Complex Systems
WROCŁAW UNIVERSITY OF SCIENCE AND TECHNOLOGY
Percolation model

Adam Kozakowski 255535

January 3, 2024

Files containing Python scripts and results of analysies are uploaded under Dropbox link:

https://www.dropbox.com/scl/fo/eqx2mm8xcdlri7r1q063p/h?rlkey=z06wg5uzmzp9fa5nzw3dklsxm&dl=0

1 Percolation model

Percolation model is theoretical model illustrating movement or filtering of fluids through porous media. It simulates square lattice such that every cell can either be occupied or empty with probablility p. For systems with high enougth p (critical probability p_c) cration of spanning (percolating) cluster can happen (cluster joining two opposite sides of the box). Values of p_c can vary among types of lattices (for square lattice $p_c = 0.59264$

Percolation theory is a concept used to describe complex systems that ... In systems perlocation can happen - .... System is in a critical state when it is on the edge of a phase transition - the system is at a balance point between order and disorder, and small changes can lead to qualitative changes of system.

2 Algorythms for analysing percolation model

2.1 The Burning Method

Used for providing information about lattice configuration and detecting whether a spanning cluster exists. The name of the method stems from its implementation. An occupied site represents a tree while an unoccupied site stands for empty space. Starting a fire at the very top of our grid, all trees in the first row will start to light up as they are in the immediate vicinity of the fire.

The algorithm is as follows:

1. Label all occupied cells in the top line with the marker t = 2.
2. Iteration step t + 1
    (a) Go through all the cells and find the cells which have label t.
    (b) For each of the found t-label cells do
        i. Check if any direct neighbor (North, East, South, West) is occupied and not burning (label is 1).
        ii. Set the found neighbors to label $t + 1$.
3. Repeat step 2 (with t = t + 1) until either there are no neighbors to burn anymore or the bottom line has been reached - in the latter case the latest label minus 1 defines the shortest path.

2.2 Hoshen-Kopelman (HK) algorithm

Hoshen-Kopelman algorithm is used to identify all the different clusters and to study how clusters are distributed. We start the algorithm by setting label of cluster k in lattice N_{ij} to $k = 2$ (since 0 and 1 are already taken) and searching for the first occupied site in N_{ij}. We then add this site to the array of masses $M_{k=2} = 1$ and set the entry in N_{ij} to k (so it is branded as pertaining to the cluster k).

We then start looping over all lattice sites N_{ij} and try to detect whether an occupied site belongs to an already known cluster or a new one. We comb through the lattice from top-left to bottom-right:

- If a site is occupied and the top and left neighbors are empty, we have found a new cluster and we set k to $k+1, N_{ij} = k$ and $M_k = 1$. We continue the loop.
- If one of the sites (top or left) has the value k_0 (i.e. has already been absorbed into a cluster), we increase the value of the corresponding array M_{k_0} by one (setting M_{k_0} to $M_{k_0}+1$). We name the new site accordingly, i.e. $N_{ij} = k_0$.
- If both neighboring sites are occupied with k_1 and k_2 respectively (assuming $k_1 \neq k_2$)- meaning that they are already part of a cluster - we choose one of them (e.g. k_1). We set the matrix entry to the chosen value, $N_{ij} \leftarrow k_1$, and increase the array value not only by one but also by the whole number of sites already in the second cluster (in the example k_2), $M_{k_1} \leftarrow M_{k_1} + M_{k_2} + 1$. Of course we have to mark the second array M_{k_2} in some way so that we know that its cluster size has been transferred over to M_{k_1} which we do by setting it to $-k_1$. We have thus branded M_{k_2} in such a way that we immediately recognize that it does not serve as a counter anymore (as a cluster cannot consist of a negative number of sites). Furthermore, should we encounter an occupied site neighboring a k_2 site, we can have a look at M_{k_2} to see that we are actually dealing with cluster k_1 (revealing the "true" cluster number).

3 Visualization of sample configurations for L = 10 and 3 values of p = 0.4, 0.6, 0.8 within two methods: Burning method and HK algorythm.

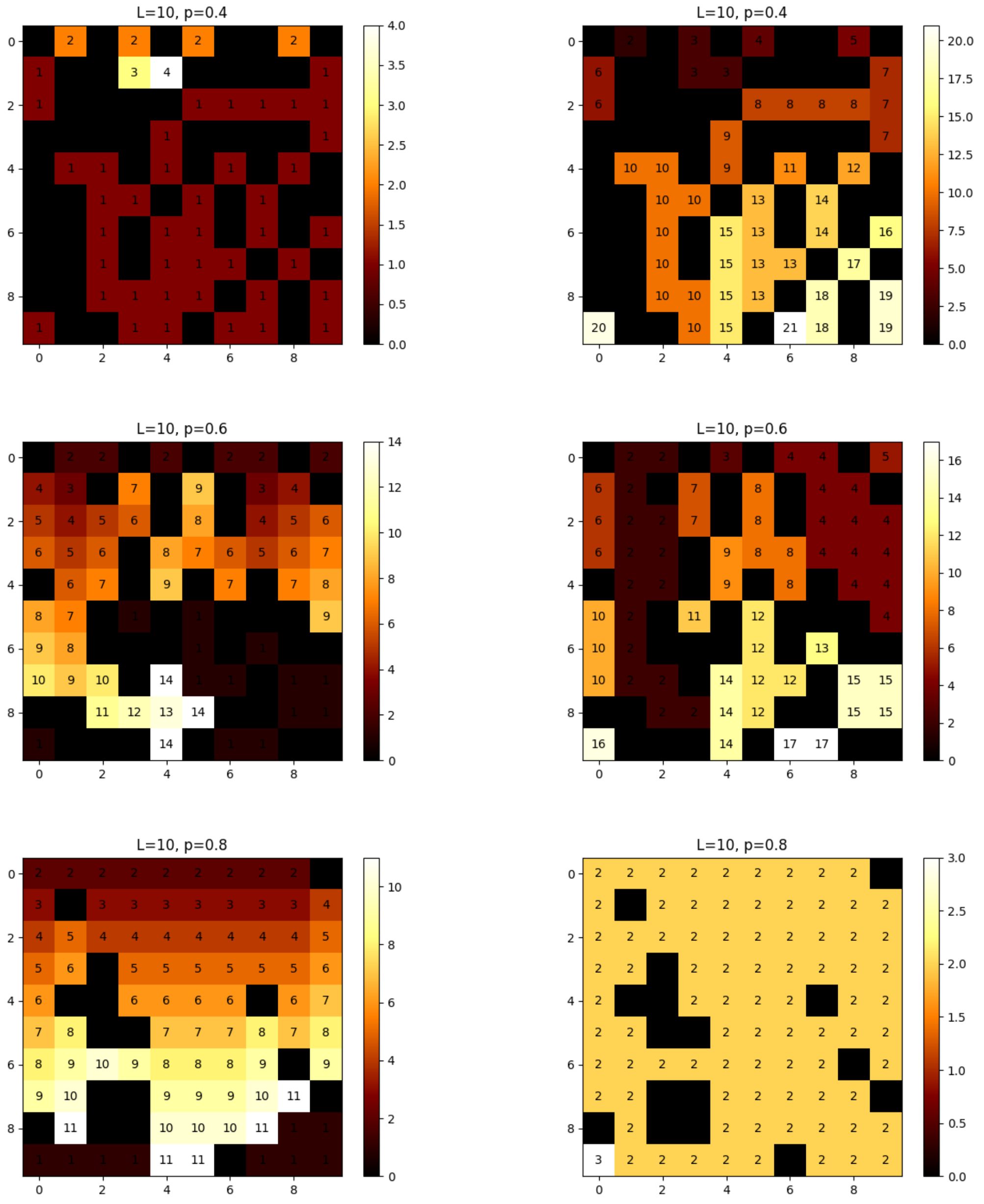

Figure 1: Visualization of sample configurations for two methods: burning method (left site), and HK algorythm (right site).

4 Probability P_{flow} that the path connecting the first and the last row exists as a function of p for $L = 10,\ 50,\ 100$

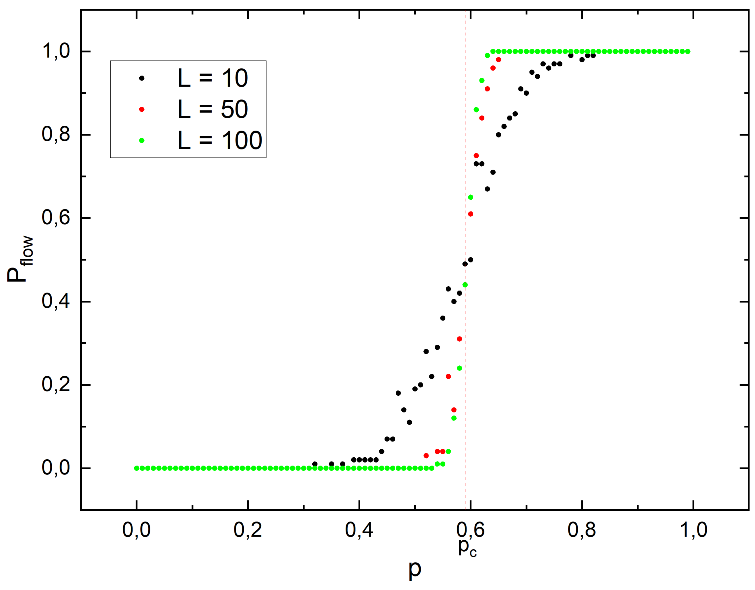

Figure 2: Probability P_{flow} of reaching end against ocupance probability p for different model sizes ($T = 10^2$)

5 Average size of the maximum cluster $\langle s_{max} \rangle$ as a function of p for $L = 10, 50, 100$.

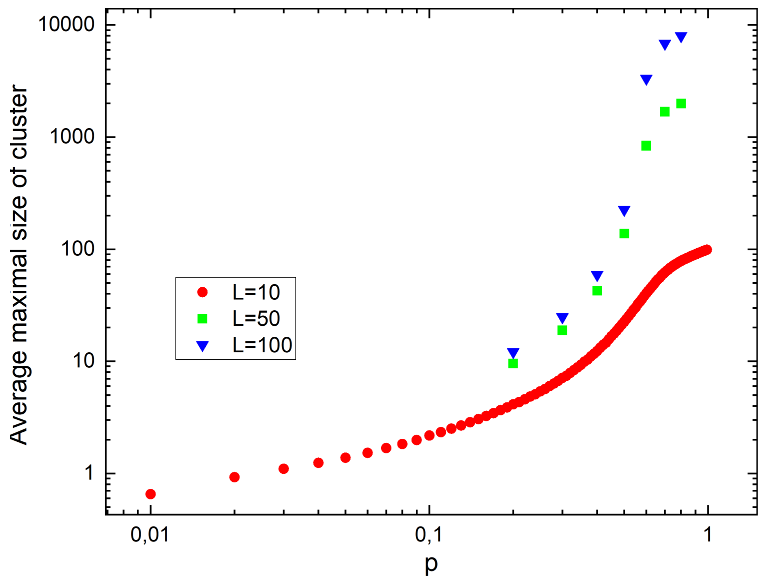

Figure 3: Average size of the maximum cluster $\langle s_{max} \rangle$ as a function of p for $L = 10, 50, 100$ ($T = 10^4$).

6 Distribution of clusters $n(s, p, L)$ for a given p.

6.1 Log-linear

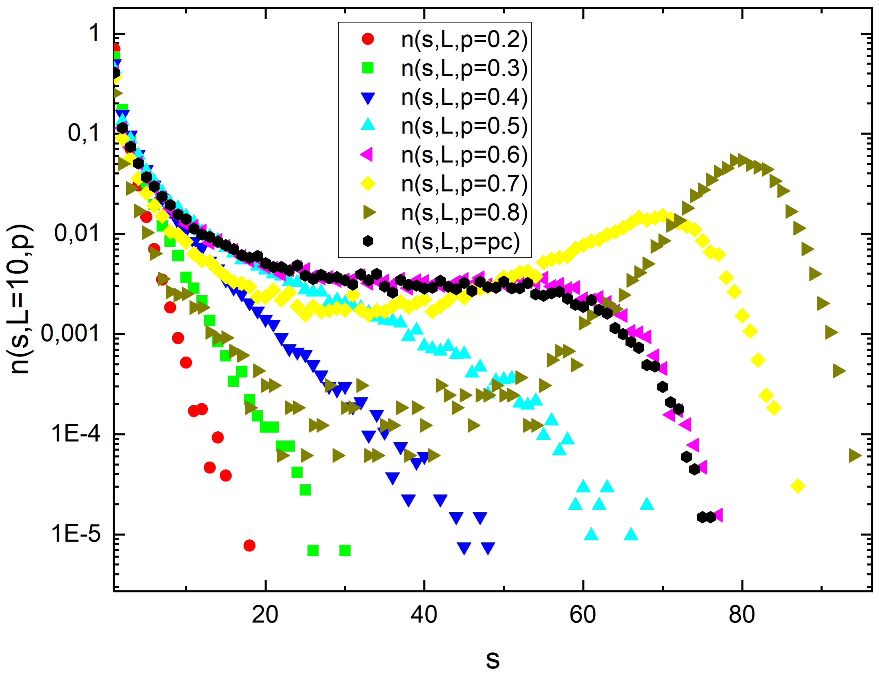

Figure 4: Distribution of clusters $n(s, p, L)$ for a given p in log-linear scale ($p_c = 0.592746$, $T = 10^4$).

6.2 Log-Log

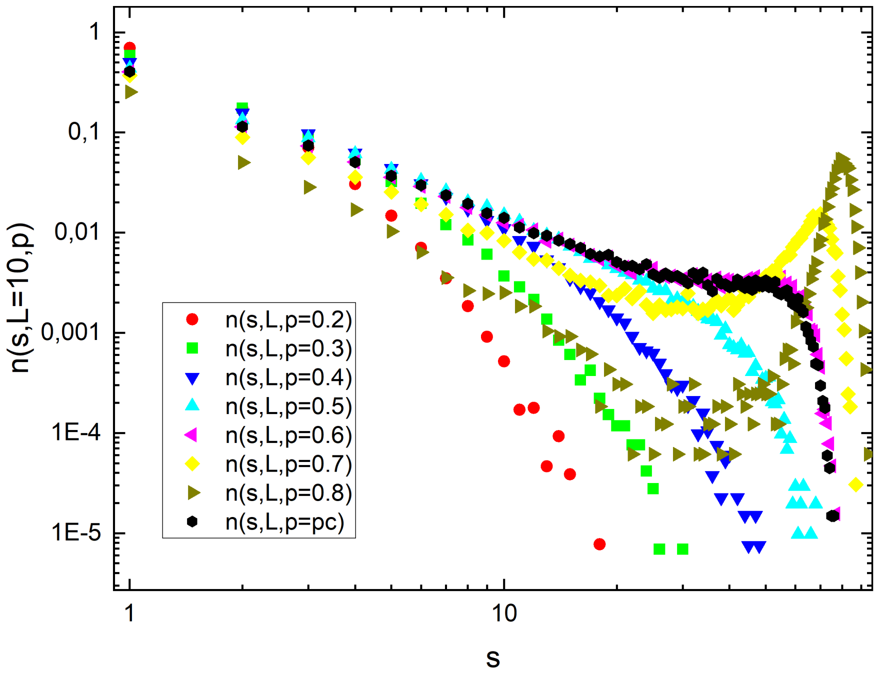

Figure 5: Distribution of clusters $n(s, p, L)$ for a given p in log-log scale ($p_c = 0.592746$, $T = 10^4$).

7 Discussion

Power law Recall power law that states:

$$\text{frequency} = \frac{1}{\text{rank}} \tag{1}$$

In case of percolation model:

$$\text{probability of getting cluster of size s} = \frac{1}{\text{s}} \tag{2}$$

This behaviour can be observed on graph 5. One can observe that law is conserved for small values of p and it disconverge for large sizes of clusters for this calues. For higher values of p law is strongly conserved till certain value of cluster size, then it rizes to high probability - one giga-cluster emerges and finaly converges to zero. This could not happend for low values of p, becouse of low probablility of percolating cluster occurance.