Community Structure of Large Networks

CS 322: (Social and Information) Network Analysis
Jure Leskovec
Stanford University

Modularity

Define **modularity** to be

Q = (number of edges within groups) – (expected number within groups)

Actual Number of Edges between i and j is

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge } (i,j), \\ 0 & \text{otherwise.} \end{cases}$$

Expected Number of Edges between i and j is

$$\text{Expected number} = \frac{k_i k_j}{2m}.$$

$m = \frac{1}{2}\sum_i k_i$

Modularity Matrix

- So Q is a sum of: $A_{ij} - \frac{k_i k_j}{2m}$ (s_i, s_j)

over pairs (i, j) that are in the same group

- Or we can write in matrix form as

$$Q = \mathbf{s}^T \mathbf{B} \mathbf{s},$$

Where *B* is a new characteristic matrix, the *modularity matrix*

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}.$$

Where **s** is a the vector whose elements are s_i

Modularity Matrix

- *s* is the linear combination of the normalized eigenvectors u_i of *B*

Write $\mathbf{s} = \sum_i a_i \mathbf{u}_i$ where $a_i = \mathbf{u}_i^T \mathbf{s}$ and then

$$Q = \mathbf{s}^T \mathbf{B} \mathbf{s} = \sum_i a_i^2 \beta_i .$$

β_i is the eigenvalue of B corresponding to eigenvector u_i

To maximize Q we want to place as much weight as possible in the terms corresponding to the *largest* eigenvalues.

- We maximize the coefficient on the largest eigenvalue by choosing:

$$s_i = \begin{cases} +1 & \text{if } i\text{th element of } \mathbf{u}_1 \geq 0, \\ -1 & \text{if } i\text{th element of } \mathbf{u}_1 < 0. \end{cases}$$

Modularity Matrix

Algorithm:

- Calculate the leading eigenvector of the modularity matrix
- Divide the vertices according to the signs of the elements

Note that there is no need to forbid the solution with all the vertices in a single group.

Example

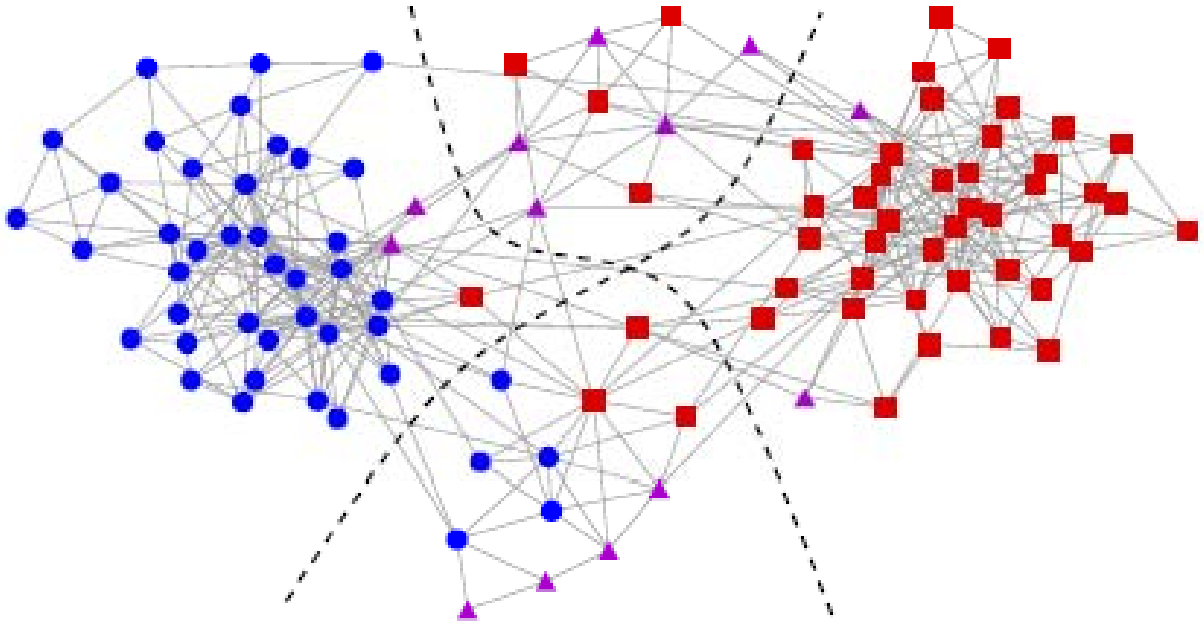


FIG. 3: Krebs' network of books on American politics. Vertices represent books and edges join books frequently purchased by the same readers. Dotted lines divide the four communities found by our algorithm and shapes represent the political alignment of the books: circles (blue) are liberal, squares (red) are conservative, triangles (purple) are centrist or unaligned.

Comparison to other methods

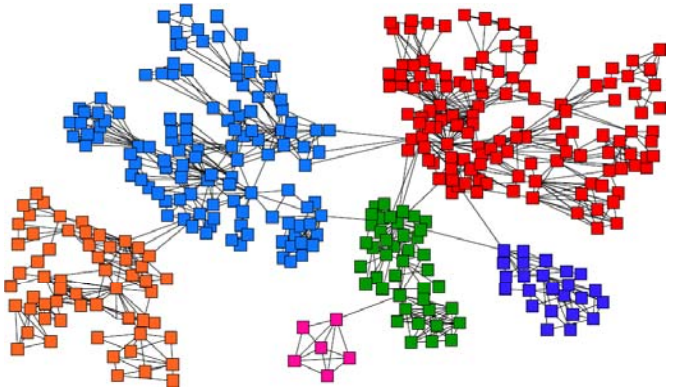
network	size n	modularity Q GN	CNM	DA	this paper
karate	34	0.401	0.381	0.419	0.419
jazz musicians	198	0.405	0.439	0.445	0.442
metabolic	453	0.403	0.402	0.434	0.435
email	1133	0.532	0.494	0.574	0.572
key signing	10 680	0.816	0.733	0.846	0.855
physicists	27 519	–	0.668	0.679	0.723

TABLE I: Comparison of modularities for the network divisions found by the algorithm described here and three other previously published methods as described in the text, for six networks of varying sizes. The networks are, in order, the

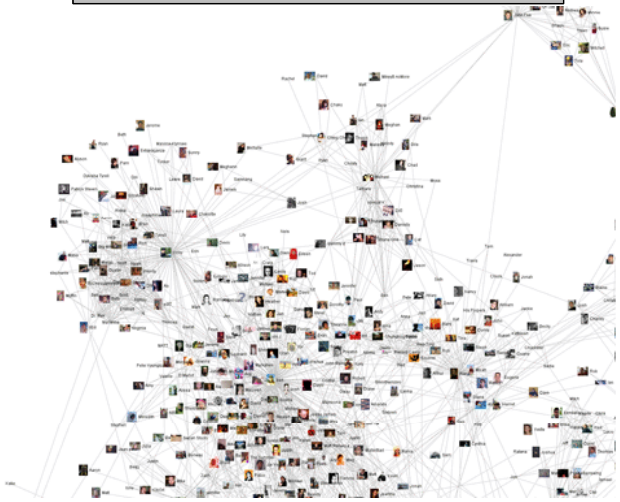
CN = Betweenness
CNM = Greedy
DA = External Optimization

Community structure in networks

- What is the cluster structure of networks?
- How does it scale from small to very large networks?
- How to think about clusters in large networks?



Physics collaborations



Part of a large social network

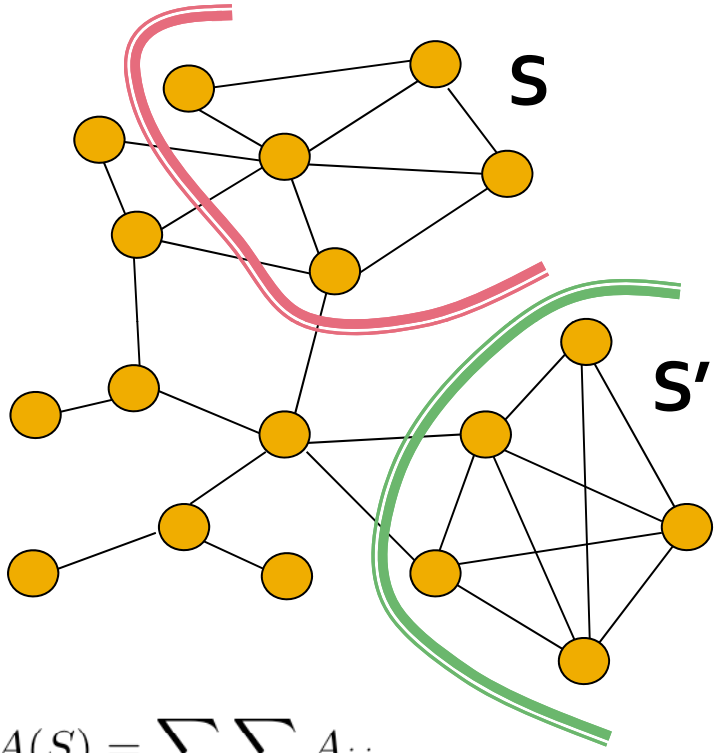
Objective function

What is a good cluster?

- Many edges internally
- Few pointing outside

Formally, **conductance:**

$$\phi(S) = \frac{\sum_{i \in S, j \notin S} A_{ij}}{\min\{A(S), A(\overline{S})\}}$$



$$A(S) = \sum_{i \in S} \sum_{j \in V} A_{ij}$$

Small Φ(S) corresponds to good clusters

[Leskovec et al. WWW '08]

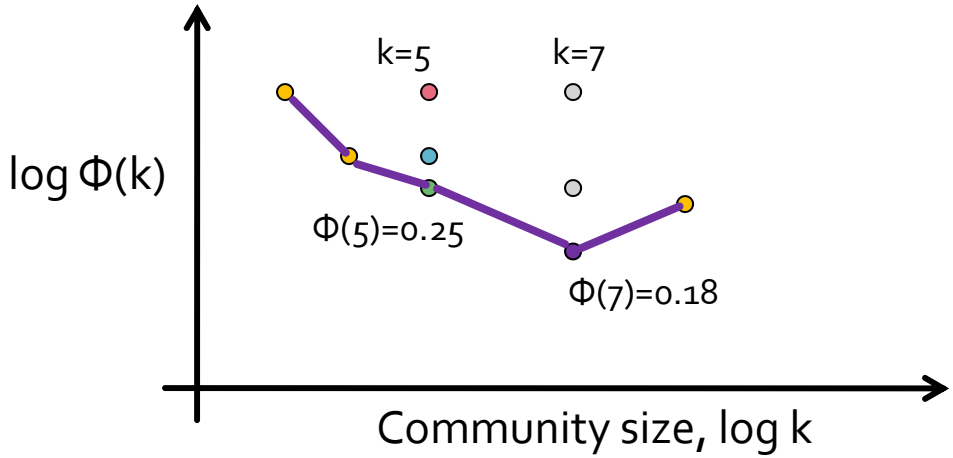
Network Community Profile Plot

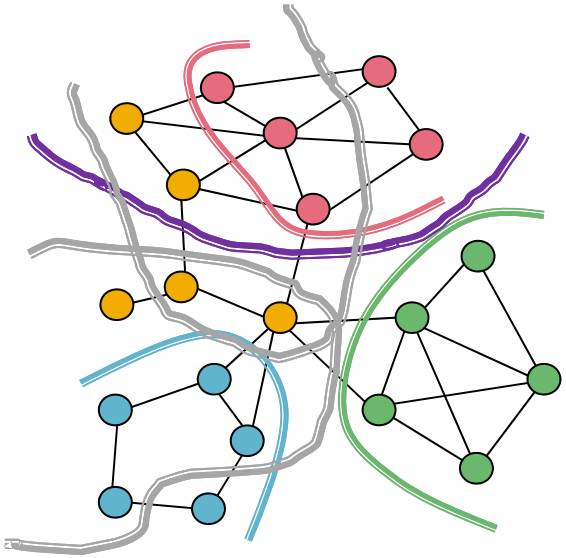
- Define:

  Network community profile (**NCP**) plot

  Plot the score of best community of size *k*

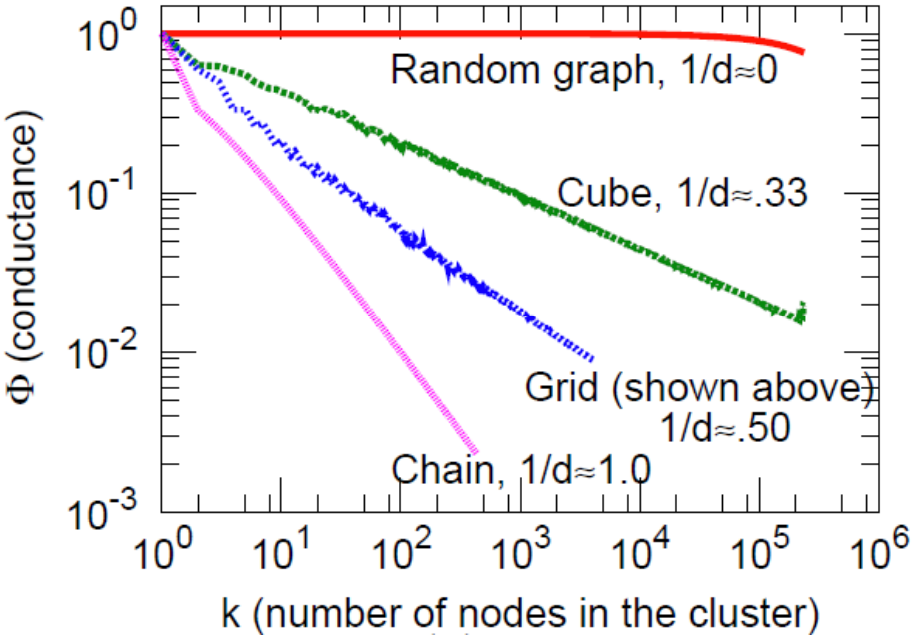
$$\Phi(k) = \min_{S \subset V, |S| = k} \phi(S)$$



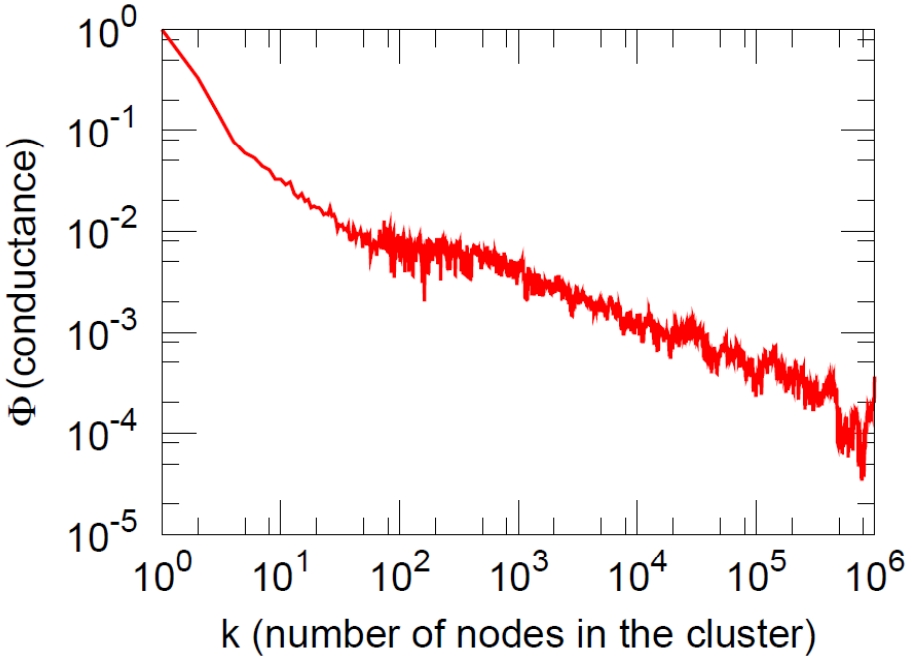


NCP plot: Meshes

- Meshes, grids, dense random graphs:



d-dimensional meshes

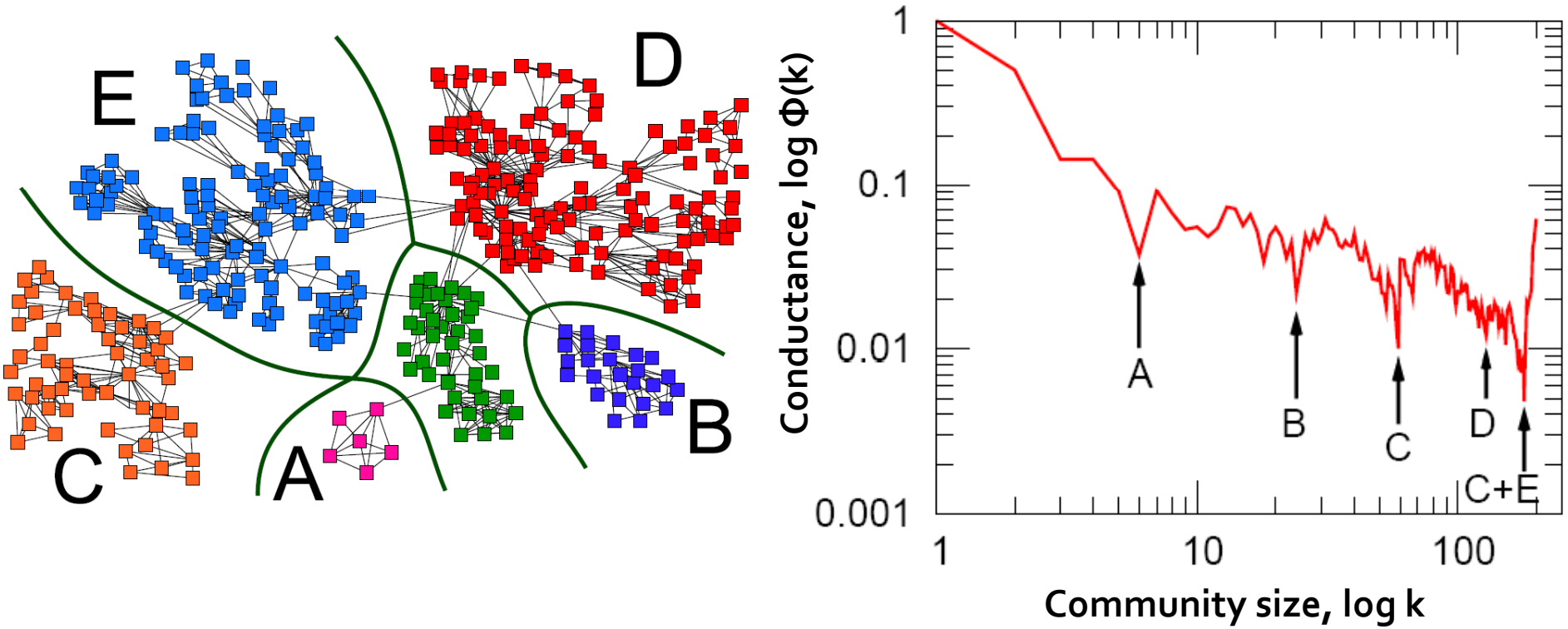


California road network

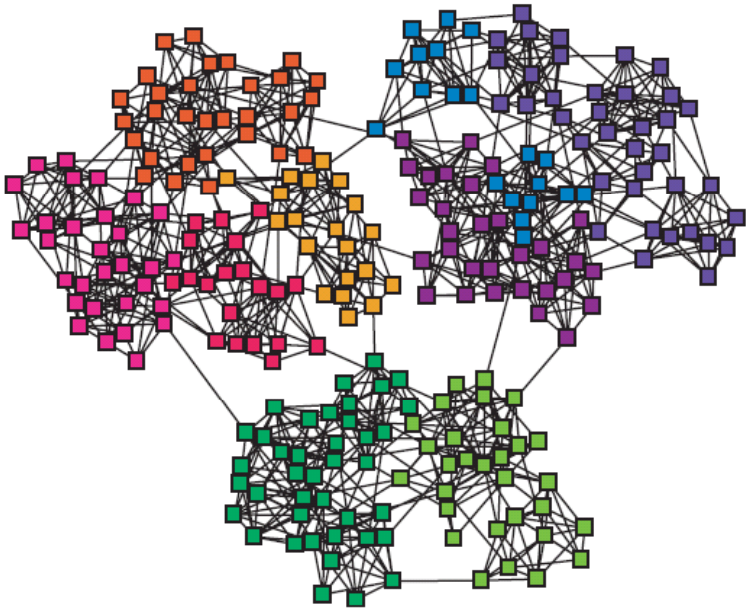
[Leskovec et al. WWW '08]

NCP plot: Network Science

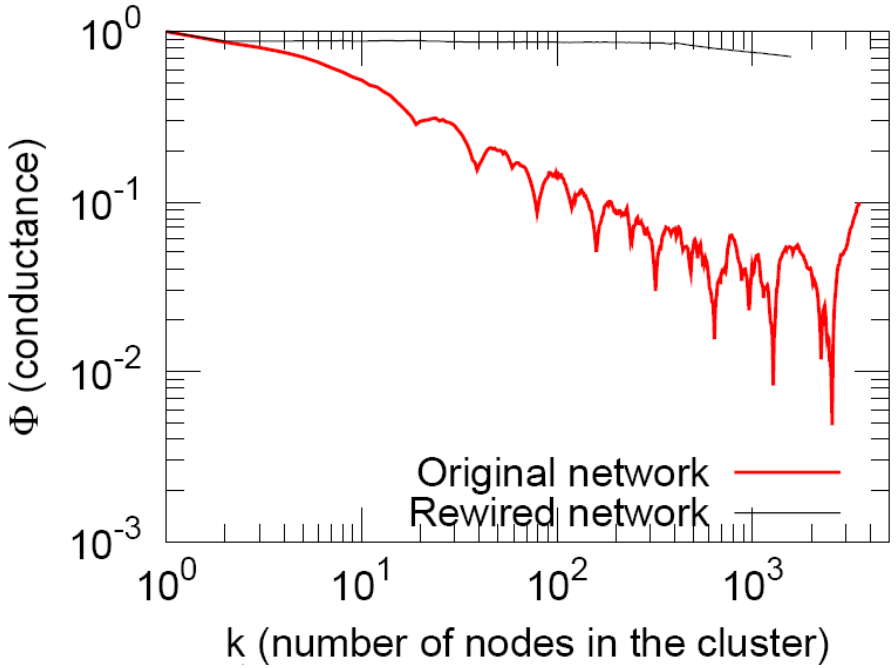
- Collaborations between scientists in networks
  [Newman, 2005]



NCP plot: Hierarchical networks



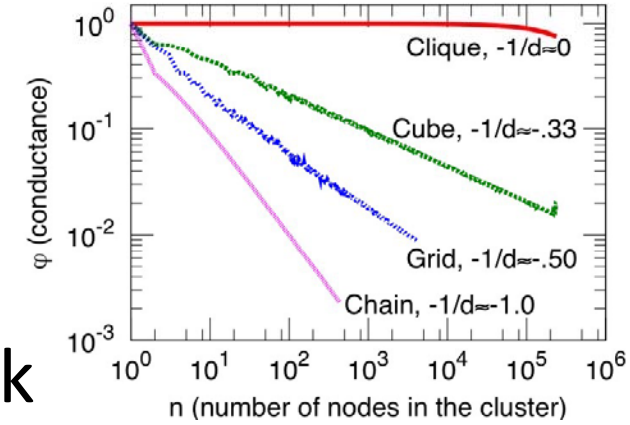
[Clauset-Moore-Newman 08]



Natural hypothesis

Natural hypothesis about NCP:

- NCP of real networks slopes downward
- Slope of the NCP corresponds to the dimensionality of the network



What about large networks?

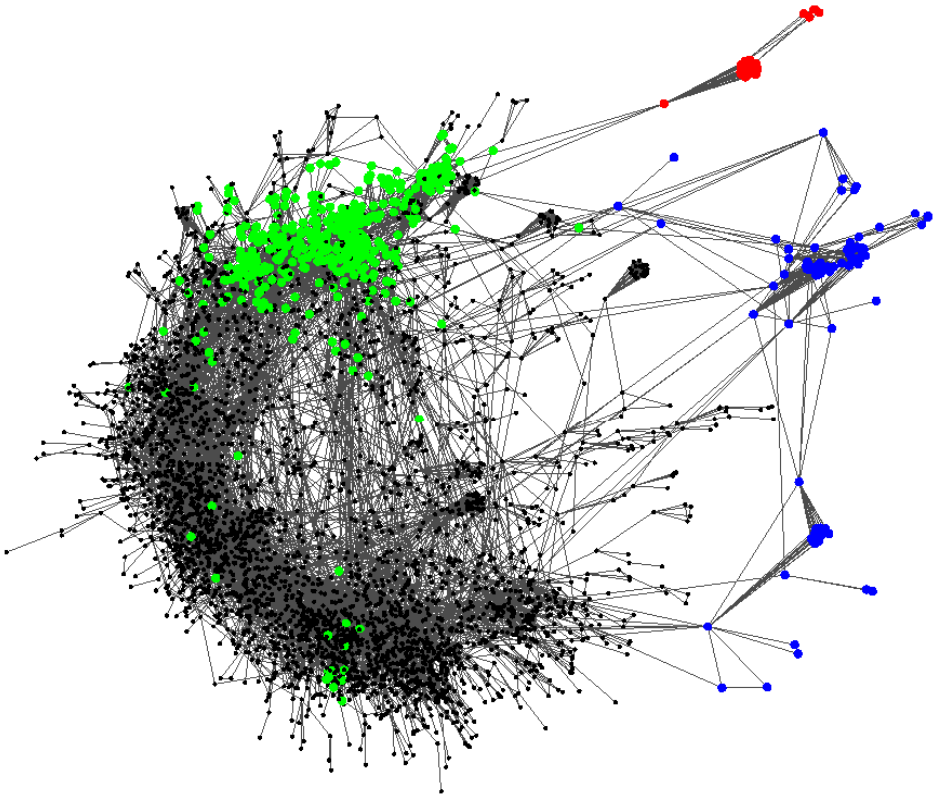
• Social nets	Nodes	Edges	Description
LIVEJOURNAL	4,843,953	42,845,684	Blog friendships [5]
EPINIONS	75,877	405,739	Trust network [28]
CA-DBLP	317,080	1,049,866	Co-authorship [5]
• Information (citation) networks			
CIT-HEP-TH	27,400	352,021	Arxiv hep-th [14]
AMAZONPROD	524,371	1,491,793	Amazon products [8]
• Web graphs			
WEB-GOOGLE	855,802	4,291,352	Google web graph
WEB-WT10G	1,458,316	6,225,033	TREC WT10G
• Bipartite affiliation (authors-to-papers) networks			
ATP-DBLP	615,678	944,456	DBLP [21]
ATM-IMDB	2,076,9		
• Internet networks			
ASSKITTER	1,719,0		
GNUTELLA	62,5		

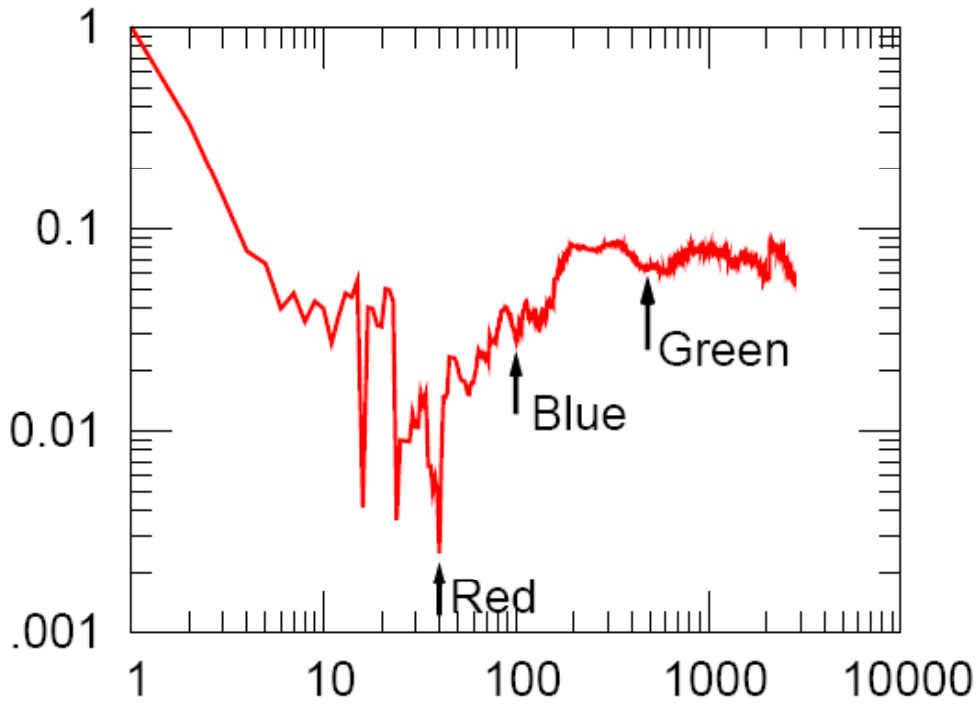
We examined more than **100 large networks**

[Leskovec et al. WWW '08]

Large networks: Very different

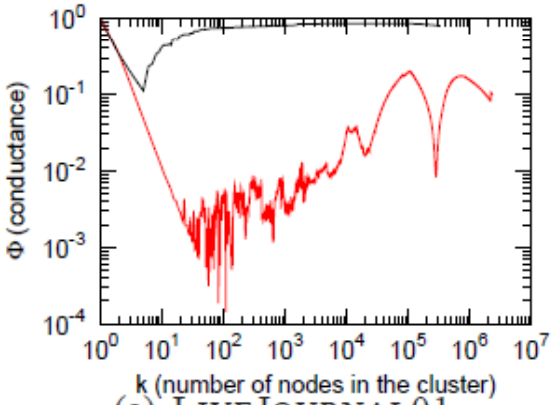
Typical example: General Relativity collaborations (n=4,158, m=13,422)



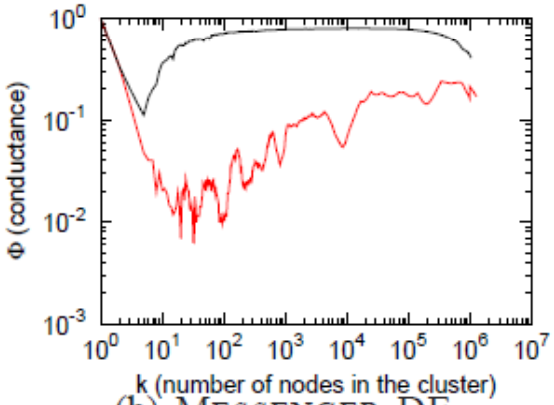


[Leskovec et al. WWW '08]

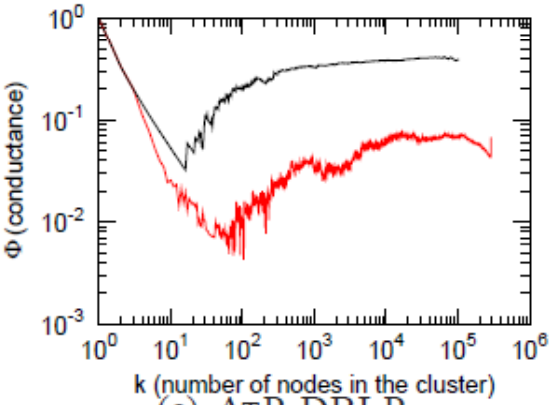
More NCP plots of networks



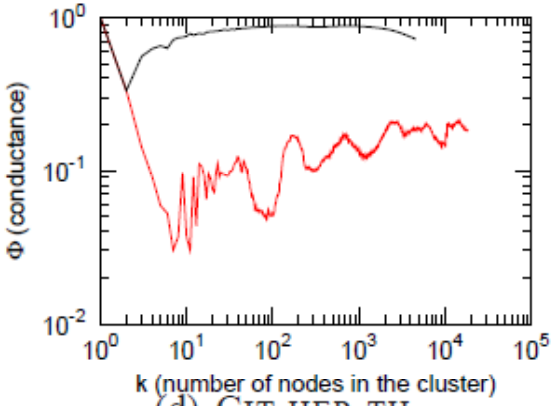
(a) LIVEJOURNAL01



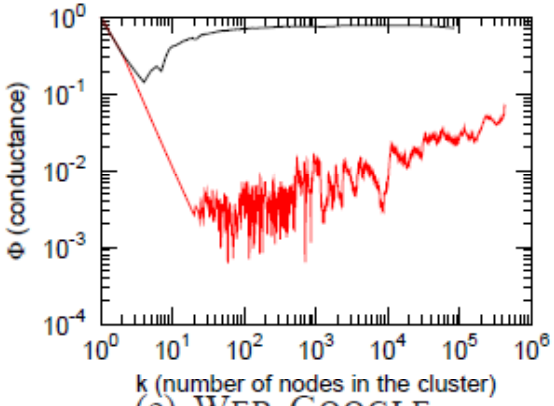
(b) MESSENGER-DE



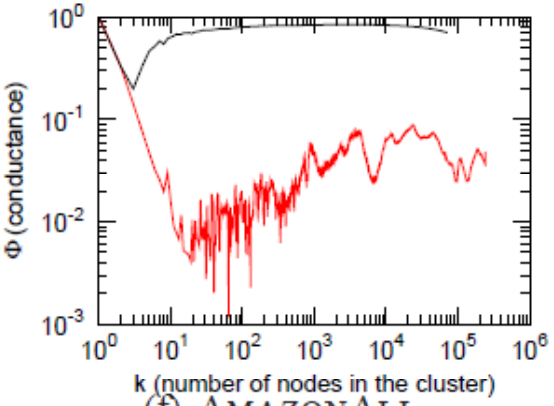
(c) ATP-DBLP



(d) CIT-HEP-TH



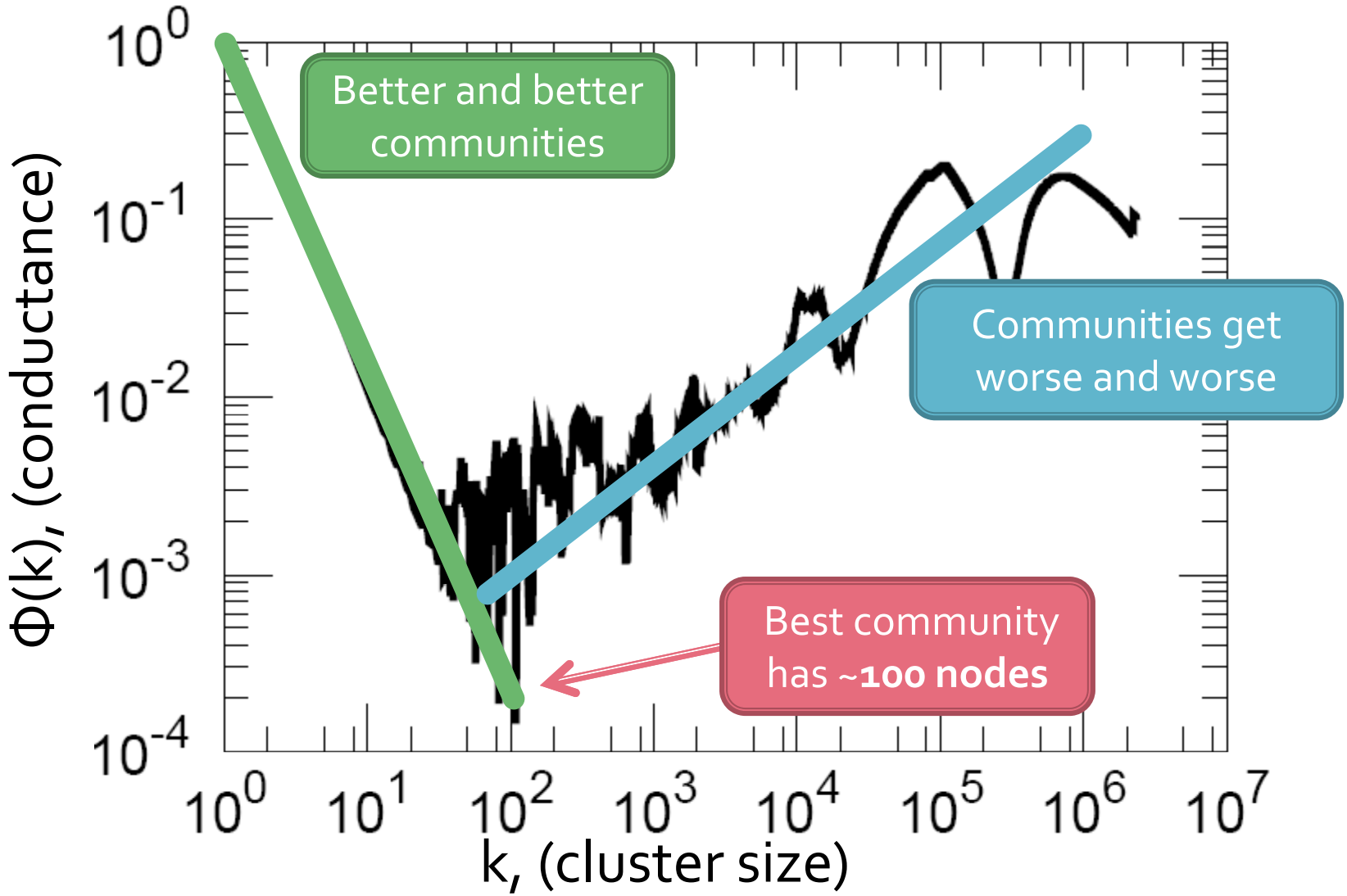
(e) WEB-GOOGLE



(f) AMAZONALL

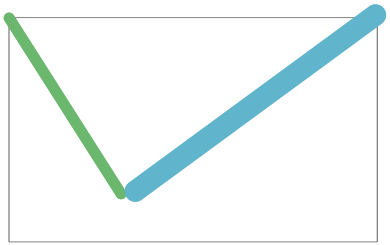
[Leskovec et al. WWW '08]

NCP: LiveJournal (n=5m, m=42m)

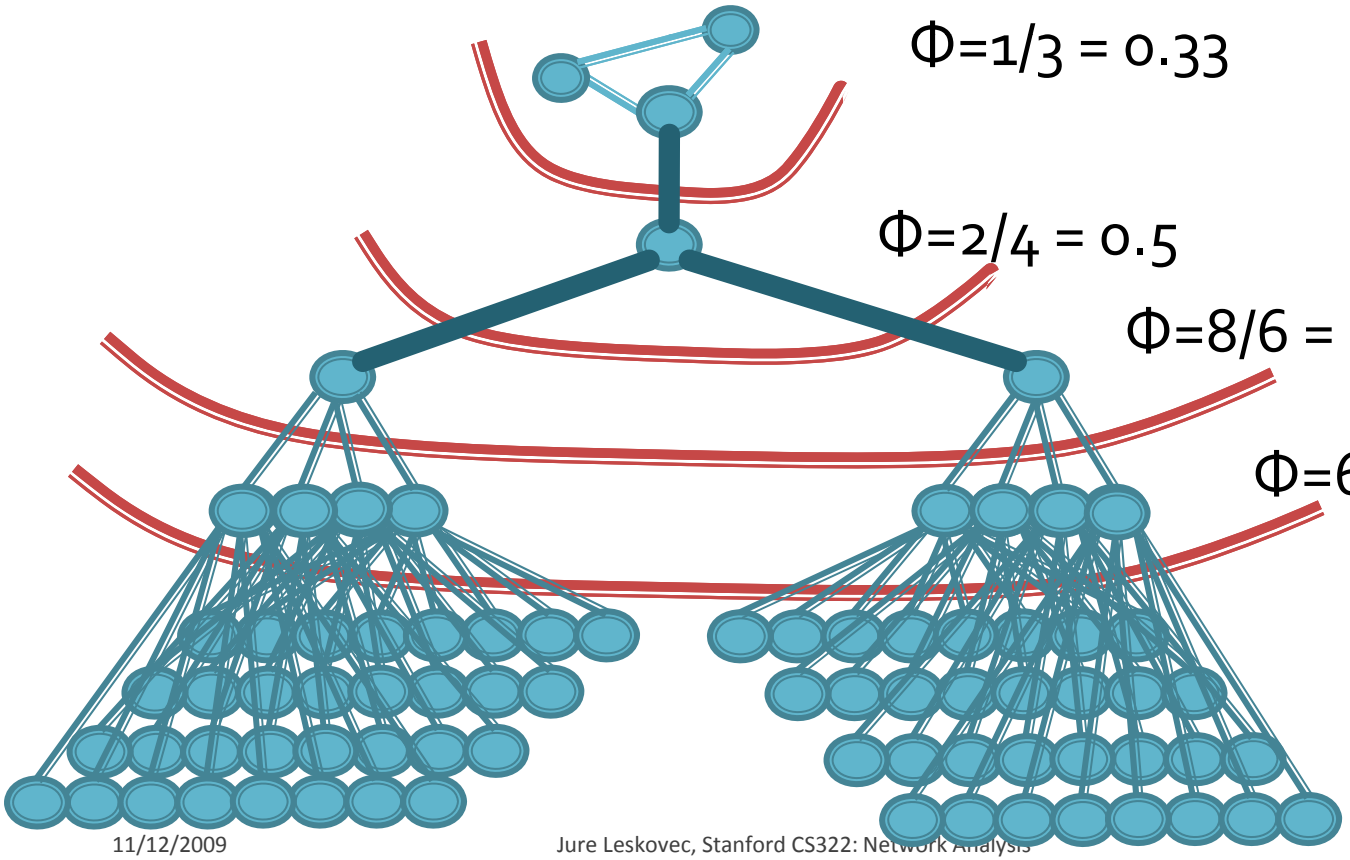


Explanation: Upward part

- Each successive edge inside the community costs **more** cut-edges



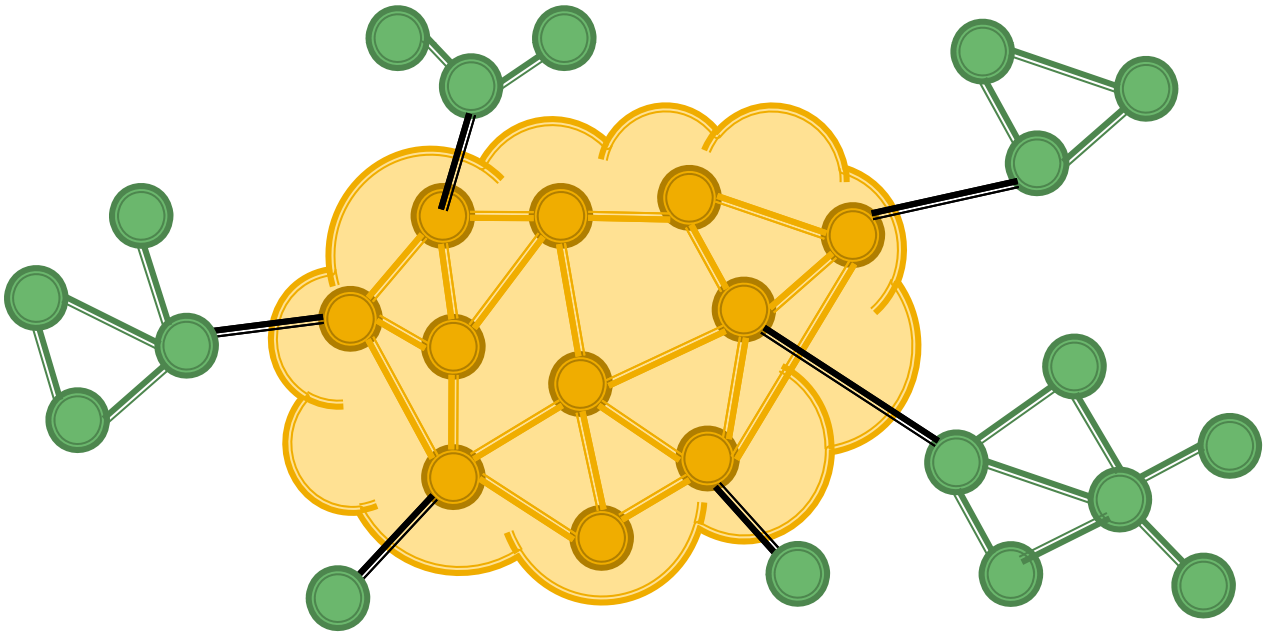
NCP plot

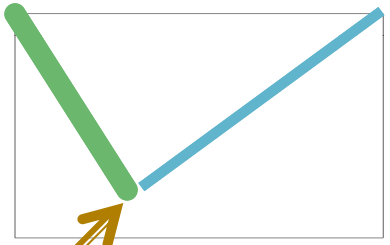


Each node has twice as many children

Explanation: Downward part

- Empirically we note that best clusters (call them whiskers) are barely connected to the network



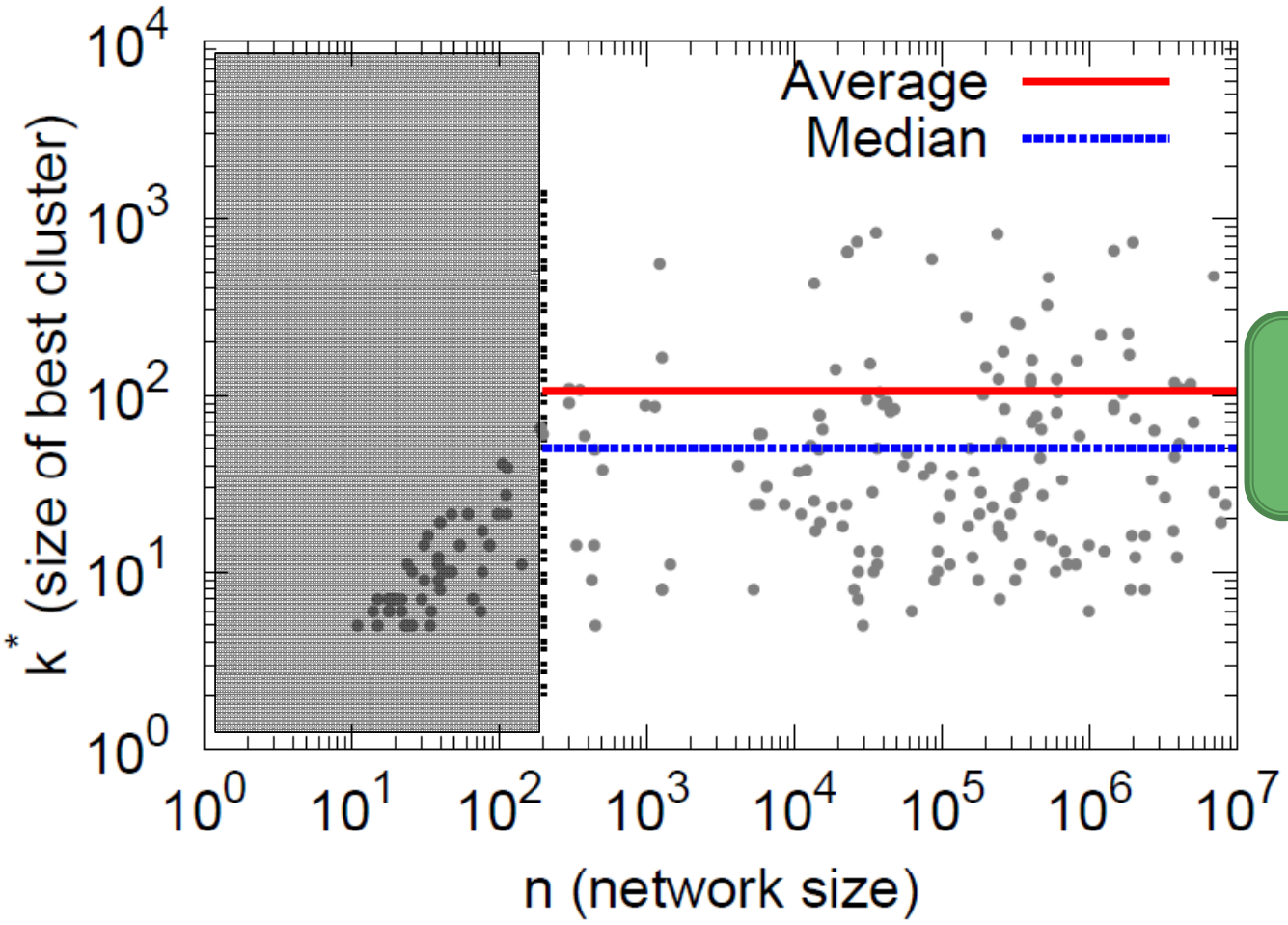


Best cluster.
How does it scale with network size?

⇒ Core-periphery structure

Cluster size is independent of the network size

[Leskovec et al. Arxiv '09]

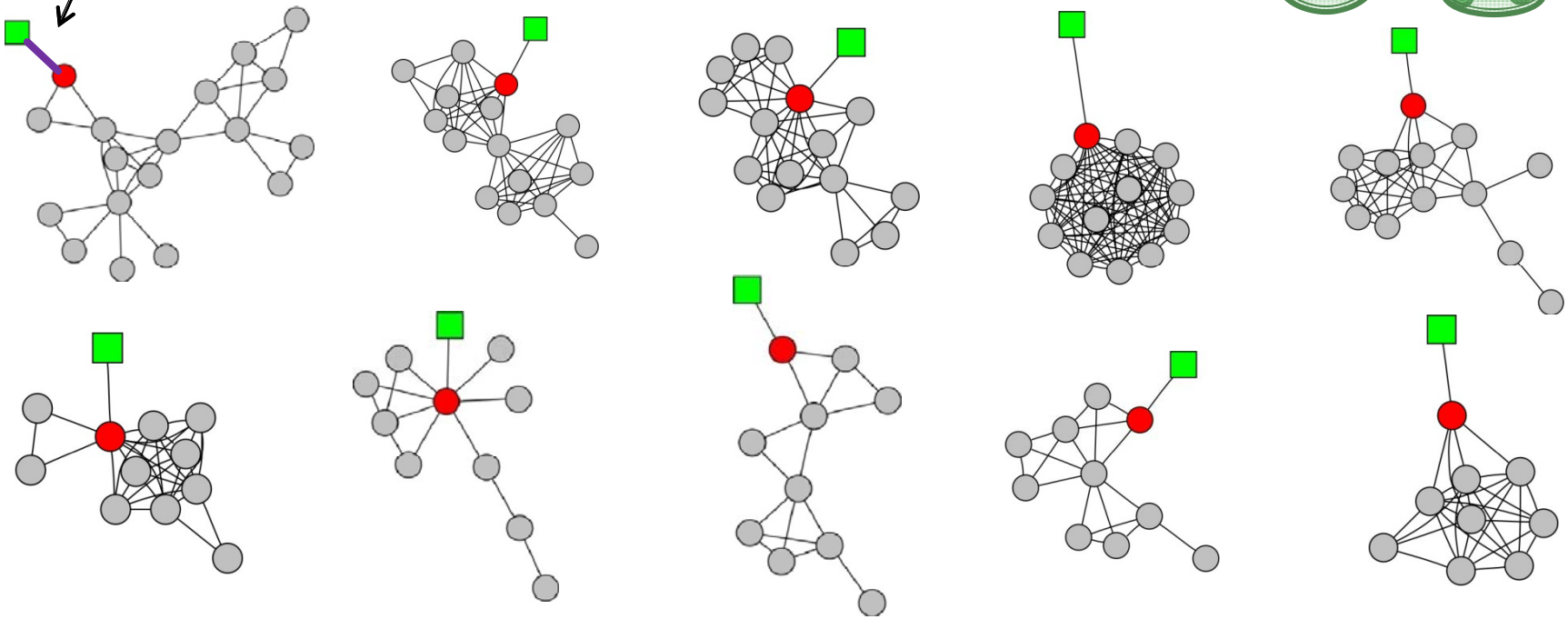


Practically **constant!**

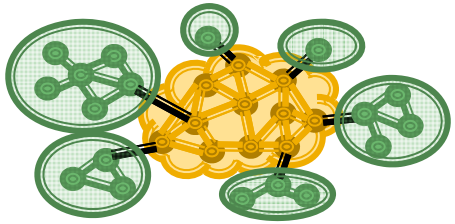
- Each dot is a different network

Whisker shapes

Edge to cut

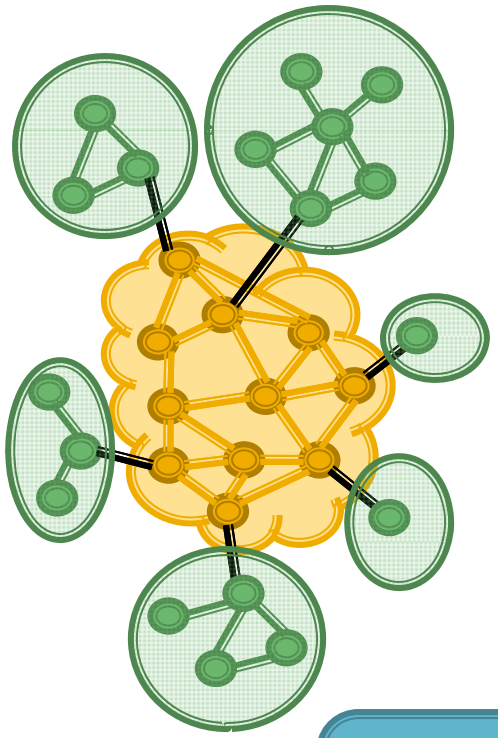


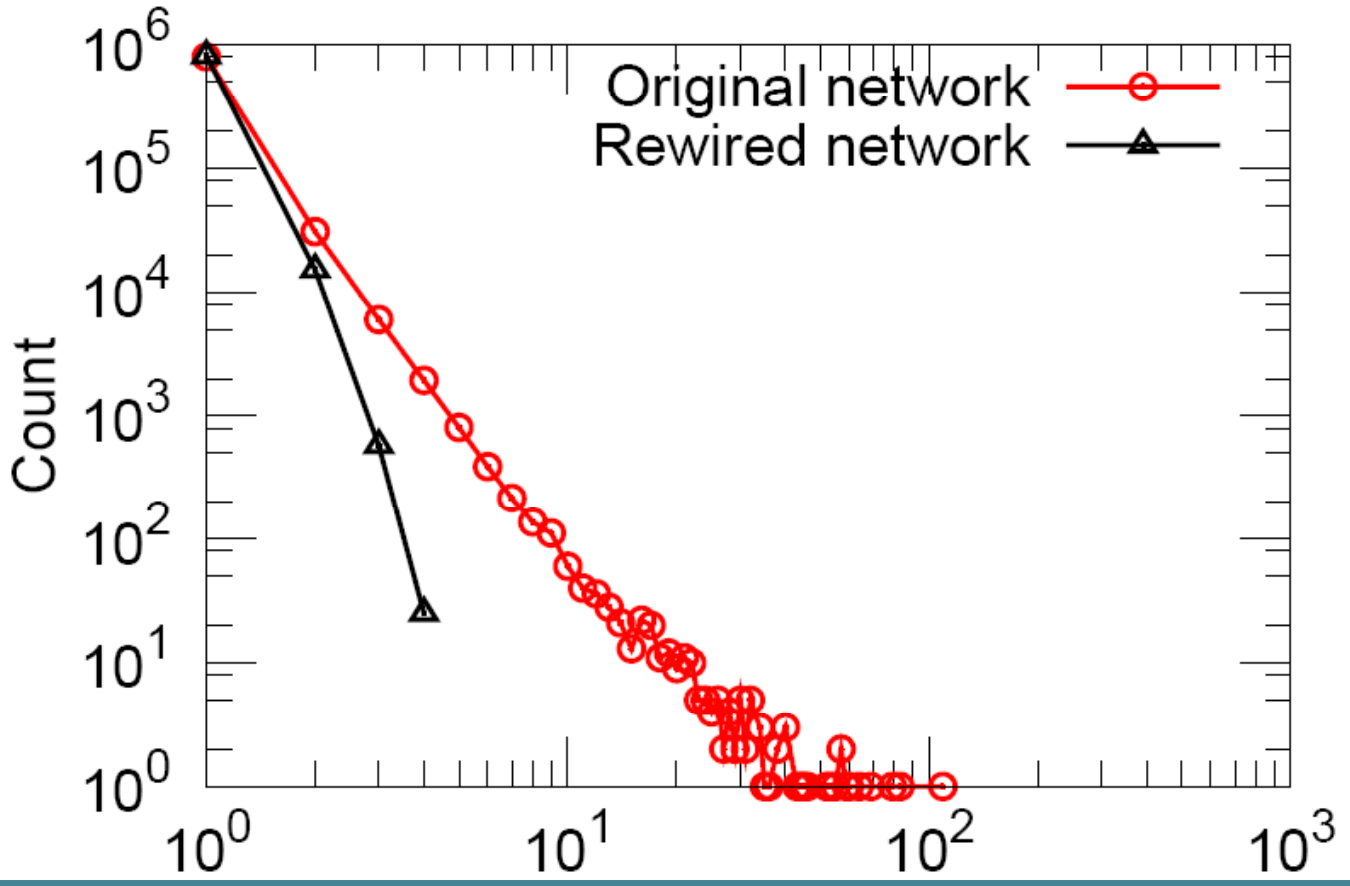
Whiskers:



Whiskers in real networks are non-trivial (richer than trees)

Whiskers: Result of edge sparsity?



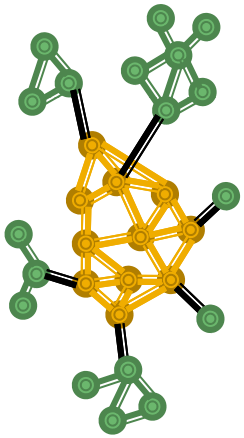


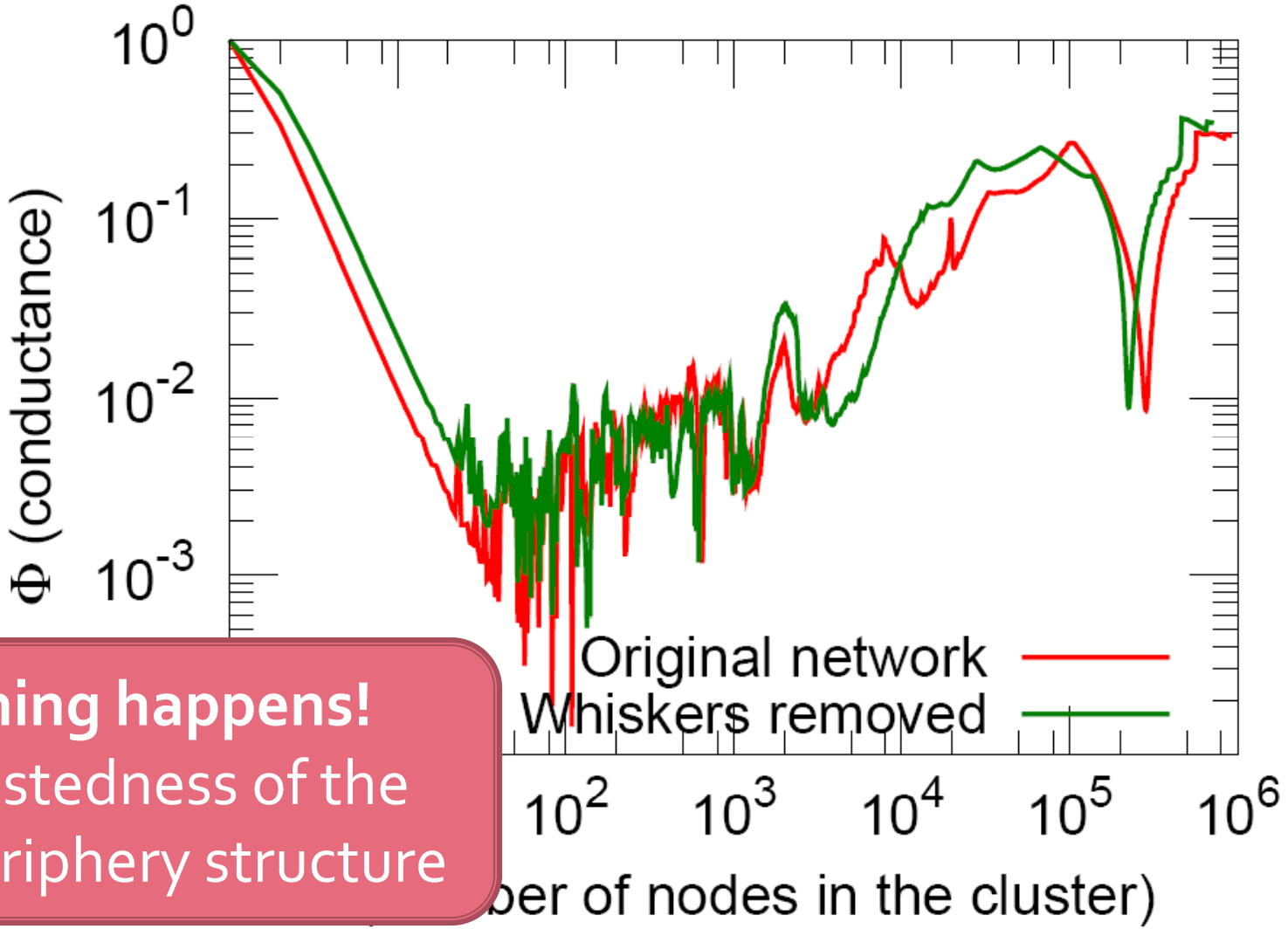
Whiske

Whiskers in real networks are **larger** than expected based on density and degree sequence

[Leskovec et al. Arxiv '09]

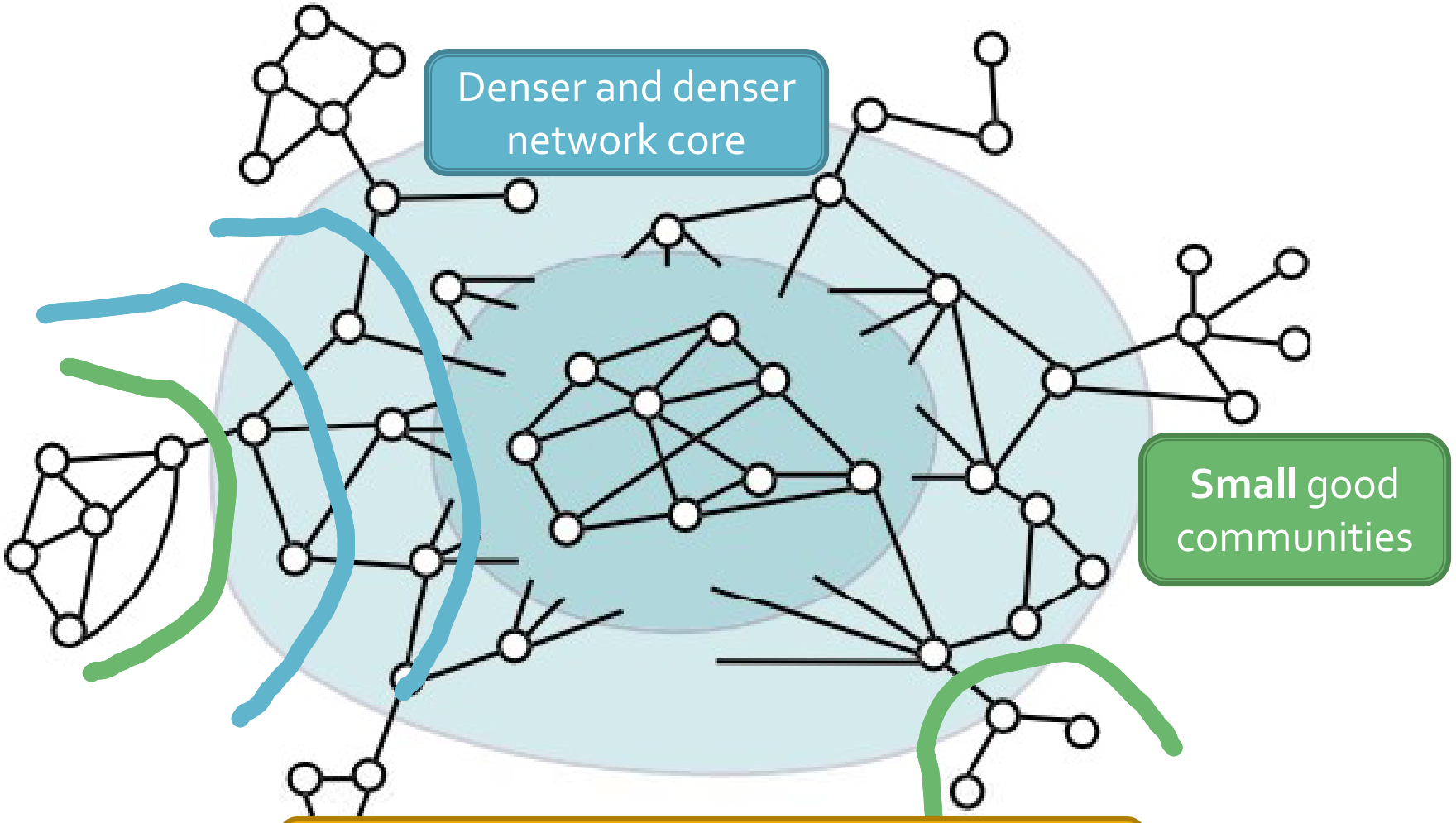
What if we remove whiskers?





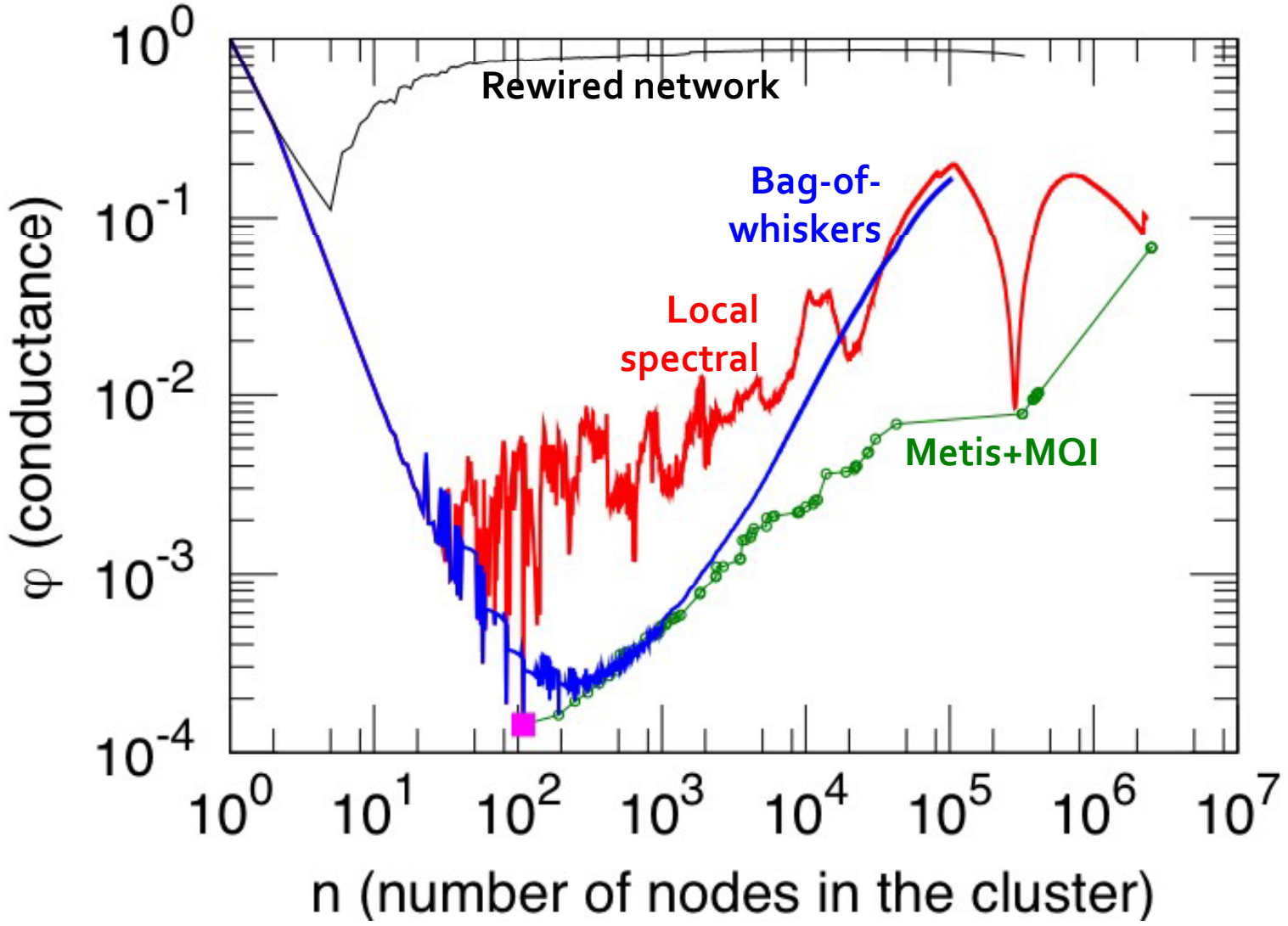
Nothing happens!
⇒ Nestedness of the core-periphery structure

Nested core-periphery

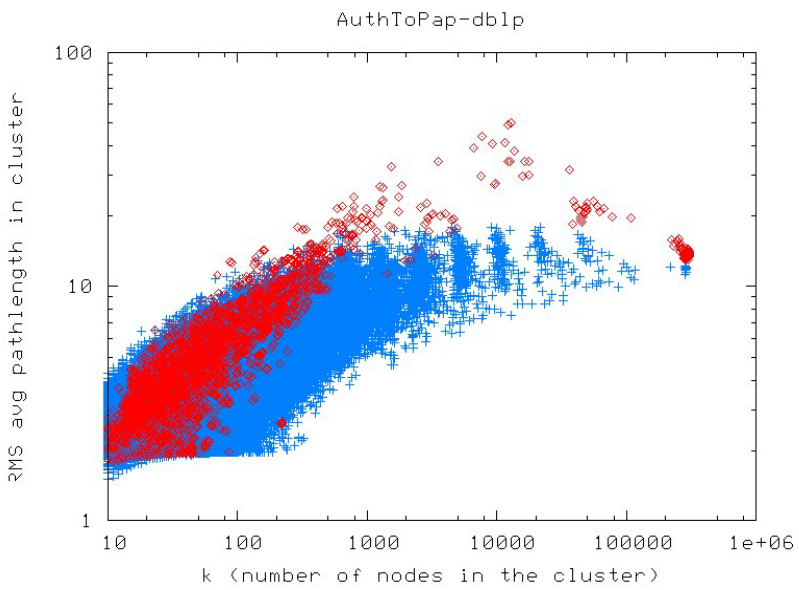


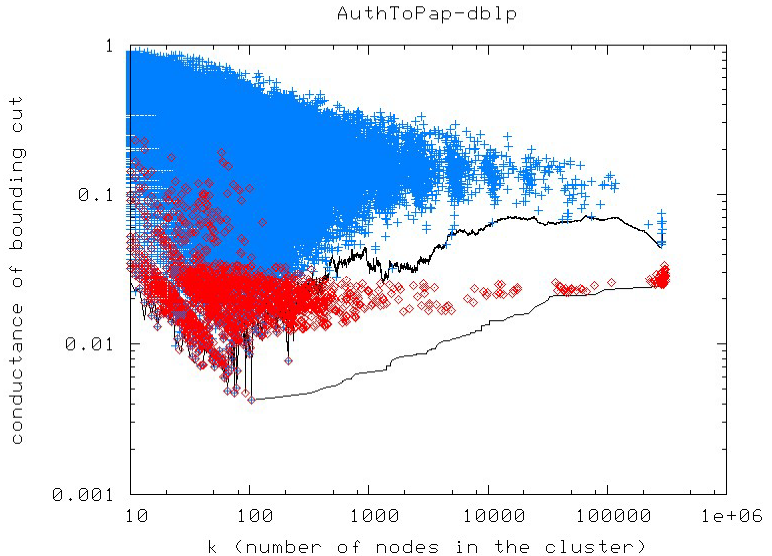
Caveat: How do we cut?

NCP: Complete picture

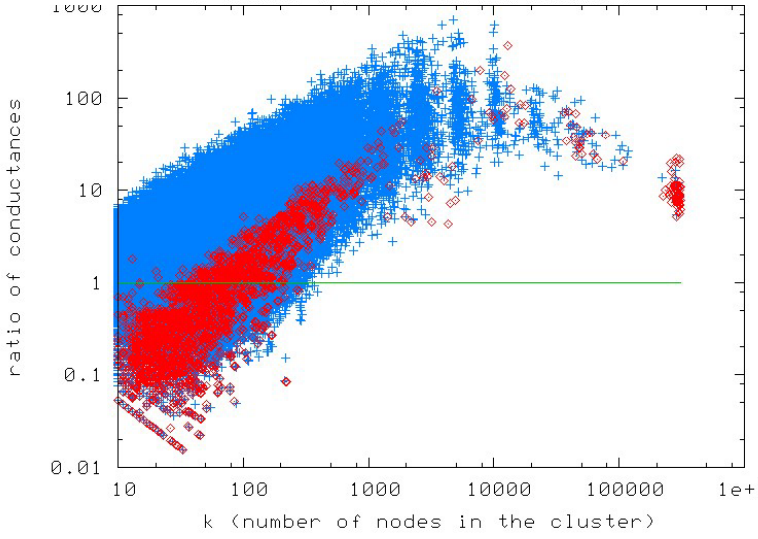


Regularized and non-regularized communities





Dots are connected clusters



ext/int

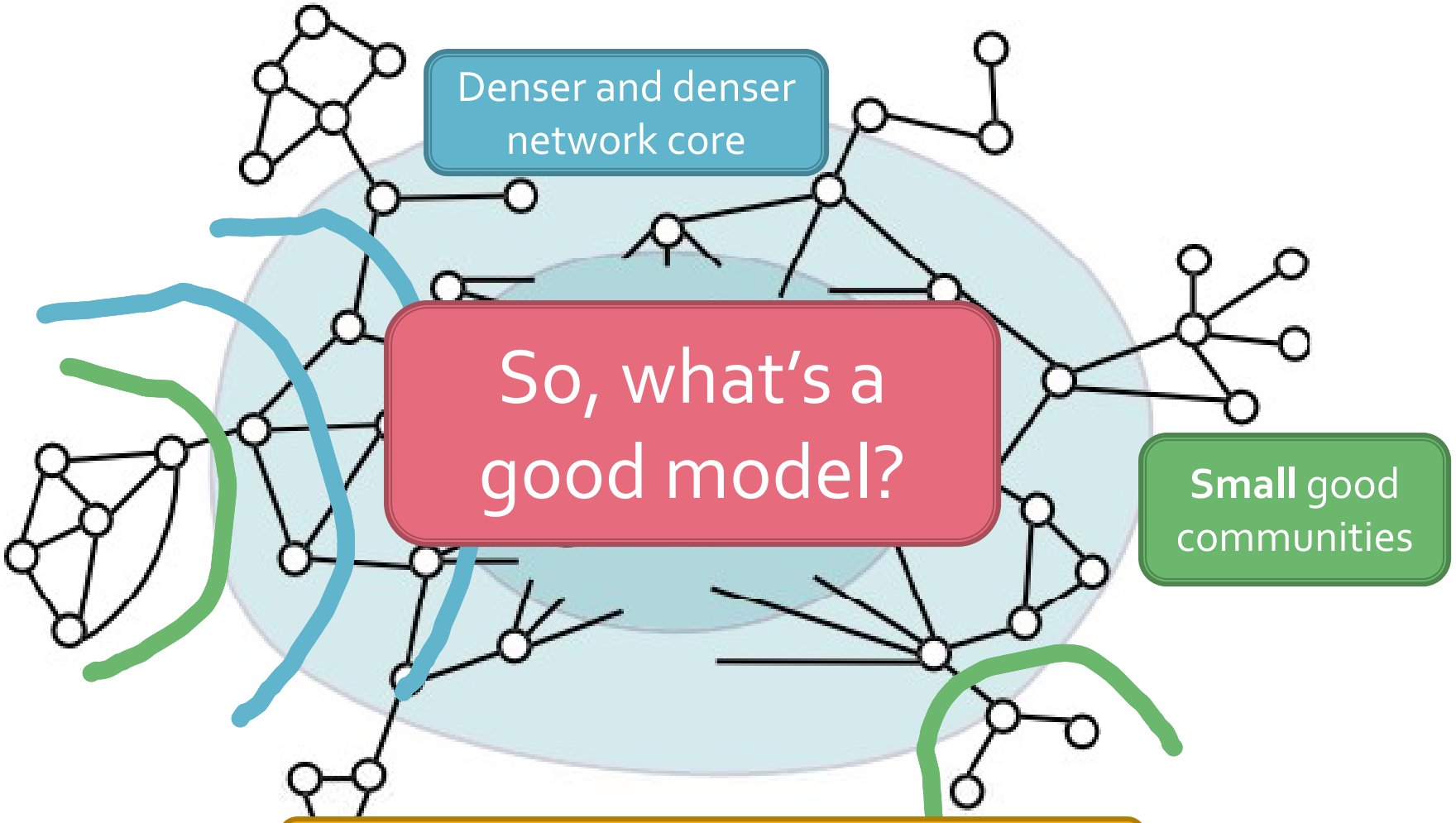
- Metis+MQI (red) gives sets with better conductance.
- Local Spectral (blue) gives tighter and more well-rounded sets.

Regularized and non-regularized communities

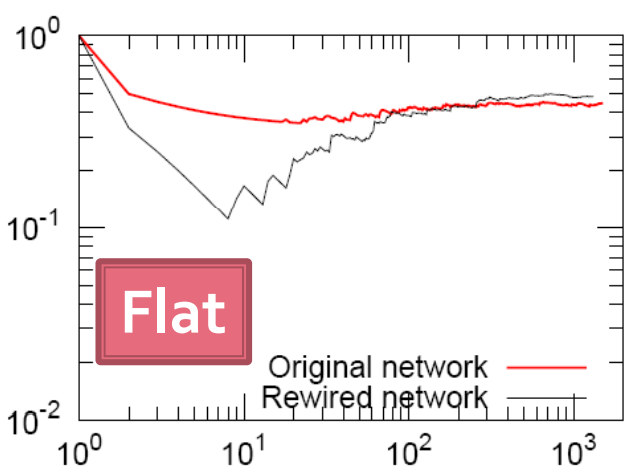
Two ca. 500 node communities from Local Spectral:

Two ca. 500 node communities from Metis+MQI:

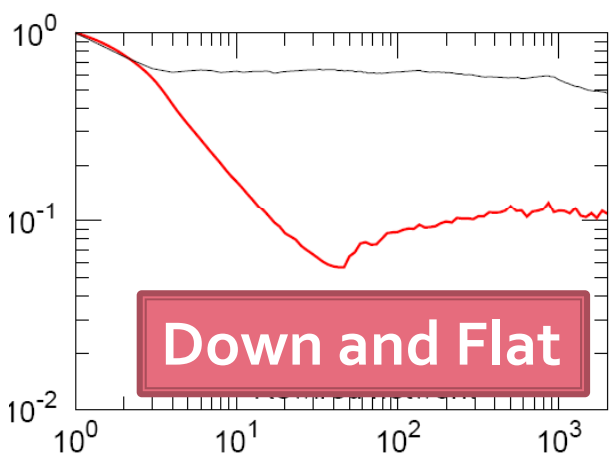
Nested core-periphery



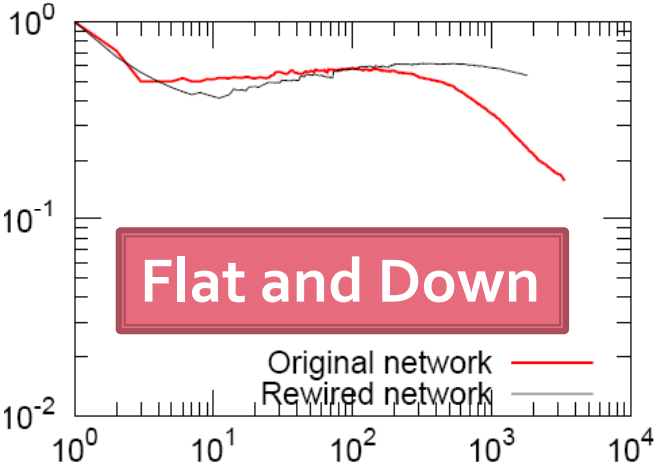
Modeling nested core-periphery



Pref. attachment



Small World

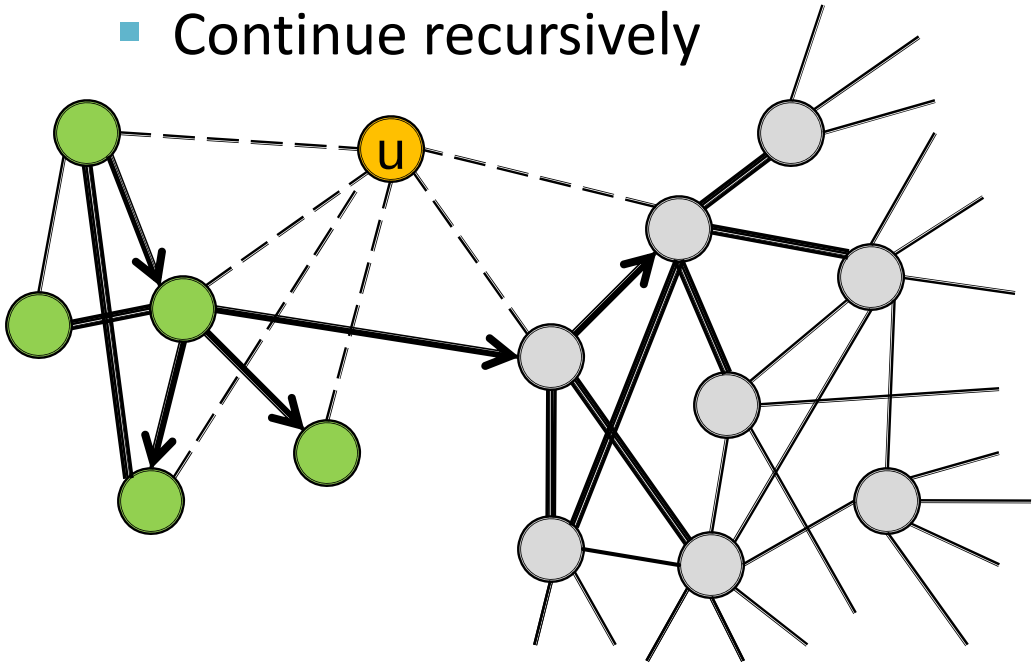


Geometric Pref. Attachment

- None of the common models works!

Forest Fire model works!

- **Forest Fire:**
  connections spread like a fire
  - New node joins the network
  - Selects a seed node
  - Connects to some of its neighbors
  - Continue recursively

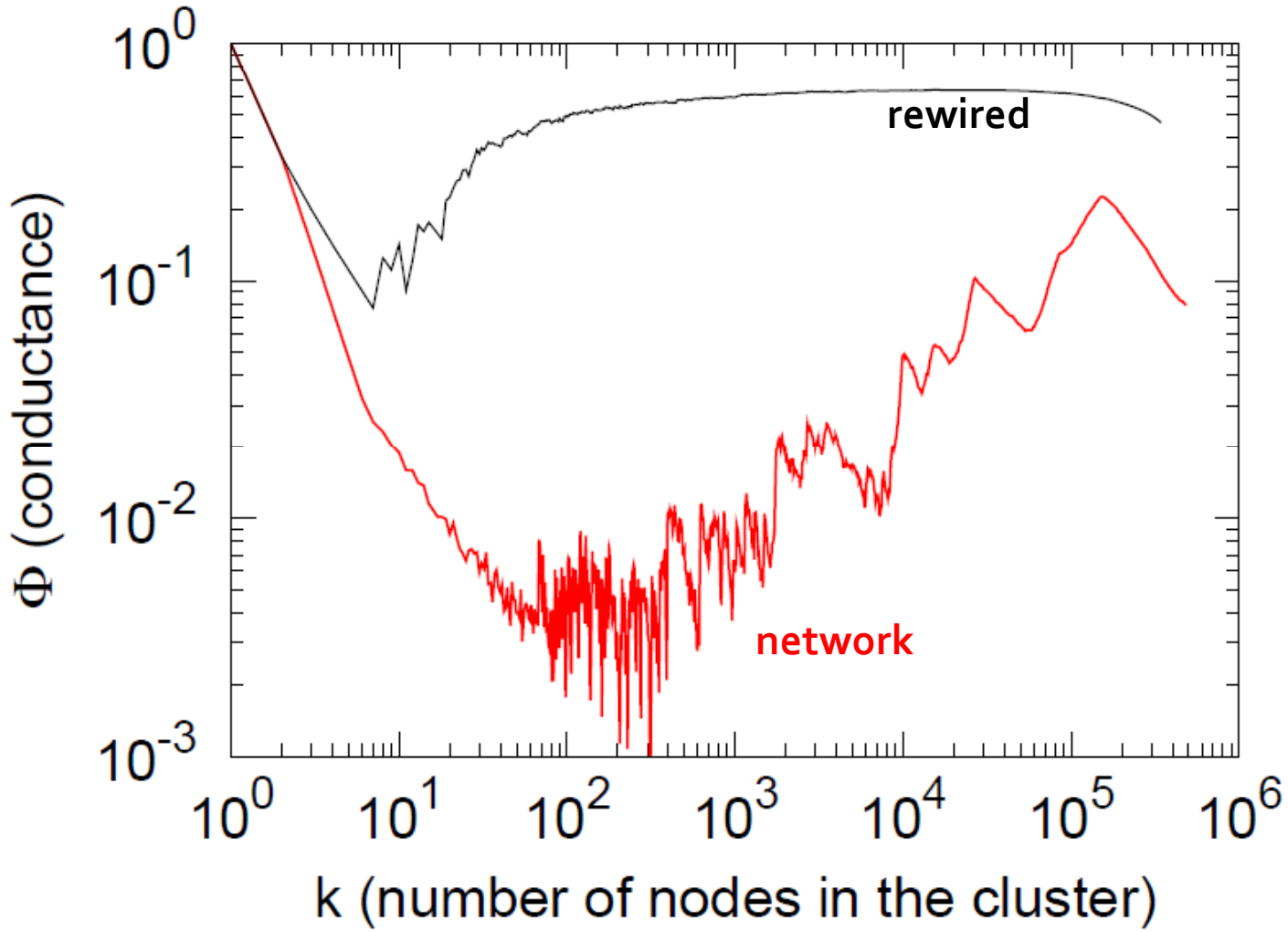


Model ingredients:

- Preferential attachment: second neighbor is not uniform at random
- Copying flavor: - since we burn seed's neighbors
- Hierarchical flavor:- burn around the seed node
- "Local" flavor:- burn "near" the seed node

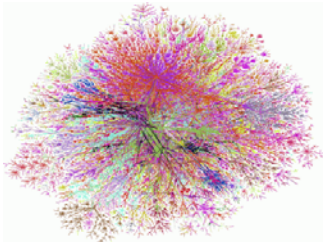
As cluster **grows** it **blends** into the **core** of the **network**

Forest Fire NCP plot

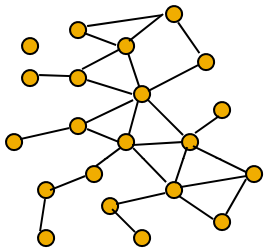


A more general question

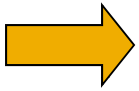
- Want to generate realistic networks:

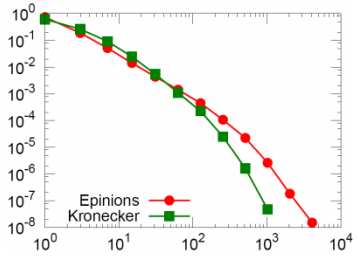


Given a real network



Generate a synthetic network



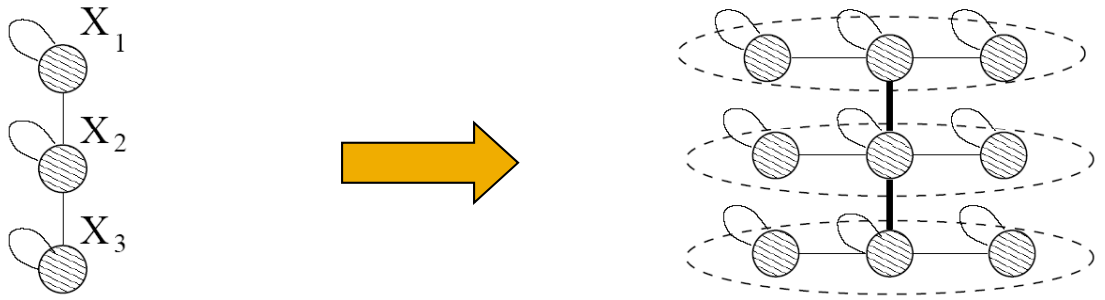


Compare graphs properties, *e.g.*, degree distribution

- Why synthetic graphs?
  - Anomaly detection, Simulations, Predictions, Null-model, Sharing privacy sensitive graphs, …
- **Q:** Which network properties do we care about?
- **Q:** What is a good model and how do we **fit** it?

Idea: Recursive graph generation

- Intuition: self-similarity leads to power-laws
- Try to mimic recursive graph / community growth
- There are many obvious (but wrong) ways:



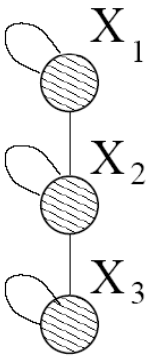
Initial graph

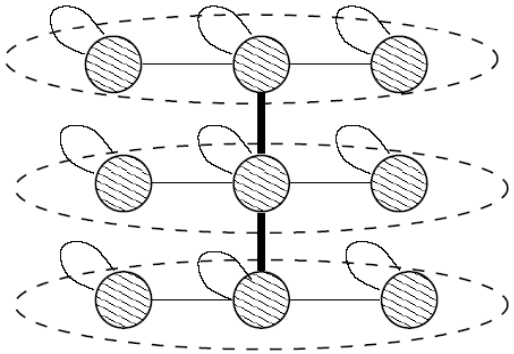
Recursive expansion

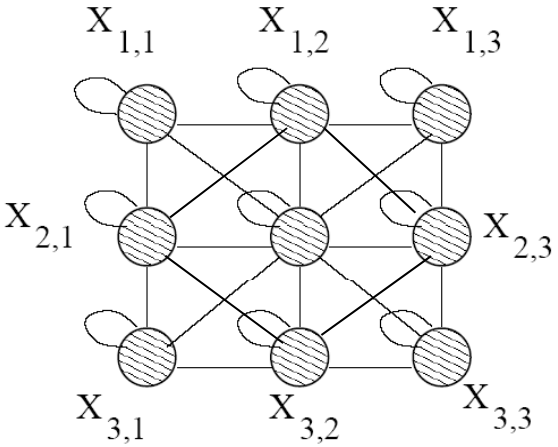
- Kronecker Product is a way of generating self-similar matrices

Kronecker product: Graph

[PKDD '05]







Intermediate stage

1	1	0
1	1	1
0	1	1

(3x3)

G_1

Adjacency matrix

G_1	G_1	0
G_1	G_1	G_1
0	G_1	G_1

(9x9)

$G_2 = G_1 \otimes G_1$

Adjacency matrix

Kronecker product: Definition

- Kronecker product of matrices A and B is given by

$$\underset{N\,x\,M}{\mathbf{C}} = \underset{K\,x\,L}{\mathbf{A} \otimes \mathbf{B}} \doteq \underset{N*K\,x\,M*L}{\begin{pmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \dots & a_{1,m}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \dots & a_{2,m}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}\mathbf{B} & a_{n,2}\mathbf{B} & \dots & a_{n,m}\mathbf{B} \end{pmatrix}}$$

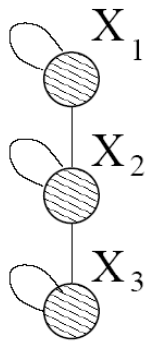
- We define a Kronecker product of two graphs as a Kronecker product of their adjacency matrices

[Leskovec et al. PKDD '05]

Kronecker graphs

- **Kronecker graph:** a growing sequence of graphs by iterating the Kronecker product

$$K_1^{[k]} = K_k = \underbrace{K_1 \otimes K_1 \otimes \ldots K_1}_{k \text{ times}} = K_{k-1} \otimes K_1$$



K_1

- Each Kronecker multiplication exponentially increases the size of the graph
- K_k has N_1^k nodes and E_1^k edges, so we get **densification**

- One can easily use multiple initiator matrices (K_1', K_1'', K_1''') that can be of different sizes

Kronecker product: Graph

[PKDD '05]

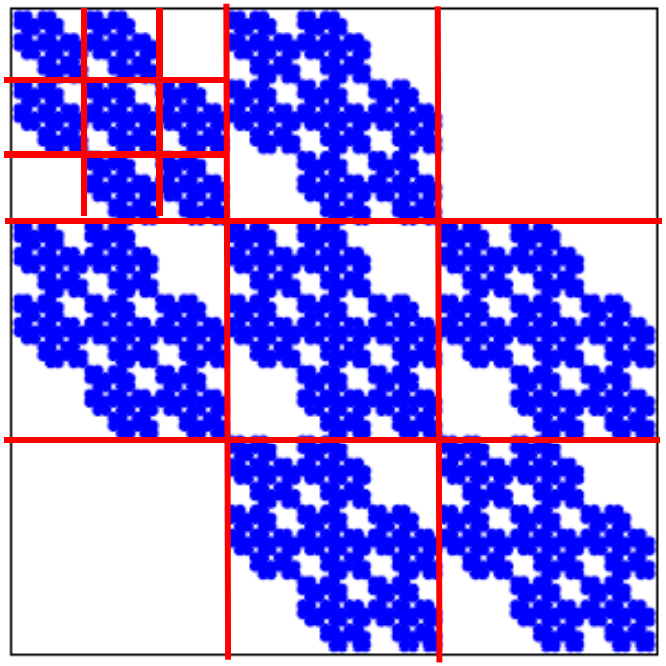
- Continuing multypling with K_1 we obtain K_4 and so on …

1	1	0
1	1	1
0	1	1

K_1

K_1	K_1	0
K_1	K_1	K_1
0	K_1	K_1

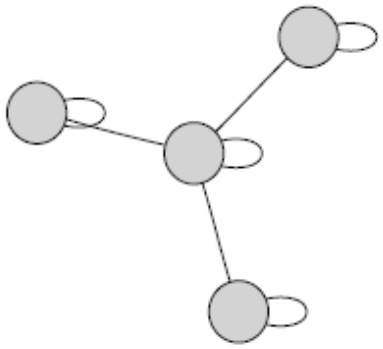
$K_2 = K_1 \otimes K_1$



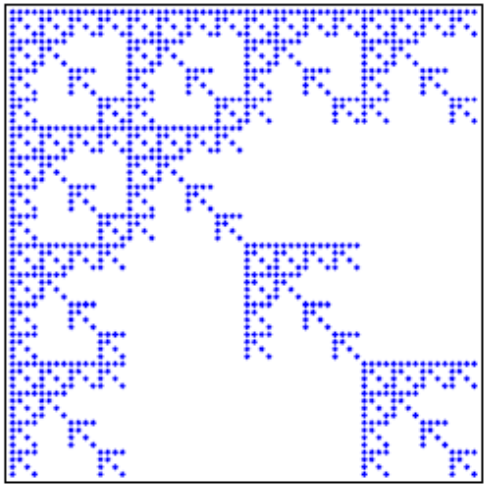
K_4 adjacency matrix

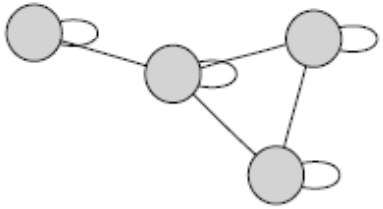
[Leskovec et al. PKDD '05]

Kronecker initiator matrices

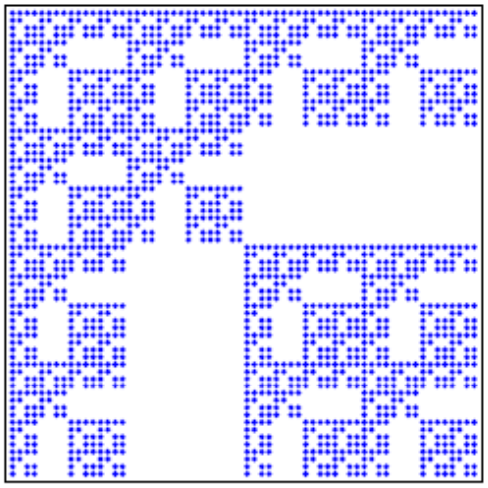


1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1





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



Initiator K_1 | K_1 adjacency matrix | K_3 adjacency matrix

Properties of Kronecker graphs

[PKDD '05]

- Kronecker graphs have many properties found in real networks:
  - Properties of static networks
    - Power-Law like Degree Distribution
    - Power-Law eigenvalue and eigenvector distribution
    - Constant Diameter
  - Properties of dynamic networks:
    - Densification Power Law
    - Shrinking/Stabilizing Diameter

[PKDD '05]

Constant Diameter – Proof Sketch

- Theorem: Constant diameter: If G_1 has diameter d then graph G_k also has diameter d

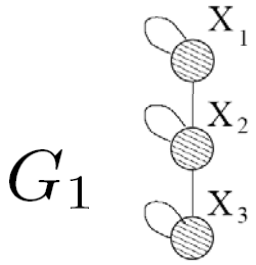
- Observation: Edges in Kronecker graphs:

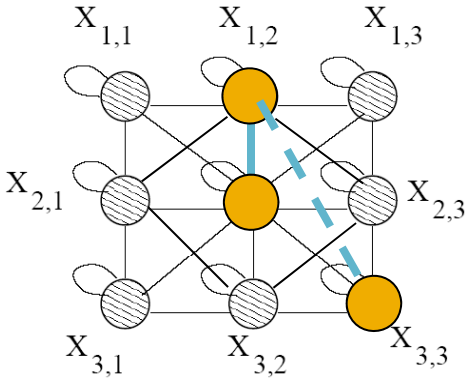
$$Edge\ (X_{ij}, X_{kl}) \in G \otimes H$$

$$iff\ (X_i, X_k) \in G\ and\ (X_j, X_l) \in H$$

where X are appropriate nodes

- Example:



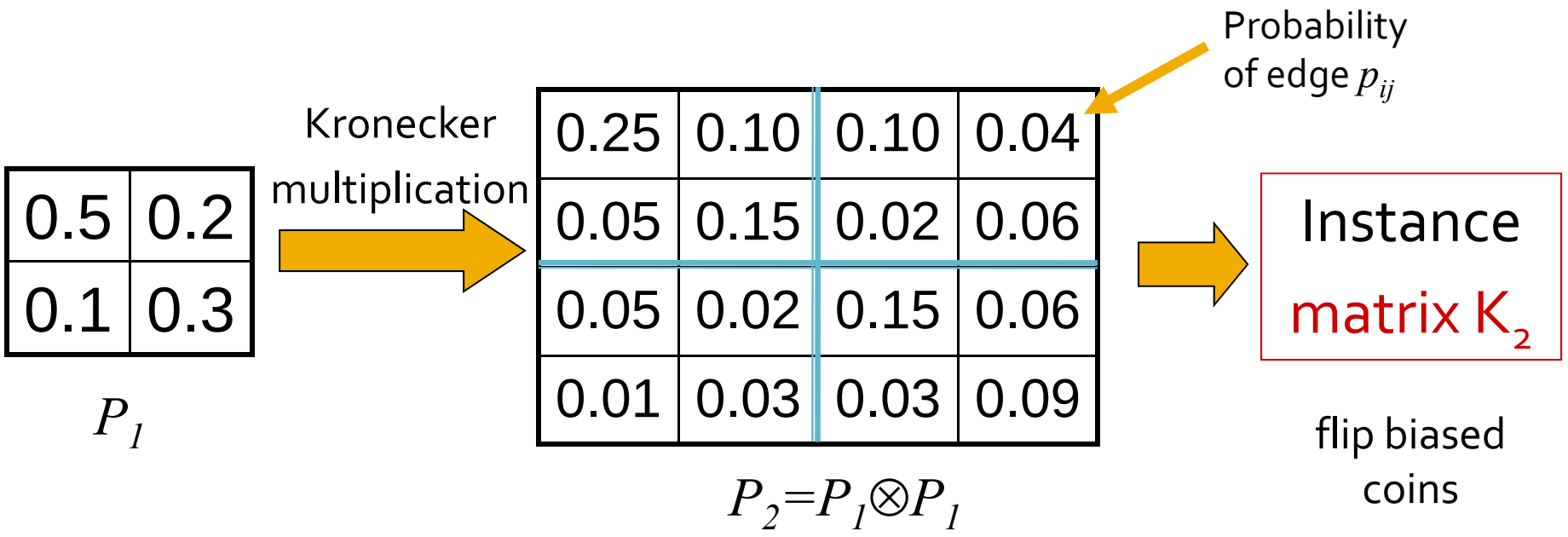


$G_2 = G_1 \otimes G_1$

[PKDD '05]

Stochastic Kronecker graphs

- Create $N_1 \times N_1$ probability matrix P_1
- Compute the k^{th} Kronecker power P_k
- For each entry p_{uv} of P_k include an edge (u,v) with probability p_{uv}



[Leskovec et al. 09]

Kronecker graphs: Interpretation

- Initiator matrix K_1 is a similarity matrix
- Node v_i is described with k binary attributes:
  $v_1, v_2, \ldots, v_k$
- Probability of a link between nodes v_i, v_j:
  $P(v_i, v_j) = \prod_a K_1[v_i(a), v_j(a)]$

$$K_1 = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & a & b \\ 1 & c & d \end{array}$$

$$K_2 = K_1 \otimes K_1$$

	v_1	v_2	v_3	v_4
v_1	a·a	a·b	b·a	b·b
v_2	a·c	a·d	b·c	b·d
v_3	c·a	c·b	d·a	d·b
v_4	c·c	c·d	d·c	d·d

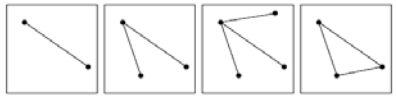
=

	v_1	v_2	v_3	v_4
v_1	a	b	a	b
v_2	c	d	c	d
v_3	a	b	a	b
v_4	c	d	c	d

$$v_2 = (0,1)$$
$$v_3 = (1,0)$$
$$P(v_2, v_4) = b \cdot c$$

Estimating Kronecker graphs

- Given a real network G

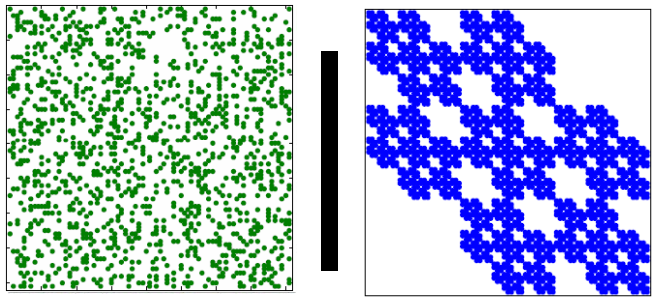
  Want to estimate initiator matrix: $K_1 = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
- Method of moments [Owen '09]
  - Compare counts of  and solve system of equations.
- Maximum likelihood [Leskovec-Faloutsos ICML '07]
  - arg max P( G_1)
- SVD [VanLoan-Pitsianis '93]
  - Can solve $\min \left\| G - K_1 \otimes K_1 \right\|_F^2$ using SVD

[Leskovec-Faloutsos ICML '07]

Kronecker graphs: Estimation

- Maximum likelihood estimation

$$\arg\max_{G_1} P(\quad \mid \quad \text{Kronecker} \quad K_1)$$



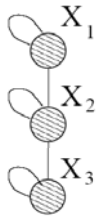
- Naïve estimation takes $O(N!N^2)$:
  - $N!$ for different node labelings:
    - **Our solution:** Metropolis sampling: $N!$ → ***(big) const***
  - N^2 for traversing graph adjacency matrix
    - **Our solution:** Kronecker product ($E << N^2$): N^2 → E
- Do gradient descent

$$K_1 = \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array}$$

Estimate the model in O(E)

Model estimation: approach

- Maximum likelihood estimation
  - Given real graph G
  - Estimate Kronecker initiator graph Θ (e.g. $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$) which



$$\arg\max_{\Theta} P(G \mid \Theta)$$

- We need to (efficiently) calculate

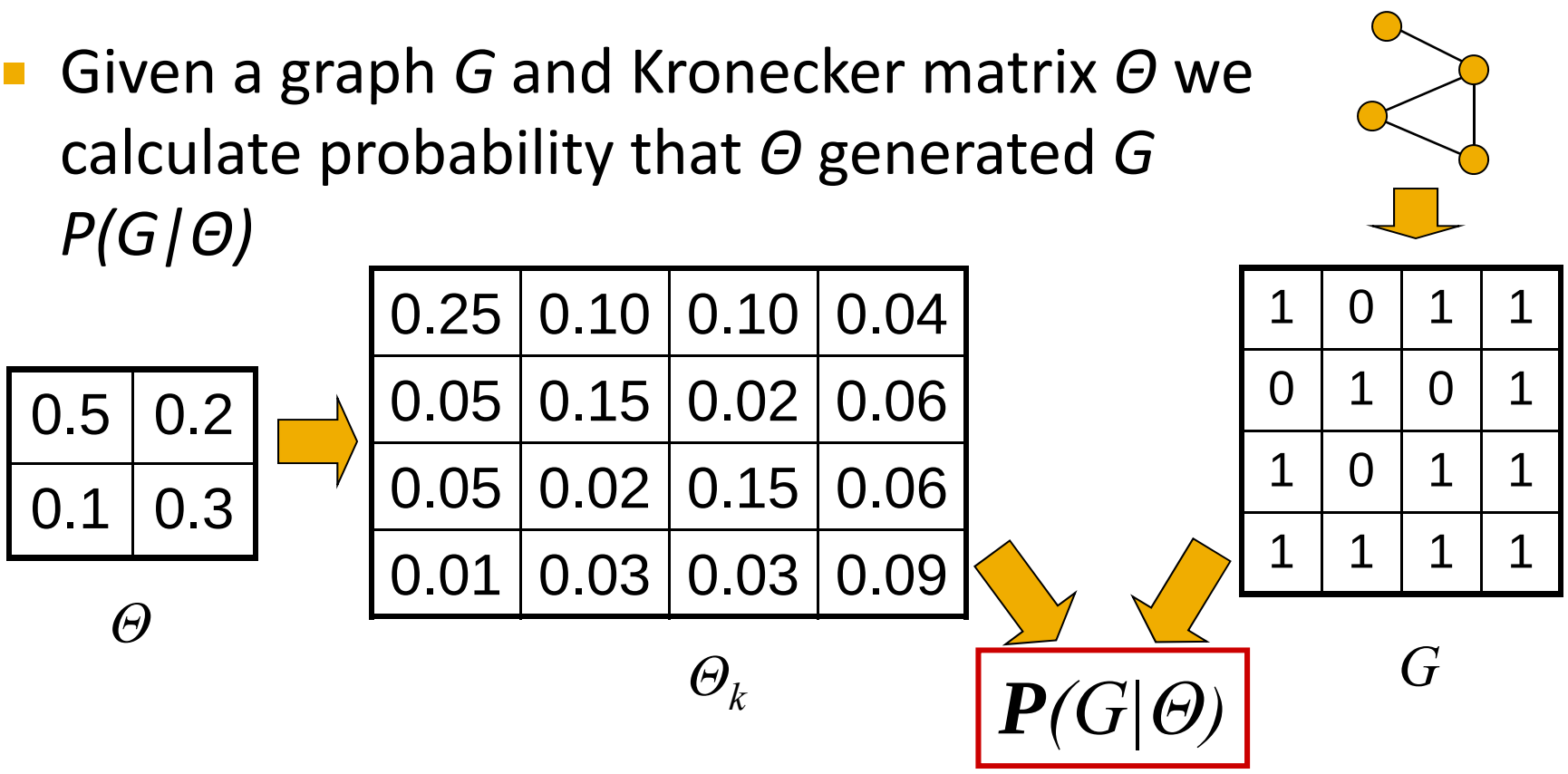
$$P(G \mid \Theta)$$

- And maximize over Θ (e.g., using gradient descent)

[ICML '07]

KronFit: Likelihood

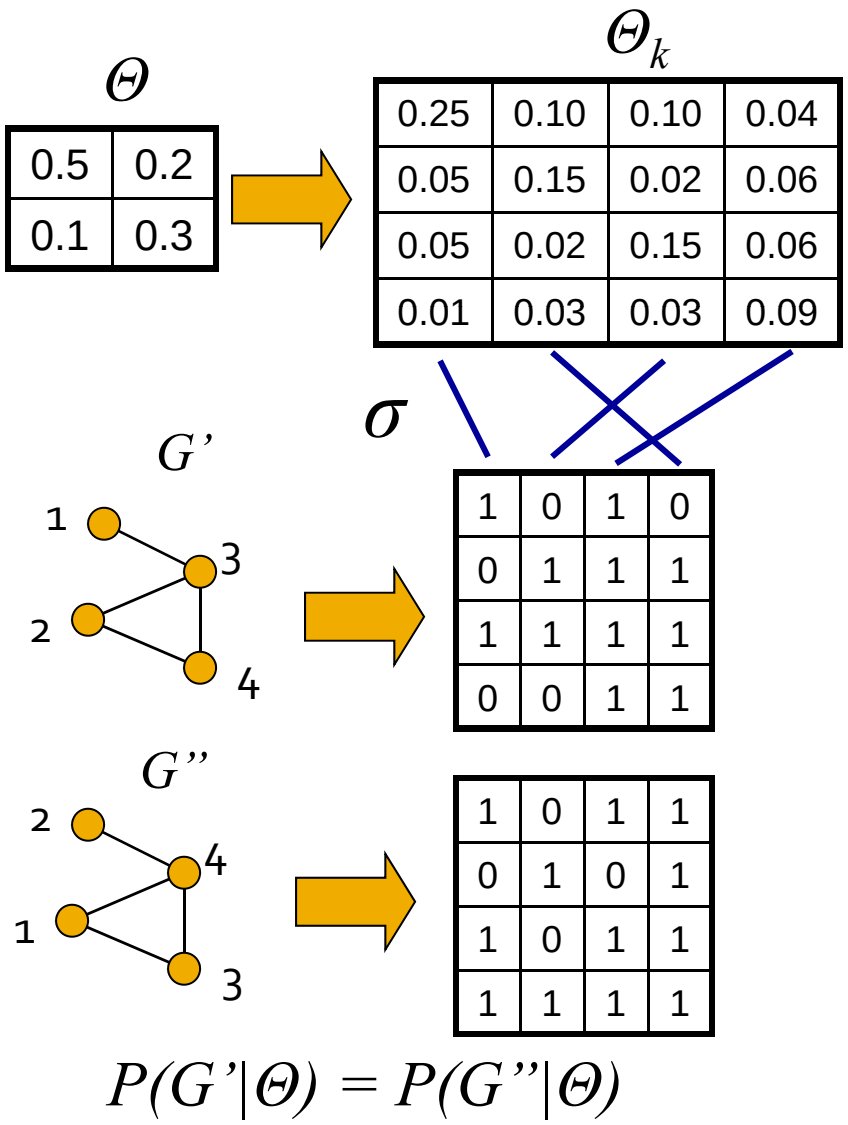
- Given a graph *G* and Kronecker matrix *Θ* we calculate probability that *Θ* generated *G* *P(G|Θ)*



$$P(G \mid \Theta) = \prod_{(u,v)\in G} \Theta_k[u,v] \prod_{(u,v)\notin G} (1-\Theta_k[u,v])$$

[ICML '07]

Challenge 1: Node correspondence



- Nodes are unlabeled
- Graphs *G'* and *G''* should have the same probability $P(G'|\Theta) = P(G''|\Theta)$
- One needs to consider all node correspondences σ

$$P(G \mid \Theta) = \sum_{\sigma} P(G \mid \Theta, \sigma) P(\sigma)$$

- All correspondences are a priori equally likely
- There are ***O(N!)*** correspondences

[ICML '07]

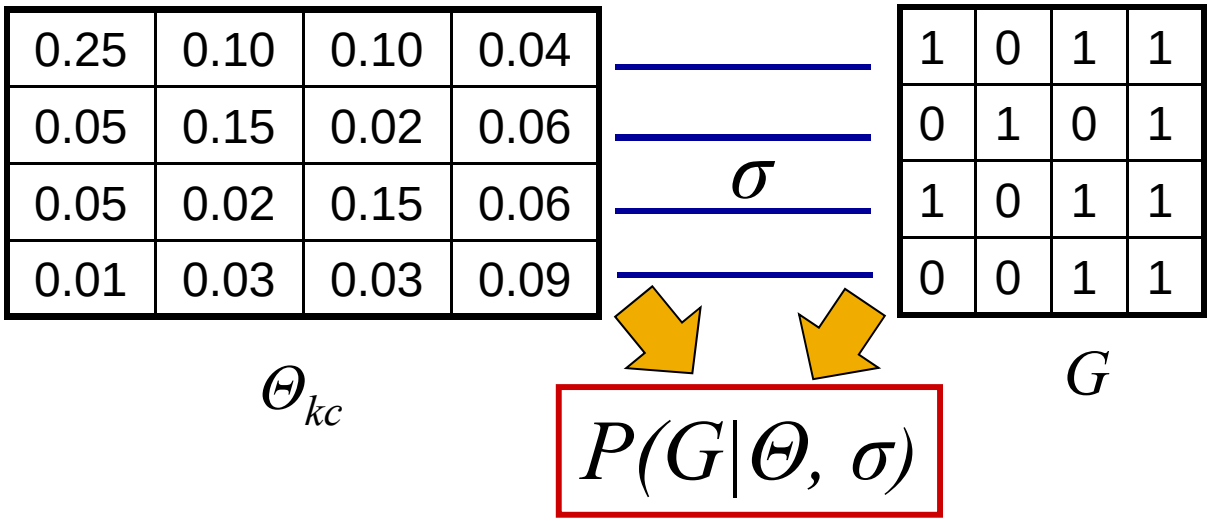
Challenge 2: calculating $P(G|\Theta,\sigma)$

- Assume we solved the correspondence problem
- Calculating

$$P(G \mid \Theta) = \prod_{(u,v)\in G} \Theta_k[\sigma_u, \sigma_v] \prod_{(u,v)\notin G} (1 - \Theta_k[\sigma_u, \sigma_v])$$

- Takes ***O(N²)*** time
- Infeasible for large graphs (N ~ 10^5)

σ... node labeling



[ICML '07]

Solution 1: Node correspondence

- Log-likelihood

$$l(\Theta) = \log \sum_{\sigma} P(G|\Theta,\sigma)P(\sigma)$$

- Gradient of log-likelihood

$$\frac{\partial}{\partial\Theta} l(\Theta) = \sum_{\sigma} \frac{\partial \log P(G|\sigma,\Theta)}{\partial\Theta} P(\sigma|G,\Theta)$$

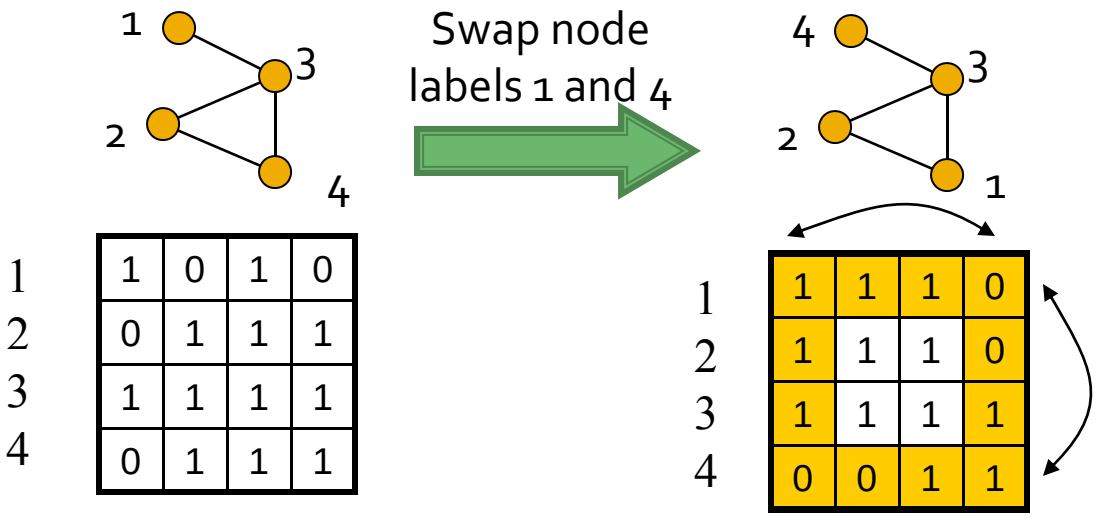
- Sample the permutations from $P(\sigma|G,\Theta)$ and average the gradients

[ICML '07]

Solution 1: Node correspondence

- Metropolis sampling
  - Start with a random permutation σ
  - Do local moves on the permutation
  - Accept the new permutation σ'
    - If new permutation is better (gives higher likelihood)
    - else accept with prob. proportional to the ratio of likelihoods (**no need to calculate the normalizing constant!**)

$$\frac{P(\sigma'|G,\Theta)}{P(\sigma|G,\Theta)}$$



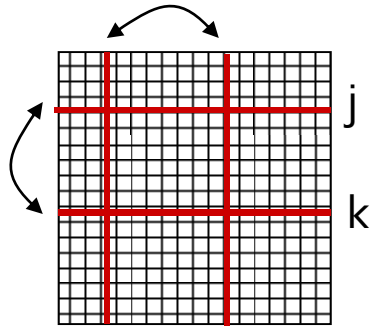
Can compute efficiently

Only need to account for changes in 2 rows / columns

Sampling node labelings (2)

$\sigma^{(0)} := (1, \ldots, N)$
repeat
  Draw j and k uniformly from $(1, \ldots, N)$
  $\sigma^{(i)} := \texttt{SwapElements}(\sigma^{(i-1)}, j, k)$
  Draw u from $U(0,1)$
  if $u > \frac{P(\sigma^{(i)}|G,\Theta)}{P(\sigma^{(i-1)}|G,\Theta)}$ **then**
    $\sigma^{(i)} := \sigma^{(i-1)}$
  end if
  i = i + 1
until $\sigma^{(i)} \sim P(\sigma|G,\Theta)$
return $\sigma^{(i)}$

Metropolis permutation sampling algorithm



- Need to efficiently calculate the likelihood ratios
- But the permutations $\sigma^{(i)}$ and $\sigma^{(i+1)}$ only differ at 2 positions
- So we only traverse to update 2 rows (columns) of Θ_k
- We can evaluate the likelihood ratio efficiently

Solution 2: Calculating $P(G|\Theta,\sigma)$

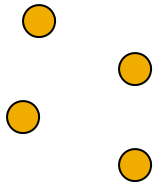
[ICML '07]

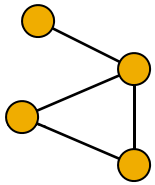
- Calculating naively $P(G|\Theta,\sigma)$ takes $O(N^2)$
- Idea:
  - First calculate likelihood of empty graph, a graph with 0 edges
  - Correct the likelihood for edges that we observe in the graph
- By exploiting the structure of Kronecker product we obtain closed form for likelihood of an empty graph

Solution 2: Calculating $P(G|\Theta,\sigma)$

[ICML '07]

- We approximate the likelihood:



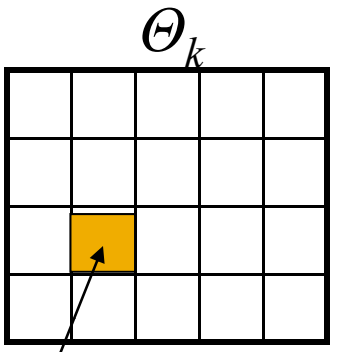


$$l(\Theta) \approx \underbrace{l_e(\Theta)}_{\text{Empty graph}} + \sum_{(u,v)\in G} \underbrace{-\log(1-\Theta_k[\sigma_u,\sigma_v])}_{\text{No-edge likelihood}} + \underbrace{\log(\Theta_k[\sigma_u,\sigma_v])}_{\text{Edge likelihood}}$$

- The sum goes only over the edges
- Evaluating $P(G|\Theta,\sigma)$ takes $O(E)$ time
- Real graphs are sparse, $E << N^2$

Calculating $P(G|\Theta,\sigma)$

- Real graphs are sparse so we first calculate likelihood of empty graph
- Probability of edge *(i,j)* is in general $p_{ij} = \theta_1^a \theta_2^b \theta_3^c \theta_4^d$
- By using Taylor approximation to p_{ij} and summing the multinomial series we obtain:



$$l_e(\Theta) = \sum_{i,j=1}^{N} \log(1 - p_{ij}) \approx -\left(\sum_{i,j=1}^{N_1} \theta_{i,j}\right)^k - \frac{1}{2}\left(\sum_{i,j=1}^{N_1} \theta_{i,j}^2\right)^k$$

- We approximate the likelihood:

Taylor approximation
$\log(1-x) \sim -x - 0.5\,x^2$

$$l(\Theta) \approx \underbrace{l_e(\Theta) +}_{\text{Empty graph}} \sum_{(u,v)\in G} \underbrace{-\log(1 - \Theta_k[\sigma_u, \sigma_v])}_{\text{No-edge likelihood}} + \underbrace{\log(\Theta_k[\sigma_u, \sigma_v])}_{\text{Edge likelihood}}$$

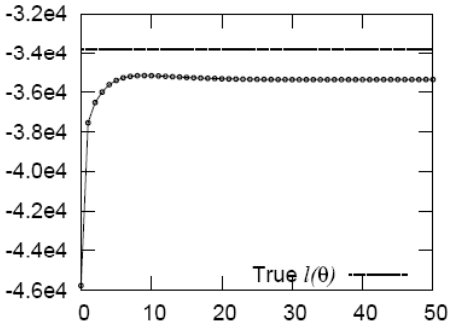
Experiments: real networks

[ICML '07]

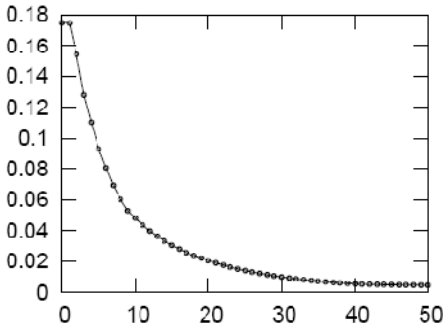
- Experimental setup
  - Given real graph
  - Stochastic gradient descent from random initial point
  - Obtain estimated parameters
  - Generate synthetic graphs
  - Compare properties of both graphs
- We do not fit the properties themselves
- We fit the likelihood and then compare the graph properties

Convergence of fitting

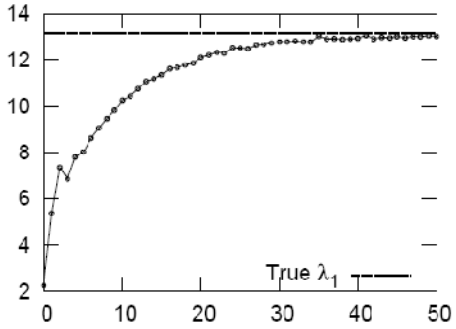
- Can gradient descent recover true parameters?
- How nice (smooth, without local minima) is optimization space?
  - Generate a graph from random parameters
  - Start at random point and use gradient descent
  - We recover true parameters 98% of the times
- How does algorithm converge to true parameters with gradient descent iterations?



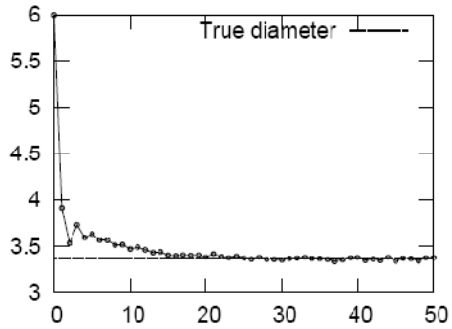
Log-likelihood



Avg abs error



1st eigenvalue



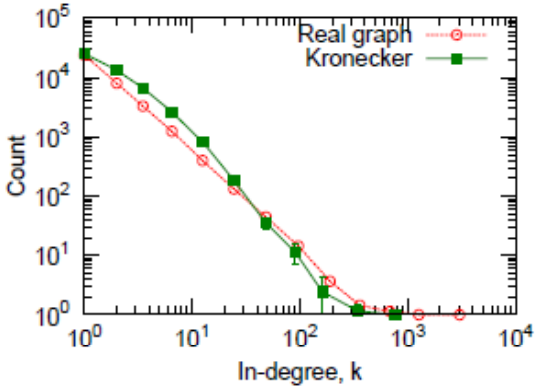
Diameter

[Leskovec-Faloutsos ICML '07]

Estimation: Epinions (n=76k, m=510k)

- Real and Kronecker are very close:

$$K_1 = \begin{bmatrix} 0.99 & 0.54 \\ 0.49 & 0.13 \end{bmatrix}$$



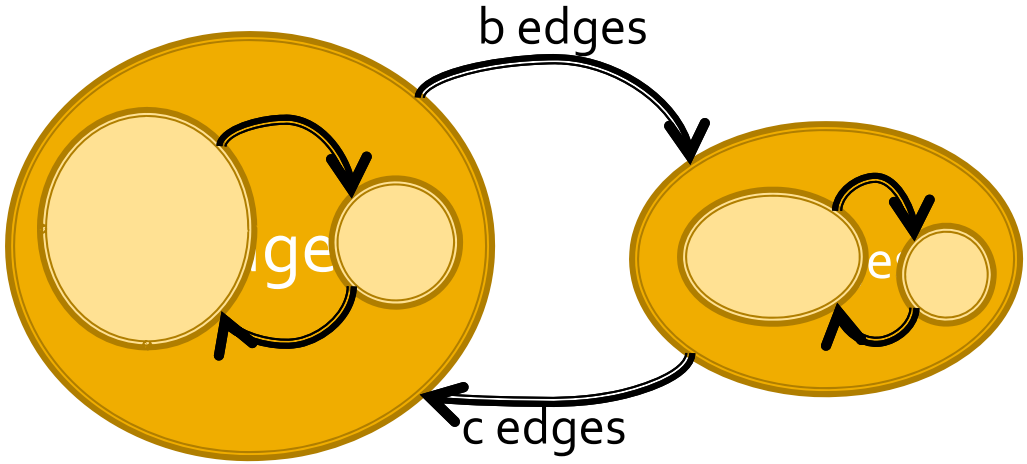
(a) In-Degree

[Leskovec et al. Arxiv 09]

Kronecker & Network structure

- What do estimated parameters tell us about the network structure?

$$K_1 = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

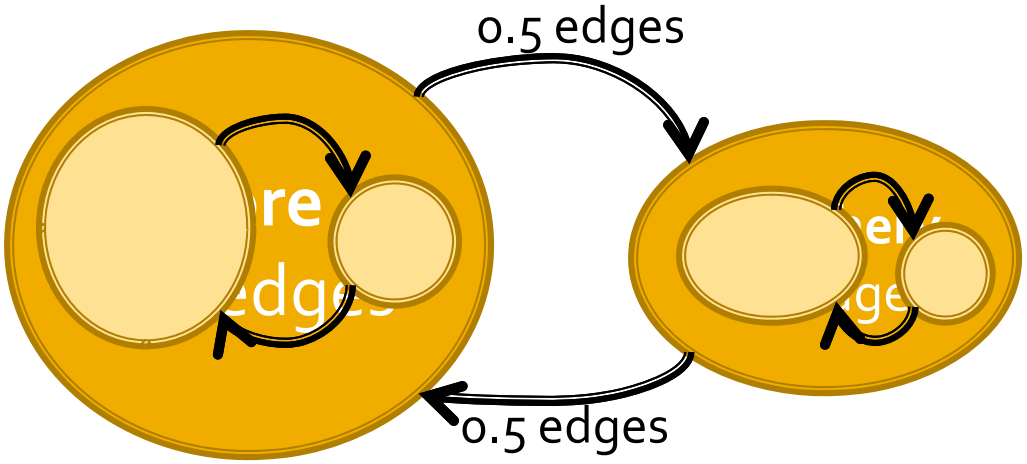


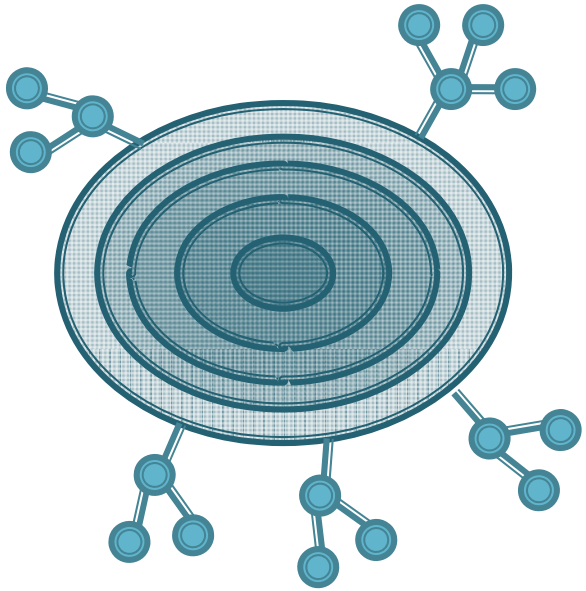
[Leskovec et al. Arxiv '09]

Kronecker & Network structure

- What do estimated parameters tell us about the network structure?

$$K_1 = \begin{bmatrix} 0.9 & 0.5 \\ 0.5 & 0.1 \end{bmatrix}$$



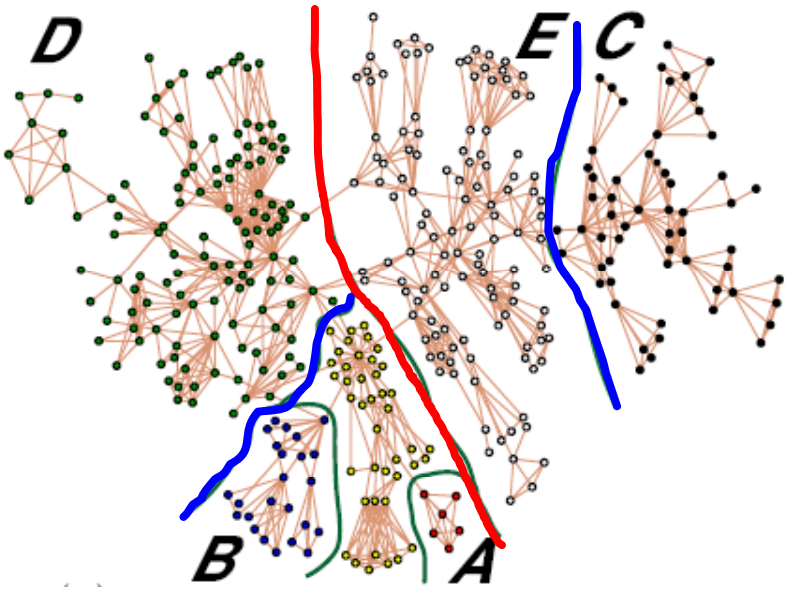


Nested Core-periphery

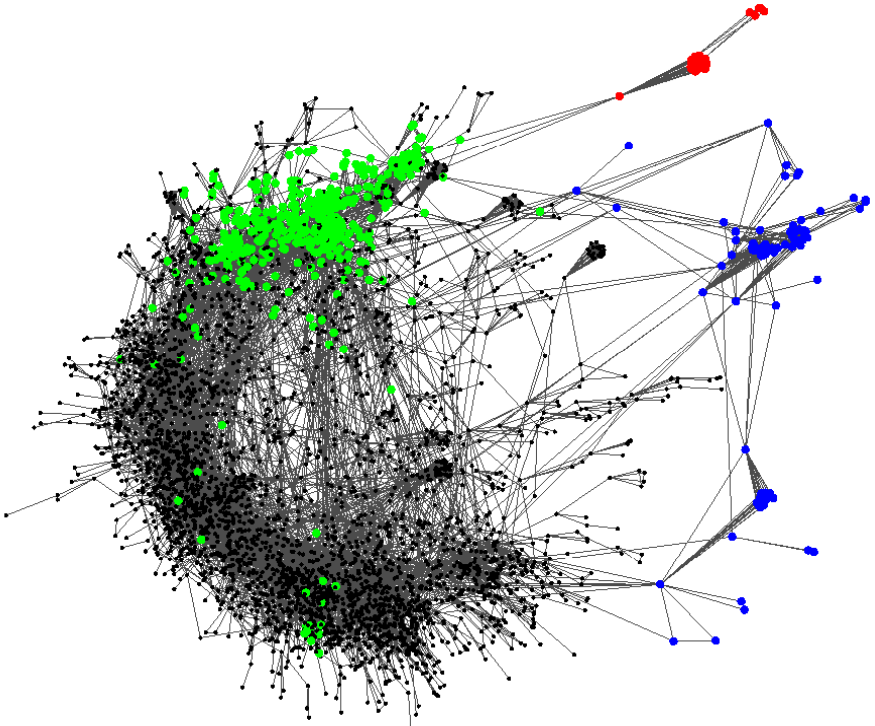
[Leskovec et al. Arxiv '09]

Small vs. Large networks

- Small and large networks are very different:



$$K_1 = \begin{bmatrix} 0.99 & 0.17 \\ 0.17 & 0.82 \end{bmatrix}$$



$$K_1 = \begin{bmatrix} 0.99 & 0.54 \\ 0.49 & 0.13 \end{bmatrix}$$