
    line=(line<level[i])?line:level[i];
}

if (line>(unsigned int)INDEX&&!(nopath))
{
    nopath=k;
}
}

hlp=0;
for (i=1; i<=index; i++)
{
    if (n[i]>0)
    {
        hlp=(hlp>n[i])?hlp:n[i];
    }
}
S+=((double)n[i])*((double)n[i]);

if (nopath)

```

```

{
icount--;
}
else
{
S-=((double)hlp)*((double)hlp);
}

}

Sp[0][icon-1] = p;
Sp[1][icon-1] = S/(double)(L2)/(double)(NRUN*p);
Wp[0][icon-1] = p;
Wp[1][icon-1] = (double)icount/(double)NRUN;

}

out1 = fopen("Wp.dat", "wt");
out2 = fopen("Sp.dat", "wt");

for (i=0; i<=NCON-1; i++)
{
fprintf(out1, "%f %f\n", Wp[0][i], Wp[1][i]);
fprintf(out2, "%f %f\n", Sp[0][i], Sp[1][i]);
}

fclose(out1);
fclose(out2);

return 1;
}

```

References

- [1] D. Stauffer and A. Aharony, Introduction to Percolation Theory, Taylor&Francis, London, 2nd edition (1992)
- [2] P. J. Flory, Molecular Size Distribution in Three Dimensional Polymers. I. Gelation, pp. 3083, II. Trifunctional Branching Units, pp. 3091, III. Tetrafunctional Branching Units, pp. 3096. *J. Am. Chem. Soc.* **63** (1941)
- [3] S. R. Broadbent and J. M. Hammersley, Percolation Processes. Crystals and Mazes, *Proc. Camb. Phil. Soc.* **53**, pp. 629 (1957)
- [4] G. Grimmett, Percolation, Springer-Verlag, New York (1989)
- [5] <http://gene.wins.uva.nl/~vdmeer/nni/percolation/docs/Percolation.htm>
- [6] <http://people.nnov.ru/fractal/perc/distr.htm>
- [7] R. G. Larson, L. E. Scriven, and H. T. Davis, Percolation Theory of Two-Phase Flow in Porous Media, *Chem. Eng. Sci.* **15**, pp. 57 (1981)
- [8] R. Cohen, K. Erez, D. ben-Avraham, and S. Havlin, Resilience of the Internet to Random Breakdowns, *Phys. Rev. Lett.* **85**, pp. 4626 (2000)
- [9] L. A. Arcangelis, S. Redner, and A. Cognilio, Anomalous Voltage Distribution of Random Resistor Networks and a New Model for the Backbone of the Percolation Threshold, *Phys. Rev. B* **31**, pp. 4725 (1985)
- [10] R. B. Laibowitz, R. F. Voss and E. I. Alessandrini, Clustering in Thin Au Films Near the Percolation Threshold, in: Percolation, Localisation and Superconductivity, Proc. of NATO Advanced Study Institute, Ser. B: Physics **109**, ed. by A. M. Goldman and S. A. Wolf, Plenum Press, New York, pp. 145 (1984)

- [11] A. Proykova and D. Stauffer, Social Percolation Mass Media Influence, *Physica A* **312/1-2**, pp. 300 (2002)
- [12] S. Solomon, G. Weisbuch, L. A. Arcangelis, N. Jan, and D. Stauffer, Social Percolation Models, *Physica A* **277**, pp. 239 (2000)
- [13] M. E. Newman and D. J. Watts, Scaling and Percolation in the Small-World Networks, *Phys. Rev. E* **60**, pp. 7332 (1999)
- [14] C. L. Henley, Statics of Self-Organised Percolation Model, *Phys. Rev. Lett.* **71**, pp. 2741 (1993)
- [15] T. S. Ray and N. Jan, Anomalous Approach to the Self-Organised Critical State in a Model for Life at the Edge of Chaos, *Phys. Rev. Lett.* **72**, pp. 4045 (1994)
- [16] M. Sahimi, Applications of Percolation Theory, Taylor&Francis, London (1994)
- [17] <http://www.tcm.phy.cam.ac.uk/~ym101/lecture/14a.ps>
- [18] J. Hoshen and R. Kopelman, Percolation and Cluster Distribution. I. Cluster Multiple Labeling Technique and Critical Concentration Algorithm, *Phys. Rev. B* **14**, pp. 3438 (1976)
- [19] M. E. Fisher, The Theory of Condensation and the Critical Point, *Physics* **3**, pp. 255 (1967)
- [20] M E. Fisher, Renormalisation Group Theory: Its Basis and Formulation in Statistical Physics, *Rev. Mod. Phys.* **70**, pp. 653 (1998)
- [21] P. J. Reynolds, H. E. Stanley, and W. Klein, Large Cell Renormalisation Group for Percolation, *Phys. Rev. B* **21**, pp. 1223 (1980)
- [22] B. B. Mandelbrot, The Fractal Geometry of Nature, W. H. Freeman, New York (1983)
- [23] A. Bunde, S. Havlin (Eds.), Fractals and Disordered Systems, Springer, Heidelberg (1991)
- [24] K. Binder, Statistical Mechanics of Finite Ising Models, *Physica* **62**, pp. 508 (1972)
- [25] A. Proykova and R. S. Berry, Analogues in Clusters of Second-Order Phase Transitions?, *Z. Phys. D* **40**, pp. 215 (1997)

- [26] H. E. Stanley, Introduction to Phase Transitions and Critical Phenomena, Oxford University Press, Oxford (1971)
- [27] B. Nienhuis, E. K. Riedel, and M. Shick, Magnetic Exponents of the Two-Dimensional q-state Potts Model, *J. Phys. A* **13**, pp. 189 (1980)
- [28] B. Nienhuis, Analytical Solution of the Two Leading Exponents of the Dilute Potts Model, *J. Phys. A* **15**, pp. 199 (1982)
- [29] J. Adler, M. Moshe, and V. Privman, Corrections to Scaling for Percolation, in: Annals of the Israel Physical Society **5**, Adam Hilger, Bristol, pp. 397 (1983)
- [30] A. Aharony and M. E. Fisher, Non-Linear Scaling Fields and Corrections to Scaling Near Criticality, *Phys. Rev. B* **27**, pp. 4394 (1983)
- [31] A. Aharony and D. Stauffer, Test of Universal Finite-Size Scaling in Two-Dimensional Site Percolation, *J. Phys. A* **30**, L301 (1997)
- [32] R. M. Ziff, C. D. Lorenz and P. Kleban, Shape-Dependent Universality in Percolation, in: A. Bunde and S. Havlin (Eds.), Proceedings of the International Conference on Percolation and Disordered Systems: Theory and Applications, *Physica A* **266**, pp. 17 (1999)