Community Structure in Networks

Thanks to Jure Leskovec, Stanford and Panayiotis Tsaparas, Univ. of Ioannina for slides

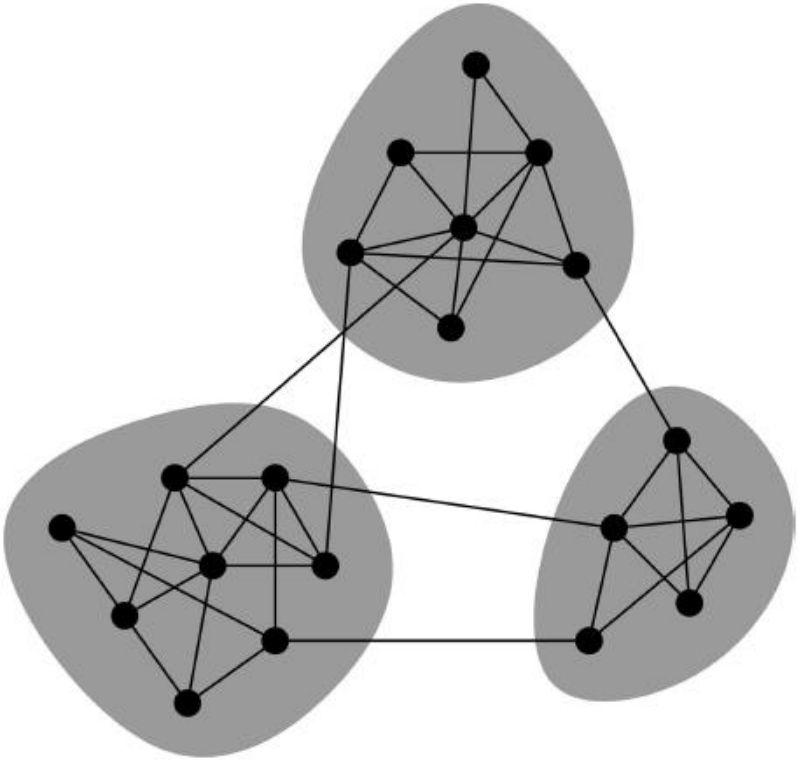
Agenda

- Network Communities
- Community Detection
  - Method 1: Girvan-Newman
  - Method 2: Modularity Optimization
- Community Detection
  - Graph Cuts
  - Spectral Clustering
- Network Profiling
- Communities: Issues and Questions

Network Communities

Networks & Communities

- We often think of networks "looking" like this:



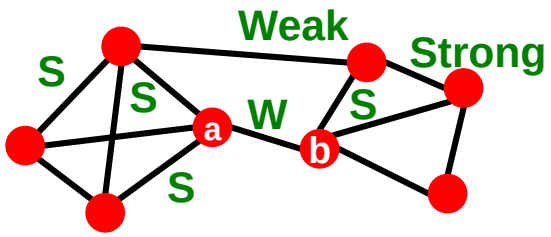
- What lead to such a conceptual picture?

Networks: Flow of Information

- **How information flows through the network?**
  - **What structurally distinct roles do nodes play?**
  - **What roles do different links (short vs. long) play?**
- **How people find out about new jobs?**
  - Mark Granovetter, part of his PhD in 1960s
  - People find the information through personal contacts
- **But:** Contacts were often **acquaintances** rather than close friends
  - **This is surprising:** One would expect your friends to help you out more than casual acquaintances
- **Why is it that acquaintances are most helpful?**

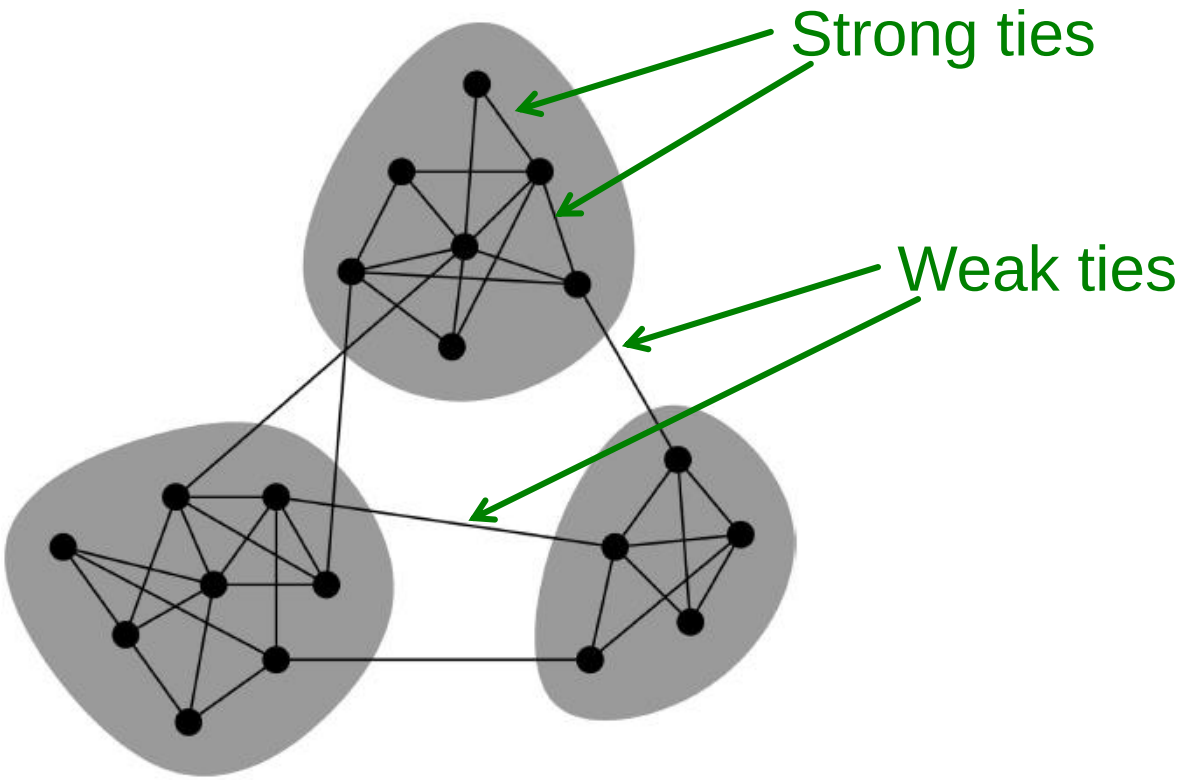
Granovetter's Explanation

- **Granovetter makes a connection between social and structural role of an edge**
- **First point: Structure**
  - Structurally embedded edges are socially **strong**
  - Long-range edges spanning different parts of the network are socially **weak**
- **Second point: Information**
  - Long-range edges allow you to gather information from different parts of the network and get a job
  - Structurally embedded edges are heavily redundant in terms of information access



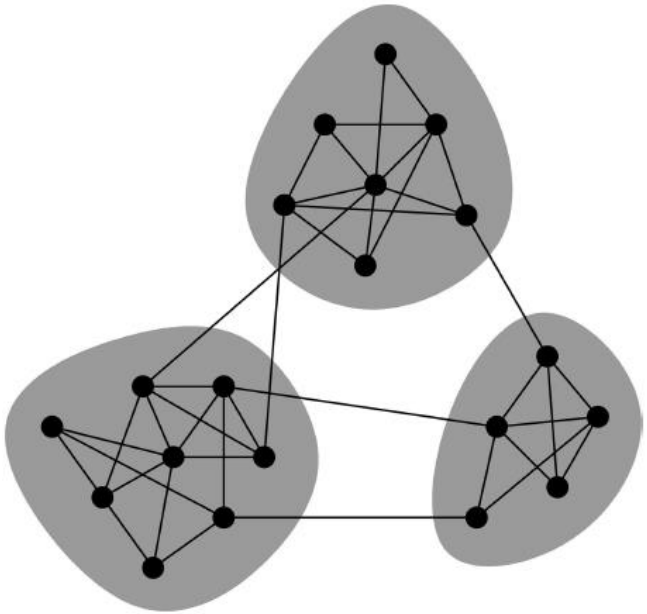
Conceptual Picture of Networks

- **Granovetter's theory leads to the following conceptual picture of networks**



Network Communities

- Granovetter's theory suggest that networks are composed of **tightly connected sets of nodes**

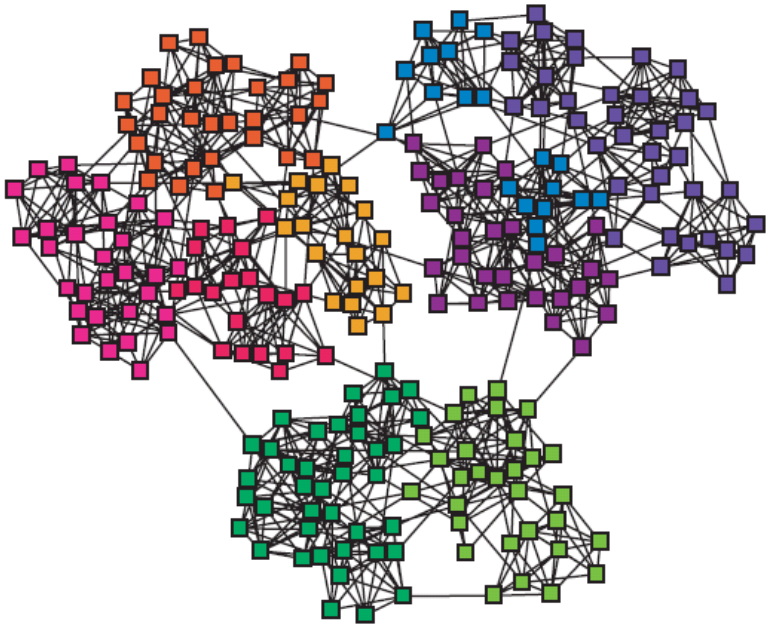


Communities, clusters, groups, modules

- **Network communities:**
  - Sets of nodes with **lots** of connections **inside** and **few** to **outside** (the rest of the network)

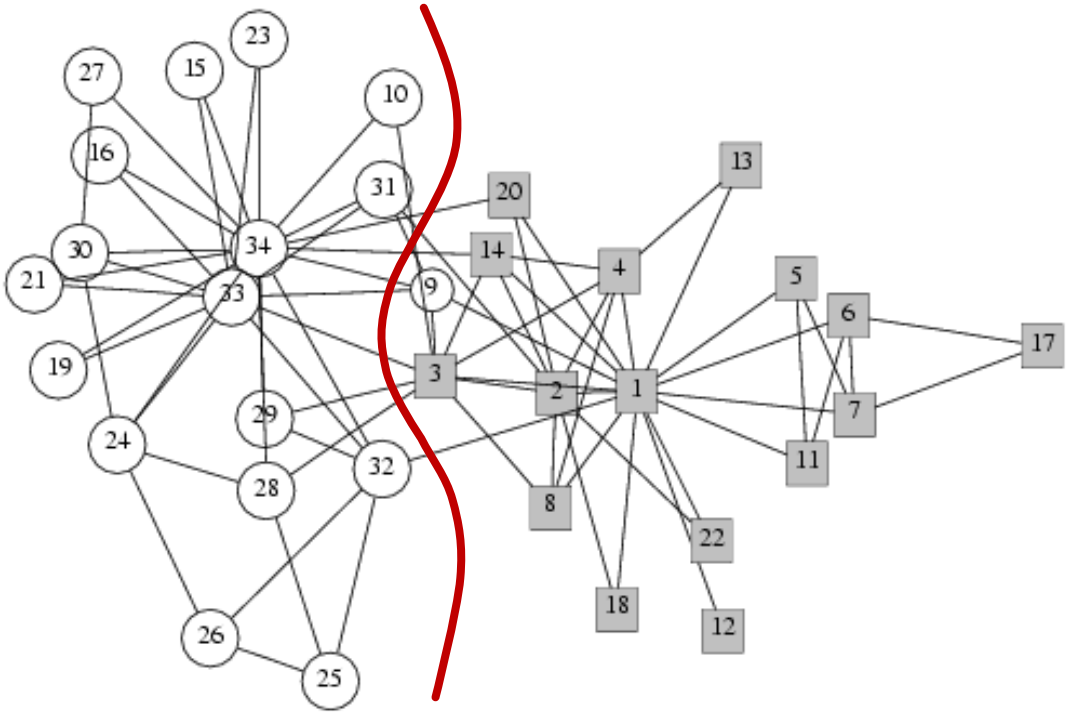
Finding Network Communities

- **How to automatically find such densely connected groups of nodes?**
- Ideally such automatically detected clusters would then correspond to real groups
- **For example:**



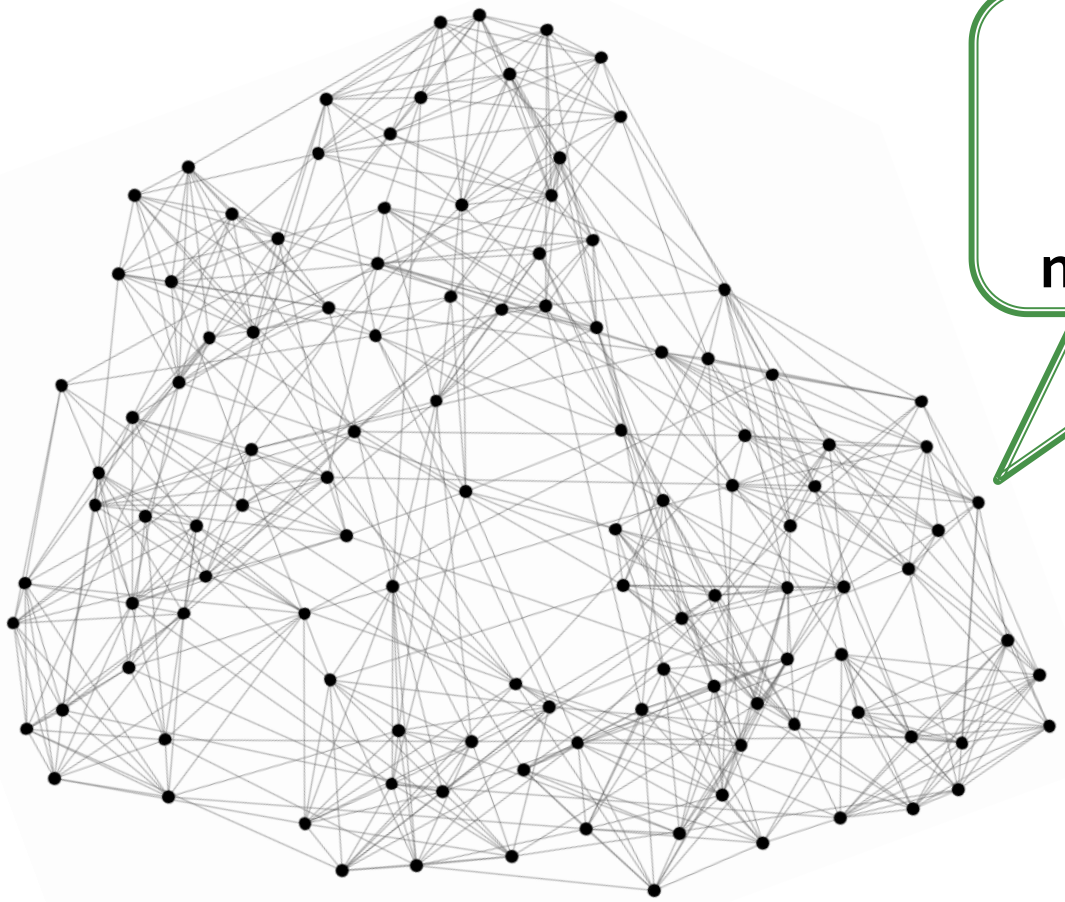
Communities, clusters, groups, modules

Social Network Data



- **Zachary's Karate club network:**
  - Observe social ties and rivalries in a university karate club
  - During his observation, conflicts led the group to split
  - Split could be explained by a minimum cut in the network

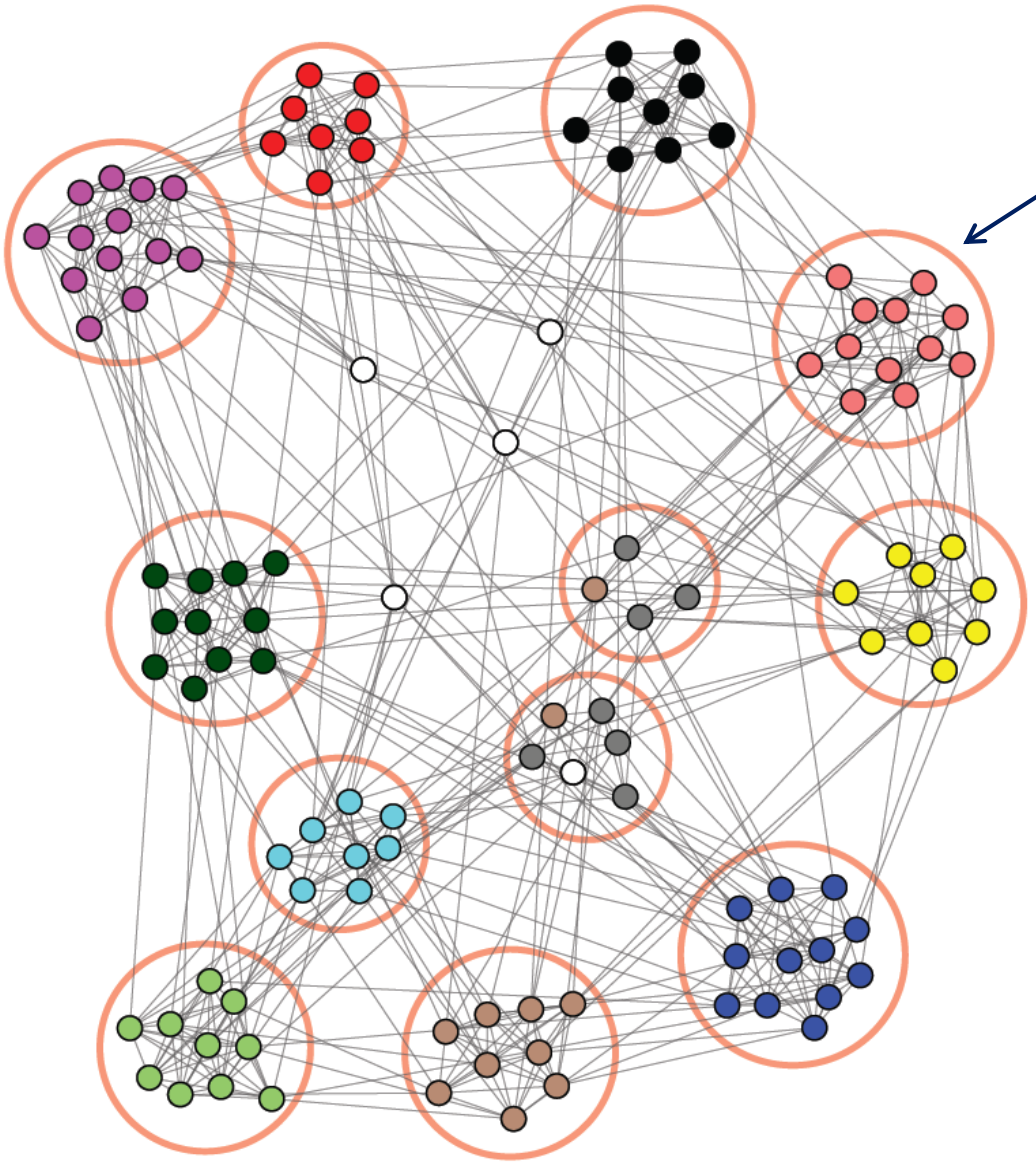
NCAA Football Network



Can we identify node groups? (communities, modules, clusters)

Nodes: Teams
Edges: Games played

NCAA Football Network

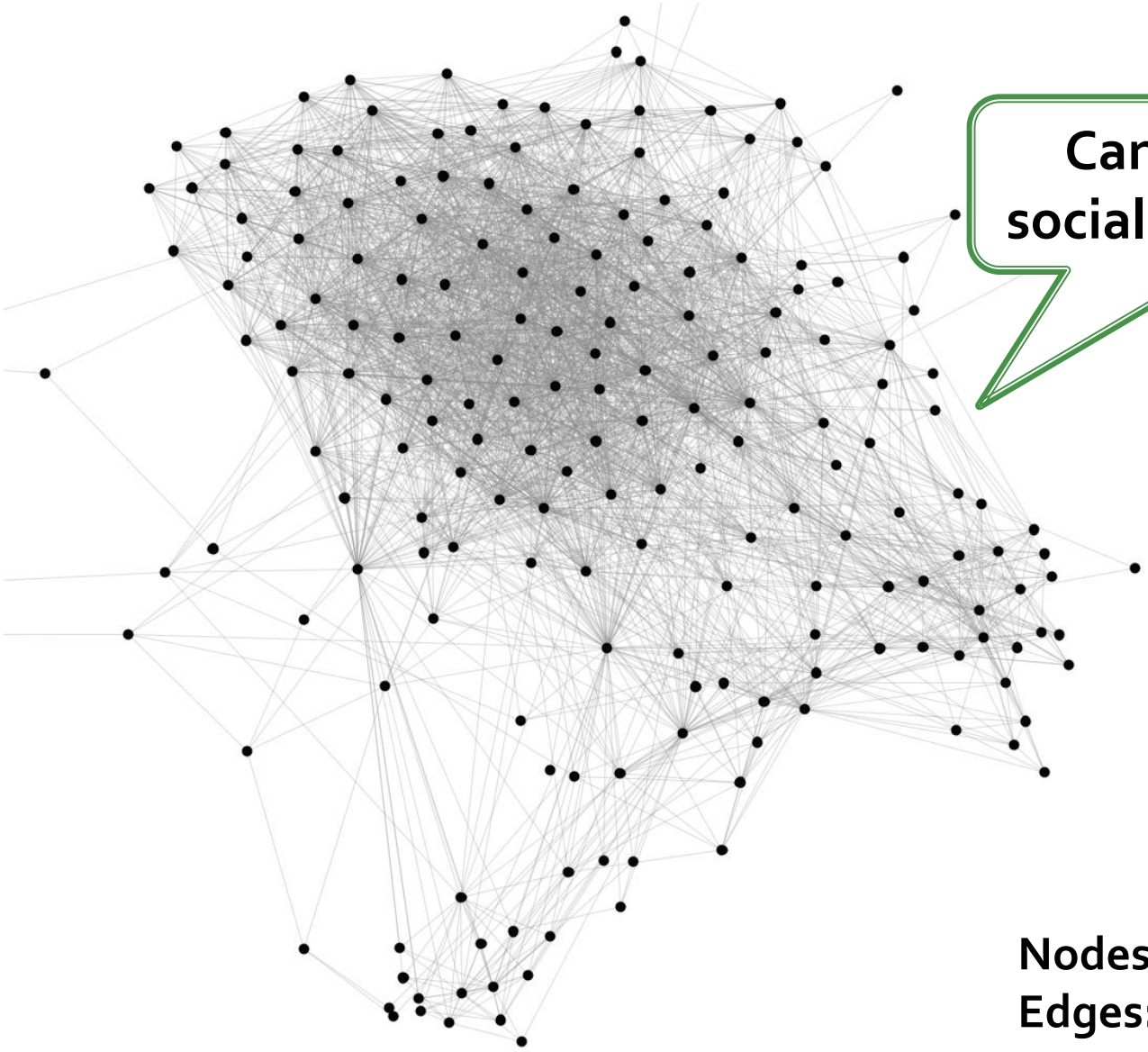


NCAA conferences

- Mid American
- Big East
- Atlantic Coast
- SEC
- Conference USA
- Big 12
- Western Athletic
- Pacific 10
- Mountain West
- Big 10
- Sun Belt
- Independents

Nodes: Teams
Edges: Games played

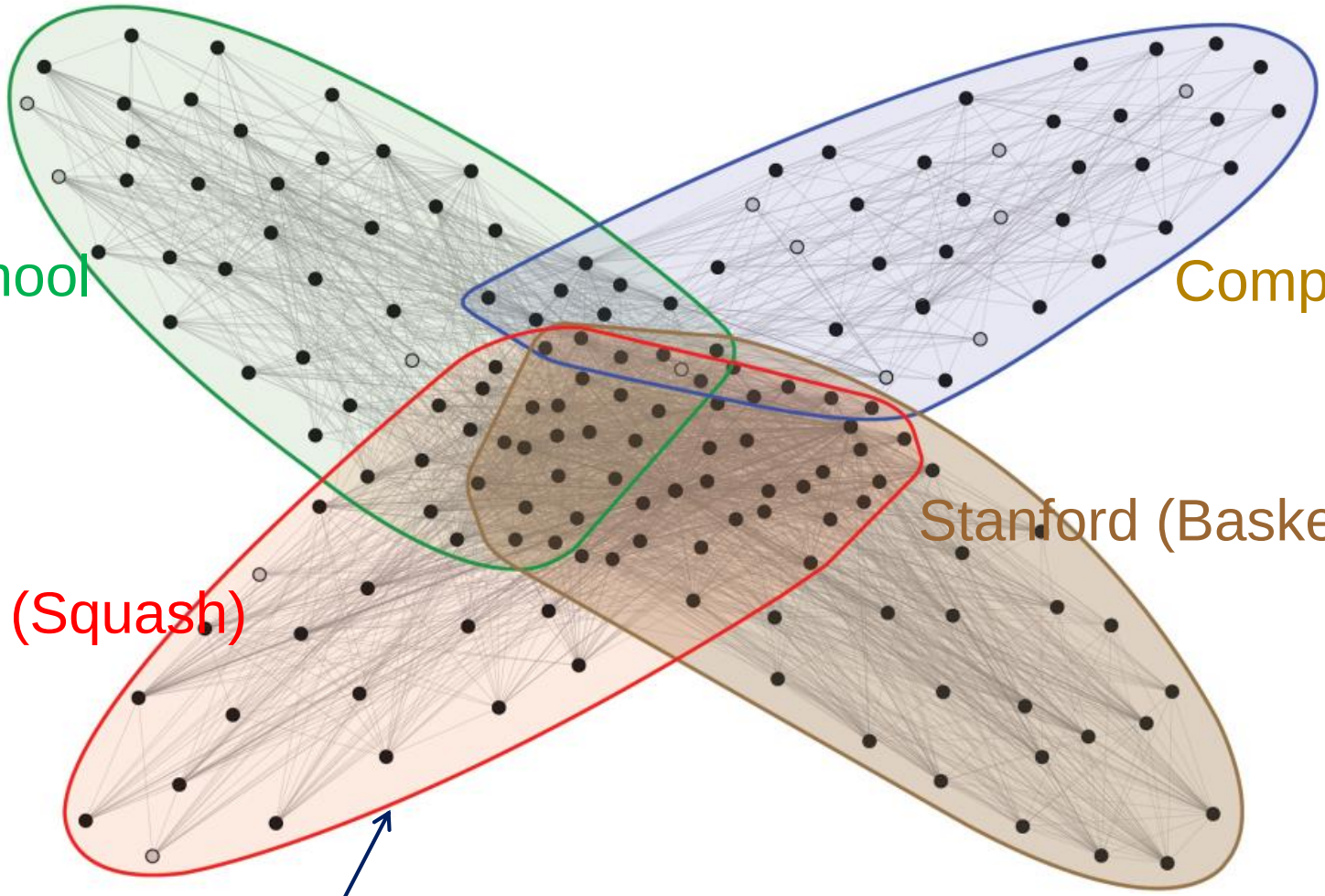
Facebook Ego-network



Can we identify social communities?

Nodes: Users
Edges: Friendships

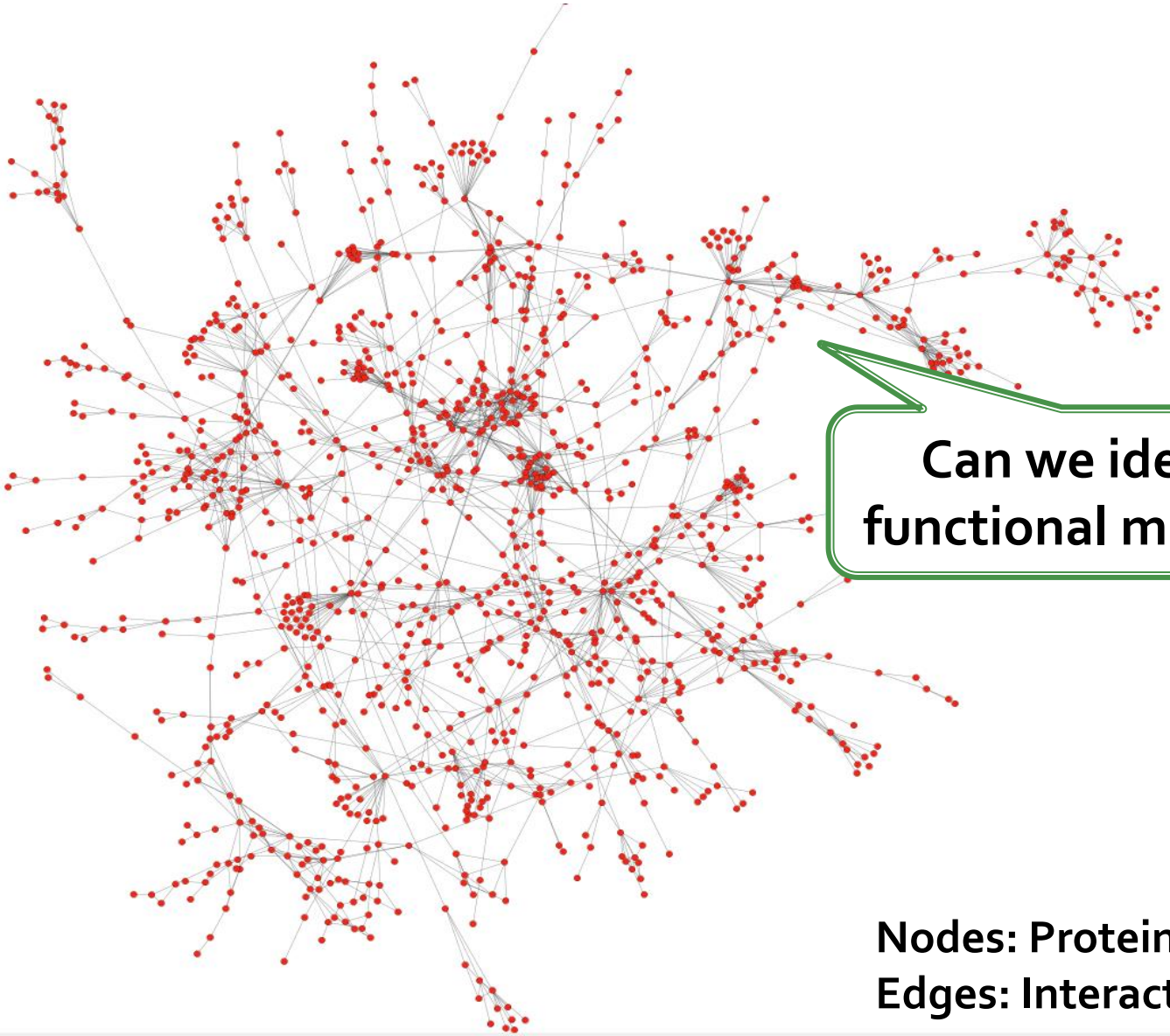
Facebook Ego-network



Social communities

Nodes: Users
Edges: Friendships

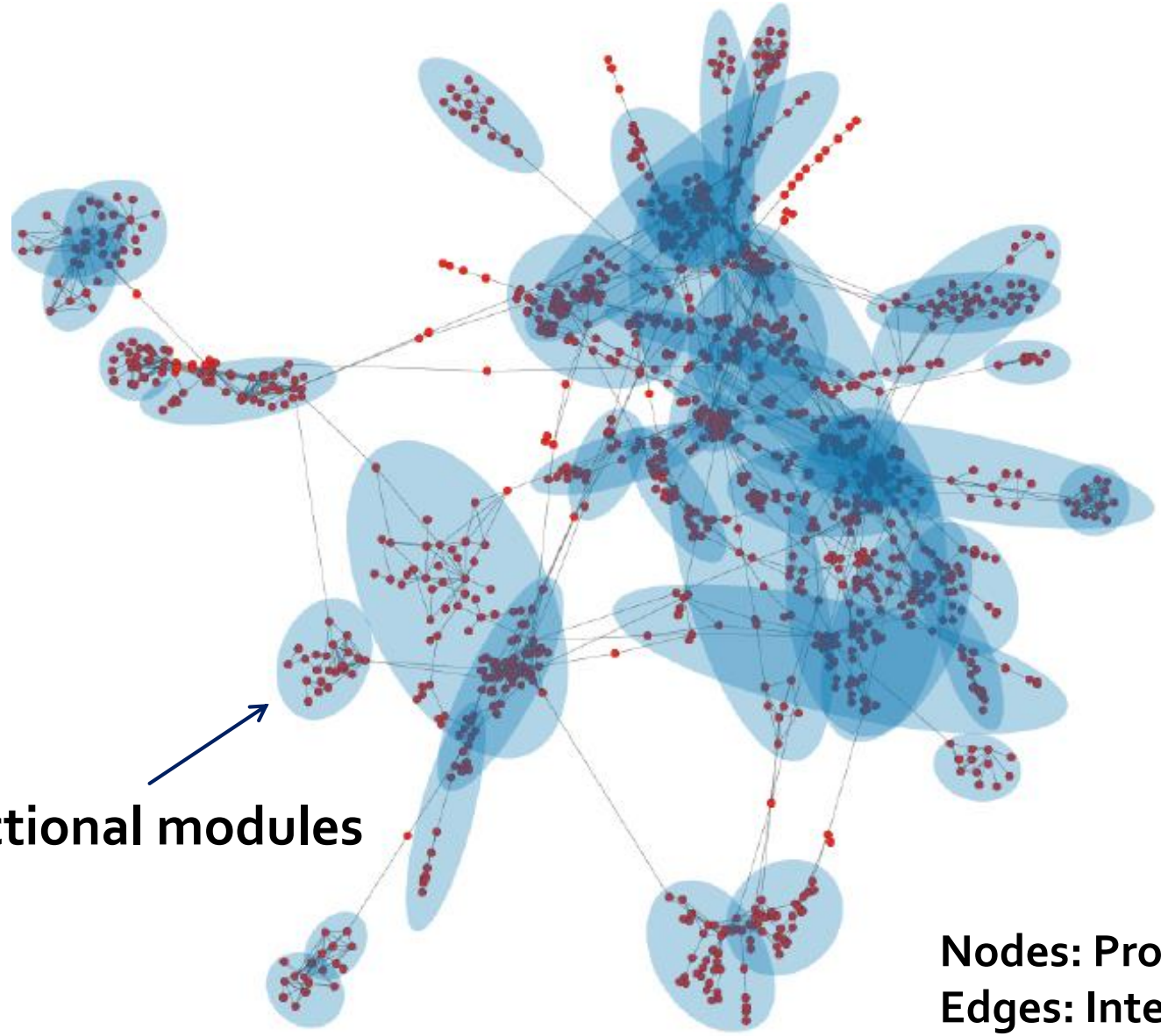
Protein-Protein Interactions



Can we identify functional modules?

Nodes: Proteins
Edges: Interactions

Protein-Protein Interactions

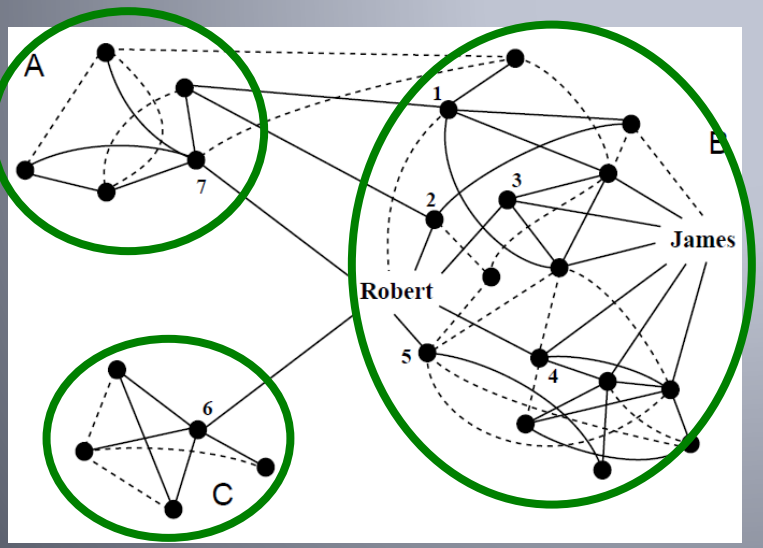


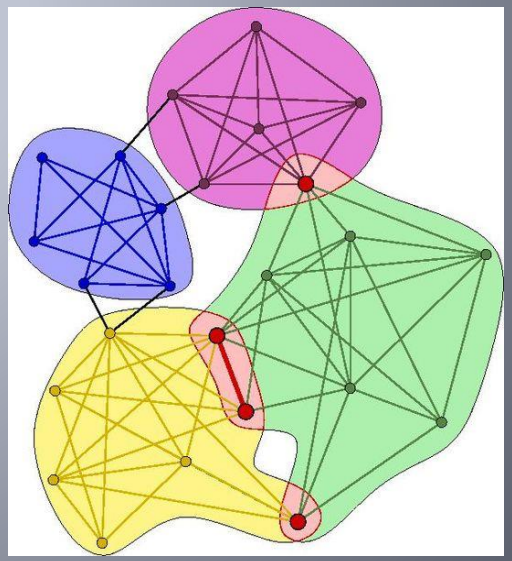
Functional modules

Nodes: Proteins
Edges: Interactions

Community Detection

How to find communities?

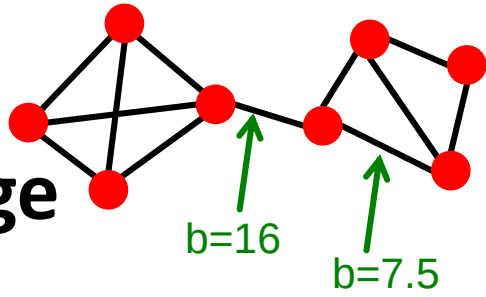


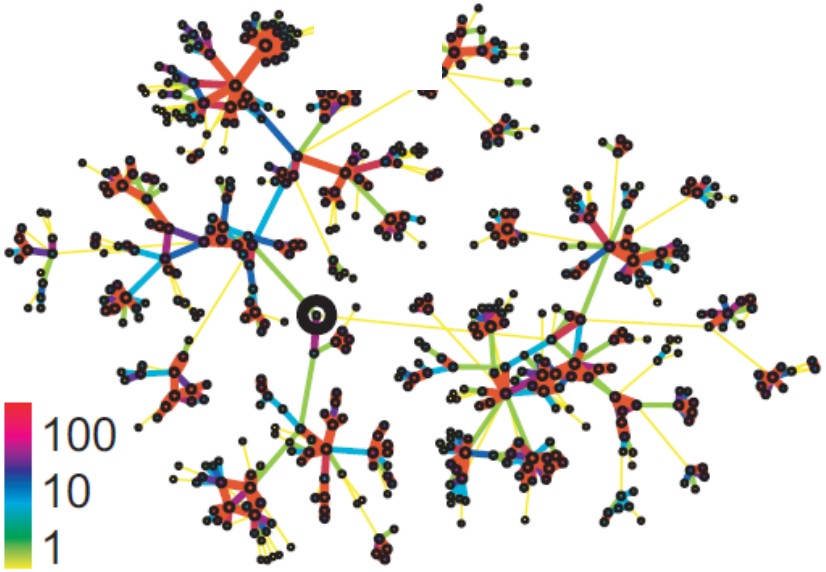


We will work with **undirected** (unweighted) networks

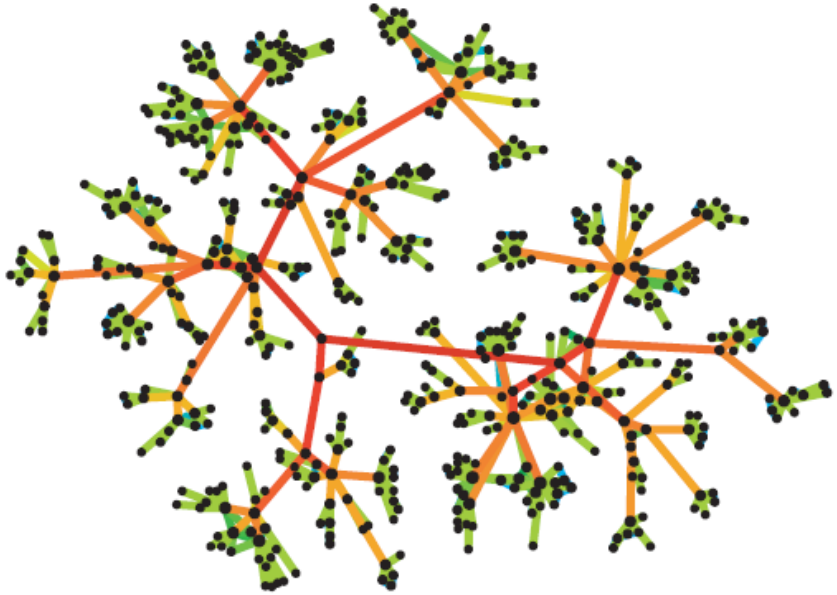
Method 1: Strength of Weak Ties

- **Edge betweenness:** Number of shortest paths passing over the edge
- **Intuition:**





Edge strengths (call volume) in a real network



Edge betweenness in a real network

[Girvan-Newman '02]

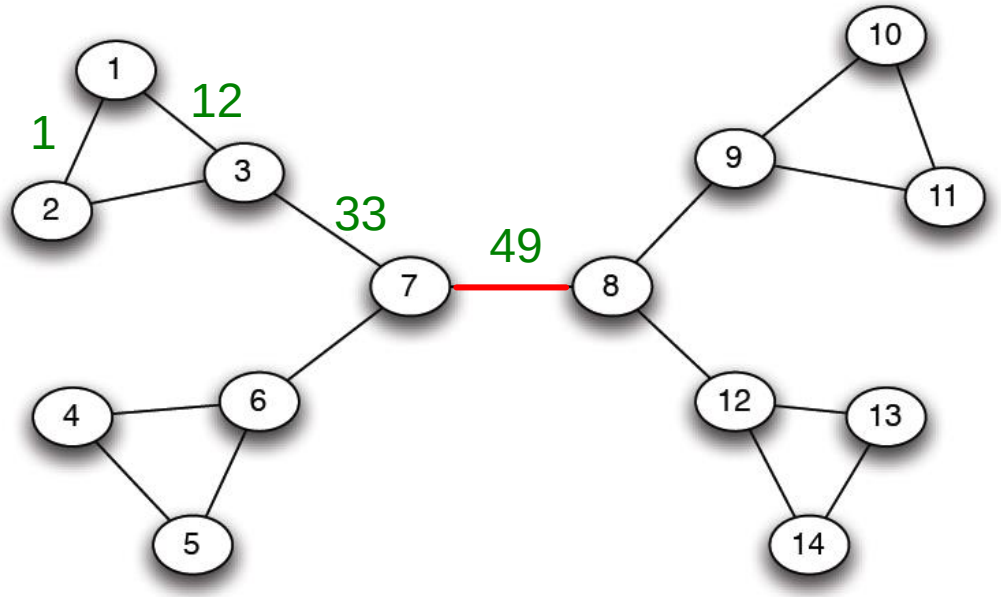
Method 1: Girvan-Newman

- Divisive hierarchical clustering based on the notion of edge **betweenness**:

  Number of shortest paths passing through the edge

- **Girvan-Newman Algorithm:**
  - **Undirected unweighted networks**
  - **Repeat until no edges are left:**
    - Calculate betweenness of edges
    - Remove edges with highest betweenness
  - Connected components are communities
  - Gives a hierarchical decomposition of the network

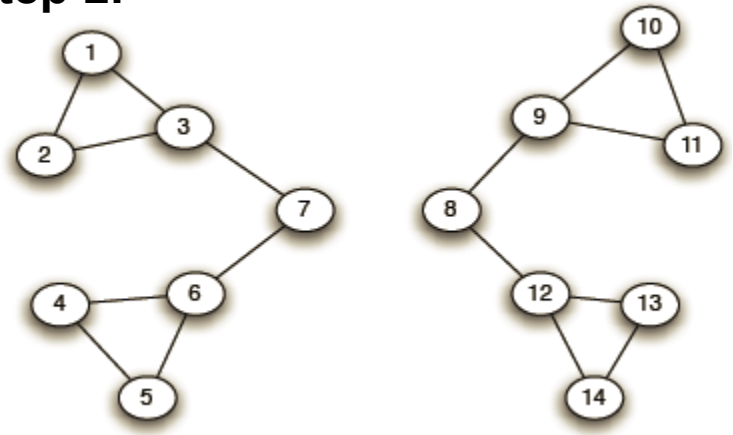
Girvan-Newman: Example



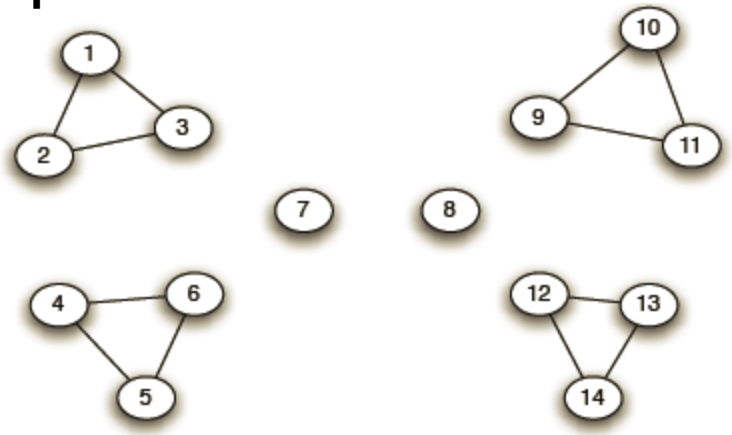
Need to re-compute betweenness at every step

Girvan-Newman: Example

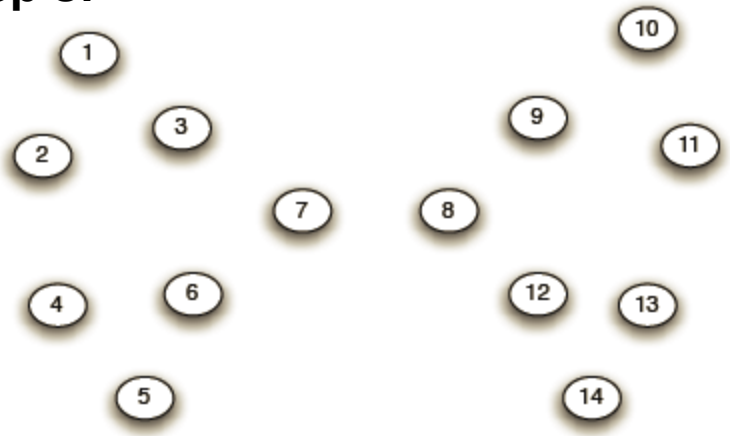
Step 1:



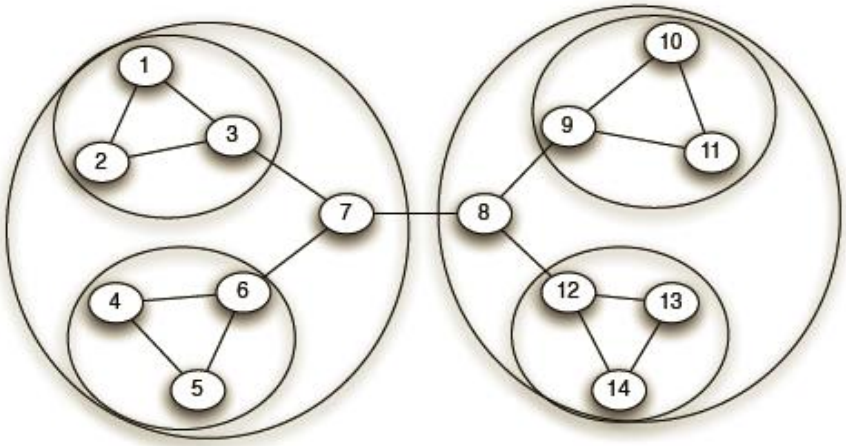
Step 2:



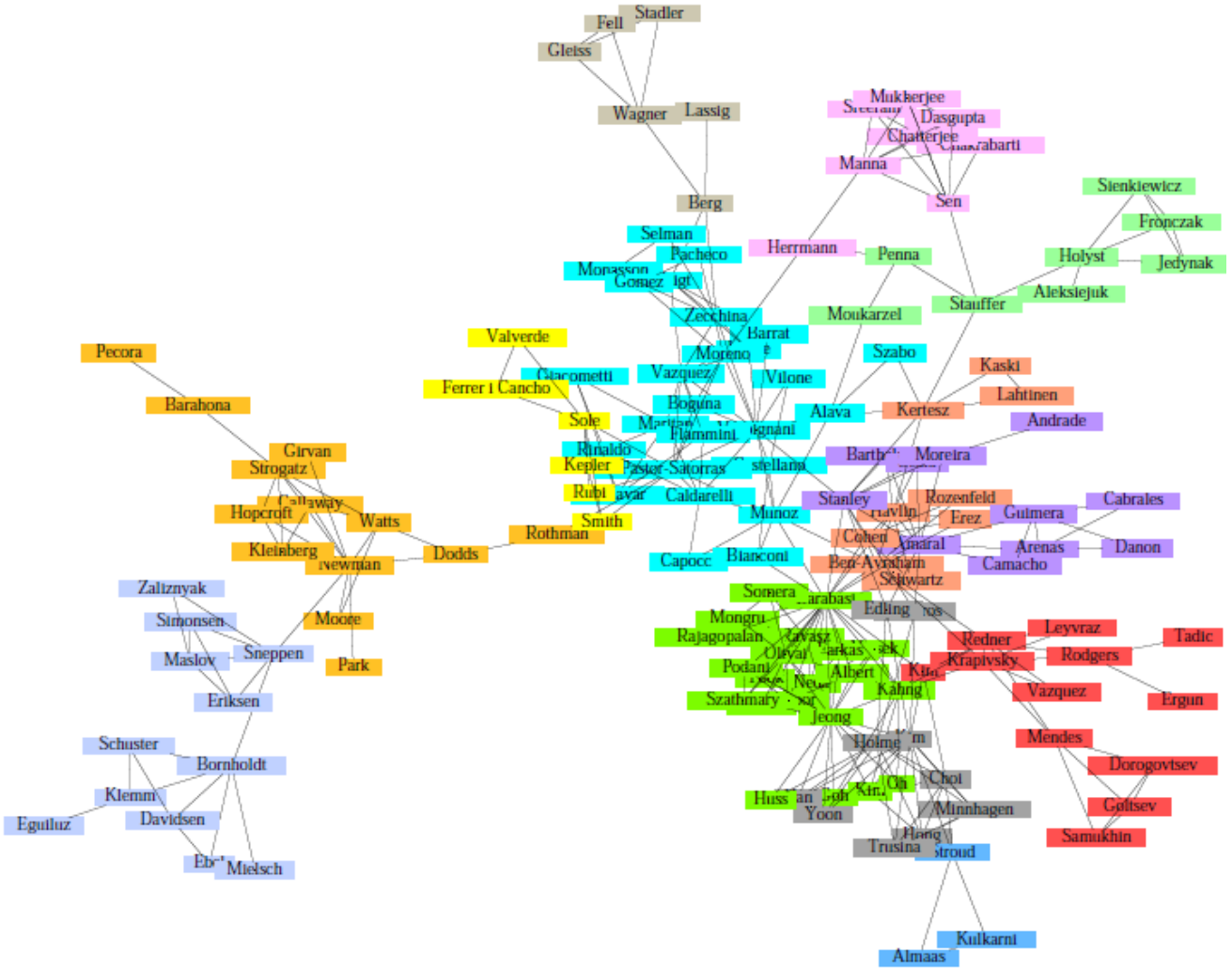
Step 3:



Hierarchical network decomposition:



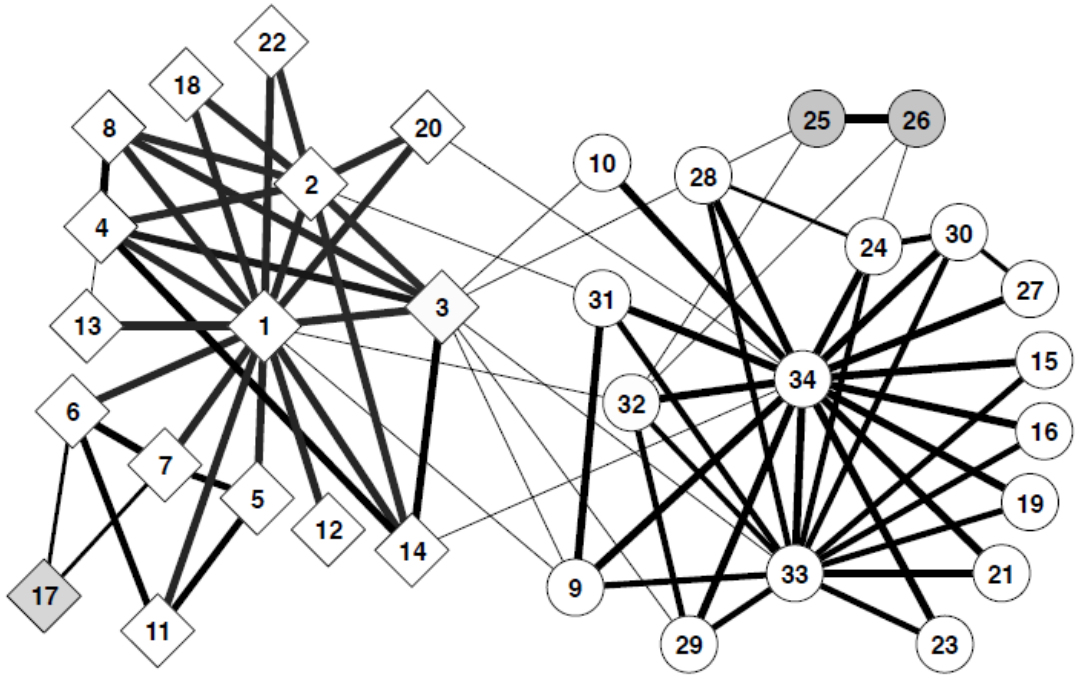
Girvan-Newman: Results

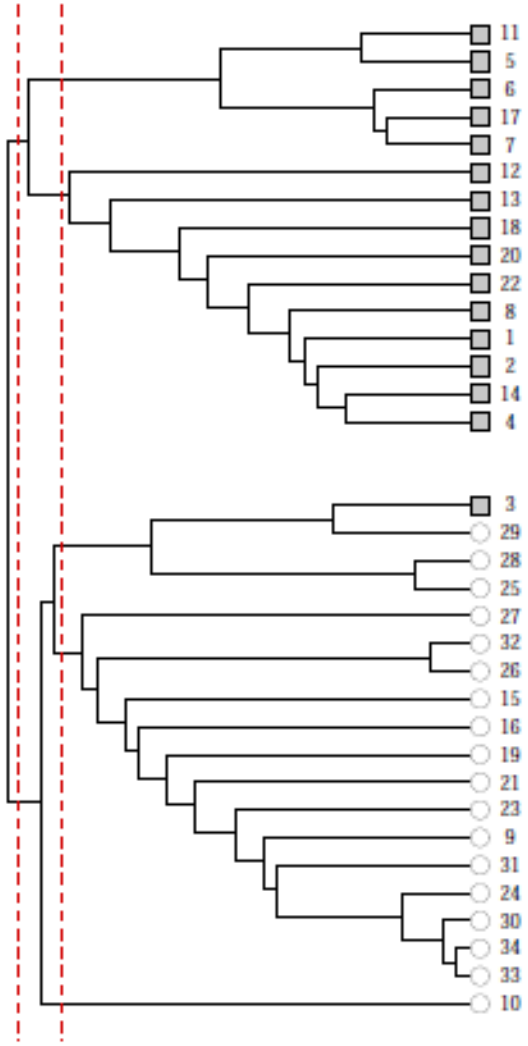


Communities in physics collaborations

Girvan-Newman: Results

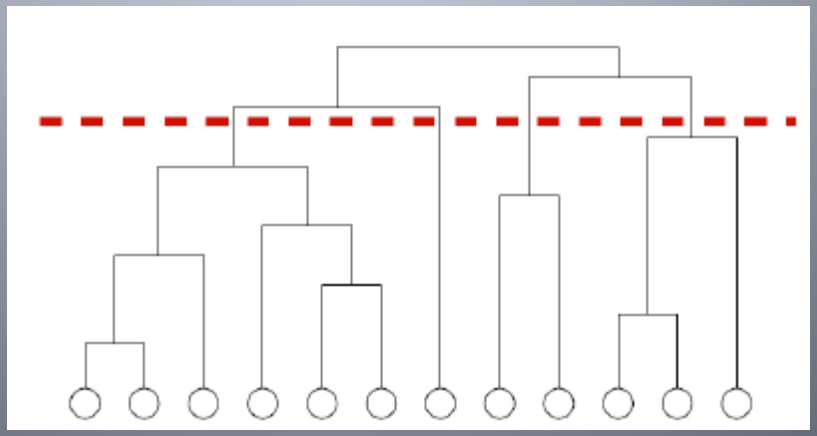
- **Zachary's Karate club:**
  Hierarchical decomposition





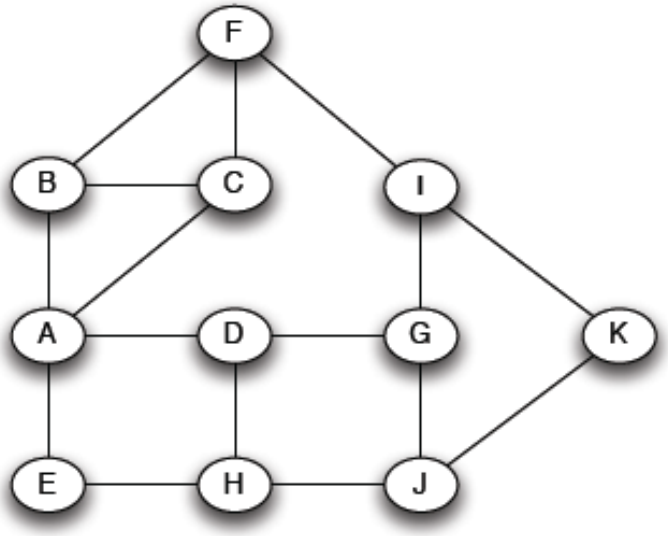
We need to resolve 2 questions

1. How to compute betweenness?
2. How to select the number of clusters?

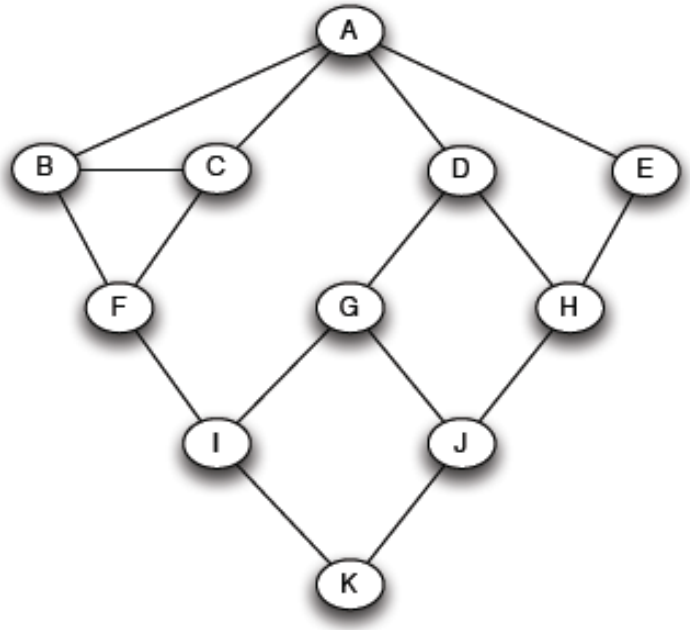


How to Compute Betweenness?

- Want to compute betweenness of paths starting at node A

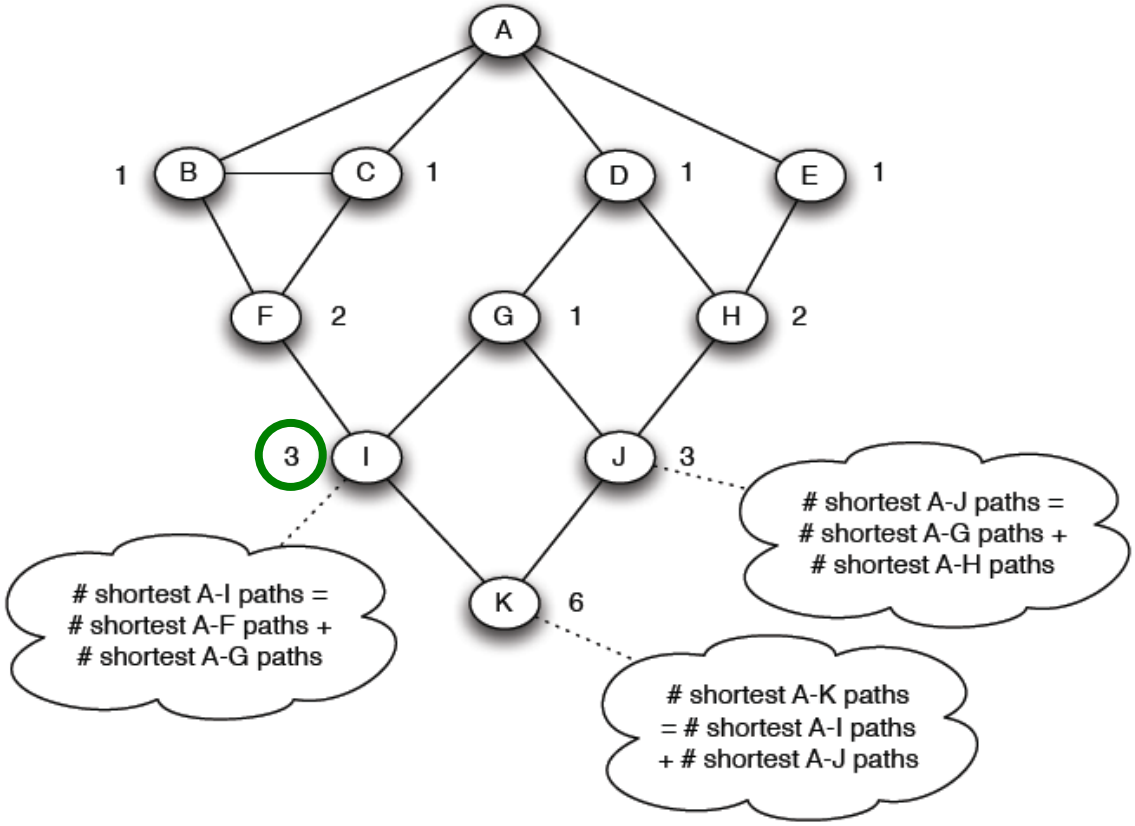


- Breadth first search starting from A:



How to Compute Betweenness?

- Count the number of shortest paths from A to all other nodes of the network:

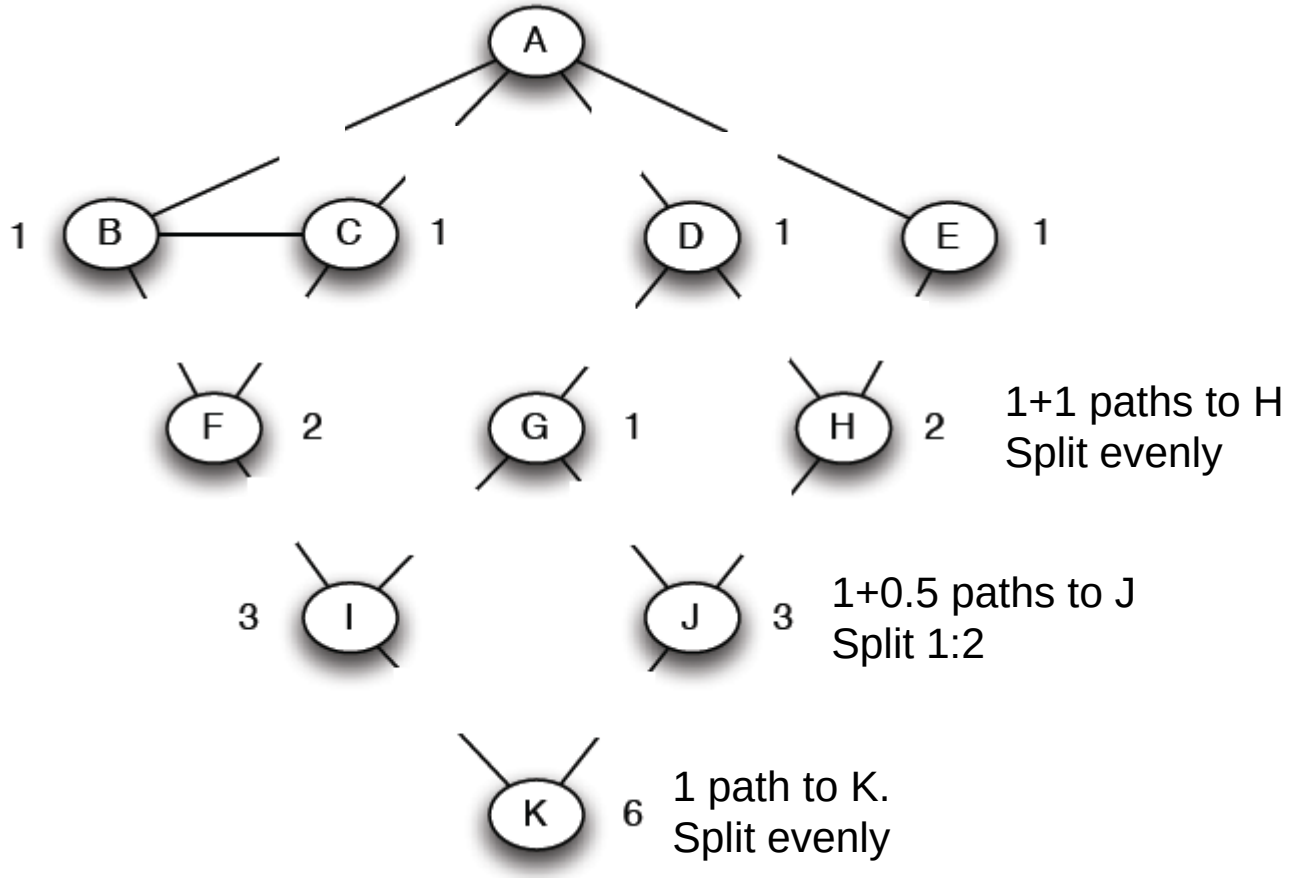


How to Compute Betweenness?

- **Compute betweenness by working up the tree:** If there are multiple paths count them fractionally

The algorithm:

- Add edge **flows**:
  - -- node flow = 1+∑child edges
  - -- split the flow up based on the parent value
- Repeat the BFS procedure for each starting node U



How to Compute Betweenness?

- **Compute betweenness by working up the tree:** If there are multiple paths count them fractionally

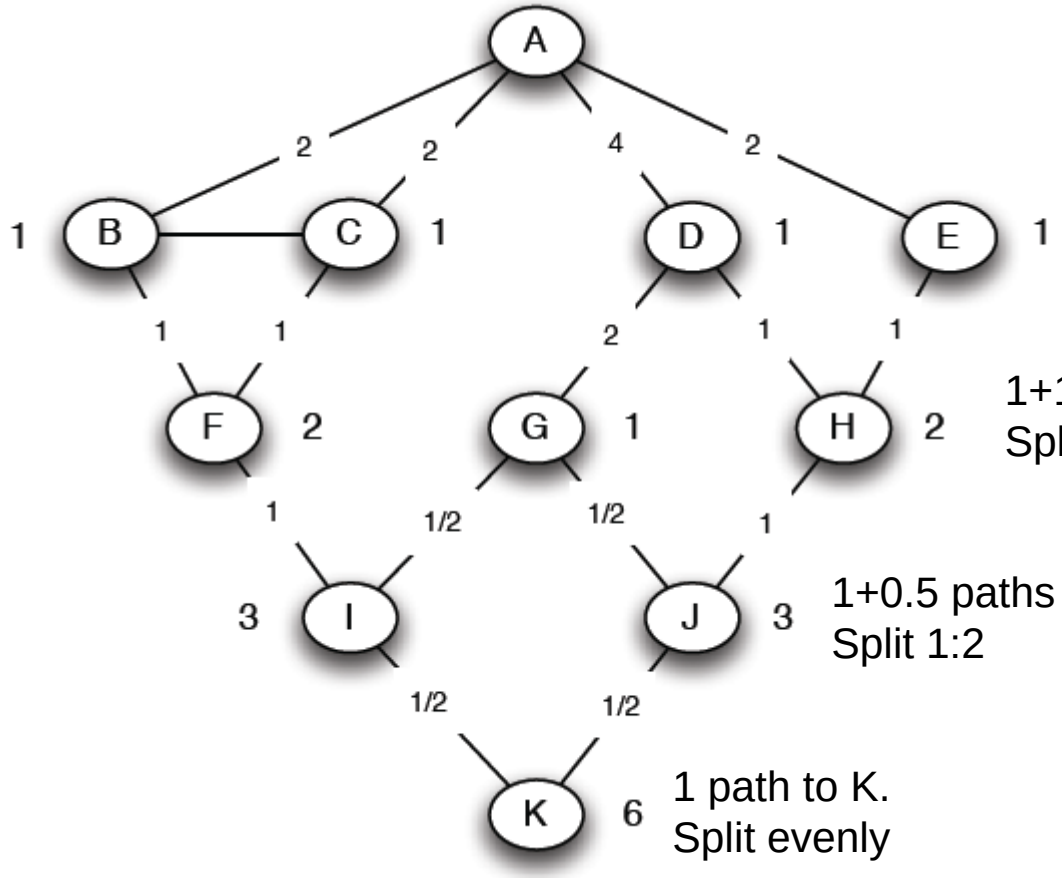
The algorithm:

•Add edge **flows**:

-- node flow = 1+∑child edges

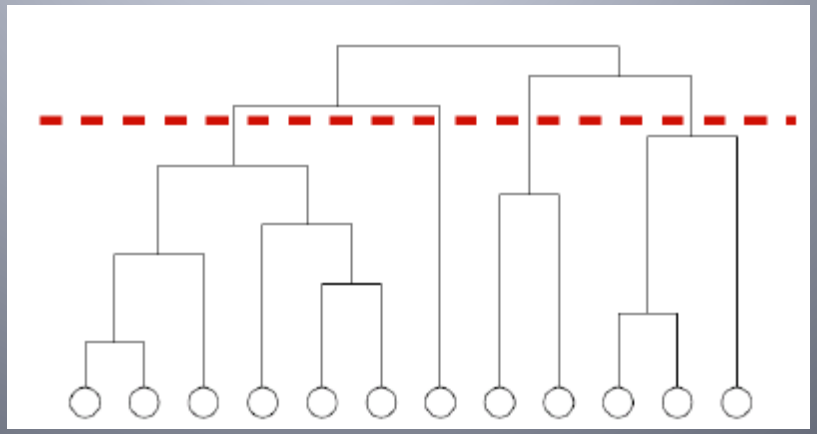
-- split the flow up based on the parent value

• Repeat the BFS procedure for each starting node U



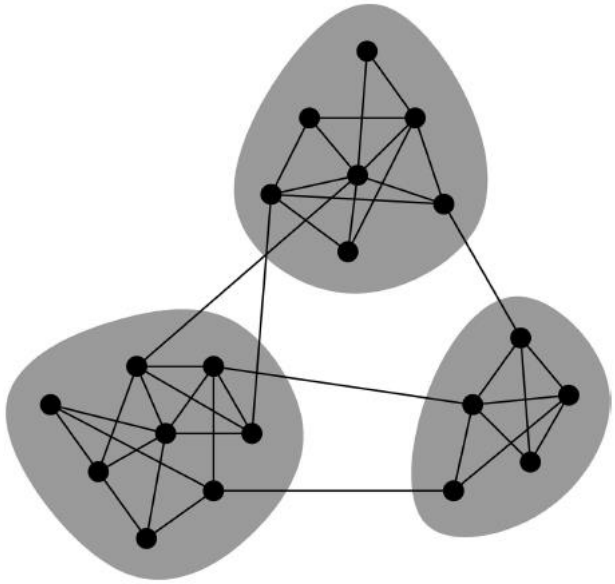
We need to resolve 2 questions

1. How to compute betweenness?
2. How to select the number of clusters?



Network Communities

- **Communities:** sets of **tightly connected nodes**
- Define: **Modularity Q**
  - A measure of how well a network is partitioned into communities
  - Given a partitioning of the network into groups $s \in S$:

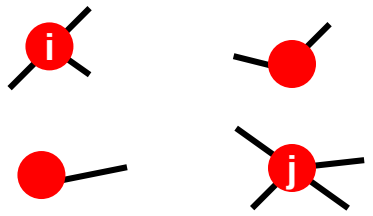


$$Q \propto \sum_{s \in S} [\ (\#\ \text{edges within group } s) - (\text{expected } \#\ \text{edges within group } s)\ ]$$

Need a null model!

Null Model: Configuration Model

- **Given real G on n nodes and m edges, construct rewired network G'**
  - Same degree distribution but random connections
  - Consider G' as a multigraph
  - **The expected number of edges between nodes i and j** of degrees k_i and k_j equals to: $k_i \cdot \frac{k_j}{2m} = \frac{k_i k_j}{2m}$
    - The expected number of edges in (multigraph) **G'**:
      - $= \frac{1}{2}\sum_{i \in N}\sum_{j \in N}\frac{k_i k_j}{2m} = \frac{1}{2}\cdot\frac{1}{2m}\sum_{i \in N} k_i\left(\sum_{j \in N} k_j\right) =$
      - $= \frac{1}{4m} 2m \cdot 2m = m$



Note: $\sum_{u \in N} k_u = 2m$

Modularity

- **Modularity of partitioning S of graph G:**
  - $\mathbf{Q} \propto \sum_{s \in S}$ **[(# edges within group** s**) –**
    (expected # edges within group s**)]**
  - $Q(G,S) = \underbrace{\frac{1}{2m}} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} \left(A_{ij} - \frac{k_i k_j}{2m} \right)$

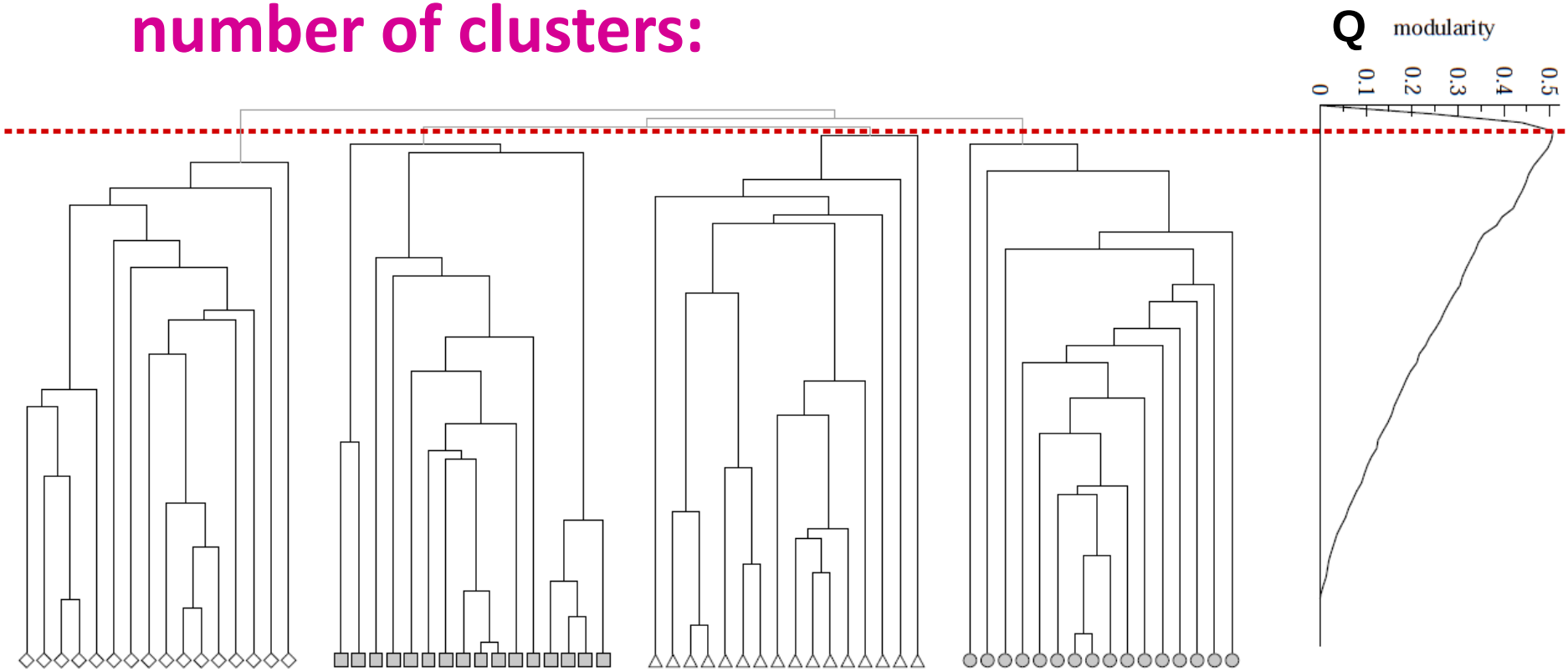
    Normalizing cost.: -1<Q<1

    A_{ij} = 1 if i→j, 0 else

- **Modularity values take range [−1,1]**
  - It is positive if the number of edges within groups exceeds the expected number
  - **0.3-0.7<Q** means significant community structure

Modularity: Number of clusters

- **Modularity is useful for selecting the number of clusters:**



Why not optimize Modularity directly?

Modularity Optimization

Method 2: Modularity Optimization

- **Let's split the graph into 2 communities!**
- **Want to directly optimize modularity!**
  - $\max_S Q(G,S) = \frac{1}{2m}\sum_{s\in S}\sum_{i\in s}\sum_{j\in s}\left(A_{ij} - \frac{k_i k_j}{2m}\right)$
- **Community membership vector s:**
  - s_i **= 1** if node ***i*** is in community **1**
    -1 if node ***i*** is in community **-1**

$\frac{s_i s_j + 1}{2} =$ 1.. if $s_i = s_j$, 0.. else

- $Q(G,s) = \frac{1}{2m}\sum_{i\in N}\sum_{j\in N}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\frac{(s_i s_j + 1)}{2}$

$$= \frac{1}{4m}\sum_{i,j\in N}\left(A_{ij} - \frac{k_i k_j}{2m}\right)s_i s_j$$

Modularity Matrix

Note: each row/col of **B** sums to **0:** $\sum_j A_{ij} = k_i$, $\sum_j \frac{k_i k_j}{2m} = k_i \sum_j \frac{k_j}{2m} = k_i$

- **Define:**
  - **Modularity matrix:** $B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$
  - **Membership:** $s = \{-1, +1\}$
- **Then:** $Q(G,s) = \frac{1}{4m} \sum_{i \in N} \sum_{j \in N} \left(A_{ij} - \frac{k_i k_j}{2m} \right) s_i s_j$

$$= \frac{1}{4m} \sum_{i,j \in N} B_{ij} s_i s_j$$

$$= \frac{1}{4m} \sum_i s_i \underbrace{\sum_j B_{ij} s_j}_{= B_{i\cdot} \cdot s} = \frac{1}{4m} s^T B s$$

- **Task:** Find $s \in \{-1,+1\}^n$ that maximizes **Q(G,s)**

Quick Review of Linear Algebra

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

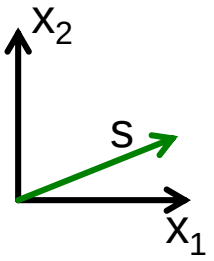
- **Symmetric** matrix **A**
  - That is positive semi-definite: $\boldsymbol{A = U \cdot U^T}$
- Then solutions $\boldsymbol{\lambda, x}$ to equation $\boldsymbol{A \cdot x = \lambda \cdot x}$:
  - **Eigenvectors** $\boldsymbol{x_i}$ ordered by the magnitude of their corresponding **eigenvalues** λ_i ($\lambda_1 \leq \lambda_2 \ldots \leq \lambda_n$)
  - $\boldsymbol{x_i}$ are **orthonormal** (orthogonal and unit length)
  - $\boldsymbol{x_i}$ form a coordinate system (basis)
  - If $\boldsymbol{A}$ is positive-semidefinite: $\lambda_i \geq 0$ (and they always exist)
- **Eigen Decomposition theorem:** Can rewrite matrix $\boldsymbol{A}$ in terms of its eigenvectors and eigenvalues: $\boldsymbol{A = \sum_i x_i \cdot \lambda_i \cdot x_i^T}$

Modularity Optimization

- **Rewrite:** $Q(G,s) = \frac{1}{4m} s^{\mathrm{T}} B s$ **in terms of its eigenvectors and eigenvalues:**

$$= s^{\mathrm{T}} \left[\sum_{i=1}^{n} x_i \lambda_i x_i^T \right] s = \sum_{i=1}^{n} s^T x_i \lambda_i x_i^T s = \sum_{i=1}^{n} (s^T x_i)^2 \lambda_i$$

- **So, if there would be no other constraints on s then to maximize Q, we make $s = x_n$**
  - **Why?** Because $\lambda_n \geq \lambda_{n-1} \geq \cdots$
    - Remember s has fixed length!
    - Assigns all weight in the sum to λ_n (largest eigenvalue)
      - All other $s^T x_i$ terms are **zero** because of orthonormality



Finding the vector *s*

- **Let's consider only the first term in the summation (because λ_n is the largest):**

$$\max_s Q(G,s) = \sum_{i=1}^{n} (s^T x_i)^2 \lambda_i \approx (s^T x_n)^2 \lambda_n$$

- **Let's maximize:** $\sum_{j=1}^{n} s_j \cdot x_{n,j}$ where $s_j \in \{-1,+1\}$
- To do this, we set:
  - $s_j = \begin{cases} +1 & if\ x_{n,j} \geq 0 \quad (\text{j−th coordinate of } x_n \geq 0) \\ -1 & if\ x_{n,j} < 0 \quad (\text{j−th coordinate of } x_n < 0) \end{cases}$
- **Continue the bisection hierarchically**

Summary: Modularity Optimization

- **Fast Modularity Optimization Algorithm:**
  - Find leading eigenvector x_n of modularity matrix **B**
  - Divide the nodes by the signs of the elements of x_n
  - Repeat hierarchically until:
    - If a proposed split does not cause modularity to increase, declare community indivisible and do not split it
    - If all communities are indivisible, stop
- **How to find x_n? Power method!**
  - Start with random $v^{(0)}$, repeat :
  - When converged ($v^{(t)} \approx v^{(t+1)}$), set $x_n = v^{(t)}$

$$v^{(t+1)} = \frac{Bv^{(t)}}{\left\| Bv^{(t)} \right\|}$$

Summary: Modularity

- **Girvan-Newman:**
  - Based on the "strength of weak ties"
  - Remove edge of highest betweenness
- **Modularity:**
  - Overall quality of the partitioning of a graph
  - Use to determine the number of communities
- **Fast modularity optimization:**
  - Transform the modularity optimization to a eigenvalue problem

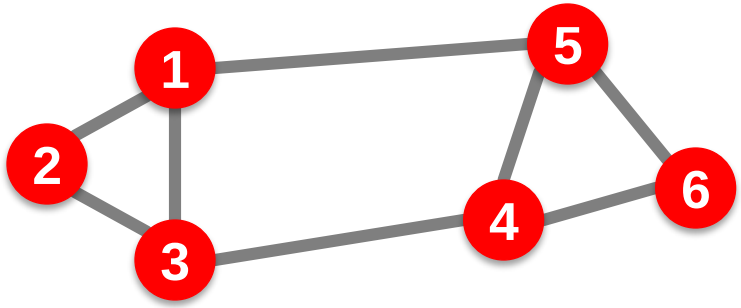
Community Detection: Graph Cuts & Spectral Clustering

Agenda

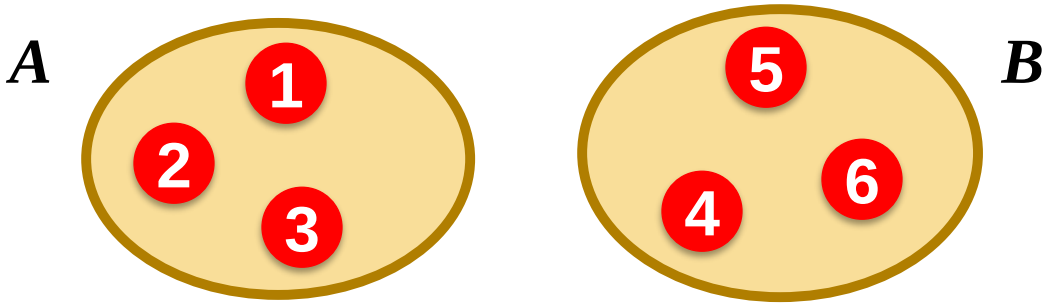
- Graph Partitioning
  - Graph Cuts
  - Spectral Clustering

Graph Partitioning

- **Undirected graph $G(V, E)$:**



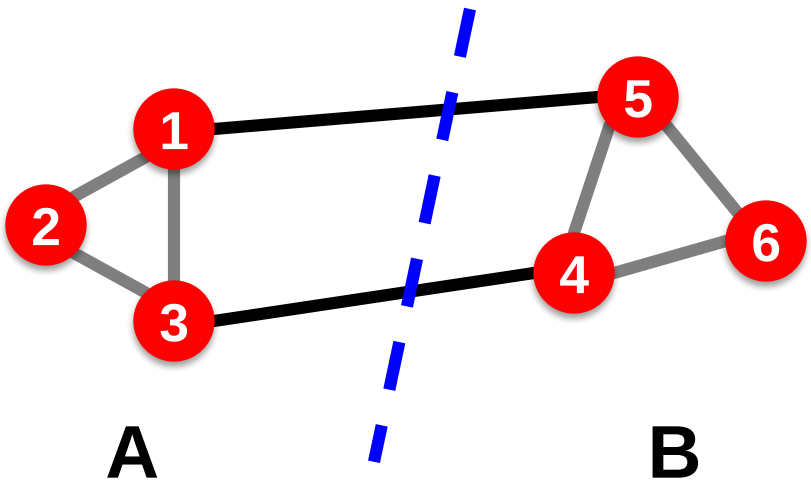
- **Bi-partitioning task:**
  - Divide vertices into two disjoint groups A, B



- **Questions:**
  - How can we define a "good" partition of G?
  - How can we efficiently identify such a partition?

Graph Partitioning

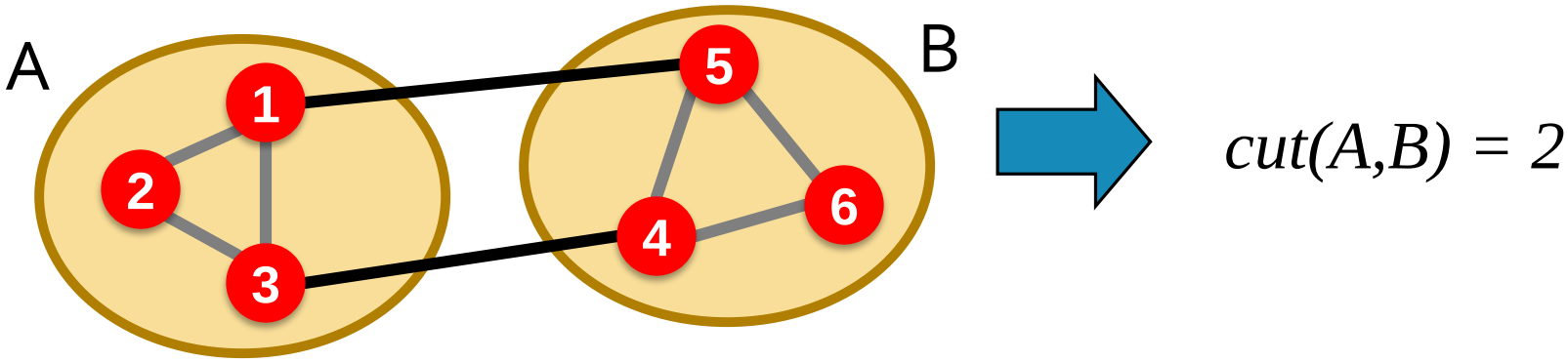
- **What makes a good partition?**
  - Maximize the number of within-group connections
  - Minimize the number of between-group connections



Graph Cuts

- **Express partitioning objectives as a function of the "edge cut" of the partition**
- **Cut:** Set of edges with only one vertex in a group:

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

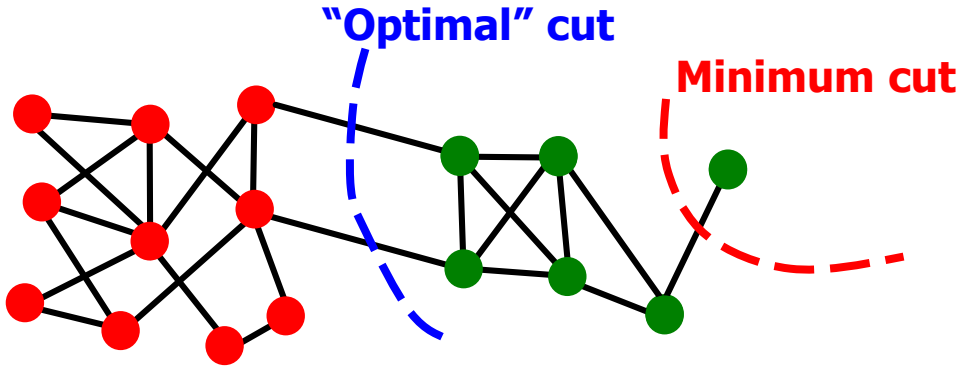


Graph Cut Criterion

- **Criterion: Minimum-cut**
  - Minimize weight of connections between groups

$$\arg\min_{A,B} cut(A,B)$$

- **Degenerate case:**



- **Problem:**
  - Only considers external cluster connections
  - Does not consider internal cluster connectivity

Graph Bisection

- Since the minimum cut does not always yield good results we need extra constraints to make the problem meaningful
- **Graph Bisection**
  - Partition the graph into two *equal sets* of nodes
- **Kernighan-Lin algorithm**
  - Start with random equal partitions
  - Swap nodes to improve some quality metric (e.g., cut, modularity, etc)

Ratio Cut

Criterion: Ratio-cut

Normalize cut by the *size* of the groups

$$\textbf{Ratio-cut} = \frac{\mathrm{Cut(U,V-U)}}{|U|} + \frac{\mathrm{Cut(U,V-U)}}{|V-U|}$$

Normalized Cut

Criterion: Normalized-cut

Connectivity between groups relative to the *density* of each group

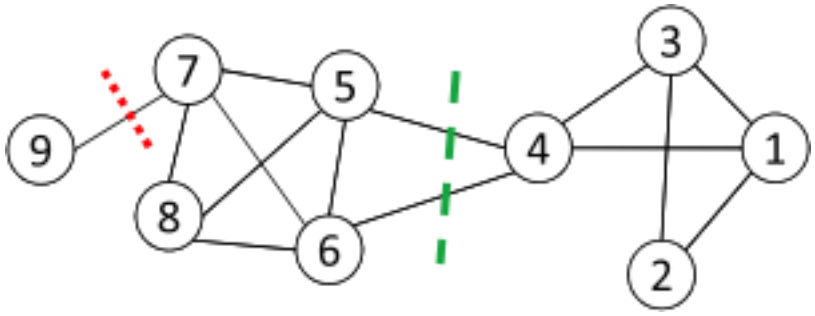
$$\textbf{Normalized-cut} = \frac{\text{Cut(U,V}-\text{U)}}{Vol(U)} + \frac{\text{Cut(U,V}-\text{U)}}{Vol(V-U)}$$

$vol(U)$: total weight of the edges with at least one endpoint in U: $vol(U) = \sum_{i \in U} d_i$

Why use these criteria?

- Produce more balanced partitions

An Example



Red is Min-Cut

$$\text{Ratio-Cut(Red)} = \frac{1}{1} + \frac{1}{8} = \frac{9}{8}$$

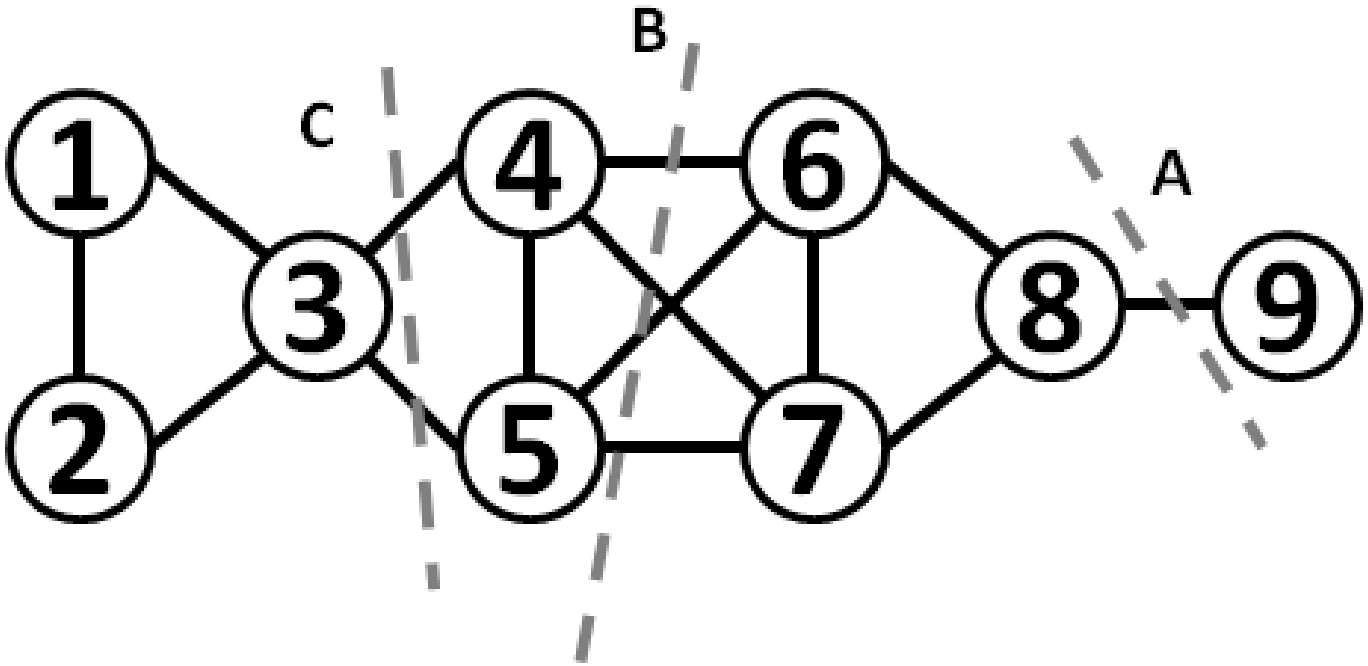
$$\text{Ratio-Cut(Green)} = \frac{2}{5} + \frac{2}{4} = \frac{18}{20}$$

$$\text{Normalized-Cut(Red)} = \frac{1}{1} + \frac{1}{27} = \frac{28}{27}$$

$$\text{Normalized-Cut(Green)} = \frac{2}{12} + \frac{2}{16} = \frac{14}{48}$$

Minimizing **Normalized-cut** is even better for Green due to density

Another Example



Which of the three cuts has the best (min, normalized, ratio) cut?

[Shi-Malik]

Graph Cut Criteria

- **Criterion: Conductance** [Shi-Malik, '97]
  - Connectivity between groups relative to the density of each group

$$\phi(A,B) = \frac{cut(A,B)}{\min(vol(A), vol(B))}$$

$\boldsymbol{vol(A)}$: total weight of the edges with at least one endpoint in $\boldsymbol{A}$: $\boldsymbol{vol(A) = \sum_{i \in A} k_i}$

- **Why use this criterion?**
  - Produces more balanced partitions
- **How do we efficiently find a good partition?**
  - **Problem:** Computing optimal cut is NP-hard

Graph Cuts

- Ratio-cut and normalized-cut can be reformulated in matrix format and solved using spectral clustering

Spectral Clustering for Graph Partitioning

Spectral Clustering Algorithms

- **Three basic stages:**
  - **1) Pre-processing**
    - Construct a matrix representation of the graph
  - **2) Decomposition**
    - Compute eigenvalues and eigenvectors of the matrix
    - Map each point to a lower-dimensional representation based on one or more eigenvectors
  - **3) Grouping**
    - Assign points to two or more clusters, based on the new representation
- But first, let's define the problem

Spectral Graph Partitioning

- A: adjacency matrix of undirected **G**
  - A_{ij} **=1** if (i, j) is an edge, else **0**
- x is a vector in $\Re^n$ with components $(x_1, \ldots, x_n)$
  - Think of it as a label/value of each node of G
- **What is the meaning of $A \cdot x$?**

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$y_i = \sum_{j=1}^{n} A_{ij} x_j = \sum_{(i,j) \in E} x_j$$

- **Entry y_i is a sum of labels x_j of neighbors of i**

Spectral Graph Theory

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

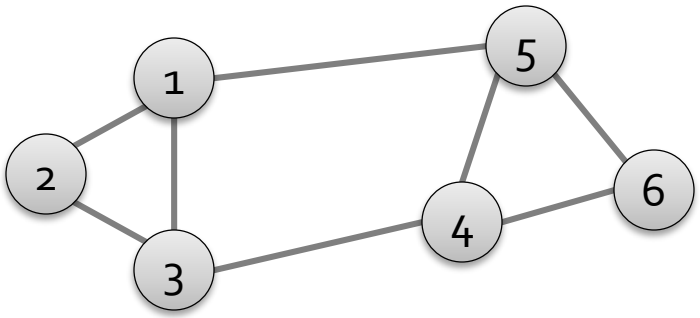
$$\boldsymbol{A \cdot x = \lambda \cdot x}$$

- **Spectral Graph Theory:**
  - Analyze the "spectrum" of matrix representing $\boldsymbol{G}$
  - **Spectrum:** Eigenvectors $\boldsymbol{x_i}$ of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues $\boldsymbol{\lambda_i}$: $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}\ \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

    Note: We sort $\boldsymbol{\lambda_i}$ in ascending (not descending) order!
  - **Spectral clustering**: use the eigenvectors of $\boldsymbol{A}$ or graphs derived by it (mostly **graph Laplacian**)

Matrix Representations

- **Adjacency matrix (A):**
  - $n \times n$ matrix
  - $A=[a_{ij}]$, $a_{ij}=1$ if edge between node i and j

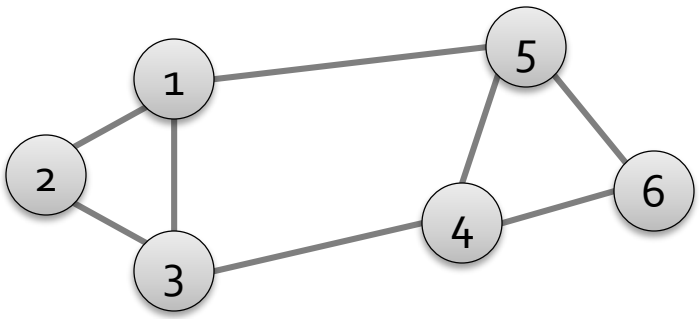


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

- **Important properties:**
  - Symmetric matrix
  - Eigenvectors are real and orthogonal

Matrix Representations

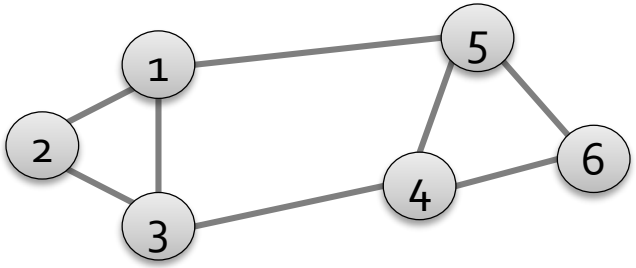
- **Degree matrix (D):**
  - $n \times n$ diagonal matrix
  - $D=[d_{ii}]$, d_{ii} = degree of node i



	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

Matrix Representations

- **Laplacian matrix (L):**
  - $n \times n$ symmetric matrix

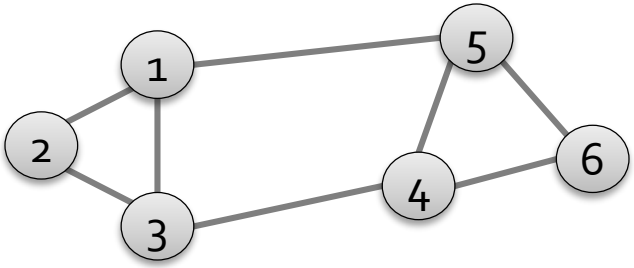


	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

$$L = D - A$$

- **Laplacian matrix L important properties:**
  - **Eigenvalues** are non-negative real numbers
  - **Eigenvectors** are real and orthogonal

Example: Eigenvalues & Eigenvectors



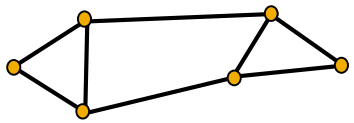
Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	−5	−1	−1	−1
	1	2	4	−2	1	0
	1	1	1	3	−1	1
	1	−1	−5	−1	1	1
	1	−2	4	−2	−1	0
	1	−1	1	3	1	−1

Spectral Clustering Algorithms

- **Three basic stages:**
  - **1) Pre-processing**
    - Construct a matrix representation of the graph
  - **2) Decomposition**
    - Compute eigenvalues and eigenvectors of the matrix
    - Map each point to a lower-dimensional representation based on one or more eigenvectors
  - **3) Grouping**
    - Assign points to two or more clusters, based on the new representation

Spectral Partitioning Algorithm

- **1) Pre-processing:**
  - Build Laplacian matrix L of the graph



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- **2) Decomposition:**
  - Find eigenvalues λ and eigenvectors x of the matrix L

$\lambda =$

0.0
1.0
3.0
3.0
4.0
5.0

$X =$

0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

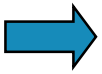
  - Map vertices to corresponding components of λ_2

How do we now find the clusters?

Spectral Partitioning

- **3) Grouping:**
  - Sort components of reduced 1-dimensional vector
  - Identify clusters by splitting the sorted vector in two
- **How to choose a splitting point?**
  - Naïve approaches:
    - Split at **0** or median value
  - More expensive approaches:
    - Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)

1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6



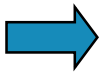
Split at 0:

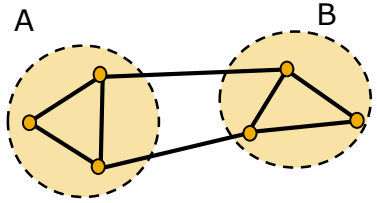
Cluster A: Positive points

Cluster B: Negative points

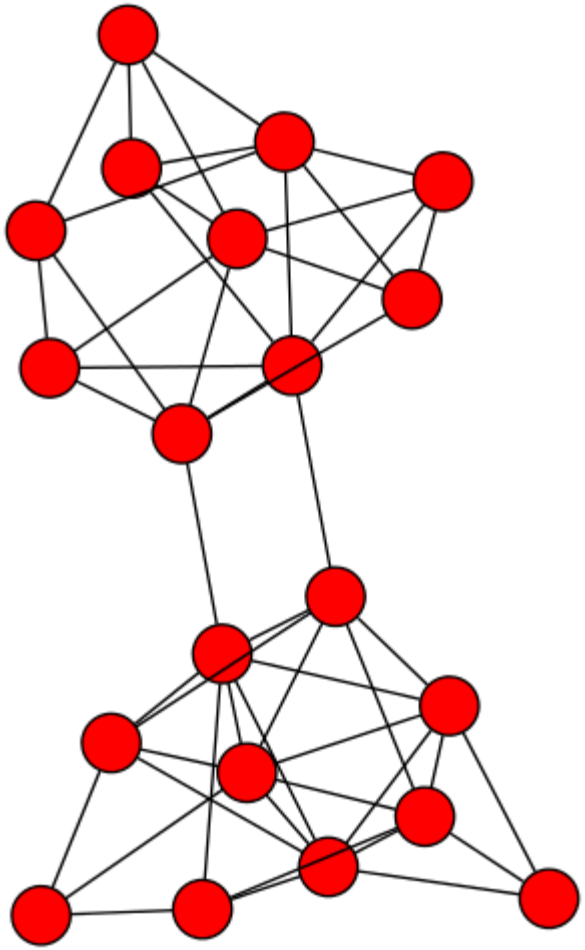
1	0.3
2	0.6
3	0.3

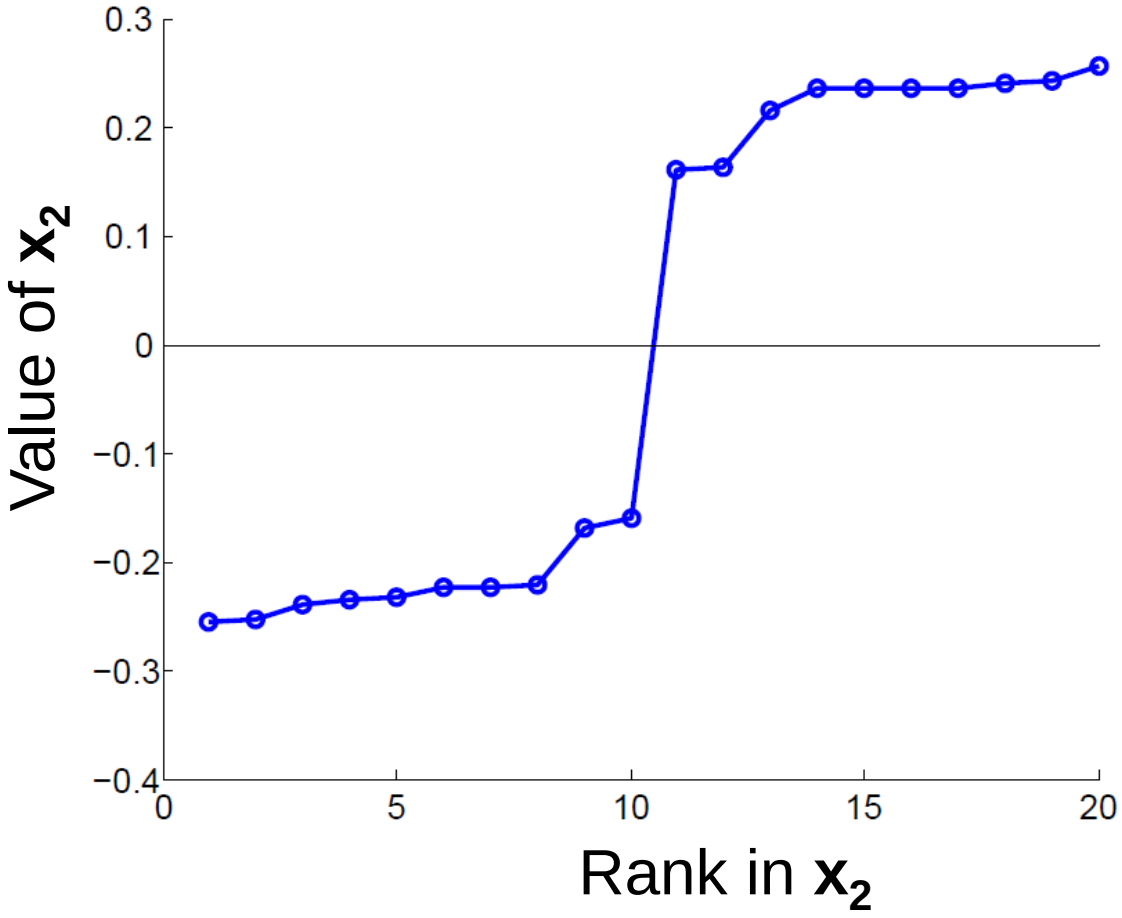
4	-0.3
5	-0.3
6	-0.6



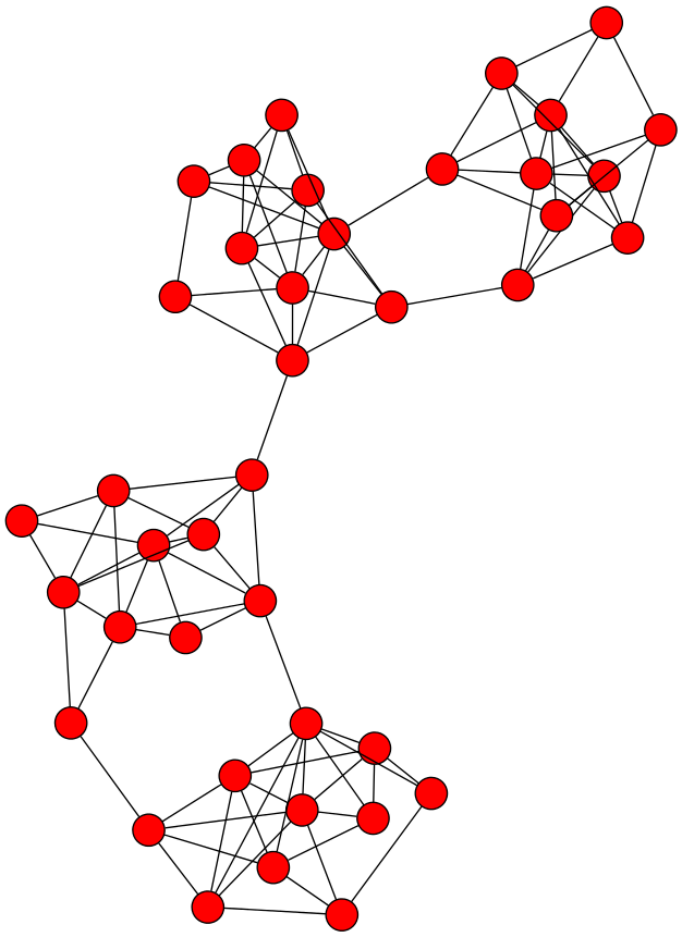


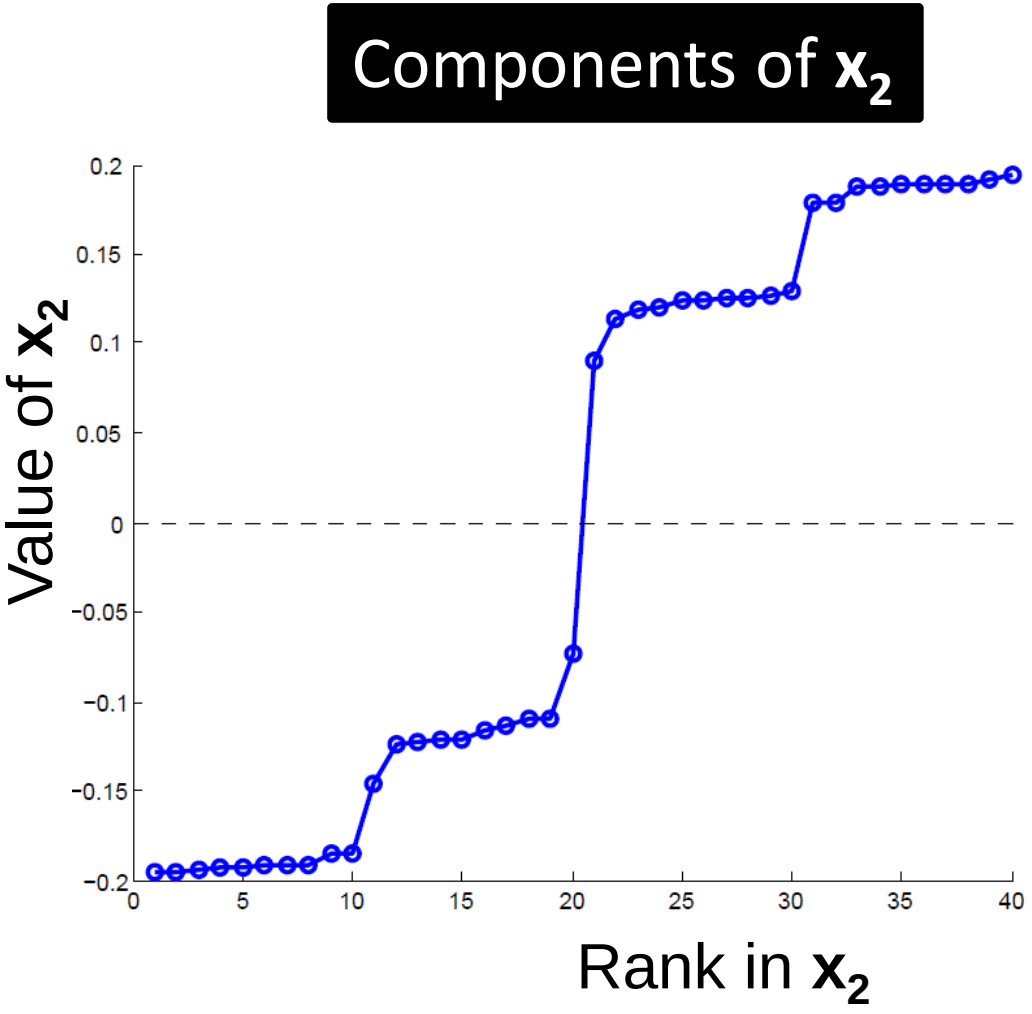
Example: Spectral Partitioning



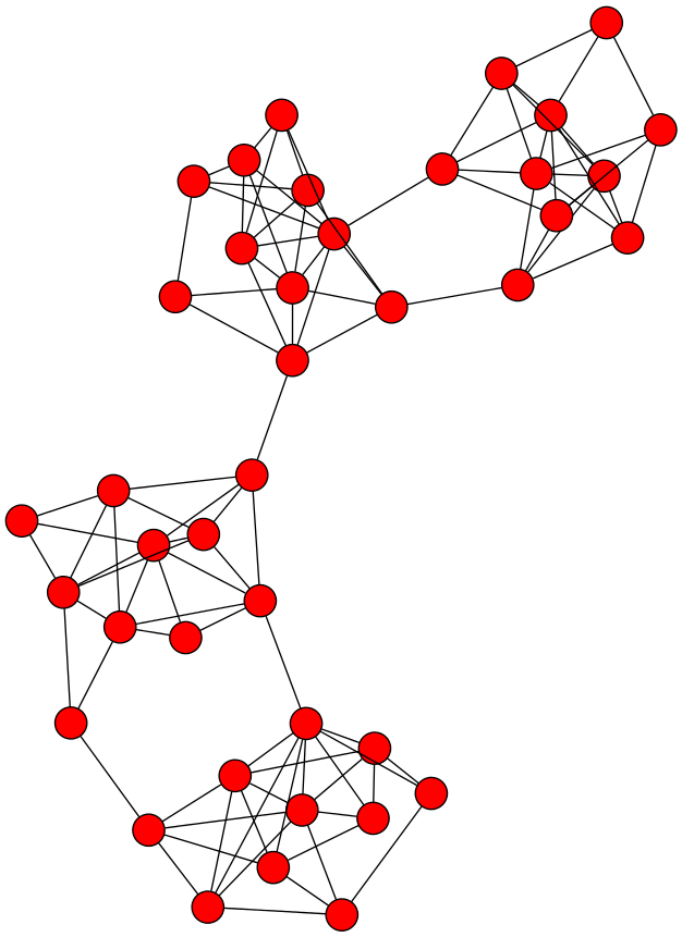


Example: Spectral Partitioning

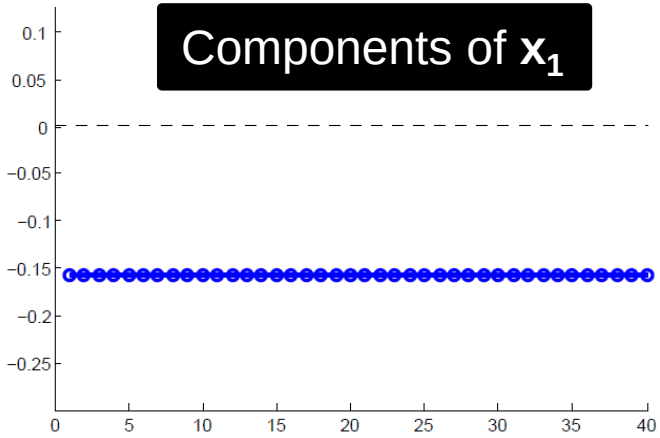


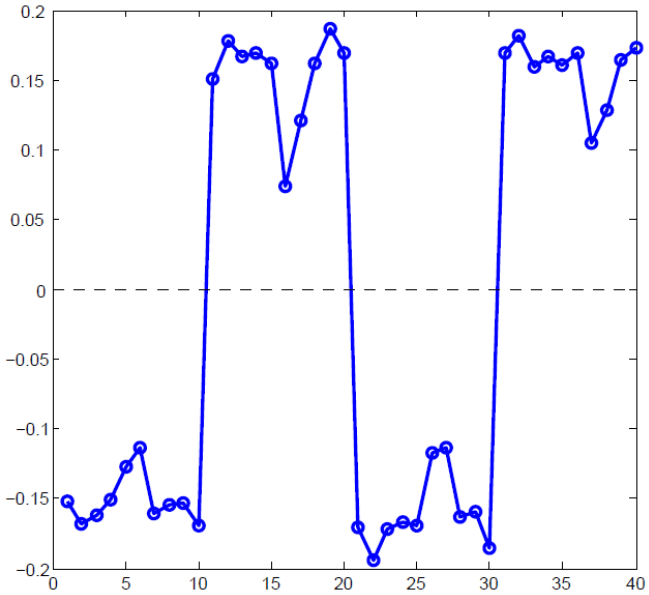


Example: Spectral Partitioning



Components of $\mathbf{x}_1$



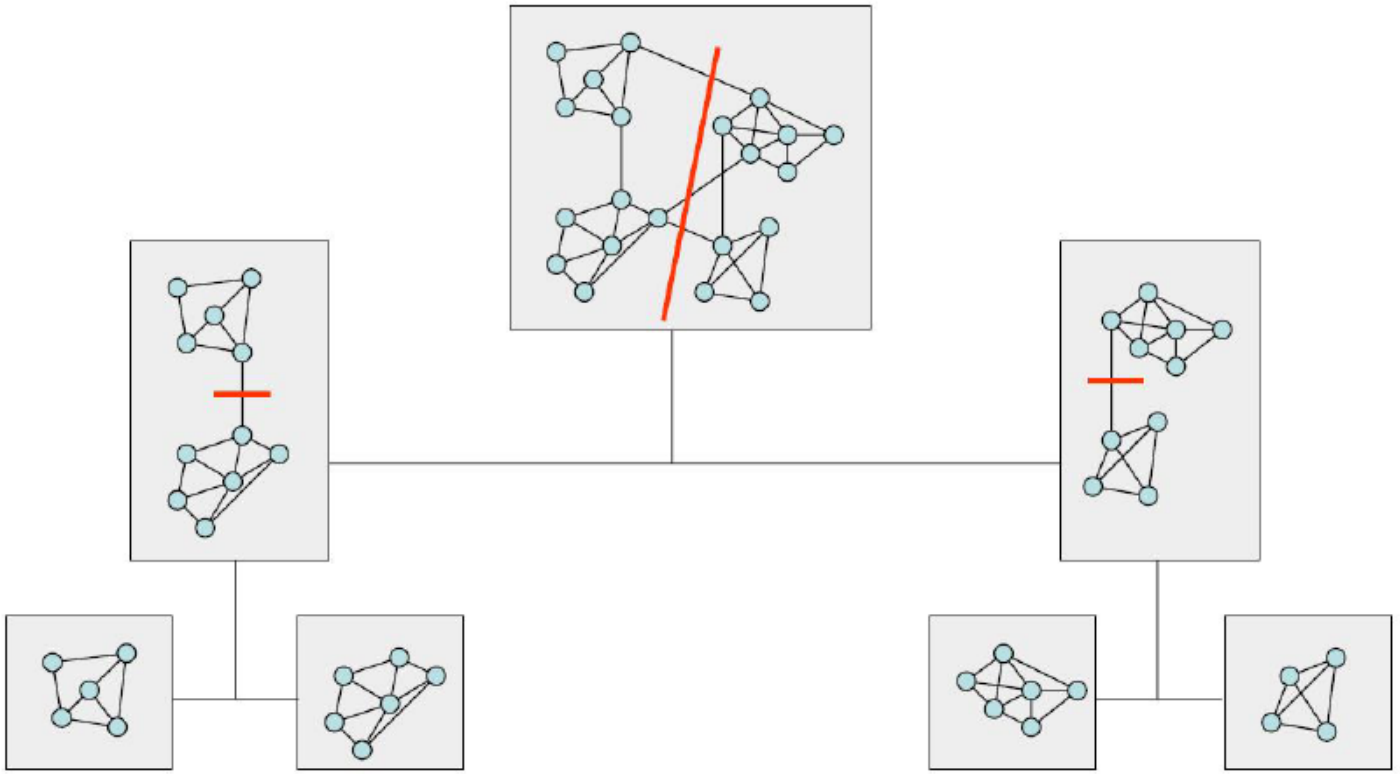


Components of $\mathbf{x}_3$

k-Way Spectral Clustering

- **How do we partition a graph into k clusters?**
- **Two basic approaches:**
  - **Recursive bi-partitioning** [Hagen et al., '92]
    - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
    - Disadvantages: Inefficient, unstable
  - **Cluster multiple eigenvectors** [Shi-Malik, '00]
    - Build a reduced space from multiple eigenvectors
    - Commonly used in recent papers
    - A preferable approach…

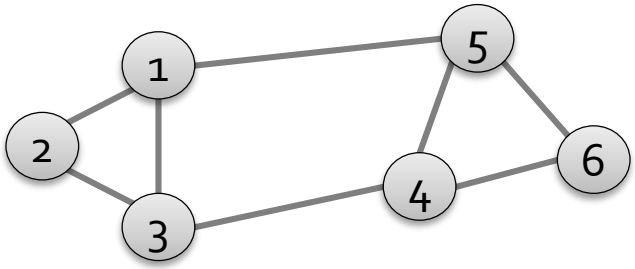
Recursive Bi-partitioning



Cluster Multiple Eigenvectors

- Use ***several of the eigenvectors*** to partition the graph
- If we use **m** eigenvectors, and set a threshold for each, we can get a partition into **2^m groups**, each group consisting of the nodes that are above or below threshold for each of the eigenvectors, in a particular pattern.

Example



Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	−5	−1	−1	−1
	1	2	4	−2	1	0
	1	1	1	3	−1	1
	1	−1	−5	−1	1	1
	1	−2	4	−2	−1	0
	1	−1	1	3	1	−1

If we use both the **2**nd and **3**rd eigenvectors:

- nodes **2** and **3** (positive in both)
- nodes **5** and **6** (negative in 2nd, positive in 3rd)
- nodes **1** and **4** alone

Note that while each eigenvector tries to produce a minimum-sized cut, successive eigenvectors have to satisfy more and more constraints => the cuts progressively worse.

Why use multiple eigenvectors?

- **Approximates the optimal cut** [Shi-Malik, '00]
  - Can be used to approximate optimal k-way normalized cut
- **Emphasizes cohesive clusters**
  - Increases the unevenness in the distribution of the data
  - Associations between similar points are amplified, associations between dissimilar points are attenuated
  - The data begins to "approximate a clustering"
- **Well-separated space**
  - Transforms data to a new "embedded space", consisting of $\boldsymbol{k}$ orthogonal basis vectors
- Multiple eigenvectors prevent instability due to information loss

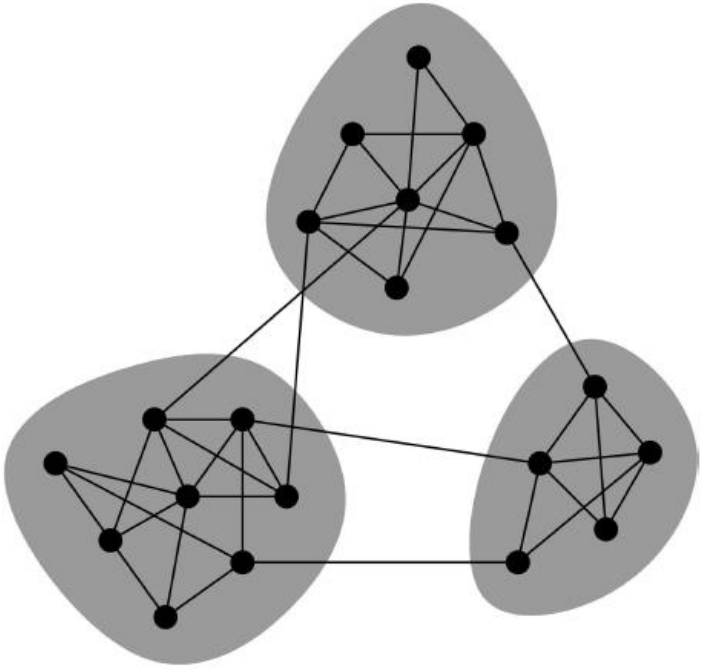
Many Other Partitioning Methods

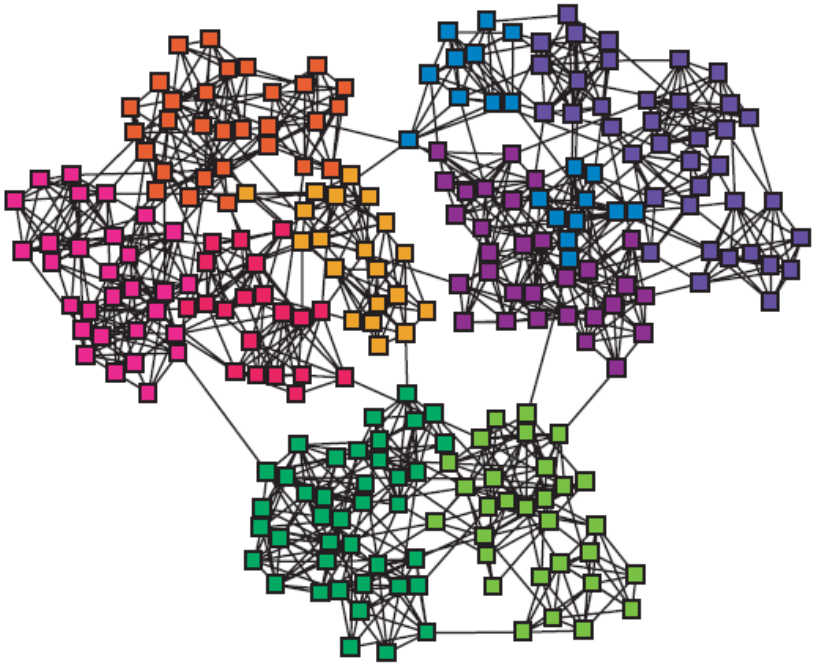
- **METIS:**
  - Heuristic but works really well in practice
  - http://glaros.dtc.umn.edu/gkhome/views/metis
- **Graclus:**
  - Based on kernel k-means
  - http://www.cs.utexas.edu/users/dml/Software/graclus.html
- **Louvain:**
  - Based on Modularity optimization
  - http://perso.uclouvain.be/vincent.blondel/research/louvain.html
- **Clique percolation method:**
  - For finding overlapping clusters
  - http://angel.elte.hu/cfinder/

How to Profile Network Communities?

Network and Communities

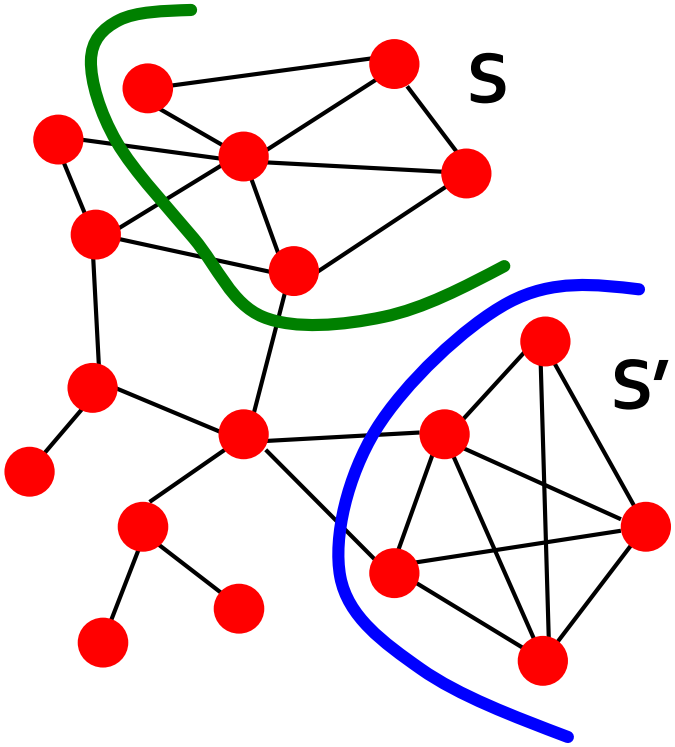
- **How should we think about large scale organization of clusters in networks?**
  - **Finding:** Community Structure





Community Score

- **How community-like is a set of nodes?**
- **A good cluster *S* has**
  - Many edges internally
  - Few edges pointing outside
- What's a good metric:
  Conductance



$$\phi(S) = \frac{|\{(i,j) \in E; i \in S, j \notin S\}|}{\sum_{s \in S} d_s}$$

Small conductance corresponds to good clusters

(Note |S| < |V|/2)

Network Community Profile Plot

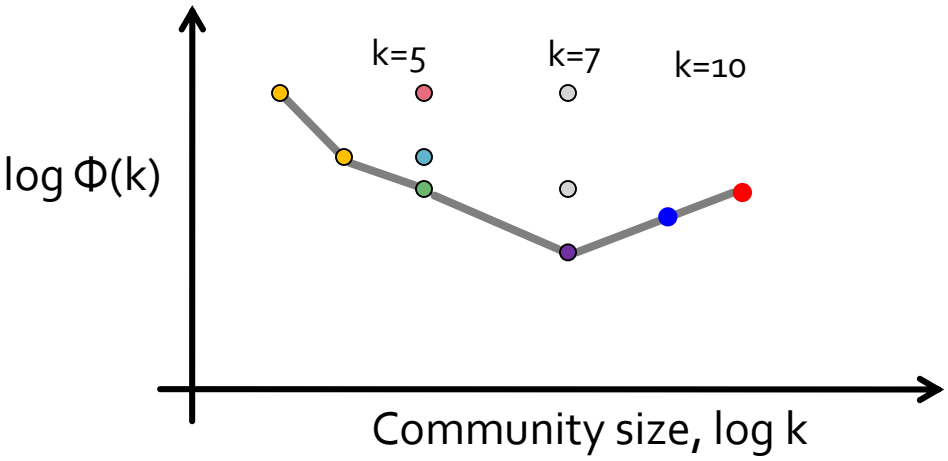
(Note |S| < |V|/2)

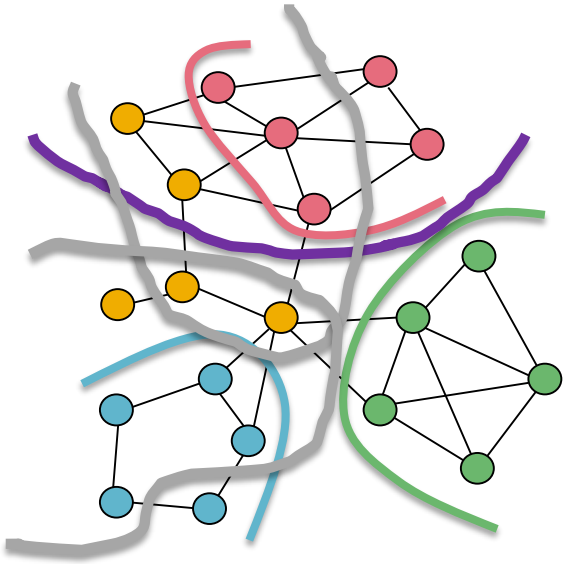
- Define:

  Network community profile (**NCP**) plot

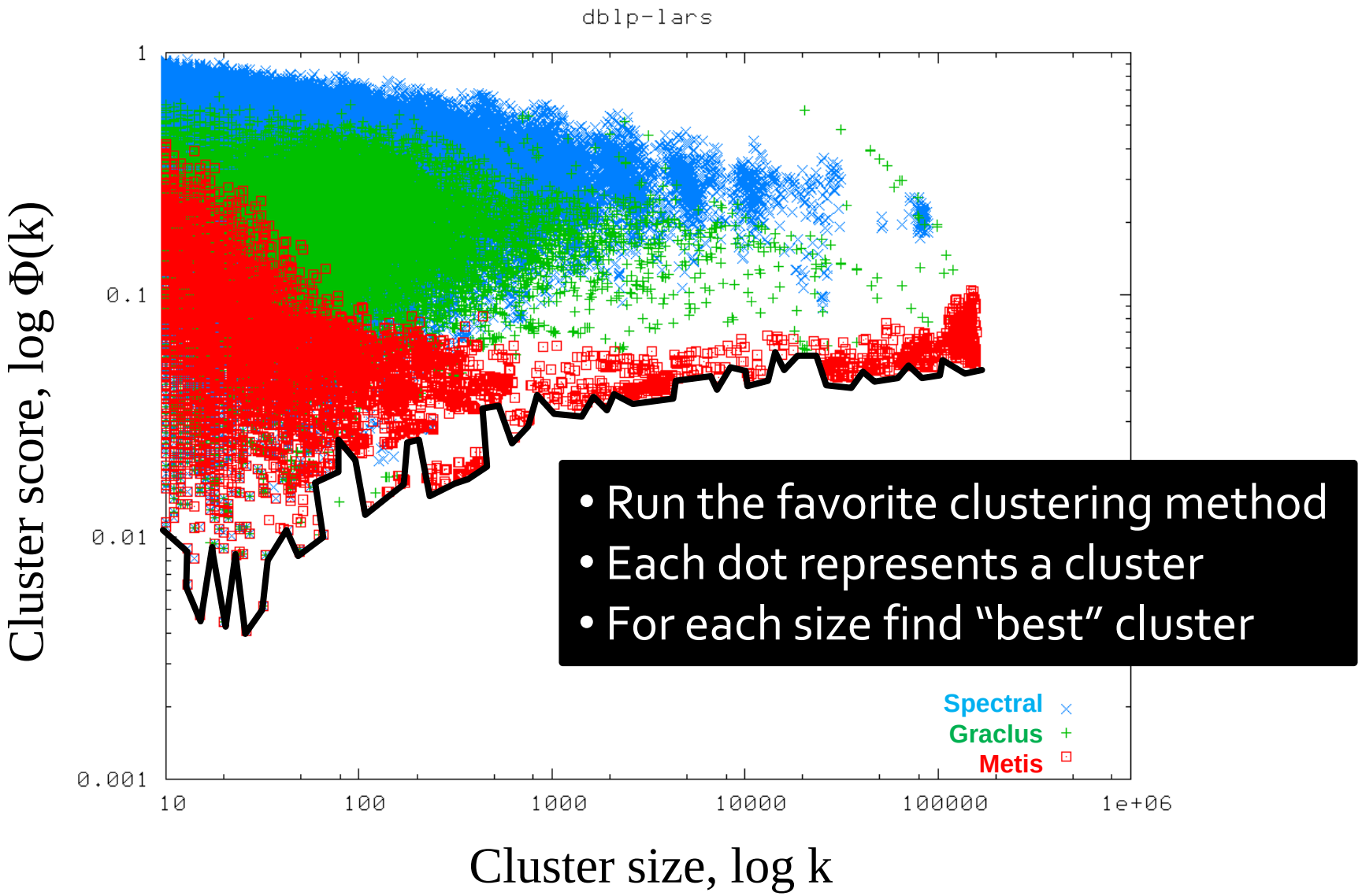
  Plot the score of **best** community of size *k*

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



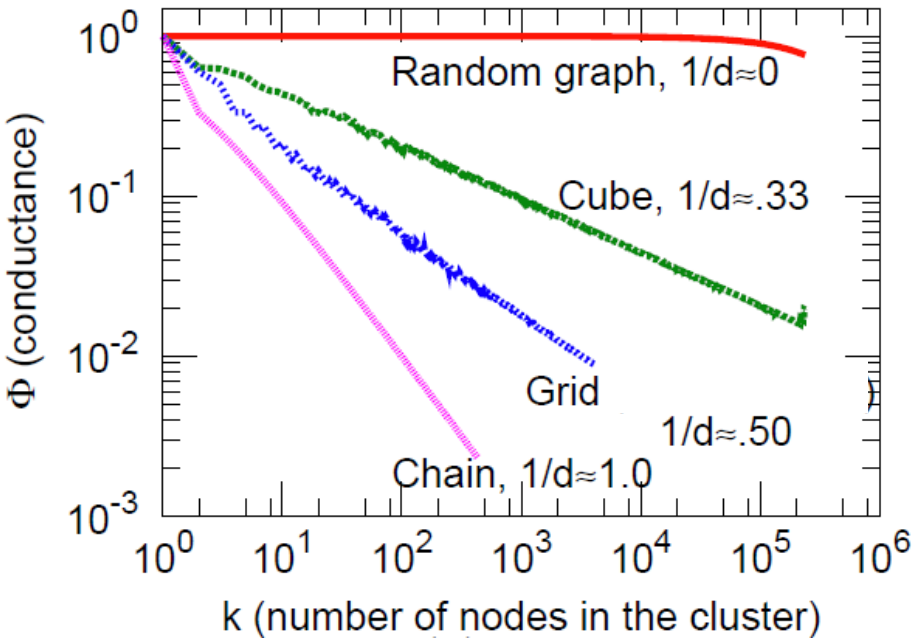


How to (Really) Compute NCP?

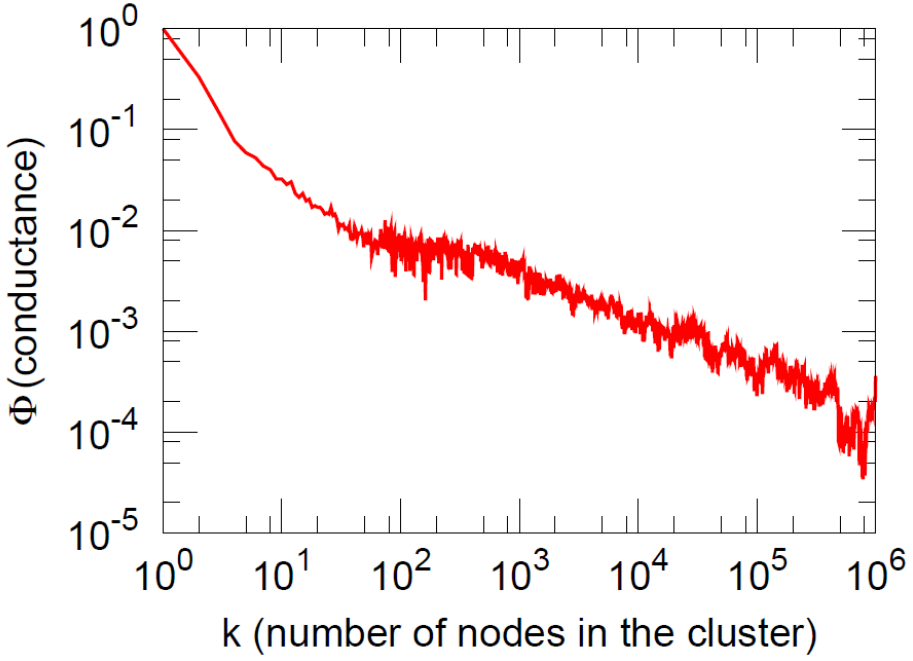


NCP Plot: Meshes

- **Meshes, grids, dense random graphs:**



d-dimensional meshes



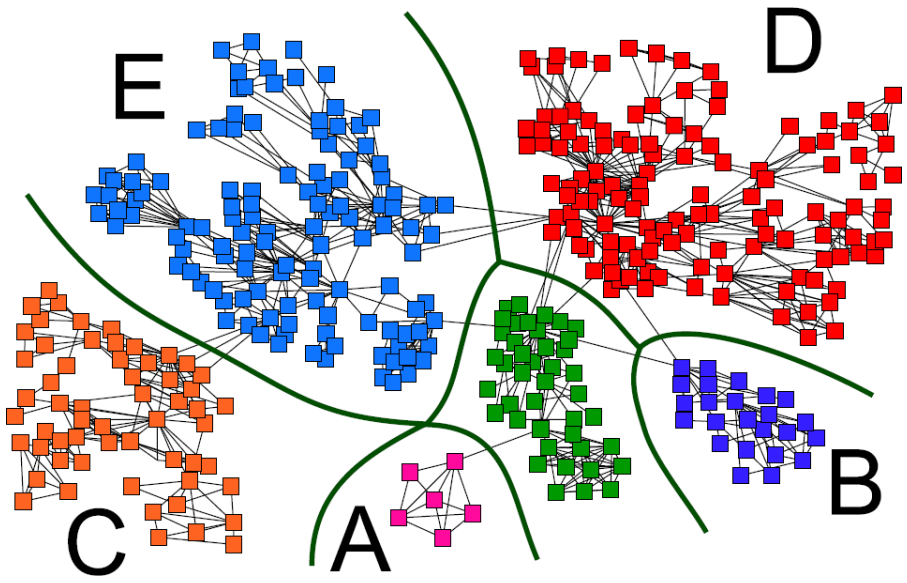
California road network

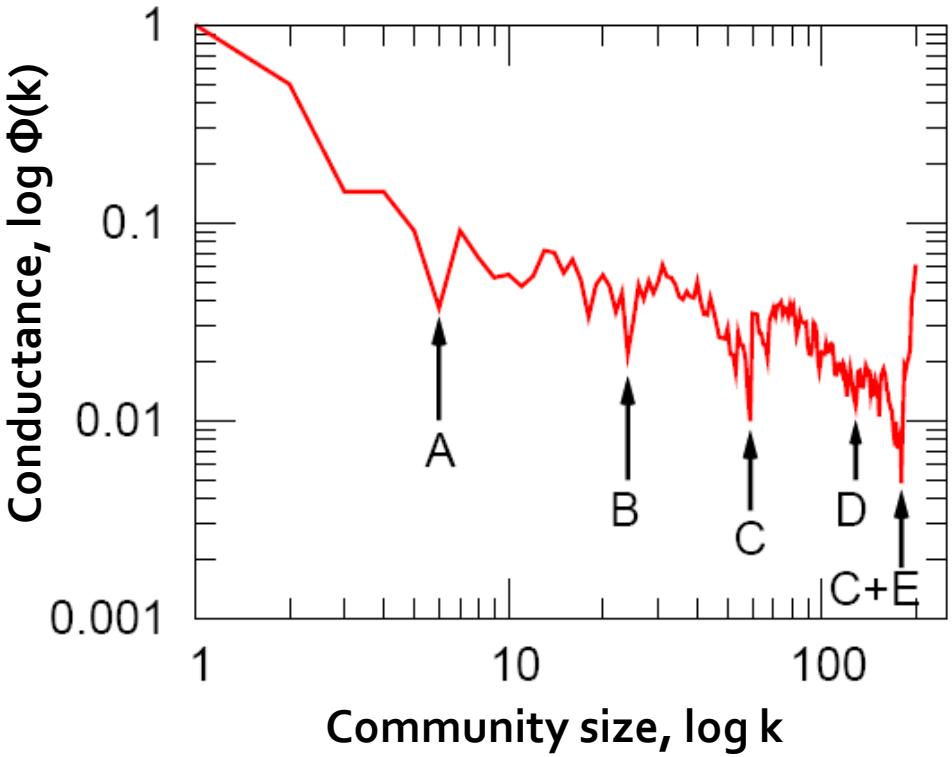
[WWW '08]

NCP plot: Network Science

- **Collaborations between scientists in networks**

  [Newman, 2005]



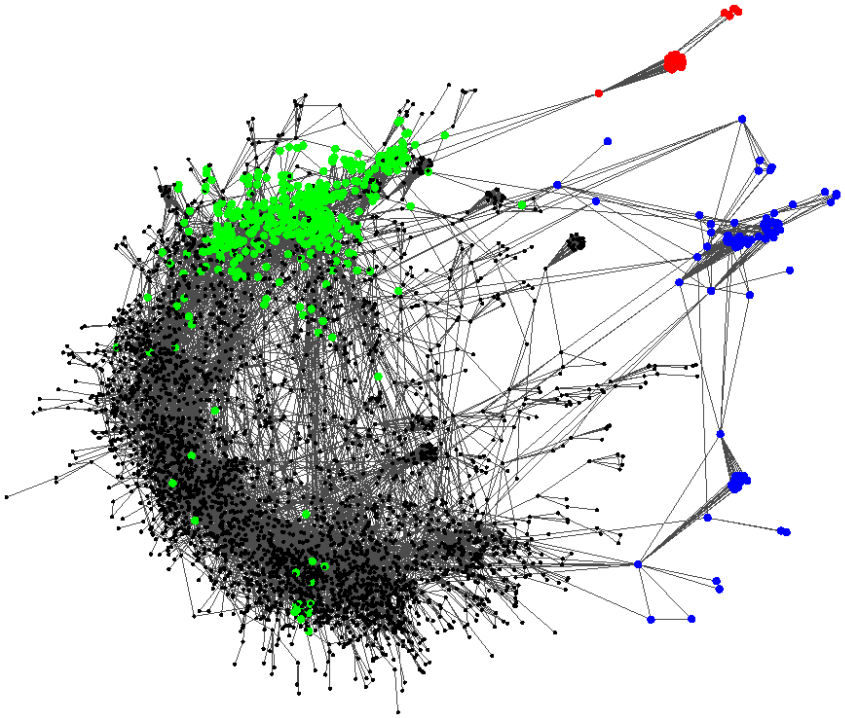


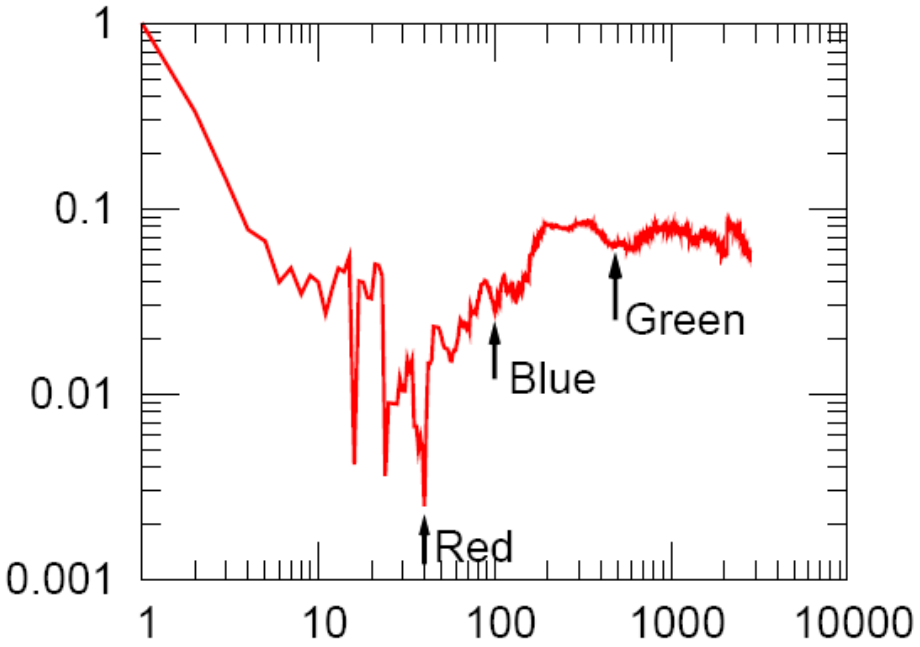
Dips in the conductance graph correspond to the "good" clusters we can visually detect

[Internet Mathematics '09]

Large Networks: Very Different

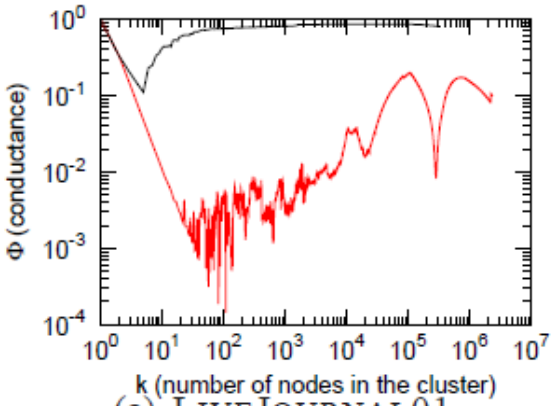
Typical example: General Relativity collaborations (n=4,158, m=13,422)



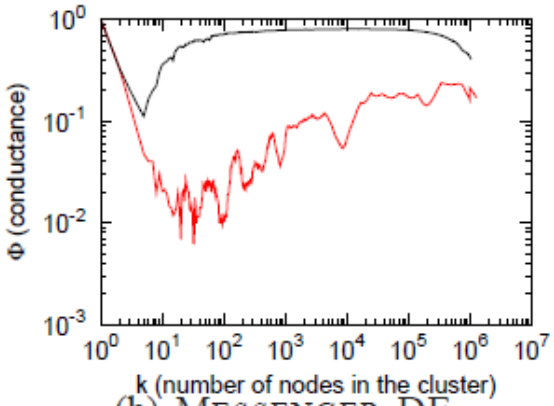


[Internet Mathematics '09]

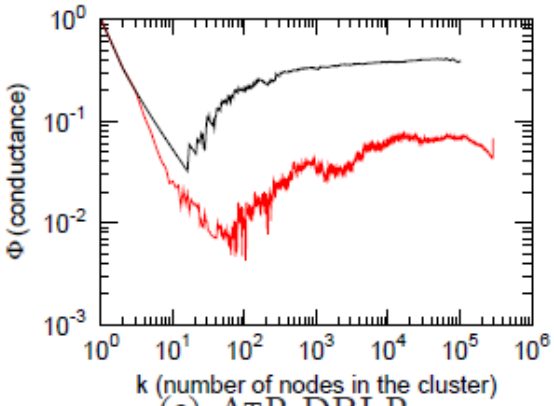
More NCP Plots of Networks



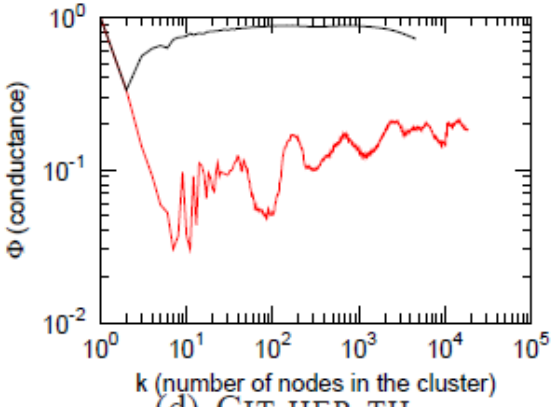
(a) LIVEJOURNAL01



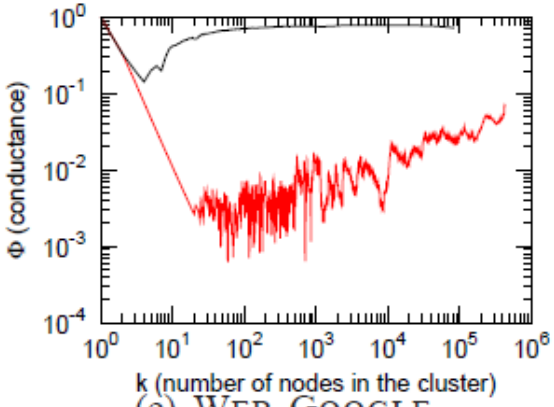
(b) MESSENGER-DE



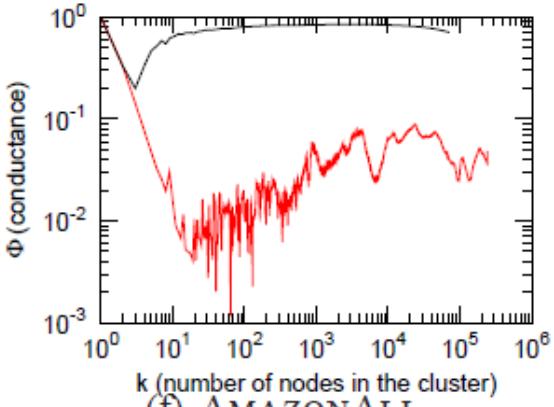
(c) ATP-DBLP



(d) CIT-HEP-TH



(e) WEB-GOOGLE

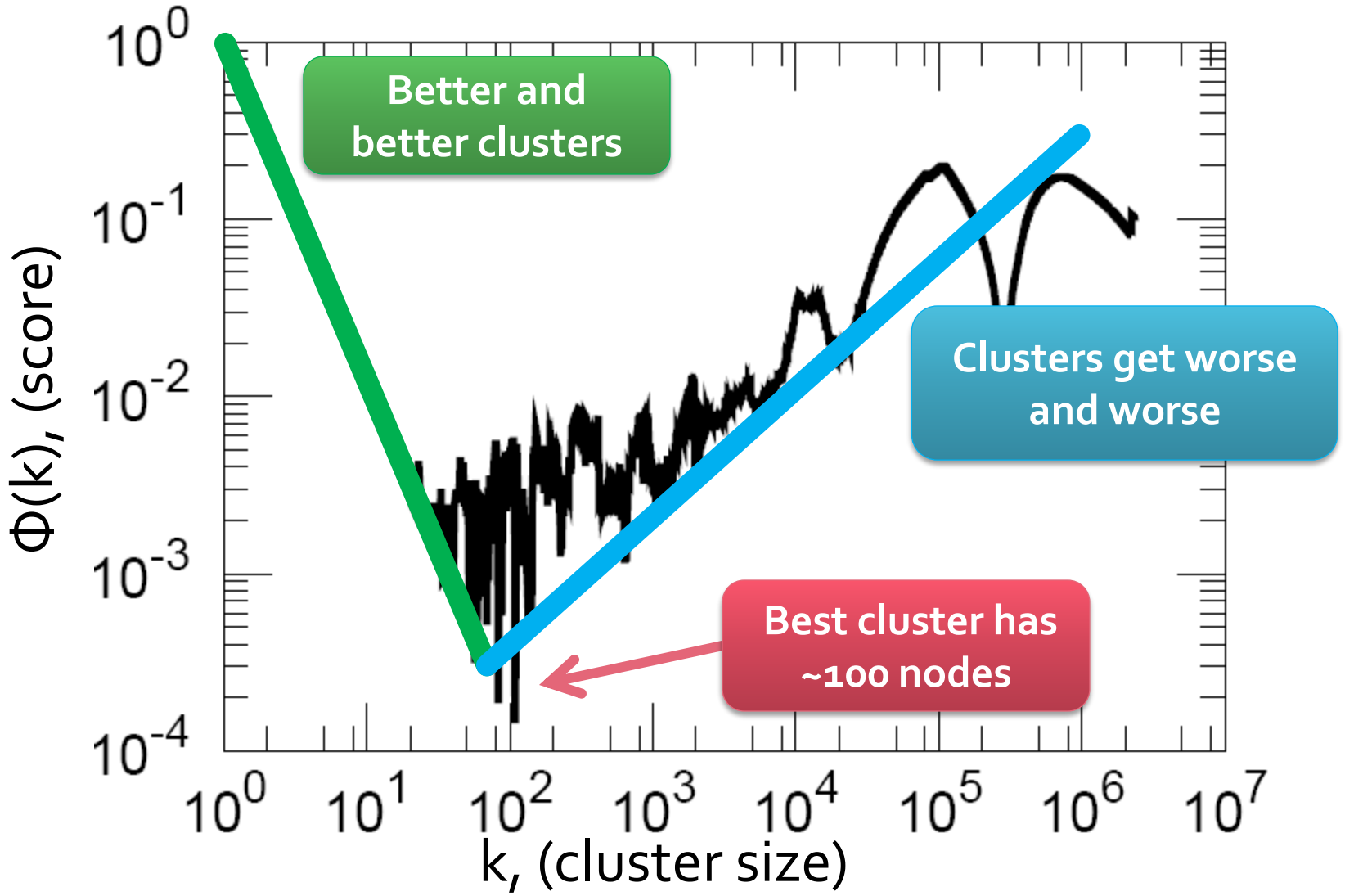


(f) AMAZONALL

-- Rewired graph

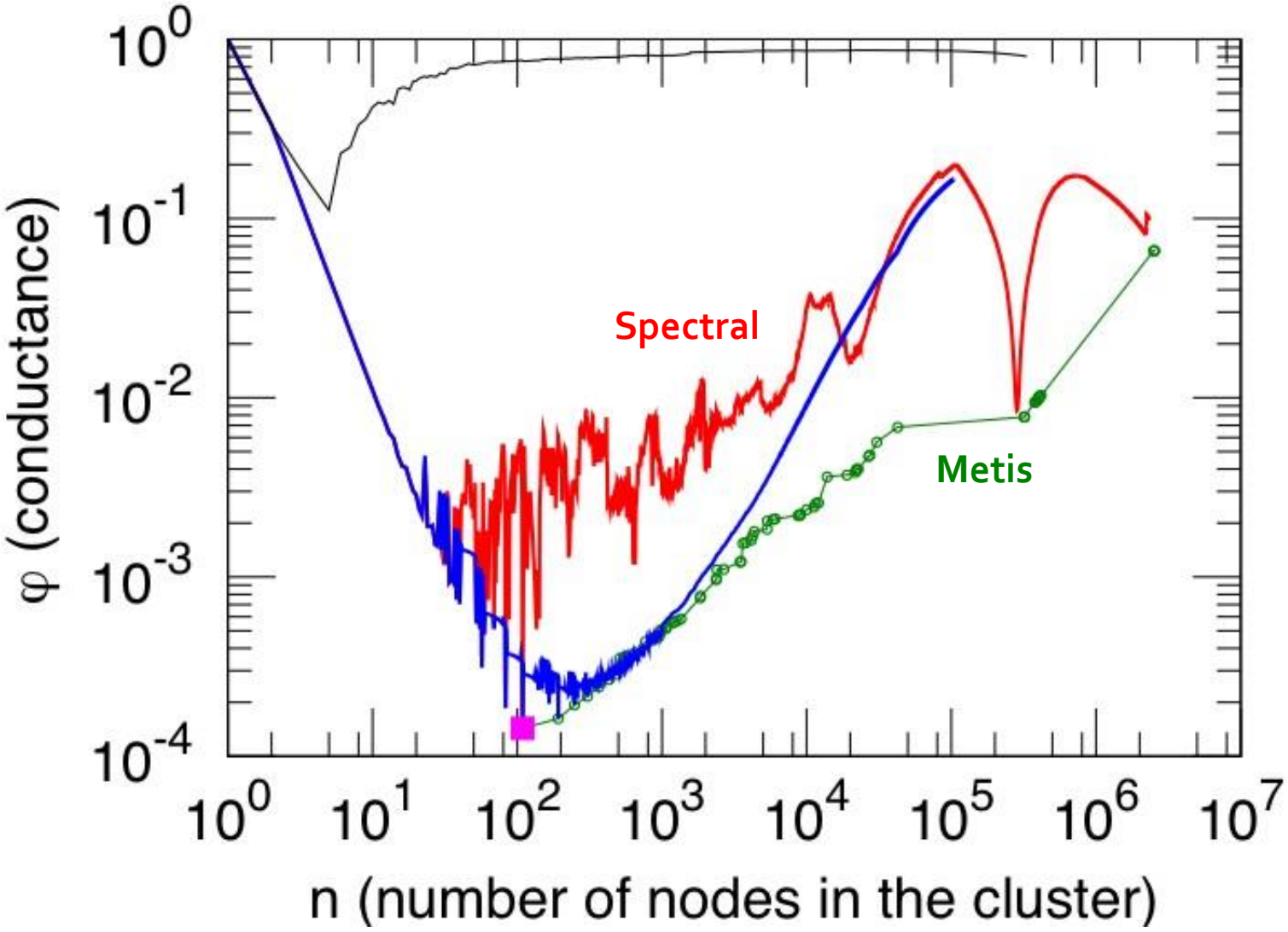
-- Real graph

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



[WWW '09]

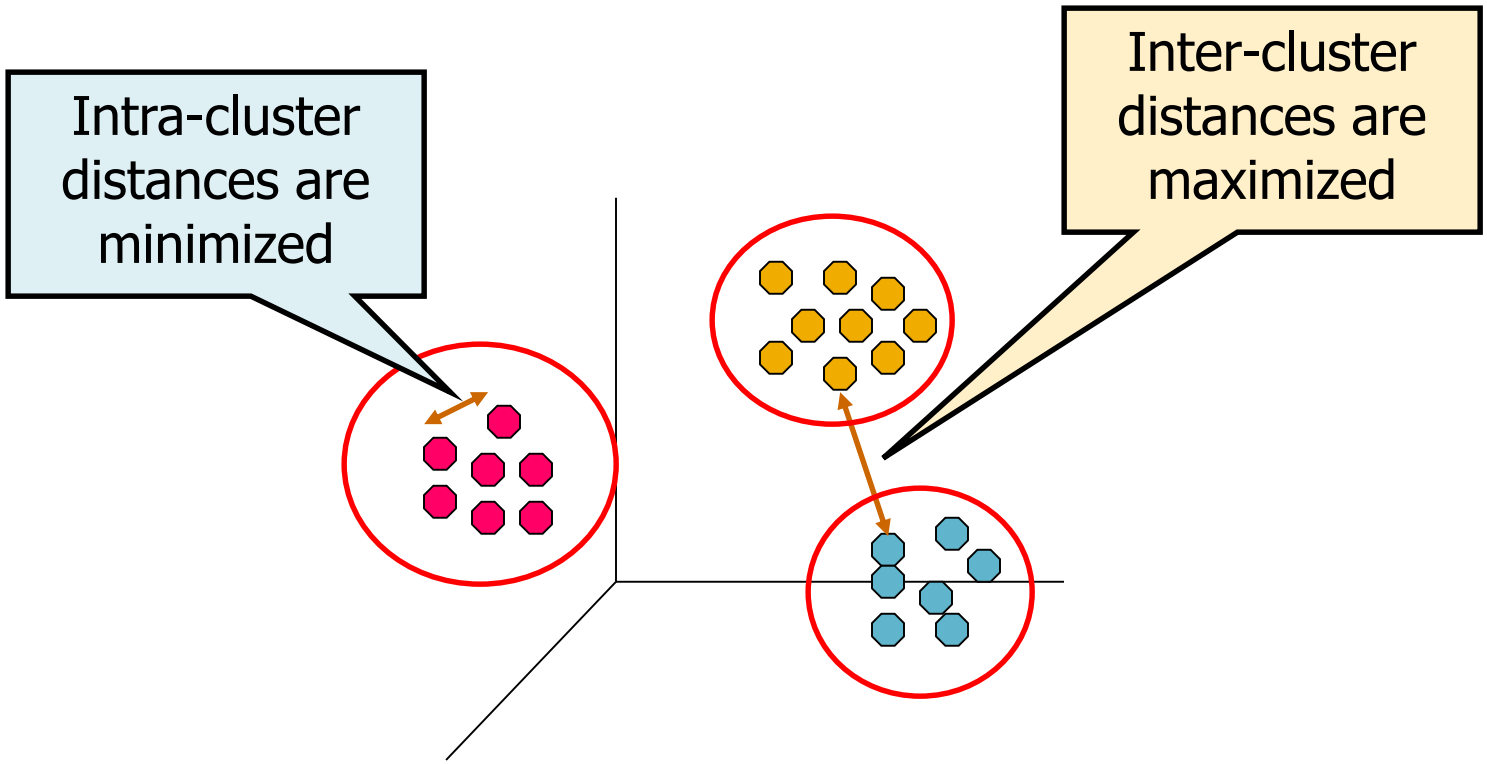
NCP: Live Journal



Communities:
Issues and Questions

What is Cluster Analysis?

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



Clusters Can Be Ambiguous

How many clusters?

Six Clusters

Two Clusters

Four Clusters

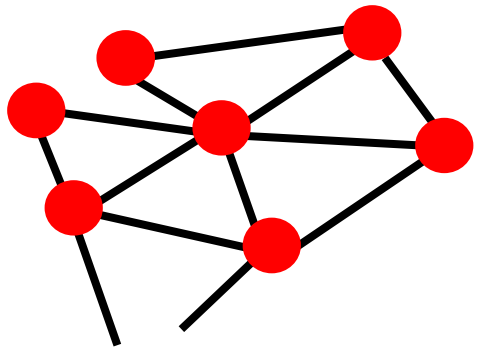
Communities: Issues and Questions

- **Some issues with community detection:**
  - Many different formalizations of clustering objective functions
  - Objectives are **NP-hard** to optimize exactly
  - Methods can find clusters that are systematically "biased"
- **Questions:**
  - **How well do algorithms optimize objectives?**
  - **What clusters do different methods find?**

[WWW '09]

Many Different Objective Functions

- **Single-criterion:**
  - Modularity: $m\text{-}E(m)$
  - Edges cut: c
- **Multi-criterion:**
  - Conductance: $c/(2m+c)$
  - Expansion: c/n
  - Density: $1\text{-}m/n^2$
  - CutRatio: $c/n(N\text{-}n)$
  - Normalized Cut: $c/(2m+c) + c/2(M\text{-}m)+c$
  - Flake-ODF: *frac. of nodes with more than ½ edges pointing outside S*



n: nodes in S
m: edges in S
c: edges pointing outside S

Many Classes of Algorithms

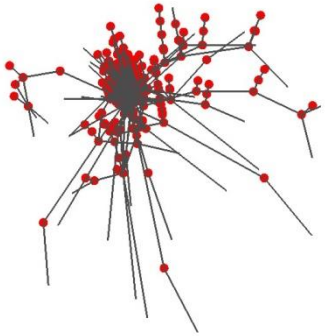
Many algorithms to implicitly or explicitly optimize objectives and extract communities:

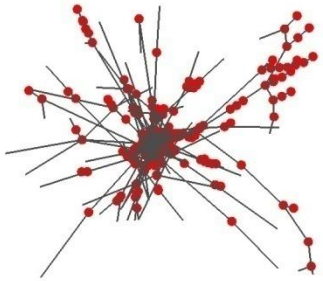
- **Heuristics:**
  - Girvan-Newman, Modularity optimization: popular heuristics
  - Metis: multi-resolution heuristic [Karypis-Kumar '98]
- **Theoretical approximation algorithms:**
  - Spectral partitioning

[WWW '09]

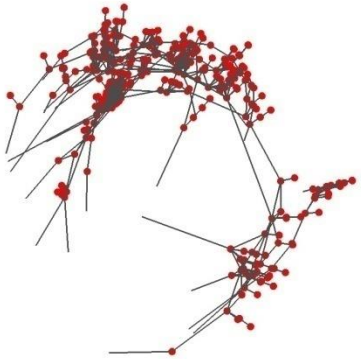
Properties of Clusters (1)

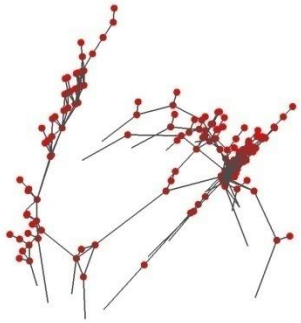
500 node communities from Spectral:





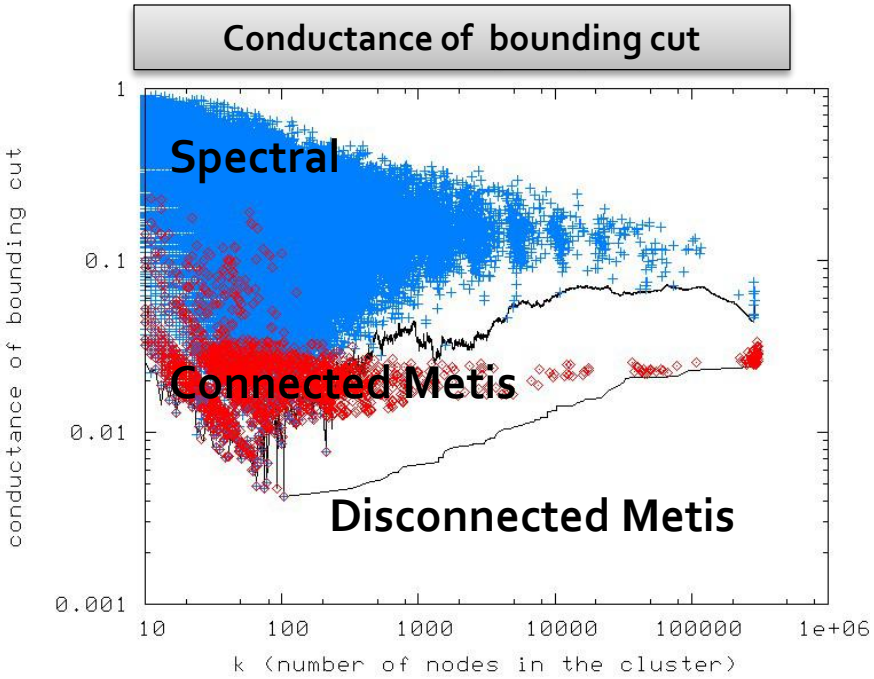
500 node communities from Metis:

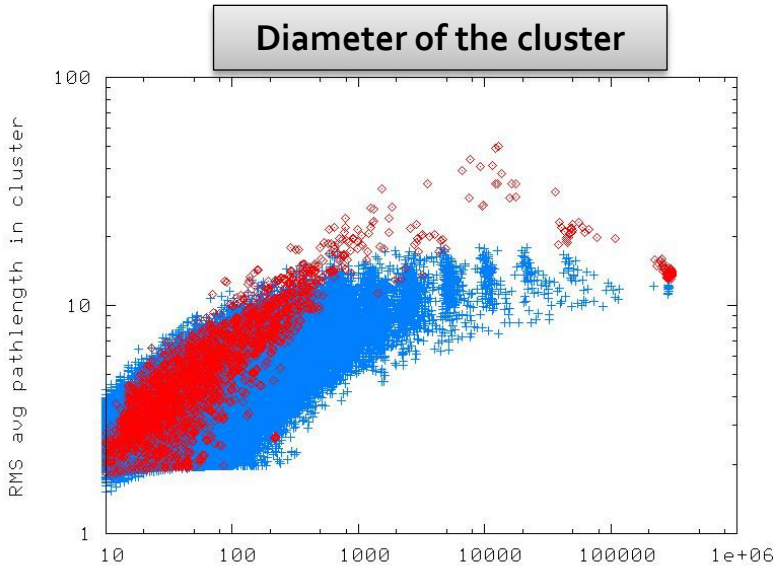


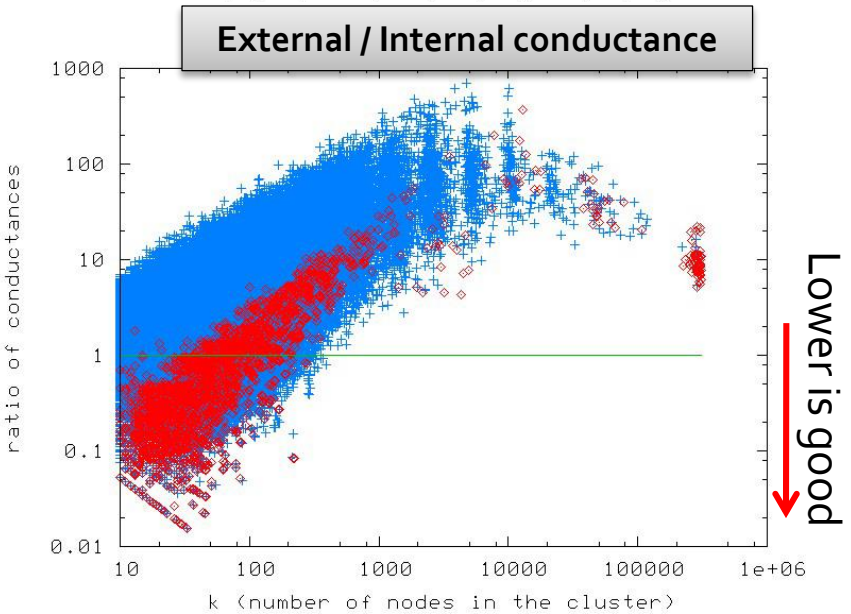


[WWW '09]

Properties of Clusters (2)



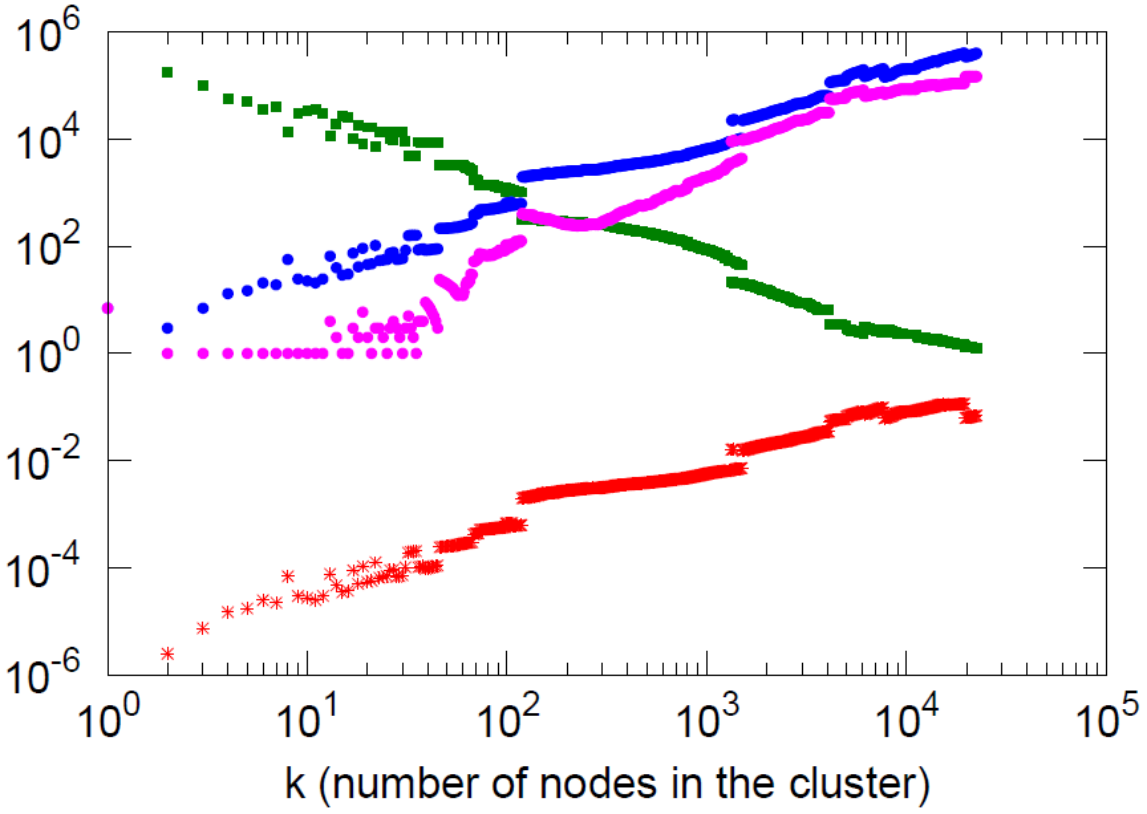




- Metis gives sets with better conductance
- Spectral gives tighter and more well-rounded sets

[WWW '09]

Single-criterion Objectives

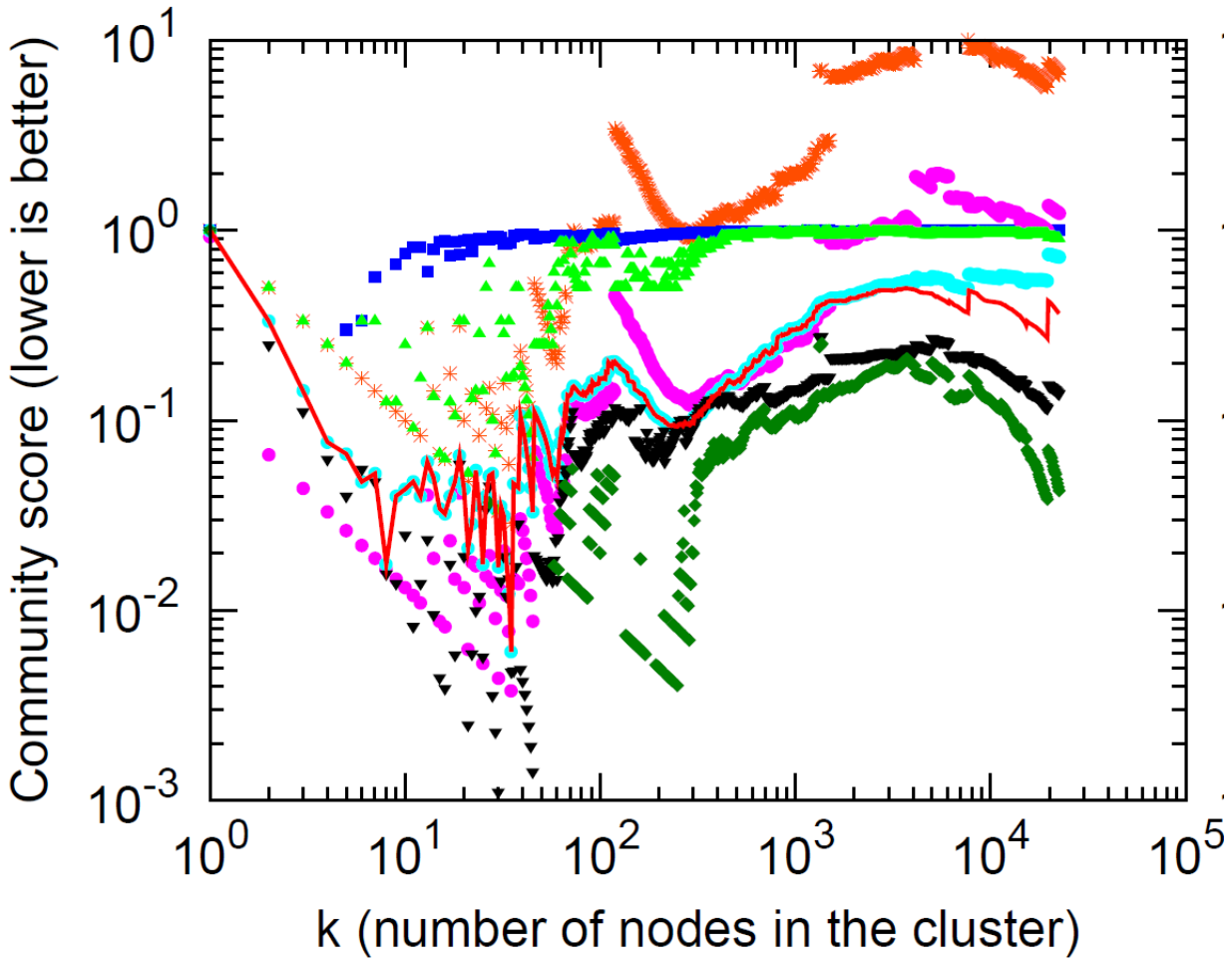


Observations:

- All measures are monotonic
- **Modularity**
  - prefers large clusters
  - Ignores small clusters

[WWW '09]

Multi-criterion Objectives



- All qualitatively similar
- **Observations:**
  - Conductance, Expansion, Norm-cut, Cut-ratio are similar
  - Flake-ODF prefers larger clusters
  - Density is bad
  - Cut-ratio has high variance