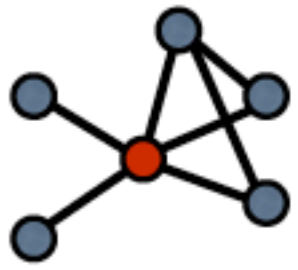
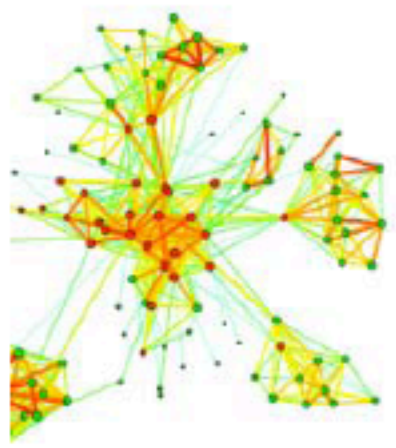
Community structure in networks

Levels of organization in networks

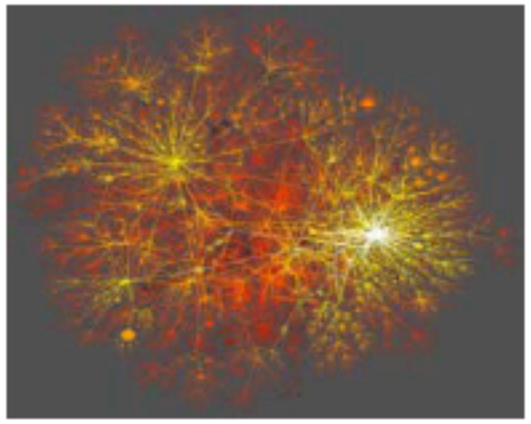
- **Microscopic:** nodes, edges, and their immediate surroundings, e.g. clustering



- **Mesoscopic:** structures containing many nodes and edges, e.g. **communities**, backbones

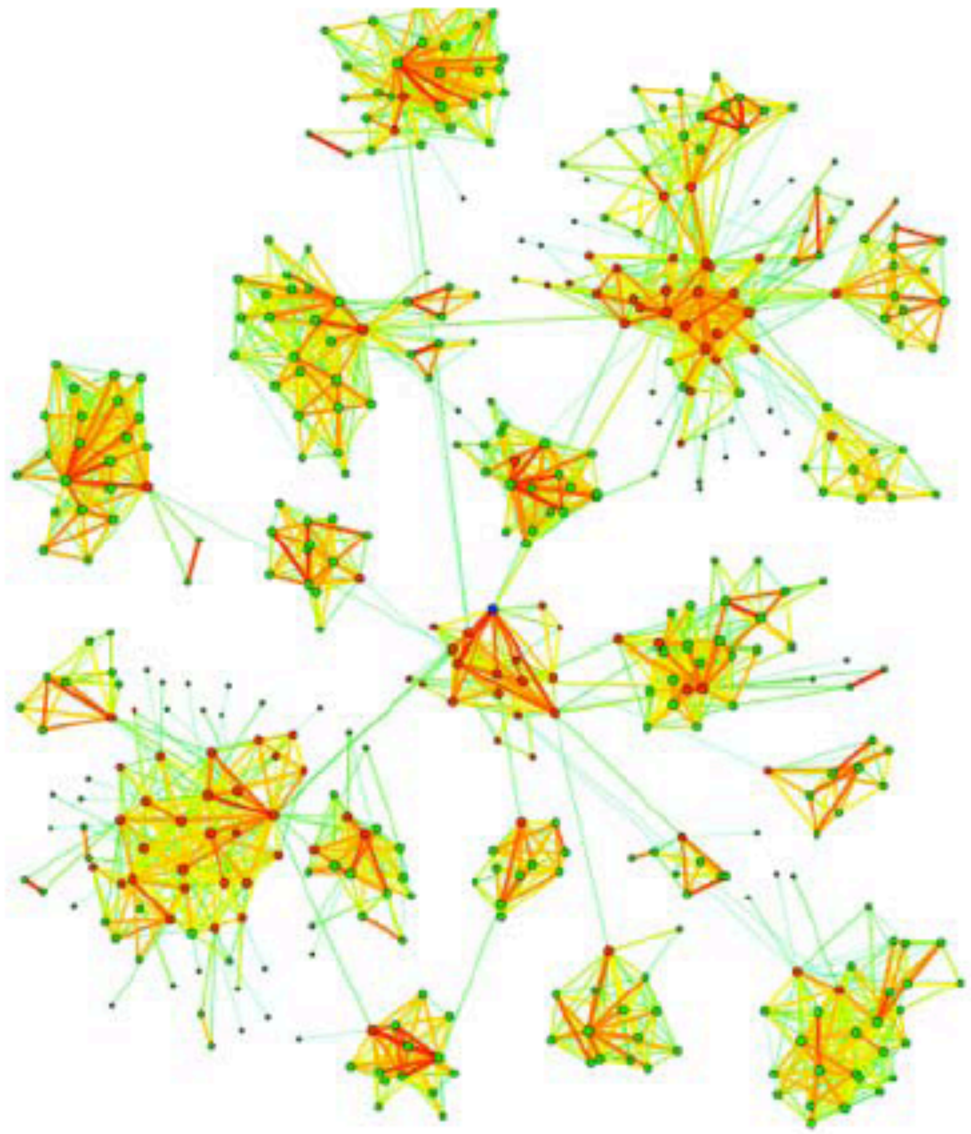


- **Macroscopic:** features at the network level, e.g. degree distributions, existence of hubs



Communities: a loose definition

- At the mesoscopic level, networks commonly display modular organization
- There are **communities, groups of nodes with dense internal connections**
- The idea is (deceivingly) simple; yet formally defining when nodes constitute a community is not!
- Communities = modules = clusters, terminology varies

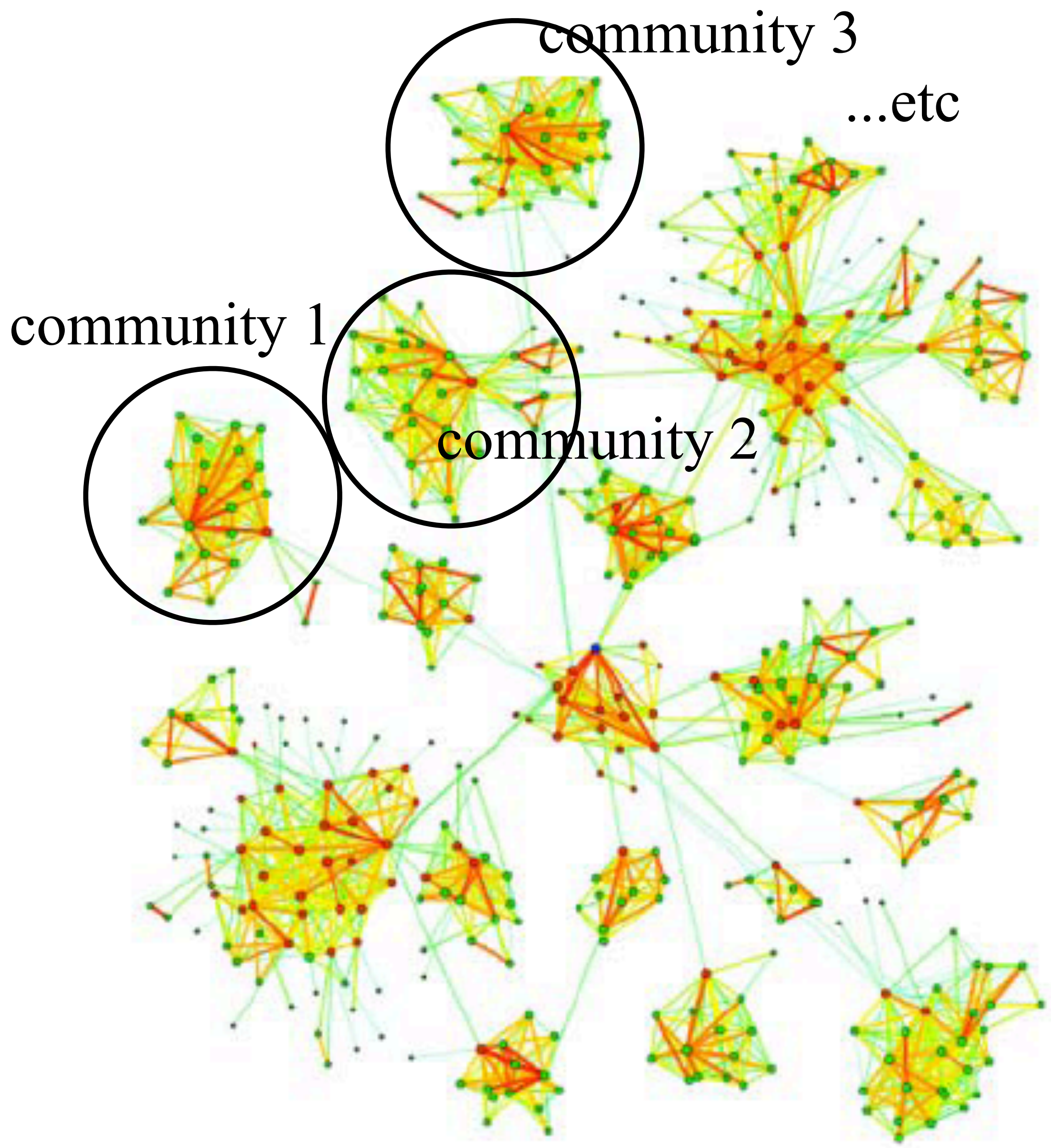


Detecting communities

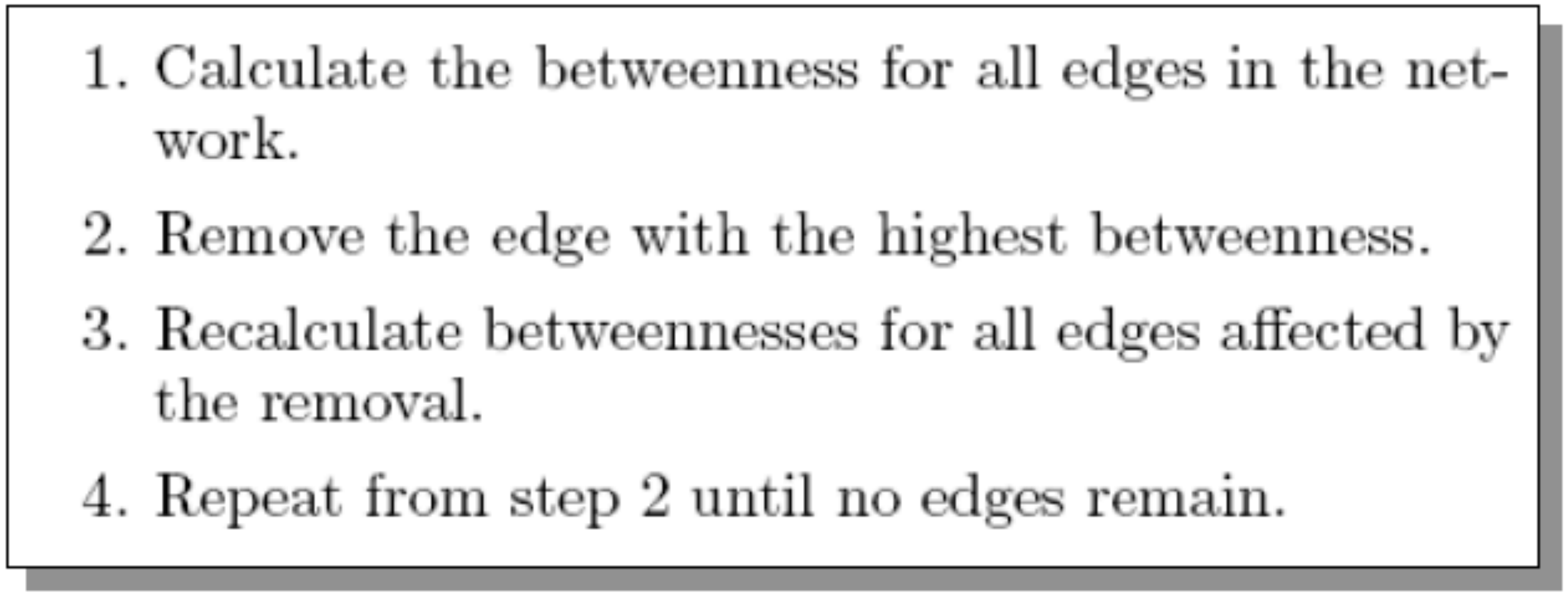
- A problem with no correct solution
- What we want is to divide assign community membership to each node, i.e. partition the network
- Outcome: community indices for each node:

```
node1 community1
node2 community1
...
node100 community12
...etc
```

- Has to be based on some definition of "community"
- The algorithm should be fast enough to be applicable for large networks



An early solution: the Girvan-Newman method

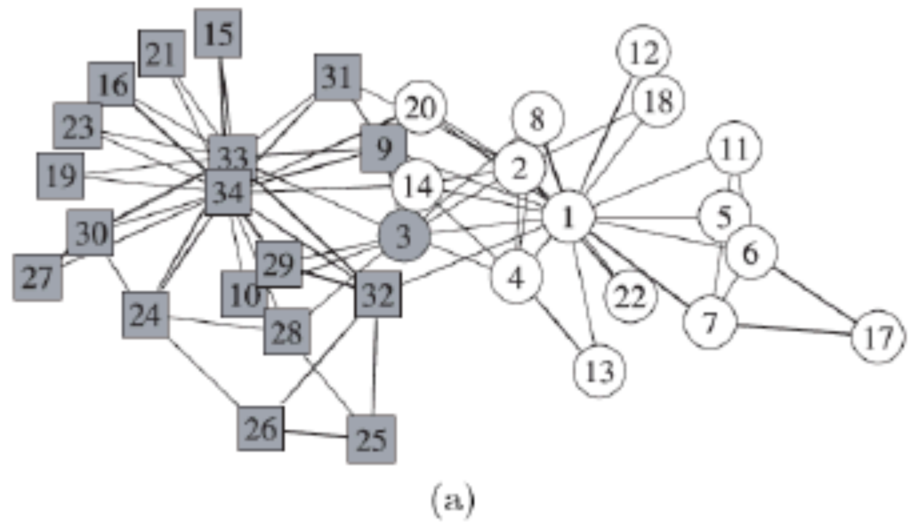


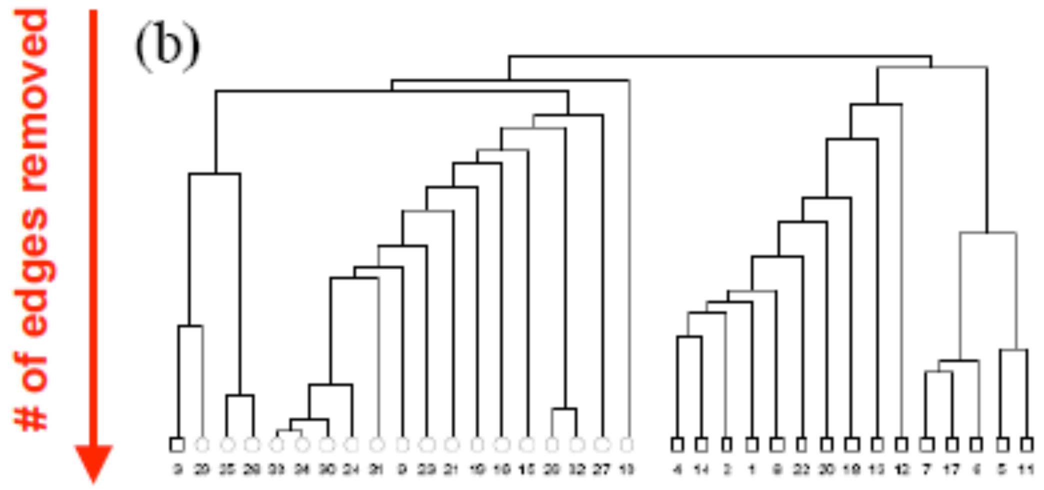
1. Calculate the betweenness for all edges in the network.
2. Remove the edge with the highest betweenness.
3. Recalculate betweennesses for all edges affected by the removal.
4. Repeat from step 2 until no edges remain.

Girvan M. and Newman M. E. J., Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99, 7821–7826 (2002)

- Edges between clusters of densely connected nodes must have a high betweenness centrality
- Solution: think of these edges as bridges between communities and remove them to get the communities! So communities = connected components.
- Instead of a single partitioning, there will be a partitioning for each removal step

An early solution: the Girvan-Newman method


(a)



- Zachary's karate club, a famous example studied in social sciences
- The club split into two due to leaders' disputes
- First level of splitting with the GN method (circles, squares) corresponds to actual split

- This method is **hierarchical**, producing a tree structure of **nested** communities
- Such structures can be presented as **dendrograms**

Modularity - which partition is the best?

- The GN method produces many possible community divisions
- In order to select the "best" and most "natural" one, Newman proposed a quality function, the **modularity:**

$$Q = \frac{1}{2L} \sum_{i \neq j} \left(a_{ij} - \frac{k_i k_j}{2L} \right) \delta\left(\sigma_i, \sigma_j\right)$$

- L = number of edges in the network,
- a_{ij} = adjacency matrix element,
- k_i, k_j = degrees of i and j,
- σ_i, σ_j = community indices of i and j

- Modularity measures how many "excess" edges there are within communities, compared with a random null model
- $k_i\, k_j / 2L$ = expected number of edges between i and j in the random null model
- Random null model = configuration model, *i.e.* a network with an identical degree sequence but otherwise as random as it gets

Modularity Optimization

- The next, logical conclusion: instead of edge removal, just **find the partition which optimizes modularity Q for the network!**

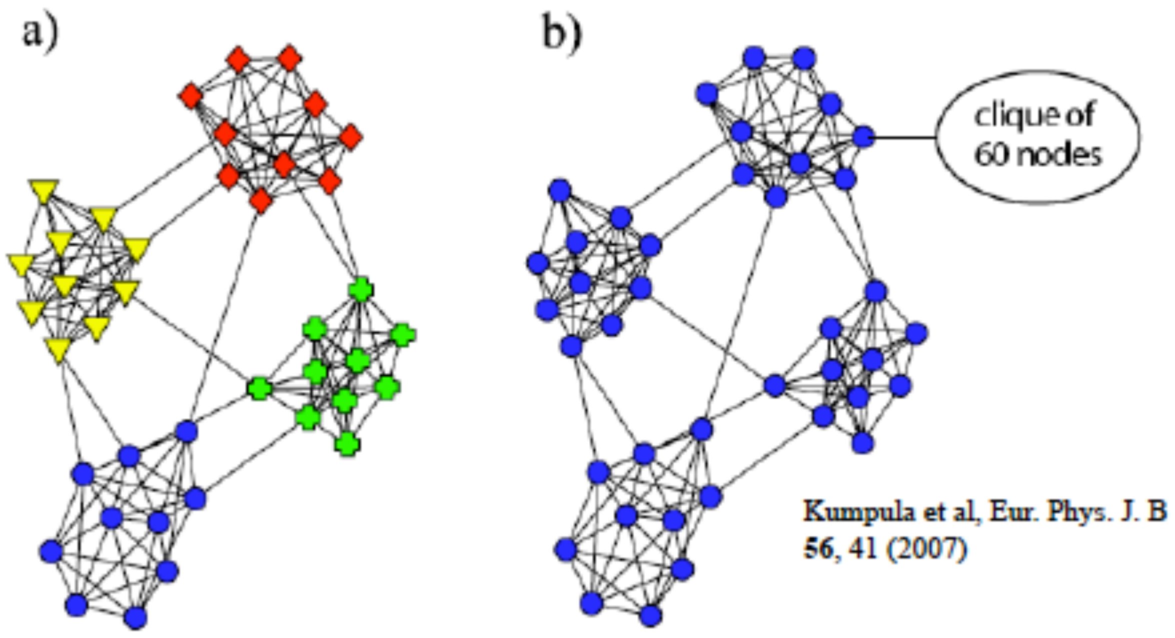
- There are numerous ways of doing this, e.g. simulated annealing
  - Initially assign community indices randomly
  - Change index of node (nodes) randomly, accept change if Q higher, otherwise accept with small probability or reject
  - Repeat, decreasing lower-Q-acceptance probabilities

- Modularity optimization has been widely used

- Problems:
  - always assumes a single "correct" partitioning
  - finding the true optimum is a NP-hard problem
  - algorithms & heuristics stochastic; outcomes can differ a lot
  - optimizing a global quantity gives rise to problems...

Modularity Optimization: Deep Problems

- The use of a global reference leads to deep problems
- **The optimal partition depends on the number of links in the network!**
- **Communities below a limiting size are not detected!** This size depends on the number of links in the network.
- This is the **resolution limit**
- See Fortunato & Barthélemy, PNAS **104**, 36 (2007)

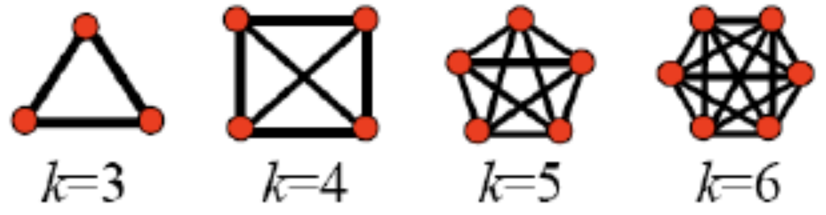


4 clusters: modularity optimized

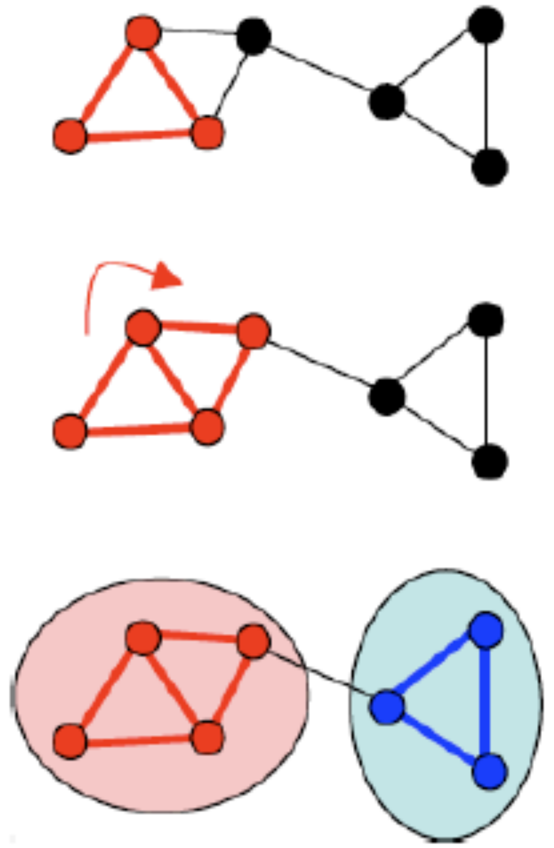
network extended so that new cluster joins by a single link: modularity optimized when clusters are merged!

Clique Percolation

- Palla *et al*, Nature **435**, 814 (2005)
- Clique = fully connected subgraph
- Communities defined as sets of nodes belonging to adjacent *k*-cliques
- Adjacent = sharing *k-1* nodes

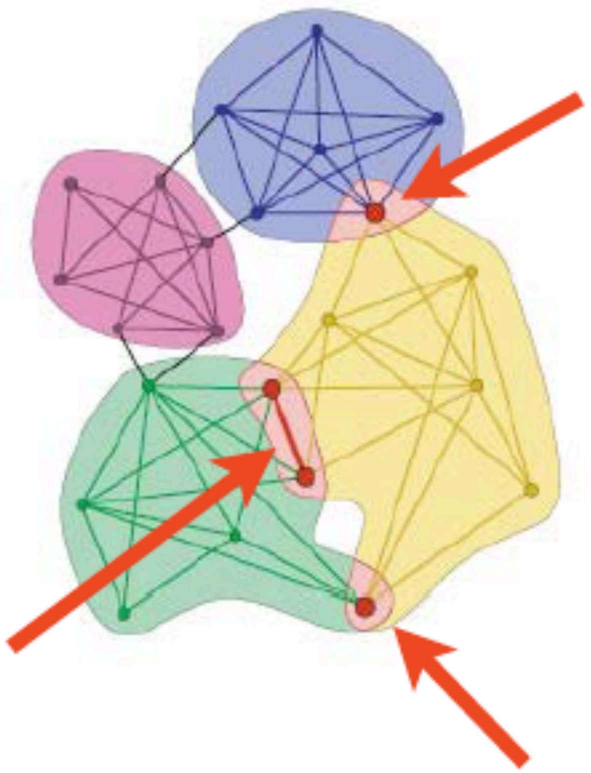


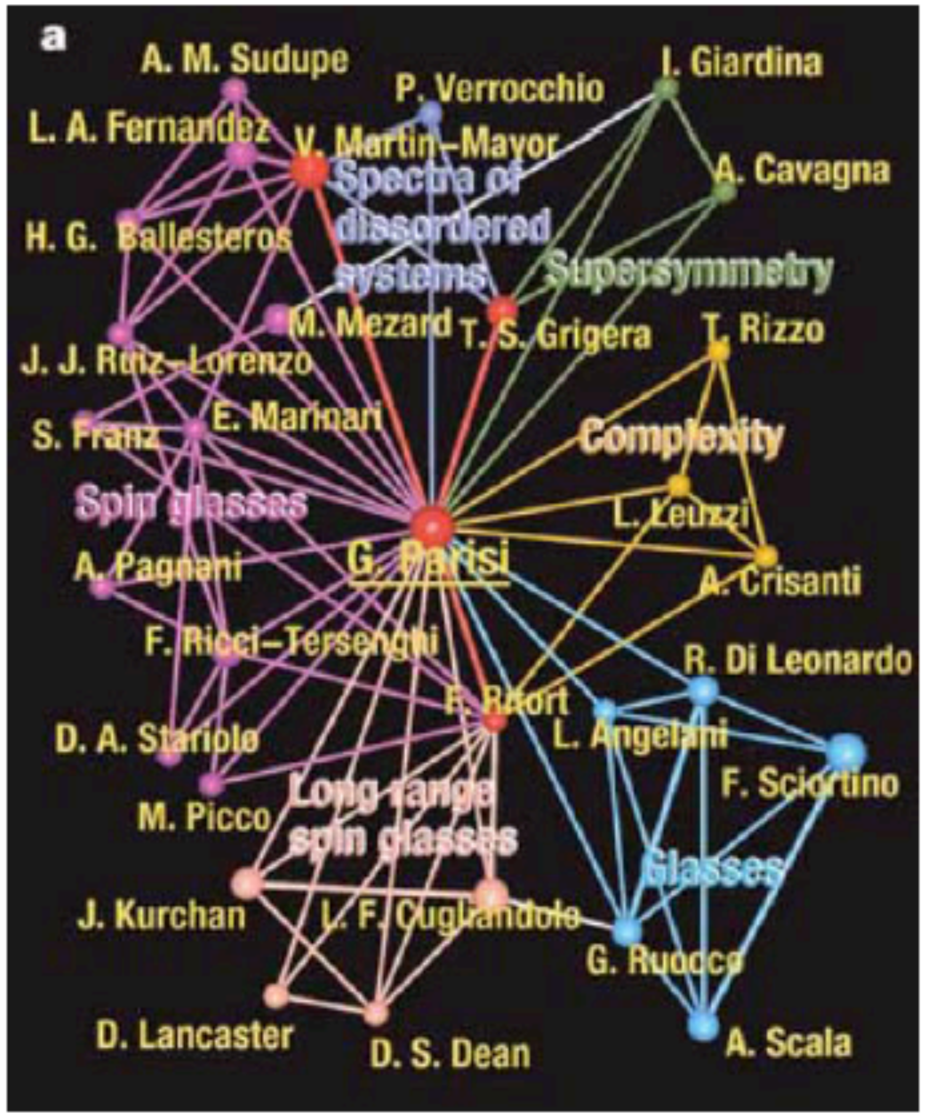
Example:
3-clique communities



Clique Percolation

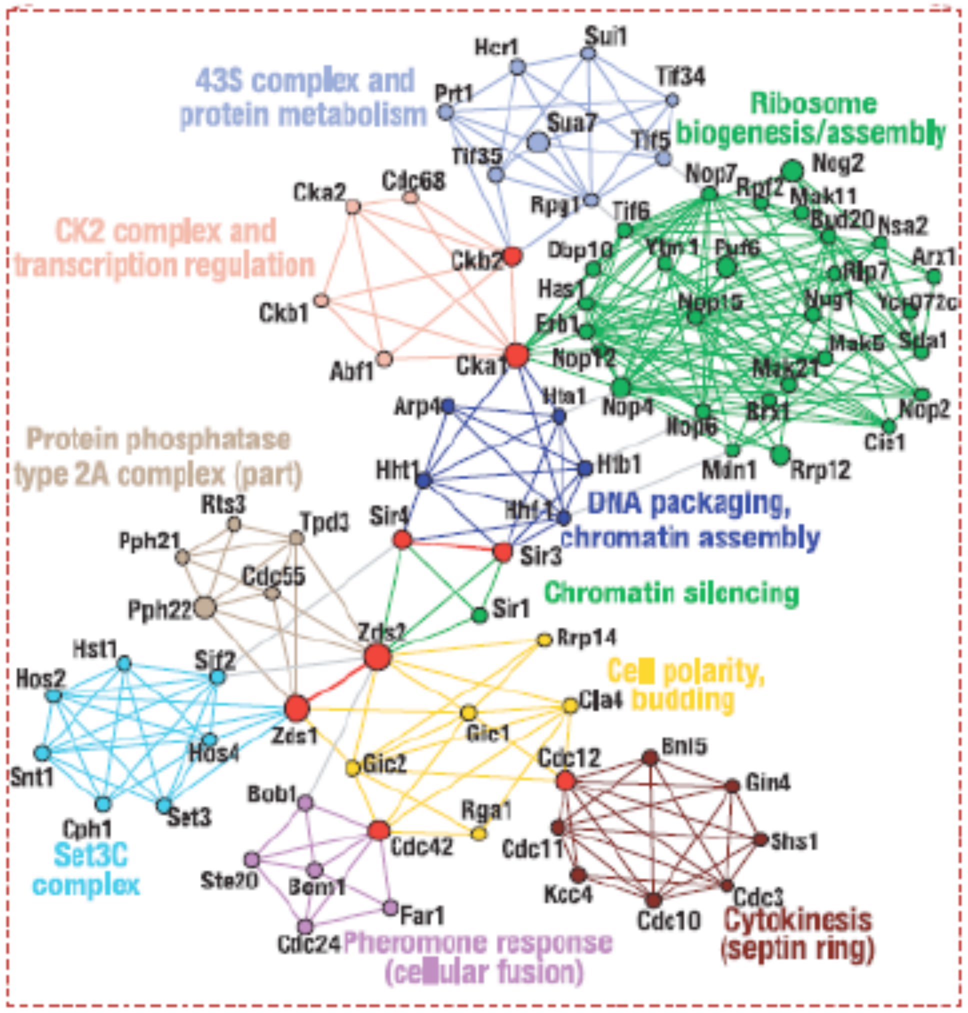
- **One node can belong to multiple communities**
- This makes sense especially for social networks





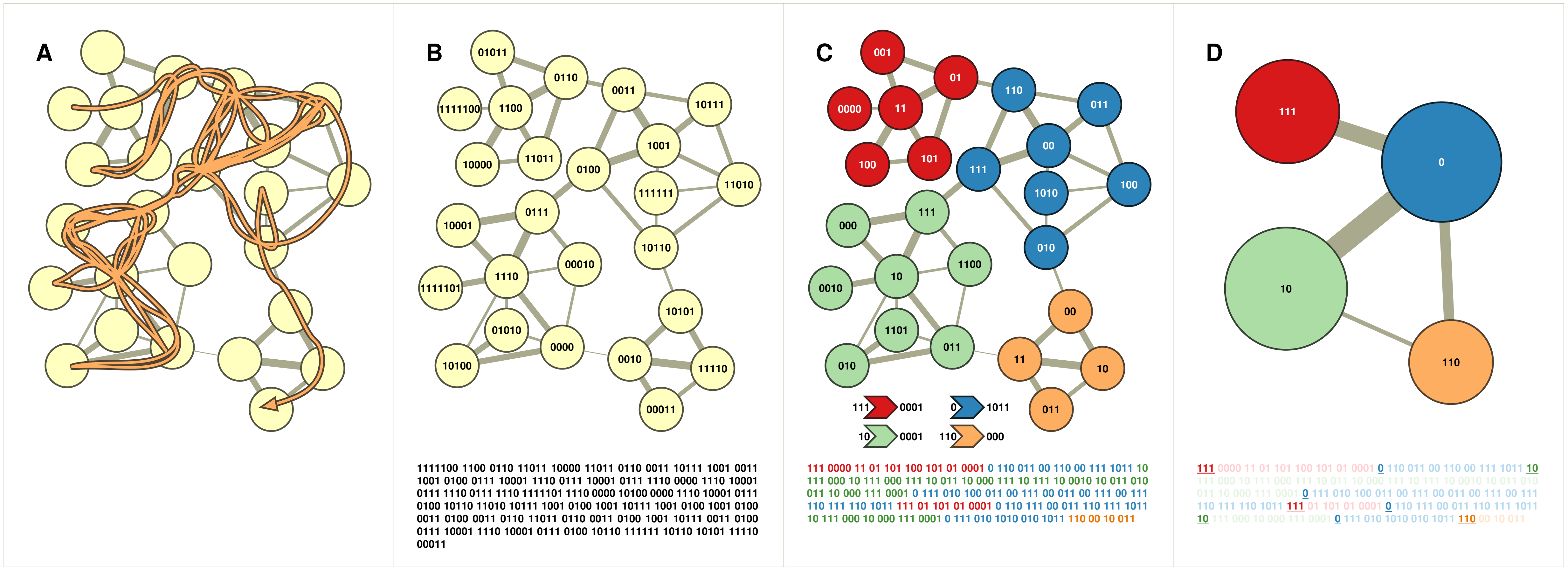
Clique Percolation - Practical Considerations

- If nodes do not participate in a clique, they are not assigned to any community (for sparse networks this can be a problem)
- Too small clique sizes can give rise to a giant clique community spanning almost the entire network
- So for extracting useful information, one has to choose a right clique size (in practice often k=3-5)
- **Free software: http://www.cfinder.org/**


43S complex and protein metabolism
Ribosome biogenesis/assembly
CK2 complex and transcription regulation
Protein phosphatase type 2A complex (part)
DNA packaging, chromatin assembly
Chromatin silencing
Cell polarity, budding
Set3C complex
Pheromone response (cellular fusion)
Cytokinesis (septin ring)

Infomap: a method based on information flow & Huffman coding

Maps of information flow reveal community structure in complex networks, Martin Rosvall and Carl T. Bergstrom, PNAS 105, 1118 (2008).



```
1111100 1100 0110 11011 10000 11011 0110 0011 10111 1001 0011
1001 0100 0111 10001 1110 0111 10001 0111 1110 0000 1110 10001
0111 1110 0111 1110 1111101 1110 0000 10100 0000 1110 10001 0111
0100 10110 11010 10111 1001 0100 1001 10111 1001 0100 1001 0100
0011 0100 0011 0110 11011 0110 0011 0100 1001 10111 0011 0100
0111 10001 1110 10001 0111 0100 10110 111111 10110 10101 11110
00011
```

```
111 0000 11 01 101 100 101 01 0001 0 110 011 00 110 00 111 1011 10
111 000 10 111 000 111 10 011 10 000 111 10 111 10 0010 10 011 010
011 10 000 111 0001 0 111 010 100 011 00 111 00 011 00 111 00 111
110 111 110 1011 111 01 101 01 0001 0 110 111 00 011 110 111 1011
10 111 000 10 000 111 0001 0 111 010 1010 010 1011 110 00 10 011
```

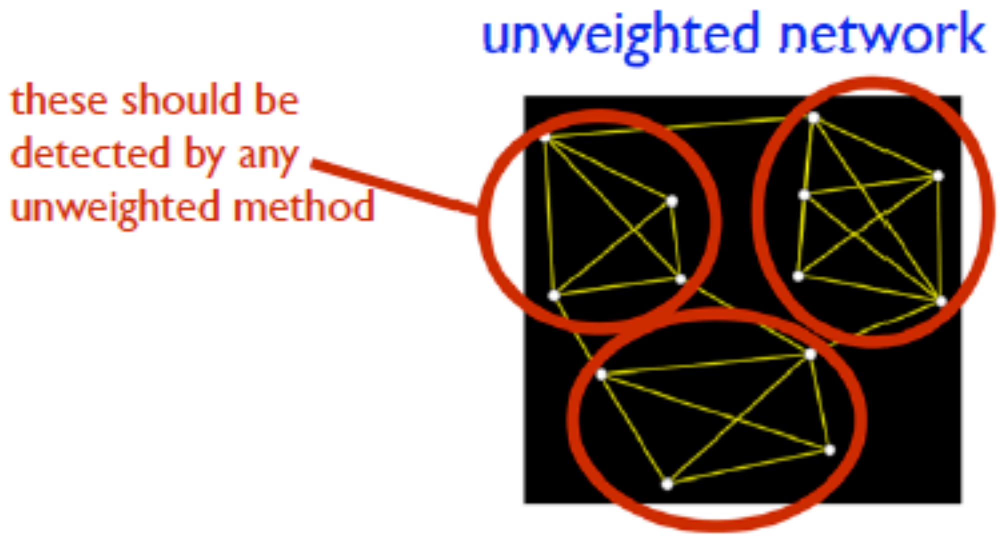
```
111 0000 11 01 101 100 101 01 0001 0 110 011 00 110 00 111 1011 10
111 000 10 111 000 111 10 011 10 000 111 10 111 10 0010 10 011 010
011 10 000 111 0001 0 111 010 100 011 00 111 00 011 00 111 00 111
110 111 110 1011 111 01 101 01 0001 0 110 111 00 011 110 111 1011
10 111 000 10 000 111 0001 0 111 010 1010 010 1011 110 00 10 011
```

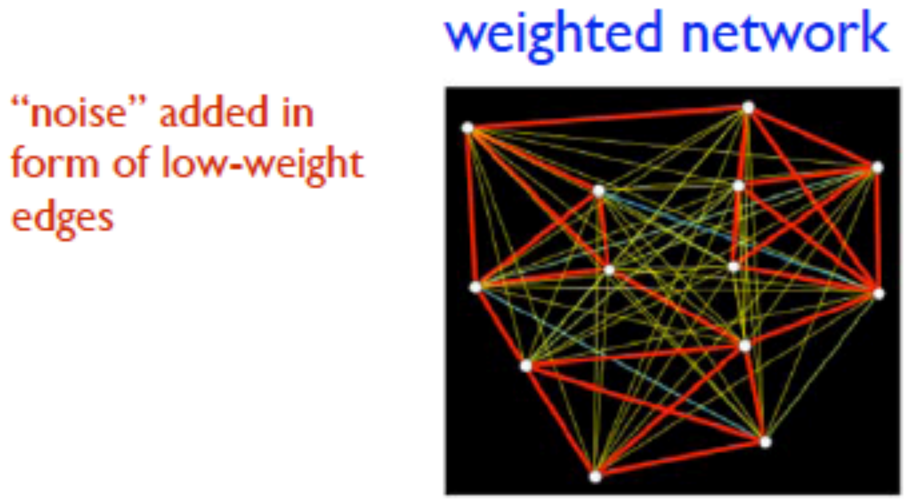
Free software: http://www.tp.umu.se/~rosvall/code.html

Note: works for weighted networks too.

Communities in Weighted Networks

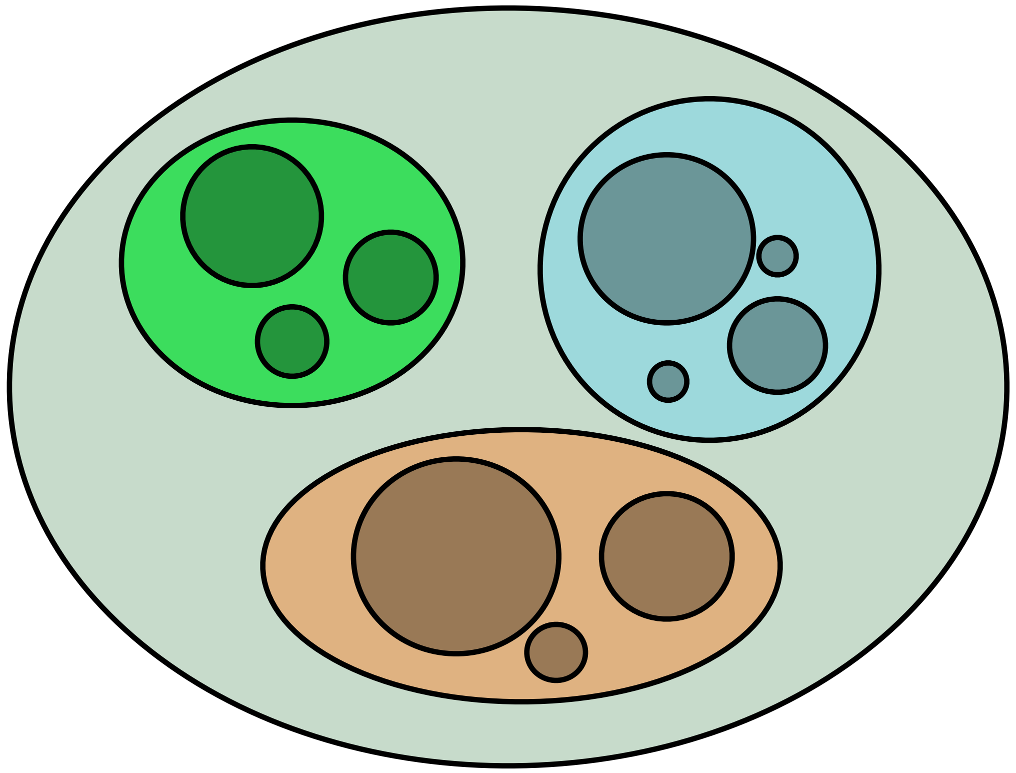
- Reasonable hypothesis: **intra-community weights are higher than inter-community weights**
- E.g. modularity optimization and the Potts method are then straightforward to generalize
- For clique percolation, a generalization based on intensities of cliques exists (computationally heavy!)





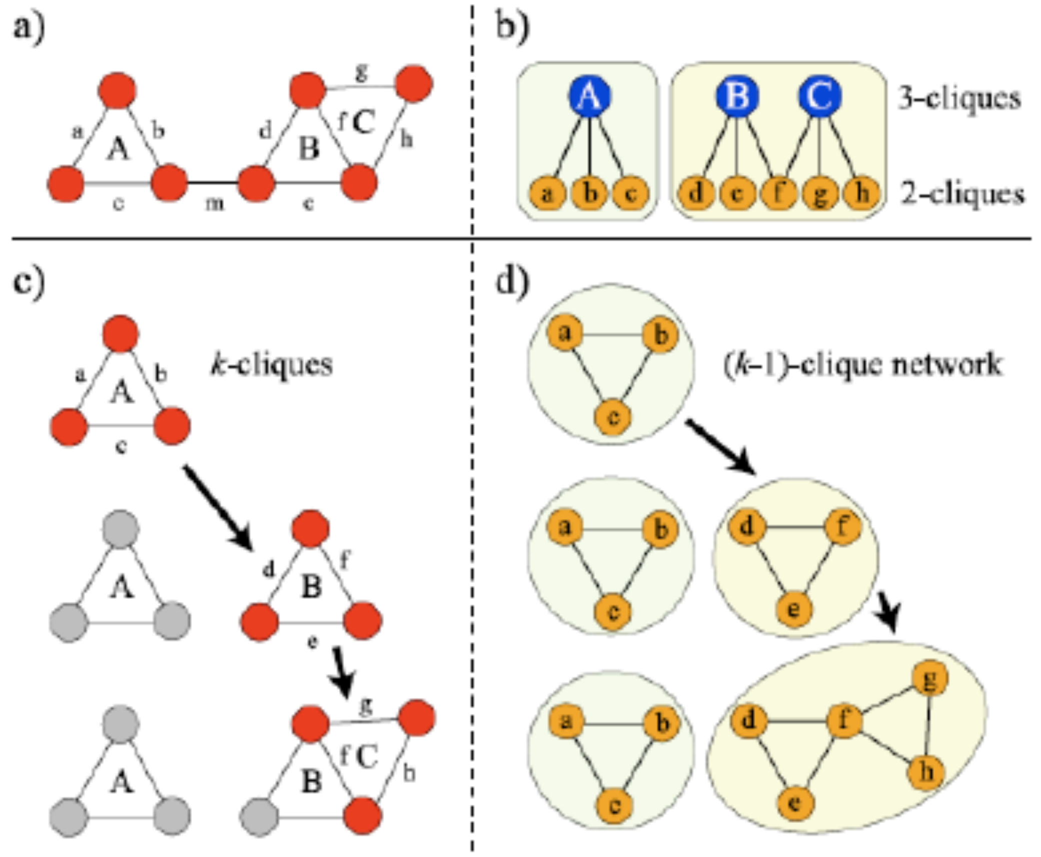
Hierarchical community structure

- Small communities form larger communities, which form larger communities, etc.
- For weighted networks, weights can define this hierarchy (strongest weights define the smallest communities etc)



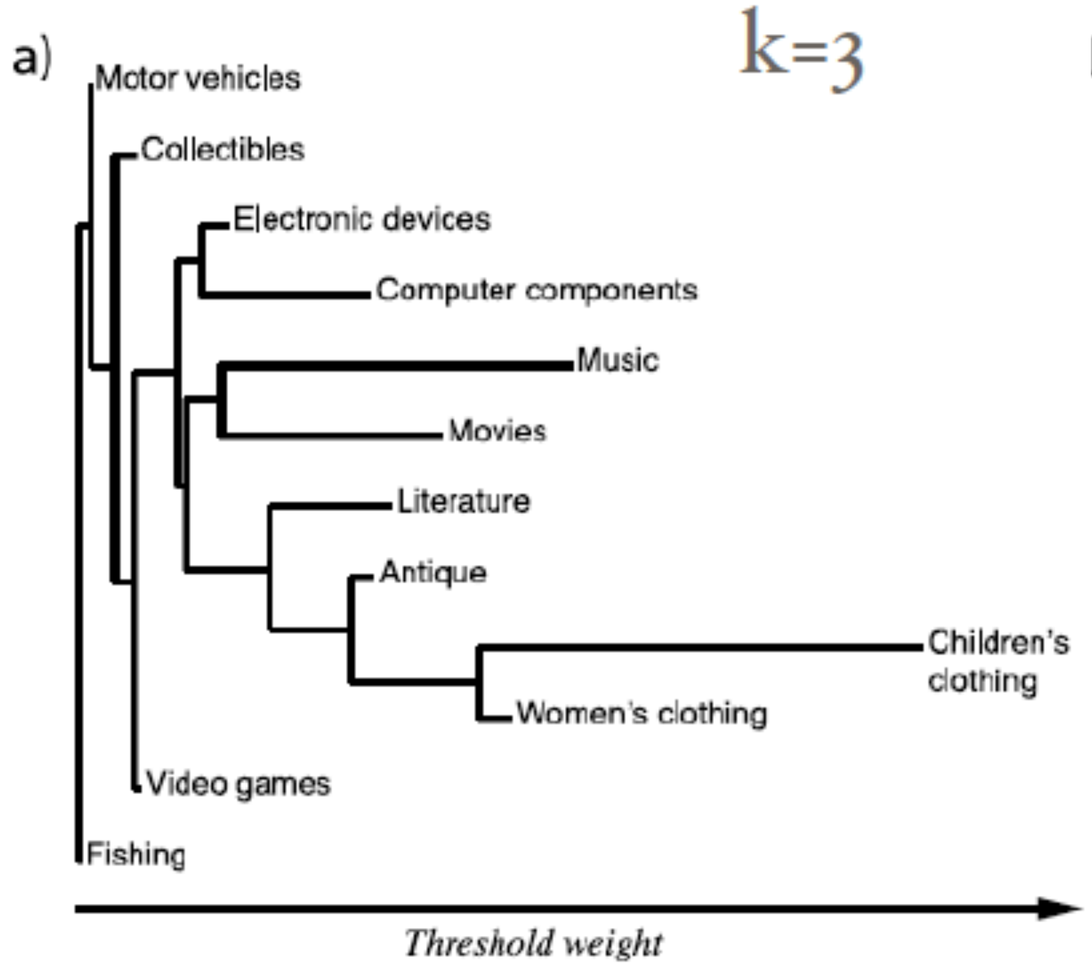
Sequential clique percolation

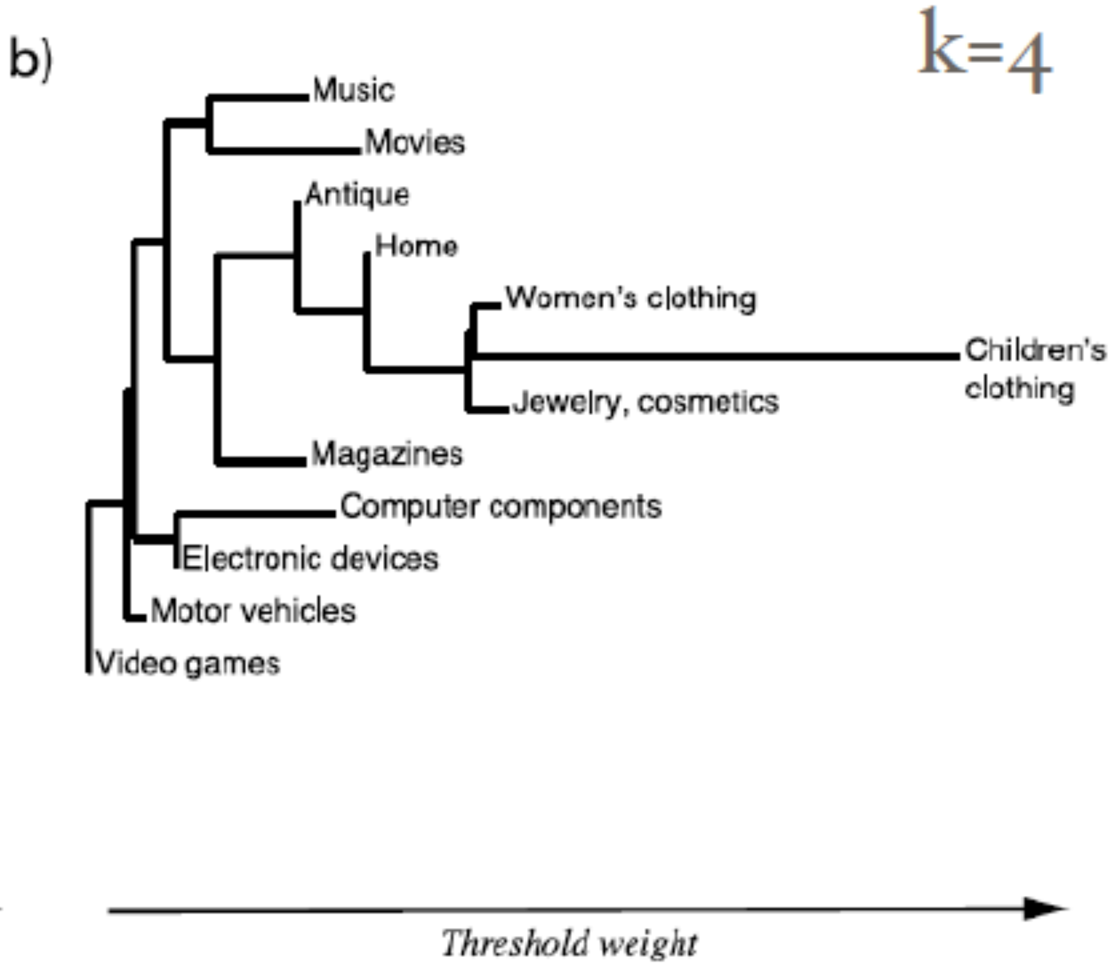
- Progressively threshold network, i.e. remove edges ordered by weight
- Simultaneously keep track of changes in *k*-clique communities
- A fast algorithm can be designed based on considering a bipartite clique network: just keep track of the connected components of a *k*-1 clique network



Kumpula et al, PRE 79, 026109 (2008)

Sequential clique percolation: community hierarchy in an e-auction network



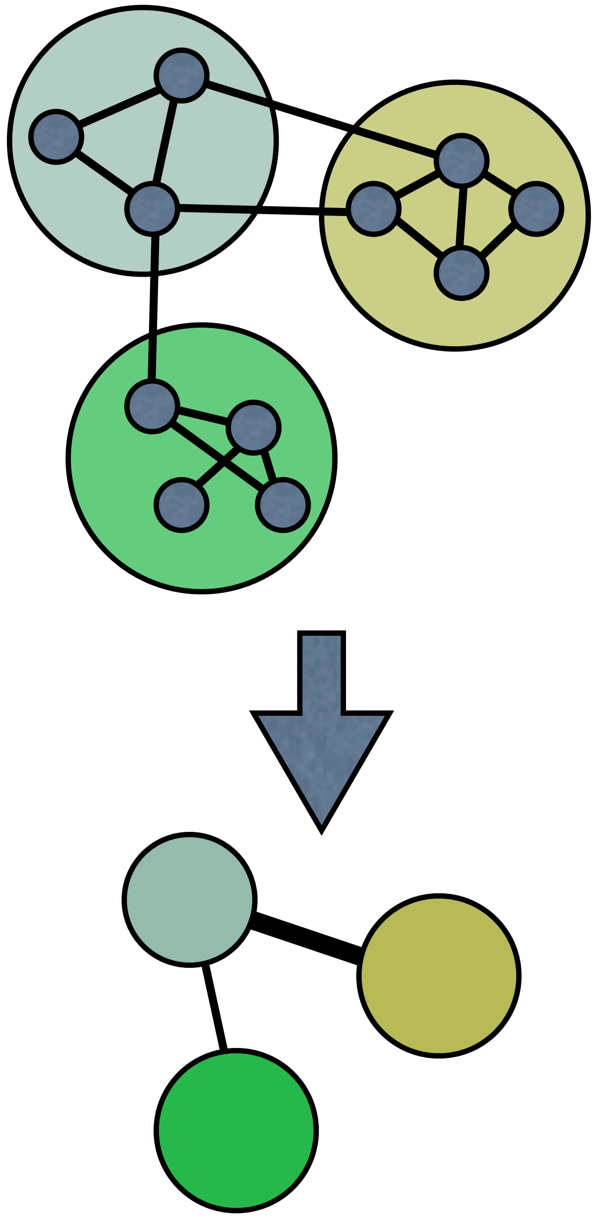


Kumpula et al, PRE 79, 026109 (2008)

Analysis of communities

Basic measures

- Size distribution of communities $P(s)$
- Edge densities inside communities $E_s/[0.5s(s-1)]$ as function of s
- Overall numbers of links inside/between communities
- If method allows for multiple community membership of a node (e.g. clique percolation), average # of communities per node; distribution of communities per node
- If extra information on nodes available (e.g. gender, age, nationality), relationship of these to community structure
- Community network: communities become nodes, numbers of links between communities become weighted links



Dynamic networks

Dynamic Networks

- Most studied networks are in fact "snapshots" of networks taken at a point in time, or aggregates over temporal activity
- What about the **dynamics** of networks?
- So far, studies have mostly focussed on the long time scale of network growth (e.g. whether attachment is preferential)

- Several time scales:
  - Fast time scale of **link activation/deactivation**
  - Longer time scale of **link rearrangement**
  - Long time scale of **network growth & decline**

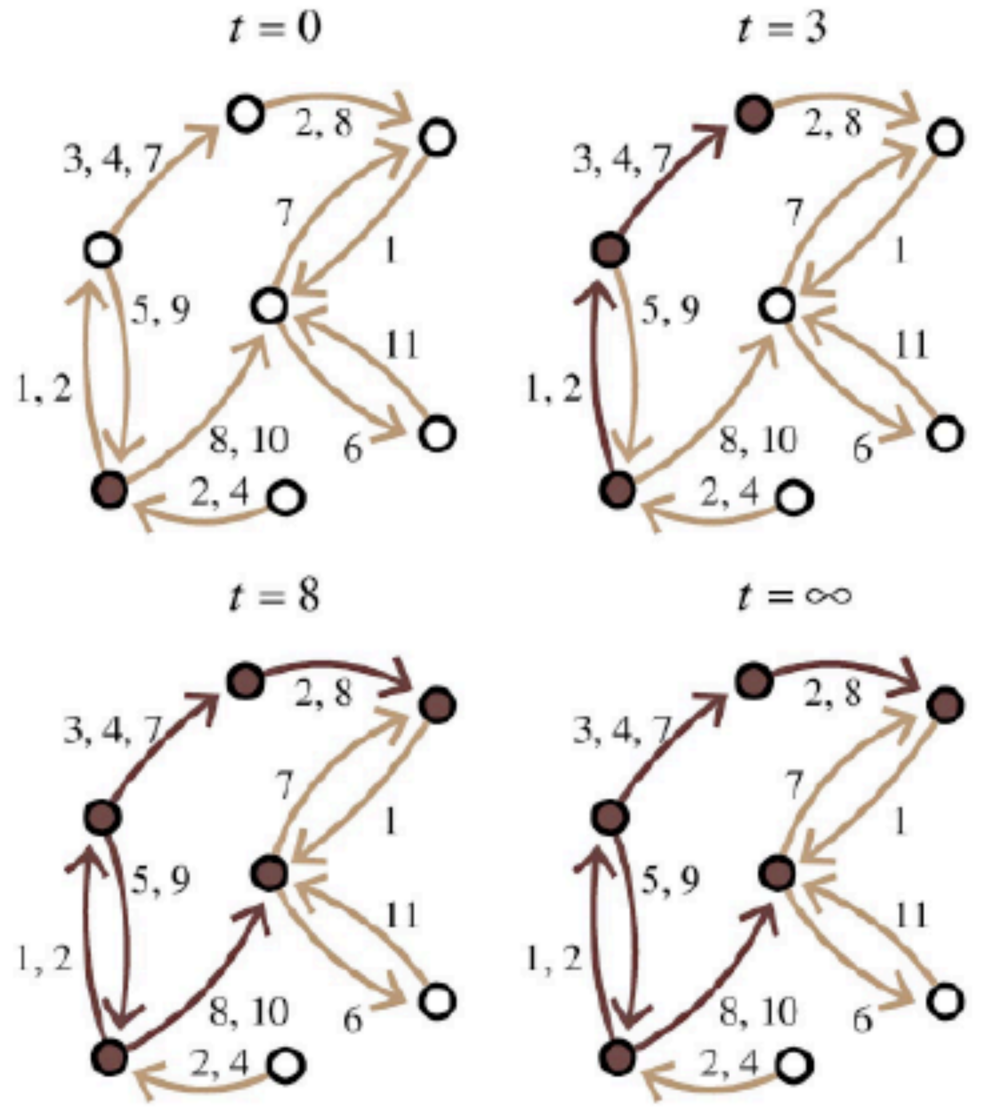
Communities: further reading

- A thorough review:
  Santo Fortunato: Community detection in graphs,
  http://arxiv.org/abs/0906.0612

- Good doctoral dissertation:
  J. Kumpula, Community Structures in Complex Networks: Detection and Modeling, http://lib.tkk.fi/Diss/2008/isbn9789512296569/

Short time scales: contact sequences and time-respecting paths

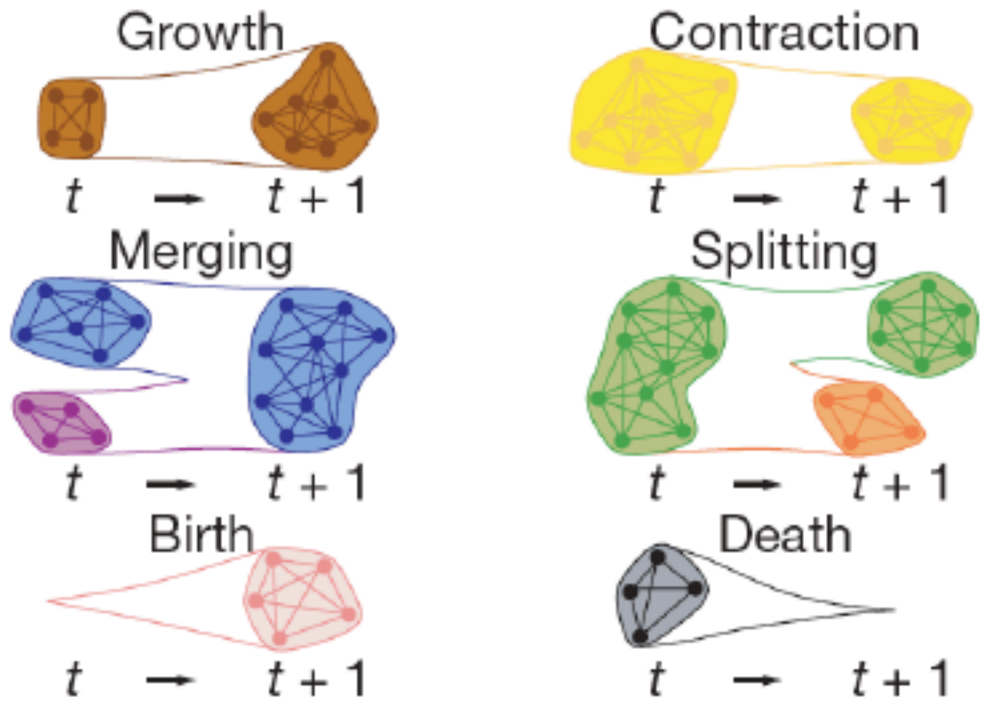
- Consider a network where links activate and deactive at points in time
- E.g. contacts between people
- Anything which spreads on such a network will have to "respect" this sequence of contacts
- The set of nodes which can be reached from a node along time-respecting paths is analogous to a connected component



P. Holme, Phys. Rev. E 71, 046119 (2005)

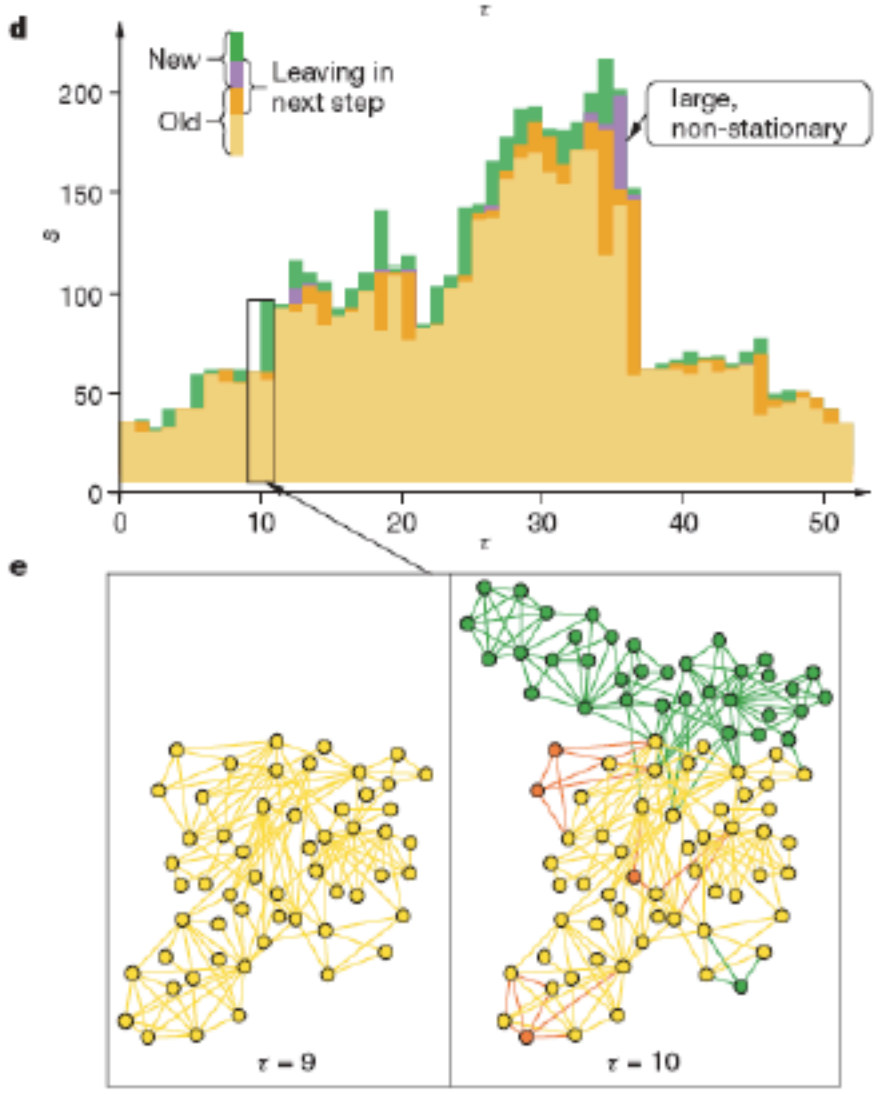
Medium time scales: Community Dynamics

- First study so far: Palla *et al*, Quantifying Social Group Evolution, *Nature* **446**, 664 (2007)
- Dynamics of communities: formation, merging, splitting, etc
- Data: co-authorships, mobile telephone call records
- Requires a lot from the community detection method!



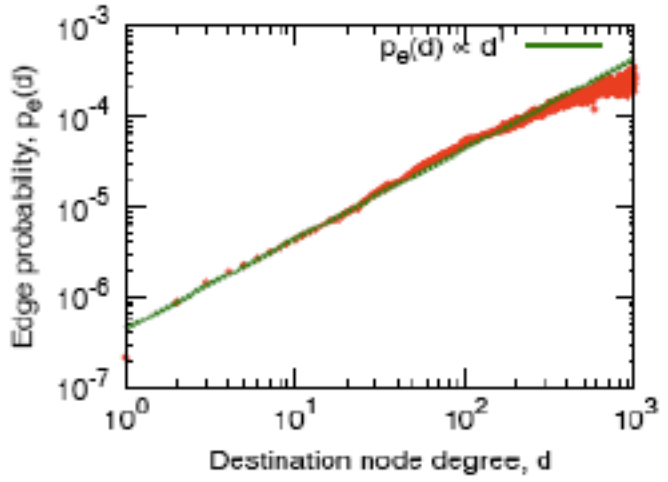
Community Dynamics: Stability of Communities

- Results indicate that
  - Large communities persist in time if there is a continuous exchange of members - communities remain, constituent people change!
  - Small communities persist if there is a small, persistent core of a few strong relationships

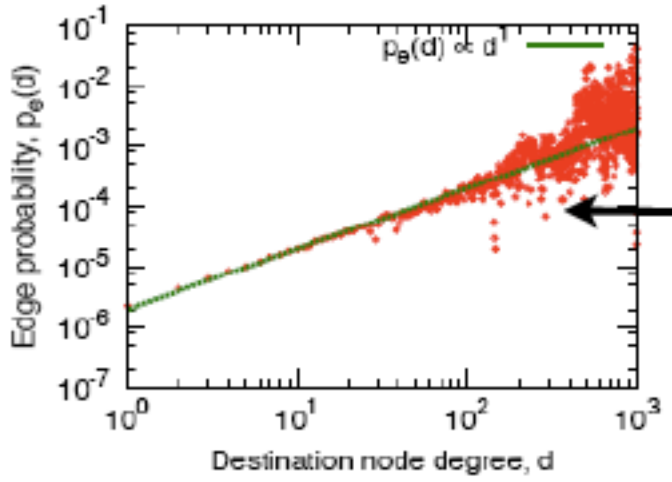


Medium to long time scales: edge dynamics & preferential attachment

Probability that a new edge connects to node of degree *d*

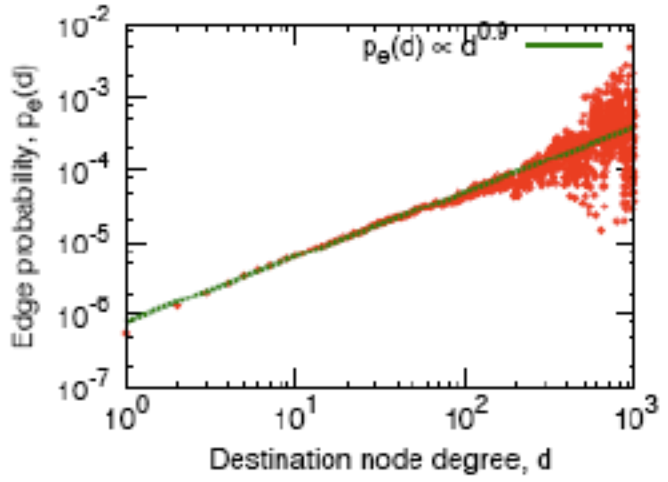


(c) FLICKR

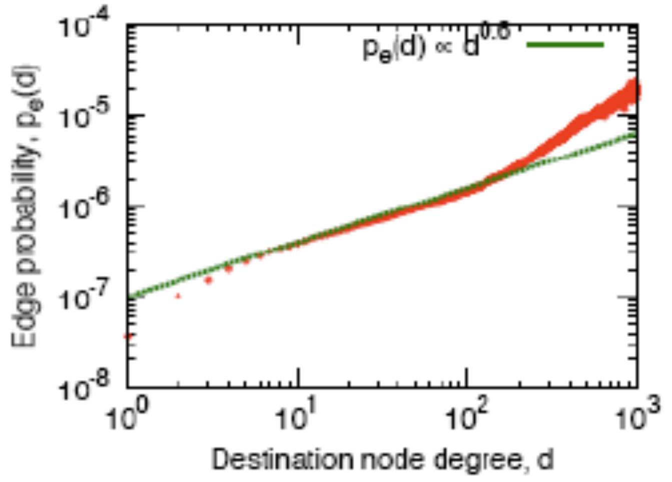


(d) DELICIOUS

Linear dependence observed: $p(d) \propto d$



(e) ANSWERS



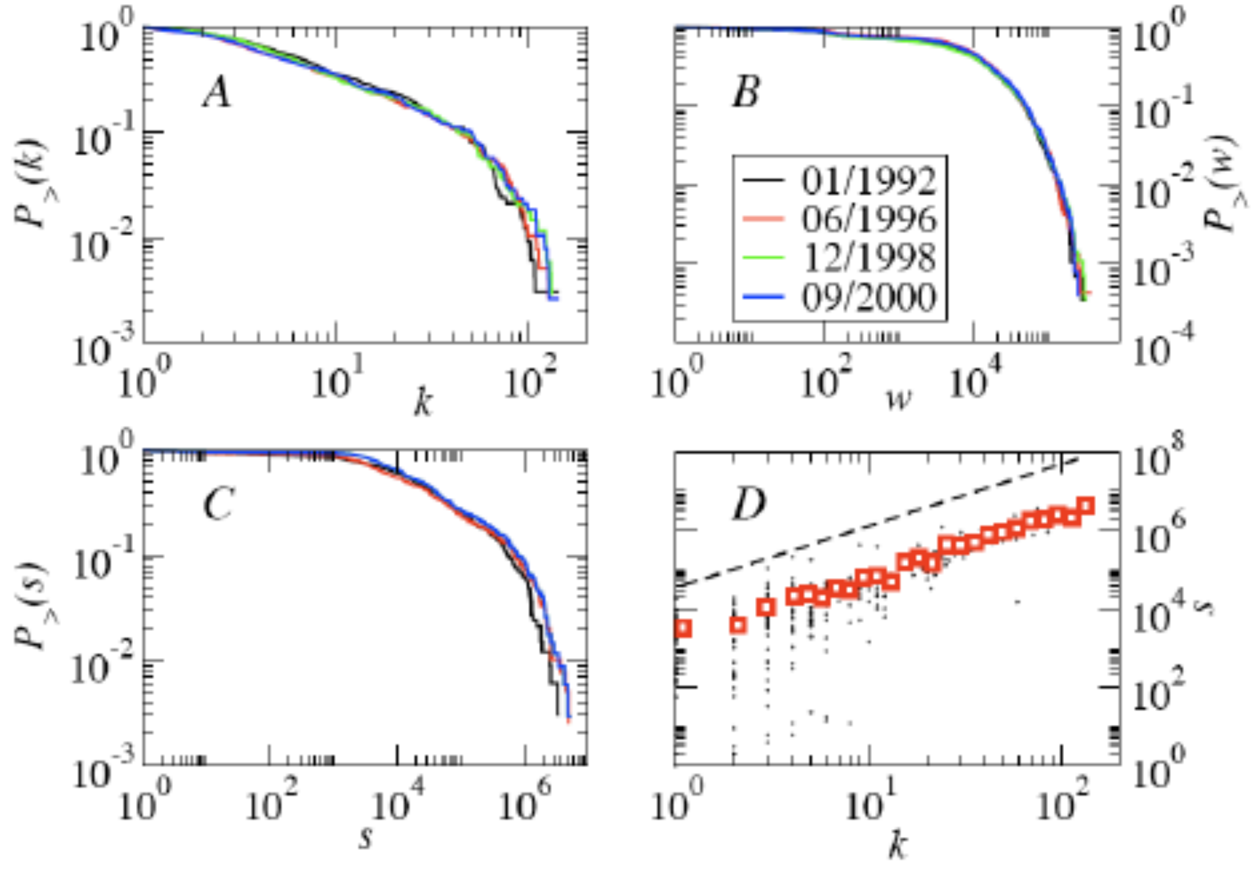
(f) LINKEDIN

Leskovec et al, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM KDD) (2008)

Medium to long time scales: dynamic networks, stationary distributions

- A. Gautreau et al,
  PNAS 106, 8847 (2009)
- Data: US airport network
- The network changes in time (nodes appear and disappear, degrees change, etc)
- Yet degree, strength and weight distributions remain stationary!

US Airport Network



Practical Tips & Tricks

Storing & handling network data

File formats:

1. Edge files (*.edg, ...)

- list of all edges
- if undirected network, edge i-j listed only once (j-i is not listed)
- one row contains:
  `node1 node2 (weight)`

2. GML (Graph Markup Language), (*.gml)

- lists node labels and then edges

```
graph
[
  directed 0
  node
  [
    id 0
  ]
  node
  [
    id 1
  ]

  ...

  node
  [
    id 4940
  ]


  edge
  [
    source 8
    target 6
  ]
  edge
  [
    source 8
    target 7
  ]
  edge
  [
    source 9
    target 8
  ]
]
```

undirected net

node ID's

edge endpoints

Storing & handling network data

- What kind of data structure should be utilized when simulating or analysing networks?
- Matlab users: if networks small enough (~thousands of nodes), use sparse matrices
- `A=spalloc(N,N,2*E)` creates an empty *sparse* adjacency matrix of size NxN, with space for E edges
- type `help sparse` for more
- Now just set an element to `1` if an edge exists between the endpoint nodes: `A(2,5)=1`
- If the network is undirected, also set the transpose: `A(5,2)=1`
- This matrix can also be used as the weight matrix W
- Note: requires that nodes are indexed `1..N` (which, anyhow, is a good practice)

Storing & handling network data

- Matlab users: another solution is to use structures which list neighbours for each node

-
  ```
  node(1).neighbours=[2 3 8];
  node(2).neighbours=[1 4 16];
  ```
  etc...

- Memory-efficient

- For efficient & fast storage e.g. with C++, see Efficient data structures for sparse network representation, J. Hyvönen, J. Saramäki, and K. Kaski, Int. J. Comp. Math. 85, 1219 (2008)

- How to get network properties now?

- E.g. for the degree of node 1:
  ```
  k=sum(A(1,:),1);
  ```
  or
  ```
  k=length(node(1).neighbours);
  ```

- vector with degrees of all nodes:
  ```
  k=sum(A,1);
  ```
  or
  ```
  for i=1:N,
    k(i)=length(node(i).neighbours);
  end;
  ```

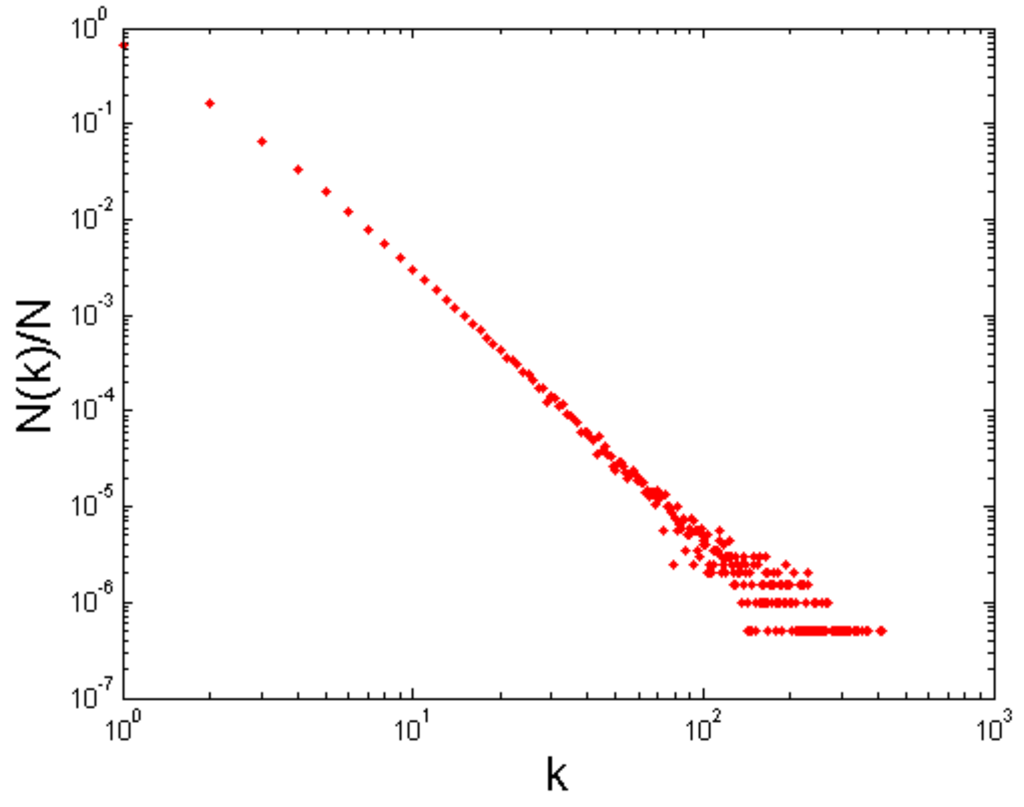
Example: E-R network model in Matlab

```
N=1000;
p=0.05;

A=spalloc(N,N,2*p*N*(N-1));
% creates a N*N sparse matrix with space for pN(N-1) edges
% (twice the expected number, just to be sure)

for i=1:N;
      for j=(i+1):N;
         r=rand(1,1);
         if r<p,
            A(i,j)=1;
            A(j,i)=1;
         end;
      end;
  end;
```

Degree Distributions

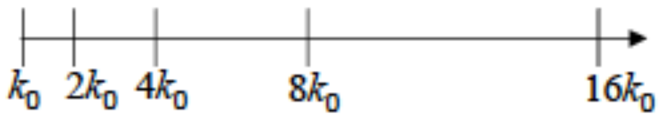
- Degree distribution $P(k)$ discrete
- The crude method (**not suggested**):
    - simply count $N(k)/N$ & plot
- Problem: not good enough for fat-tailed distributions due to missing values & noise at tail!

example: BA network, N=10000, averaged over 200 realizations



Estimating Degree Distributions: Logarithmic Binning

- Estimate probabilities of finding vertices within bins of multiplicative sizes
- Generate bin vector:

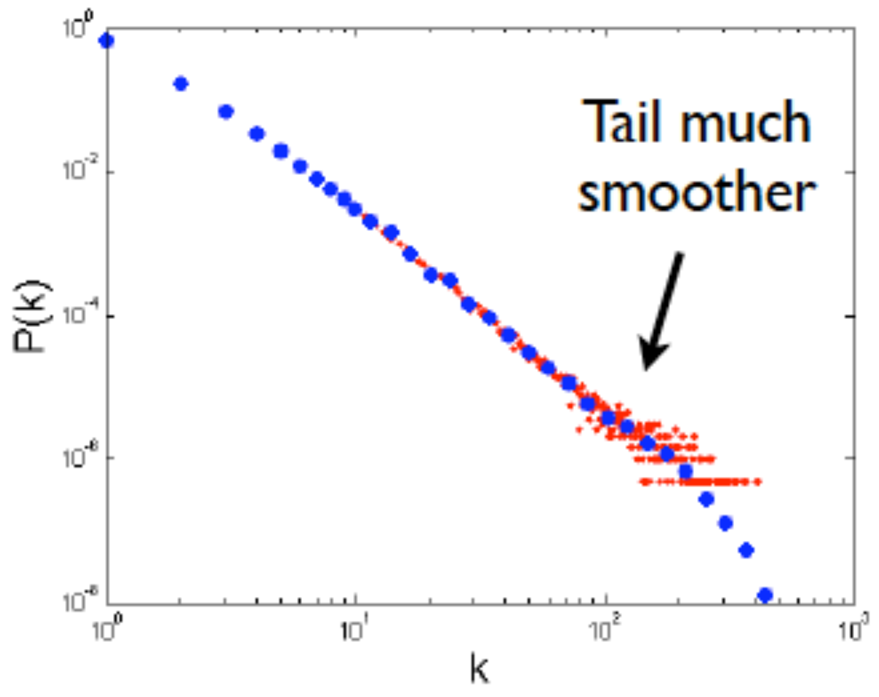


Here bin sizes double; any other factor than 2 will do as well, as long as there are enough data points within bin limits (e.g. 1.2)

- Count numbers of nodes in each bin, i.e. # of nodes with degrees within bin limits, N_{bin}
- Divide by bin width to get $P_{bin} = N_{bin} / \Delta k_{bin}$
- Normalize P_{bin}

- Plot P_{bin} vs bin center degree

example: BA network, N=10000, averaged over 200 realizations



● = log-binned

Note: for small degrees, e.g. in [1,10], linear bins are often used

Example: log binning with Python

Step 1: generate bin limit vector

```
def generateLogbins(minvalue,maxvalue,factor,uselinear=True):
    '''Generates a binning vector containing bin limits
       for log-binning. Inputs:
       minvalue, maxvalue = min and max values of data (e.g. degrees) to be binned,
       factor = multiplicative factor for increasing bin size,
       uselinear=[True|False] for making the first 10 bins linear.'''

    if uselinear:
          # for degree distributions, the first 10 degrees are often
          # binned in linear bins. If so, set uselinear=True

        bins=[-0.5,0.5,1.5,2.5,3.5,4.5,5.5,6.5,7.5,8.5,9.5,10.5]
        i=12

    else:

          # set the first bin lower limit s.t. first data point
          # falls in the bin center

        bins=[]
        bins.append(minvalue*2.0/(1+factor))
        i=1

    while bins[i-1]<maxvalue:

          # generate the rest of (log) bin limits. The next bin limit
          # is always the previous limit times the factor

        bins.append(bins[i-1]*factor)
        i+=1

    return bins
```

Example: log binning with Python

Step II: count # of nodes in bins, divide by width to get distribution

```python
# first initialize a vector that counts the number of data points
# within each bin (e.g. degrees)

Nbin=[]

for i in range(len(bins)-1):
          Nbin.append(0)

# go through each degree in the network, and find which bin the
# degree belongs to. Increase the number counter of that bin by
# one.

for k in degrees:
          mybin=findbin(bins,k)
          Nbin[mybin]+=1

# next, calculate probability density per bin, and also bin centers

Pbin=[]
bincenters=[]

for i in range(len(bins)-1):

          binwidth=bins[i+1]-bins[i]
          # to get prob. density, divide number of data points by bin width
          Pbin[i]=Nbin[i]/binwidth
          bincenters[i]=(bins[i+1]-bins[i])/2.0

# finally normalize probabilities to sum to one
Pbin=Pbin/sum(Pbin)
```

```python
def findbin(bins,value):
    '''Finds out the bin where input param value belongs to.
    Uses range halving for fast output - move lower and upper limits
    of search according to whether the input param value is higher or
    lower than the bin limit in the middle of the (lower,upper) range.
    Inputs: bins - list of bin limits,
    value - value to be found within bin limits'''

    lowerlimit=0
    upperlimit=len(bins)-1

    while (upperlimit-lowerlimit)>1:

        halfpoint=int(ceil(0.5*(upperlimit+lowerlimit)))

        if (value>=bins[halfpoint]):

            lowerlimit=halfpoint

        else:

            upperlimit=halfpoint

    return lowerlimit
```

Example: log binning with Matlab

create bin limit vector: linear up to 10, then limits multiplied by const factor

```
function [P,kc]=logbin(Nk,kmax,factor);
% Nk: vector whose k:th element= #of nodes of degree k, kmax=max k
bins=0.5:10.5; % vector of bin LOWER LIMITS
               % up to 10, bins LINEAR, after 10, LOGARITHMIC
i=12;
while bins(i-1)<kmax,
    bins(i)=bins(i-1)*factor; % factor controls bin spread
    i=i+1;
end;
```

count how many vertices in each bin

```
Nbin=zeros(1,length(bins)); % vector for total # of vertices in bin
bin=0;
for k=1:length(Nk),     % here Nk is SORTED in ascending k
    if k>=bins(bin+1), % if k exceeds lower bin limit of next bin
      bin=bin+1;        % move to next bin
    end;
    Nbin(bin)=Nbin(bin)+Nk(k); % add data vect contents to bin sum
end;
```

divide by bin width

```
for i=1:length(bins)-1,      % divide by bin width
    width=bins(i+1)-bins(i);
    Nbin(i)=Nbin(i)/width;
end;
```

calculate bin center degrees

```
for i=1:length(bins)-1,
    kc(i)=0.5*(bins(i)+bins(i+1)); % vector of bin centers
end;
```

normalize

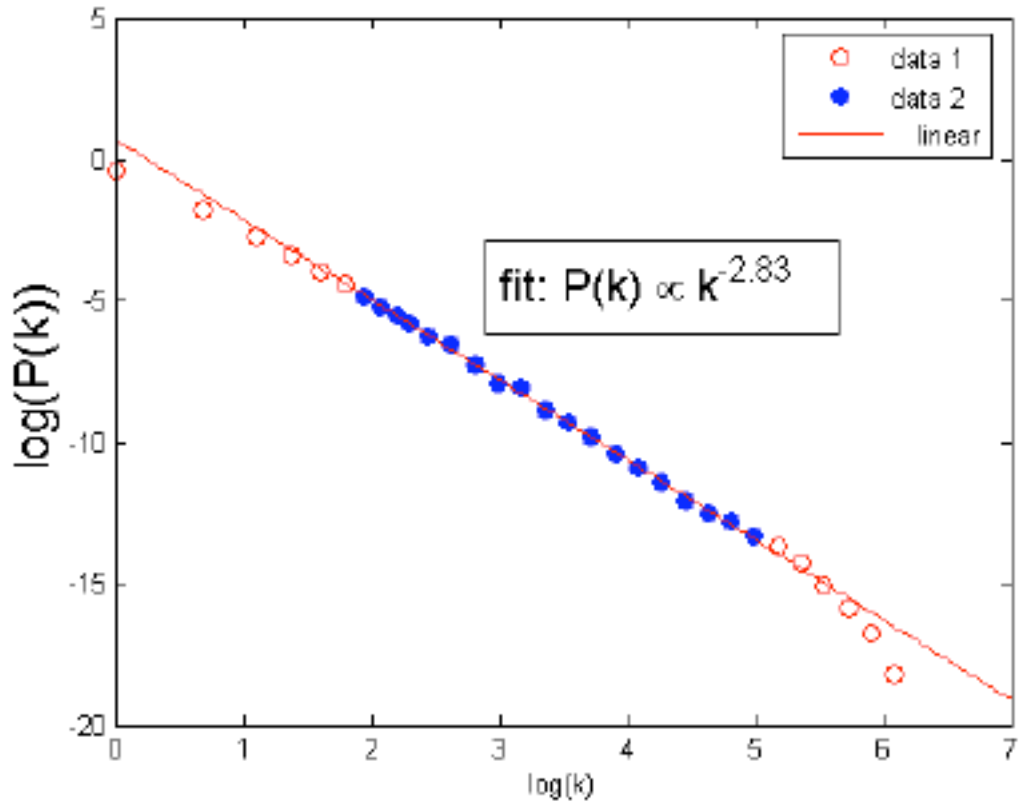
```
P=Nbin(1:length(bins)-1);
P=P/sum(Nk); % probability density vector, normalized
```

Estimating power law exponents (the crude, easy way)

- Fit a line to $\log(P(k))$ vs $\log(k)$
- If the distribution follows a power law, this typically happens at the tail, so the fitting range has to be chosen
- Exponents tend to be underestimated!
- Left: BA networks: fit ϒ=2.83, true ϒ=3 at large N

(BA power law is asymptotic, actually P(k) = 2m(m+1)/k(k+1)(k+2), and N=10000 is not large enough)

example: BA network, N=10000, averaged over 200 realizations



● = used in fit

For learning how to **properly** statistically test the hypothesis whether the data is a power law (or some other distribution), see A. Clauset et al, SIAM Review 51, 661-703 (2009) (also at http://arxiv.org/abs/0706.1062)

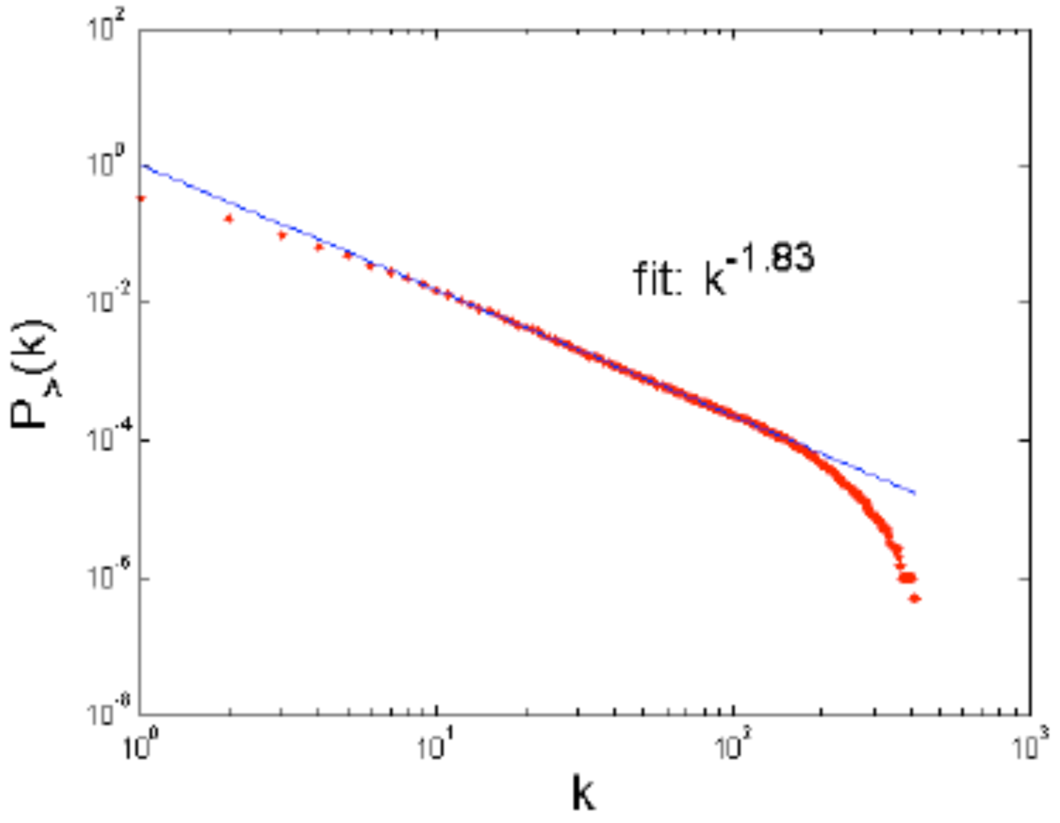
Estimating Degree Distributions: Cumulative Distributions

- For power laws
$$P(k > k') \sim \sum_{k'=k}^{\infty} k'^{-\gamma} \sim k'^{-(\gamma-1)}$$
i.e. the cumulative distribution is also a power law

- For other distributions, depends... (exp → exp, lognormal → something strange, etc)

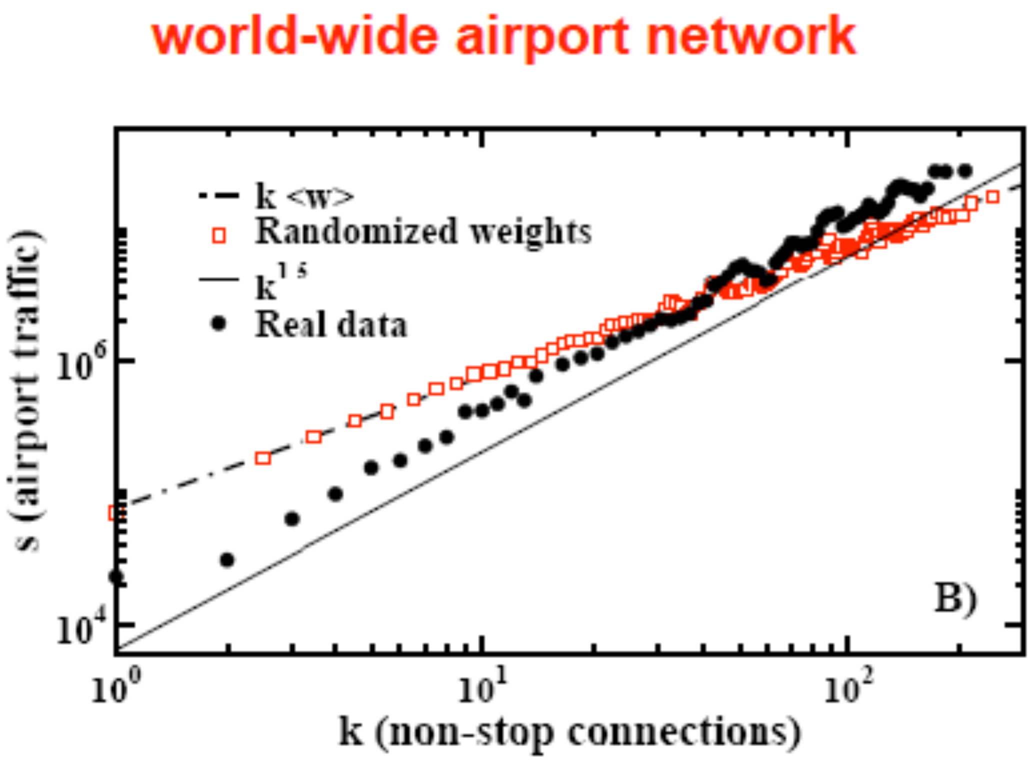
- Recipe: sort nodes by degree, for each degree count how many nodes have a higher degree, divide by N

example: BA network, N=10000, averaged over 200 realizations



Log-binned averages

- Just as for degree distribution, create bin vector
- For your quantity of interest, e.g. C(k), calculate the average for nodes with degrees inside the bin
- Plot as function of bin centers



Clustering coefficient

- If using (sparse) adjacency matrix, easy to calculate:

```
t=0.5*diag(A^3);
% vector with # of triangles
around each node
k=sum(A,1); %degrees

c=zeros(1,N);
for i=1:N,
if (k(i)>1), c(i)=2*t(i)/(k(i)*
(k(i)-1));
end; end;
```

Note: c is not defined for vertices of degree 1!
Here, it is chosen to be 0 for these

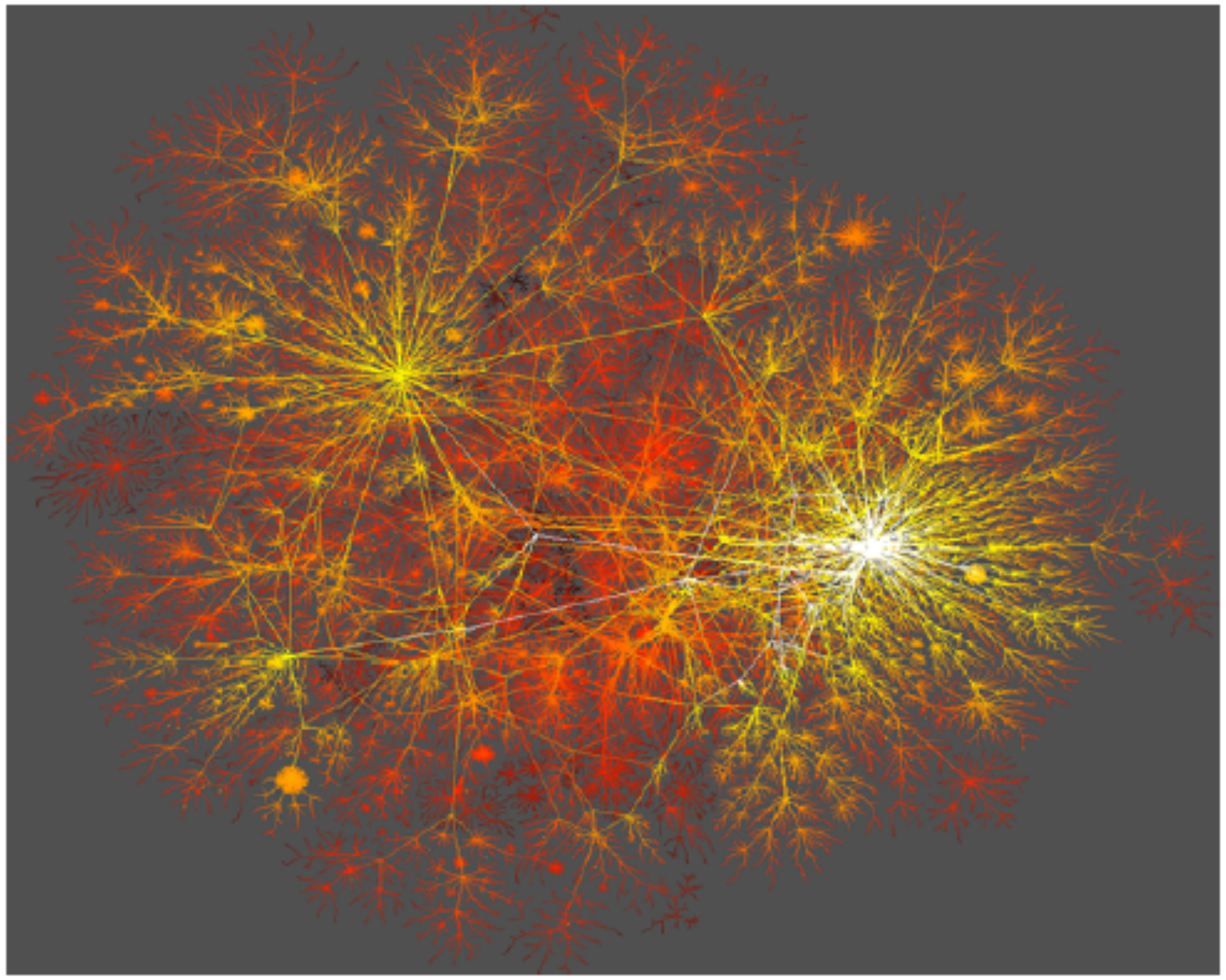
- If using link lists, you have to use loops to get the # of links in the neighbourhood of each vertex
- Then just average over all vertices of each degree to get $c(k)$ & plot $c(k)$ vs k on (usually log-log)
- OR, average in logarithmic bins, ie over all vertices falling in each bin & plot as function of bin center degrees

Network Visualization

- The not-so-easy target: to map a hyperdimensional object into 2D

- Many algorithms exist, most are variations of the following:

    - Nodes = repulsive electric charges, amount of charge equals degree
    - Links = springs
    - Then just balance the tensions, i.e. minimize energy, using simulated annealing or other tricks



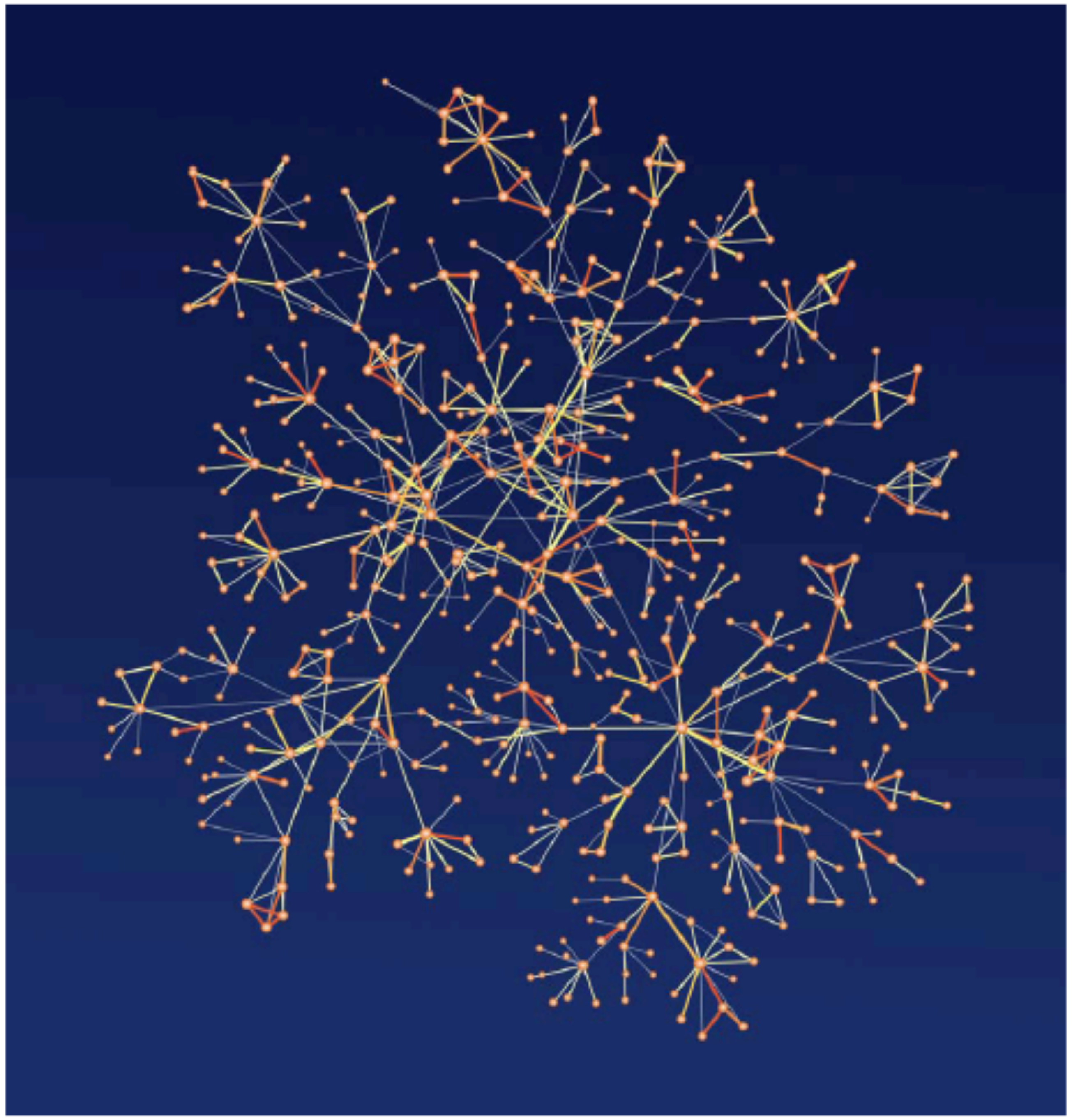
Network Visualization: Freeware Packages

- PAJEK (Win only), http://vlado.fmf.uni-lj.si/pub/networks/pajek/

- Himmeli (Win/Mac/Linux) http://www.finndiane.fi/software/himmeli/

  - Himmeli allows for weighted networks.

  - There is also an online version (data in through a web interface, images out).



Online resources

Network data

- Mark Newman's data and link collection: http://www-personal.umich.edu/~mejn/netdata/
- Uri Alon's pages (biological networks, also motif analysis software): http://www.weizmann.ac.il/mcb/UriAlon/

Software - Community Detection

- INFOMAP - http://www.tp.umu.se/~rosvall/code.html
- CFinder (clique percolation) - http://www.cfinder.org/

Software - Network Visualization

- Himmeli - http://www.finndiane.fi/software/himmeli/
- Pajek - http://vlado.fmf.uni-lj.si/pub/networks/pajek/

Software - General

- iGraph, a general network library (C++, R, Python): http://igraph.sourceforge.net/

Online resources

Research groups, laboratories, people

- Institute for Scientific Interchange, Torino, Italy: http://www.isi.it/
- CABDyN Complexity Center, Oxford, UK: http://www.cabdyn.ox.ac.uk/
- Center for Models of Life, Niels Bohr Institute, Denmark: http://cmol.nbi.dk/index.php
- Large Graphs and Networks Group at UCL, Louvain-la-Neuve, Belgium: http://www.inma.ucl.ac.be/networks/
- Institute for Cross-Disciplinary Physics and Complex Systems, Mallorca, Spain: http://ifisc.uib.es/
- Albert-László Barabási's group: http://www.barabasilab.com
- Nicholas Christakis at Harvard Dept. of Sociology: http://www.wjh.harvard.edu/soc/faculty/christakis/
- Mark Newman: http://www-personal.umich.edu/~mejn/
- My department in Helsinki: http://www.becs.hut.fi/

Online resources

Video lectures

- http://videolectures.net/janos_kertesz/
- http://videolectures.net/albert_laszlo_barabasi/
- http://videolectures.net/shlomo_havlin/
- http://videolectures.net/mark_newman/
- http://videolectures.net/alessandro_vespignani/
- http://videolectures.net/santo_fortunato/

Online resources

Doctoral theses with good introductory texts on complex networks (+lots of references)

- Jussi Kumpula: Community structure in complex networks - detection and modeling. http://lib.tkk.fi/Diss/2008/isbn9789512296569/isbn9789512296569.pdf
- Tapio Heimo: Complex networks and spectral methods: an econophysics approach to equity markets http://lib.tkk.fi/Diss/2009/isbn9789512297313/isbn9789512297313.pdf
- Riitta Toivonen: Social networks: modelling structure and dynamics. http://www.lce.hut.fi/~rtoivone/publications/DissertationIntroduction_RiittaToivonen.pdf

My contact details

- email: jari.saramaki@tkk.fi
- Skype: jari.saramaki
- web page: http://www.lce.hut.fi/~jsaramak/