Algorithms in Percolation

Problem: how to identify and measure cluster size distribution

Single-Cluster growth
"Leath-Alexandrowicz method"

Paul Leath
Rutgers University

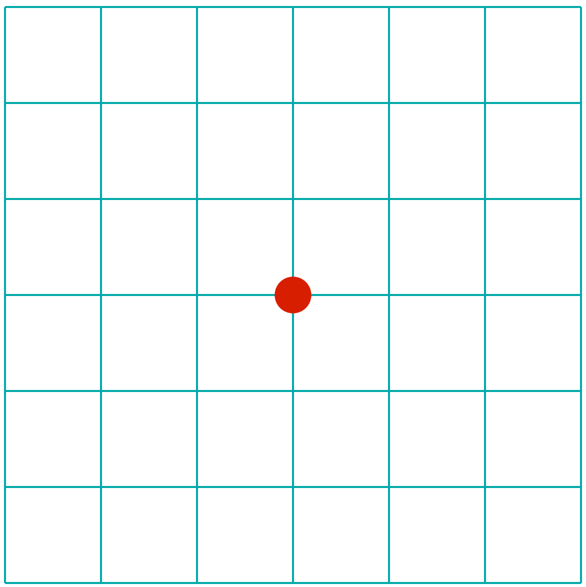
P. L. Leath, Phys. Rev. B **14**, 5046 (1976)

Z. Alexandrowicz, Phys. Lett. A 80, 284 (1980).

Leath-Alexandrowicz Algorithm

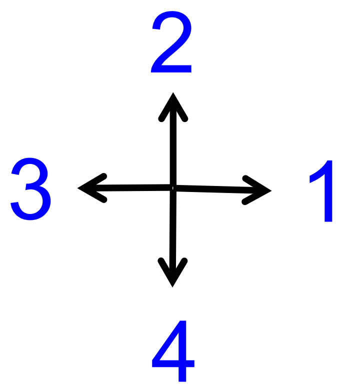
- "Grow" clusters by adding sites one at a time from an initial seed
- Two methods:
  - "Breadth first" or "First-In-First-Out" (FIFO)
    - Requires making a list or queue
  - "Depth first" or "Last-In-Last-Out" (LIFO)
    - Can be done using stack and recursion

FIFO site percolation

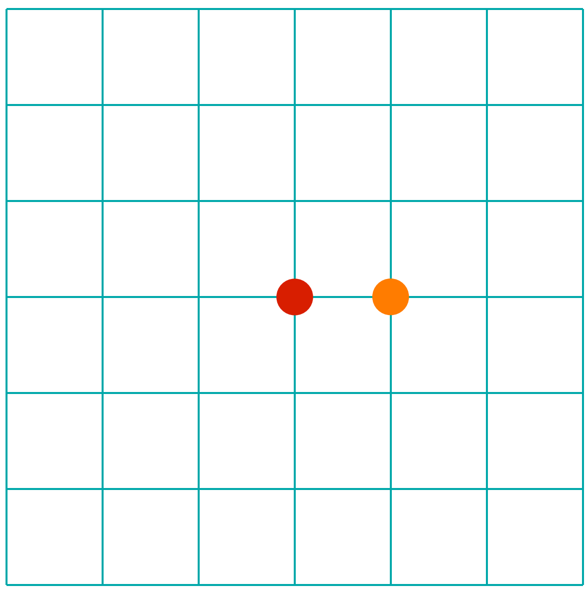


Start with a seed site that is “wet”

Check neighbors in this order:



FIFO site percolation

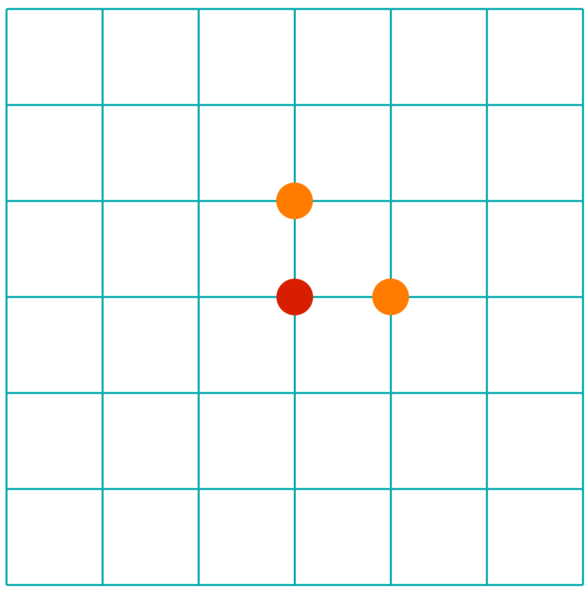


Occupy a site with probability p

Orange = first shell

Unchecked sites queue:

FIFO site percolation

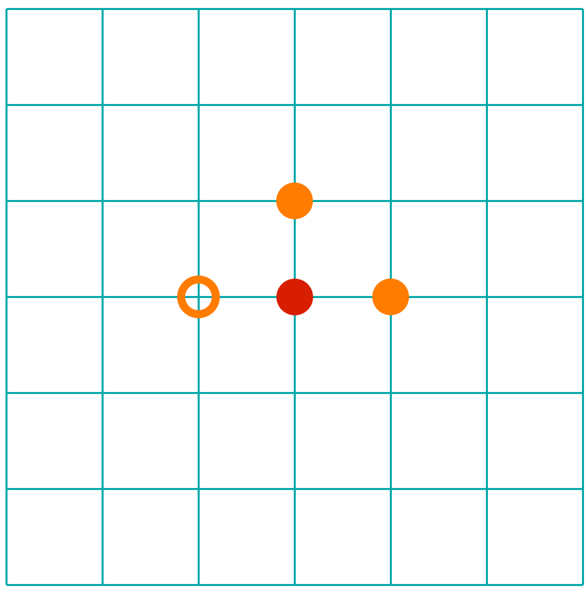


Occupy a site with probability p

Orange = first shell

Unchecked sites queue:

FIFO site percolation

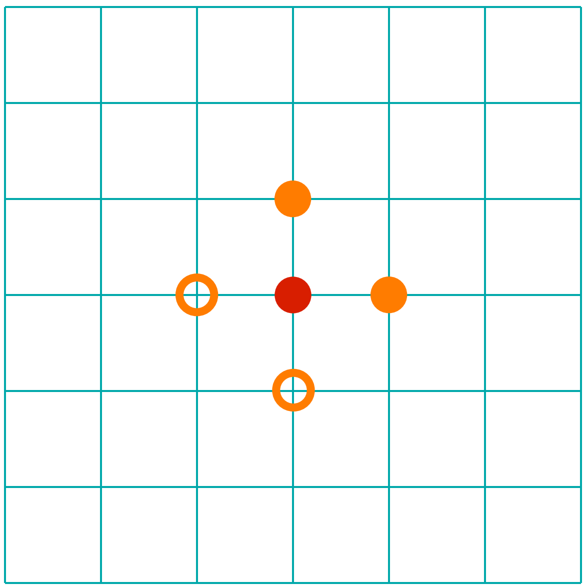


Make site “vacant”
with probability 1-p

Orange = first shell

Unchecked sites queue:

FIFO site percolation

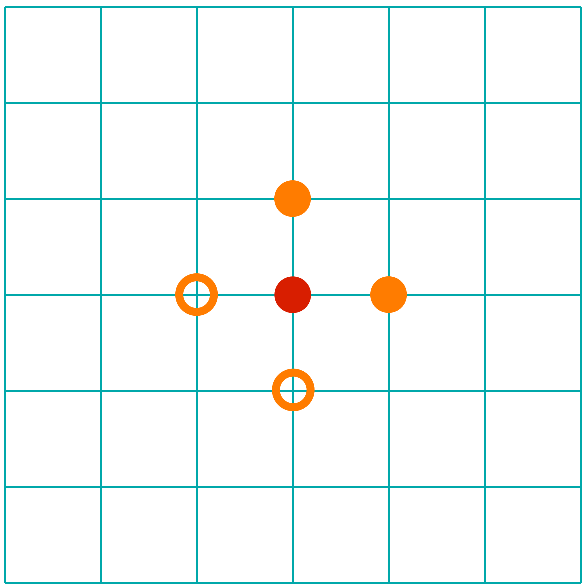


Make site “vacant” with probability 1-p

Orange = first shell

Unchecked sites queue:

FIFO site percolation

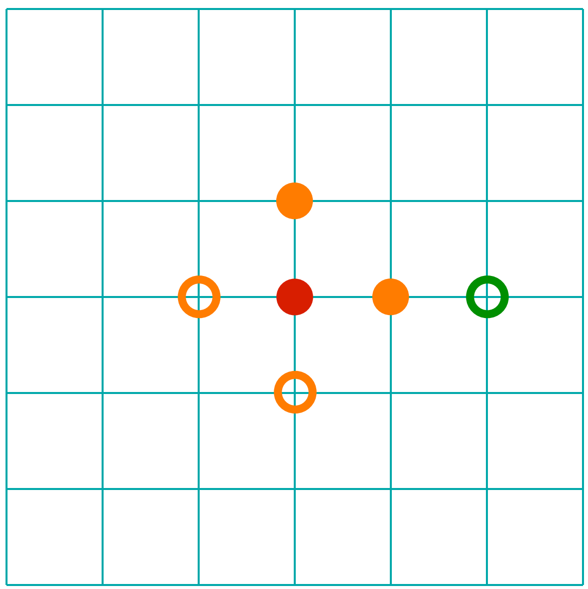


Make site “vacant” with probability 1-p

Orange = first shell

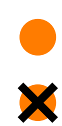
Unchecked sites queue:

FIFO site percolation

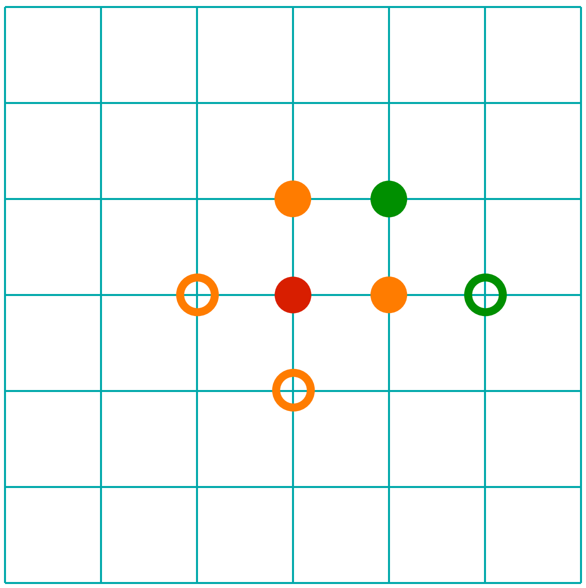


Green = second shell

Unchecked sites queue:



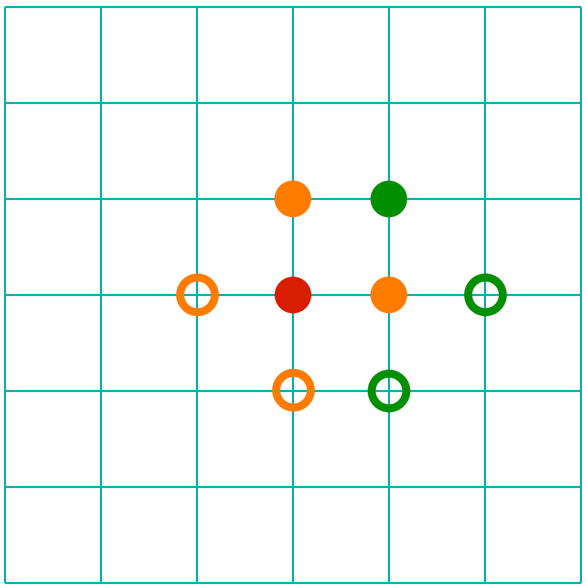
FIFO site percolation



Green = second shell

Unchecked sites queue:

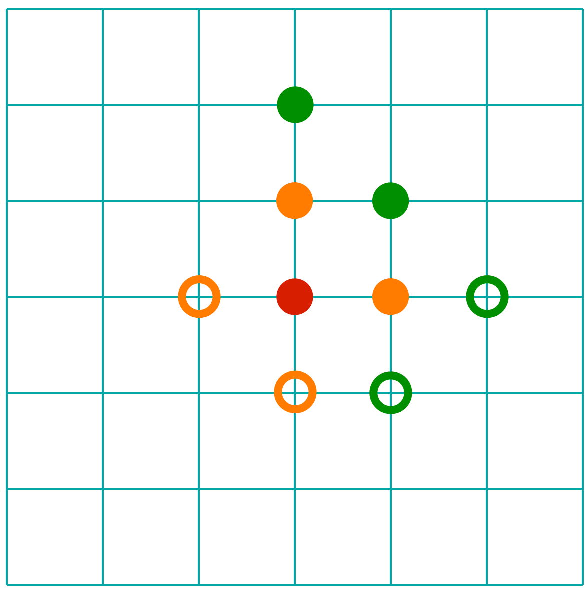
FIFO site percolation



Green = second shell

Unchecked sites queue:

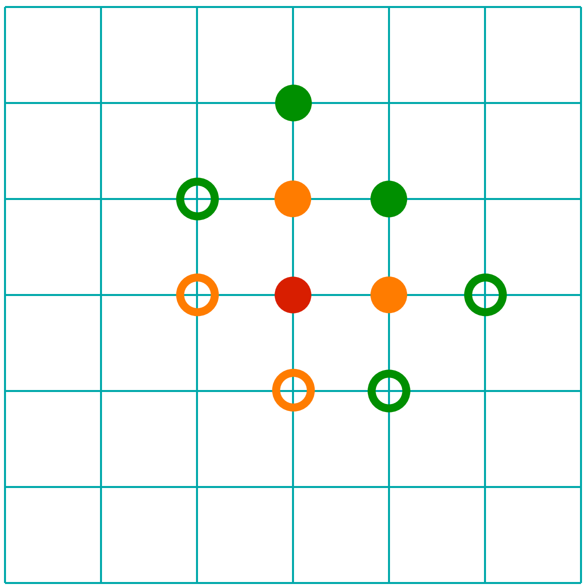
FIFO site percolation



Green = second shell

Unchecked sites queue:

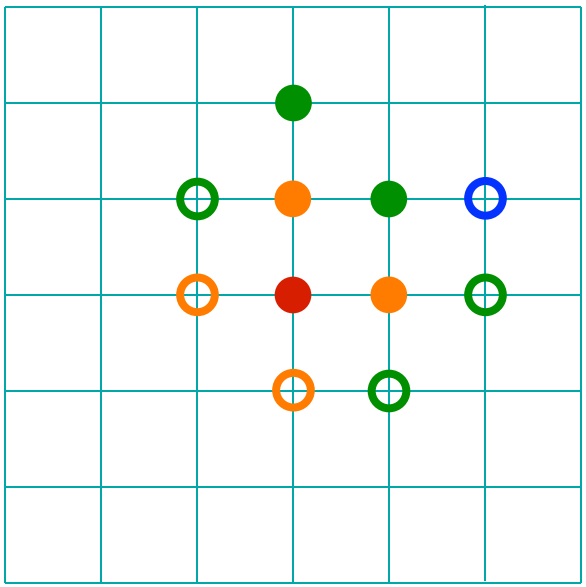
FIFO site percolation



Green = second shell

Unchecked sites queue:

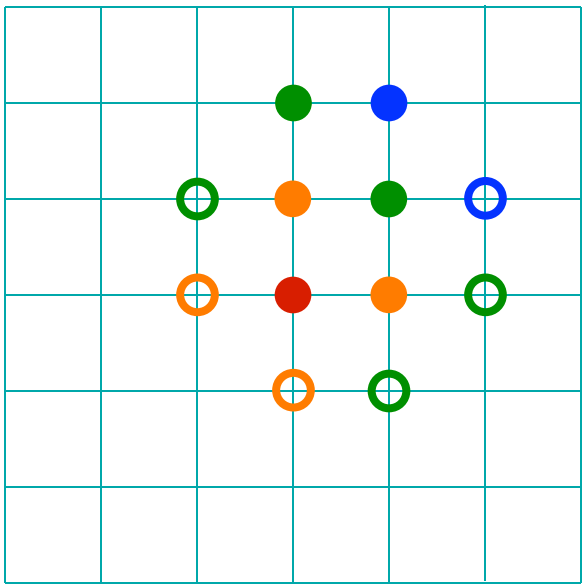
FIFO site percolation



Blue = third shell

Unchecked sites queue:

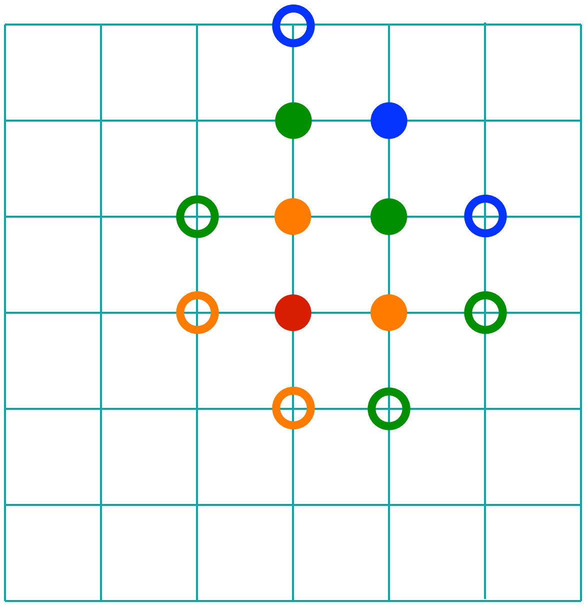
FIFO site percolation



Blue = third shell

Unchecked sites queue:

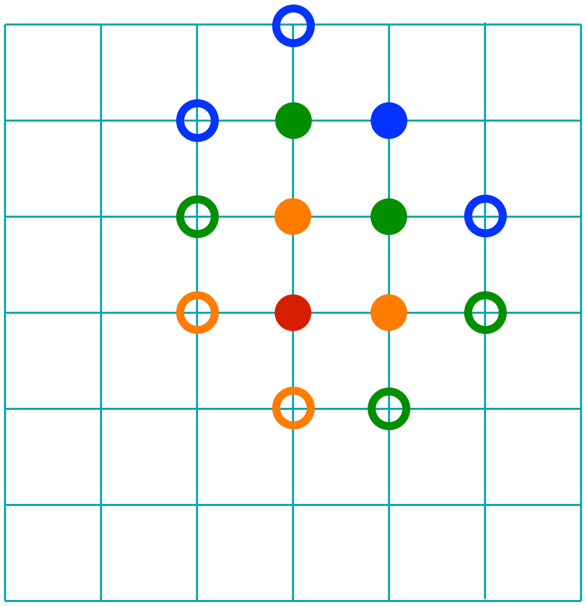
FIFO site percolation



Blue = third shell

Unchecked sites queue:

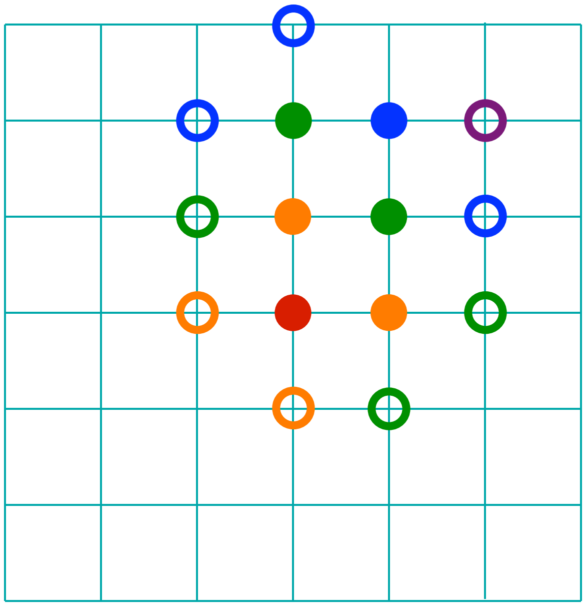
FIFO site percolation



Blue = third shell

Unchecked sites queue:

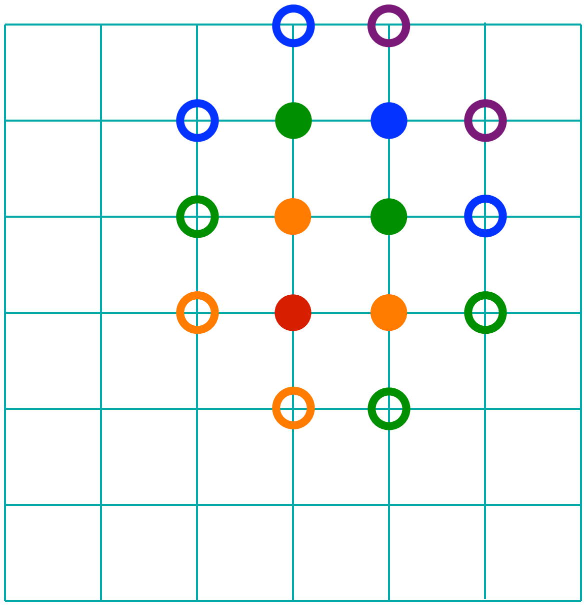
FIFO site percolation



Violet = fourth shell

Unchecked sites queue:

FIFO site percolation



Violet = fourth shell

Unchecked sites queue:

FIFO algorithm

“Pop” new growth site from queue

For (neighbors = 1 to 4)

if (neighbor == unvisited)

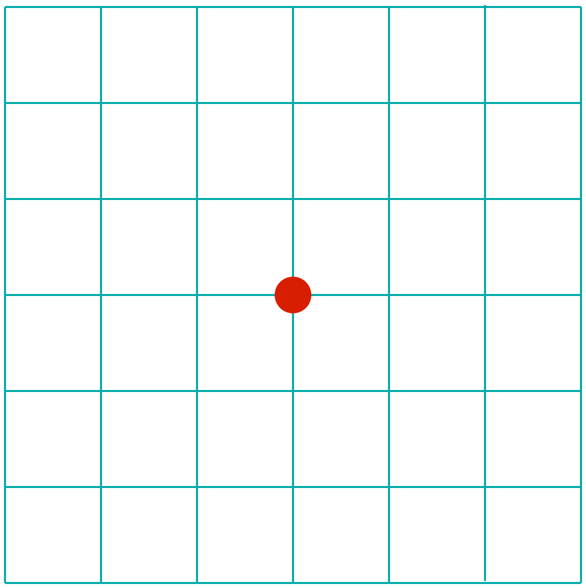
if (randomnumber < prob)

neighbor = occupied

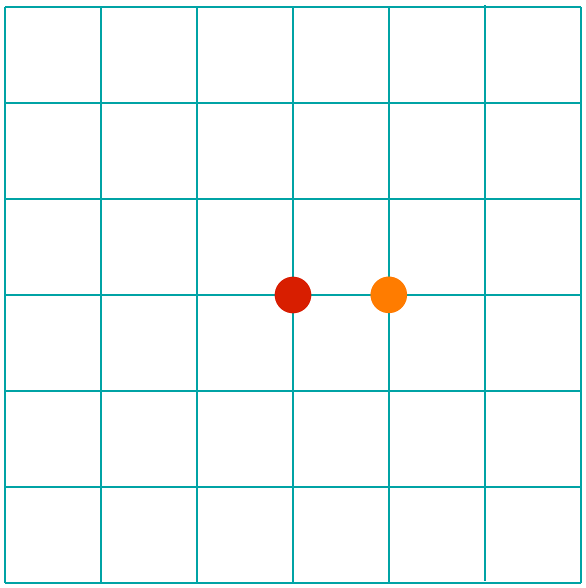
“push” neighbor on queue

else neighbor = vacant.

LIFO site percolation

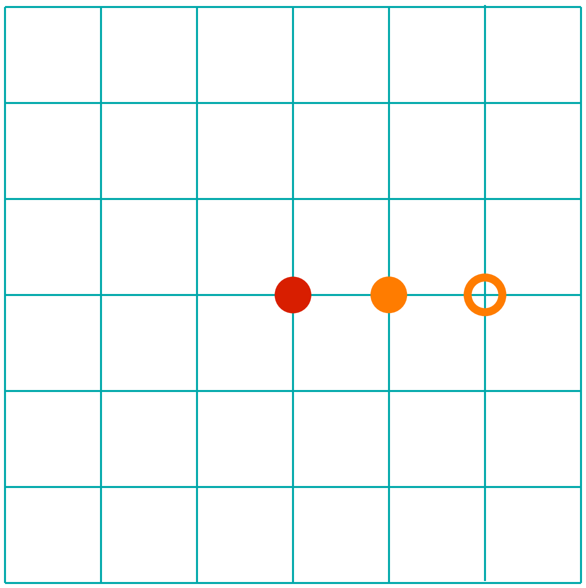


LIFO site percolation



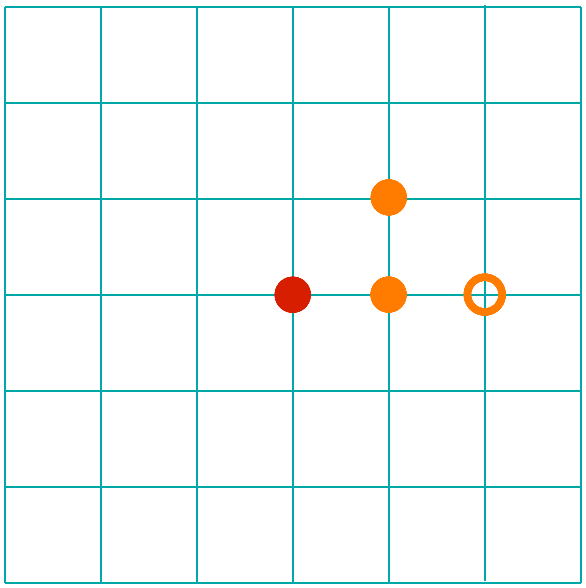
Orange = growth from first neighbor of seed

LIFO site percolation



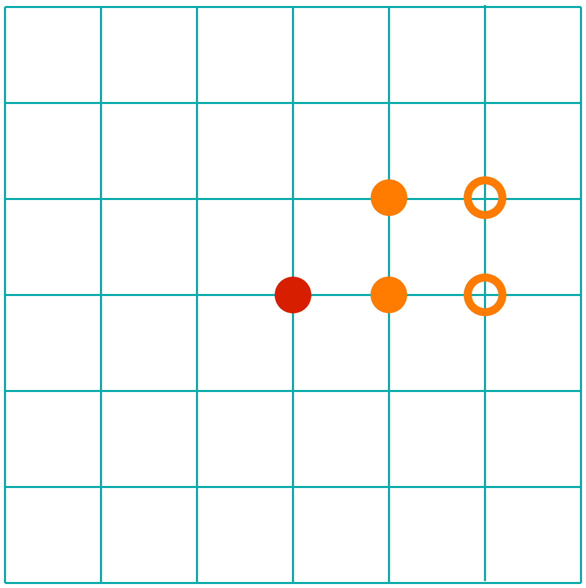
Orange = growth from first neighbor of seed

LIFO site percolation



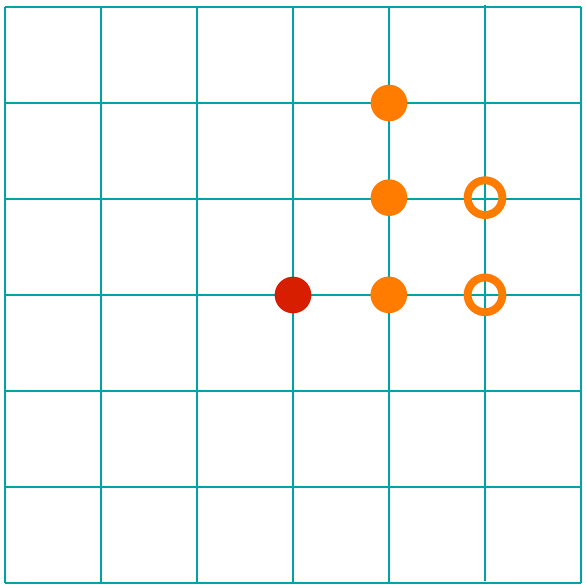
Orange = growth from first neighbor of seed

LIFO site percolation



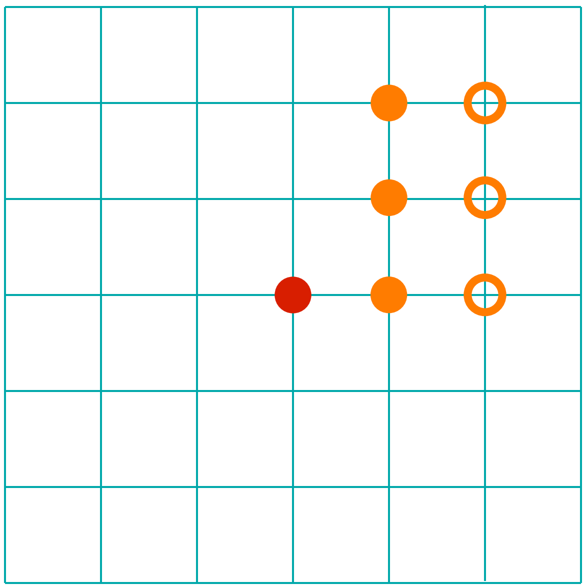
Orange = growth from first neighbor of seed

LIFO site percolation



Orange = growth from first neighbor of seed

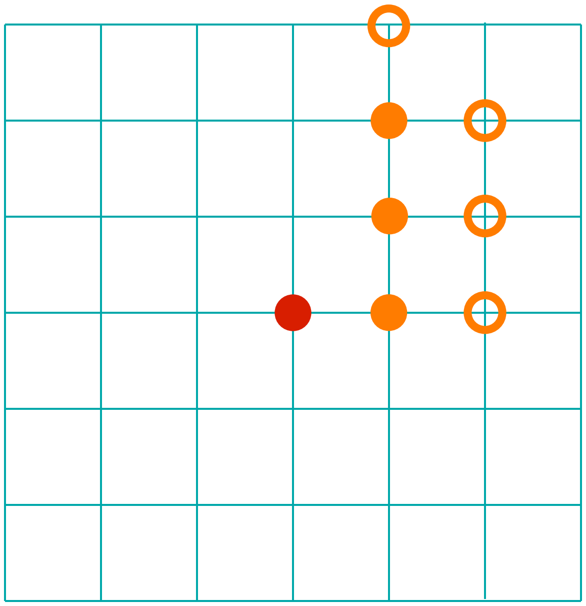
LIFO site percolation



Orange = growth from first neighbor of seed

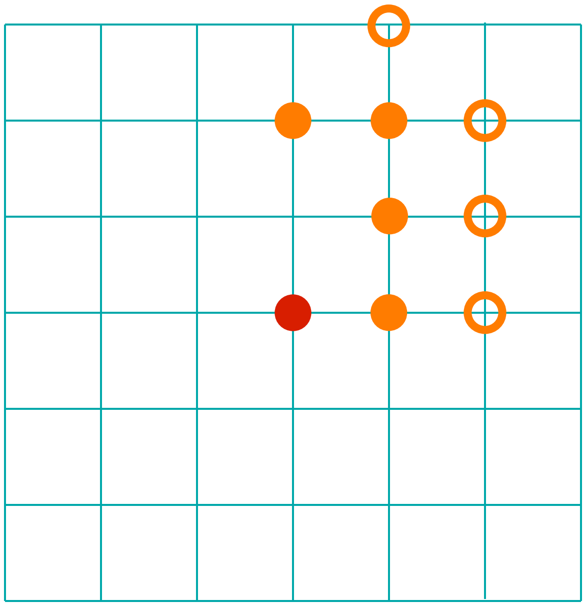
Put all growing sites on a stack

LIFO site percolation



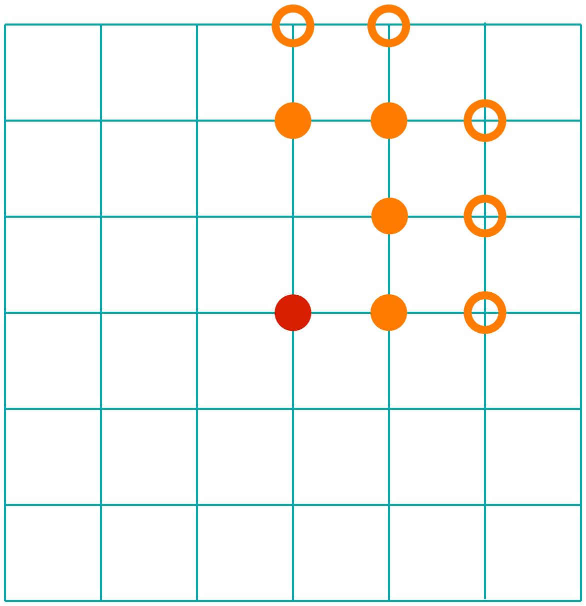
Orange = growth from first neighbor of seed

LIFO site percolation



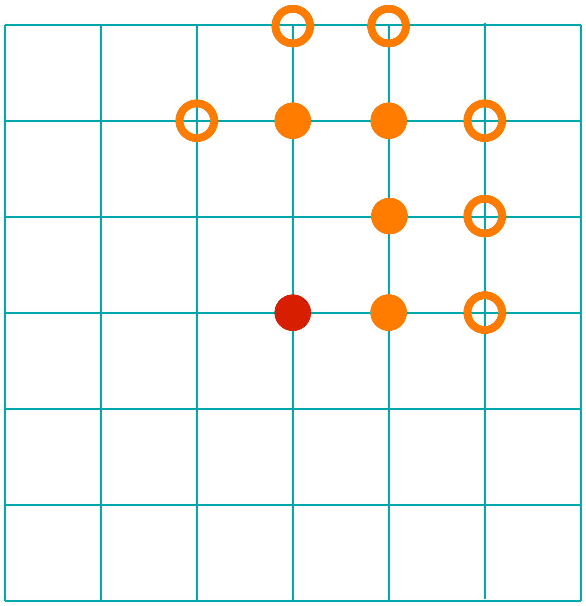
Orange = growth from first neighbor of seed

LIFO site percolation



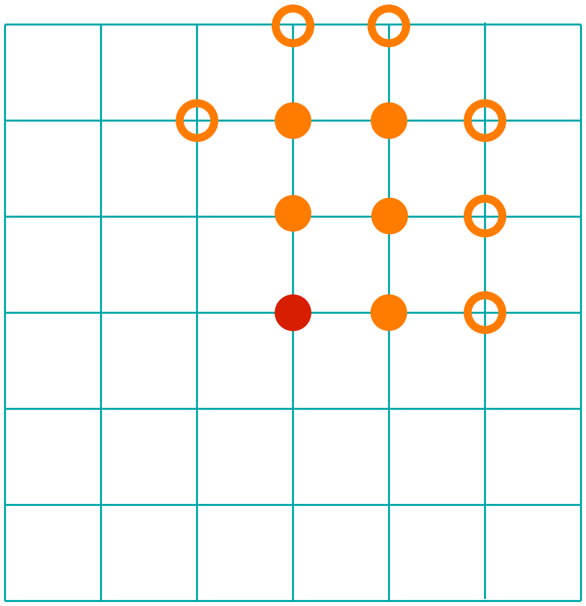
Orange = growth from first neighbor of seed

LIFO site percolation



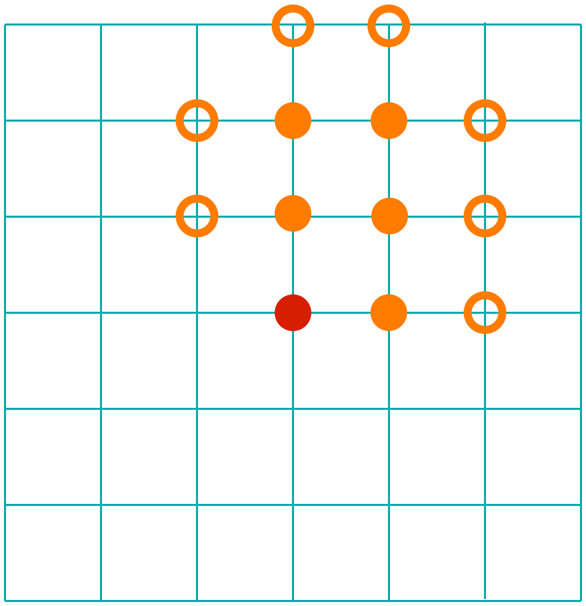
Orange = growth from first neighbor of seed

LIFO site percolation



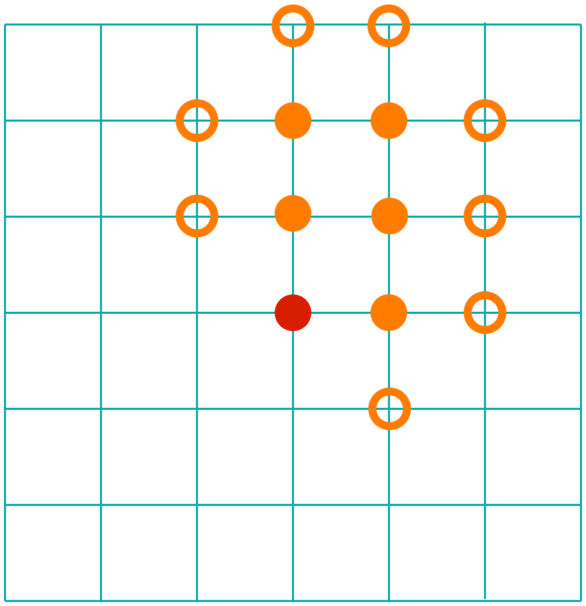
Orange = growth from first neighbor of seed

LIFO site percolation



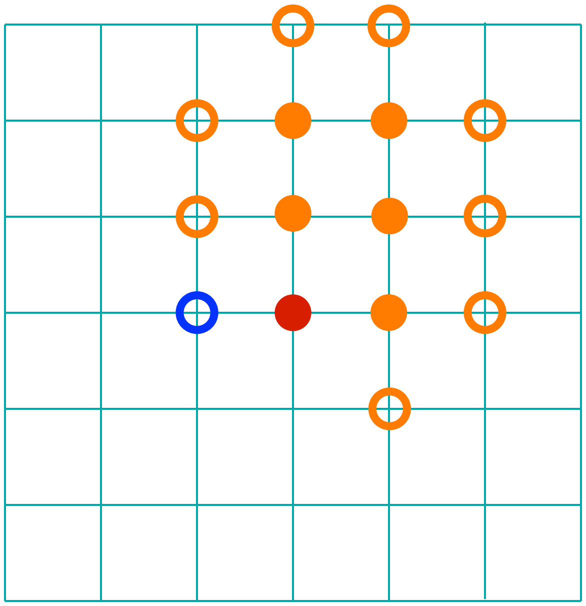
Orange = growth from first neighbor of seed

LIFO site percolation



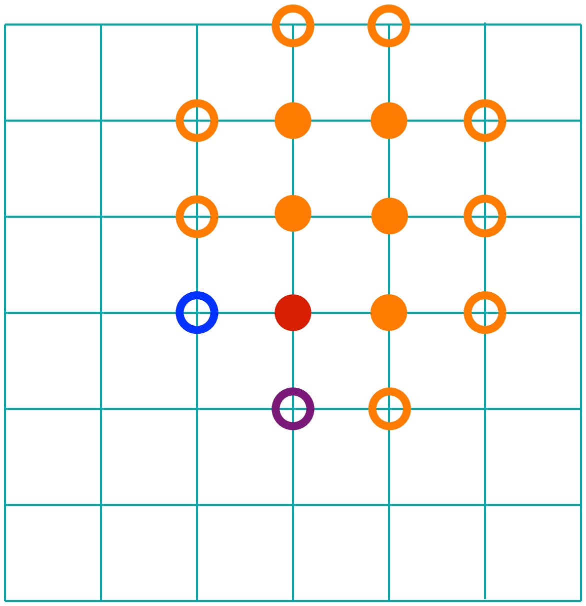
Orange = growth from first neighbor of seed

LIFO site percolation



Blue = growth from third neighbor of seed

LIFO site percolation

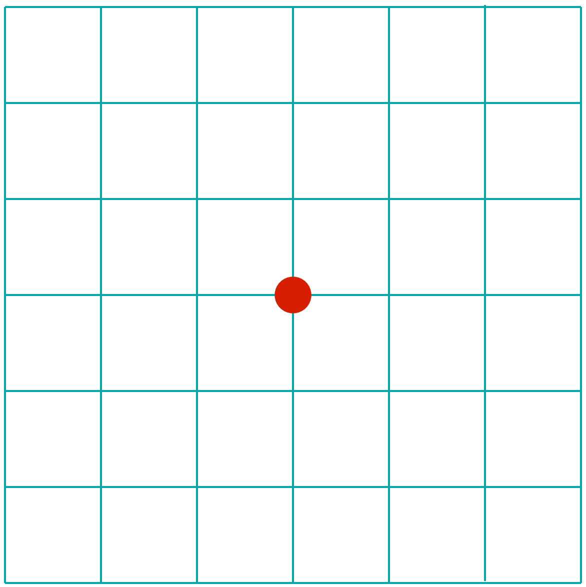


Violet = growth from fourth neighbor of seed

LIFO algorithm (can also use recursion)

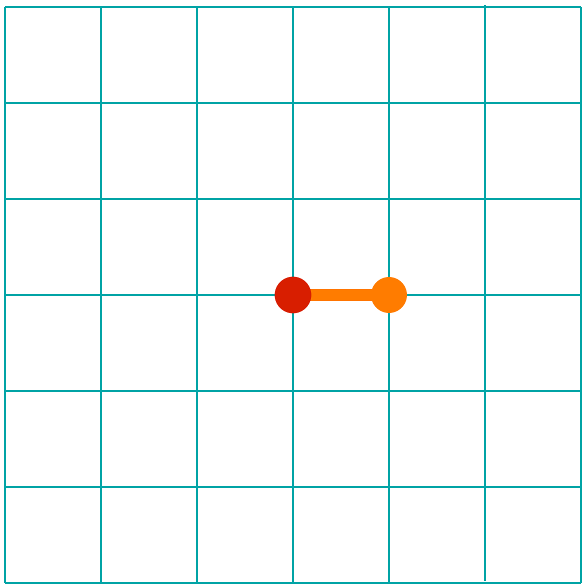
```
Get new growth site from stack
For (neighbors = 1 to 4)
        if (neighbor == unvisited)
                if (randomnumber < prob)
                        neighbor = occupied
                        put neighbor on stack
                else neighbor = vacant.
```

FIFO bond percolation



Red = seed "activated" or "wet" site

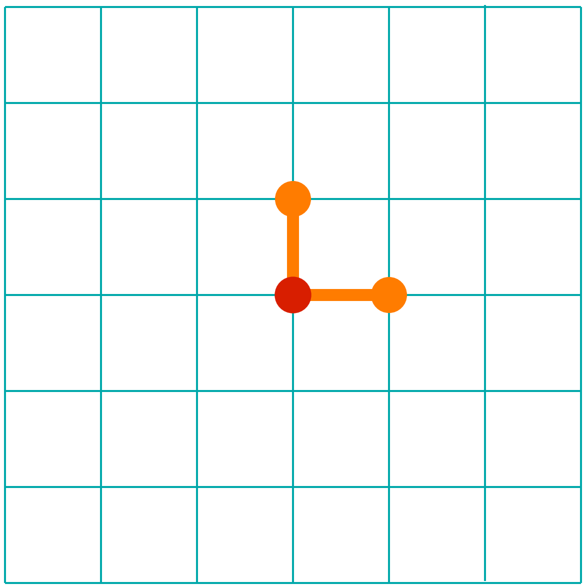
FIFO bond percolation



Orange = first shell of bonds

Add bonds to dry sites with probability p

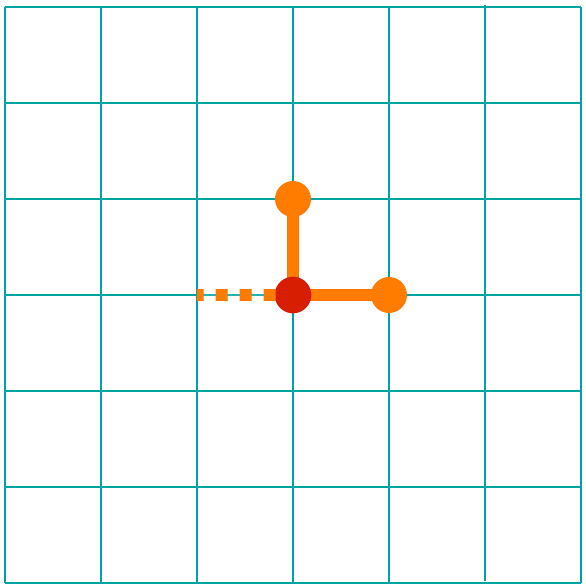
FIFO bond percolation



Orange = first shell of bonds

Add bonds to dry sites with probability p

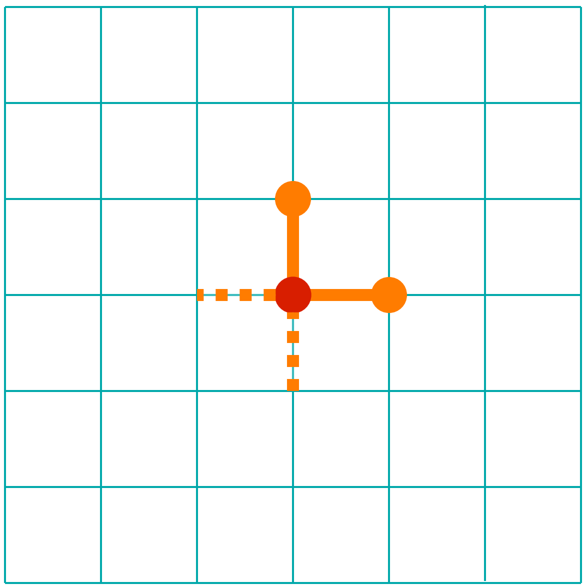
FIFO bond percolation



Orange = first shell of bonds

Vacant bond with probability 1 - p

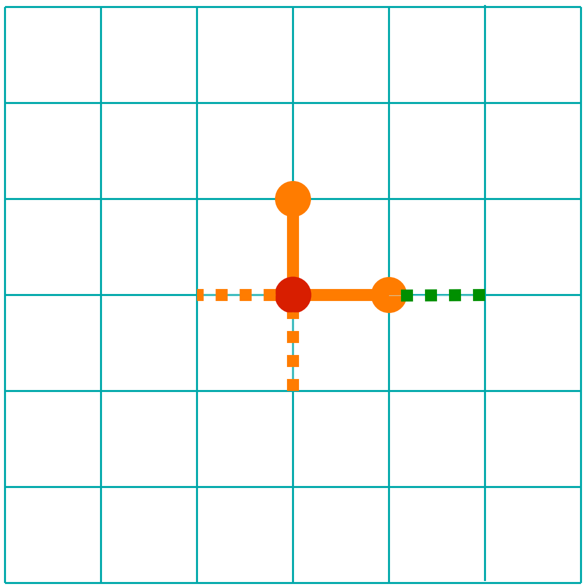
FIFO bond percolation



Orange = first shell of bonds

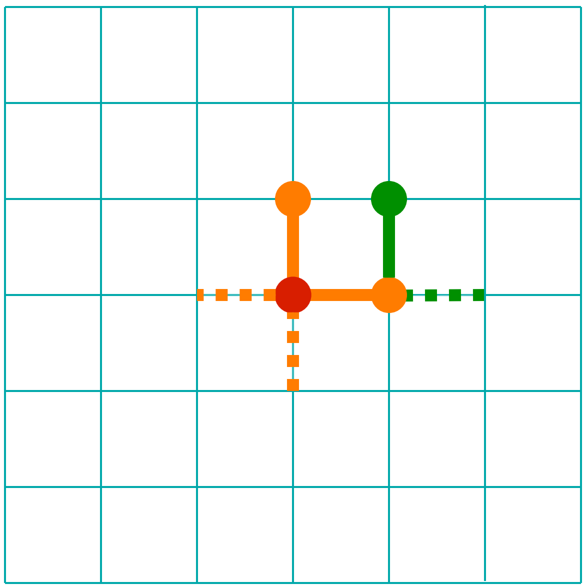
Vacant bond with probability 1 - p

FIFO bond percolation



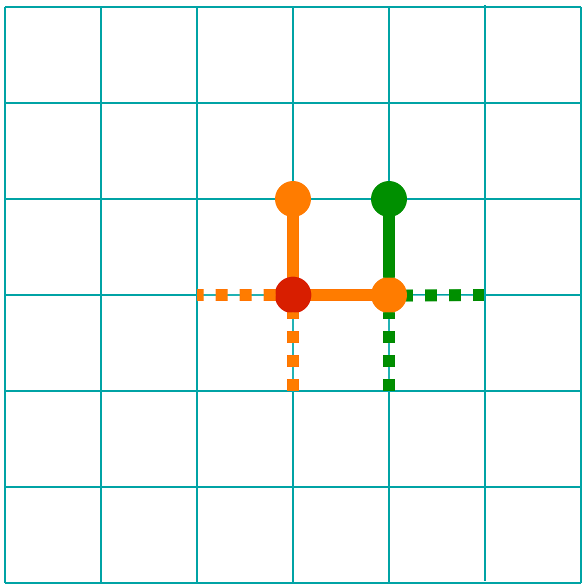
Green = second shell of bonds

FIFO bond percolation



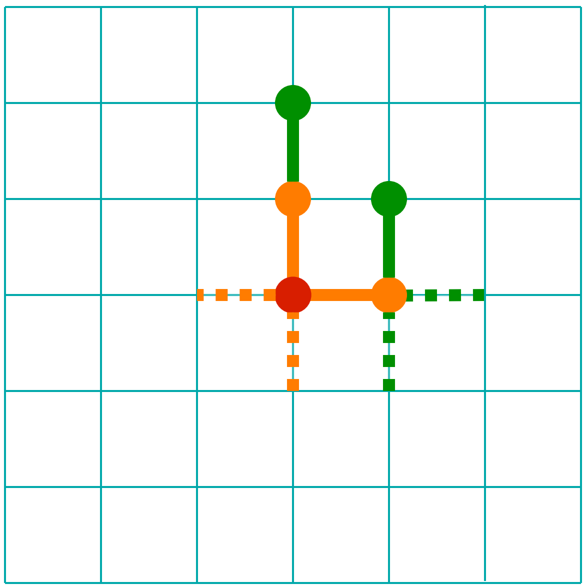
Green = second shell of bonds

FIFO bond percolation



Green = second shell of bonds

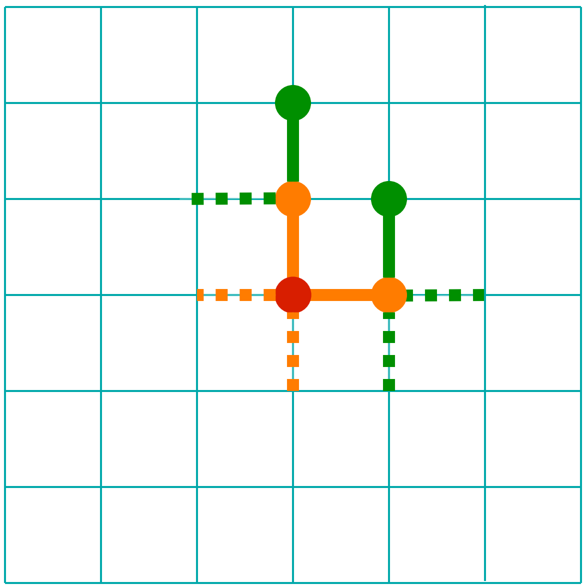
FIFO bond percolation



Green = second shell of bonds

Here we are concerned with the wet sites only and do not add bonds already wet sites

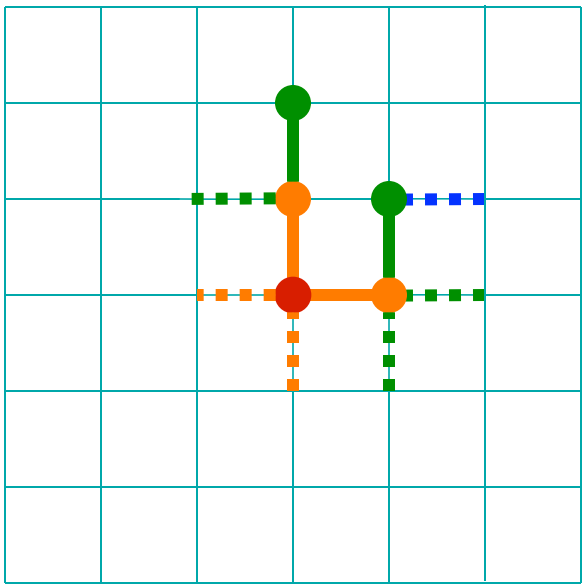
FIFO bond percolation



Green = second shell of bonds

Here we are concerned with the wet sites only and do not add bonds already wet sites

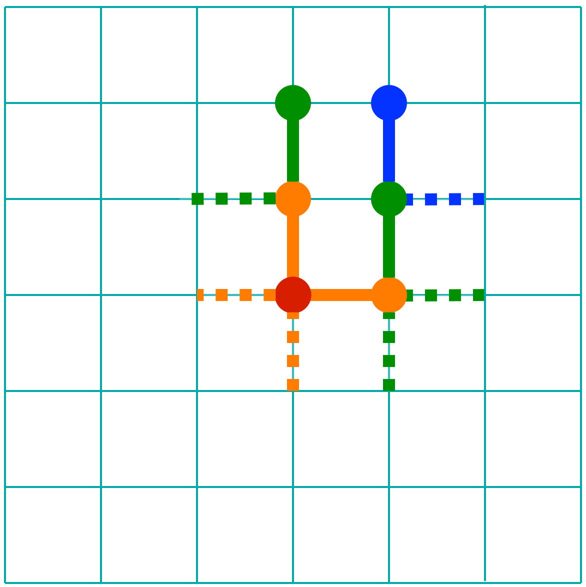
FIFO bond percolation



Blue = third shell of bonds

Here we are concerned with the wet sites only and do not add bonds already wet sites

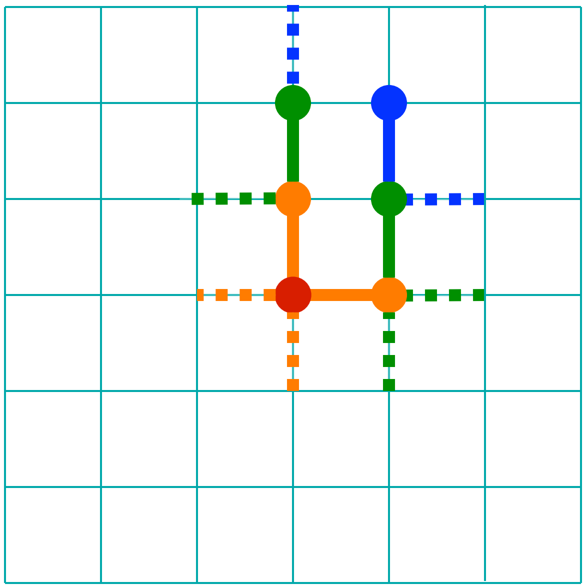
FIFO bond percolation



Blue = third shell of bonds

Here we are concerned with the wet sites only and do not add bonds already wet sites

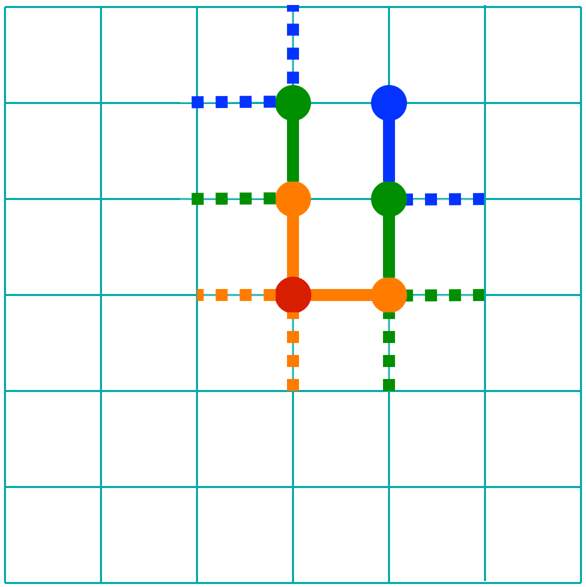
FIFO bond percolation



Blue = third shell of bonds

Here we are concerned with the wet sites only and do not add bonds already wet sites

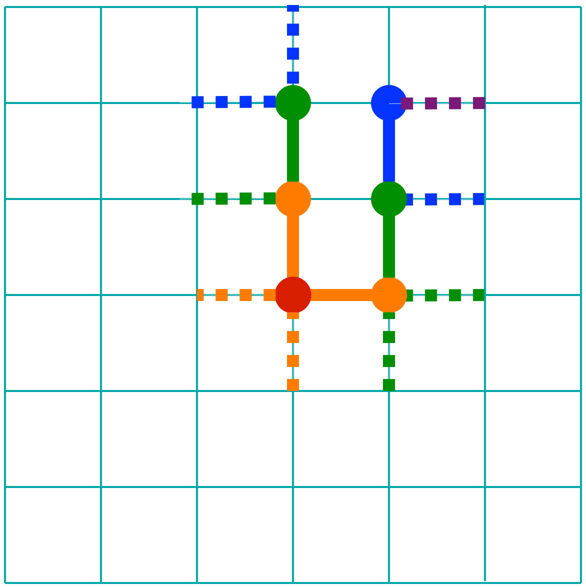
FIFO bond percolation



Blue = third shell of bonds

Here we are concerned with the wet sites only and do not add bonds already wet sites

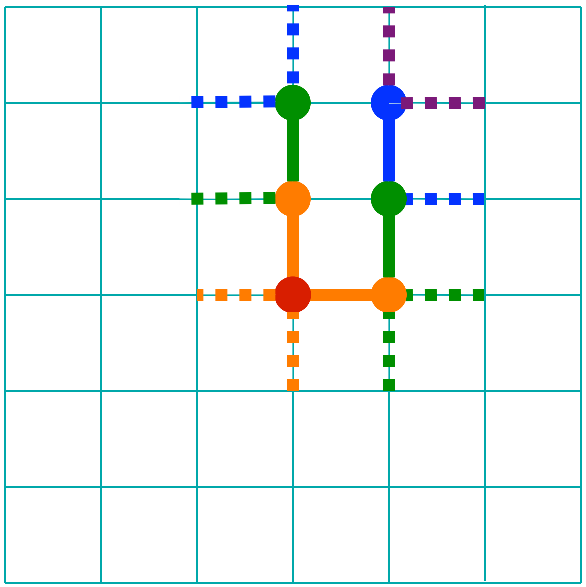
FIFO bond percolation



Violet = fourth shell of bonds

Here we are concerned with the wet sites only and do not add bonds already wet sites

FIFO bond percolation



Violet = fourth shell of bonds

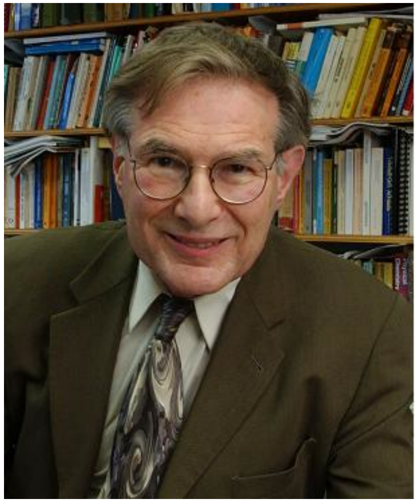
The final object is a minimally spanning tree that connects to every wet site of the cluster

FIFO bond percolation (finding wetted sites)

```
"Pop" new growth site from queue
For (neighbors = 1 to 4)
      if (neighbor == unvisited)
            if (randomnumber < prob)
                  neighbor = occupied
                  "push" neighbor on queue
            ~~else neighbor = vacant.~~
```

Identical to FIFO site perc. except for this line being taken out

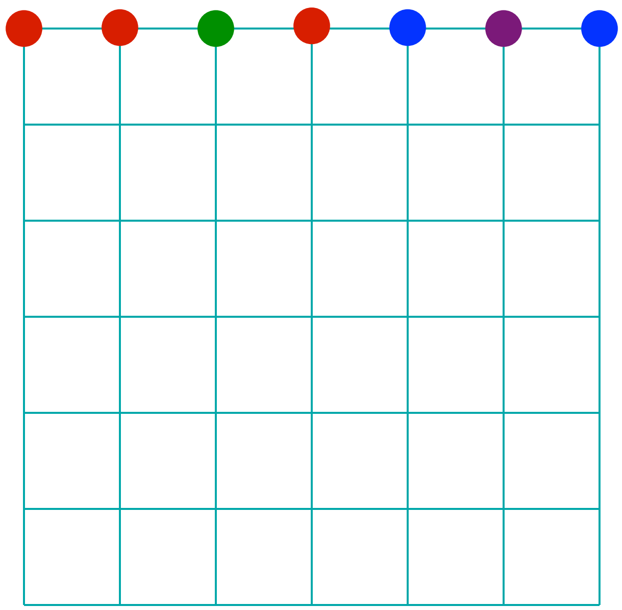
Hoshen-Kopelman algorithm 1976



Raoul Kopelman,
University of Michigan

J. Hoshen and R. Kopelman, Phys. Rev. B 14:3438 (1976).

Hoshen-Kopelman Algorithm

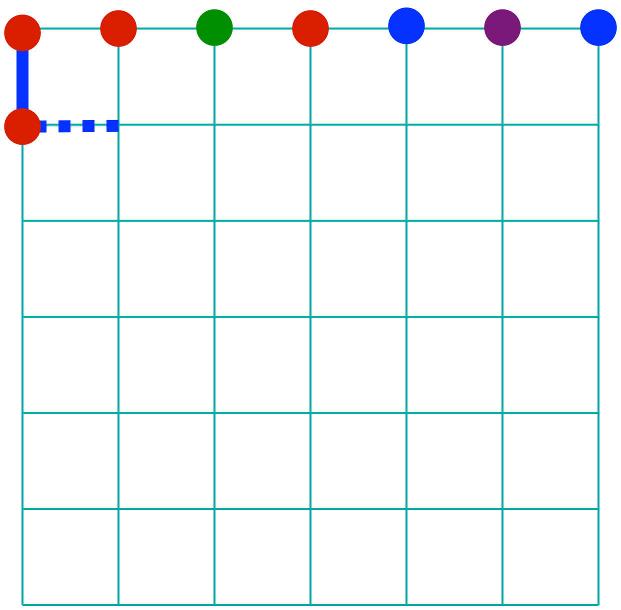


(Bond percolation)

Look at last row of growing inteface.

Each color represents a connected cluster

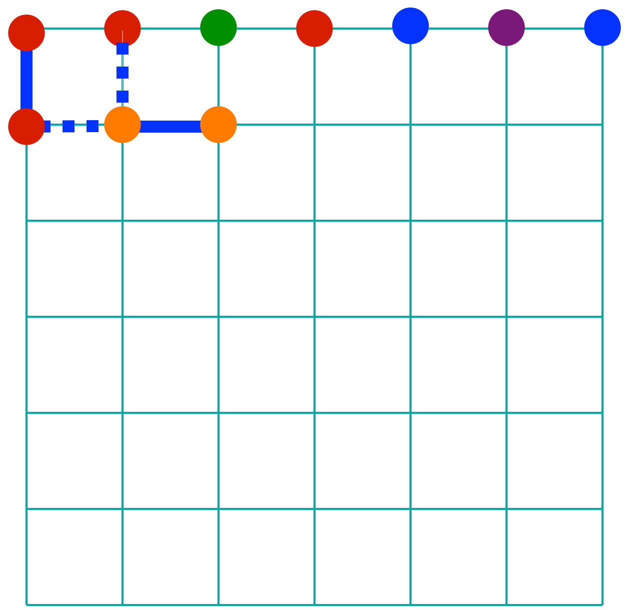
Hoshen-Kopelman Algorithm



(Bond percolation)

Add bonds sequentially with probability p

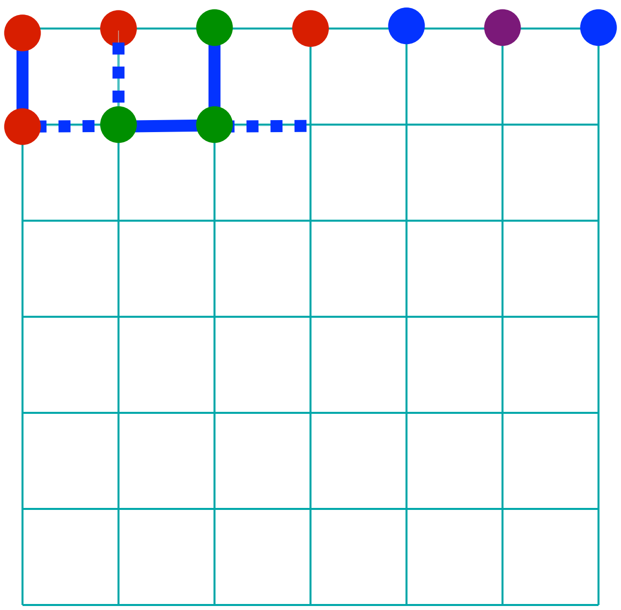
Hoshen-Kopelman Algorithm



(Bond percolation)

Add bonds sequentially with probability p

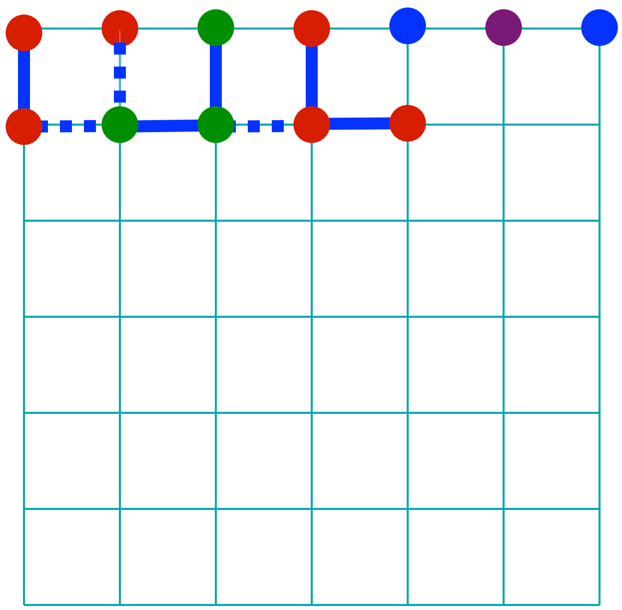
Hoshen-Kopelman Algorithm



(Bond percolation)

Add bonds sequentially with probability p

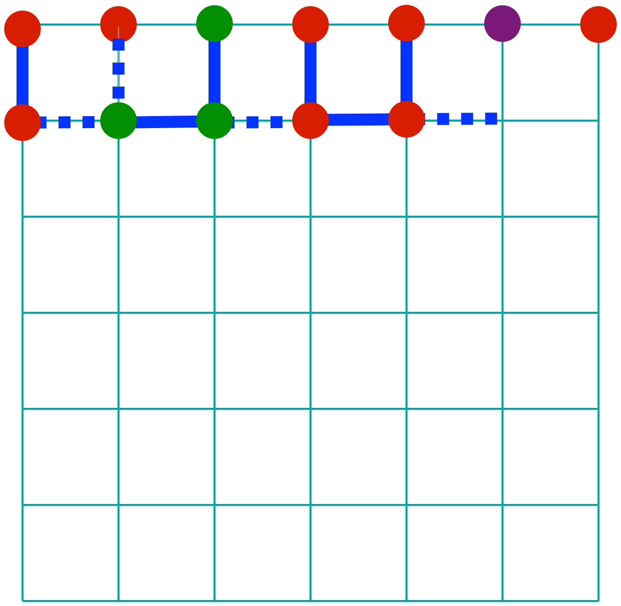
Hoshen-Kopelman Algorithm



(Bond percolation)

Add bonds sequentially with probability p

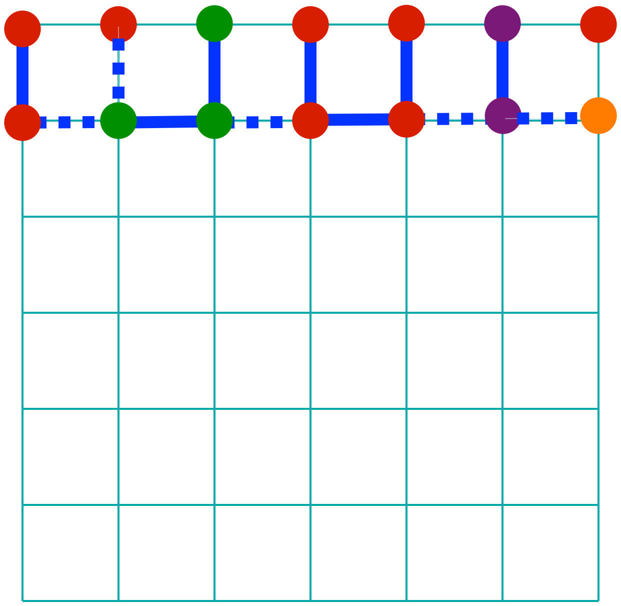
Hoshen-Kopelman Algorithm



(Bond percolation)

Add bonds sequentially with probability p

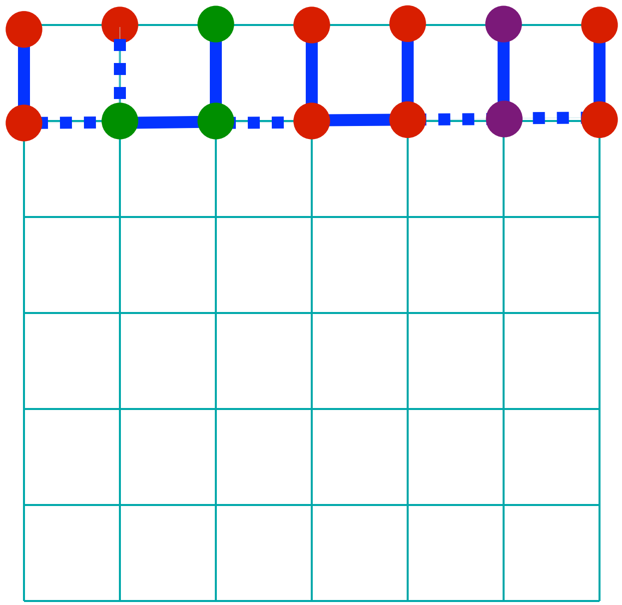
Hoshen-Kopelman Algorithm



(Bond percolation)

Add bonds sequentially with probability p

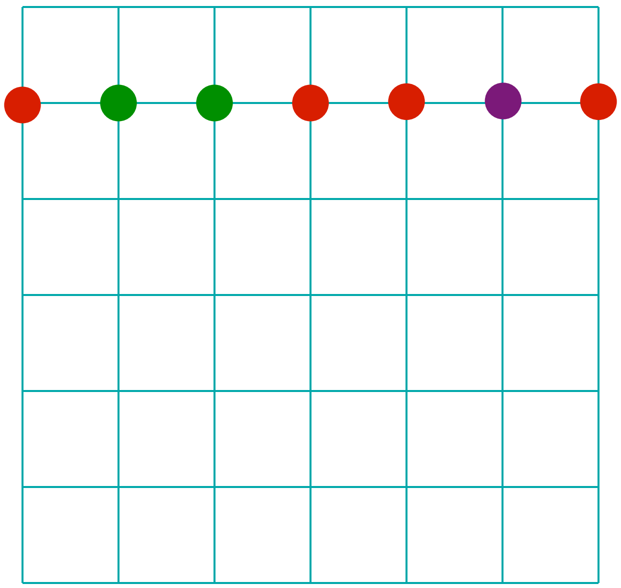
Hoshen-Kopelman Algorithm



(Bond percolation)

Add bonds sequentially with probability p

Hoshen-Kopelman Algorithm



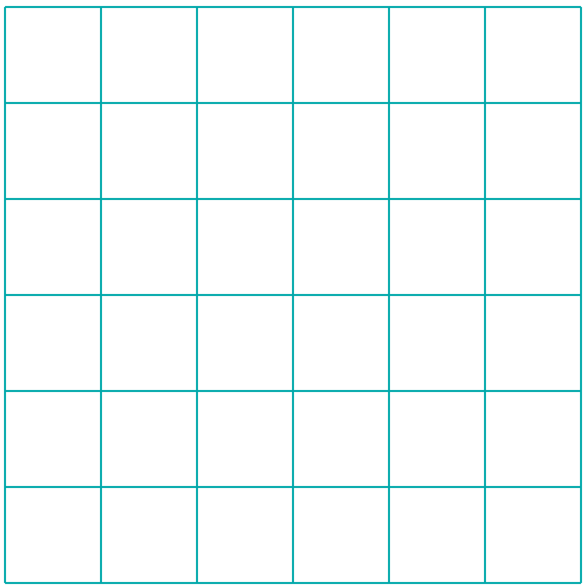
(Bond percolation)

Repeat!!!!

Can simulate lattices as large as 100,000,000 x 100,000,000 this way!!!!!

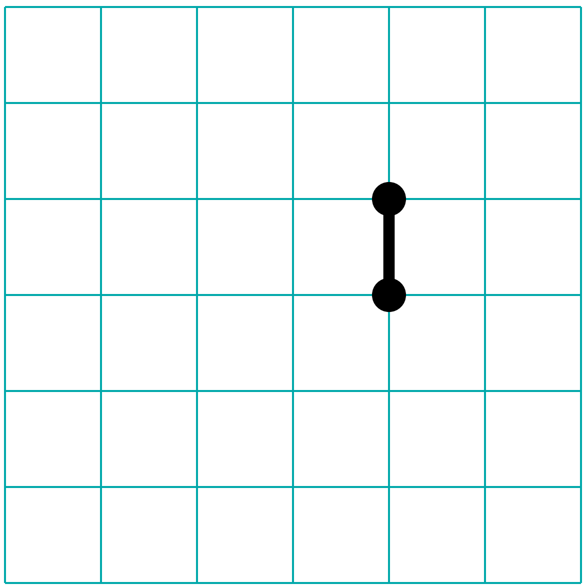
Newman-Ziff algorithm 2000

Newman-Ziff Algorithm



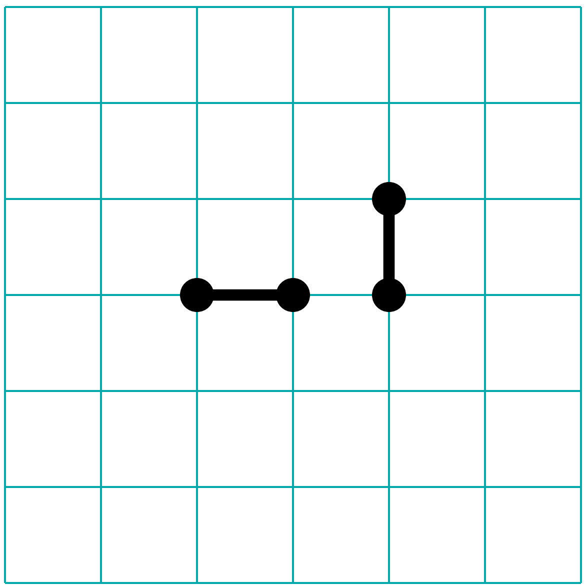
Start with an empty lattice, compute Q $(\Gamma_0) = Q_0$

Newman-Ziff Algorithm



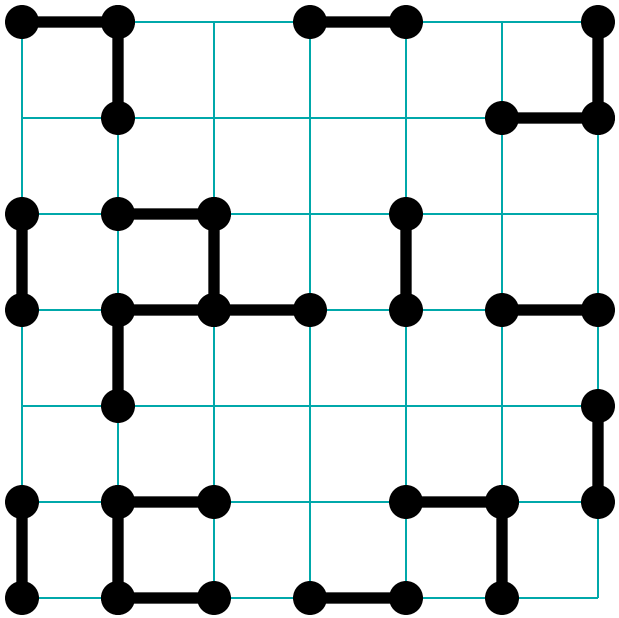
Randomly occupy a bond, compute $Q(\Gamma_1)$

Newman-Ziff Algorithm



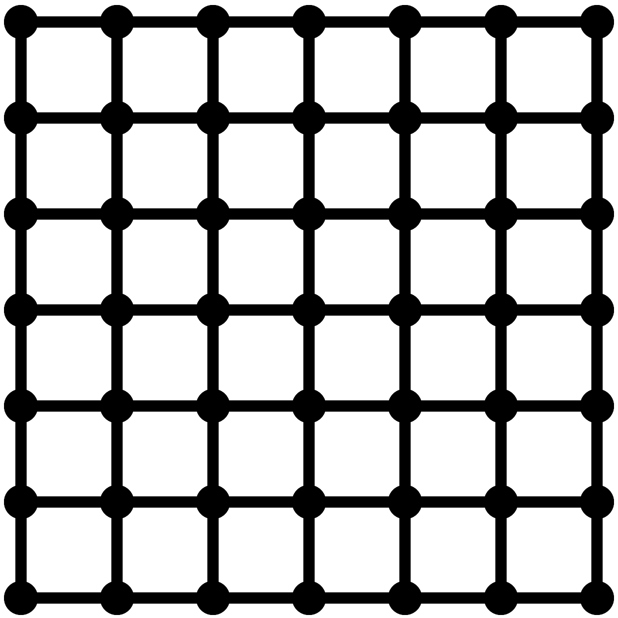
Randomly occupy an unoccupied bond, compute $Q(\Gamma_2)$

Newman-Ziff Algorithm



And so on and compute $Q(\Gamma_b)$ with *b* number of bonds

Newman-Ziff Algorithm



Until all bonds are occupied, compute $Q(\Gamma_M)$

M is the total number of bonds

Newman-Ziff Algorithm

- Any quantity as a function of *p* is computed as

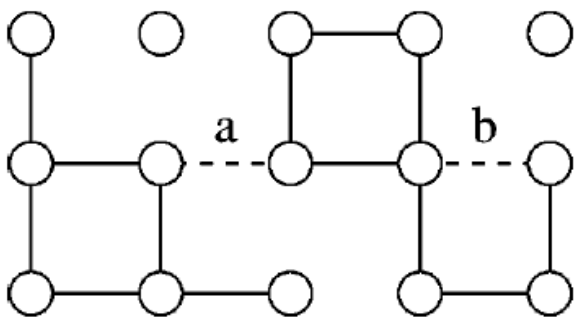
$$\langle Q \rangle = \sum_{b=0}^{M} \frac{M!}{b!(M-b)!} p^b (1-p)^{M-b} Q_b$$

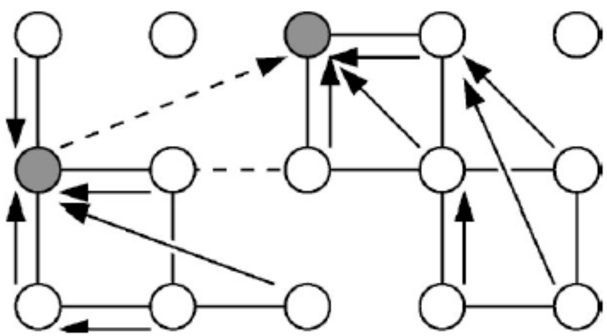
where $Q_b = Q(\Gamma_b)$.

- Each sweep takes time of O(N)

M. E .J. Newman and R. M. Ziff, Phys. Rev. Letters 85, 4104 (2000)

- Bonds are added one at a time, and a bookkeeping scheme is used to keep track of the cluster structure.

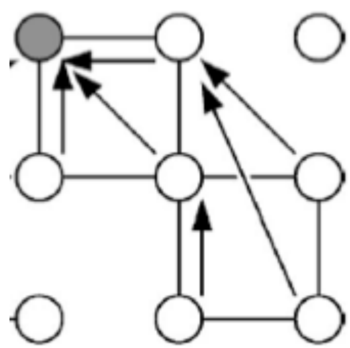




Mark Newman, U. Michigan

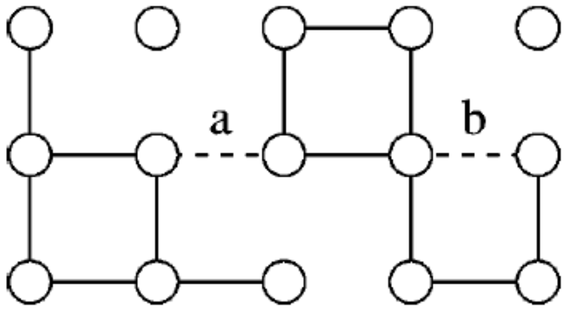
data structure

- At each lattice site is a variable, ptr[i]
- If the ptr[i] > 0, it gives the position of another site on the cluster (a link).
- If ptr[i] < 0, "i" is the root of the cluster, and |ptr[i]| gives the number of sites belonging to the cluster.

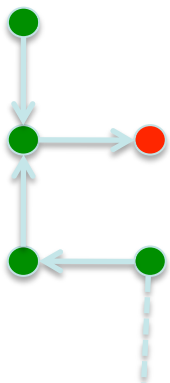


procedure

- Initially, a random ordering of the bonds of the system is made
- 1. A bond is chosen randomly, and *findroot* is used to find the root at each end (and the link paths are collapsed)
- a) If the two ends belong to different clusters, the two are merged.
- b) If both ends of the bond are in the same cluster, nothing is done.

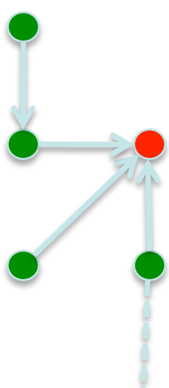


findroot



Before

```
int findroot(int i)
{
    if (ptr[i]<0) return i;
    return ptr[i] = findroot(ptr[i]);
}
```



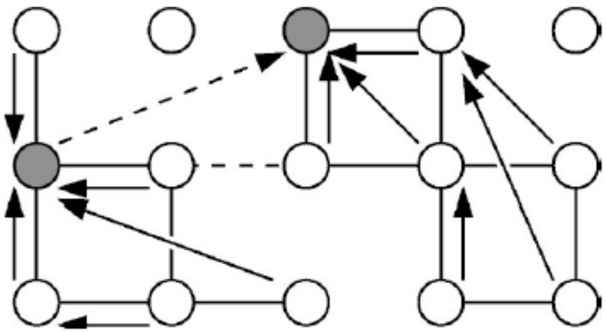
After

-- findroot jumps from link to link until it gets to the root

-- when the recursive calls "unwind", they rename every link to point to the root

merging

-- the root of the smaller cluster is linked to the root of the larger cluster, and the size is adjusted accordingly

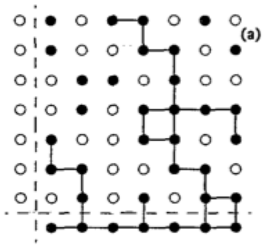


convolution

To go from the canonical (fixed number of bonds) to the grand-canonical (fixed occupancy *p*), one must convolve with a binomial distribution:

$$Q(p) = \sum_{n=0}^{N} B(N,n,p) Q_n = \sum_{n=0}^{N} \binom{N}{n} p^n (1-p)^{N-n} Q_n$$

Q_n = quantity (such as mean size) for a system of fixed *n*
Q(p) = same quantity for a fixed probability *p*



Example of Exact canonical-grand canonical: the crossing probability of a square system, up to 7 x 7 (from exact enumeration):

L	$R_L(p,q)$
2	$2p^2q^2+4p^3q+p^4$
3	$3p^3q^6+22p^4q^5+59p^5q^4+67p^6q^3+36p^7q^2+9p^8q+p^9$
4	$4p^4q^{12}+60p^5q^{11}+390p^6q^{10}+1452p^7q^9+3416p^8q^8+5272p^9q^7+5414p^{10}q^6+3736p^{11}q^5$ $+1752p^{12}q^4+560p^{13}q^3+120p^{14}q^2+16p^{15}q+p^{16}$
5	$5p^5q^{20}+124p^6q^{19}+1418p^7q^{18}+9958p^8q^{17}+48\,171p^9q^{16}+170\,391p^{10}q^{15}+456\,051p^{11}q^{14}$ $+942\,077p^{12}q^{13}+151\,813\,3p^{13}q^{12}+191\,788\,7p^{14}q^{11}+190\,335\,9p^{15}q^{10}+148\,630\,8p^{16}q^9$ $+915\,643p^{17}q^8+446\,538p^{18}q^7+172\,749p^{19}q^6+52\,871p^{20}q^5+12\,650p^{21}q^4+2300p^{22}q^3$ $+300p^{23}q^2+25p^{24}q+p^{25}$
6	$6p^6q^{30}+220p^7q^{29}+3830p^8q^{28}+42\,200p^9q^{27}+330\,862p^{10}q^{26}+196\,683\,2p^{11}q^{25}+922\,005\,1p^{12}q^{24}$ $+349\,865\,68p^{13}q^{23}+109\,429\,240p^{14}q^{22}+285\,726\,952p^{15}q^{21}+628\,339\,894p^{16}q^{20}$ $+117\,065\,617\,2p^{17}q^{19}+185\,451\,985\,6p^{18}q^{18}+250\,279\,719\,2p^{19}q^{17}+287\,954\,750\,7p^{20}q^{16}$ $+282\,477\,386\,8p^{21}q^{15}+236\,295\,381\,8p^{22}q^{14}+168\,645\,572\,0p^{23}q^{13}+102\,808\,519\,7p^{24}q^{12}$ $+536\,110\,144p^{25}q^{11}+239\,427\,498p^{26}q^{10}+915\,847\,20p^{27}q^9+299\,432\,38p^{28}q^8$ $+832\,262\,0p^{29}q^7+194\,684\,2p^{30}q^6+376\,992p^{31}q^5+58\,905p^{32}q^4+7140p^{33}q^3+630p^{34}q^2$ $+36p^{35}q+p^{36}$
7	$7p^7q^{42}+354p^8q^{41}+8637p^9q^{40}+135\,542p^{10}q^{39}+153\,891\,8p^{11}q^{38}+134\,800\,33p^{12}q^{37}$ $+948\,508\,47p^{13}q^{36}+551\,119\,224p^{14}q^{35}+269\,732\,922\,5p^{15}q^{34}+112\,862\,456\,29p^{16}q^{33}$ $+408\,335\,758\,12p^{17}q^{32}+128\,871\,332\,816p^{18}q^{31}+357\,226\,485\,246p^{19}q^{30}$ $+874\,366\,412\,699p^{20}q^{29}+189\,748\,991\,302\,9p^{21}q^{28}+366\,204\,287\,877\,7p^{22}q^{27}$ $+629\,886\,980\,328\,3p^{23}q^{26}+966\,956\,844\,729\,7p^{24}q^{25}+132\,585\,068\,442\,89p^{25}q^{24}$ $+162\,424\,120\,333\,36p^{26}q^{23}+177\,768\,801\,987\,90p^{27}q^{22}+173\,788\,593\,629\,74p^{28}q^{21}$ $+151\,728\,375\,886\,87p^{29}q^{20}+118\,300\,132\,565\,60p^{30}q^{19}+823\,920\,775\,762\,1p^{31}q^{18}$ $+512\,857\,828\,295\,4p^{32}q^{17}+285\,516\,297\,755\,8p^{33}q^{16}+142\,265\,267\,827\,2p^{34}q^{15}$ $+634\,745\,588\,151p^{35}q^{14}+253\,562\,760\,568p^{36}q^{13}+905\,980\,448\,53p^{37}q^{12}+288\,886\,115\,91p^{38}q^{11}$ $+818\,938\,813\,8p^{39}q^{10}+205\,207\,815\,2p^{40}q^9+450\,849\,373p^{41}q^8+858\,971\,97p^{42}q^7$ $+139\,838\,16p^{43}q^6+190\,688\,4p^{44}q^5+211\,876p^{45}q^4+18424p^{46}q^3+1176p^{47}q^2$ $+49p^{48}q+p^{49}$

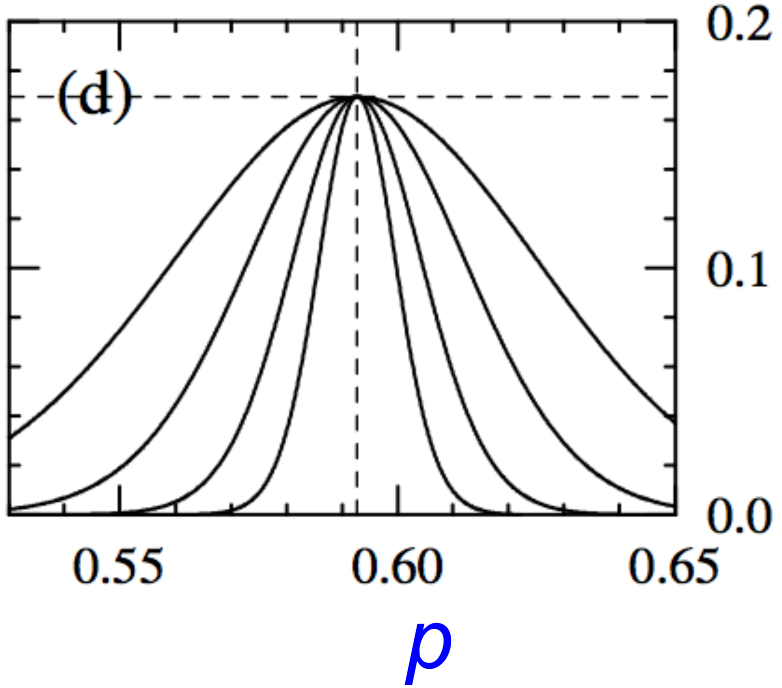
example: probability of wrapping a square torus in one direction but not the other for all values of p

- L x L, L = 32, 64, 128, 256
- Reaches maximum ≈ p_c
- Excellent convergence:

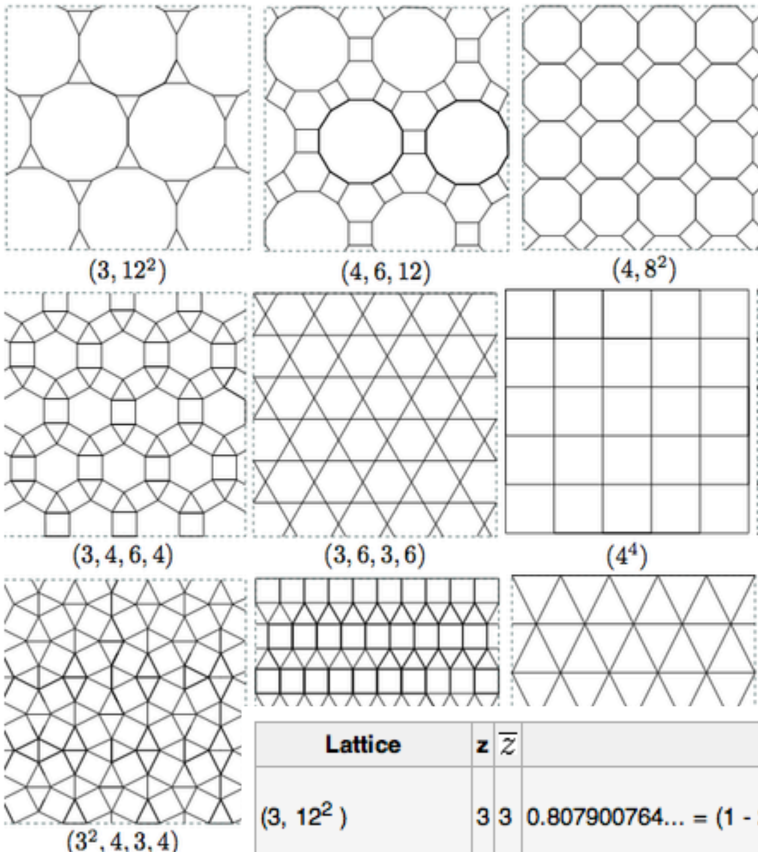
$$p - p_c \sim L^{-2-1/\nu} = L^{-11/4}$$

Easily find

- $p_c = 0.5927462..$



Thresholds on 2d regular and Archimedean lattices



"Percolation Threshold" Wikipedia page

Lattice	z	$\bar{z}$	Site Percolation Threshold	Bond Percolation Threshold
$(3, 12^2)$	3	3	0.807900764... = $(1 - 2 \sin(\pi/18))^{1/2}$ [4]	0.74042195(80)[5], 0.74042081(10) [6], 0.74042077(2)[7],
(4, 6, 12)	3	3	0.747806(4) [4]	0.69373383(72)[5]
$(4, 8^2)$	3	3	0.729724(3) [4]	0.67680232(63)[5]
honeycomb (6^3)	3	3	0.697043(3)[4] 0.6970413(10) [6]	0.652703645... = $1-2 \sin(\pi/18)$, $1+ p^3-3p^2=0$[8]
kagomé (3, 6, 3, 6)	4	4	0.652703645... = $1 - 2 \sin(\pi/18)$ [8]	0.524404978(5)[7], 0.52440499(2)[9], 0.52440516(10) [6], 0.5244053(3) [10]
(3, 4, 6, 4)	4	4	0.621819(3) [4]	0.52483258(53)[5]
square (4^4)	4	4	0.59274621(13) [11], 0.59274621(33) [12], 0.59274598(4) [13][14], 0.59274605(3)[9]	1/2
$(3^4, 6)$	5	5	0.579498(3) [4]	0.43430621(50)[5]
puzzle $(3^2, 4, 3, 4)$	5	5	0.550806(3) [4]	0.41413743(46)[5]
$(3^3, 4^2)$	5	5	0.550213(3) [4]	0.41964191(43) [5]
triangular (3^6)	6	6	1/2	0.347296355... = $2 \sin(\pi/18)$, $1+ p^3-3p=0$[8]

Correction-to-scaling exponent for two-dimensional percolation

$$n_s(p_c) \sim A s^{-\tau}(1 + C s^{-\Omega} + \ldots),$$

n_s = number of clusters of size s, at the critical threshold pc. τ = 187/91.

TABLE I. History of determinations of Ω, $\omega = D\Omega = (91/48)\Omega$, and $\Delta_1 = \Omega/\sigma = (91/36)\Omega$. Numbers in parentheses represent errors in last digit(s), and are shown on original values only.

Year	Author	Method	Ω	ω	Δ_1
1976	Gaunt and Sykes [4]	Series	0.75(5)	1.42	1.90
1978	Houghton, Reeve, and Wallace [5]	Field theory	0.54–0.68	0.989–1.28	1.32–1.71
1979	Hoshen *et al.* [6]	MC	0.67(10)	1.27	1.69
1980	Pearson [7]	Conjecture	$64/91 \approx 0.703$	1.333	1.778
1980	Nakanishi and Stanley [8]	MC	0.6–1		
1982	Nienhuis [9]	Field theory	$96/91 \approx 1.055$	2	2.667
1982,1983	Adler, Moshe, and Privman [10,11]	Series $p < p_c$	0.5	0.95	1.26
	Adler, Moshe, and Privman [10,11]	Series	0.66(7)	1.25	1.67
1983	Aharony and Fisher [12,13]	RG theory	$55/91 \approx 0.604$	$55/48 \approx 1.15$	$55/36 \approx 1.53$
1983,1984	Margolina *et al.* [13,14]	MC	0.64(8)	1.21	1.62
	Margolina *et al.* [13,14]	Series	0.8(1)	1.52	2.02
1985	Adler [15]	Series	0.63(5)	1.19	1.59
1986	Rapaport [16]	MC	0.71–0.74		
1998	MacLeod and Jan [17]	MC	0.65(5)	1.23	1.64
1999	Ziff and Babalievski [18]	MC	0.77(2)	1.46	1.95
2001	Tiggemann [19]	MC	0.70(2)	1.33	1.77
2003	Aharony and Asikainen [20,21]	Theory (hulls)	72/91	1.5	2
2007	Tiggemann [22]	MC	0.73(2)	1.38	1.85
2008	Kammerer, Höfling, and Franosch [23]	MC	0.77(4)	1.46	1.95
2010	This work	Theory	$72/91 \approx 0.791$	1.5	2

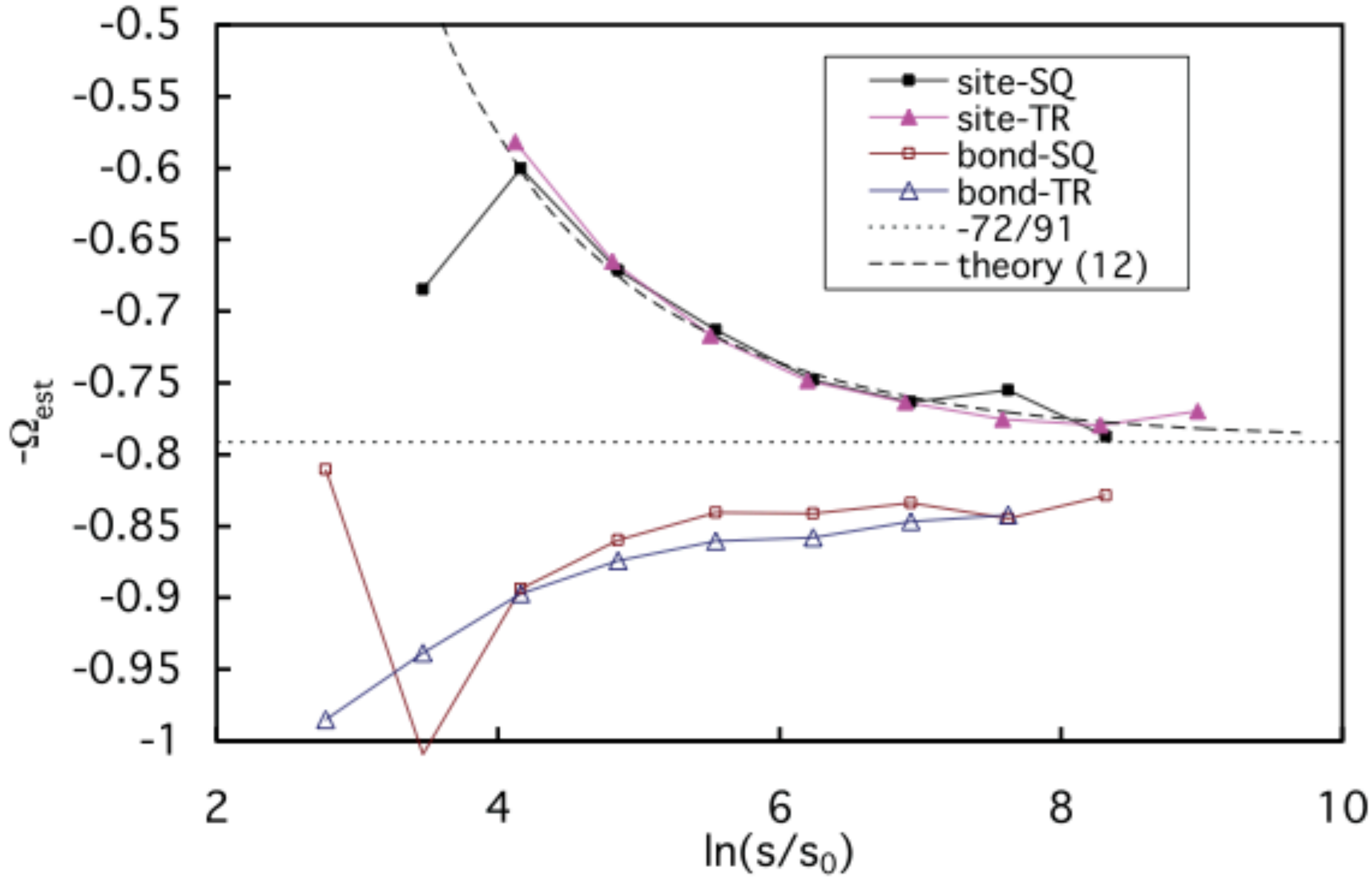
Used Cardys' result for crossing of an annulus to find size distribution

$$\Pi(\tau) = \frac{\eta(-1/3\tau)\eta(-4/3\tau)}{\eta(-1/\tau)\eta(-2/3\tau)} = (3/2)^{1/2}\frac{\eta(3\tau)\eta(3\tau/4)}{\eta(\tau)\eta(3\tau/2)}, \qquad (3)$$

where $\eta(\tau) = q^{1/24}\prod_{n=1}^{\infty}(1-q^n) = \sum_{n=-\infty}^{\infty}(-1)^n q^{(6n+1)^2/24}$

$$\Pi(R/R_1) = \sqrt{\frac{3}{2}}\left(\frac{R}{R_1}\right)^{-5/48}\left\{1 - \left(\frac{R}{R_1}\right)^{-3/2} + \left(\frac{R}{R_1}\right)^{-2} - \left(\frac{R}{R_1}\right)^{-7/2} + 2\left(\frac{R}{R_1}\right)^{-4} - \dots\right\}, \qquad (5$$

Simulation results – up to 2.5 x 10^{11} clusters up to size s = 1000.

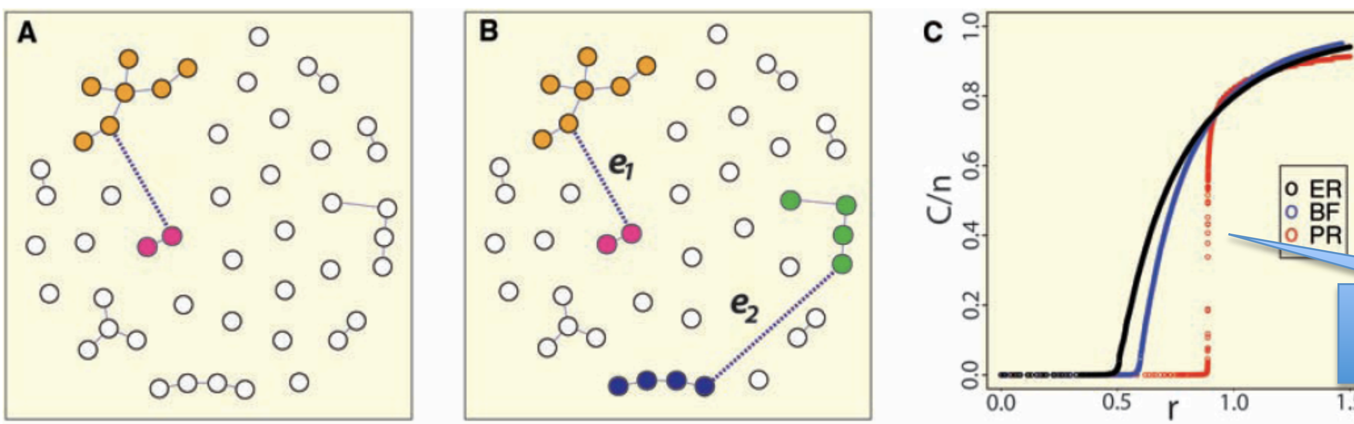


Explosive growth in clusters created through a biased “Achlioptas” growth process on a regular lattice

Achlioptas process

- Recently, Achlioptas, D'Sousa, and Spencer considered cluster growth on random (Erdös-Rényi) lattices by the so-called Achlioptas process:
  - Pick *two* bonds
  - Calculate weight = product of masses of the two clusters the bond connects
  - Choose bond of *lower* weight

Achlioptas et al, Science 2009



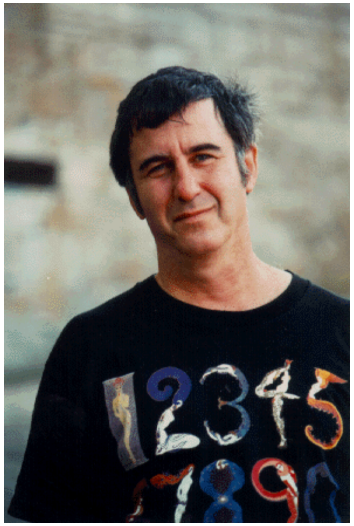
Explosive growth

- ER = Erdös-Rényi (regular percolation), BF = Bounded size rule, PR = product rule. C/n = maximum cluster size divided by the number of sites

- They find "Explosive Growth" in the PR model.

Dimitris Achlioptas, UCSC

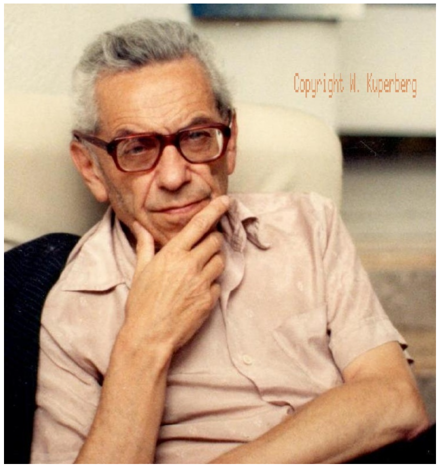
Raissa D’Sousa, UCD

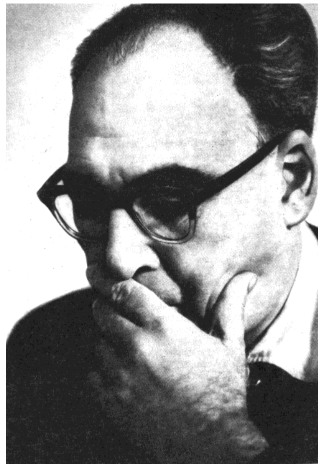


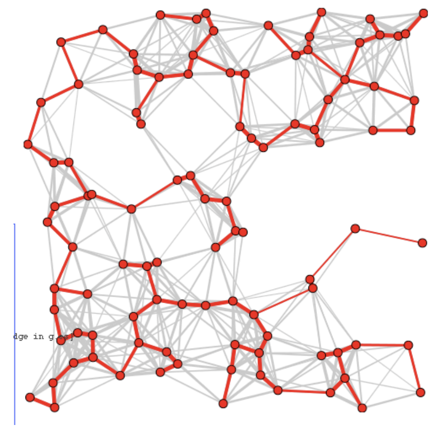
Joel Spencer, NYU

Erdős Rényi Random Graph

Alfréd Rényi 1921-1970

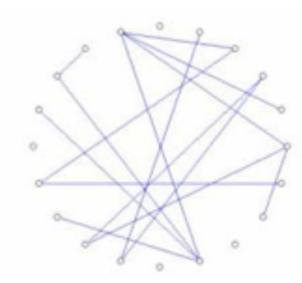


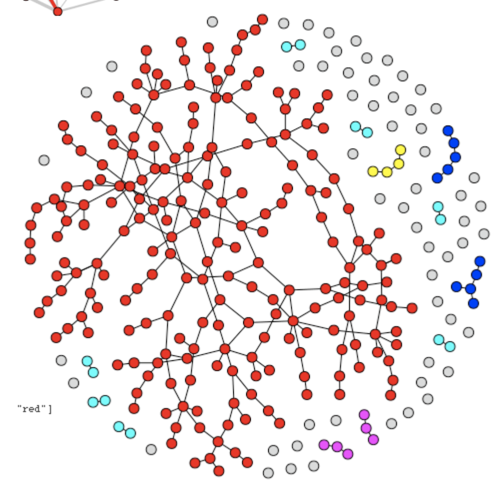




Minimum spanning tree

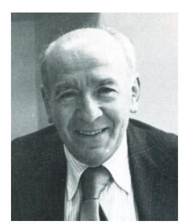
Paul (Pál) Erdős 1913-1996





Tree form

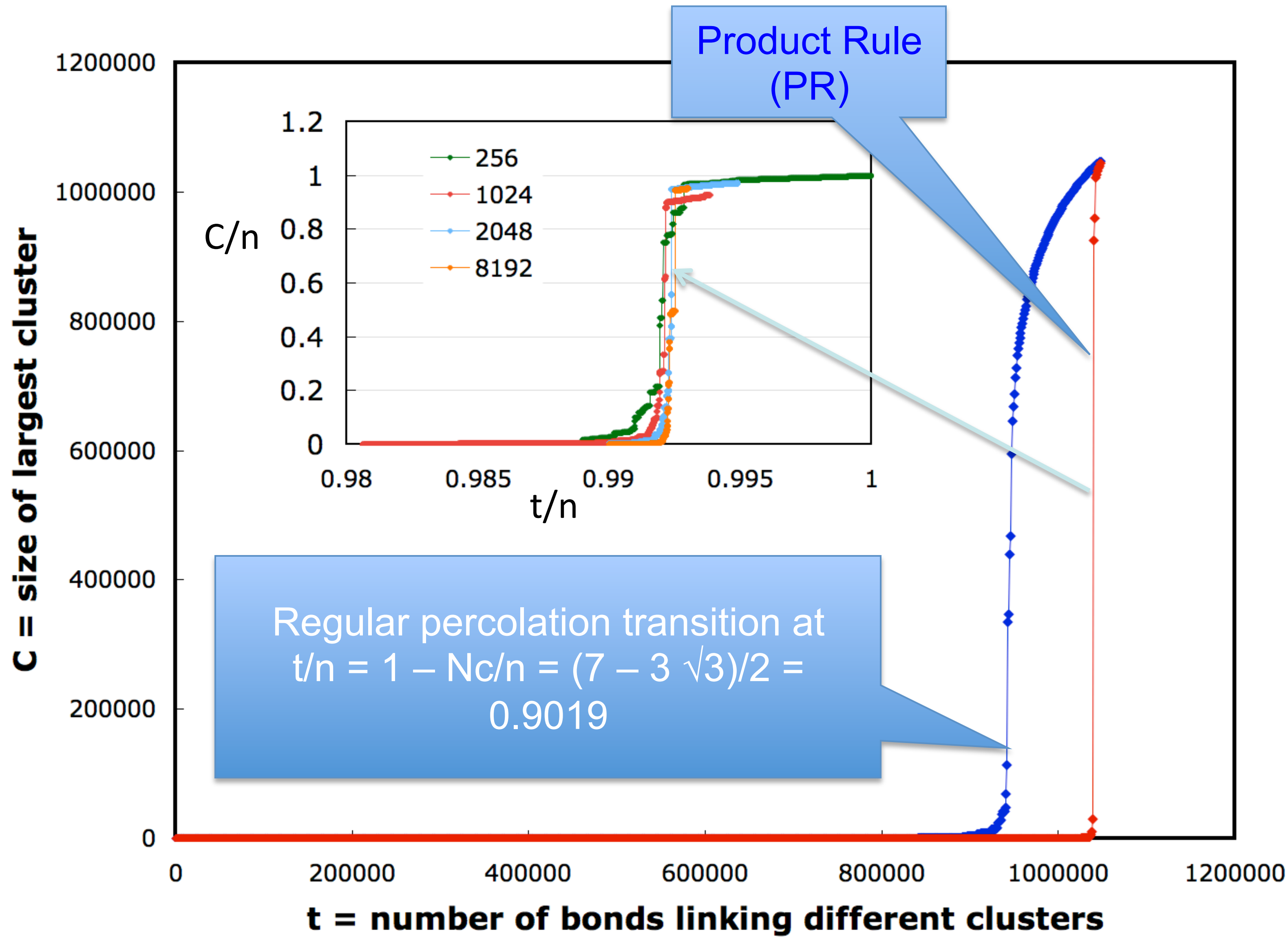
My Erdös number is 2 by way of Mark Kac



Achlioptas processes on a regular (percolation) lattices

- Define t = time = number of bonds added to connect *distinct* clusters.
- Then, the number of clusters is $n - t$, where n is the initial number of sites, since adding additional

For lattice percolation, I find (1024x1024 lattice):



The SIR model on a square lattice

Susceptible-Infected-Recovered

With David de Souza and Tânia Tomé.

- That is S $\rightarrow$ I with rate $(1-c)I_{neighbors}/4$
- I $\rightarrow$R with rate c

- (I remains I for an exponentially distributed time)