Introduction to Computational Physics

Lecture Percolation

Andreas Adelmann

Paul Scherrer Institut, Villigen
E-mail: andreas.adelmann@psi.ch

https://moodle-app2.let.ethz.ch/course/view.php?id=15323

Plan for today

4 Percolation I

While the molecular dynamics and Monte Carlo methods can, in principle, be used to study any physical system, difficulties appear close to **phase transitions**. To cover phase transitions percolation is a very simple and purely geometric method.
Although there is no dynamics of any kind involved, percolation nevertheless exhibits complex behavior sufficient to study for example phase transition. This simple model will allow us to introduce many concepts that we will need later for the simulation of dynamic systems. These concepts include:

- Phase transitions
- Scaling
- Finite size effects and finite size scaling
- Monte Carlo Simulations

Percolation models can be applied to a variety of problems:

4 Percolation II

- *Oil fields*: The "swiss cheese model" can be used as a simplified model for the storage and flow of liquids in porous media. The porous stone is modeled by spherical cavities distributed randomly in the surrounding medium. If the density of cavities gets larger they start to overlap and form large cluster of cavities. Important questions that can be asked include:
  - For oil drilling we want to know how the amount of oil stored varies with the size of the oil field.

  - Fluids can only flow through the medium of there is a cluster connecting opposite sides. Such a cluster is called a "percolating" cluster. What density of cavities we need to create a percolating cluster.
  - Next we want to know how the the speed of fluid flow through the medium depends on the density.

4 Percolation III

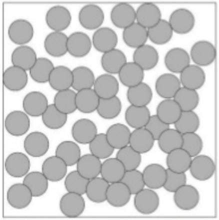
- **Forest fires**: We model forest fires by assuming that a tree neighboring a burning tree catches fire with a probability p. Fire fighters will want to know:
  - How much of the forest will burn?
  - Will the fire spread throughout the whole forest?
- **Spread of diseases**: The spread of diseases can be modeled in a simplified way similar to the forest fire model. Now the important question is: what part of the population will fall ill? Will the disease by an epidemic, spreading throughout the whole country, or even an endemic, spreading over the whole world?
- **Conductance of wire meshes**:
  - How many links can be cut in a wire mesh so that it is still' conducting?
  - What is the resistivity as a function of the ratio of cut links?
- **Vulnerability of the internet**: The internet was designed in 1964 to make computer networks reliable even in the case of attacks in war time. The ' most urgent question is:

4 Percolation IV

  - What portion of the internet is still connected if a fraction p of switches fails?

- **Gelation of liquids**: Gelation of liquids can be modeled by allowing a molecule to form a bond with a neighboring molecule with probability p. Again the interesting questions are:
  - What is the average size of molecule clusters as a function of p?
  - What is the critical concentration at which the largest molecule cluster percolates and the liquid solidifies?
- **Baking of cookies**: A nice household example is given in the textbook by Gould and Tobochnik. Distribute several drops of dough randomly on a cookie tray. When baking the dough will spread and cookies that are close will bake together. If the number of drops is too large we will obtain a percolating cookie, spanning the baking tray from one edge to the other.

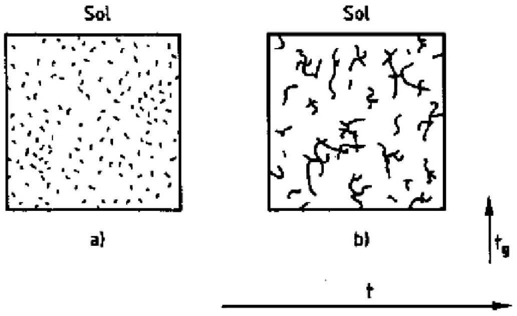
4 Percolation V



Cookies (circles) placed at random on a large sheet.

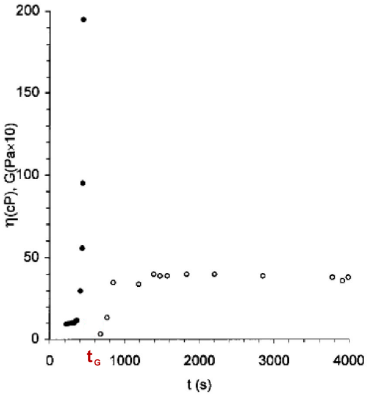
For a detailed discussion we refer to the book by Stauffer and Aharony.

4 Sol Gel I



- a Monomers are dispersed in the liquid (sol)
- b When the liquid is cooled down, the monomers start polymerizing and grow
- c Once a huge macromolecule (which spans through the whole container) has formed the Gel transition

4 Sol Gel II



- Viscosity (filled circles) and shear modulus (open circles)
- At the gelation time t_G the viscosity diverges to infinity and the shear modulus, previously 0, increases to a finite value

4.1 Infinite System I

Following [22] (Hunt) and [10] (Stauffer) we will introduce important parameters. We note, in case of $1d$ and infinite dimensional lattices one can obtain exact results for parameter of interest. Our aim however is to discover these parameters with computer experiments.

Definition

A cluster is a group of nearest neighbouring occupied sites.

Percolation theory deals with **the numbers** and **properties** of the clusters formed when sites are occupied with probability p.

4.1 Infinite System II

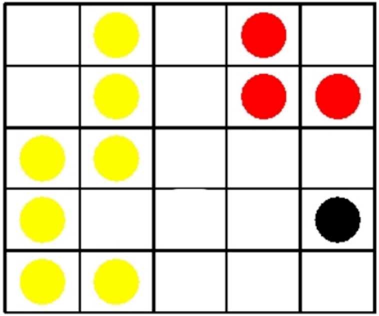


Figure: Percolation in a 2d square lattice $\mathcal{L}$ of linear size $L = 5$. Sites are occupied with probability p. In the lattice above, we have one cluster of size $s = 7$, a cluster of size $s = 3$ and one clusters of size $s = 1$ (isolated site).

4.1 Infinite System III

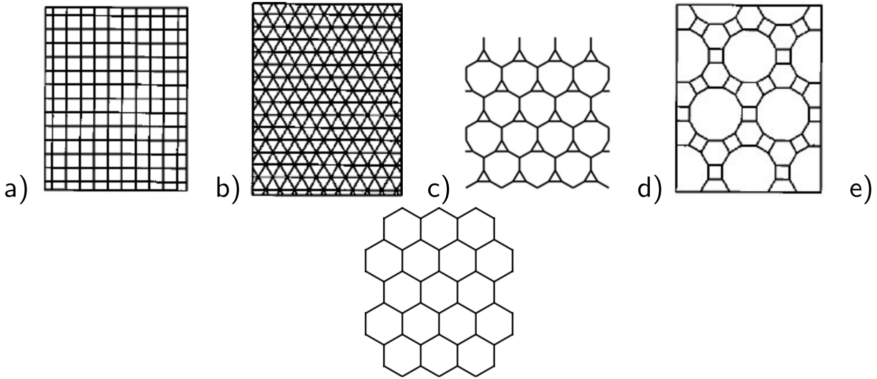


Figure: Different lattices: a) Square lattice, b) Triangular lattice, c) Martini lattice, d) 4,6,12) lattice, e) Honeycomb lattice

4.1 Infinite System IV

Definition

Neighborhood $\mathcal{N}_{\mathbf{i}}$ consisting of $\mathbf{i}$'s nearest neighbors. For example, in a 2D square lattice, the neighborhood of site (i_1, i_2) is formed by the neighbors in the north $(i_1, i_2 + 1)$, east $(i_1 + 1, i_2)$, south $(i_1, i_2 - 1)$ and west $(i_1 - 1, i_2)$, i.e.

$$\mathcal{N}_{(i_1,i_2)} = \{(i_1, i_2 + 1), (i_1 + 1, i_2), (i_1, i_2 - 1), (i_1 - 1, i_2)\}. \tag{1}$$

- we assume a fully empty lattice
- we go through each lattice site and occupy it with *occupation probability p*, each independently of all other sites
- the main goal of percolation theory is to study the formation of so-called *clusters*

Generally, such a system has many clusters, possibly even infinitely many.

4.1 Infinite System V

- continuously increasing the occupation probability $p = 0, \ldots, p = 1$, we can observe that the clusters get bigger and bigger.
- t some point a cluster of infinite size appears $\Rightarrow$ the system is *percolated*

Definition

We say a system is at percolation, or percolates, if sufficiently many objects are connected locally such that a global connection emerges. This global connection is a continuous **chain** or **cluster** of locally connected objects, which is unbounded in size (in infinite systems), or of the order of the system size (in finite systems).

- percolation also refers to a stochastic process of increasing connectivity and eventual emergence of the giant cluster
- this emergence in an ensemble of system configurations constitutes a phase transition

4.1 Infinite System VI

- the central quantity in percolation settings is the cluster size distribution n_s
- computationally, the setting of percolation is a graph
- a typical setting is a regular lattice of sites connected to their nearest neighbours
- in site percolation, all sites are subsequently occupied.
- in bond percolation, it is the bonds that are subsequently added to form a giant cluster of connected sites.

We will not cover bond percolation in this expose.

4.1.1 Phase Transition I

As soon as the occupation probability p reaches a certain value the *percolation threshold or critical point* p_c, a phase transition from the non-percolated system (e.g. sol phase) to the percolated system (e.g. gel phase) containing an infinitely-sized cluster happens.

p_c is a characteristic value

One finds different percolation thresholds for different lattices.

4.1.1 Phase Transition II

lattice	p_c **site**	p_c **bond**
cubic (body-centered)	0.246	0.1803
cubic (face-centered)	0.198	0.119
cubic (simple)	0.3116	0.2488
diamond	0.43	0.388
honeycomb	0.6962	0.65271
4-hypercubic	0.197	0.1601
5-hypercubic	0.141	0.1182
6-hypercubic	0.107	0.0942
7-hypercubic	0.089	0.0787
square	0.592746	0.50000*
triangular	0.50000*	0.34729*

Figure: Percolation threshold for different lattices for site percolation and bond percolation. Numbers with a star (*) can be calculated analytically.

4.1.1 Phase Transition III

- The phase transition occuring in the percolation problem is a second-order phase transition.
- The corresponding order parameter $P(p)$ is called *percolation strength* and is defined as the fraction of sites belonging to the infinite cluster.

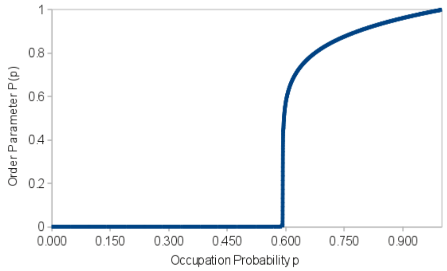
Trivially, we can identify the two situations $P(p < p_c) = 0$ (because no infinite cluster exists) and $P(p = 1) = 1$ (because every site is occupied and therefore connected to all other sites).

- Above but close to the critical point p_c, i.e. $p \gtrsim p_c$, the order parameter behaves as a power law
- the so-called *critical exponent* β strongly depends on the problem that we consider (here in particular the lattice shape and the dimension)

Percolation Strength

$$P(p \gtrsim p_c) \sim |p - p_c|^\beta.$$

4.1.1 Phase Transition IV



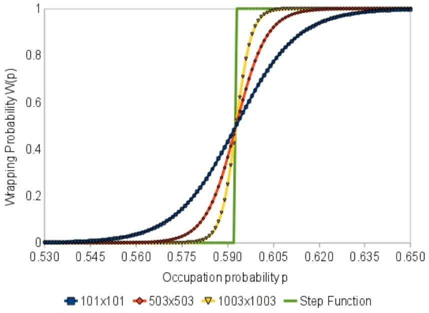


Figure: Wrapping probability as a function of the occupation probability for different lattice sizes and a step function for comparison. One observes that $p_c = 0.592746$

Note:

4.1.1 Phase Transition V

- by looking at the values for different lattices, it becomes clear that different geometries have significantly different thresholds
- for example, the honeycomb lattice has the highest 2D threshold
- The order parameter is also the quantity that defines the *phase diagram* which determines the regions in parameter space that belong to a certain phase.
- In our case, the phase diagram is one-dimensional (depending only on the occupation probability p) and bisected in two regions: The trivial (=non-percolated) phase for $0 \leq p < p_c$ and the percolated phase for $p_c < p \leq 1$.

4.1.1 Phase Transition VI

Besides the order parameter, we can introduce another important quantity, the so-called *wrapping probability* $W(p)$. It is the probability that the system is percolated and is given by

Wrapping Probability

$$W(p) = \begin{cases} 0, & 0 \leq p < p_c \\ 1, & p_c < p \leq 1. \end{cases}$$

4.1.2 Cluster Size Distribution I

Besides the appearance of an infinite cluster, we might be interested in how many clusters we have and what their sizes are.
This information is all contained in the so-called *cluster-size distribution* which is defined as

Cluster-Size Distribution

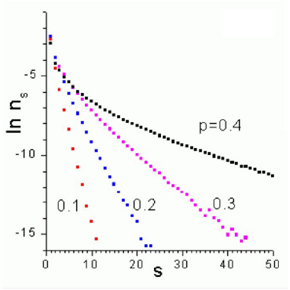
$$n_s(p) = \lim_{L\to\infty} \frac{N_s(p,L)}{L},$$

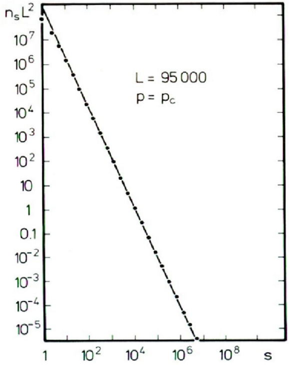
where $N_s(p,L)$ is the number of clusters of size s for given occupation probability p and system's side length L .
It is found that it behaves fundamentally different in the different parameter regions

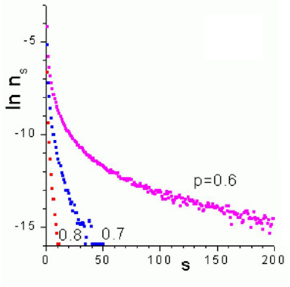
$$n_s(p) \sim \begin{cases} s^{-\theta} e^{as} & p < p_c, \\ s^{-\tau} & p = p_c, \\ e^{-bs^{1-\frac{1}{d}}} & p > p_c, \end{cases} \tag{2}$$

4.1.2 Cluster Size Distribution II

where the parameters θ, a, τ, b depend on the concrete case and d is again the system's dimension. Cluster-size distribution for the three different regions $p < p_c$, $p = p_c$ and $p > p_c$ for a finite size square lattice.







4.1.2 Cluster Size Distribution III

It turns out that important for us is the second moment of the cluster-size distribution

$$\chi(p) = \left\langle s^2 \right\rangle' (p) = \sum_s{}' s^2 n_s(p), \tag{3}$$

where the $'$ indicates that the sum is without the infinite cluster (otherwise $\chi(p > p_c) = \infty$).
Close to the phase transition the second moment behaves as

$$\chi(p \approx p_c) \sim |p - p_c|^{-\gamma}, \tag{4}$$

where γ is again a critical exponent characterizing the phase transition.

4.2 Finite System I

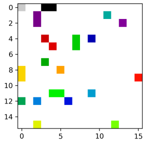
Modelling this process is surprisingly simple.

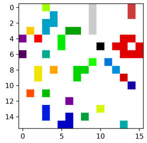
- we create a square lattice with side length L (e.g. $L = 16$)
- every cell can either be occupied or empty
- the initial configuration is that all fields are empty
- We fill each cell with probability p, that is, for each cell we create a random number $z \in [0, 1[$ and compare it to p. If $z < p$, the cell will be marked as occupied otherwise the cell remains empty

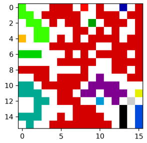
This can be done for different values of p and, one immediately notices:

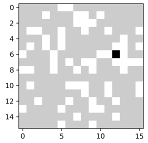
- or small values of p, most cells are empty
- for big values of p, most cells are occupied
- we discover $p = p_c = 0.592...$ when for the first time percolating cluster cluster forms

4.2 Finite System II

$p = 0.1$


$p = 0.2$


$p = p_c$


$p = 0.7$


While p_c is defined for infinite clusters, we can also find it for finite sizes; in that case, p_c is the average probability at which a percolating cluster first occurs. The critical probability is sometimes referred to as the percolation threshold.

4.2 Finite System III

```julia
L1 = 5
L2 = 5
p = 0.6
lattice = zeros(Int64, L1, L2)
for j in 1:L2, i in 1:L1
    if rand() < p
        lattice[i, j] = 1
    end
end
```

4.2.1 The Burning Method I

- while the previously noted observation of a critical probability was still rather general, it would be desirable to have a tool at one's disposal that could provide exact information about a configuration and detect whether a spanning cluster exists.
- the burning method provides us with exactly this information
  - a boolean feedback (yes/no)
  - minimal path length (the minimal distance between opposite sides, following only occupied sites)

Let's assume that we have a grid with occupied and unoccupied sites.

- an occupied site represents a tree while an unoccupied site stands for empty space
- if we start 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

4.2.1 The Burning Method II

- obviously, not only the first row of trees will fall victim to this forest fire, as the neighbours of the trees in the first row will soon catch on fire as well
- the second iteration step is thus that all trees that are neighbours of already burning trees are being torched.

And so what?

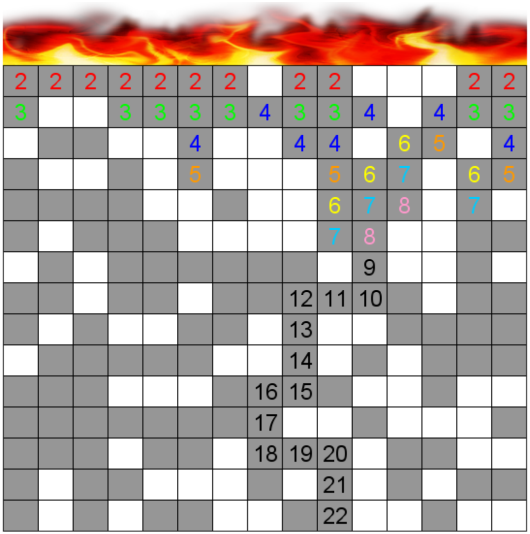
- the iterative method only comes to an end when the fire finds no new victims to devour and consequently dies out or if the fire has reached the bottom
- if the inferno has reached the bottom, we can read off the length of the shortest path from the iteration counter since the number of iterations defines the minimum shortest path of the percolating cluster.

4.2.1 The Burning Method III

The algorithm is as follows:

1. Label all occupied cells in the top line with the marker t=2.
2. Iteration step t+1:
    1. Go through all the cells and find the cells which have label t.
    2. For each of the found t-label cells do
        1. Check if any direct neighbor (North, East, South, West) is occupied and not burning (label is 1).
        2. 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.

4.2.1 The Burning Method IV



4.2.2 The Hoshen-Kopelman Algorithm I

- ☺ percolation threshold
- ☺ fraction of sites in the spanning cluster

The natural extension would be to know how the different clusters are distributed.
In fact, there are several such algorithms; the most popular (and for this purpose most efficient) algorithm is the **Hoshen-Kopelman** algorithm, which was developed in 1976.
Setup:

- lattice as a matrix $N_{ij} \in \mathbb{B}$, 0 (unoccupied) and 1 (occupied)
- an array $M_k \in \mathbb{Z}$ denotes the mass of cluster k (number of sites belonging to a given cluster)

4.2.2 The Hoshen-Kopelman Algorithm II

Initial condition: We start the algorithm by setting $k = 2$ (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 $M_{k=2} = 1$ 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} (from top-left to bottom-right) and try to detect whether an occupied site belongs to an already known cluster or a new one.

- If a site is occupied and the top and left neighbours 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. $N_{ij} = k_0$.

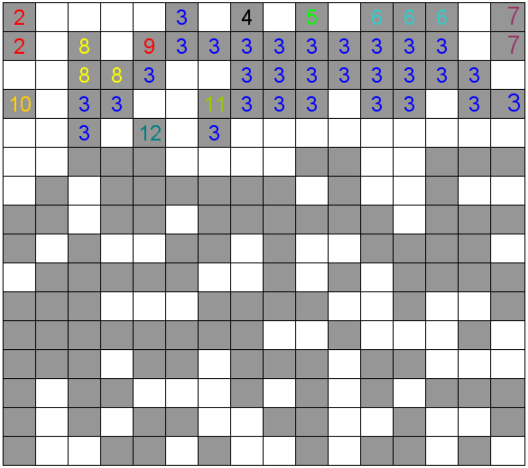
4.2.2 The Hoshen-Kopelman Algorithm III

- 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).

4.2.2 The Hoshen-Kopelman Algorithm IV

The last point is crucial, as we usually deal with a number of sites that are marked in such a way that we have to first recursively detect which cluster they pertain to before carrying out any further part of the algorithm. The recursive detection stops as soon as we have found a k_0 with $M_{k_0} \geq 0$ (i.e. a "true" cluster number).
Once all the sites N_{ij} have been visited, the algorithm ends up with a number l of clusters (where $l < k_{max}$). The only thing left to do is to construct a histogram of the different cluster sizes. This is done by looping through all the clusters $k \leftarrow 2..k_{max}$ while skipping negative M_k.

4.2.2 The Hoshen-Kopelman Algorithm V



4.2.2 The Hoshen-Kopelman Algorithm VI

Let us write down the Hoshen-Kopelman algorithm:

```
1. $k \leftarrow 2$, $M_k \leftarrow 1$
2. for all $i, j$ of $N_{ij}$
    1. if top and left are empty (or non-existent) $k \leftarrow k+1$, $N_{ij} \leftarrow k$, $M_k \leftarrow 1$
    2. if one is occupied with $k_0$ then $N_{ij} \leftarrow k_0$, $M_{k_0} \leftarrow M_{k_0} + 1$
    3. if both are occupied with $k_1$and $k_2$ (and $k_1 \neq k_2$) then choose one, e.g. $k_1$ and $N_{ij} \leftarrow k_1$, $M_{k_1} \leftarrow M_{k_1} + M_{k_2} + 1$, $M_{k_2} \leftarrow -k_1$
    4. If both are occupied with $k_1$, $N_{ij} \leftarrow k_1$, $M_{k_1} \leftarrow M_{k_1} + 1$
    5. if any of the $k$'s considered has a negative mass $M_k$, find the original cluster they reference and use its cluster number and weight instead
3. for $k \leftarrow 2..k_{max}$ do
    1. if $M_k > 0$ then $n(M_k) \leftarrow n(M_k) + 1$
```

Remarks:

- the algorithm is very efficient since it scales linearly with the number of sites