Fast percolation

Reminder

Percolation is the study of the global connection behaviour of a network when removin a certain fraction of its nodes, measured in the (relative) size of the largest connected component.

- Goal: learn a way of informed percolation which optimally disconnects a network (remove minimum amount of nodes with maximum effect)
- Maybe useful: for preventing desease spreading, or to find critical nodes in the network

Setting up a model

We attach to each node a Bernoulli random variable of failure:

$$\hat{\mathbf{p}}_i = \sigma(\mathbf{w}^T \cdot \mathbf{info}_i),$$

$$\mathbf{Fail}_i(\varphi) \sim \text{Bernoulli}\left(\hat{\mathbf{p}}_i \, \frac{\varphi}{\frac{1}{n}\sum_i \hat{\mathbf{p}}_i}\right),$$

where σ is the logistic function, **info** is a vector of local information (eg., degree), and **w** the weighting of this information (a logistic model, sort of).

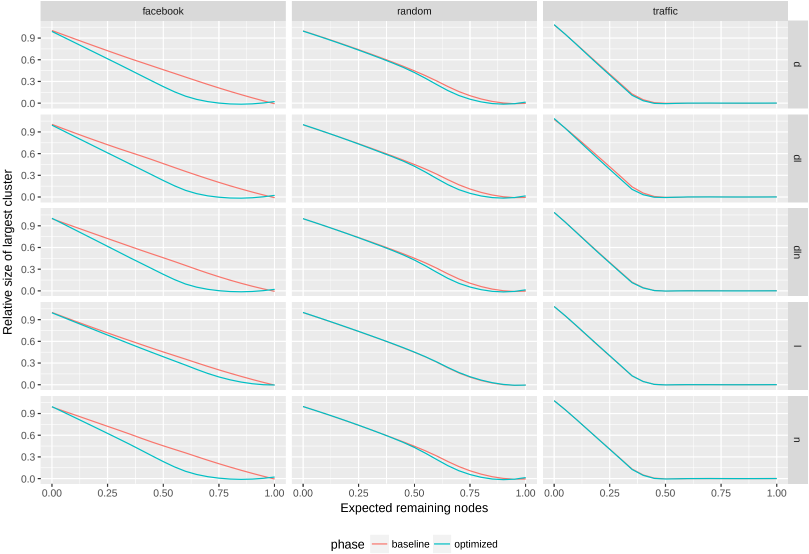
The value φ varies in $[0;1]$ and describes the expected fraction of failing nodes. The individual probabilities are scaled accordingly.

Practical approach

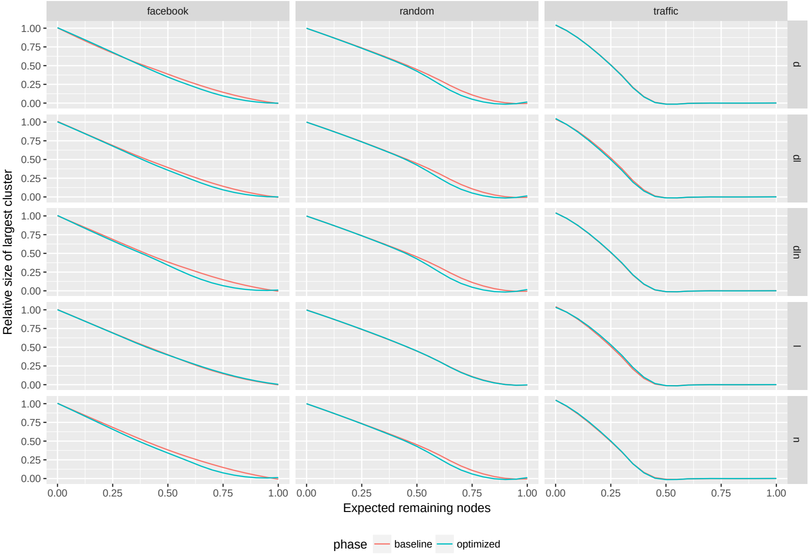
- Minimize the area under the percolation curve with respect to the weights **w**, using simulated annealing
- Sampling the curves (several times) at each parameter evaluation
- Local information used: local clustering coefficient, degree, inverse degree, and size of 2-neighbourhood
- Compare some different network types:

	Facebook		Traffic		Random	
	Train	Test	Train	Test	Train	Test
Vertices	572	347	7388	13389	2000	2000
Edges	3192	2519	10591	21246	5000	5000
Density	0.0195	0.0419	0.0004	0.0002	$\approx$ 0.0025	

Training


facebook
random
traffic
p
dl
dlp
l
u
0.9
0.6
0.3
0.0
0.00
0.25
0.50
0.75
1.00
Relative size of largest cluster
Expected remaining nodes
phase
baseline
optimized

Testing


facebook
random
traffic
p
dl
dlu
l
u
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
Relative size of largest cluster
Expected remaining nodes
phase
baseline
optimized

Observations[1]

- It works, basically, although the impact is not great
- The parametrization seems can be very dependent on the network, and not transferable in general (weights are not very consistent between graph types)
- Using the local clustering coefficient does not seem to gain anything
- I also tried inverse degree, 3-neighbourhood, but they don’t work better
- Interestingly, normalizing the information by maximum degree makes things worse

[1] Code and data: `https://github.com/phipsgabler/netsci-02`