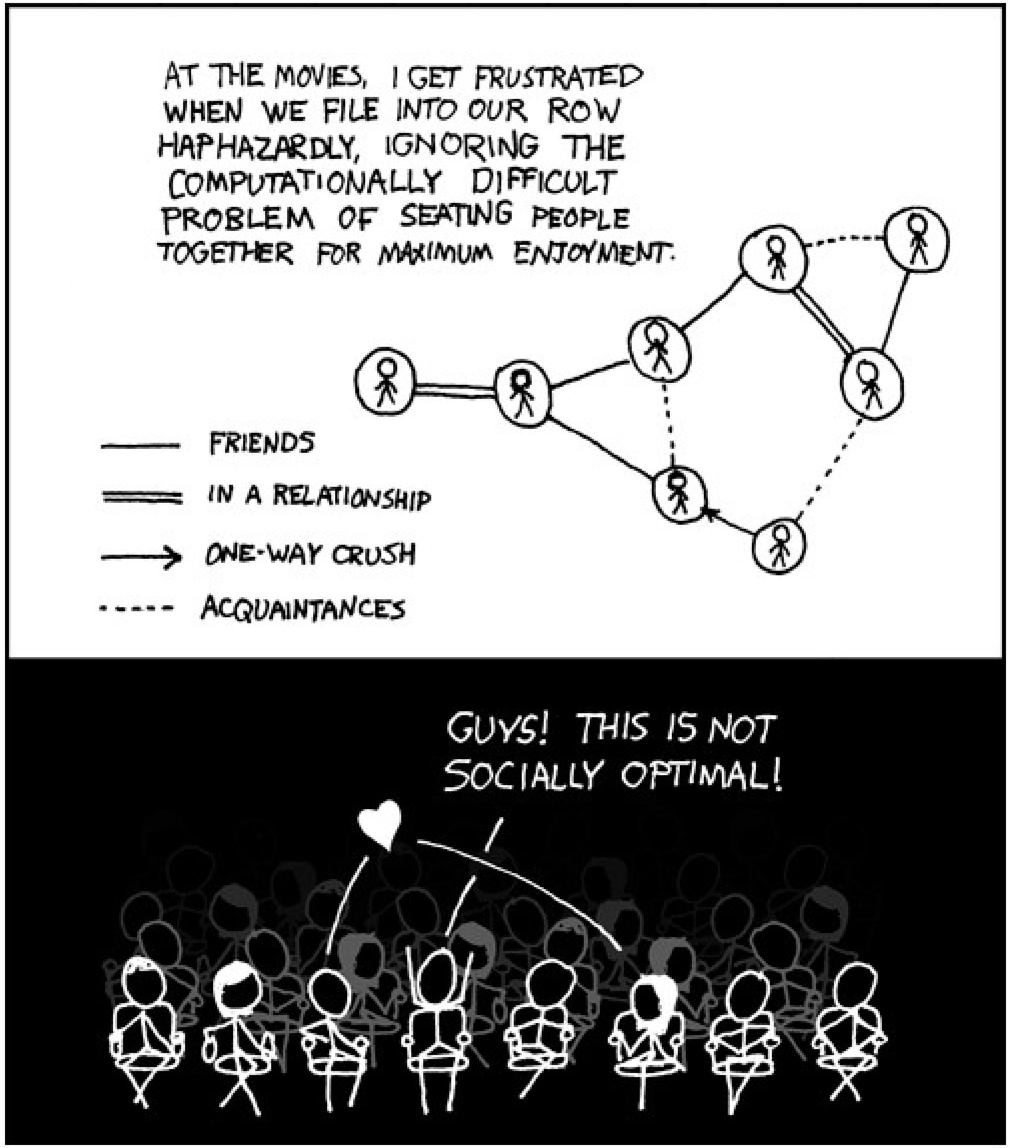
Web Mining e Analisi delle Reti Sociali

Community Discovery

Michele Berlingerio

Movie Seating

|< < Prev Random Next > >|



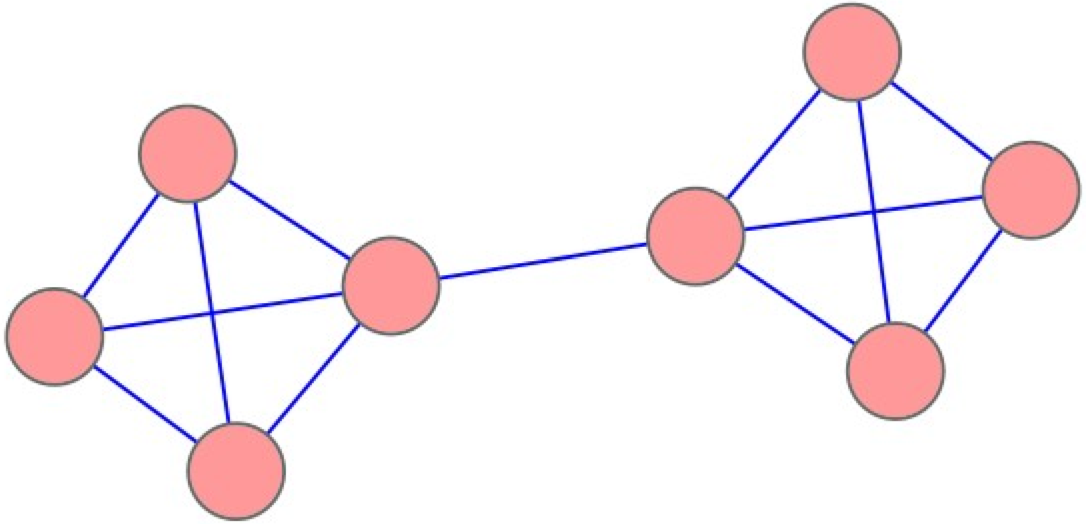
|< < Prev Random Next > >|

Permanent link to this comic: http://xkcd.com/173/
Image URL (for hotlinking/embedding): http://imgs.xkcd.com/comics/movie_seating.png

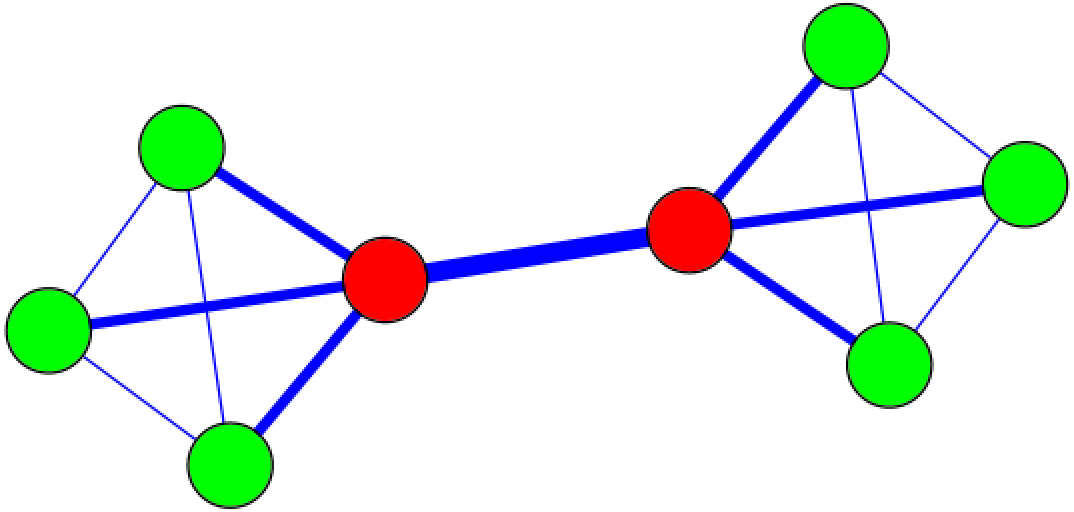
Comunità

- Insieme di nodi “simili”
- Similitudini:
  - Caratteristiche comuni
    - Topologiche / Semantiche
  - Interazioni
  - … una qualsiasi misurabile funzione di “similitudine”
- Caveat
  - I nodi potrebbero non conoscere la propria appartenenza
  - I link possono essere positivi o negativi

Quali comunità?

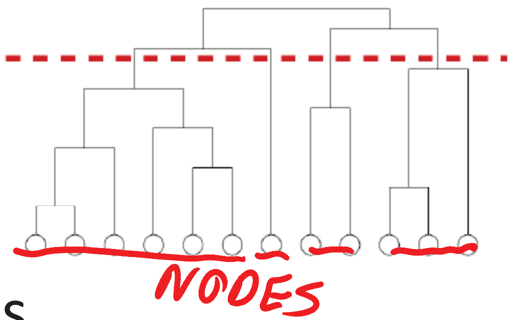


Come individuarle?

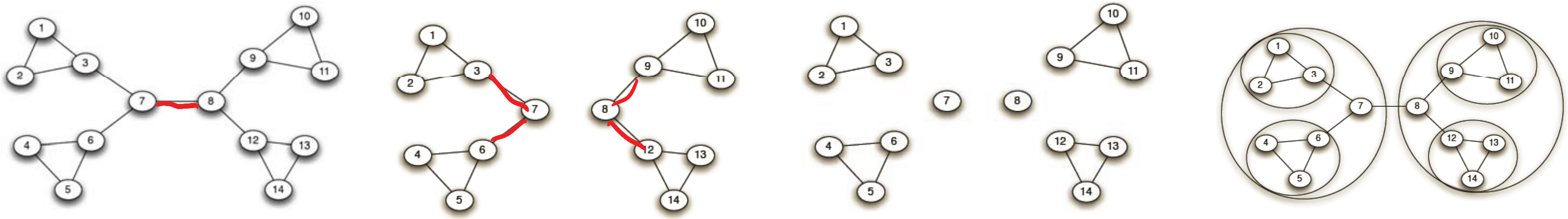


Method1: Girvan-Newman

- Divisive hierarchical clustering based on edge betweenness:
  - Number of shortest paths passing through the edge
- **Girvan-Newman Algorithm:**
  - 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



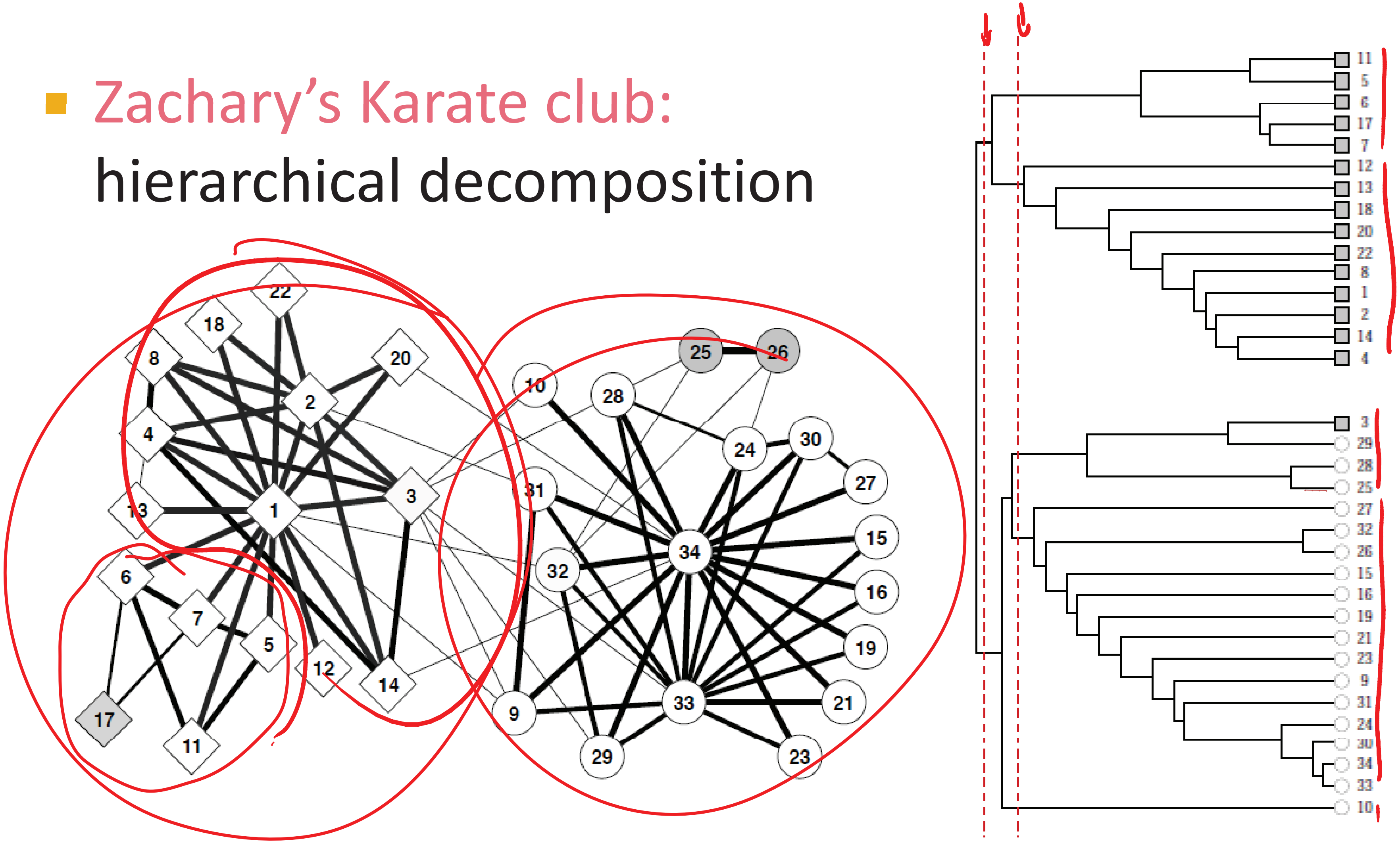
- Example:



[Newman-Girvan PhysRevE '03]

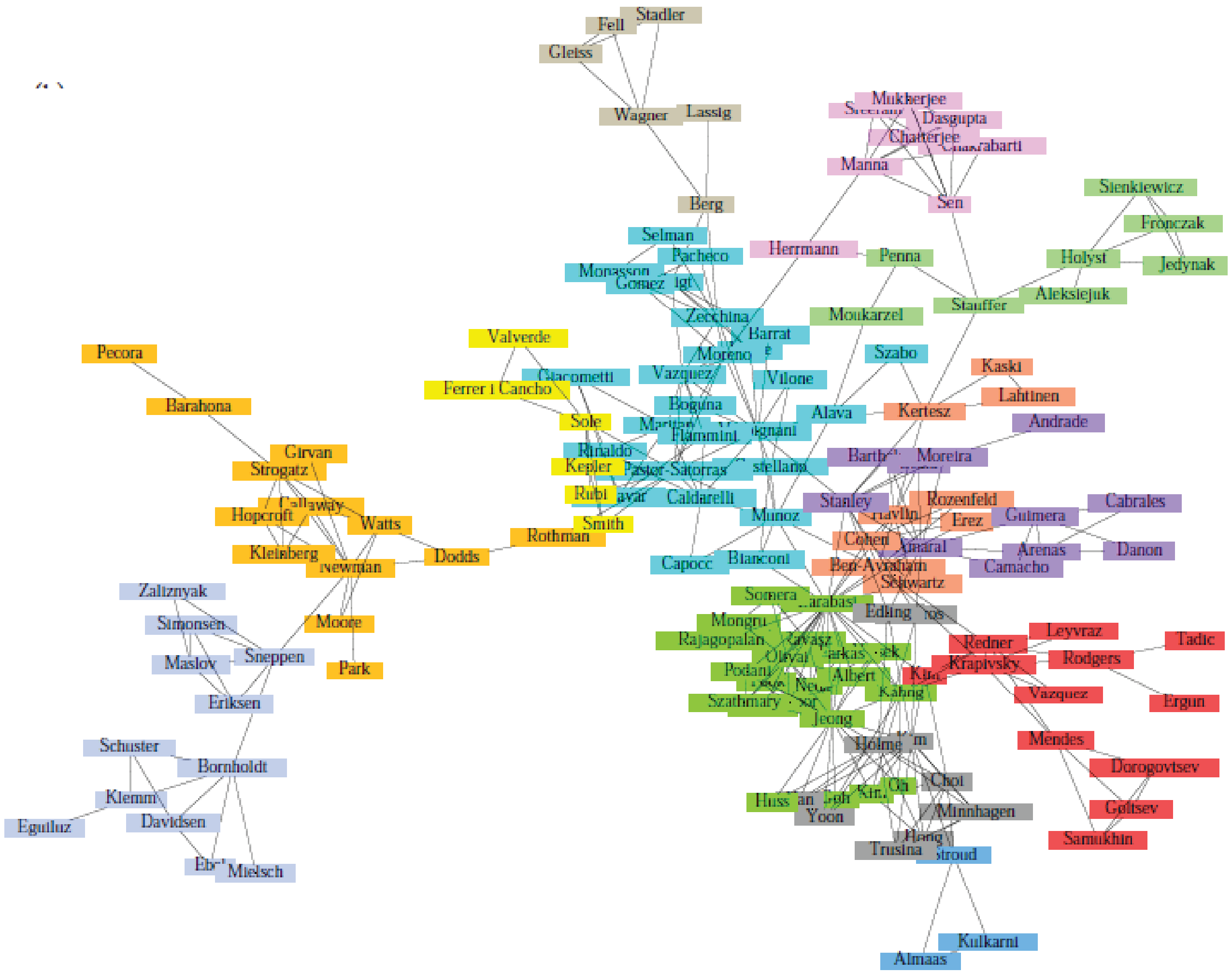
Girvan-Newman: Example

- Zachary's Karate club: hierarchical decomposition



[Newman-Girvan PhysRevE '03]

Girvan-Newman: Example



Communities in physics collaborations

How to select the number of clusters?

Define **modularity** to be

$$Q = (\text{number of edges within groups}) - (\text{expected number within groups})$$

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…number of edges

Modularity: Definition

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

$c_i \in \{1 \ldots k\}$

- Then:

$\frac{1}{4m} \sum_{\text{GROUPS}} \left[\sum_{i,j} A_{ij} - \sum_{ij} \frac{k_i \cdot k_j}{2m} \right]$

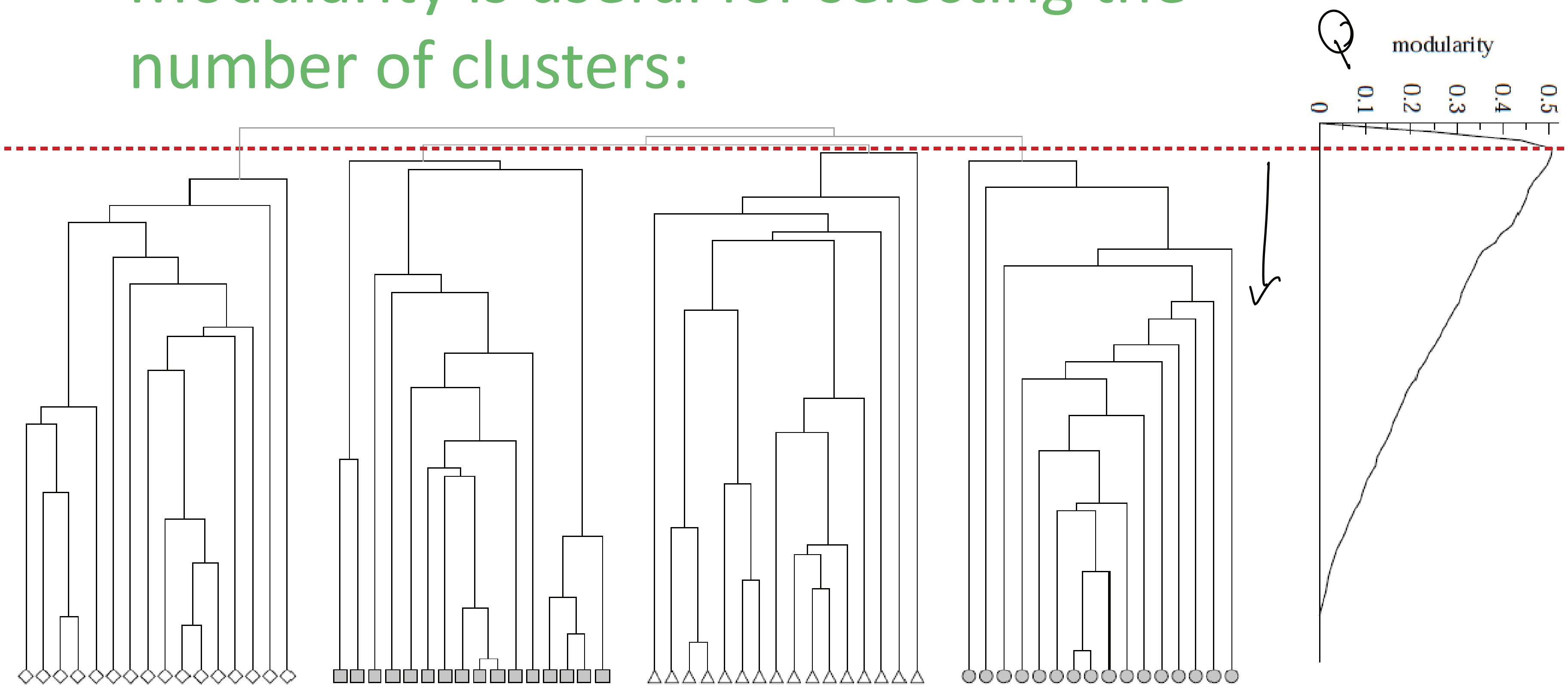
$$Q = \frac{1}{4m} \left[\sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \right]$$

m … number of edges
A_{ij} … 1 if (i,j) is edge, else 0
k_i … degree of node i
c_i … group id of node i
δ(a, b) … 1 if a=b, else 0

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

Modularity: Number of clusters

- Modularity is useful for selecting the number of clusters:



Why not optimize modularity directly?

Method2: Modularity optimization

- Consider splitting the graph in two communities
- Modularity *Q* is: $\sum_{\substack{i,j \text{ in} \\ \text{same group}}} A_{ij} - \frac{k_i k_j}{2m}$

$$Q = \sum_{ij} B_{ij} s_i s_j = \sum_i s_i \sum_j B_{ij} s_j = s^T B s$$

- Or we can write in matrix form as

$$Q = \frac{1}{4m} \sum_{ij} \underbrace{\left(A_{ij} - \frac{k_i k_j}{2m} \right)}_{B_{ij}} s_i s_j = \frac{1}{4m} \mathbf{s}^T \mathbf{B} \mathbf{s}$$

  - **s** … vector of group memberships s_i={+1, -1}
  - *B* … *modularity matrix*

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

Note: each row (column) of B sums to 0

Fast Modularity Optimization

- Task: Find $s \in \{-1,+1\}^n$ that maximizes Q
- Rewrite Q in terms of eigenvalues β_i of B

$$Q = s^{\mathrm{T}}\left[\overbrace{\sum_{i=1}^{n} u_i \beta_i u_i^{\mathrm{T}}}^{=B}\right]s = \sum_i s^{\mathrm{T}} u_i \beta_i u_i^{\mathrm{T}} s = \sum_{i=1}^{n} \left(s^{\mathrm{T}} u_i\right)^2 \beta_i$$

$\beta_1 > \beta_2 > \beta_3 > \ldots$

- To maximize Q, easiest way is to make $\mathbf{s} = \lambda u_1$
  - Assigns all weight in the sum to β_1 (largest eigval)
    - (all other $\mathbf{s}^{\mathrm{T}}\mathbf{u}_i$ terms zero because of orthonormality)
    - Unfortunately, elements of **s** must be ± 1
    - In general, finding optimal **s** is NP-hard

Finding a splitting strategy

$$\arg\max_{s} Q = \sum_{i=1}^{n} \left(s^{\mathrm{T}} u_i\right)^2 \beta_i \approx \left(\sum_{i=1}^{n} s_i u_{1i}\right)^2 \beta_1 \qquad \beta_1 = \max \beta_i$$

- Heuristic: try to maximize only the β_1 term

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

- Similar in spirit to the spectral partitioning algorithm (we will explore it next time)
- Continue the bisection hierarchically

Fast Modularity Optimization

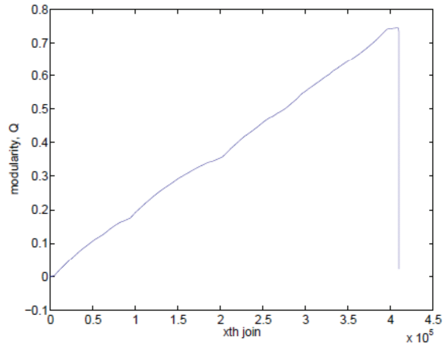
- Fast Modularity Optimization Algorithm:
  - Find leading eigenvector u_1 of modularity matrix B
  - Divide the nodes by the signs of the elements of u_1
  - Repeat hierarchically until:
    - If a proposed split does not cause modularity to increase, declare community indivisible and do not split it
    - If all communities are indivisible, stop
- How to find u_1? Power method!
  - Iterative multiplication, normalization
  - Start with random v, until convergence:

$$v_{k+1} = \frac{Bv_k}{\left\| Bv_k \right\|}$$

Even more heuristic approaches

- Also, can combine with other methods:
  - Randomly divide the nodes into two groups
  - Move the node that, if moved, will increase Q the most
  - Repeat for all nodes, with each node only moved once
  - Once complete, find intermediate state with highest Q
  - Start from this state and repeat until Q stops increasing
  - Good results for "fine-tuning" the spectral method

- CNM Algorithm (Clauset-Newman-Moore '04):
  - (1) Separate each vertex solely into n community
  - (2) Calculate ΔQ for all possible community pairs
  - (3) Merge the pair of the largest increase in Q
  - Repeat (2)&(3) until one community remains
  - Cross cut the dendogram where Q is maximum



Comparison to other methods

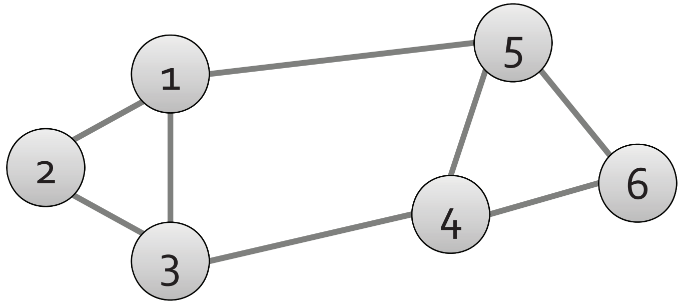
network	size n	modularity Q GN	CNM	DA	Fast modularity
karate	34	0.401	0.381	0.419	0.419
jazz musicians	198	0.405	0.439	0.445	0.442
metabolic	453	0.403	0.402	0.434	0.435
email	1133	0.532	0.494	0.574	0.572
key signing	10 680	0.816	0.733	0.846	0.855
physicists	27 519	–	0.668	0.679	0.723

GN = Girvan-Newman, $O(n^3)$
CNM = Greedy merging ($n \log^2 n$)
DA = External Optimization $O(n^2 \log^2 n)$

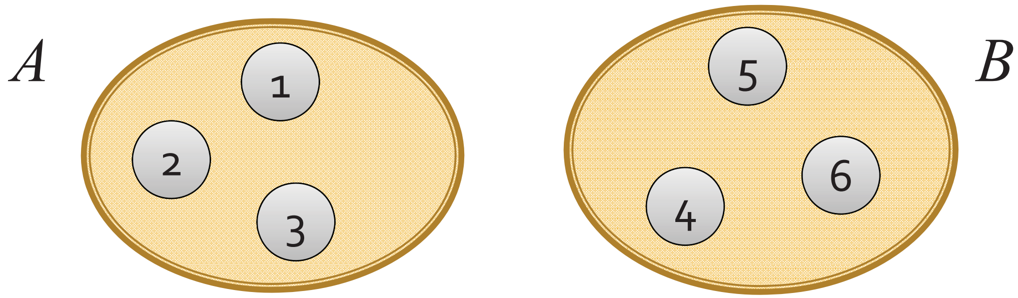
- Issues with modularity:
  - May not find communities with less than √m links
  - NP-hard to optimize exactly [Brandes et al. '07]

Method4: Graph partitioning

- Undirected graph G(V,E):



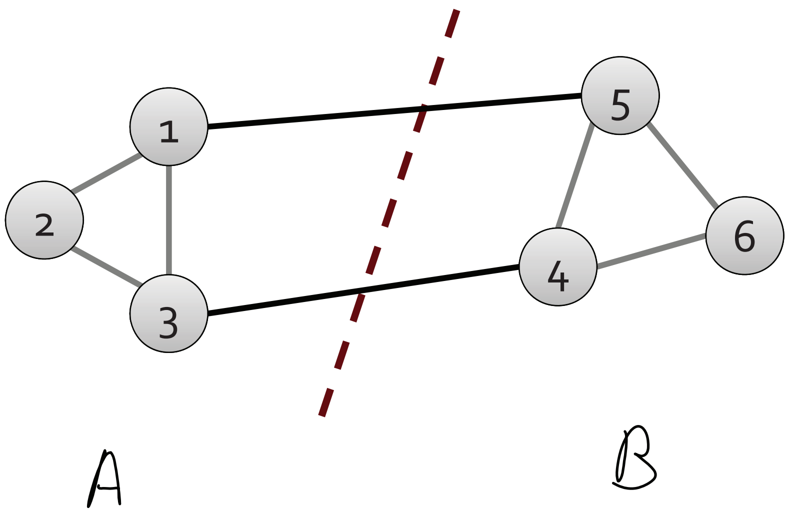
- Bi-partitioning task:
  - Divide vertices into two disjoint groups (A,B)



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

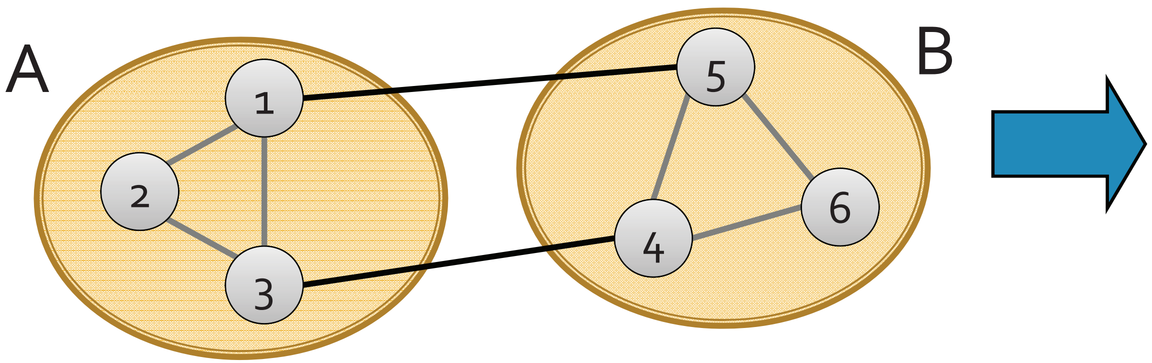
Graph partitioning

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



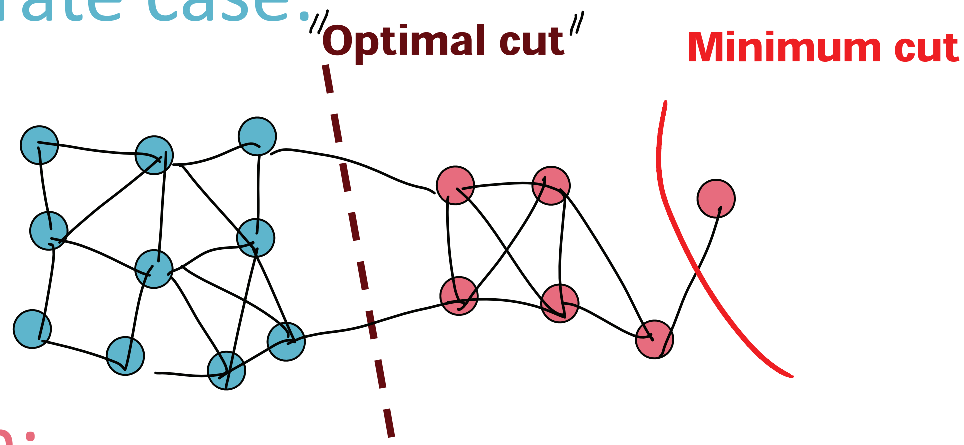
$cut(A,B) = 2$

Graph Cut Criterion

- Criterion: Minimum-cut
  - Minimise weight of connections between groups

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

- Degenerate case:



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

[Shi-Malik]

Graph Cut Criteria

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

$$Ncut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

Vol(A): The total weight of the edges originating from group A.

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

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 $(x_1, \ldots, x_n)$
  - just a label/value of each node of G
- What is the meaning of $A \cdot x$?

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

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

- Entry y_j is a sum of labels x_i of neighbors of j

What is the meaning of Ax?

- j^{th} coordinate of Ax:
  - Sum of the x-values of neighbors of j
  - Make this a new value at node j

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

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

$$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$$

Example: d-regular graph

- Suppose all nodes in G have degree d and G is connected
- What are some eigenvalues/vectors of G?

$Ax = \lambda x$ What is λ? What x?

$x = (1, 1 \ldots 1)$

$A \cdot x = (d, d, d, \ldots d)$

$\Rightarrow \lambda = d$

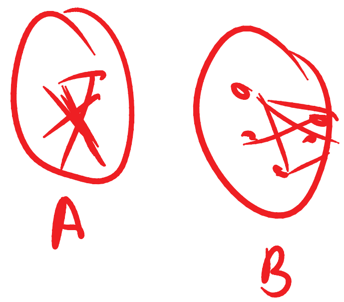
$Ax = \lambda x$

d

$(1, \ldots 1)$ $(1, 1, \ldots 1)$

Example: Graph on 2 components

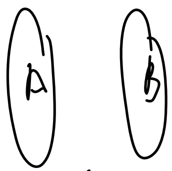
- What if G is not connected?
  - Say G has 2 components, each d-regular



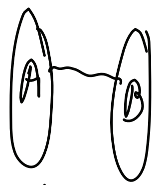
- What are some eigenvectors?
  - $x =$ Put all 1s on A and 0s on B or vice versa

$x' = (\underbrace{0, 0, 0 \ldots 0}_{A}, \underbrace{1, 1, 1, 1}_{B})$ $Ax = (0, 0, 0, d, d \ldots)$

$x'' = (1, \ldots 1, 0 \ldots 0)$ $Ax = (d \ldots d, 0 \ldots 0)$



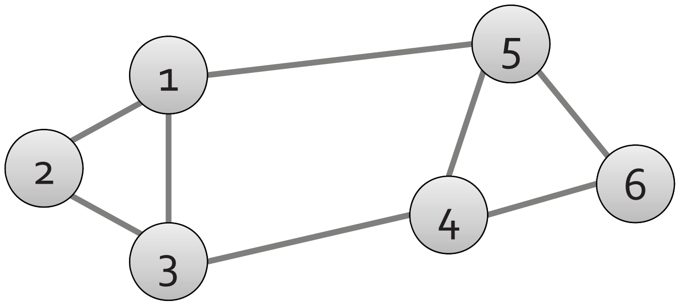
2 dim



$\lambda_1 - \lambda_2 \approx 0$

Matrix Representations

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

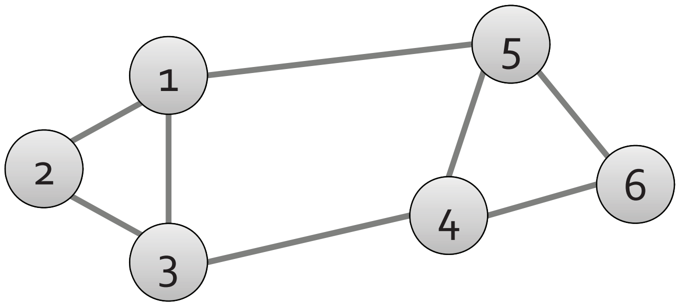


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

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

Matrix Representations (continued)

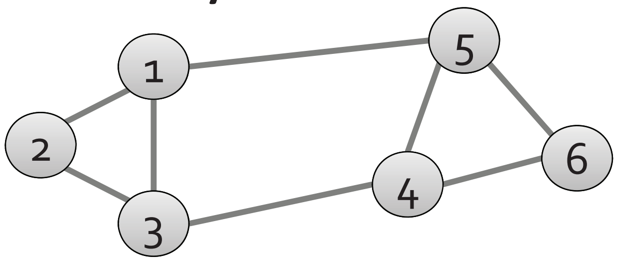
- Degree matrix (D):
  - $n \times n$ diagonal matrix
  - $D=[d_{ii}]$, d_{ii} = degree of node i



	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

$$L = D - A$$

- What is trivial eigenvector/ eigenvalue?

$x = (1, 1, 1, 1 \ldots 1)$

$Lx = 0 \cdot x$ $\lambda = 0$

- Important properties:
  - Eigenvalues are non-negative real numbers
  - Eigenvectors are real and orthogonal

λ_2 as optimization problem

- For symmetric matrix M:

$$\lambda_2 = \min_{x} \frac{x^T M x}{x^T x} = x^T M x$$

- What is the meaning of min $x^T L x$ on G?

$$x^T L x = \sum_{(i,j)\in E} (x_i - x_j)^2$$

$$x^T L x = \sum_{i,j=1}^{m} L_{ij} x_i x_j = \sum_{i,j=1}^{m} (D_{ij} - A_{ij}) x_i x_j$$

λ_2 as optimization problem

- What else do we know about x?
  - x is unit vector $\sum x_i^2 = 1$
  - x is orthogonal to 1st eigenvector $(1, \ldots, 1)$ thus:

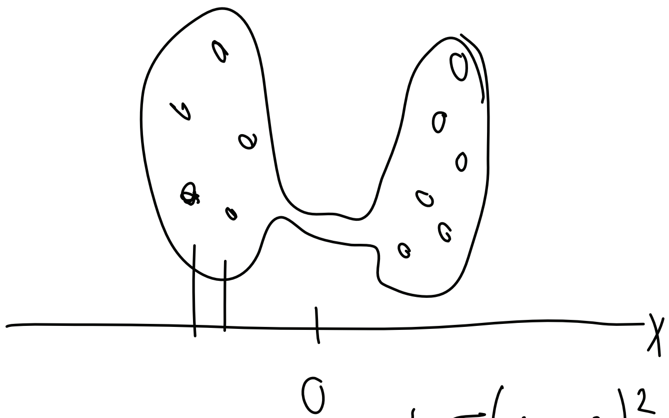
$$\sum x_i \cdot 1 = \sum_i x_i = 0$$

- Then:

$$\lambda_2 = \min_{\substack{\text{All labelings of} \\ \text{nodes so that} \\ \text{sum}(x_i)=0}} \frac{\sum (x_i - x_j)^2}{\underbrace{\sum x_i^2}_{=1}}$$

$\sum x_i^2 = 1$

$\sum x_i = 0$



$$\min \sum_{i,j \in E} (x_i - x_j)^2$$

$\sum x_i = 0$, $\sum x_i^2 = 1$

Find Optimal Cut [Fiedler'73]

- Express partition (A,B) as a vector

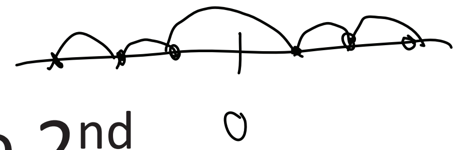
$$x_i = \begin{cases} +1 \text{ if } i \in A \\ -1 \text{ if } i \in B \end{cases}$$

- We can minimize the cut of the partition by finding a non-trivial vector x that minimizes:

$$\min \quad f(x) = \sum_{(i,j)\in E} (x_i - x_j)^2$$

Rayleigh Theorem

$$\min_{x} f(x) = \sum_{(i,j)\in E} (x_i - x_j)^2 = x^T L x$$



- The minimum value is given by the 2nd smallest eigenvalue λ_2 of the Laplacian matrix L

- The optimal solution for x is given by the corresponding eigenvector λ_2, referred as the Fiedler vector

So far...

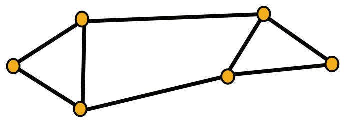
- How to define a "good" partition of a graph?
  - Minimise a given graph cut criterion
- How to efficiently identify such a partition?
  - Approximate using information provided by the eigenvalues and eigenvectors of a graph
- Spectral Clustering

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

- 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

- 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

x

  - Map vertices to corresponding components of λ_2

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

How do we now find the clusters?

Spectral Partitioning (continued)

- 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, mean or median value
  - More expensive approaches:
    - Attempt to minimise normalized cut criterion in 1-dimension

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

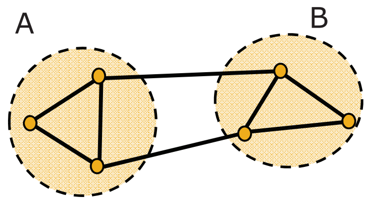
Split at 0:

Cluster A: Positive points

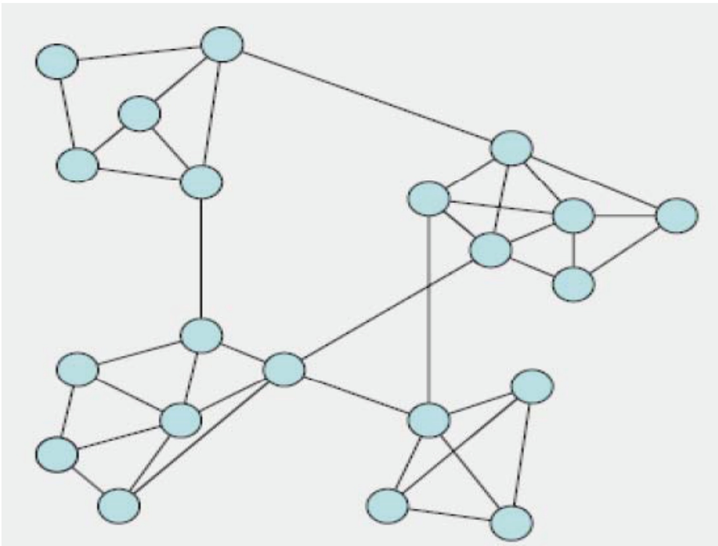
Cluster B: Negative points

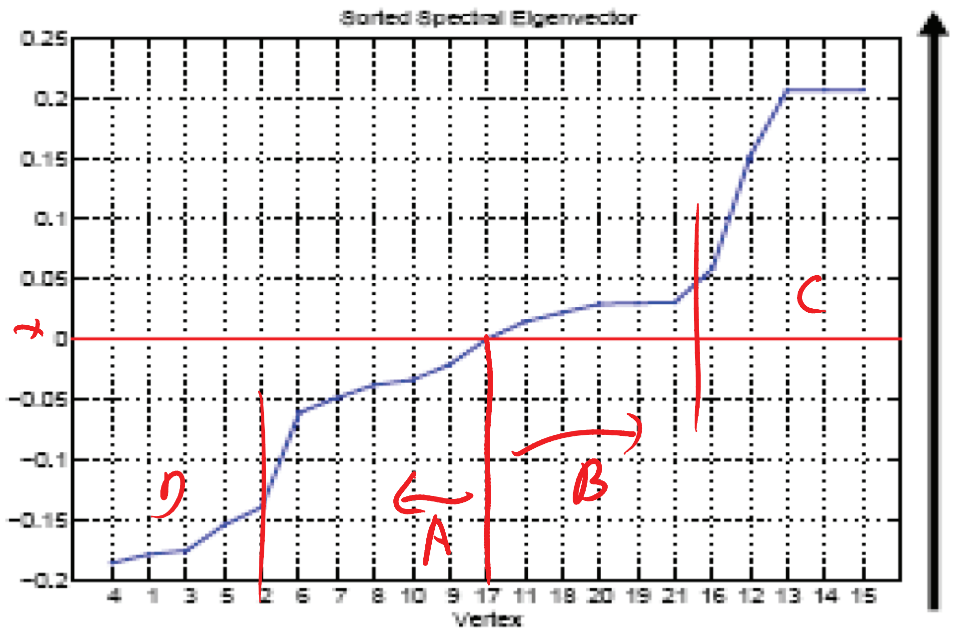
1	0.3
2	0.6
3	0.3

4	-0.3
5	-0.3
6	-0.6



Example: Spectral partitioning





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…

k-Eigenvector Clustering

- k-eigenvector Algorithm [Ng et al.,'01]
  - Pre-processing:
    - Construct the scaled adjacency matrix A':

$$A' = D^{-1/2} A D^{-1/2}$$

  - Decomposition:
    - Find the eigenvalues and eigenvectors of A'
    - Build embedded space from the eigenvectors corresponding to the k largest eigenvalues
  - Grouping:
    - Apply k-means to reduced $n \times k$ space to get k clusters

$A' \Rightarrow \lambda = \{..\}$

$X =$

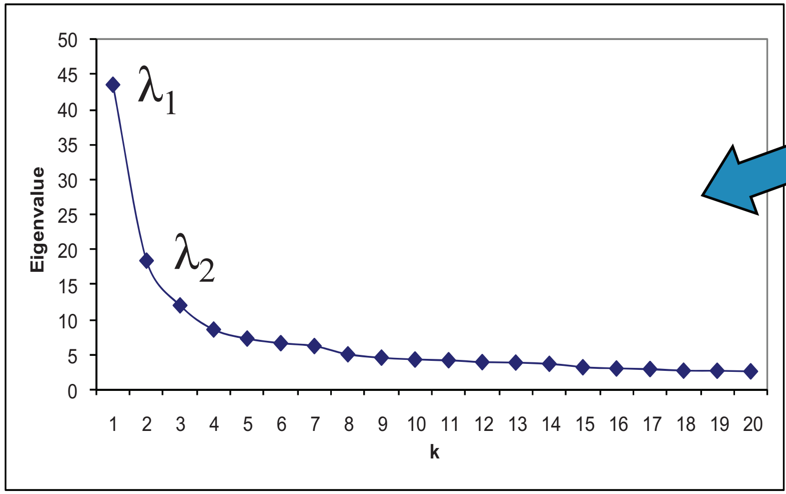
k

Why use multiple eigenvectors?

- Approximates the optimal cut [Shi-Malik, '00]
  - Can be used to approximate the 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 k orthogonal basis vectors
- NB: Multiple eigenvectors prevent instability due to information loss

How to select k?

- Eigengap:
  - The difference between two consecutive eigenvalues
- Most stable clustering is generally given by the value k that maximises eigengap: $\Delta_k = |\lambda_k - \lambda_{k-1}|$
- Example:

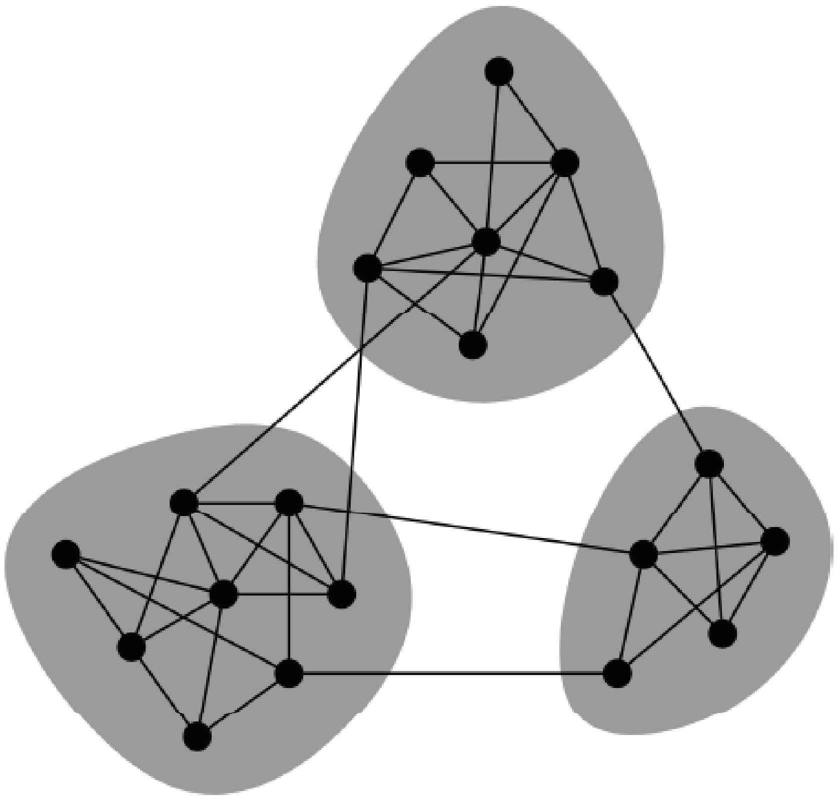


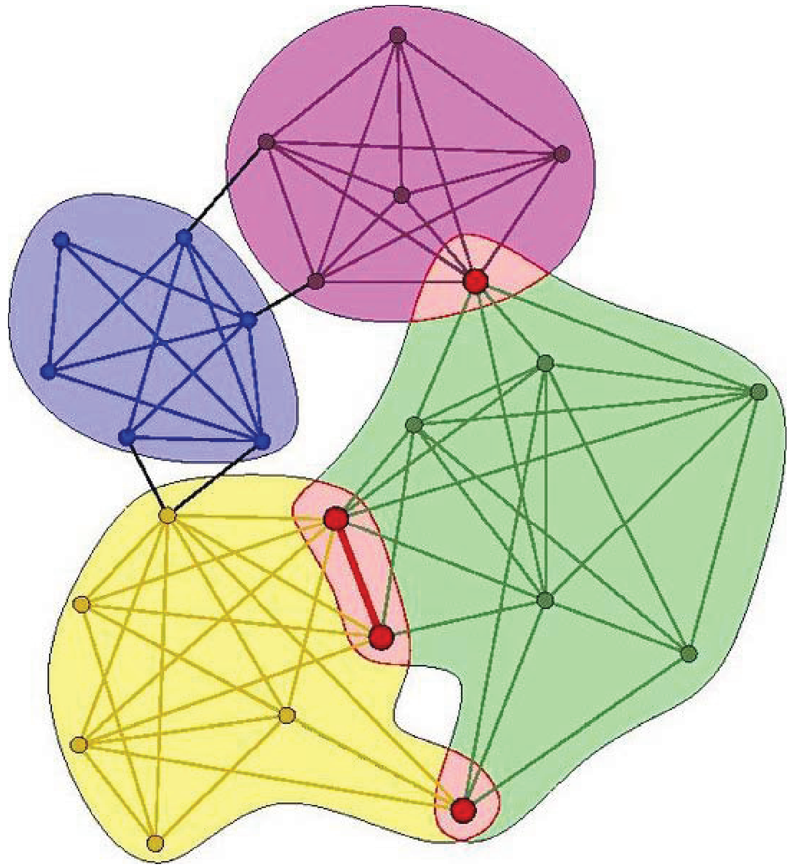
$\max \Delta_k = |\lambda_2 - \lambda_1|$

$\Rightarrow$ Choose k=2

Overlapping communities

- Non-overlapping vs. overlapping communities

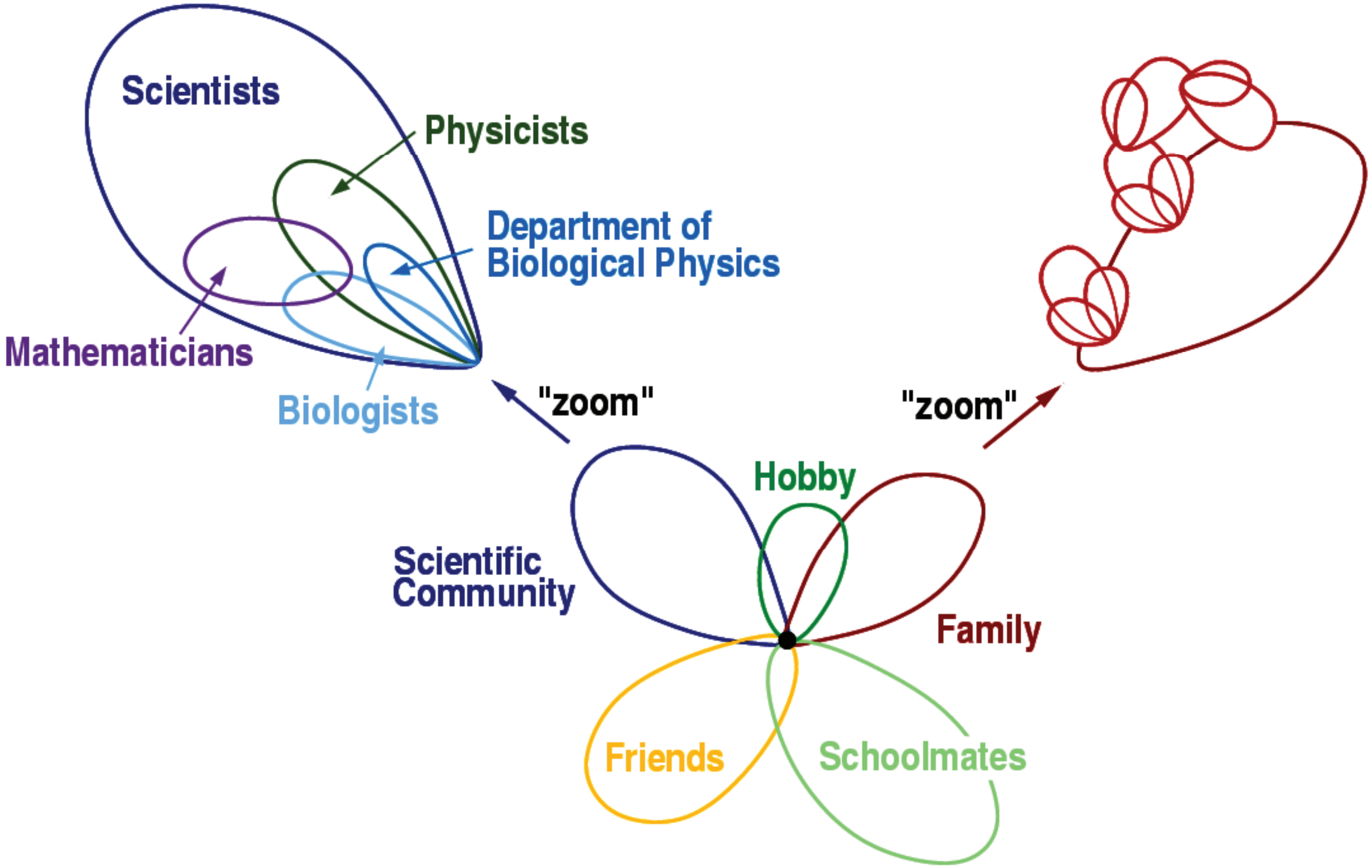




[Palla et al., '05]

Overlaps of social circles

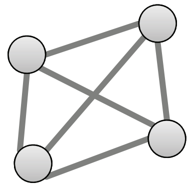
- A node belongs to many social circles



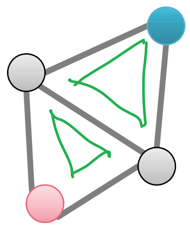
[Palla et al., '05]

Clique Percolation Method (CPM)

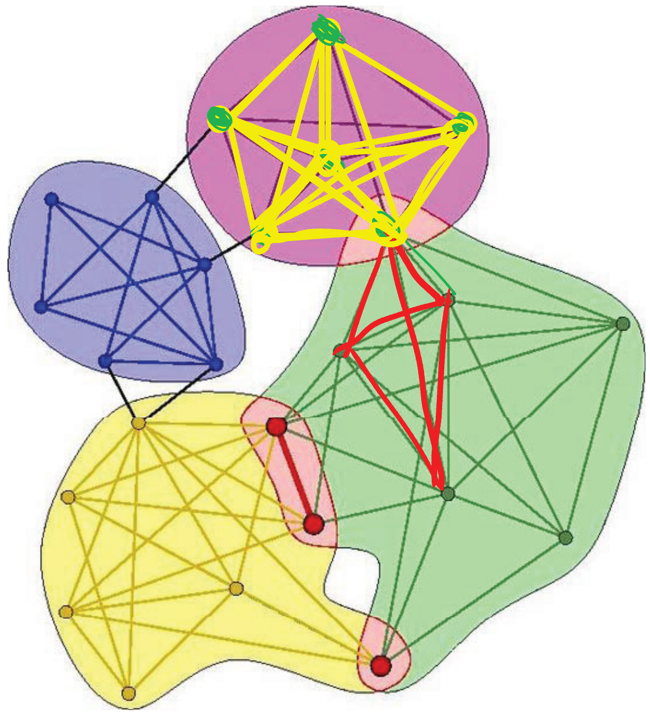
- Two nodes belong to the same community if they can be connected through adjacent k-cliques:
  - k-clique:
    - Fully connected graph on k nodes
  - Adjacent k-cliques:
    - overlap in k-1 nodes
- k-clique community
  - Set of nodes that can be reached through a sequence of adjacent k-cliques



4-clique



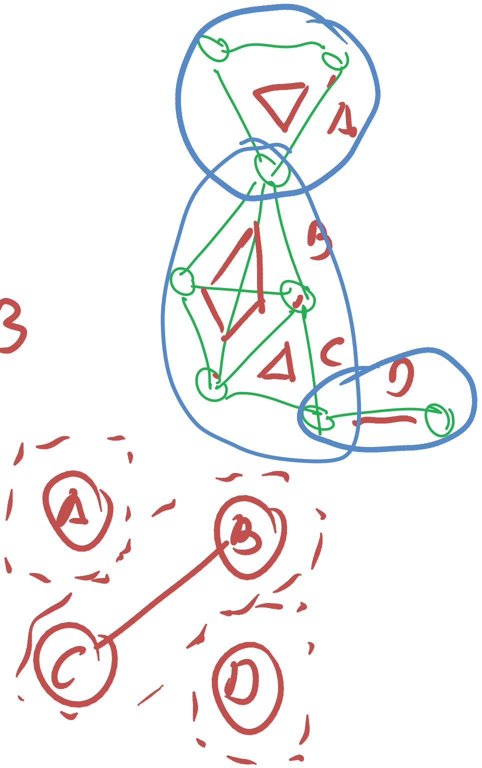
adjacent 3-cliques



Method5: CPM: Steps

- Clique Percolation Method:
  - Find maximal-cliques (not k-cliques!)
  - Clique overlap matrix:
    - Each clique is a node
    - Connect two cliques if they overlap in at least k-1 nodes
  - Communities:
    - Connected components of the clique overlap matrix

$k = 3$

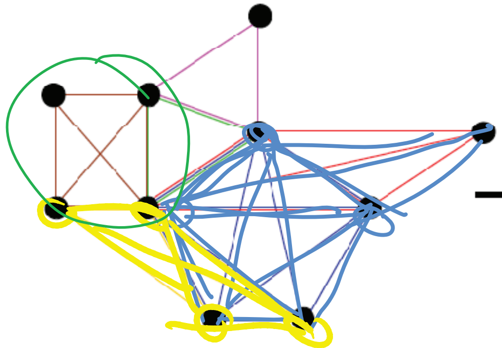


In practice $k = 3, 4, 5$

- How to set k?
  - Set k so that we get the "richest" (most widely distributed cluster sizes) community structure

CPM method: Example

- Start with graph and find maximal cliques
- Create clique overlap matrix
- Threshold the matrix at value *k-1*
  - If $a_{ij}<k-1$ set 0
- Communities are the connected components of the thresholded matrix



(1) Graph

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

(2) Clique overlap matrix

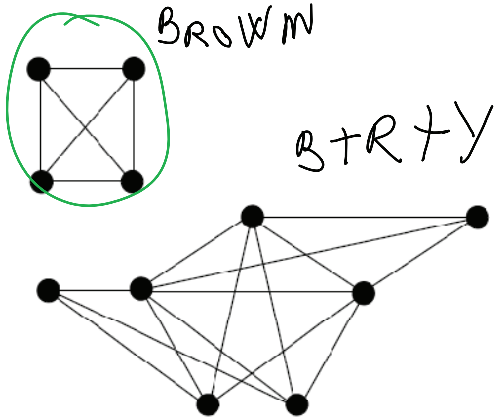
k=4

B→R→Y

Br

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

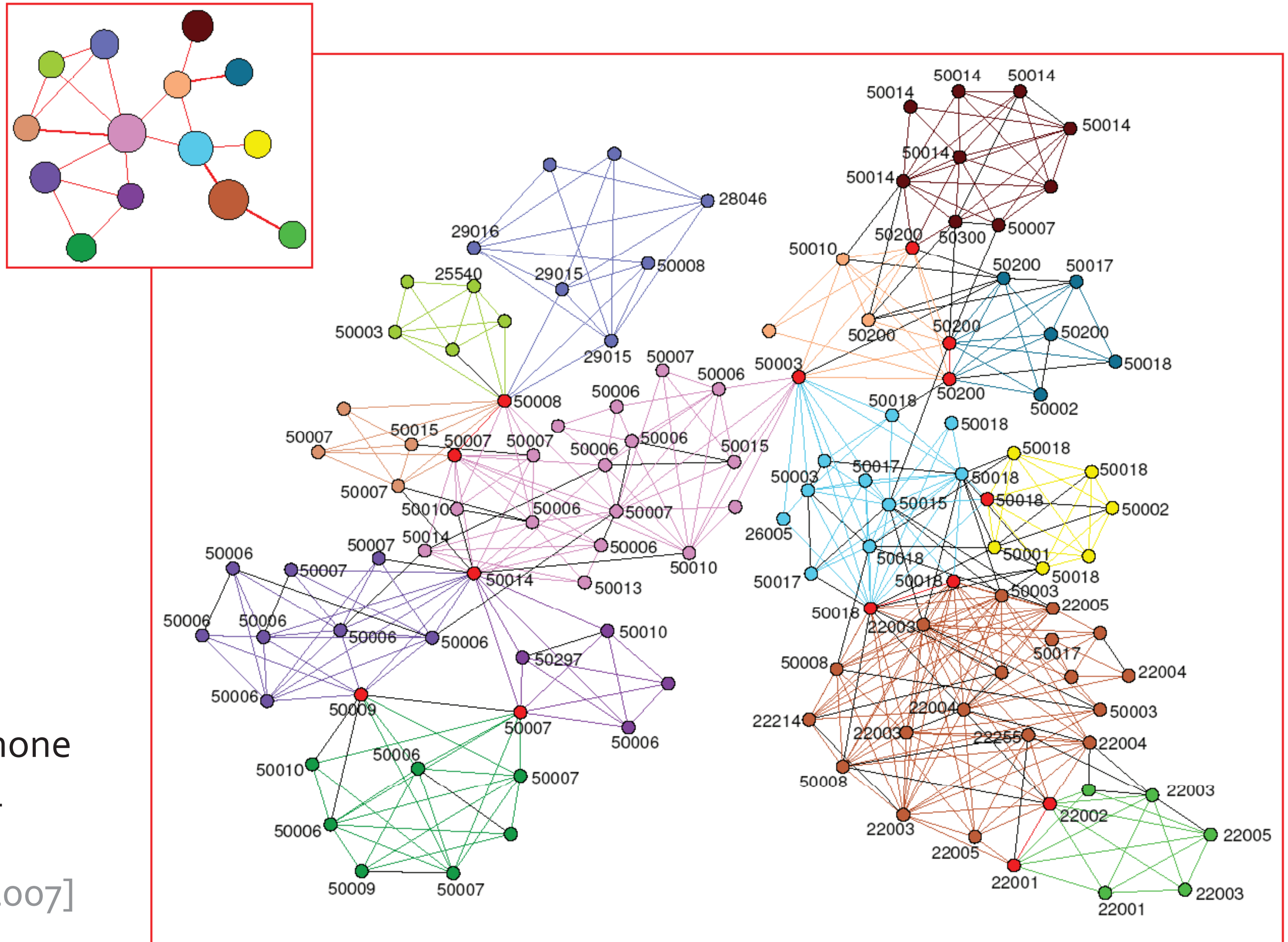
(3) Thresholded matrix at k=4



(4) Communities (connected components)

[Palla et al., '07]

Example: Phone call network

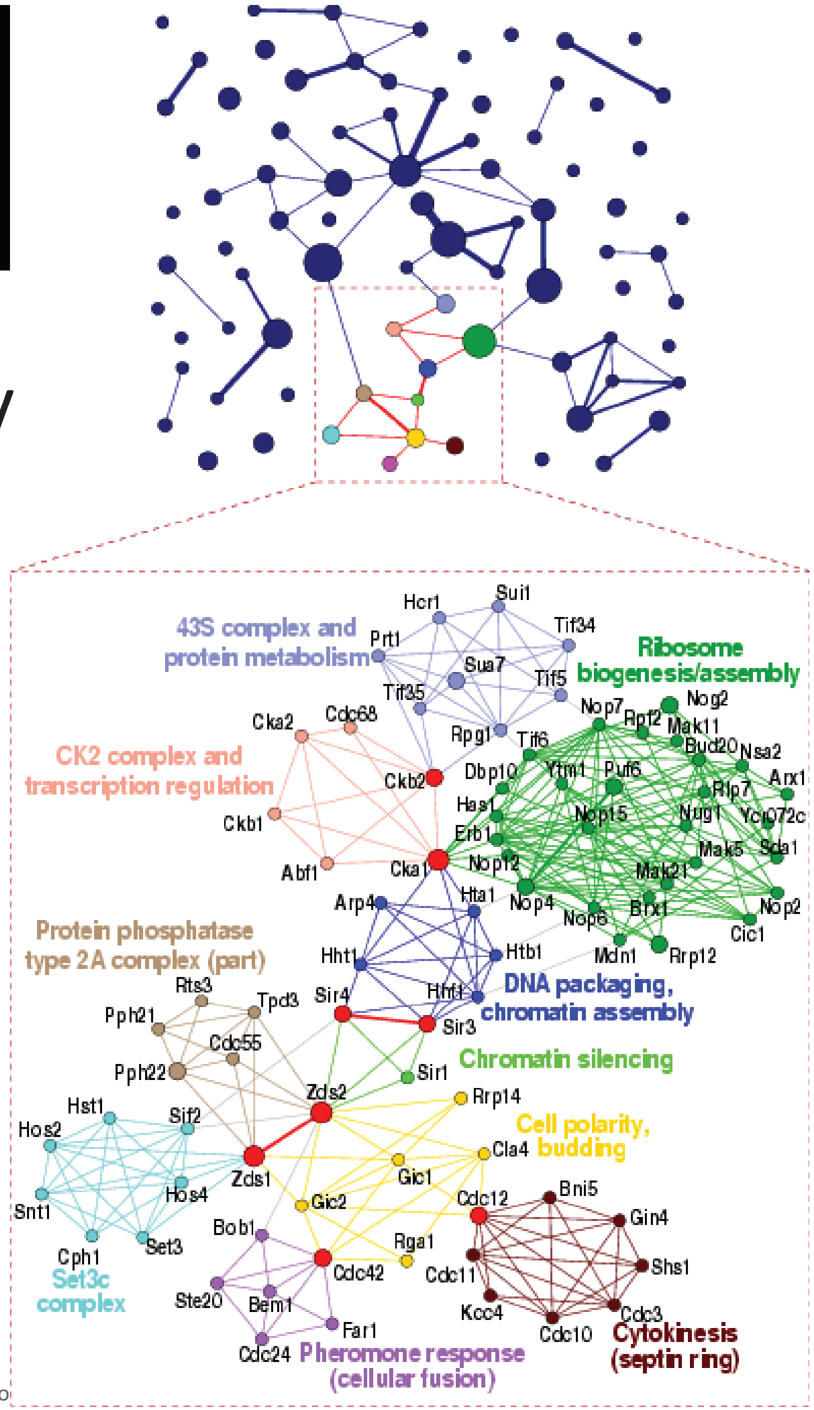


Communities in a "tiny" part of a phone calls network of 4 million users
[Barabasi-Palla, 2007]

Example (2)

- Each node is a community
- Nodes are weighted for community size
- Links are weighted for overlap size

- DIP "core" data base of protein interactions (S. cerevisiase, yeast)



	Name	Overlap	Dir	Weight	Dyn	NoPar	MDim	Incr	Multip	Complexity	BESn	BESm	Year	Ref
Feature Distance	Evolutionary*				✓			✓		$\mathcal{O}(n^2)$	5k	?	2006	[20]
	MSN-BD			✓					✓	$\mathcal{O}(n^2ck)$	6k	3M	2006	[21]
	SocDim	✓		✓			✓			$\mathcal{O}(n^2 \log n)*$	80k	6M	2009	[23]
	PMM			✓			✓			$\mathcal{O}(mn^2)$	15k	27M	2009	[24]
	MRGC		✓		✓		✓		✓	$\mathcal{O}(mD)$	40k	?	2007	[22]
	Infinite Relational					✓	✓			$\mathcal{O}(n^{2c}D)$	160	?	2006	[25]
	Find-Tribes				✓				✓	$\mathcal{O}(mnK^2)$	26k	100k	2007	[26]
	AutoPart		✓			✓		✓		$\mathcal{O}(mk^2)$	75k	500k	2004	[29]
	Timefall				✓	✓			✓	$\mathcal{O}(mk)$	7.5M	53M	2008	[31]
	Context-specific Cluster Tree					✓			✓	$\mathcal{O}(mk)$	37k	367k	2008	[30]
Internal Density	Modularity			✓						$\mathcal{O}(mk \log n)$	400k	2.5M	2004	[12]
	Directed modularity		✓	✓						$\mathcal{O}(n^2 \log n)$	50	?	2008	[55]
	External Optimization		✓	✓						$\mathcal{O}(n^2 \log n)$	27k	?	2005	[56]
	Local modularity			✓				✓		$\mathcal{O}(n^2)$	400k	2.5M	2005	[57]
	Modularity Unfolding			✓						$\mathcal{O}(mk)$	118M	1B	2008	[58]
	Multislice modularity		✓	✓	✓		✓		✓	$\mathcal{O}(mkD)$	2k	?	2010	[59]
	MetaFac				✓		✓			$\mathcal{O}(mnD)$	?	2M	2009	[60]
	Variational Bayes		✓			✓				$\mathcal{O}(mk)$	115	613	2008	[61]
	$LA \to IS^2*$	✓	✓							$\mathcal{O}(mk + n)$	16k	?	2005	[62]
	Local Density		✓			✓		✓		$\mathcal{O}(nK \log n)$	108k	330k	2005	[63]
Bridge	Edge Betweenness		✓							$\mathcal{O}(m^2n)$	271	1k	2002	[4]
	CONGO*	✓								$\mathcal{O}(n \log n)$	30k	116k	2008	[95]
	L-Shell	✓						✓		$\mathcal{O}(n^3)$	77	254	2005	[96]
	Internal-External Degree	✓								$\mathcal{O}(n^2 \log n)$	775k	4.7M	2009	[97]
Diffusion	Label Propagation			✓		✓		✓		$\mathcal{O}(m + n)$	374k	30M	2007	[109]
	Node Colouring				✓				✓	$\mathcal{O}(ntk^2)$	2k	?	2007	[110]
	Kirchhoff	✓		✓						$\mathcal{O}(m + n)$	115	613	2004	[111]
	Communication Dynamic	✓	✓		✓			✓		$\mathcal{O}(mnt)$	160k	530k	2008	[112]
	GuruMine		✓		✓					$\mathcal{O}(TAn^2)$	217k	212k	2008	[8]
	DegreeDiscountIC		✓							$\mathcal{O}(k \log n + m)$	37k	230k	2009	[113]
	MMSB	✓	✓							$\mathcal{O}(nk)$	871	2k	2007	[114]
Close	Walktrap									$\mathcal{O}(mn^2)$	160k	1.8M	2006	[131]
	DOCS	✓								?	325k	1M	2009	[132]
	Infomap		✓	✓						$\mathcal{O}(m \log^2 n)$	6k	6M	2008	[133]
Structure	K-Clique	✓								$\mathcal{O}(m^{\frac{\ln m}{10}})$	20k	127k	2005	[3]
	S-Plexes Enumeration									$\mathcal{O}(mn)$	?	?	2009	[142]
	Bi-Clique	✓							✓	$\mathcal{O}(m^2)$	200k	500k	2008	[141]
	EAGLE	✓	✓	✓						$\mathcal{O}(3^{\frac{n}{3}})$	16k	31k	2009	[143]
Link	Link modularity	✓		✓					✓	$\mathcal{O}(2mk \log n)$	20k	127k	2009	[151]
	Link Jaccard*	✓		✓					✓	$\mathcal{O}(n\bar{K}^2)$	885k	5.5M	2010	[152]
NoD	Hybrid*	✓	✓	✓		✓				$\mathcal{O}(nk\bar{K})$	325k	1.5M	2010	[154]
	Multi-relational Regression			✓			✓			?	?	?	2005	[155]

Table 1: Resume of the community discovery methods.