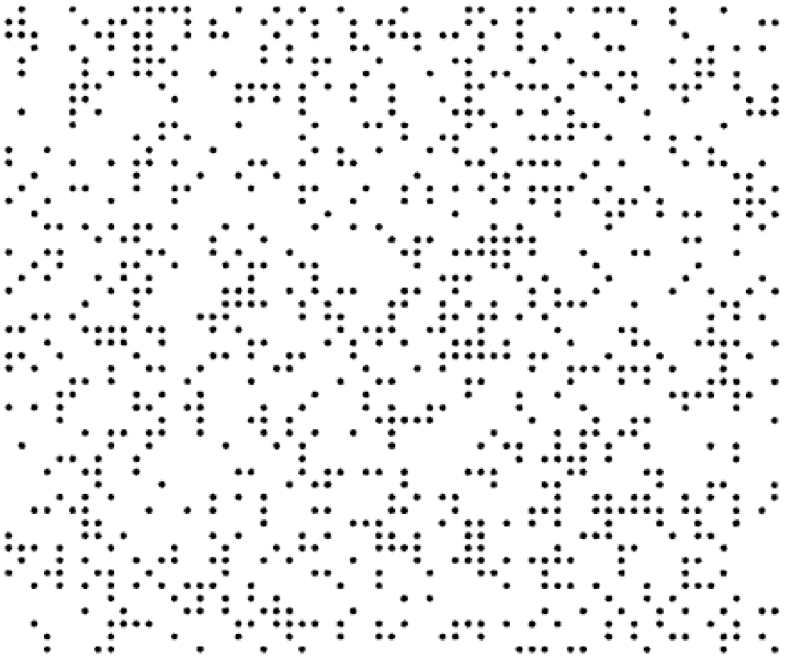
Simulations in Statistical Physics

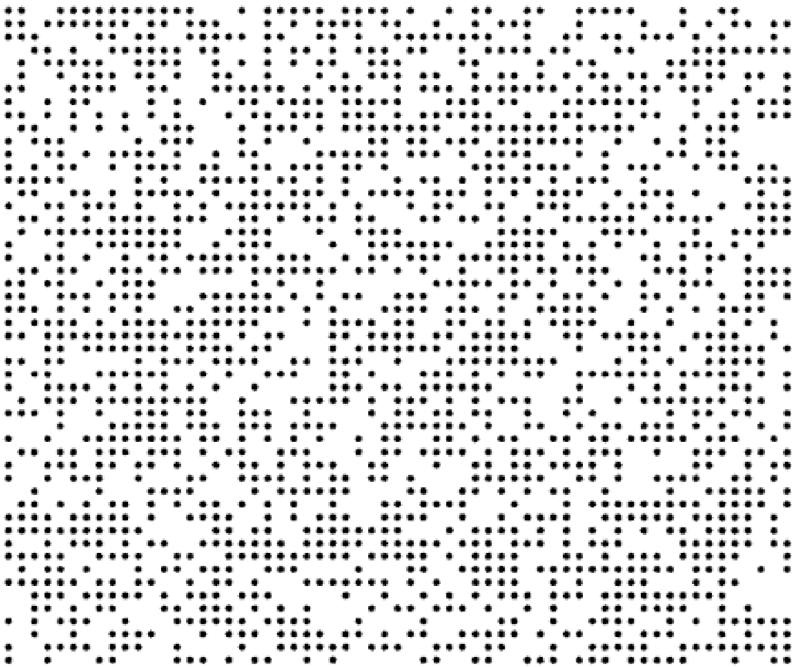
Course for MSc physics students

János Kertész

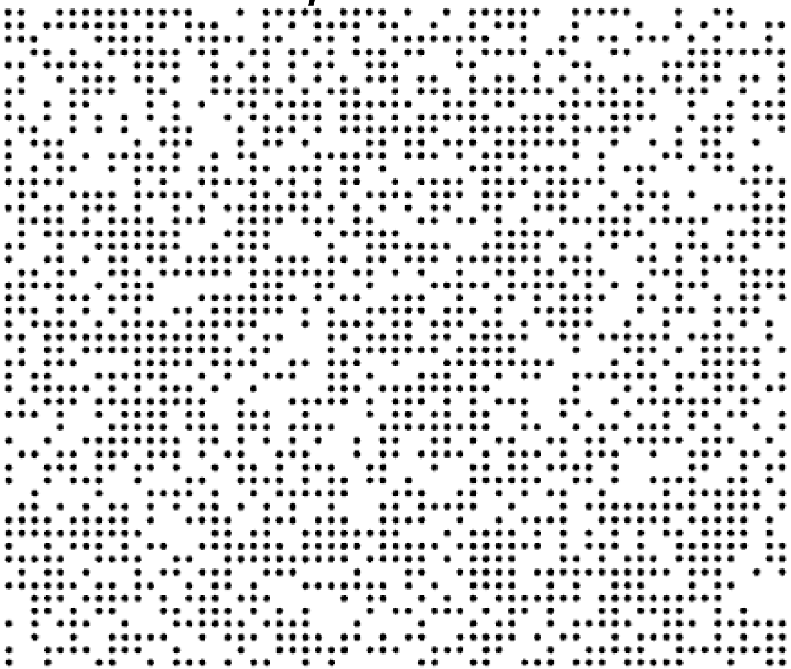
Lecture 4



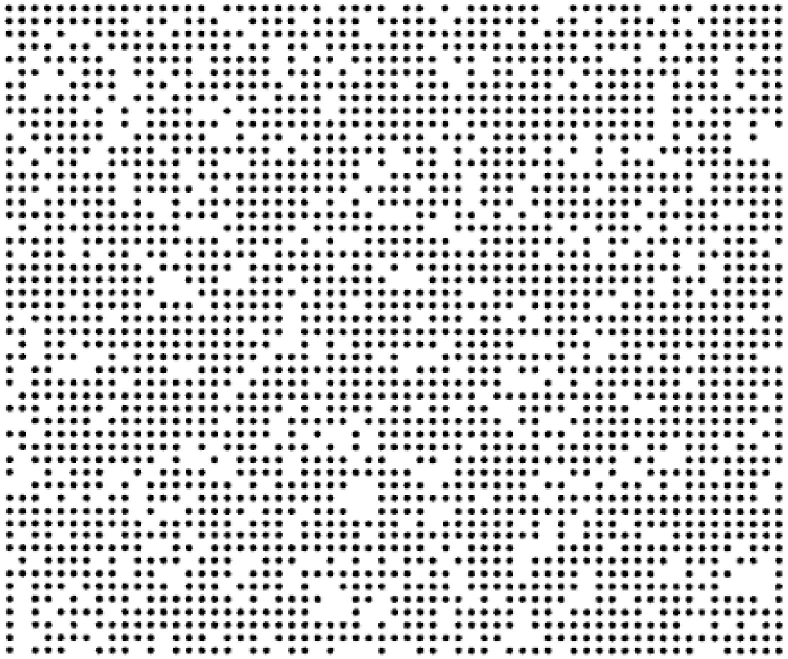
p =0.3



p=0.58

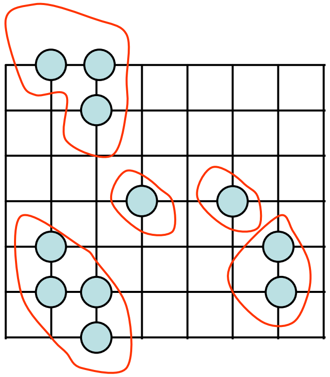


p =0.6



p =0.8

Task: Given a set of occupied sites in a d-dimensional lattice identify and count the clusters!

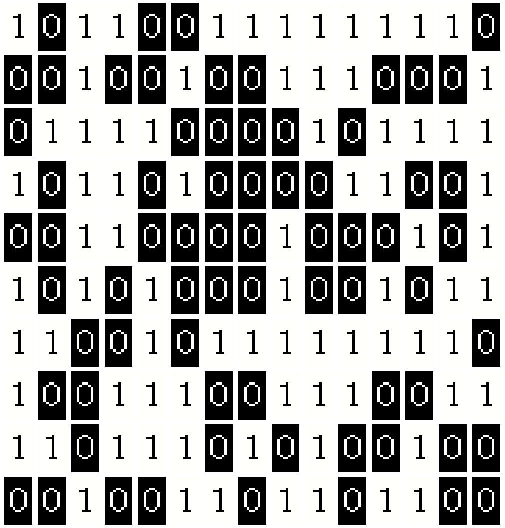


$$n_1 = 2/L^2 \quad n_2 = 1/L^2 \quad n_3 = 1/L^2 \quad n_4 = 1/L^2$$

The Hoshen-Kopelman algorithm identifies the clusters, determines their sizes by sweeping through the lattice once as a typewriter (starting top left and ending bottom right).

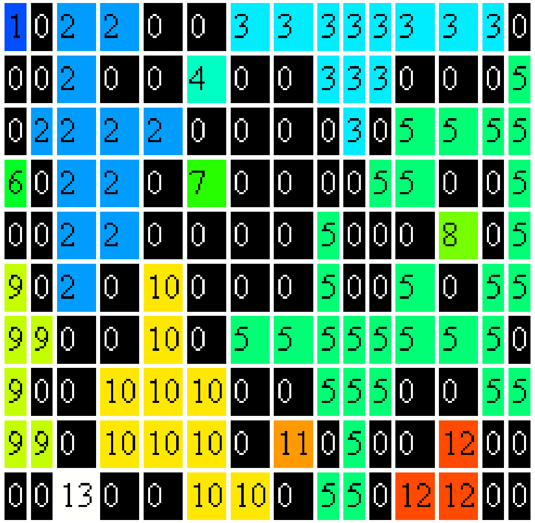
Let us store the information about the lattice is stored in the array IS(*I*,*J*), where *I* and *J* go from 1 to *L*. (For simplicity we introduce the algorithm in d=2, the generalization to higher dimensions is straightforward. IS(*I*,*J*) = 1 if the site is occupied and IS(*I*,*J*) = 0 if it is emty.
Clusters are labeled by a variable LABEL(*I*,*J*).

Lattice



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

Aim (+ size of the cluster with label LAB)



1	0	2	2	0	0	3	3	3	3	3	3	3	3	0
0	0	2	0	0	4	0	0	3	3	3	0	0	0	5
0	2	2	2	2	0	0	0	0	3	0	5	5	5	5
6	0	2	2	0	7	0	0	0	0	5	5	0	0	5
0	0	2	2	0	0	0	0	5	0	0	0	8	0	5
9	0	2	0	10	0	0	0	5	0	0	5	0	5	5
9	9	0	0	10	0	5	5	5	5	5	5	5	5	0
9	0	0	10	10	10	0	0	5	5	5	0	0	5	5
9	9	0	10	10	10	0	11	0	5	0	0	12	0	0
0	0	13	0	0	10	10	0	5	5	0	12	12	0	0

When we go through the lattice, we open a new cluster label, whenever there is neither occupied upper nor left neighbor.

Clusters like that with label 5 are problematic!

Book keeping using the array KLASS(LAB) with label arguments.

KLASS(LAB) contains info about d=2

i) If LAB is a true label and in this case about the size of the cluster (KLASS(LAB)>0)

ii) If LAB is not a proper one, which is its ancestor LABN: (KLASS(LAB)=-LABN) But LABN may not be a proper label so repeat procedure…

a) If the site is unoccupied we set label L^2

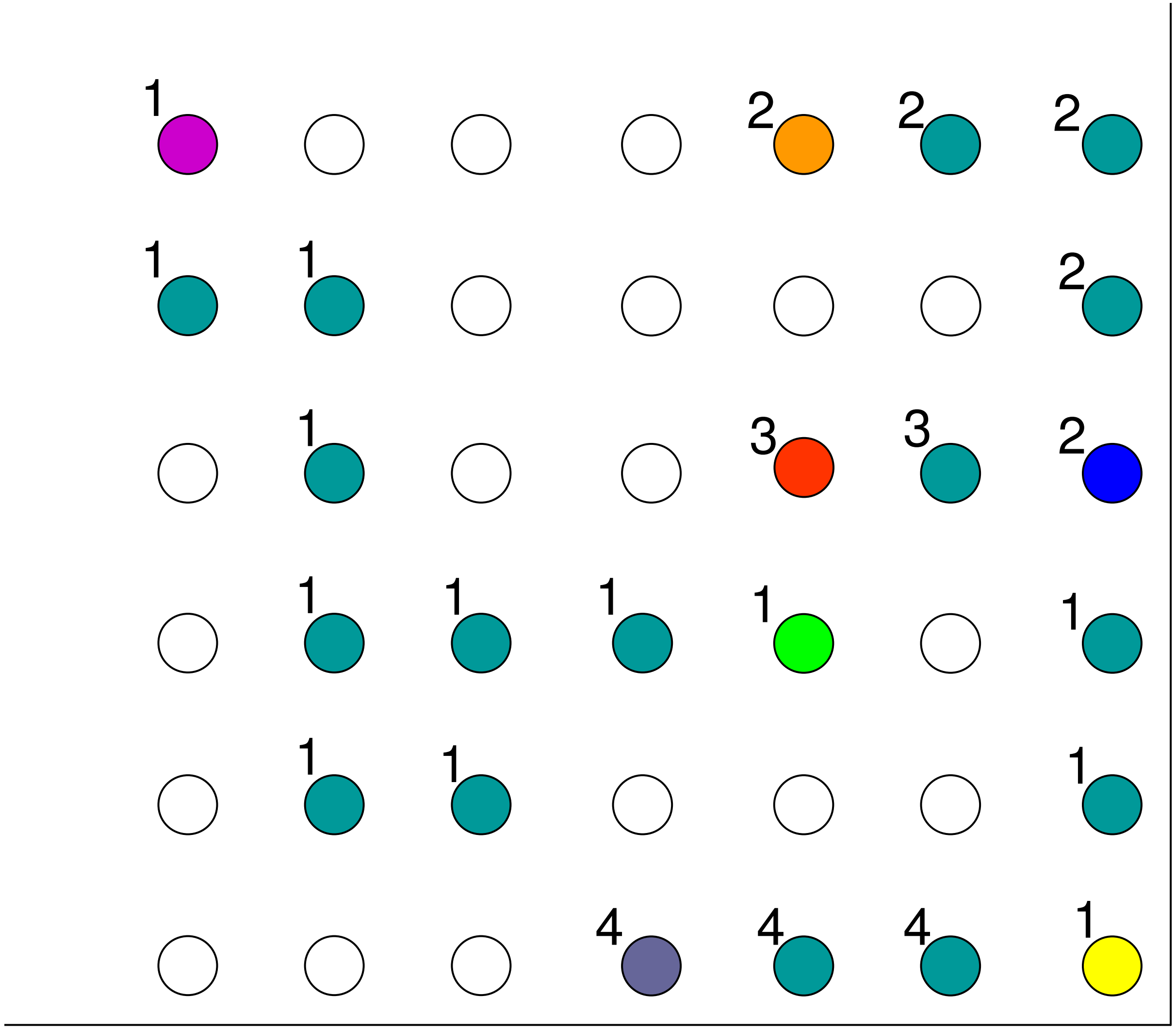
At an occupied (bulk) site we have to look only at the left and upper neighbor.

b) If none are occupied, open a new label NL, and KLASS(LN)=1

If only left neighbor is occupied, take its label over (it was a true label and nothing happened since then)

c) If the only the upper neighbor is occupied with label LAB search for the root label with LT=KLASS(-KLASS(-KLASS(-…(LAB)))) until a positive label is found. Take that label, as proper, increas KLASS(LN) by one, and declare KLASS(LAB)=-LT

d) If both upper and left are occupied, look at the root label of the upper, take the smaller and unify the two clusters.



● ● ● ● new labels NL

NL=NL+1
KLASS(NL)=1

● Status:
KLASS(1)=4
KLASS(2)=4
KLASS(3)=2
Action:
KLASS(2)=
KLASS(2)+KLASS(3)+1
KLASS(3)=-2

● KLASS(2)=-1 etc

● Status: KLASS(4)=3, KLASS(1)=20…
Action KLASS(1)=KLASS(1)+KLASS(4)+1
KLASS(4)=-1

(The Hoshen-Kopelman algorithm is an example of the Union-Find algorithms, which find equivalence classes by efficient book keeping. More general U-F is needed, e.g., for cluster check on an arbitrary graph.)

Computational demand is little (log) more than linear in L^d (because of the search for the root labels).

Having gone through the sample, the 0<labels<LABMAX will be the identifiers of the clusters, and the corresponding KLASS values will tell the sizes. Thus a statistics about cluster sizes are trivially obtained.

It is convenient to introduce helical BC-s. The first and last raws can be easily connected by going through once more the first raw with imagining that the last one is its upper neighbor.

The cluster numbers follow also a scaling form and right at the critical point they have a power law decay $n_s \sim s^{-\tau}$

Sometimes we ask only, whether there is an occupied path through the sample. This question can be handled by a simplified H-K algorithm:

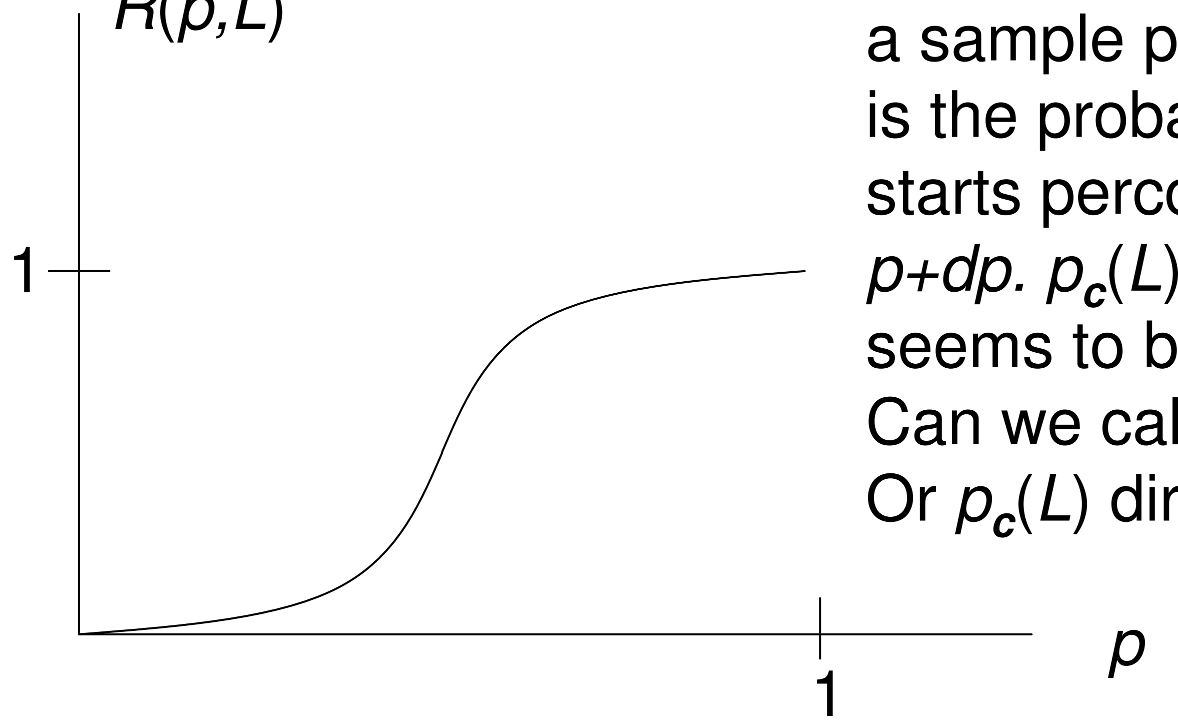
We give all the occupied sites in the first raw the label 1

Then we apply HK and check in every raw whether the true label 1 has occurred. If it does in the last raw – the sample „conducts”; if in any of the rows the check fails we stop the search with that sample.

For random percolation the memory demand can be substantially decreased. We occupy the sample simultaneously with the cluster check. The decision about occupation is done by comparing a generated random number $r \in (0,1)$ with *p.* If $r < p$ the site is occupied, otherwise it is empty. This decision is made when we first visit the site. And since we never see it again if it has been checked from below, we put new information into the corresponding array element. This way only an array of $L^{d-1}+1$ elements is needed to store the information about occupation.

What is the finite size critical point (percolation threshold)? It is not uniquely defined. E.g., the average *p* where the samples start percolating. How to calculate?

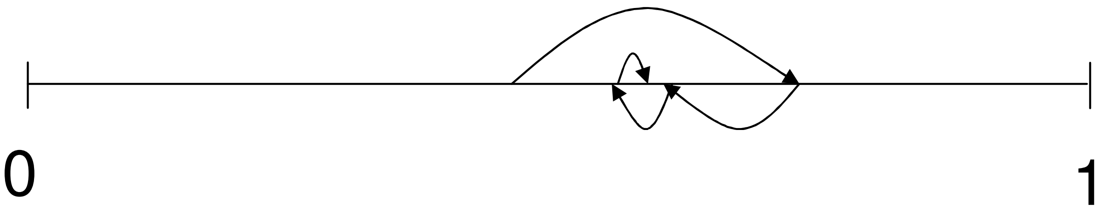
One possibility is to generate many samples, check percolation and take the ratio *R*(*p,* L) of # of percolating samples to all. The obtained curve will be:



R is the probability distribution that a sample percolates. *P*(*p,* L)=d*R/*d*p* is the probability that a sample starts percolating between *p* and *p+dp.* $p_c(L)$=$\langle p\rangle_P$ over this density seems to be a good definition. Can we calculate *P*(*p,* L) directly? Or $p_c(L)$ directly?

In order to find p_c for one particular arrangement, we need to fill up the sample gradually. This takes too long time.

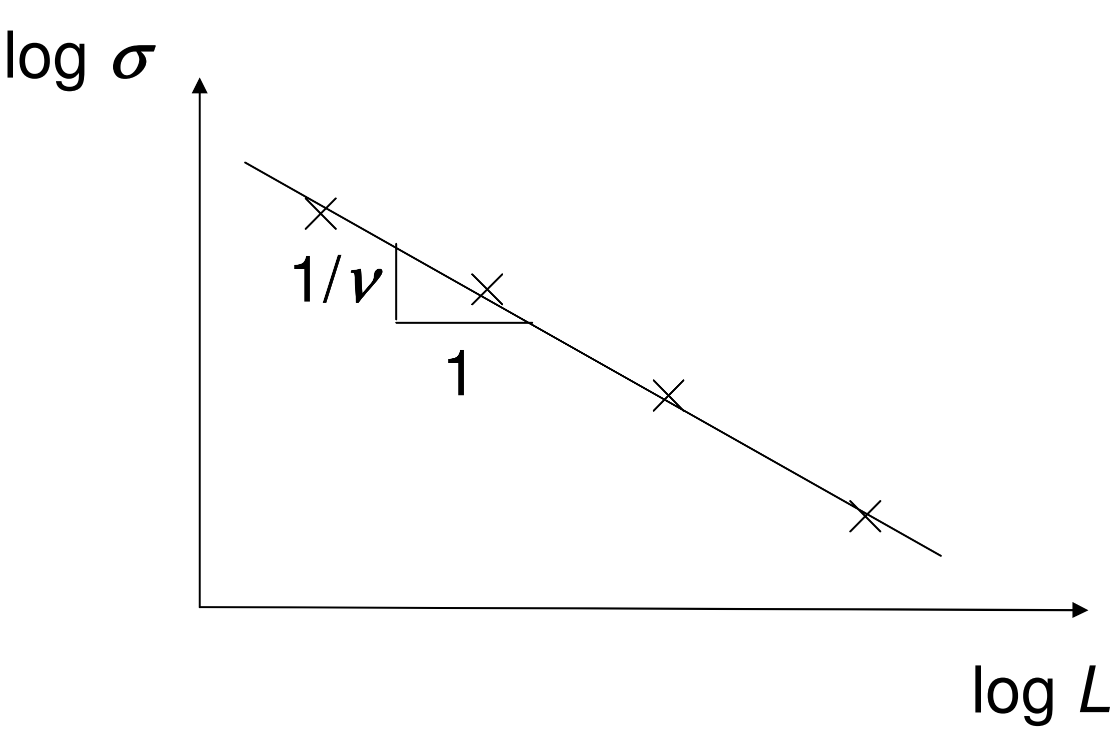
Make use of the fact that the sequence of random numbers will be the same if the seed is the same. Start from a seed and a p and use dichotomic confinement.

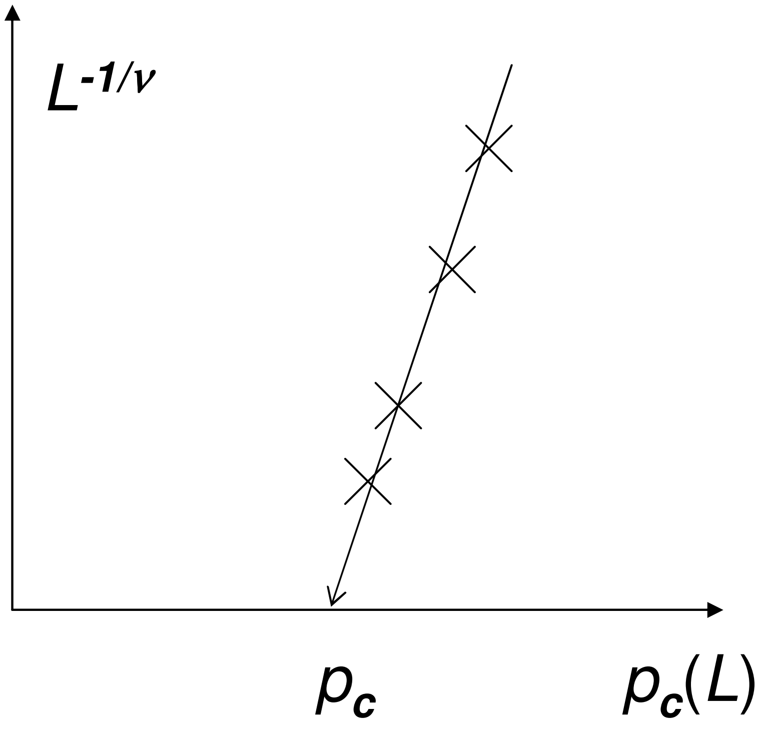


In the n-th step, move back with $(1/2)^n$ if the sample percolates,
move forward if not

Use always the same seed → find the actual p_c(sample)
with $(1/2)^n$ precision.

$$\left.\begin{array}{ll}\text{Take an average over } p_c(\text{sample}) \rightarrow & p_c(L)\\ \text{and calculate standard deviation} & \sigma(L)\end{array}\right\}\ \text{use FSS}$$



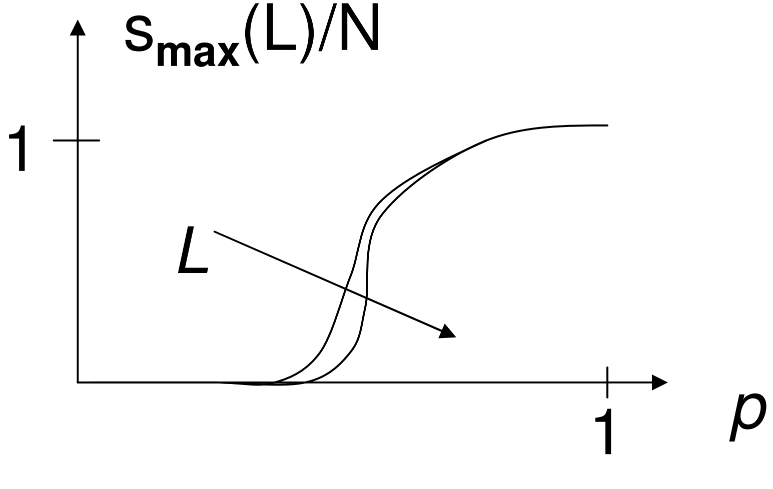


How to get another independent exponent? E.g.,

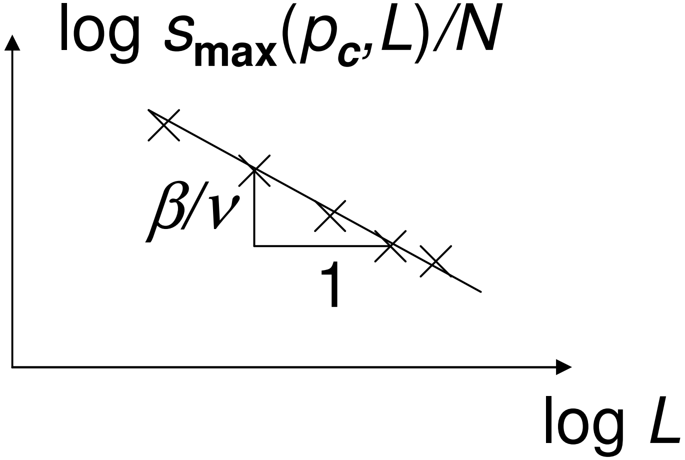
$$P_\infty \sim (p - p_c)^\beta \quad \text{FSS:} \quad P_\infty(p_c, L) \sim L^{-\beta/\nu}$$

What is P_∞ on a finite lattice? One possible definition: The relative weight of the largest cluster. This will not be 0 for

$p < p_c$, but increases sharply from p_c



FSS for P_∞



How does the size of the largest cluster at p_c scale with L?

$$s_{\max}/N \sim L^{-\beta/\nu}$$

$$N = L^d$$

$$s_{\max} \sim L^{d-\beta/\nu}$$

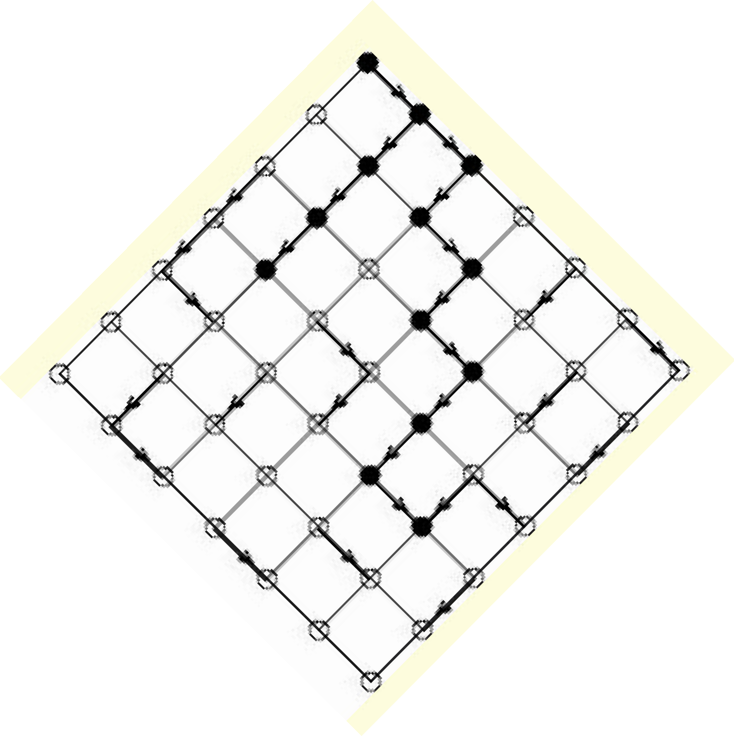
Usually the mass scales with L^{d}, this is, e.g., the case for the larges cluster size above the threshold.
If the mass m of an object scales as , $m \sim L^D$ where $d_{\mathbf{topological}} < D < d_{\mathbf{embedding}}$ then the object is a fractal.

The „incipient infinite cluster at threshold is a fractal,

$$D = d - \beta/\nu$$

Analytical results on percolation:
1d trivial
2d very difficult, but mostly solvable. E.g., 2d exponents known ($\nu = 4/3$, $\beta = 5/36$). Many thresholds (not universal quantity!) are known, e.g., triangular site p_c = square bond $p_c = 1/2$.

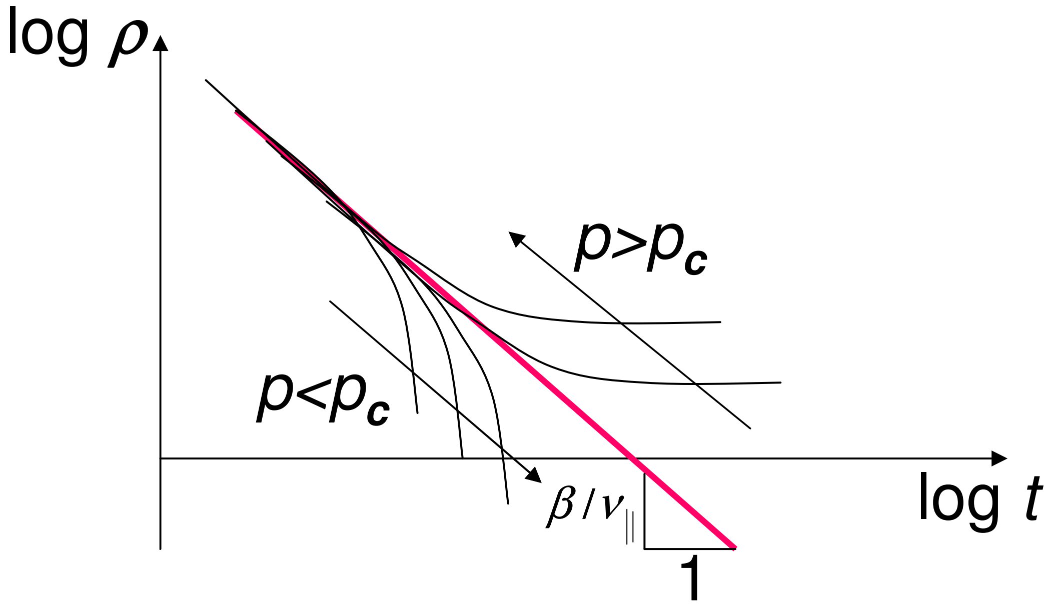
Directed percolation



Percolation in strong N wind or gravity

This problem is more difficult than ordinary percolation, because it can be considered as a time dependent process. Accordingly instead of 2, there are 3 independent exponents: $\nu_{\perp}, \nu_{||}$ and β and these are not known analytically even in 1+1 dimensions (d+1 is a usual notation in this in this case). However, due to the absence of backflow, this problem is numerically much simpler than ordinary percolation and the exponents of 1+1 dimensional directed percolation are known to highest precision (upt 5-6 digits) among the non-trivial ones.

How to calculate the critical point and the exponents? Let us deal with the square lattic (tilted by 45º). We can simulate very large samples, as we will see.
Imagine that the upper most raw is „wet" and we deal with bond percolation. The density ρ of wet sites will be less in the next raw, etc. How does ρ depend on t (we consider the z direction as time).



Even at p_c the plain power law would not hold forever on a finite size sample, but on the very large samples possible to simulate this effect is not seen

The critical point can be found in a dichotomic way by.
The exponent $\beta/\nu_{\|}$ is obtained from the slope of $\rho(t)$ at p_c on a log-log plot.
FSS argument: $\rho(\Delta p,t,L) \sim b^{-\beta/\nu_{\perp}}\rho(b^{1/\nu_{\perp}}\Delta p, t/b^{z}, L/b)$

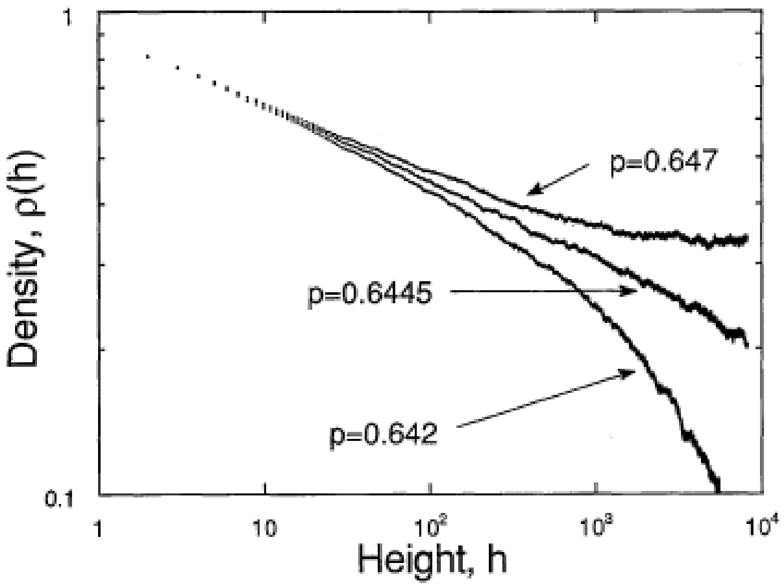
With $z=\nu_{\|}/\nu_{\perp}$ In a large enough sample we can take $L=\infty,\ \ \Delta p=0$

Defining $b=t^{1/z}$ we get the result above. If we look at the scaling with *L* such as the *t* – dependence can be ignored ($t\rightarrow\infty$), we can measure $\beta/\nu_{\perp}$. Finally, *z* can be measured by measuring how the length of finite clusters at criticality scale with the width.

Square site directed percolation

Journal of the Physical Society of Japan
Vol. 62, No. 7, July, 1993, pp. 2200-2203

S. Isogami, M. Matsushita



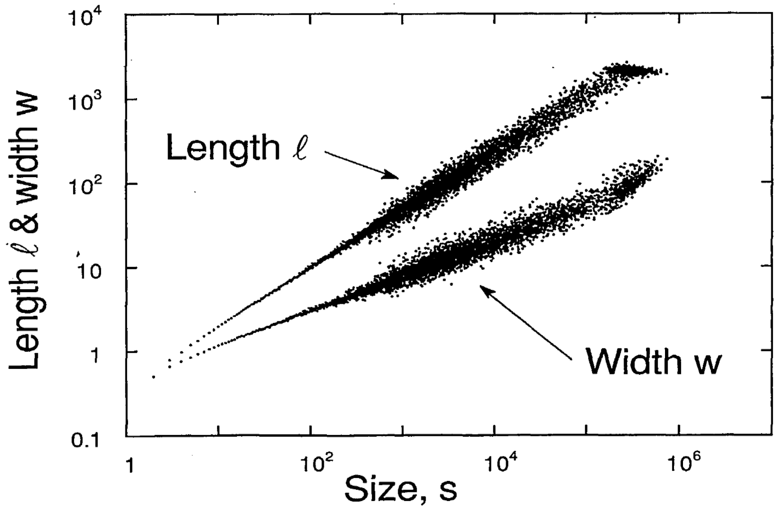


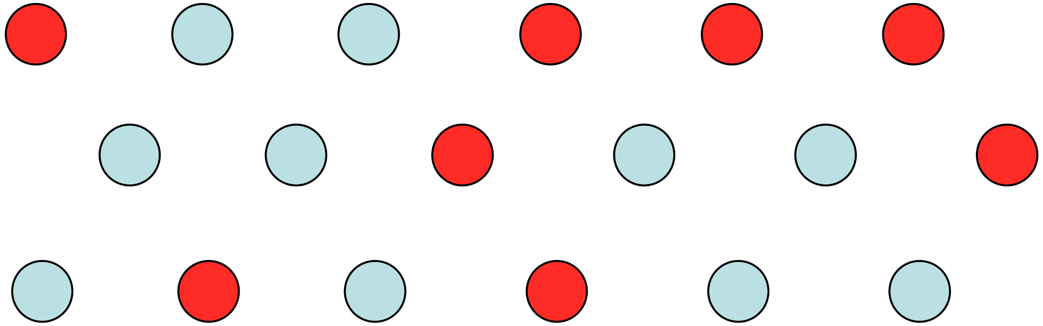
Fig. 4. Relationship of the length l and width w to the size s of clusters for bond concentration $p=0.6445$.

(Self affine fractals)

	$d=1$	$d=2$	$d=3$	$d \geq 4$
β	0.276486 ± 0.000008	0.583 ± 0.003	0.813 ± 0.009	1
$\nu_{\perp}$	1.096854 ± 0.000004	0.733 ± 0.008	0.584 ± 0.005	$1/2$
$\nu_{\|}$	1.733847 ± 0.000006	1.295 ± 0.006	1.11 ± 0.01	1

H. Hinrichsen: Adv. Phys. **49**, 815 (2000)

Simulation trick: The process is Markovian (no backsteps) Percolation depends only locally on the state of previous neighboring sites. The information is only binary. If we store the information about the wet sites in bits, simple bitwise logical decisions allow for a parallel treatment.

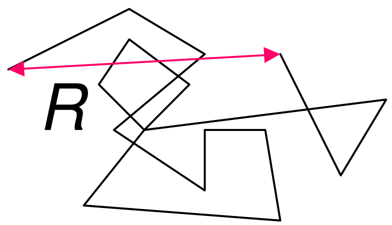


Bitwise AND with ↖
OR
Bitwise AND with ↗
neighbors

Take care of:
- End of words (usually 32 bits)
- Even – odd differences
- BC-s

(Simple example of „multispin coding” = poor men’s parallel computing)

The problem of **Conformation of a polymer chain** is related to another **geometrical phase transition.**
Consider a polymer chain in a good solvent. How to characterize its conformation?



3d object. How does the end-to-end distance *R* depend on the number of monomers (mass)?
Fractal scaling $R \sim N^{1/D} \sim N^{\nu}$

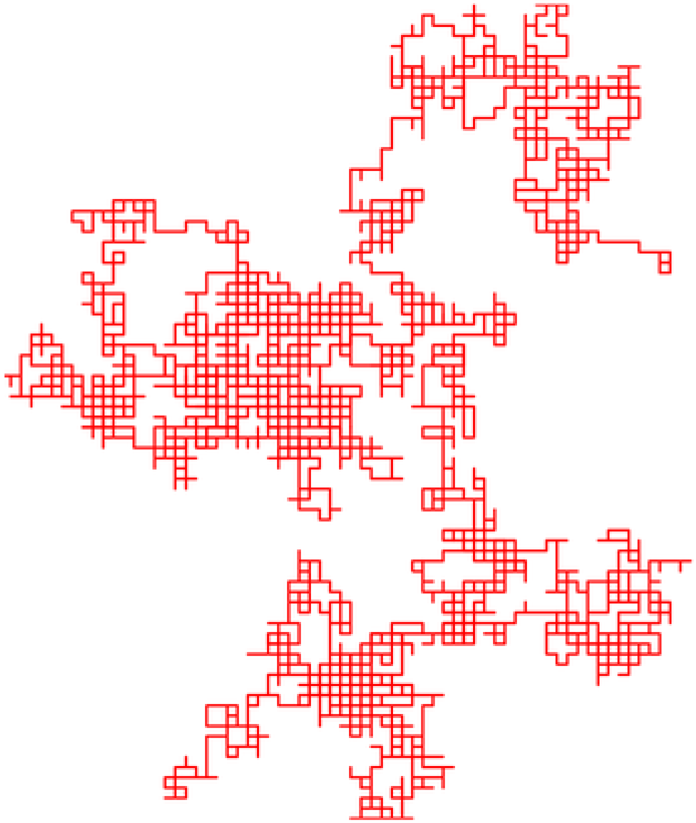
Simplest model: The polymer is a random walk

In this case we know from the theory of diffusion: $\nu = 1/2$

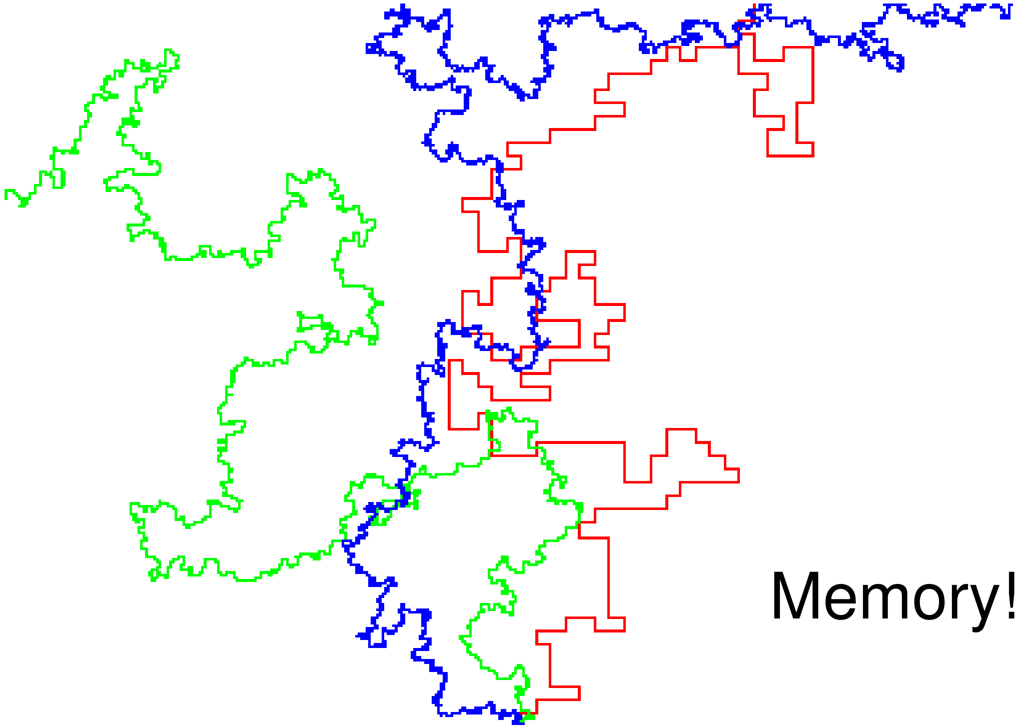
(Scattering) experiments show different value. What is wrong?

Monomers have a strong short range repulsion (excluded volume). Model: Self-avoiding walk (SAW)
Let us study this on lattices! (Justified by universality.)

Self-avoiding random walk is a walk with no intersections. In 2d:



Random walk on the square lattice



Memory!

SAW

http://polymer.bu.edu/java/java/saw/sawapplet.html

(Why phase transition? Grand canonical ensemble → critical μ)

The configurations to be considered are given.
The stat. phys. problem is not yet defined:

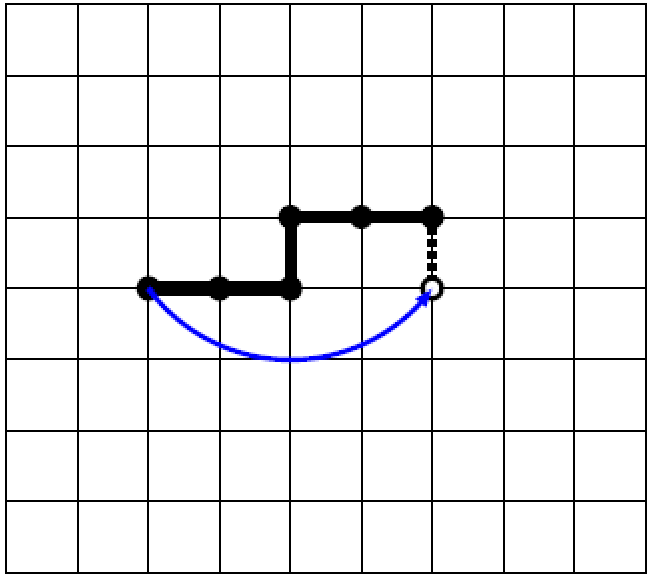
What is the weight of a particular configuration?

Random walk: All walks of length *N* have the same weight.
SAW: All self-avoiding walks of the length *N* have the same weight

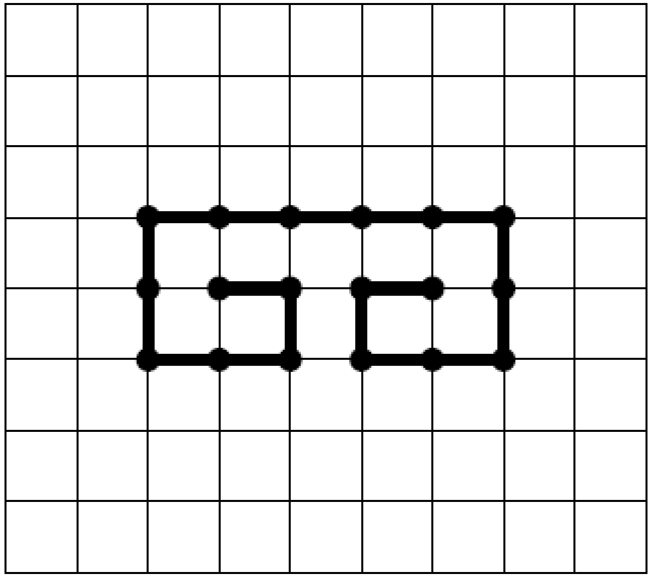
Simplest simulation: Generate a plain random walk and throw it away, when an intersection occurs. The probability of intersections grows exponentially → only very short (*N*~10) walks. Far from „Asymptotia”.

Simple correction: If intersection occurs, step back, and correct the weight: We assign to each monomer *i* the weight $z_i \,/\, q_i$, where z_i is the # of available neighbors and q_i is the coordination number. z_i is updated when intersection occurs. The total weight is $\prod_{i=1}^{N} z_i \,/\, q_i$

Reptation method



We start from a polymer of given length and do the a snake-like motion (tail→head). If the chain gets stuck tail and head are exchenged.
Configurations are generated from each other: Consideration of relaxation time for sampling is necessary.



Trapped configuration (weight=0)
Reptation is non-ergodic, but only slightly. Such configurations are so rare that their absence does not influence the results.

$$\nu = \begin{cases} 3/4 & d = 2, \\ 0.59... & d = 3, \\ 1/2 & d = 4. \end{cases}$$

B. Li, et al
J. Stat. Phys.
80, 661 (1995)

The problem of polymer conformation is very rich:
- Good/bad solvent
- Polymer-polymer interactions
- Thermal effects

Some of these problems can be approached in a purely geometric way.

What about thermal effects? In reality the weights are not binary (1 or 0). There is a Hamiltonian describing the monomer-monomer interaction and the weights are given by the (canonical) distribution.

How to simulate a Hamiltonian system?