Computer Simulations in Physics

Course for MSc physics students

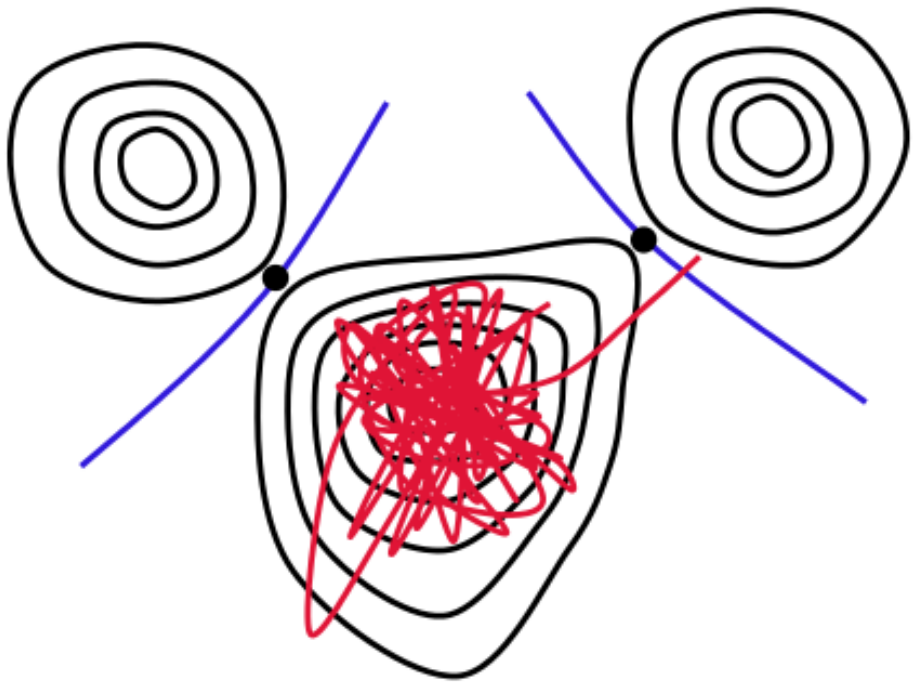
Janos Török

Department of Theoretical Physics

March 5, 2020

Kinetic Monte Carlo

- Particle sits in a potential well for ages . . .
- What to do?

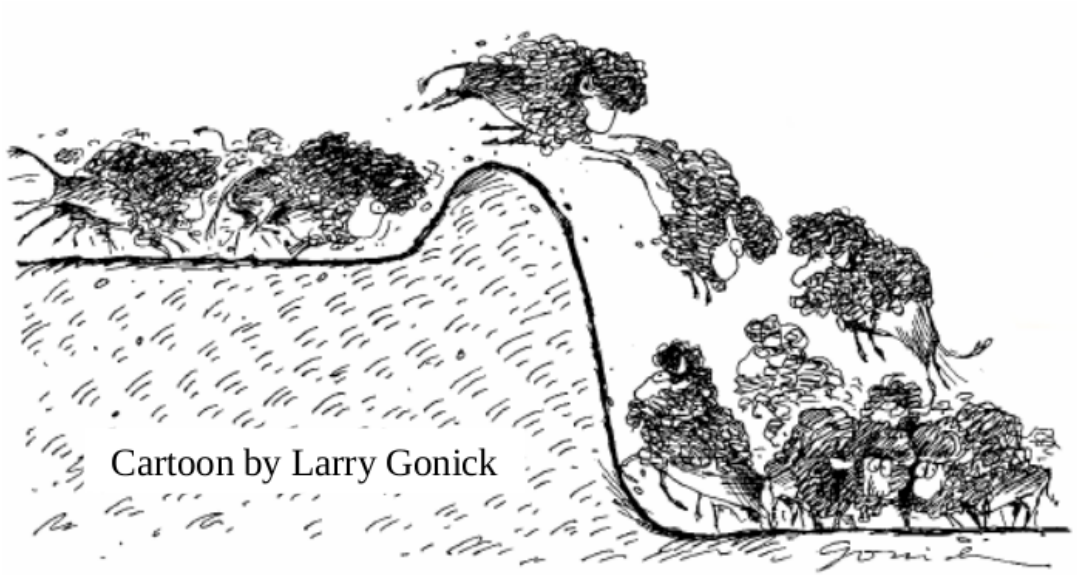


Kinetic Monte Carlo

- Long lasting steady state positions
- Slow thermally activated processes
- Infrequent-event system

Solution:

- Consider only jumps between neighboring energy wells
- Probability of jump $P \sim \exp(-\beta E_b)$
- Rate of jump $i \rightarrow j$, $k_{ij} = E_b$.



Kinetic Monte Carlo

- All possible moves i
- Rates for moves k_i
- Calculate the cumulative function $K = \sum_i k_i$
- Get a uniform random number u (between 0 and 1)
- Execute the event i for which $\sum_{j=1}^{i} k_j > uK > \sum_{j=1}^{i-1} k_j$
- Get new uniform random number u' (between 0 and 1)
- Update time to $t = t + \Delta t$, $\Delta t = -\log(u')/k_i$
- Recalculate rates, which have changed
- Restart loop

Kinetic Monte Carlo

- Rates
  - Physics
  - Molecular dynamics
- Must include all rates!
- Used for:
  - Surface diffusion
  - Surface growth
  - Syntering
  - Domain evolution

Example....

Methods

- Molecular Dynamics
  - General
- Event Driven Dynamics
  - Hard objects, at low density
- Contact Dynamics
  - Rigid particles
- Kinetic Monte Carlo
  - Infrequent events, bonded particles

Parallelization

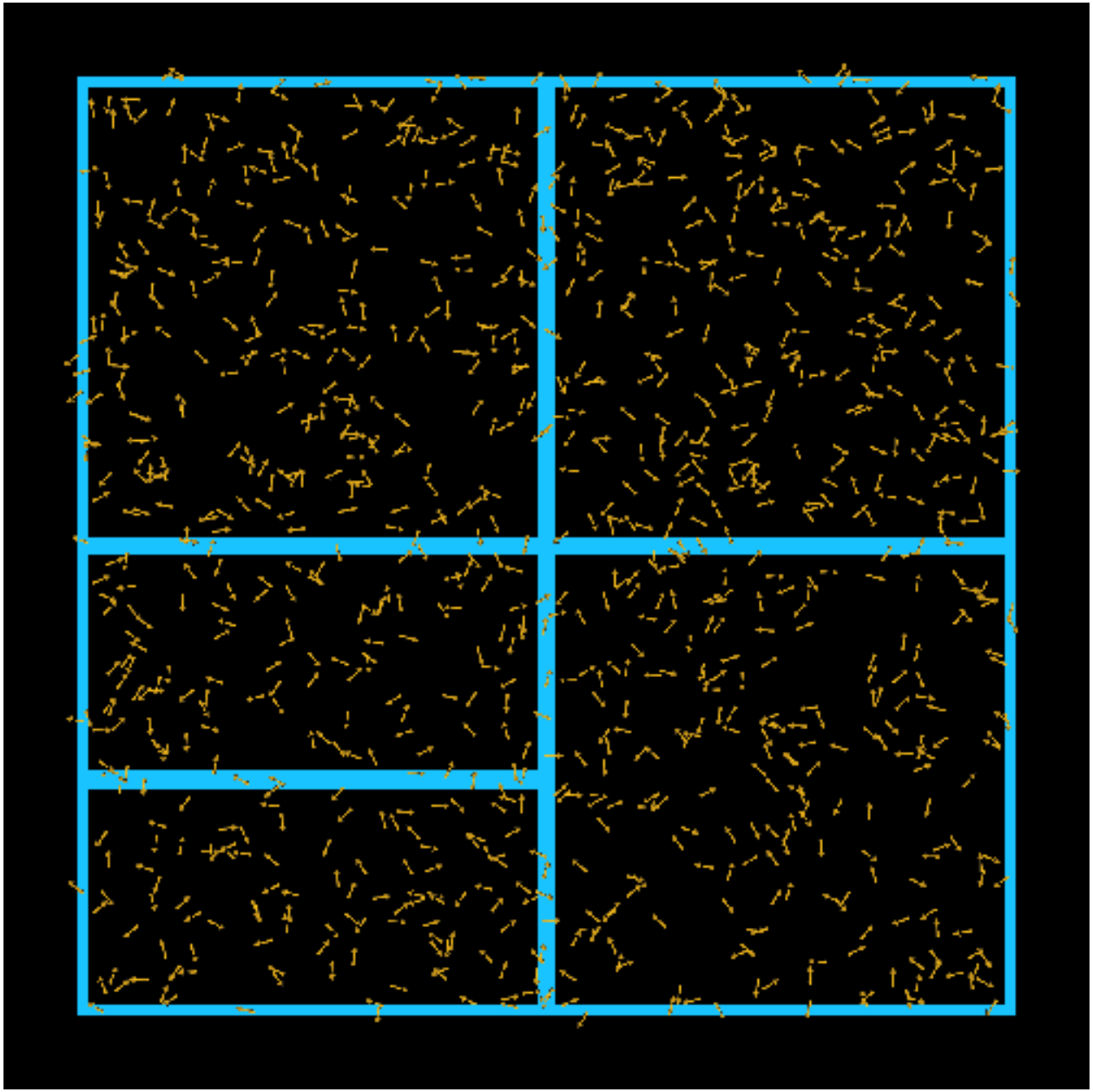
- Why?
  - The speed of one core processor is limited
  - Larger system sizes
  - Multi-core processors
  - On multi-core system inter-processor data change is fast
- Why not?
  - Computing power is lost
  - Much more code development
  - Very often ensemble average is needed
  - Inter-computer communication is terribly slow

RAM $\rightarrow$ $\sim$15GB/s, Ethernet 125MB/s, Infiniband $\sim$1GB/s

Parallelization: How?

- Code asks for more instances (e.g. run a loop in parallel)
  - Fork, multi-threading
  - Used in desktop applications
  - Punished on clusters
  - Shared memory
- Operating system (or even multiple machines) launches the code multiple times which can communicate
  - Now de facto standard: MPI (Message passing interface)
  - Communication is standardized, environment can be inhomogeneus
- GPU:
  - High number of cores
  - Non-standard processors
  - Non-standard libraries
  - Limited memory

Parallelization (Bird flocking model)



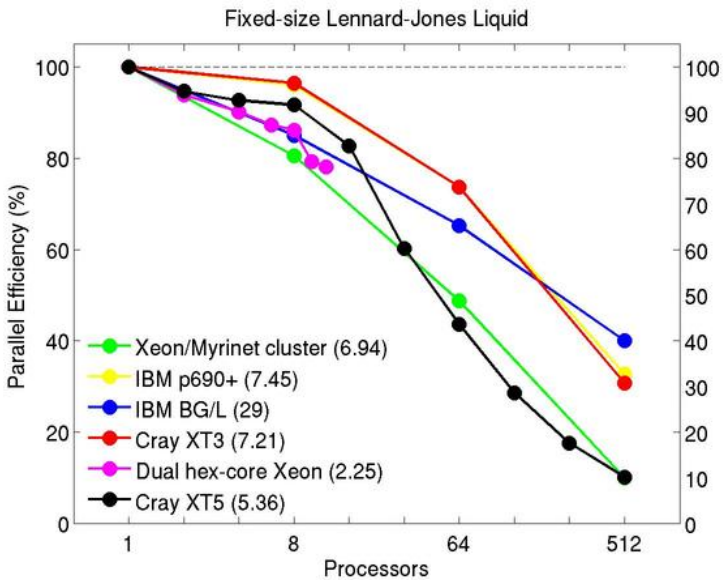
Parallelization

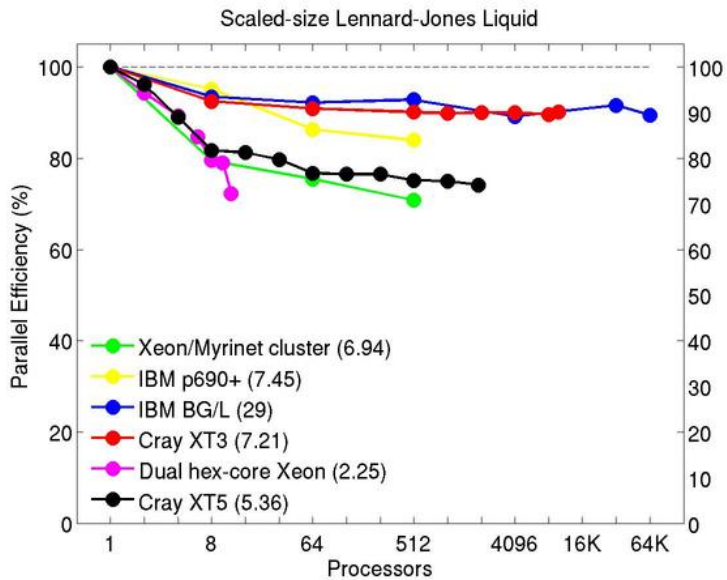
Extra steps needed:

- Molecular dynamics
  - Short range interactions: Box must be duplicated, Verlet in parallel
  - Long range: Parallel fast Fourier transformation
- Contact dynamics
  - Short range interactions: Box must be duplicated, Iteration in parallel
- Event Driven Dynamics
  - List must be global, no way!
- Kinetic Monte Carlo
  - List must be global, no way!

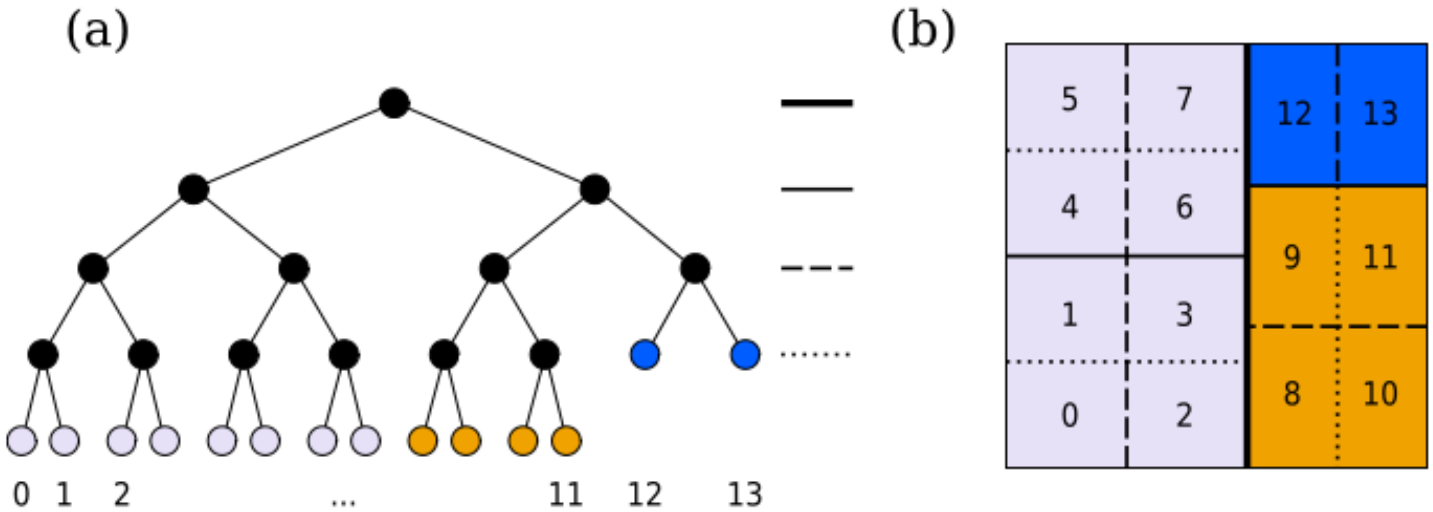
Efficiency of parallelization

- Large systems are needed
- Boundary must be minimal
- System size can be increased simulation time not really





Efficiency of parallelization



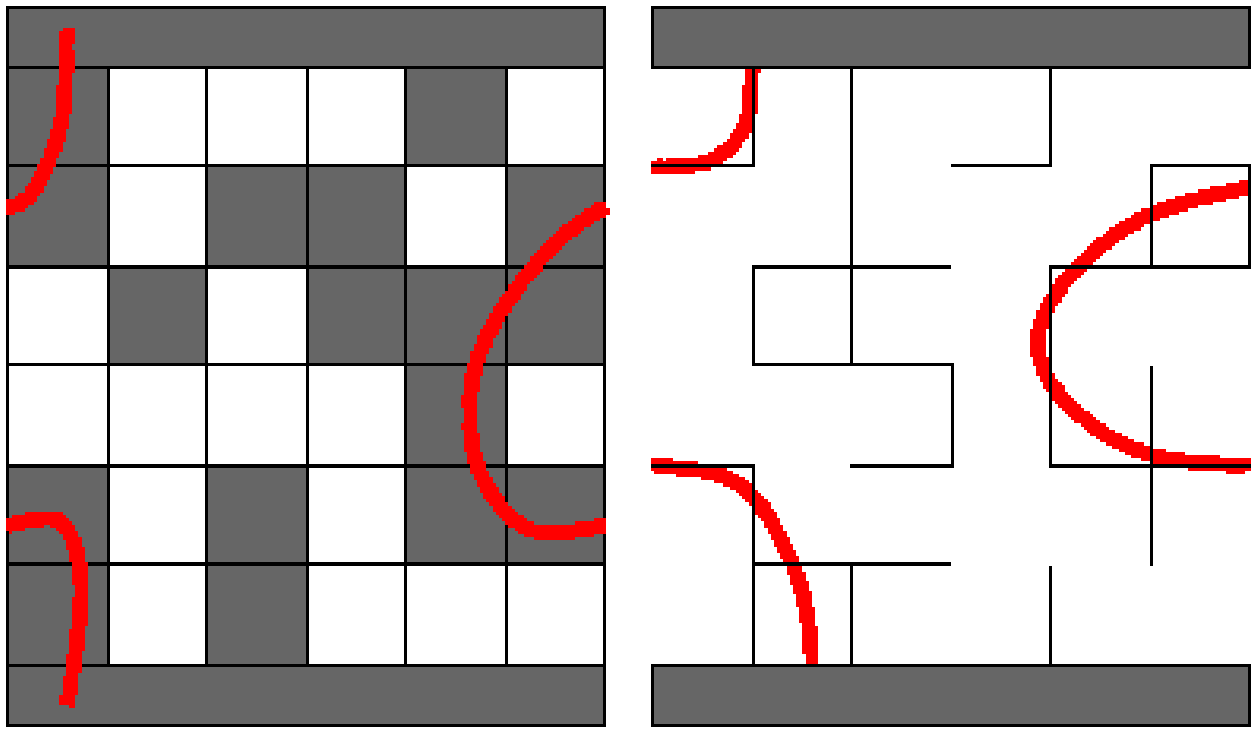
- Calculate time spent in a branch
- Calculate $\sigma_T = \sqrt{\langle T^2 \rangle - \langle T \rangle^2} / \langle T \rangle$
- Move line if necessary ($\sigma_T > \sigma_T^*$)
- Lower in tree (up in Fig), larger the mass of the border
- Only rarely, data transfer is expensive

Percolation

Percolation

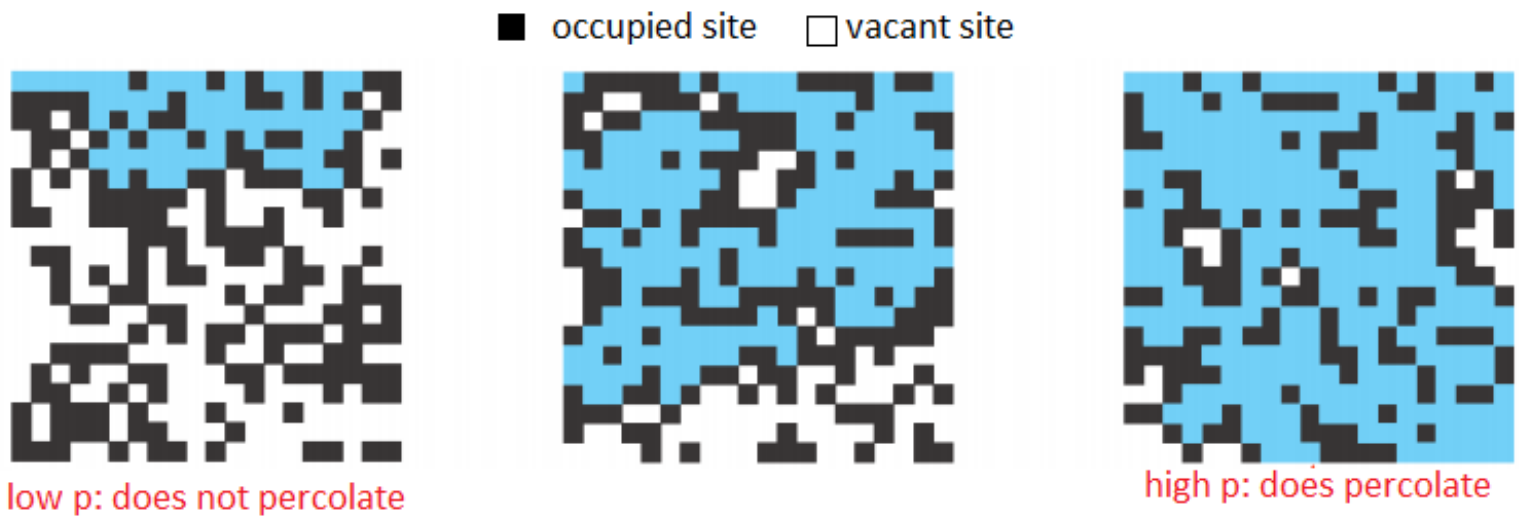
Behavior of connected cluster

- Site percolation
- Bond percolation



Percolation model

- Random environment
- With probability p site vacant (conducts)
- Two states: percolates or not!



Percolation theory

Questions (in infinite systems):

1. Is there an infinite cluster in infinite systems?
2. How many infinite clusters are there?
3. Mean cluster size (without the inifinte one)?
4. Cluster size distribution

Answers:

1. Above a critical density with probability 1 below it with probability 0
2. Only 1!
3. Decreases as a power low away from the critical density
4. Power law

Percolation theory

Questions (in infinite systems):

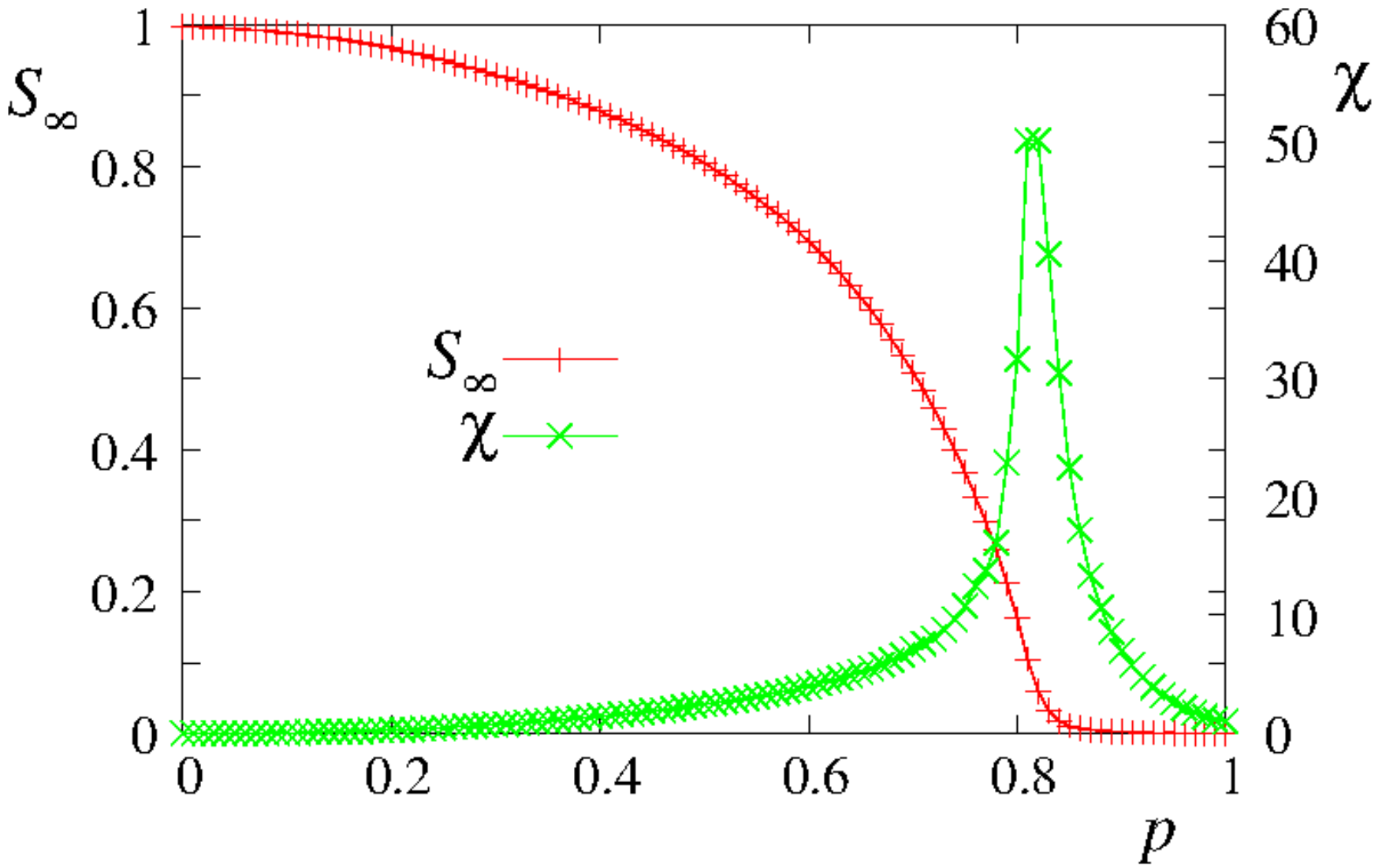
1. Is there an infinite cluster in infinite systems?
2. How many infinite clusters are there?
3. Cluster size distribution (n_s)
4. Mean cluster size (without the inifinte one)? ($S = \sum_s s^2 n_s$)

Answers:

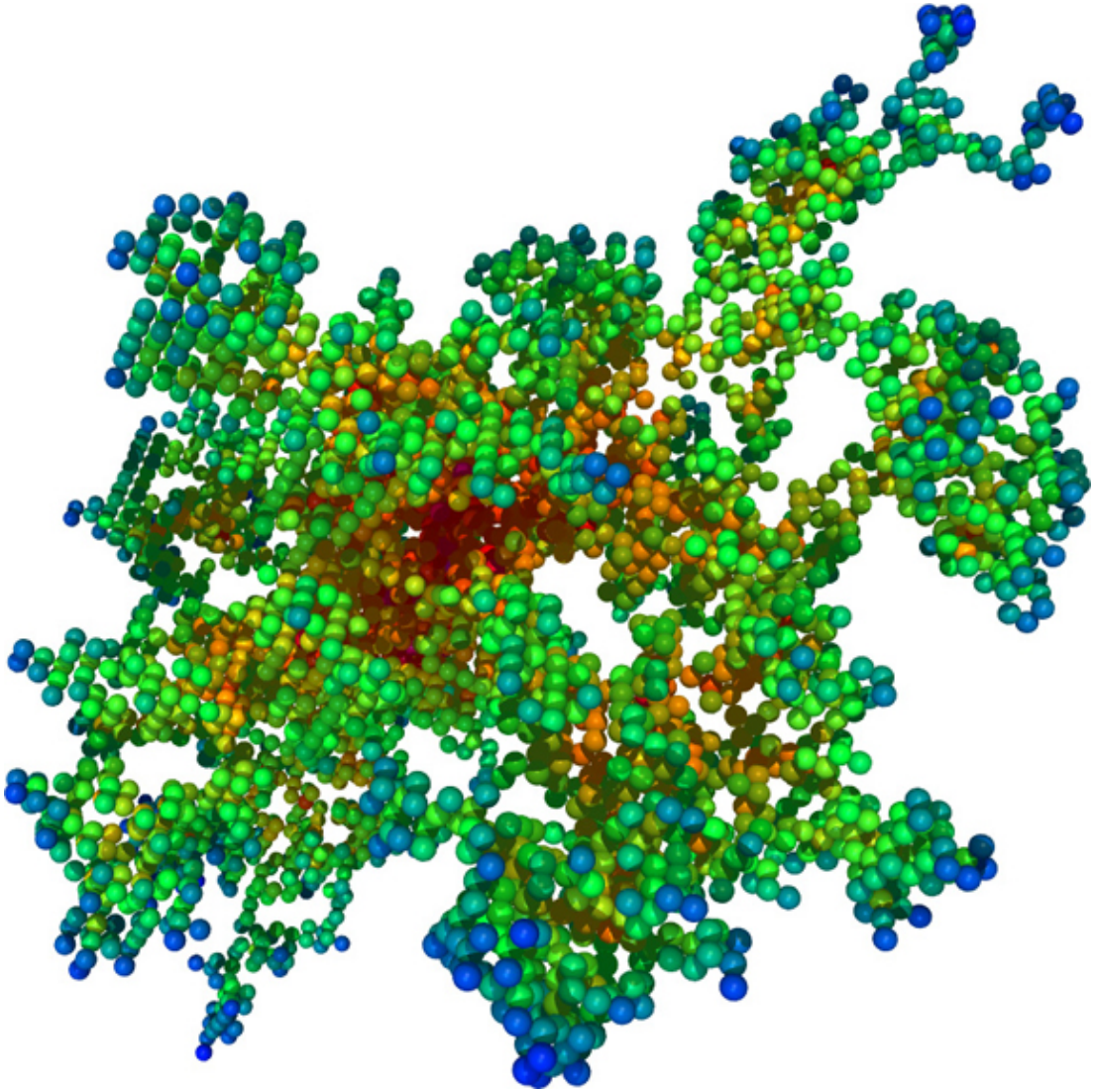
1. if $p > p_c$ then yes, otherwise no
2. Only 1!
3. $n_s \sim s^{-\tau}$
4. $S \sim |p - p_c|^{-\gamma}$

Like a second order phase transition in a geometric system!

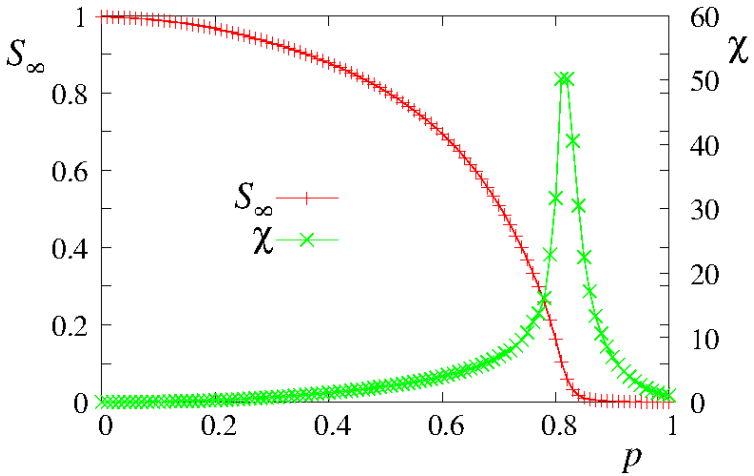
Percolation model

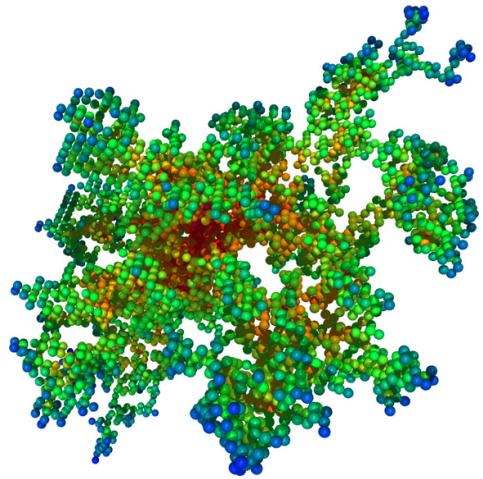


Percolation model



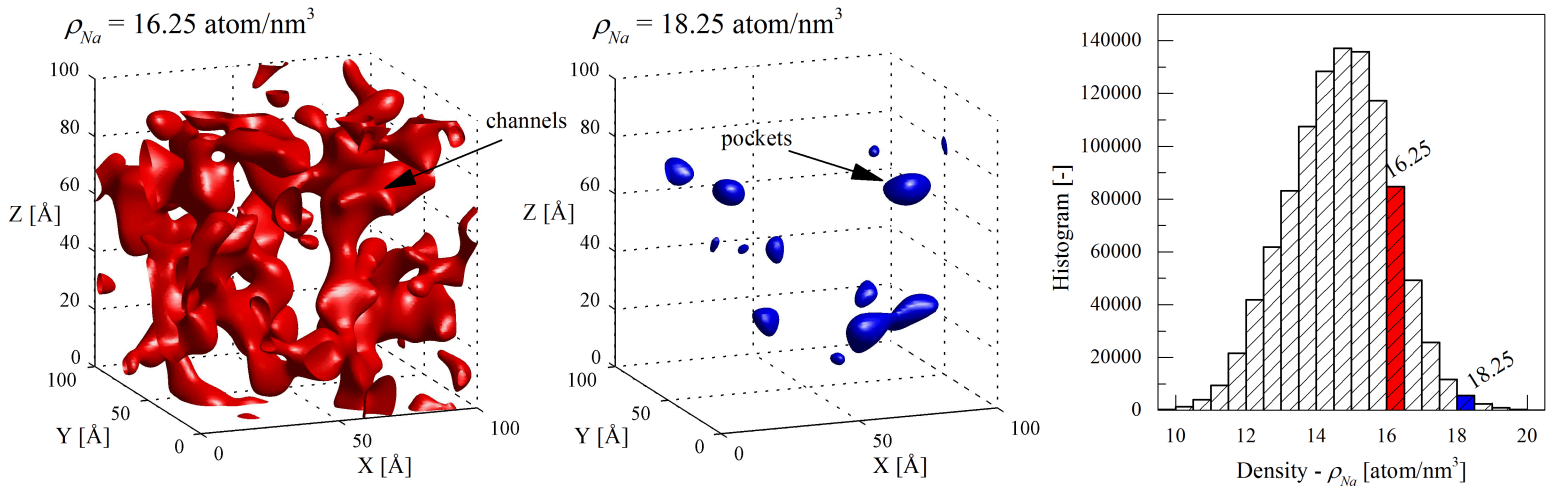
Percolating cluster

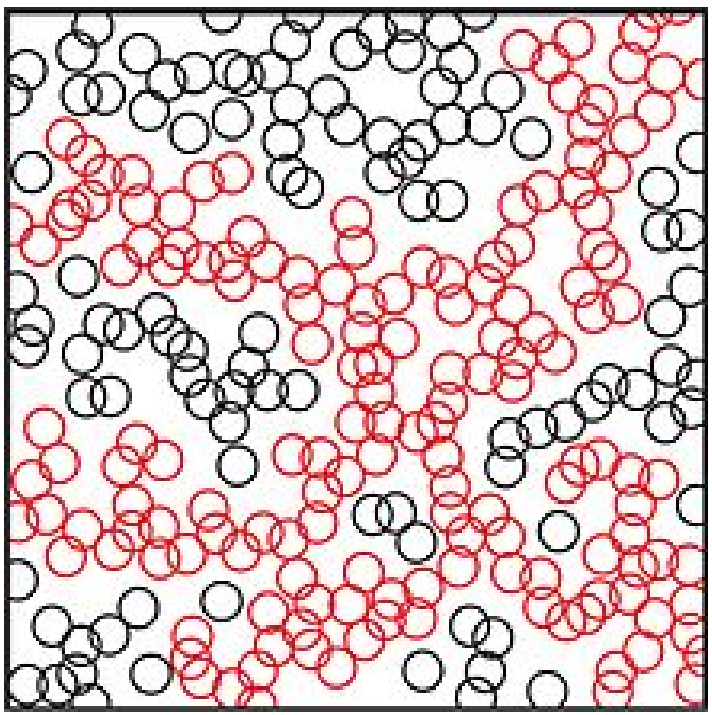


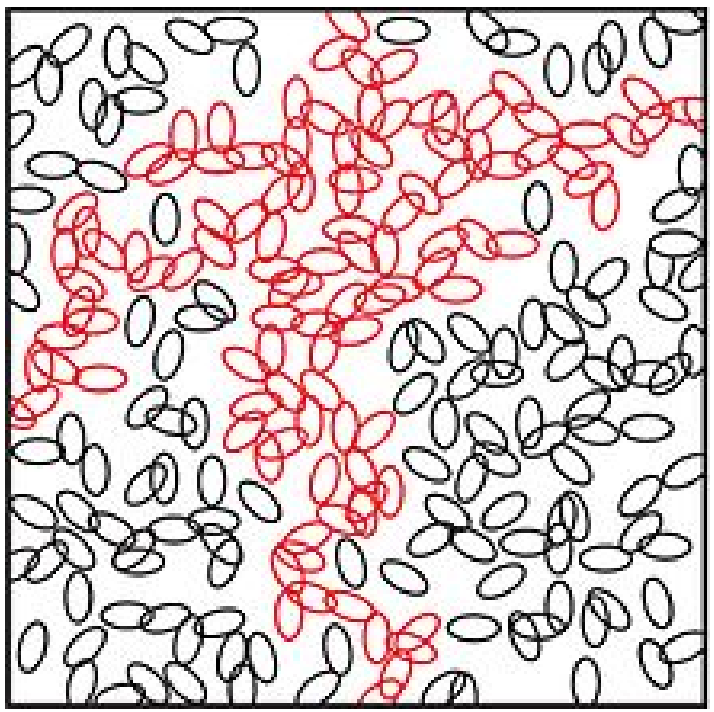


- Largest cluster
  - fractal with fractal dimension of d_f
  - $S_\infty \sim \begin{cases} \xi_f^d \log(N/\xi^d) & p < p_c \\ N^{d_f/d} & p = p_c \\ NP_\infty(p) & p > p_c \end{cases}$
- Largest not infinite cluster: size $\sim |p - p_c|^{-\nu}$

Percolation theory: Importance







Percolation theory: Importance

- COFFEE!!!!
- Non-equilibrium statistical physics
- Image analysis
- Percolation on networks: Phase transitions
- Percolation on networks: robustness, fragility
- Flodings

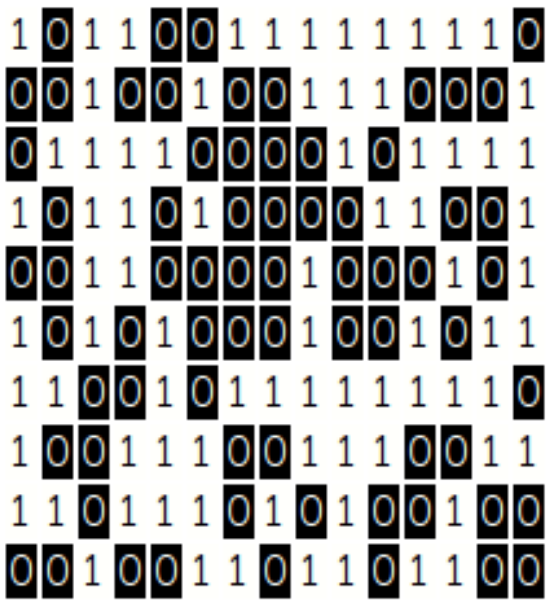
Percolation model

Bond [site] percolation

- Let us have a lattice (network)
- Each bond [site] is occupied with probability p
- (unoccupied with probability $1 - p$)
- A cluster is a set of sites connected by occupied bonds
  [A cluster is a set of occupied sites]

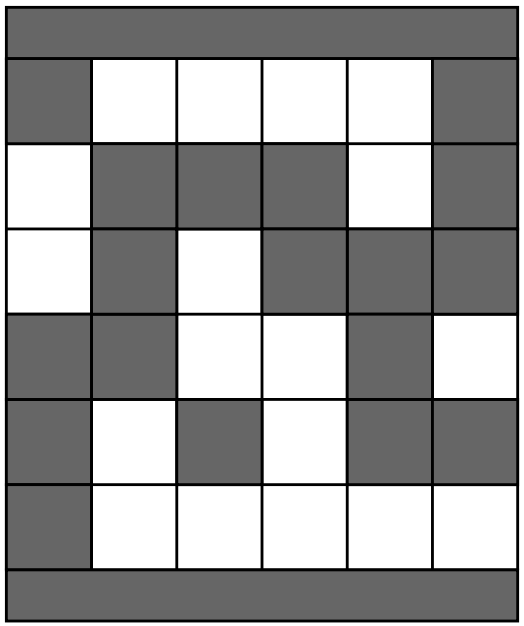
Hoshen-Kopelman Algorithm

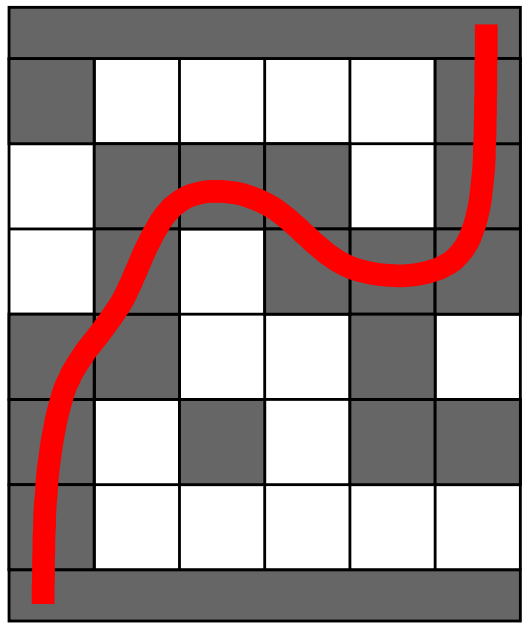
- Numerical task: find clusters
- Identify clusters
- Visit all sites
- Mark them with numbers



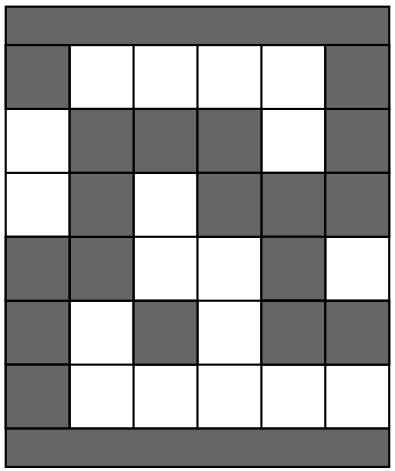
Hoshen-Kopelman Algorithm

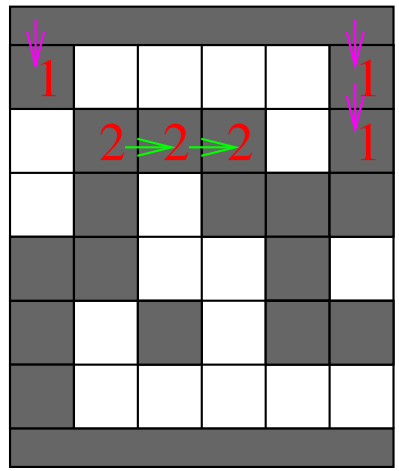
- Site percolation
- Open boundary conditions
- Go through site in typewriter style
- Check left and above





Hoshen-Kopelman Algorithm

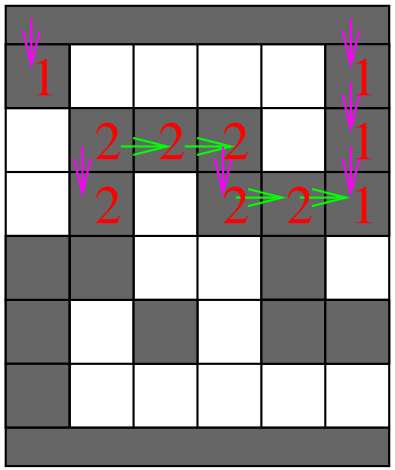


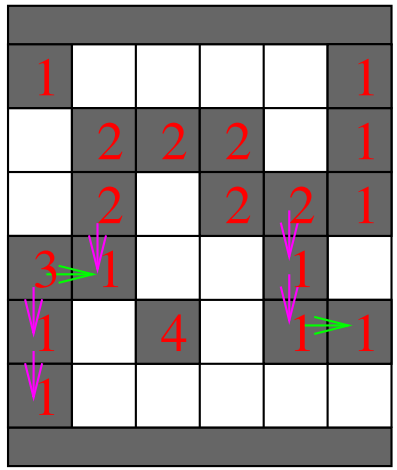


link[1]=1
link[2]=2

- Go through sample in typewriter style
- If site is occupied, look left and up
  - if no neighbour $\rightarrow$ new number
  - if only one is occupied $\rightarrow$ inherit number

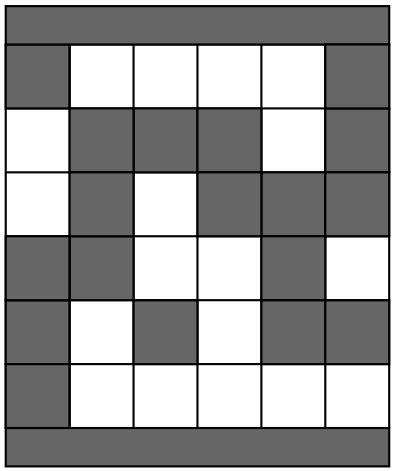
Hoshen-Kopelman Algorithm

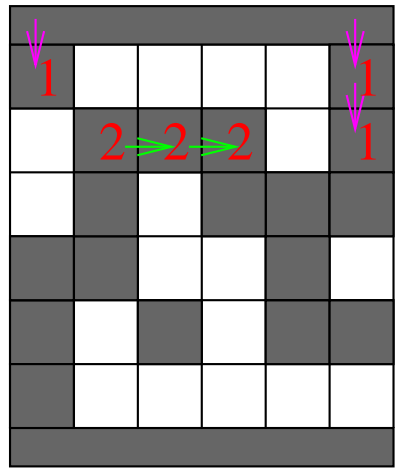


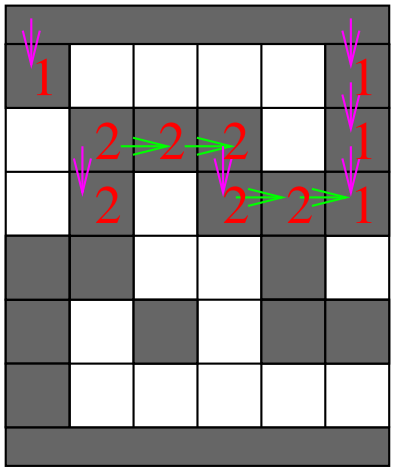


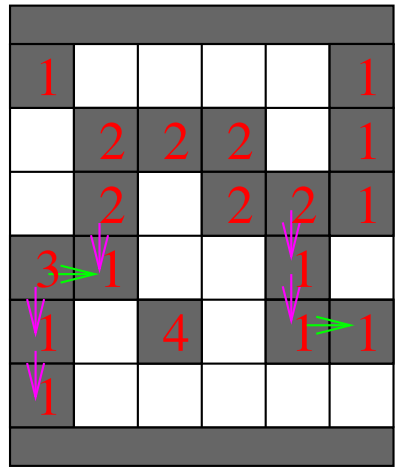
- ...
- If site is occupied, look left and up
  - ...
  - if both sites are occupied $\rightarrow$ then link the two
    - a link which points to itself (`link[i]=i`) is a root link
    - find the root of both sites (`link[i]=j;link[j]=k;...;link[l]=m;link[m]=m` $\rightarrow$ `root[i]=m`)
    - connect the one with larger root larger to the smaller (`root[i]>root[j]` $\rightarrow$ `link[root[i]]=root[j]`)
    - use this number to mark the site

Hoshen-Kopelman Algorithm



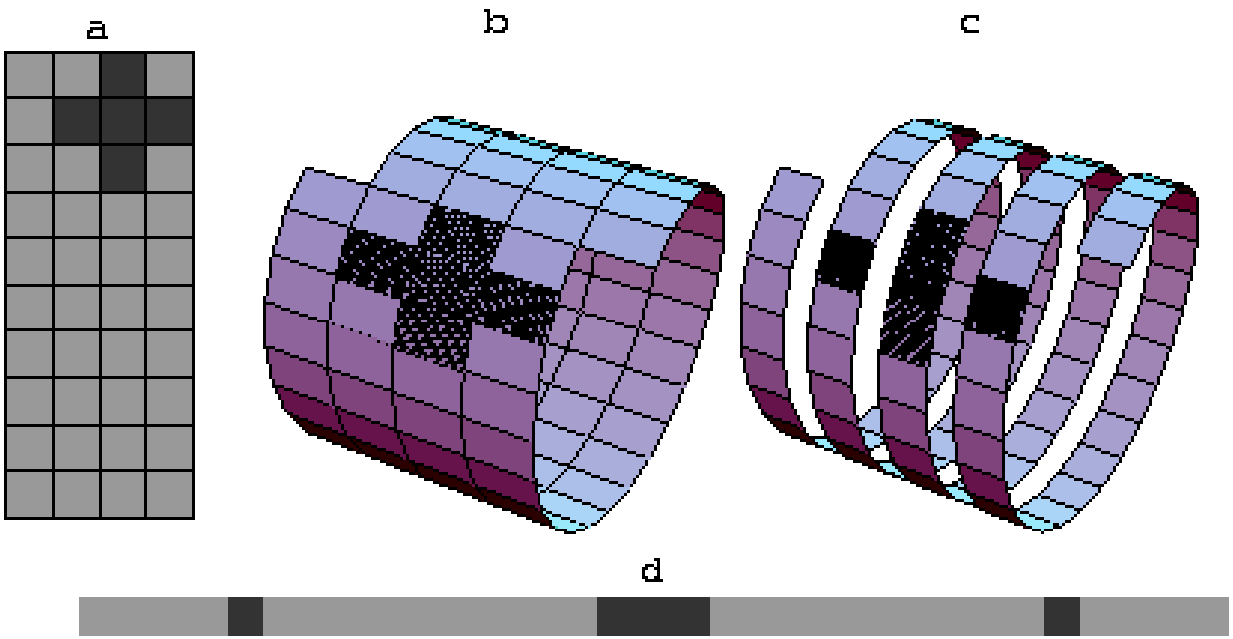






Hoshen-Kopelman Algorithm

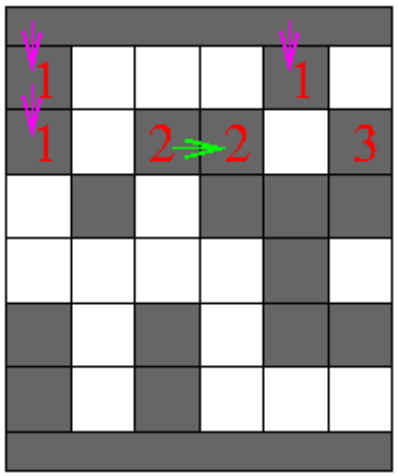
- Site percolation
- Helical boundary conditions (rolled up ont dimensional lattice)
- Go through site in typewriter style
- Check left and above (as before)



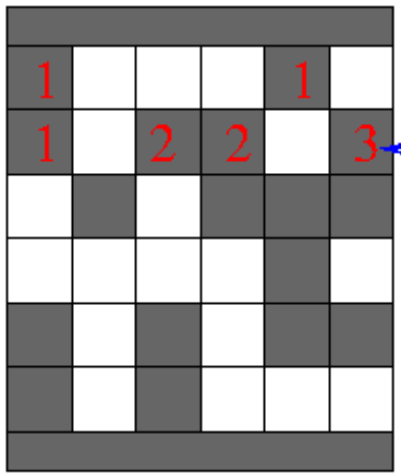
Hoshen-Kopelman Algorithm

- Site percolation
- Periodic boundary conditions
- Go through site in typewriter style
- Check left and above (as before)
- After each line if first and last site is occupied link them

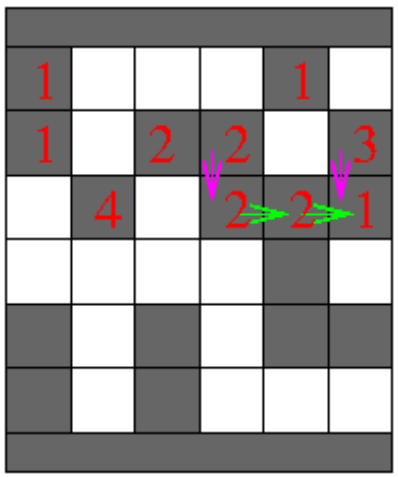
Hoshen-Kopelman Algorithm, Periodic BC



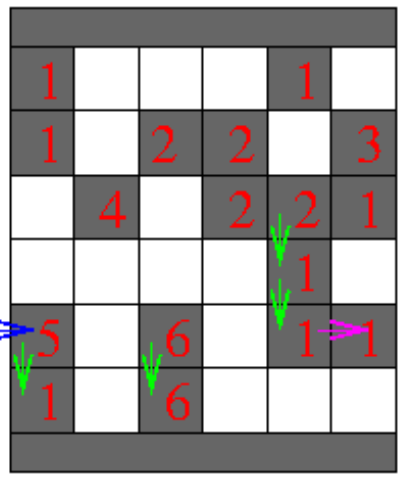
link[1]=1
link[2]=2
link[3]=3



link[1]=1
link[2]=2
link[3]=1

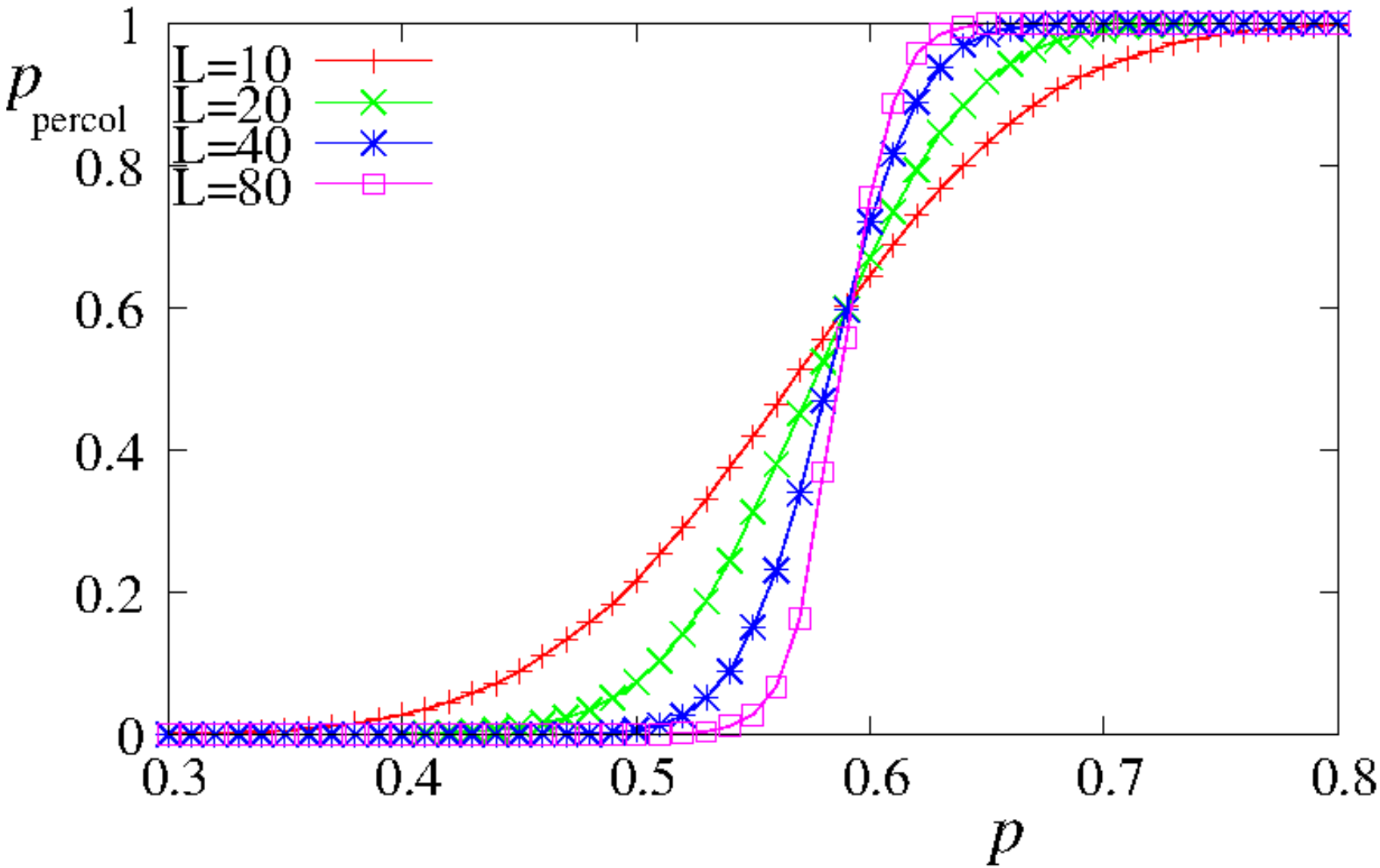


link[1]=1
link[2]=1
link[3]=1
link[4]=4



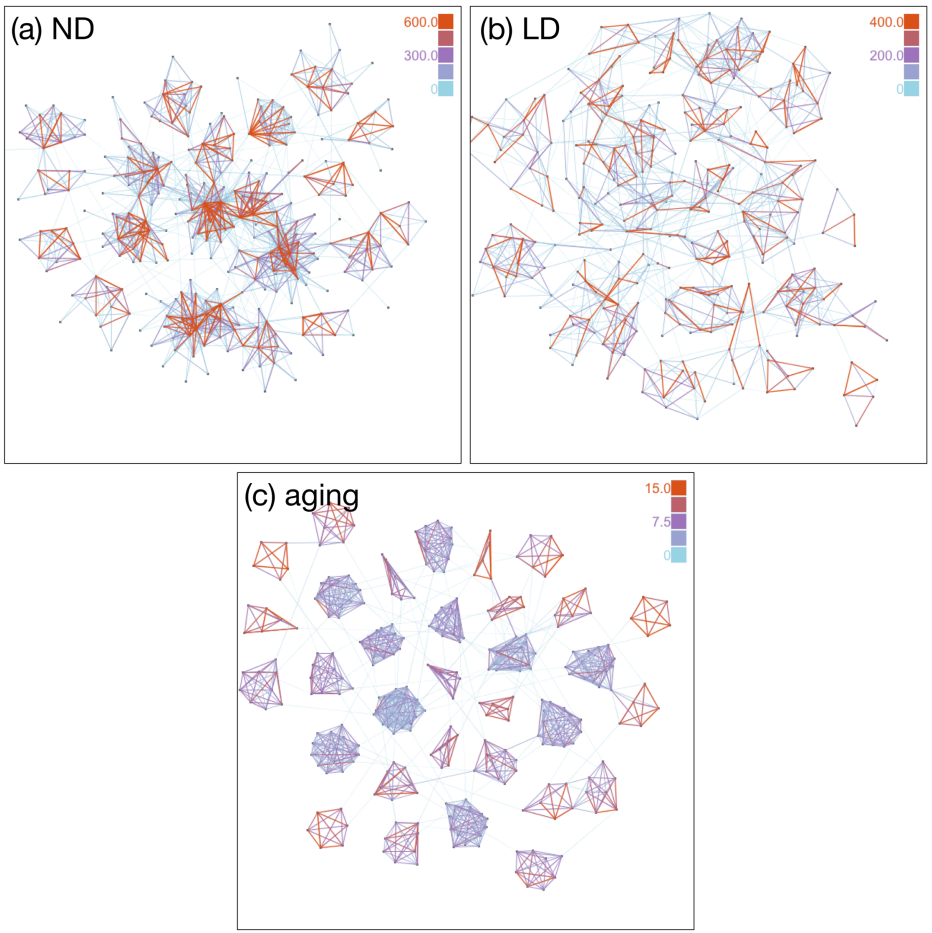
link[1]=1
link[2]=1
link[3]=1
link[4]=4
link[5]=1
link[6]=6

Result

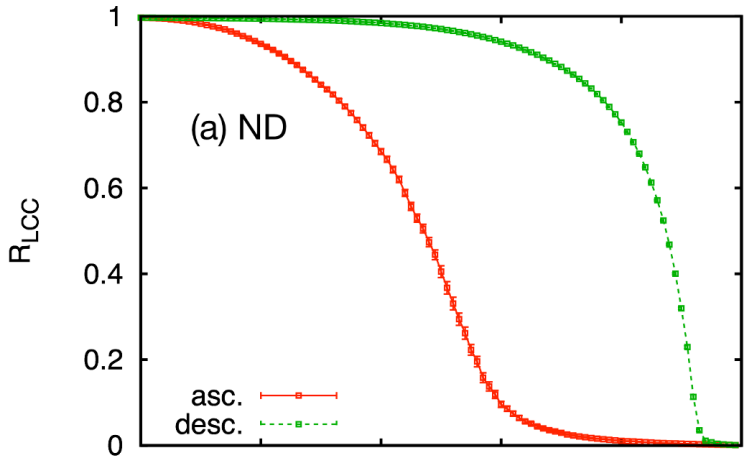


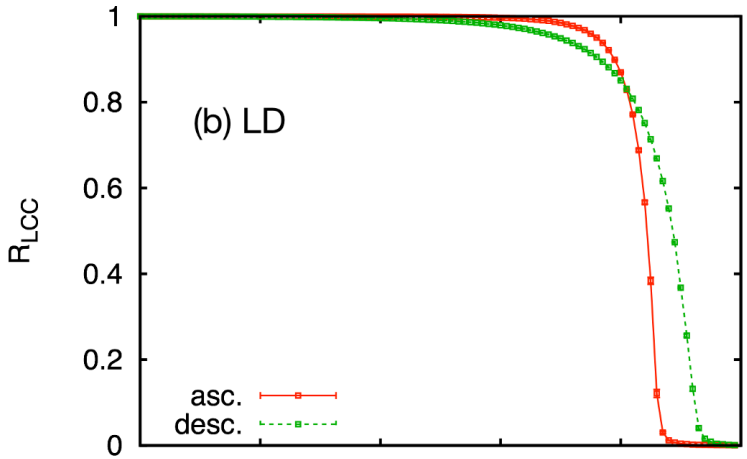
Link removal percolation on networks

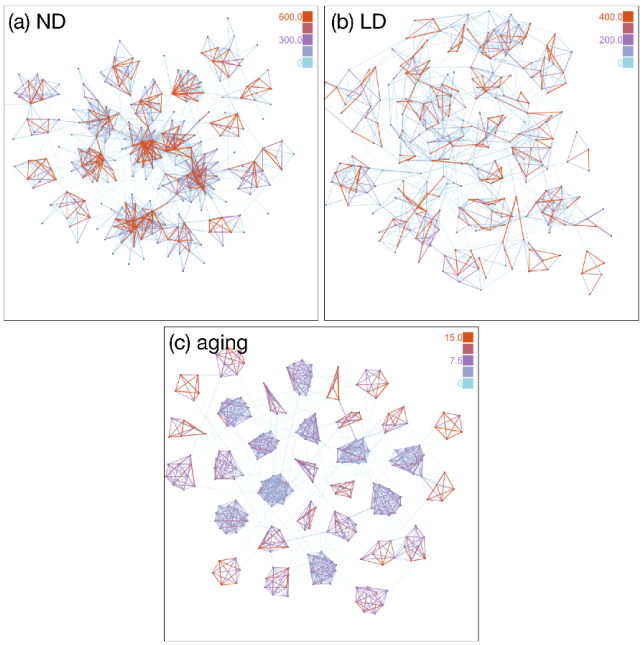
- Granovetter hypothesis: The strength of the weak ties
- Human communities have strong connections
- These communities are connected with weak ties
- Test the structures with Link removal percolation

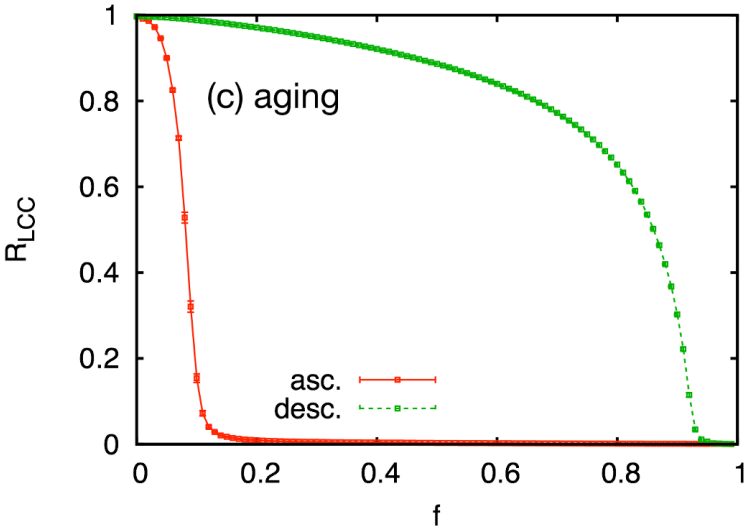


Link removal percolation on networks









Percolation on networks (graphs)

- Network is defined by nodes and links
- Two arrays:
  - `node[]` list of nodes
  - `link[i][]` list of links of node `i`
  - `link[i][j]` is a link between `i` and `link[i][j]`
- Cluster: nodes connected with links
- Links can be directed `link[i][j]` is a link from `i` $\rightarrow$ `link[i][j]`

Stack (Verem – Hole/Pitfall)

- Last in forst out (LIFO)
- Code:

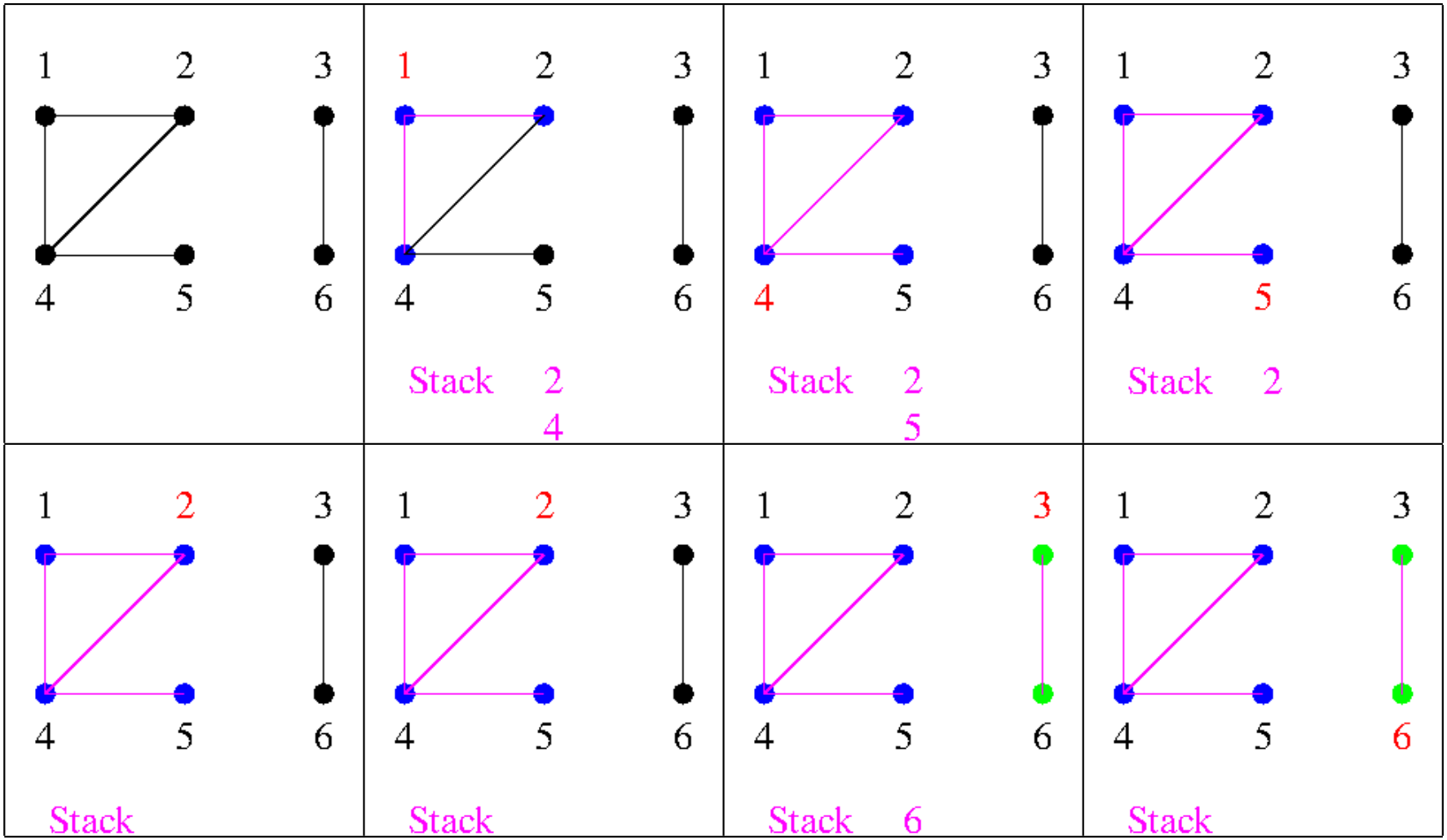
```
int Stack_size = Hopefully_large_enough_number;
int stack[Stack_size];
int sp=0;

void push(int item) {
  stack[sp++] = item;
  if (sp == Stack_size) enlarge_array(stack);
}

int pop() {
  return(stack[--sp]);
}
```

- Error handling?
- Size of the stack?

Algorithm percolation on networks (graphs)

1. Go through each node
2. If the node is unmarked, mark it with a new code and put it in the stack
3. Inner loop starts here: Get a node from the stack
4. Go through each unmarked link of the node
5. Put other end of links in the stack if it is not marked
6. Mark unmarked nodes with the code of the original node
7. If the stack not empty Go to 3.
8. If the stack empty Go to 1.

Algorithm percolation on networks (graphs)



Algorithm percolation on graphs

1. This algorithm can be used on any lattices
2. Mode general
3. There are better algorithms
4. Many new versions

Practice

1. Go through each node
2. If the node is unmarked, mark it with a new code and put it in the stack
3. Inner loop starts here: Get a node from the stack
4. Go through each unmarked link of the node
5. Put other end of links in the stack if it is not marked
6. Mark unmarked nodes with the code of the original node
7. If the stack not empty Go to 3.
8. If the stack empty Go to 1.

Data structure:

- `link[a][i] = b` $\rightarrow$ node a has connection to node b
- `nlink[a]` (only C) number of links node a has. Others (c++, python) check the length of the array.

Advanced: Get the critical point for random graph with average degree $\langle k \rangle$