

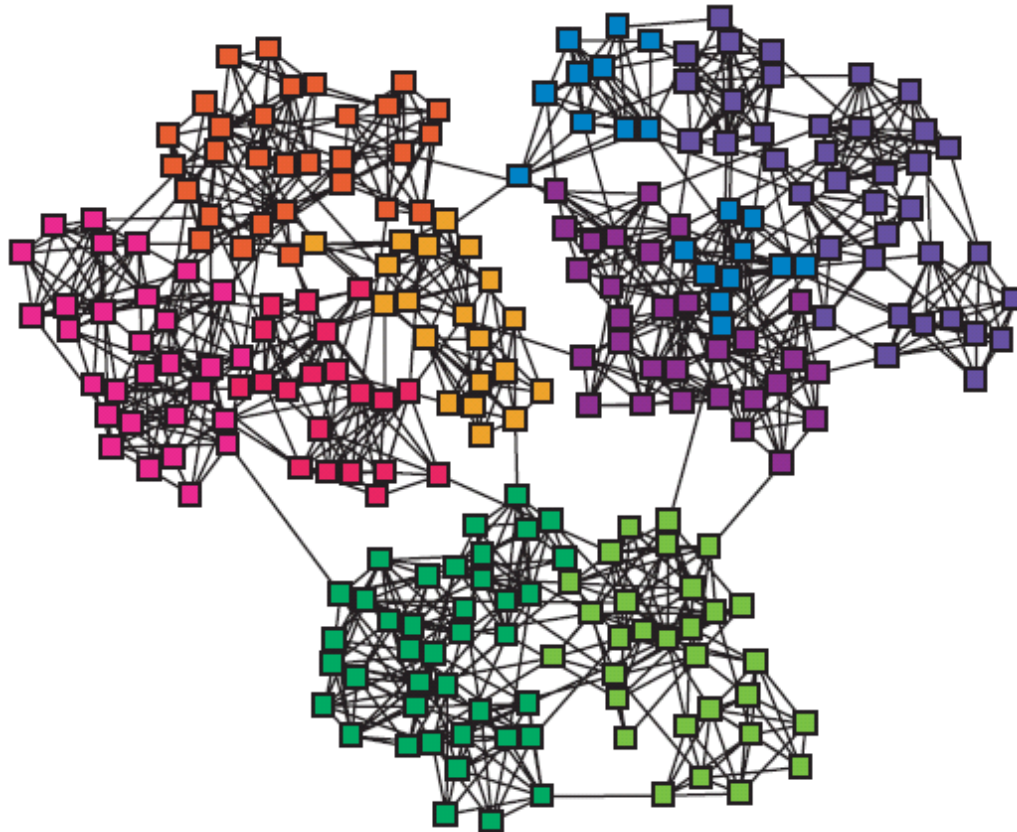
Mining Social-Network Graphs

Analysis of Large Graphs: Community Detection

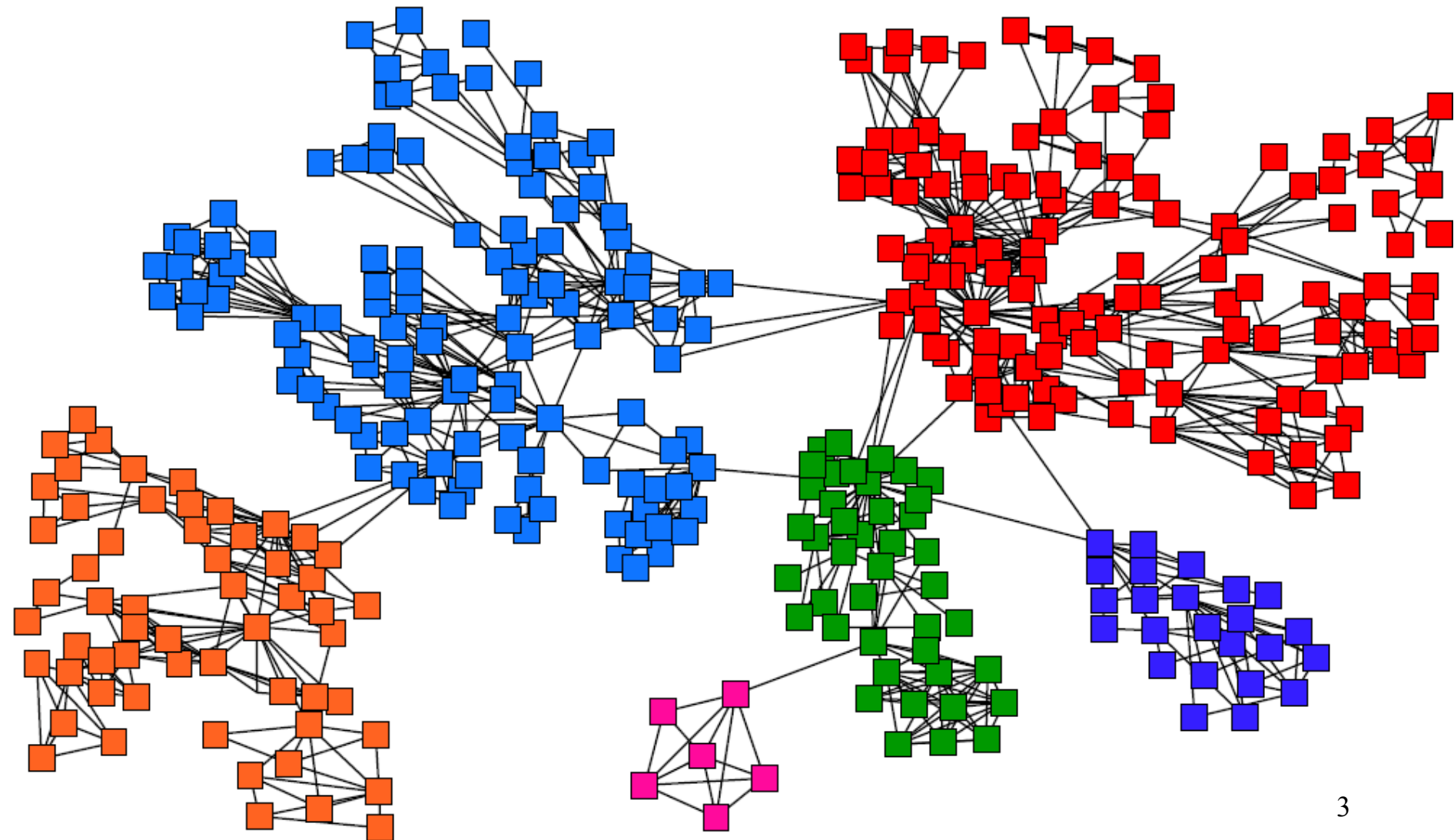
With slide contributions from J. Leskovec, Anand Rajaraman, Jeffrey D. Ullman

Networks & Communities

- We often think of networks being organized into **modules, cluster, communities:**

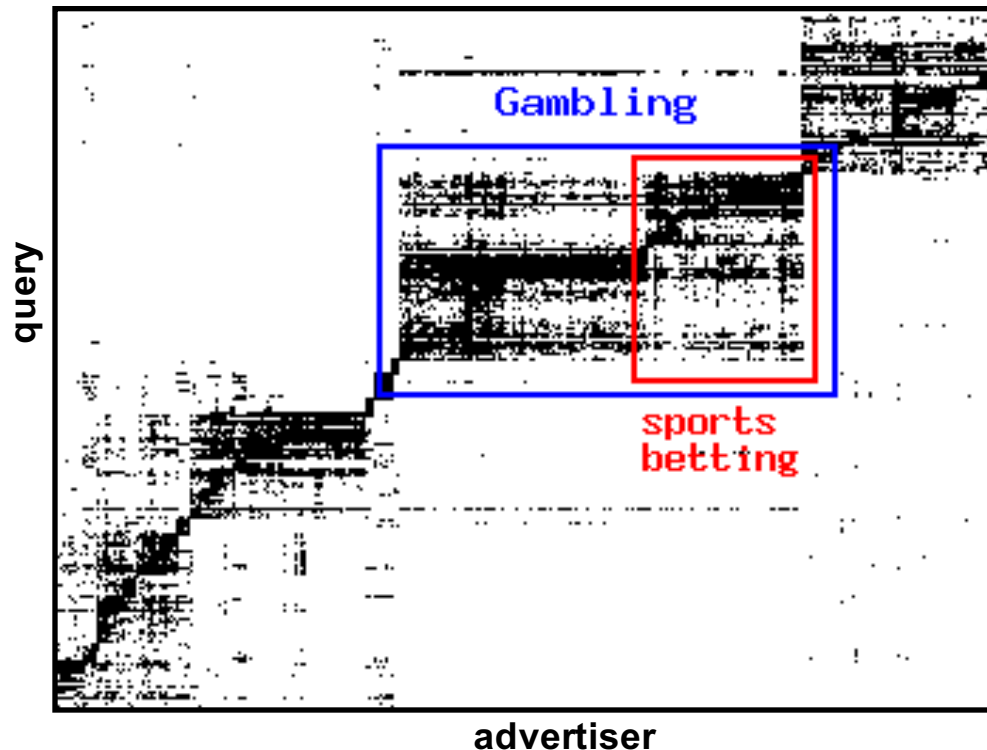


Goal: Find Densely Linked Clusters



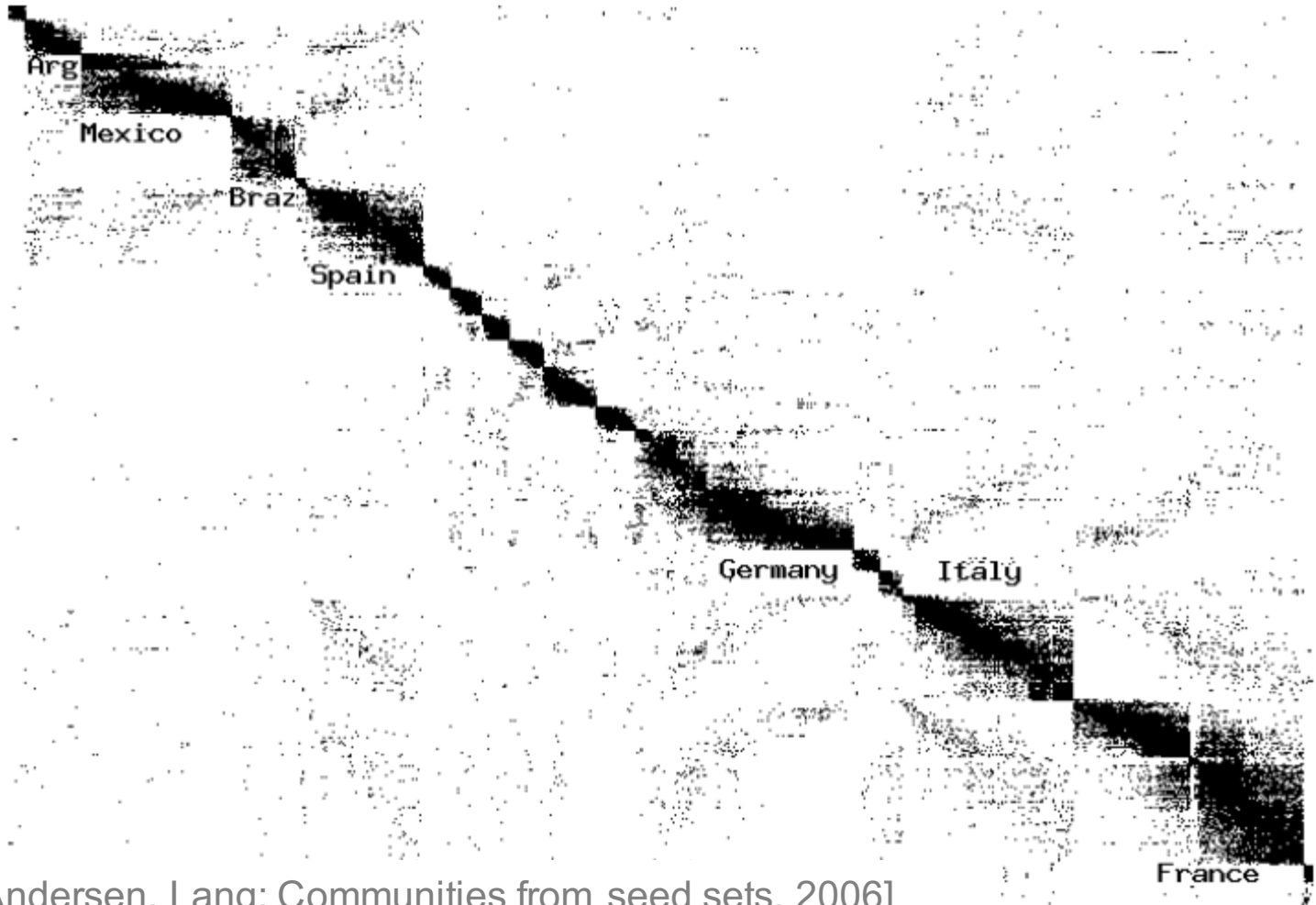
Micro-Markets in Sponsored Search

- Find micro-markets by partitioning the query-to-advertiser graph:



Movies and Actors

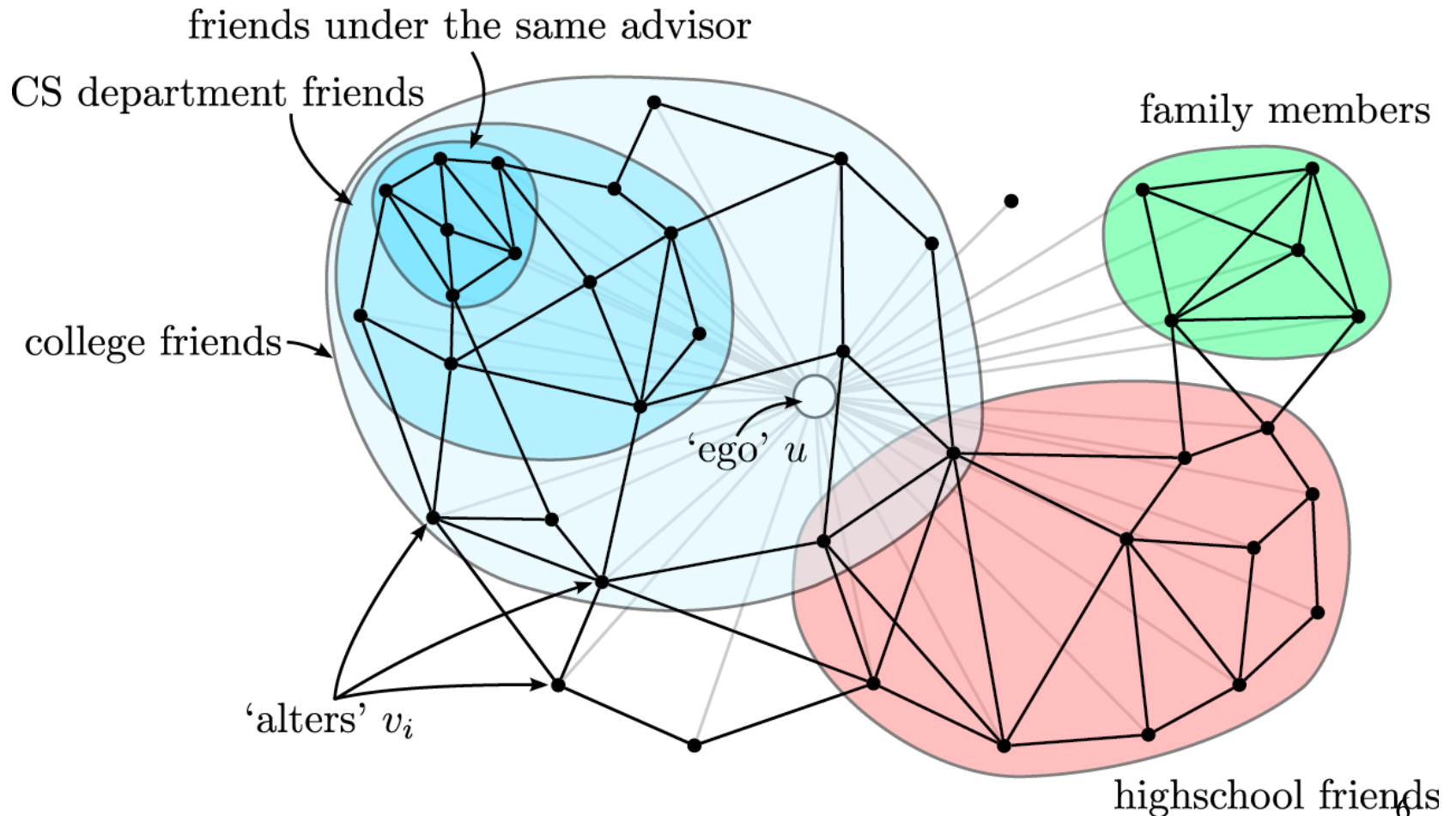
❑ Clusters in Movies-to-Actors graph:



[Andersen, Lang: Communities from seed sets, 2006]

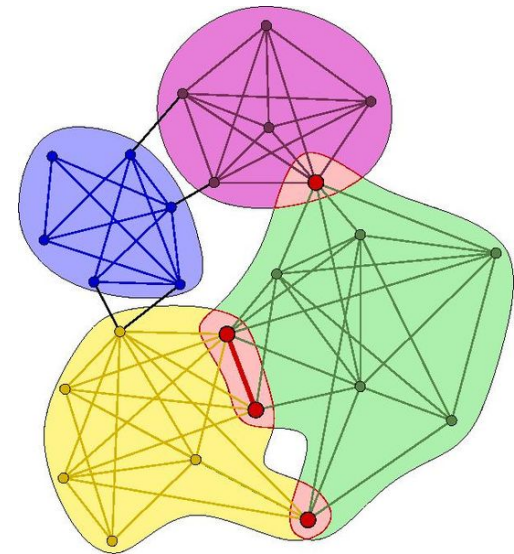
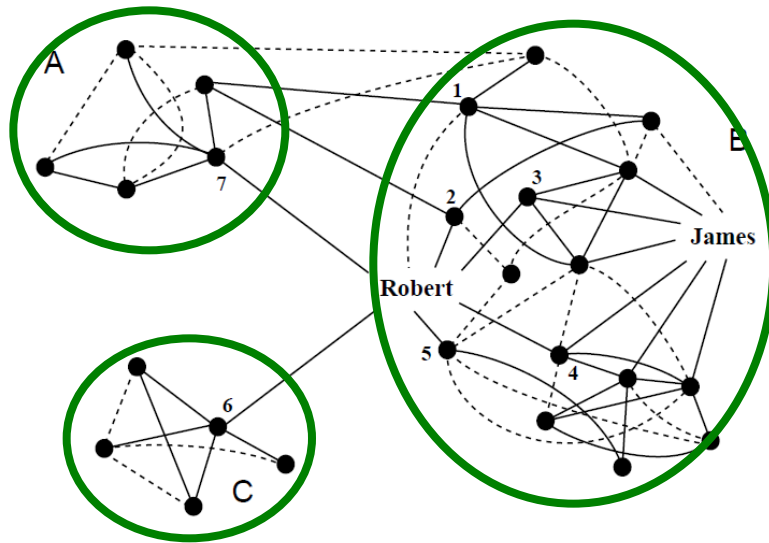
Twitter & Facebook

□ Discovering social circles, circles of trust:



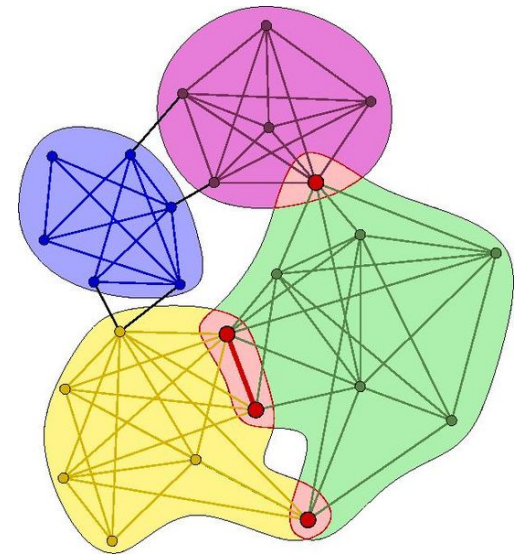
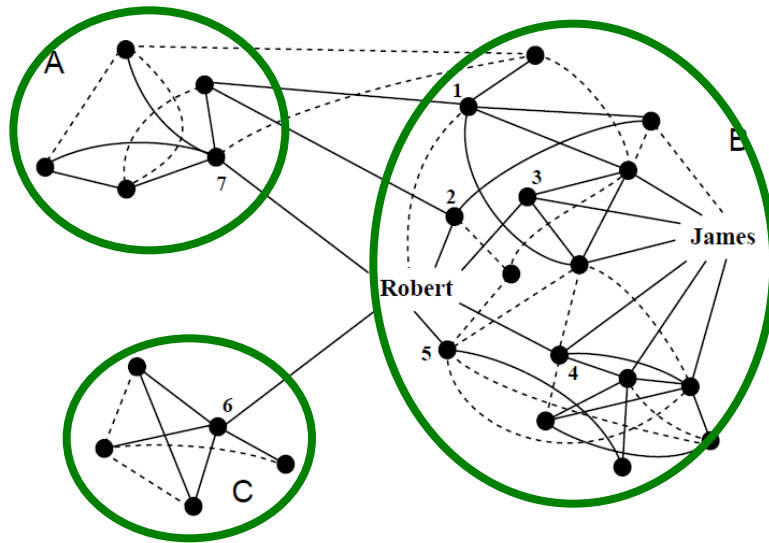
COMMUNITY DETECTION (GRAPH BASICS)

How to find communities?



COMMUNITY DETECTION (ALGORITHMS AND METHODS)

How to find communities?

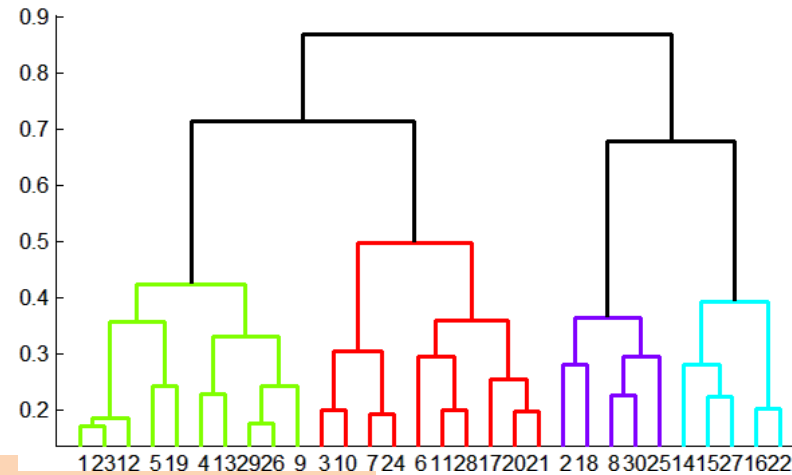


We will work with **undirected** (unweighted) networks 8

Recall: Methods of Clustering

□ Hierarchical:

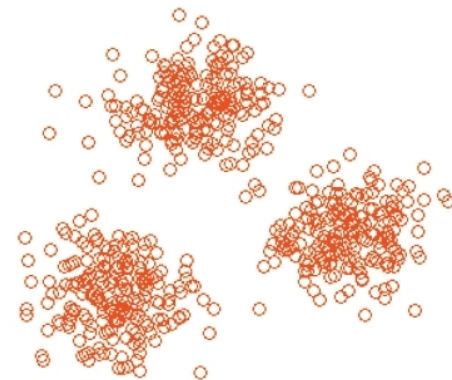
- **Agglomerative** (bottom up):
 - Initially, each point is a cluster
 - Repeatedly combine the two “nearest” clusters into one
 - **Used a distance metric**



- Today: ➤ **Divisive** (top down):
- Start with one cluster and recursively split it

□ Point assignment:

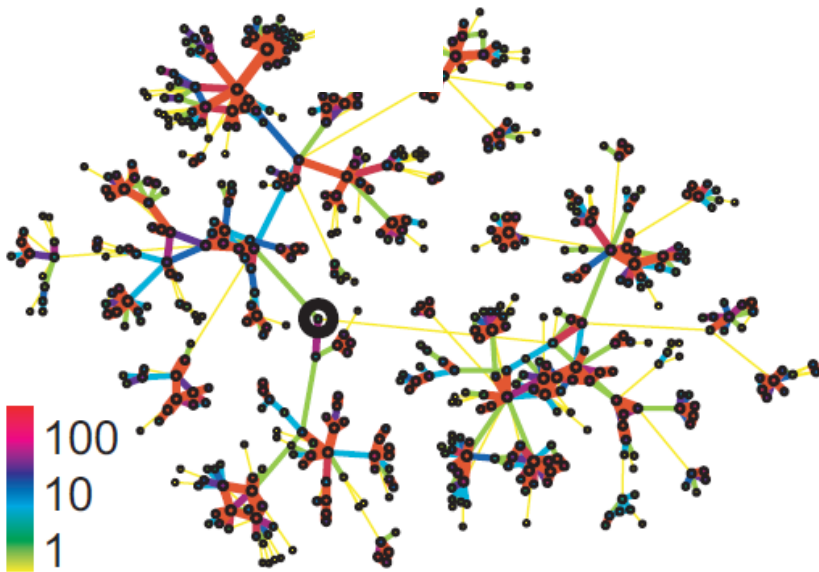
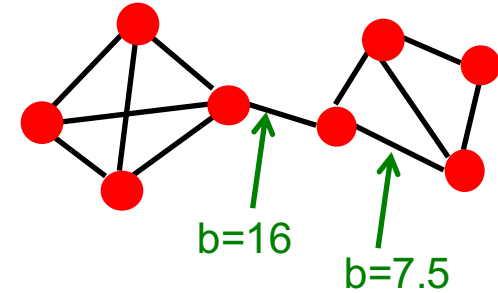
- Maintain a set of clusters
- Points belong to “nearest” cluster
- **Used a distance metric**



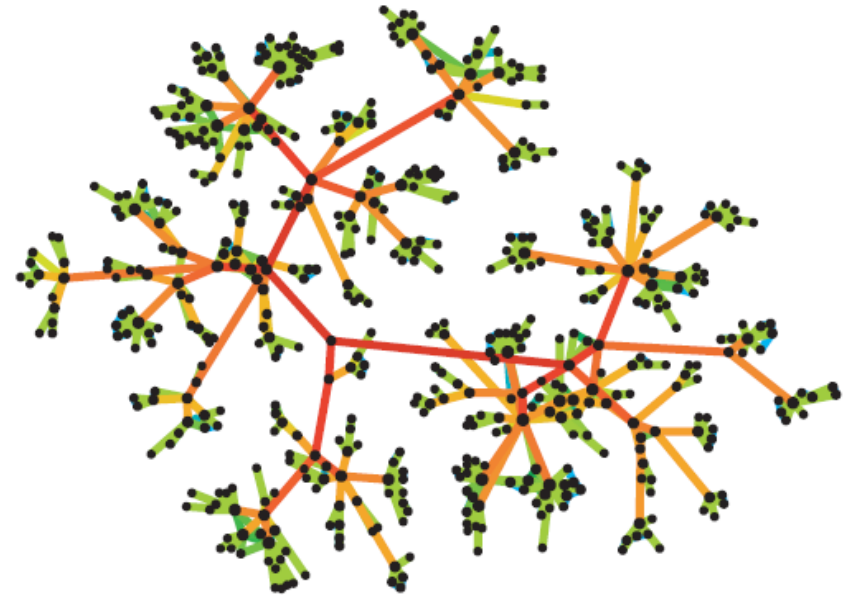
Betweenness Concept

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

□ **Intuition:**

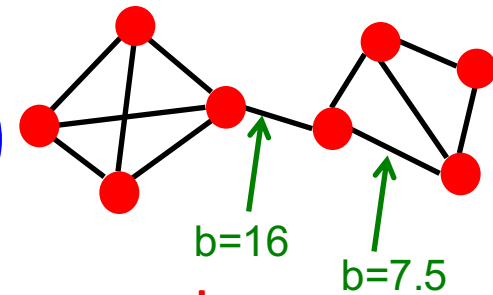


**Edge strengths (call volume)
in a real network**



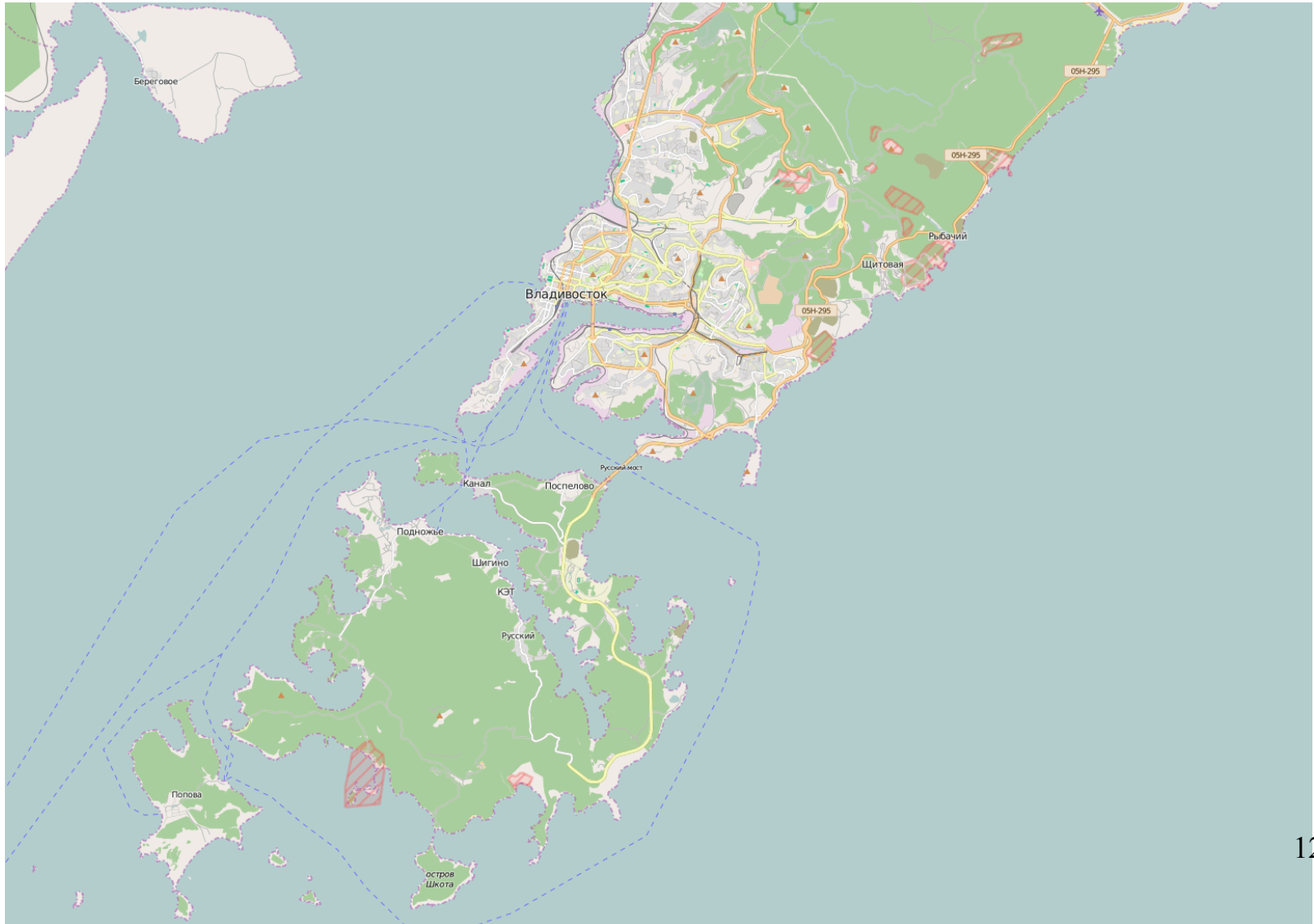
**Edge betweenness
in a real network**

Betweenness Concept (Cont'd)



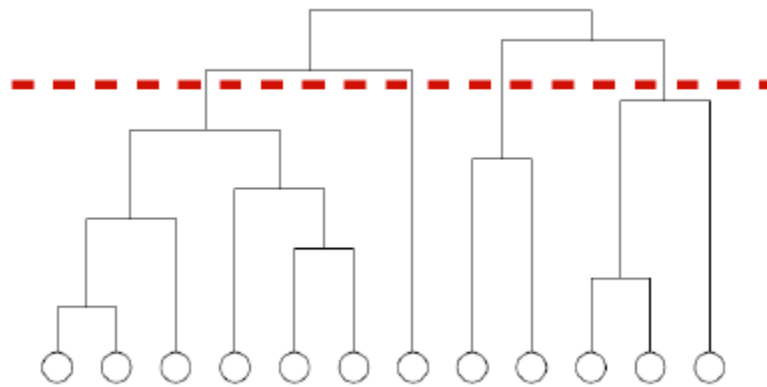
- ❑ Find edges in a social network graph that are **least likely** to be inside a community
- ❑ Betweenness of edge (a, b):
 - number of pairs of nodes x and $y \rightarrow x, y \in C$
 - **edge (a,b) lies on the shortest path between x and y**
- ❑ If there are several shortest paths between x and y , edge (a,b) is credited with **the fraction of those shortest paths** that include edge (a,b)
- ❑ **A high score is bad:** suggests that edge (a,b) runs between two different communities
 - **a and b are in different communities.**

The Russian Bridge



WE NEED TO RESOLVE 2 QUESTIONS

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



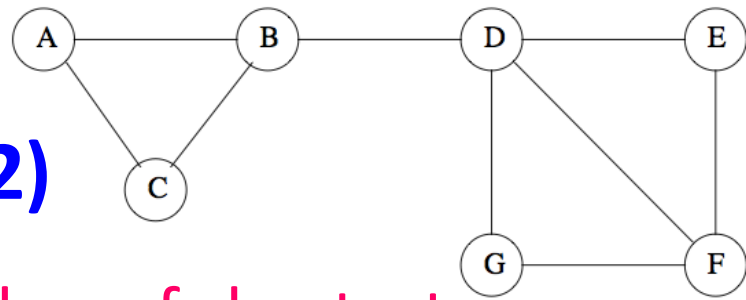
The Girvan-Newman Algorithm

- ❑ Want to **discover communities using divisive hierarchical clustering**
 - Start with one cluster (the social network) and recursively split it
- ❑ **Will do this** based on the notion of edge **betweenness**:
Number of shortest paths passing through the edge
- ❑ **Girvan-Newman Algorithm:**
 - Visits each node X once
 - Computes the number of shortest paths from X to **each of the other nodes** that go through each of the edges
- ❑ **Repeat:**
 - Calculate betweenness of edges
 1. **Thresholding** to remove high betweenness edges, or
 2. **Remove edges** with highest betweenness: **between** communities
- ❑ **Connected components are communities**
- ❑ Gives a hierarchical decomposition of the network.

Girvan-Newman Algorithm (1)

- ❑ Visit each node X once and **compute the number of shortest paths from X to each of the other nodes that go through each of the edges**
- ❑ **1) Perform a breadth-first search (BFS) of the graph, starting at node X**
 - The level of each node in BFS is length of the shortest path from X to that node
 - So edges that go between nodes on the same level can never be part of a shortest path from X
 - **Edges between levels are called DAG edges** (DAG = Directed Acyclic Graph)
 - **Each DAG edge is part of at least one shortest path from root X.**

Girvan-Newman Algorithm (2)



❑ **2) Label each node by the number of shortest paths that reach it from the root node**

➤ **Example: BFS starting from node E, labels assigned**

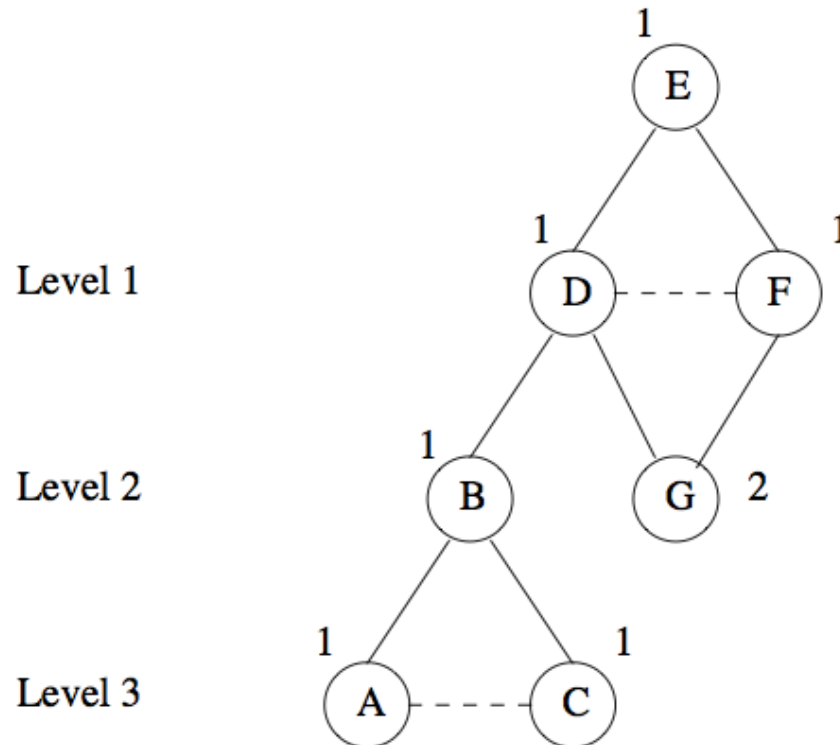
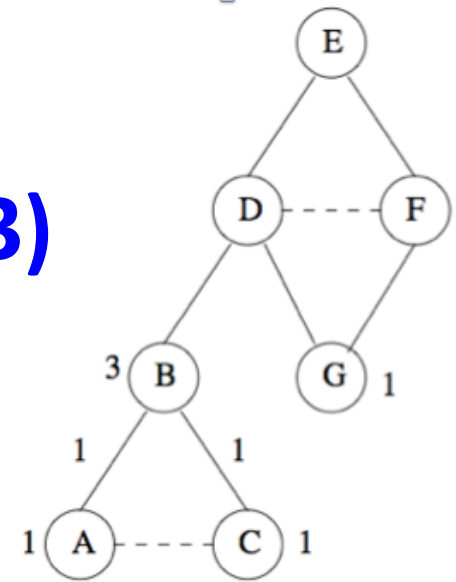


Figure 10.4: Step 1 of the Girvan-Newman Algorithm

Girvan-Newman Algorithm (3)

3) Calculate for each edge e , the sum over all nodes Y (of the fraction) of the shortest paths from the root X to Y that go through edge e

- Compute this sum for nodes and edges, starting from the bottom of the graph
- Each node other than the root node is given a credit of 1
- Each leaf node in the DAG gets a credit of 1
- Each node that is not a leaf gets credit = 1 + sum of credits of the DAG edges from that node to level below
- A DAG edge e entering node Z (from the level above) is given a share of the credit of Z proportional to the fraction of shortest paths from the root to Z that go through e .



Girvan-Newman Algorithm (4)

- Assign node and edge values starting from bottom

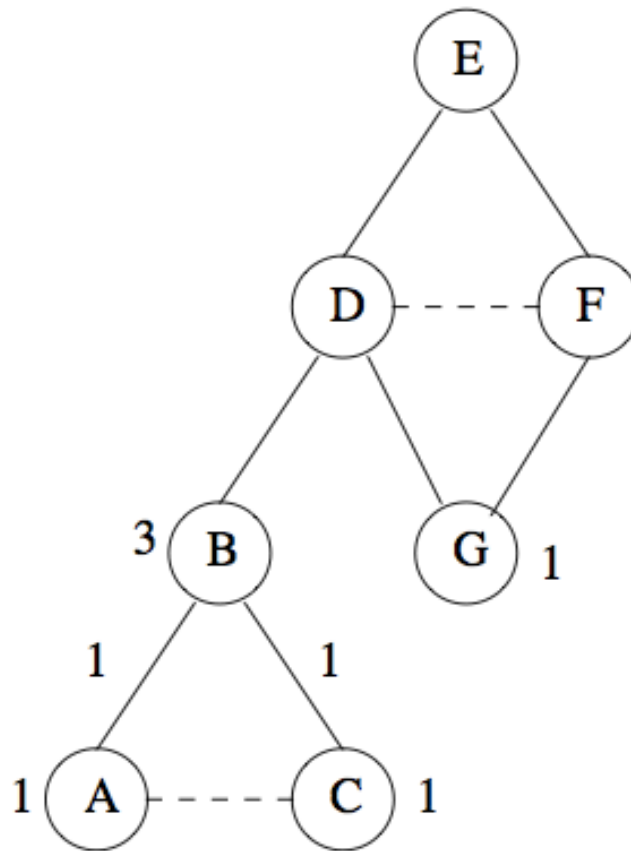
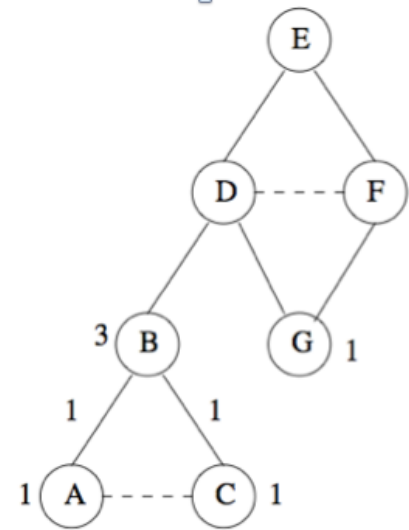


Figure 10.5: Final step of the Girvan-Newman Algorithm – levels 3 and 2

Girvan-Newman Algorithm (5)



Assigning credits:

- ❑ A and C are **leaves**: get credit = 1
- ❑ Each of these nodes **has only one parent**, so their credit=1 is **given to edges** (B,A) and (B,C)
- ❑ At level 2, G is a **leaf**: gets credit = 1
- ❑ B gets credit **1 + credit of DAG edges entering from below**
 $= 1 + 1 + 1 = 3$
- ❑ B has only **one parent**, so edge (D,B) **gets entire credit of node B** = 3
- ❑ **Node G has 2 parents (D and F): how do we divide credit of G between the edges?**

Girvan-Newman Algorithm (6)

❑ In this case, **both D and F have just one shortest path from E to each of those nodes**

- So, **give half credit of node G to each of those edges**
- $\text{Credit} = 1/(1 + 1) = 0.5$

❑ **In general, how we distribute credit of a node to its edges depends on number of shortest paths**

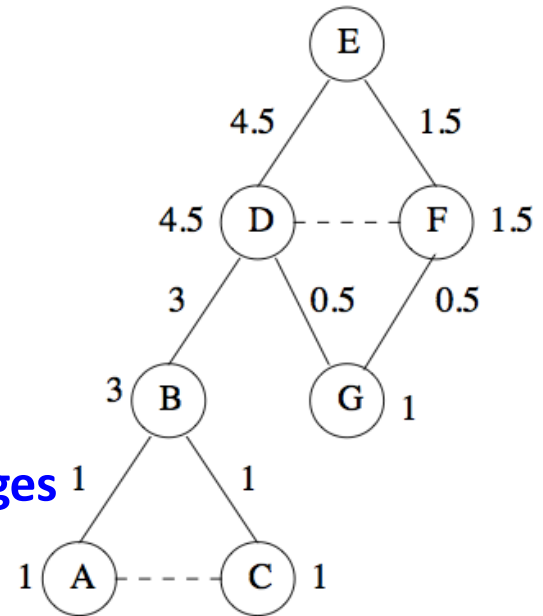
- Say there were 5 shortest paths to D and only 3 to F
- Then credit of edge (D,G) = $5/8$ and credit of edge (F,G) = $3/8$

❑ Node D gets credit = **1 + credits of edges below it** =
 $1 + 3 + 0.5 = 4.5$

❑ Node F gets credit = $1 + 0.5 = 1.5$

❑ D has **only one parent**, so Edge (E,D) gets credit = 4.5 from D

❑ Likewise for F: Edge (E,F) gets credit = 1.5 from F.



Girvan-Newman Algorithm (7): Completion of Credit Calculation starting at node E

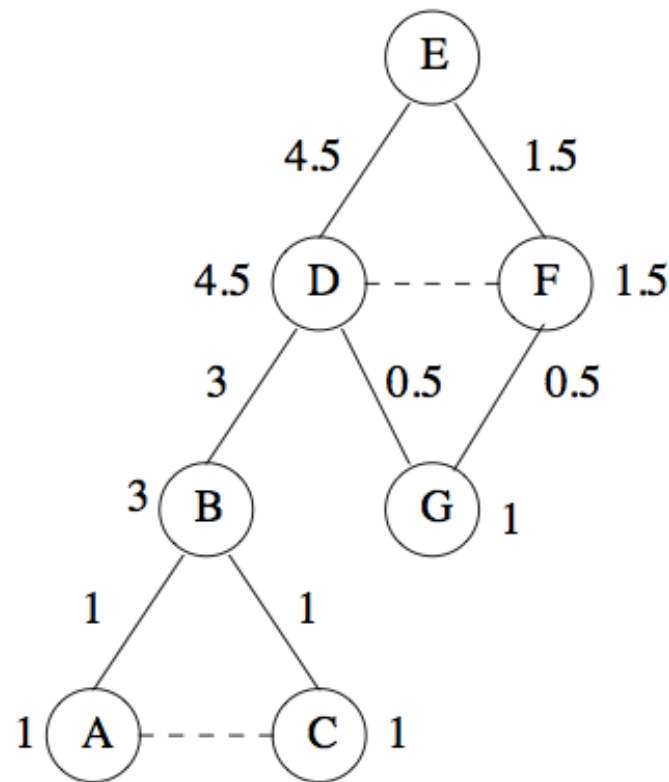


Figure 10.6: Final step of the Girvan-Newman Algorithm – completing the credit calculation

Girvan-Newman Algorithm (8): Overall Betweenness Calculation

- To complete betweenness calculation, must:
 - Repeat this for every node as root
 - Sum the contributions on each edge
 - Divide by 2 to get true betweenness
 - since every shortest path will be counted twice, once for each of its endpoints.

Using Betweenness to Find Communities: Clustering

- ❑ **Betweenness scores for edges of a graph behave something like a distance metric**
 - Not a true distance metric
- ❑ **Could cluster by taking edges in increasing order of betweenness and adding to graph one at a time**
 - At each step, connected components of graph form clusters
- ❑ **Girvan-Newman: Start with the graph and all its edges and remove edges with highest betweenness**
 - Continue until graph has broken into suitable number of connected components
 - **Divisive hierarchical clustering** (top down)
 - Start with one cluster (the social network) and recursively split it.

Using Betweenness to Find Communities (2)

- (B,D) has highest betweenness (12)
- Removing edge would give natural communities we identified earlier: {A,B,C} and {D,E,F,G}

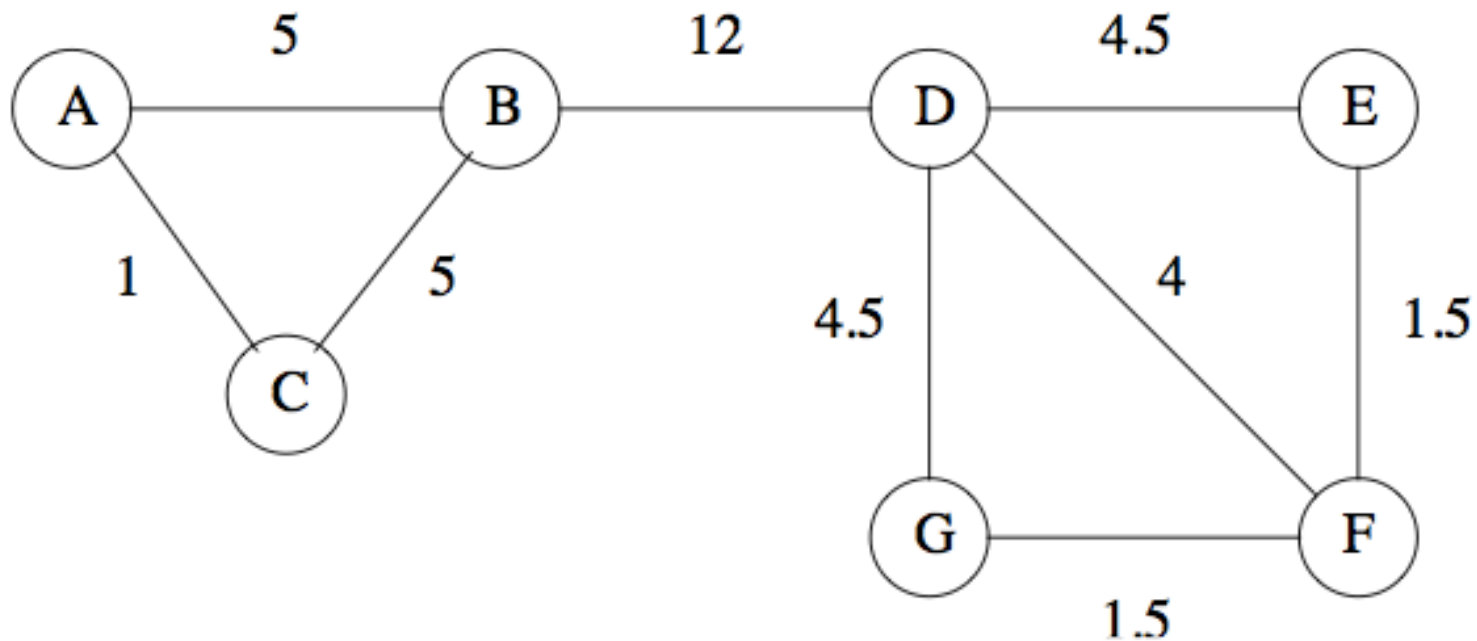


Figure 10.7: Betweenness scores for the graph of Fig. 10.1

Using Betweenness to Find Communities (3): Thresholding

- ❑ Could continue to remove edges with highest betweenness

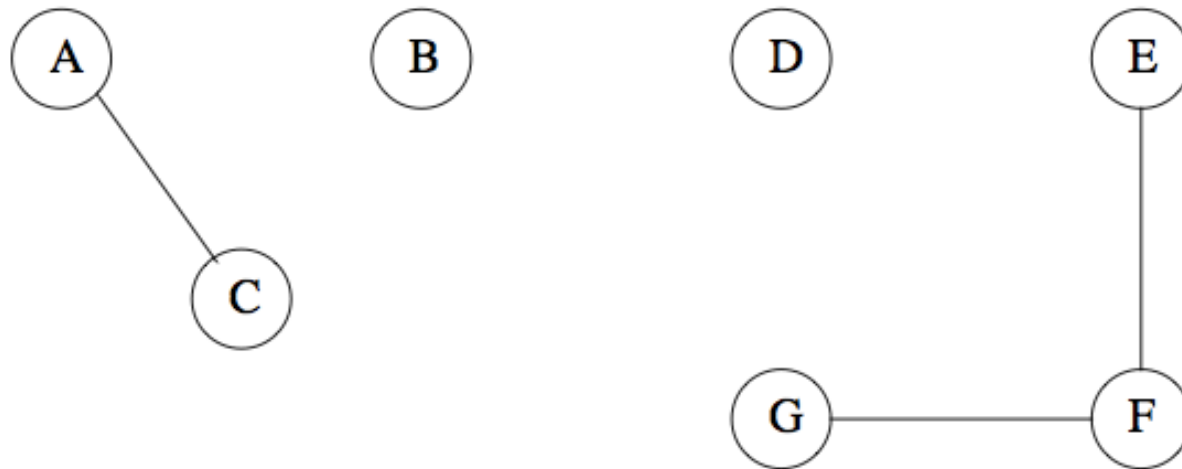


Figure 10.8: All the edges with betweenness 4 or more have been removed

Run Girvan-Newman Iteratively for Community Detection

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

Number of shortest paths passing through the edge

- ❑ **Girvan-Newman Algorithm:**

- » Undirected unweighted networks

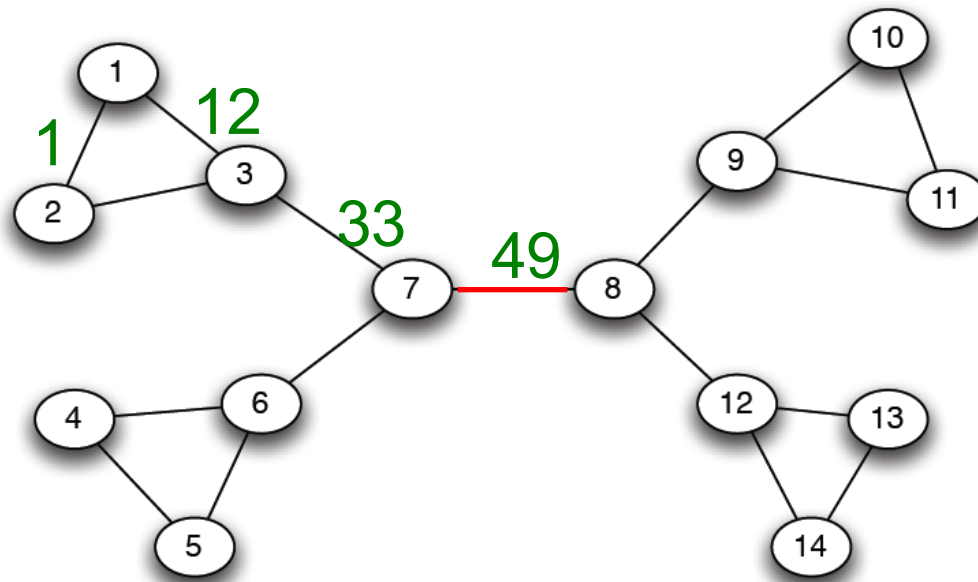
- **Repeat until no edges are left:**

- Calculate betweenness of edges
- **This time: remove edges with highest betweenness**

- Connected components are communities

- Gives a hierarchical decomposition of the network.

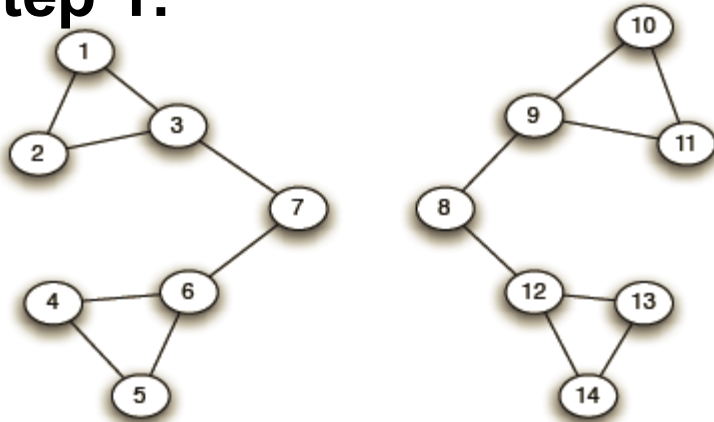
Girvan-Newman: Example



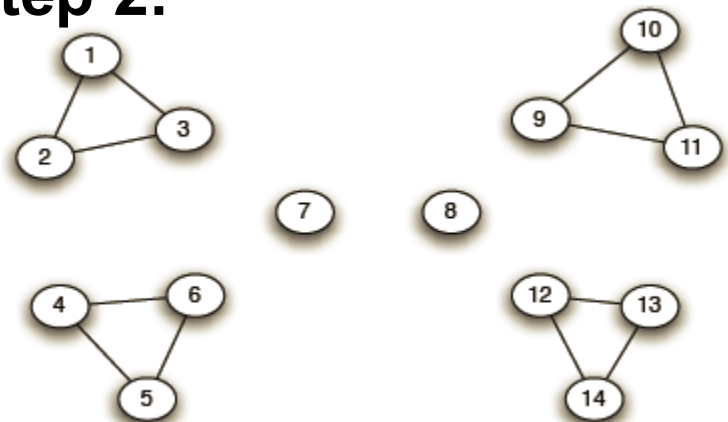
Need to re-compute
betweenness at
every step

Girvan-Newman: Example

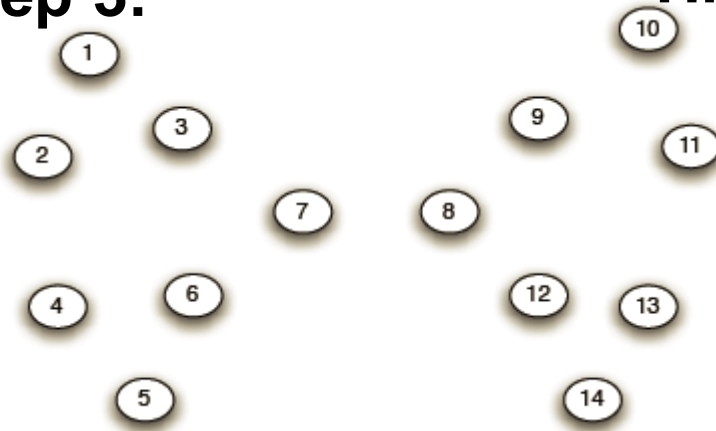
Step 1:



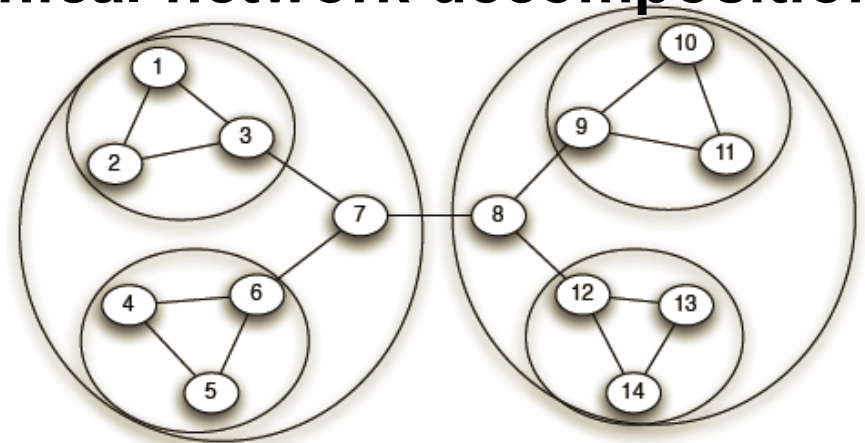
Step 2:



Step 3:

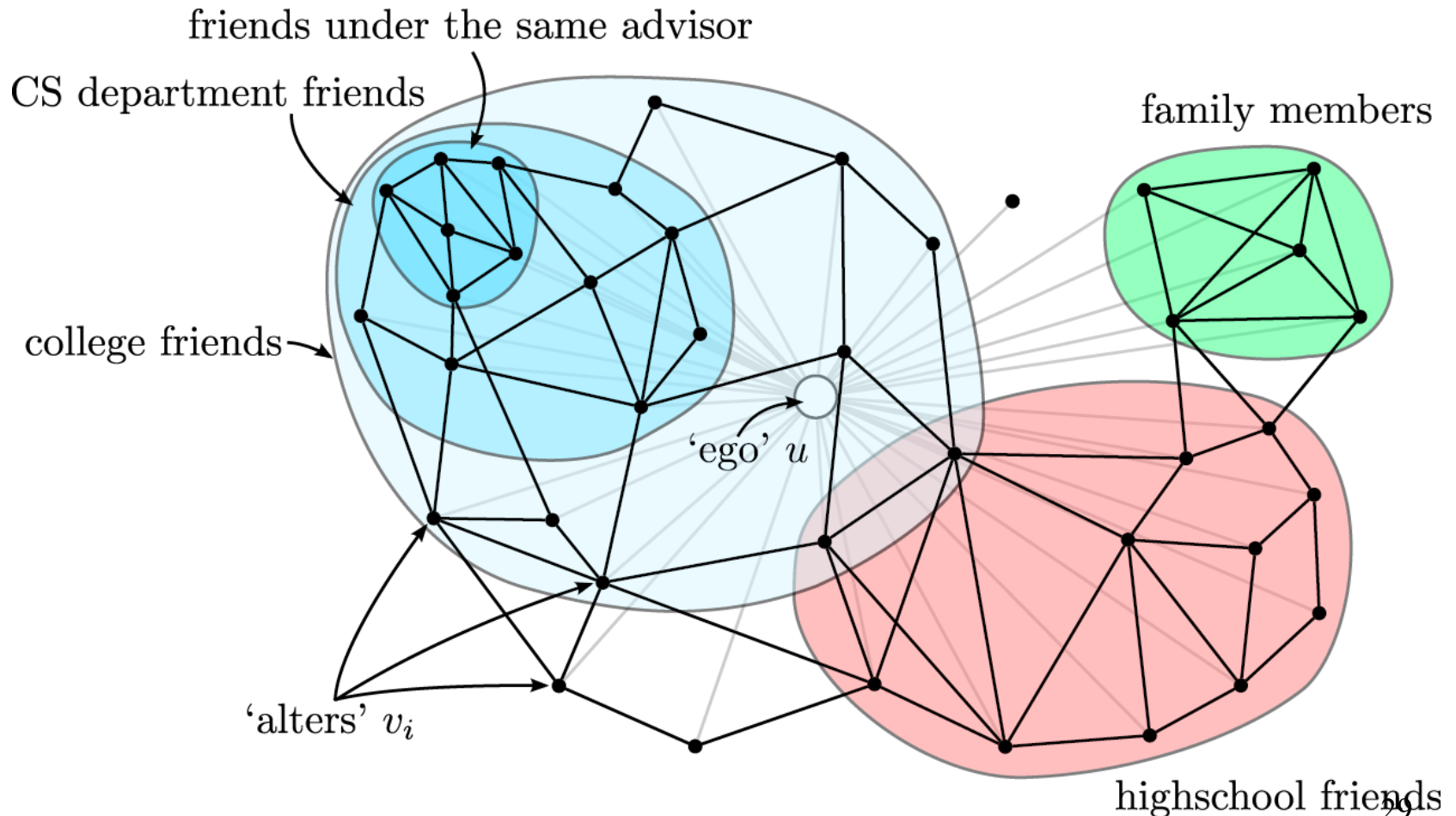


Hierarchical network decomposition:

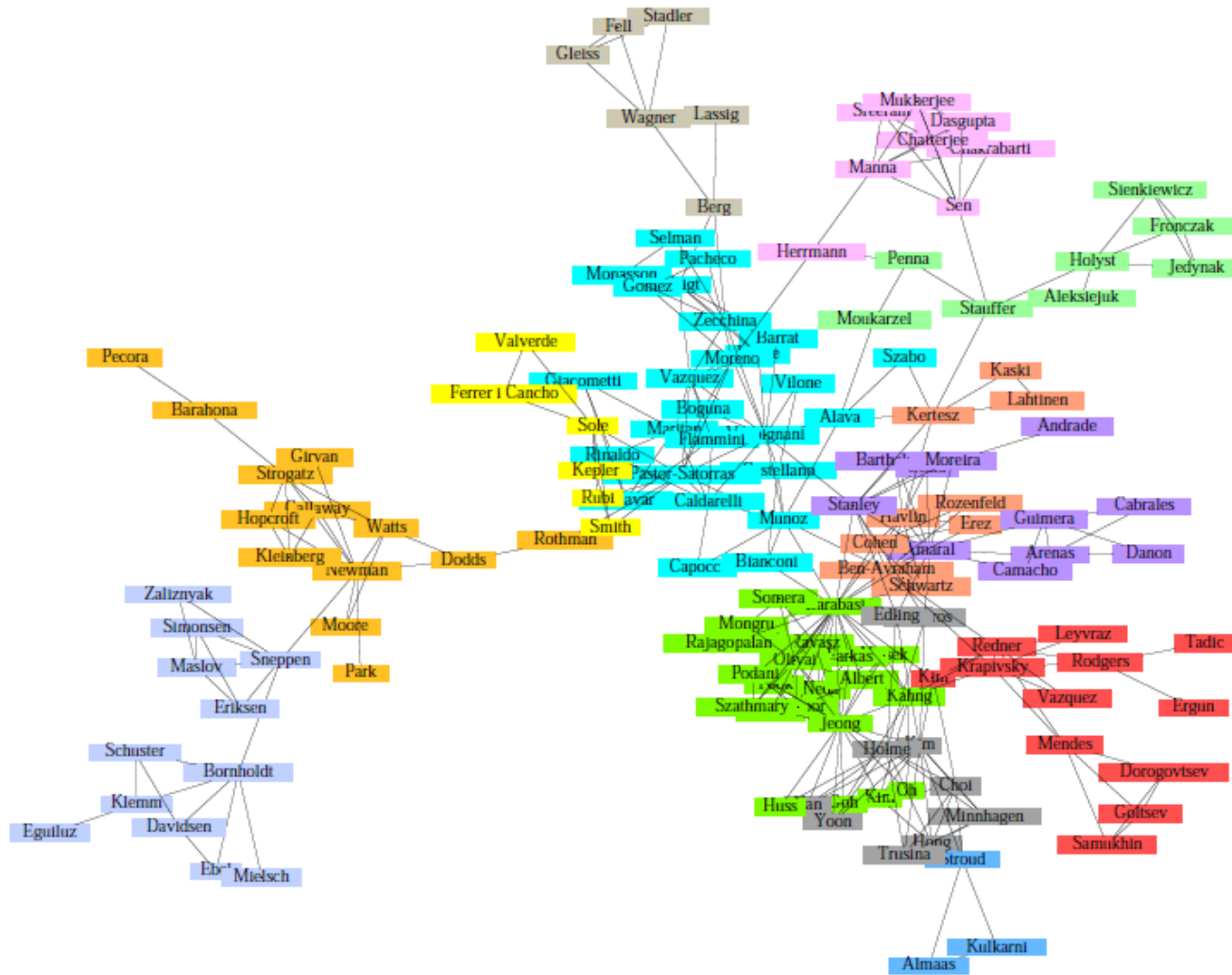


Recall: Twitter & Facebook

□ Discovering social circles, circles of trust:



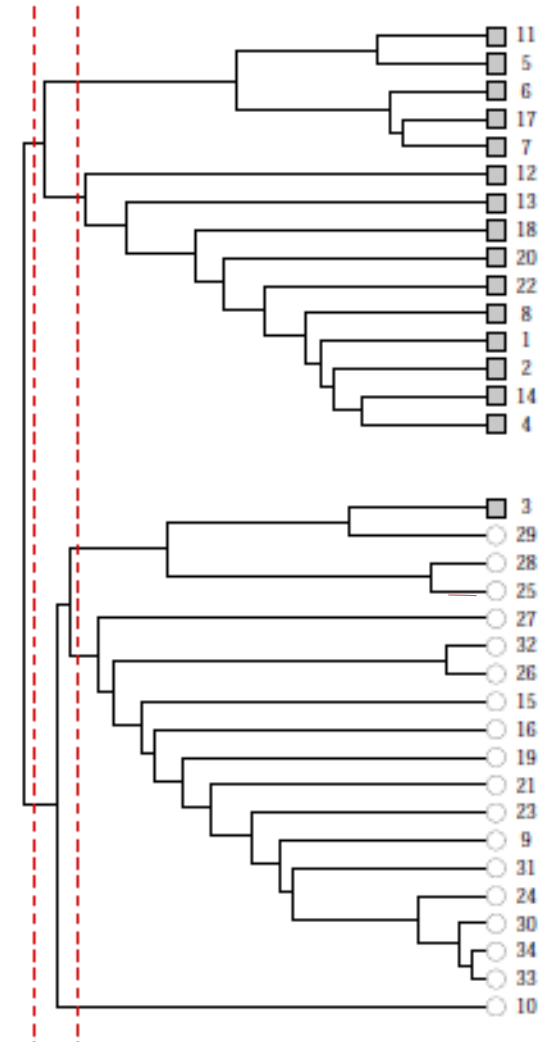
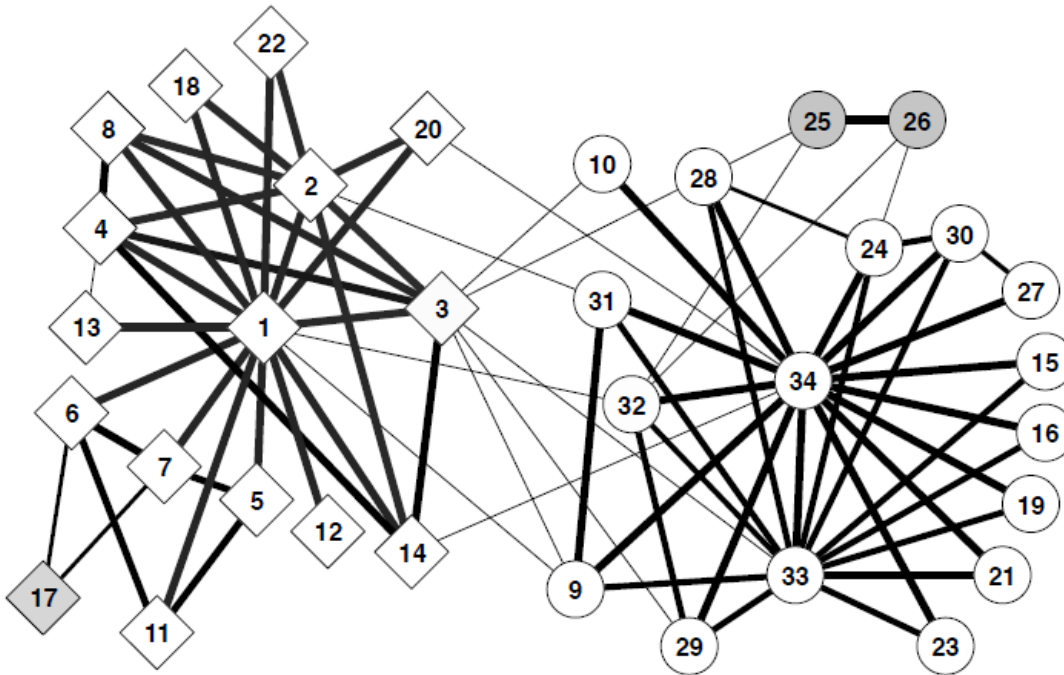
Girvan-Newman: Results



Communities in physics
collaborations

Girvan-Newman: Results

□ Zachary's Karate club: Hierarchical decomposition

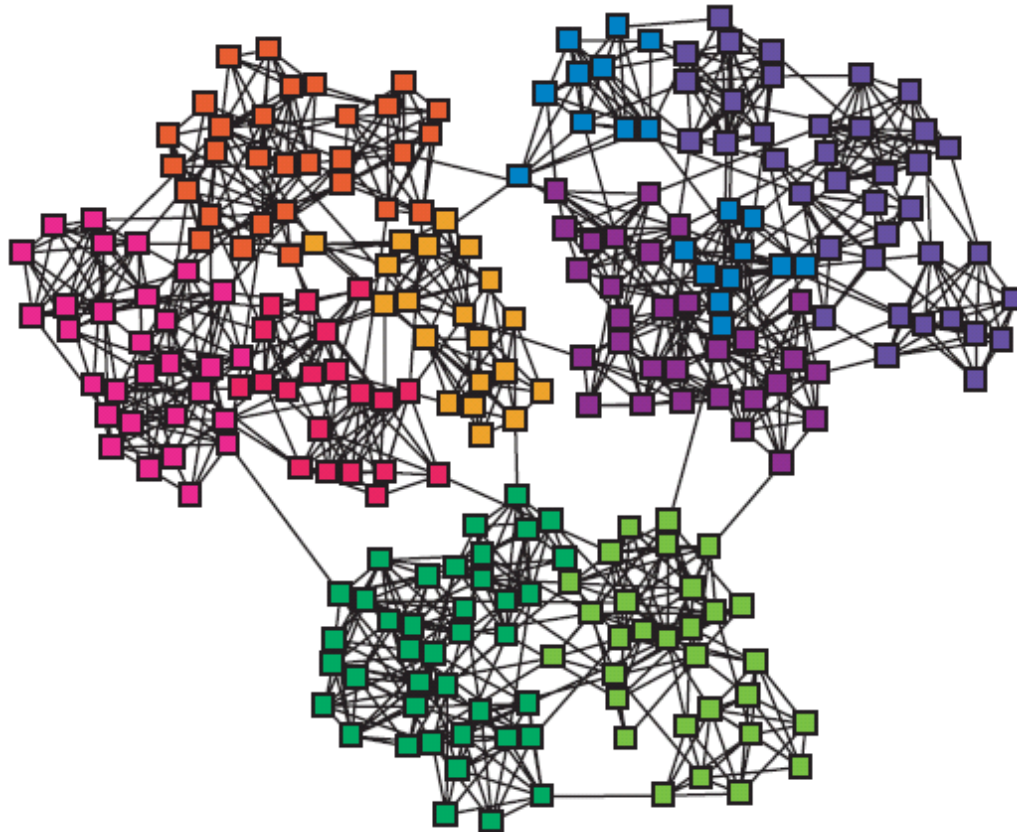


Mining Social-Network Graphs

Analysis of Large Graphs: Community Detection Cont.

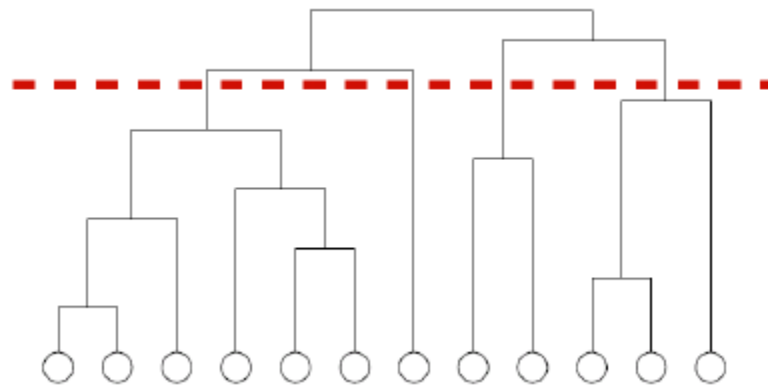
Networks & Communities

- We often think of networks being organized into **modules, cluster, communities:**



WE NEED TO RESOLVE 2 QUESTIONS

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



Network Communities

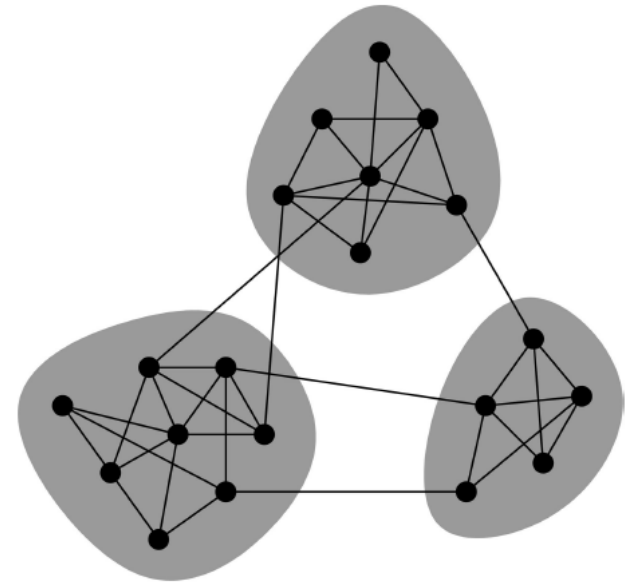
❑ **Communities:** sets of tightly connected nodes

❑ Define: **Modularity Q**

- A measure of how well a network is partitioned into communities
- Given a partitioning of the

network into groups $s \in S$:

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



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

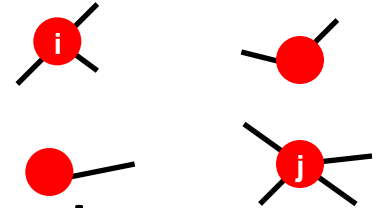
i and j of degrees k_i and k_j equals to: $k_i \cdot \frac{k_j}{2m} = \frac{k_i k_j}{2m}$

• 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 :

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

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

Normalizing $\underbrace{\hspace{1cm}}$ cost.: $-1 < Q < 1$

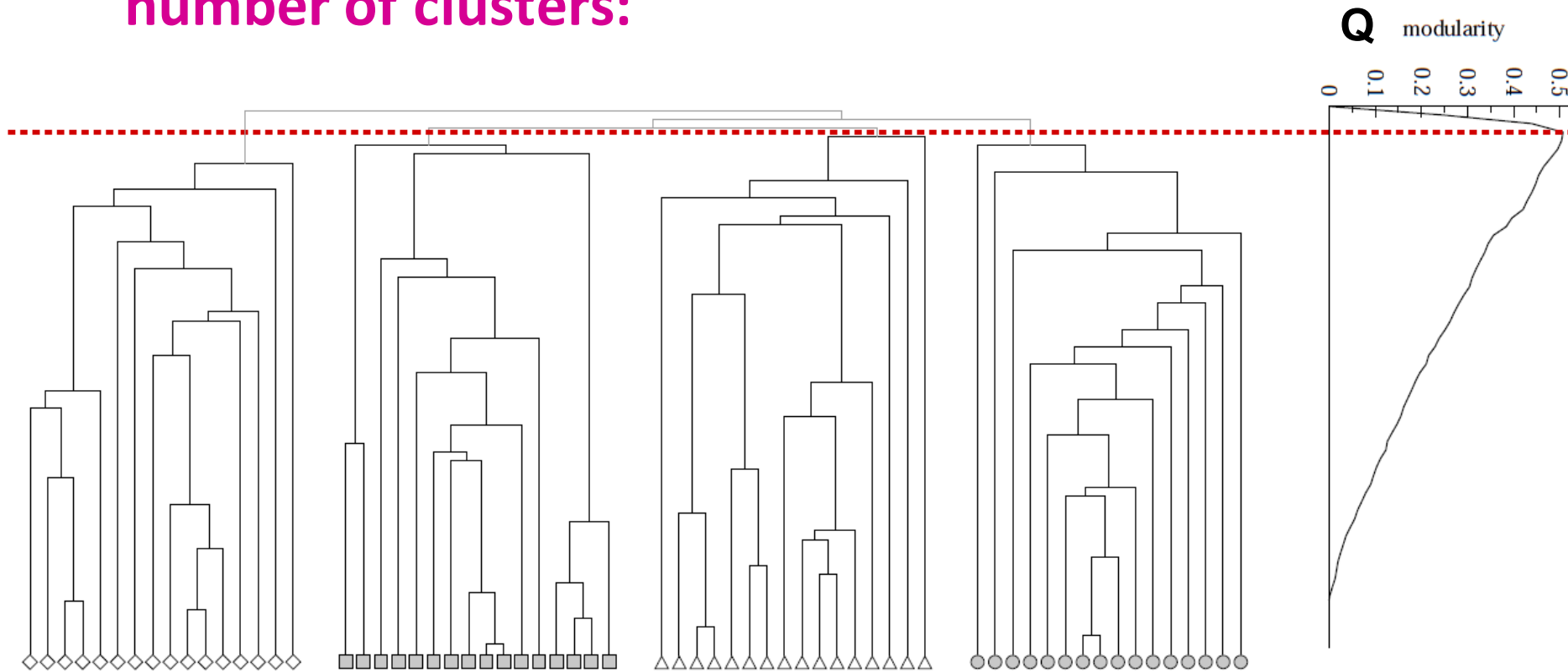
$A_{ij} = 1$ if i connects j ,
0 else

□ 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:



Another approach to organizing social-network graphs

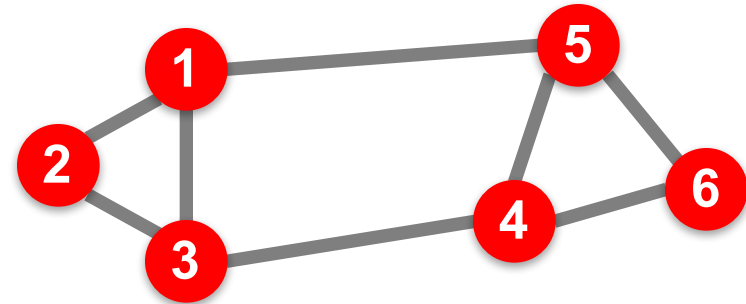
SPECTRAL CLUSTERING

Partitioning Graphs

- ❑ Another approach to organizing social networking graphs
- ❑ **Problem:** partitioning a graph to minimize the number of edges that connect different components (communities)
- ❑ Goal of minimizing the cut size
- ❑ If you just joined Facebook with only one friend
 - Don't want to partition the graph with you disconnected from rest of the world
 - Want components to be not too unequal in size.

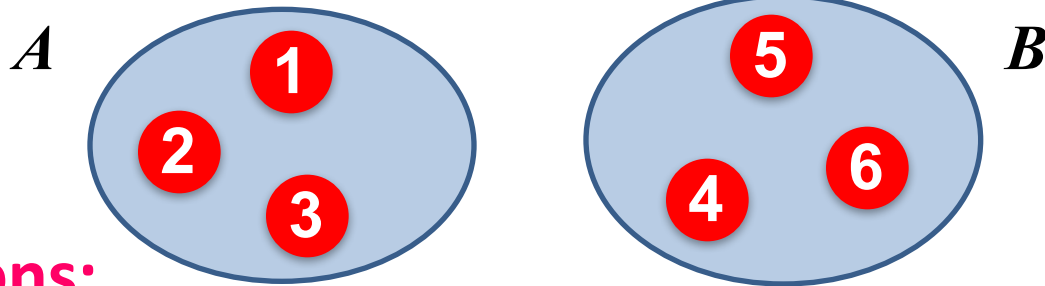
Graph Partitioning

□ Undirected graph



□ Bi-partitioning task:

➤ Divide vertices into two disjoint groups



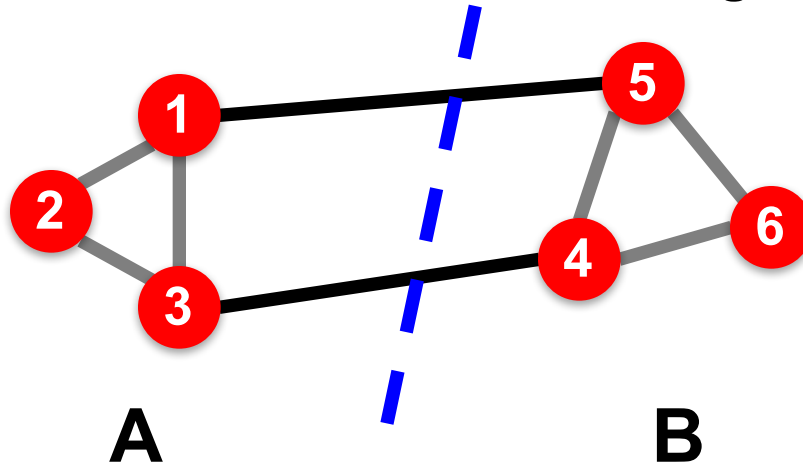
□ Questions:

- How can we define a “good” partition of ?
- How can we efficiently identify such a partition?

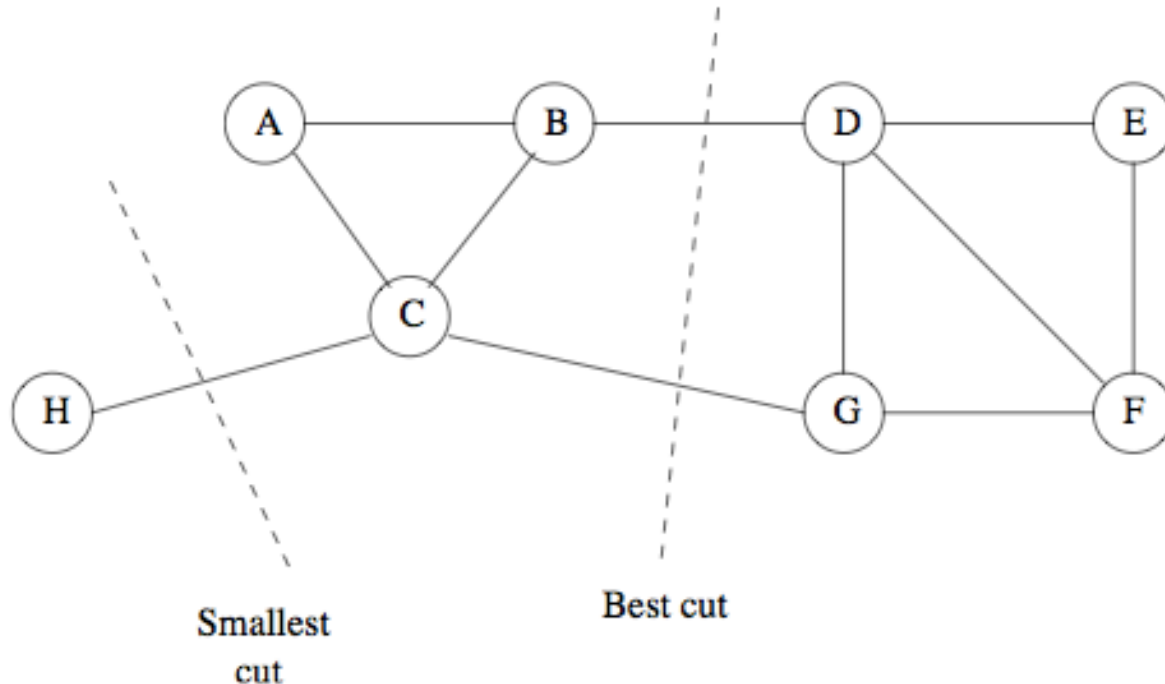
Graph Partitioning

❑ What makes a good partition?

- Divide nodes into two sets so that the **cut (set of edges that connect nodes in different sets) is minimized**
- Want the two sets to be approximately equal in size
- **Maximize** the number of within-group connections
- **Minimize** the number of between-group connections



Example 10.14

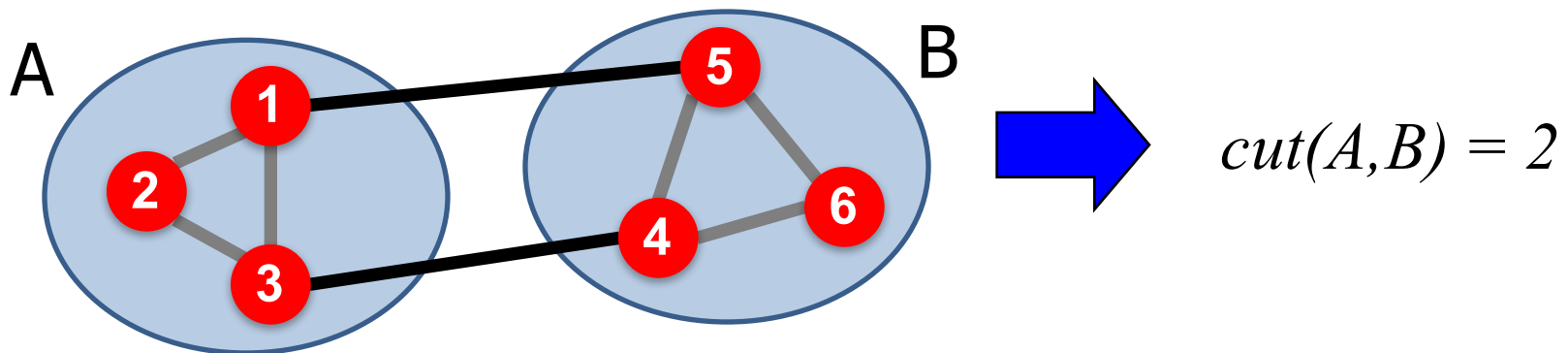


- ❑ If we minimize cut: best choice is to put H in one set, other nodes in other set
- ❑ But: we reject partitions where one set is too small
- ❑ Better is to use cut with (B,D) and (C,G)
- ❑ **Smallest cut is not necessarily the best cut**

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:

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



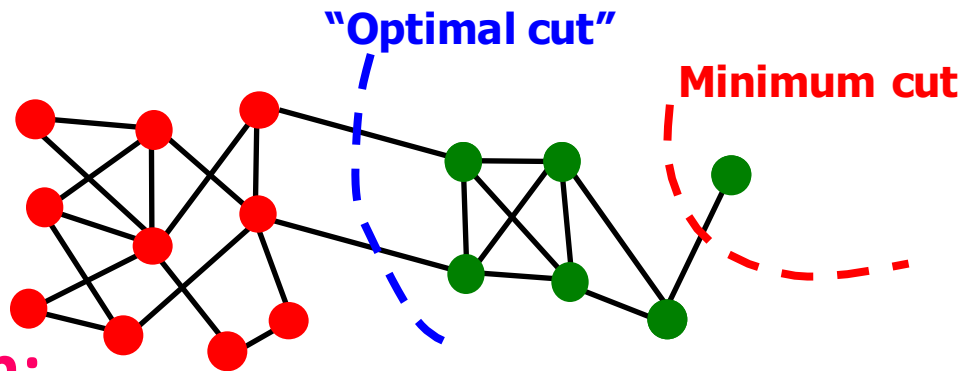
Graph Cut Criterion

❑ Criterion: **Minimum-cut**

- Minimize weight of connections between groups

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

❑ Degenerate case:



❑ Problem:

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

Graph Cut Criteria

□ Criterion: **Normalized-cut** [Shi-Malik, '97]

- Connectivity between groups relative to the density of each group

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

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

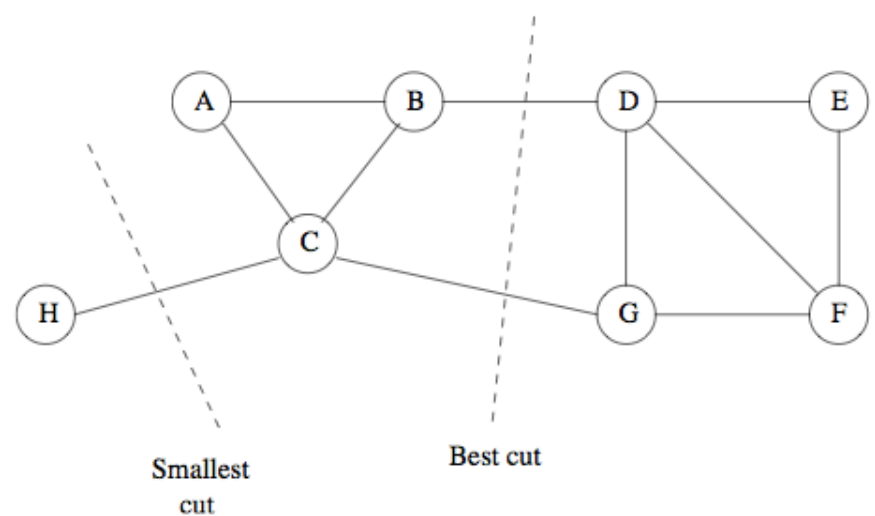
■ **Why use this criterion?**

- Produces more balanced partitions

□ **How do we efficiently find a good partition?**

- **Problem:** Computing optimal cut is NP-hard

Example 10.15



❑ Partition nodes of graph into two disjoint sets S and T

❑ Normalized Cut for S and T is:

$$\frac{\text{Cut}(S,T)}{\text{Vol}(S)} + \frac{\text{Cut}(S,T)}{\text{Vol}(T)}$$

❑ If we choose $S=\{H\}$ and $T=\{A,B,C,D,E,F,G\}$ then $\text{Cut}(S,T) = 1$

➤ $\text{Vol}(S) = 1$ (number of edges with at least one end in S)

➤ $\text{Vol}(T) = 11$: all edges have at least one node in T

➤ Normalized cut is $1/1 + 1/11 = 1.09$

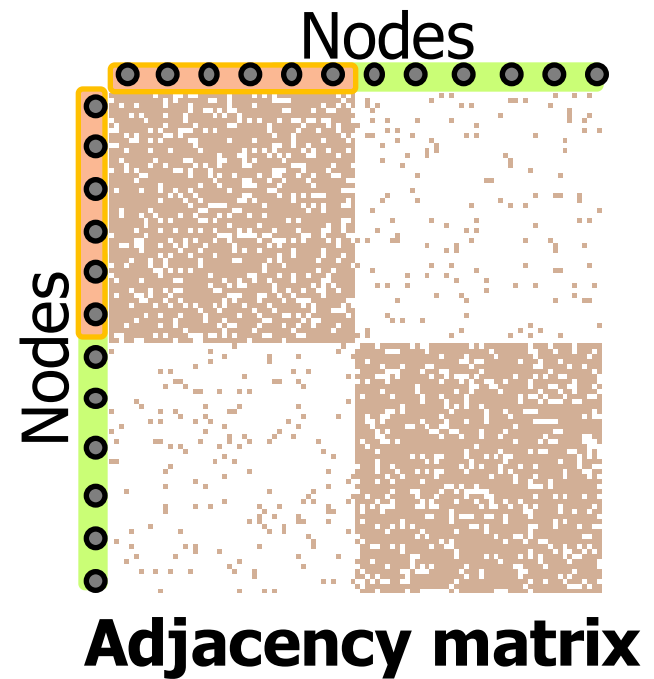
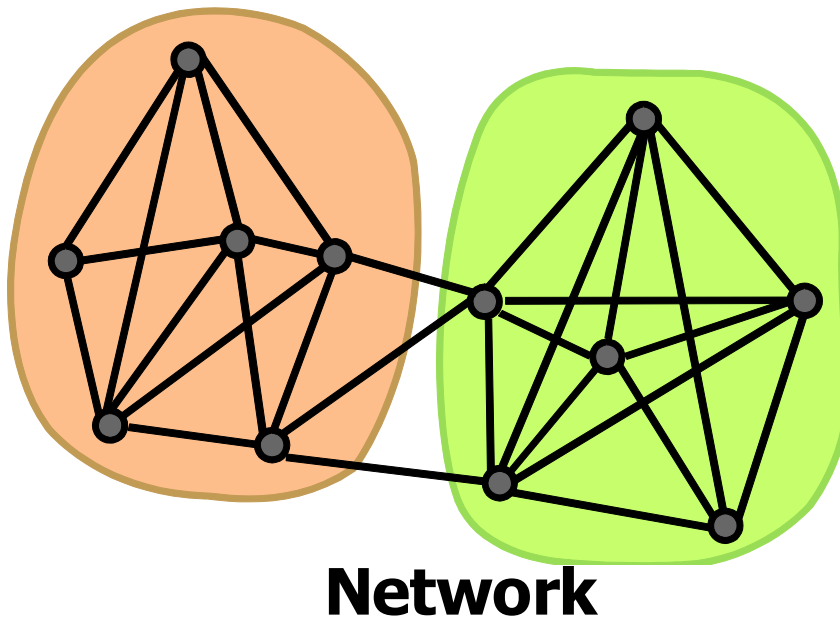
❑ For cut (B,D) and (C,G) : $S = \{A,B,C,H\}$, $T = \{D,E,F,G\}$, $\text{Cut}(S,T) = 2$

❑ $\text{Vol}(S) = 6$, $\text{Vol}(T) = 7$, normalized cut: $2/6 + 2/7 = 0.62$

Using Matrix Algebra to Find Good Graph Partitions

- ❑ Three matrices that describe aspects of a graph:
 - Adjacency Matrix
 - Degree Matrix
 - Laplacian Matrix: difference between degree and adjacency matrix
- ❑ Then get a good idea of how to partition graph from eigenvalues and eigenvectors of its Laplacian matrix.

Finding clusters



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, \dots, x_n)

➤ Think of it as a label/value of each node of G

□ **What is the meaning of $A \cdot x$?**

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$
$$y_i = \sum_{j=1}^n A_{ij} x_j = \sum_{(i,j) \in E} x_j$$

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

Eigenvectors and Eigenvalues

- An **eigenvector** of a square matrix **A** is a non-zero vector **v** that, when the matrix multiplies **v**, yields the same as when some scalar multiplies **v**, the scalar multiplier often being denoted by λ
- That is:

$$\mathbf{Av} = \lambda \mathbf{v}$$

- The number λ is called the **eigenvalue** of **A** corresponding to **v**.

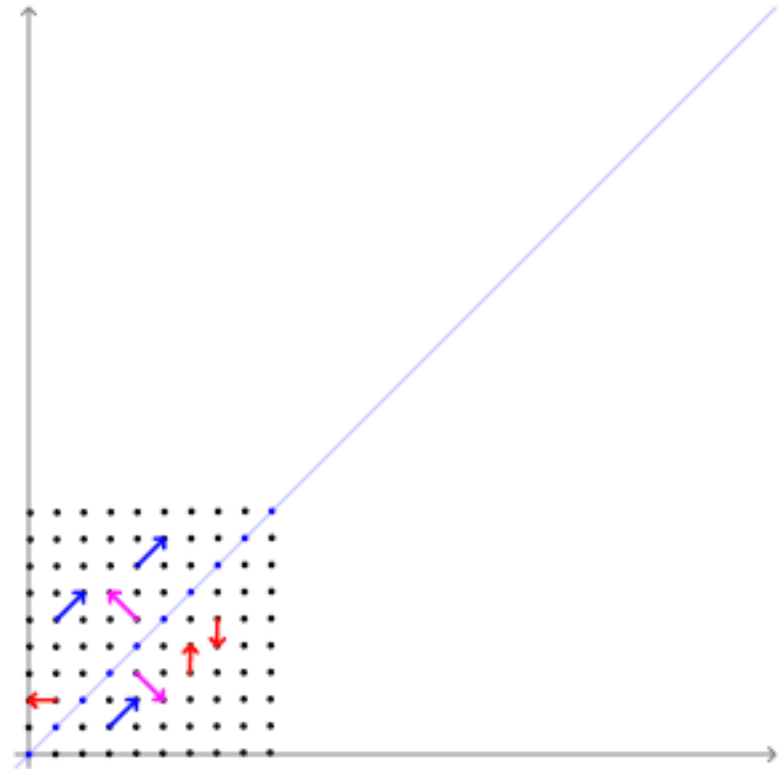
Eigenvalues and Eigenvectors Example

- The transformation matrix $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ preserves the direction of vectors parallel to $\mathbf{v} = (1, -1)^T$ (in purple) and $\mathbf{w} = (1, 1)^T$ (in blue). The vectors in red are not parallel to either eigenvector, so, their directions are changed by the transformation.

$$Ax = \lambda x$$

https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors

<http://setosa.io/ev/eigenvectors-and-eigenvalues/>



What is the meaning of Ax ?

□ j^{th} coordinate of $A \cdot x$:

- Sum of the x -values of neighbors of j
- Make this a new value at node j

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

□ Spectral Graph Theory:

- Analyze the “spectrum” of matrix representing G
- **Spectrum:** Eigenvectors x_i of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues λ_i :

$$\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$$

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$$

Example: d-regular graph

- Suppose all nodes in G have degree d and G is connected
- What are some eigenvalues/vectors of G ?
- $A \cdot x = \lambda \cdot x$ What is λ ? What x ?
 - Let's try: $x = (1, 1, \dots, 1)$
 - Then: $A \cdot x = (d, d, \dots, d) = \lambda \cdot x$. So: $\lambda = d$
 - We found eigenpair of G : $x = (1, 1, \dots, 1), \lambda = d$

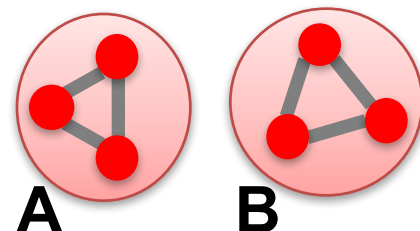
Remember the meaning of $y = A \cdot x$:

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

Example: Graph on 2 components

□ What if G is not connected?

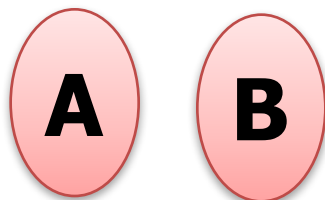
- G has 2 components, each d -regular



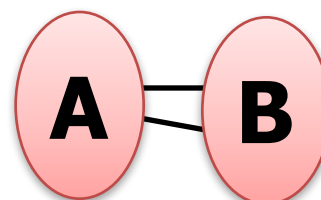
□ What are some eigenvectors?

- x = Put all 1s on A and 0s on B or vice versa
 - $x' = (1, \dots, 1, 0, \dots, 0)$ then $A \cdot x' = (d, \dots, d, 0, \dots, 0)$
 - $x'' = (\underbrace{0, \dots, 0}_A, \underbrace{1, \dots, 1}_B)$ then $A \cdot x'' = (0, \dots, 0, d, \dots, d)$
 - And so in both cases the corresponding $\lambda = d$

□ A bit of intuition:



$$\lambda_1 = \lambda_2$$

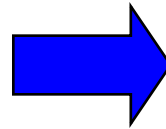
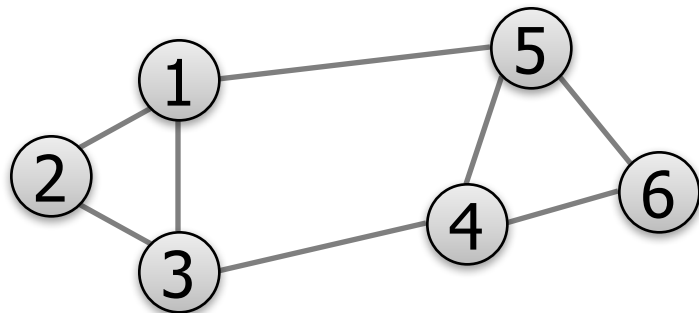


$$\lambda_1 \approx \lambda_2$$

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

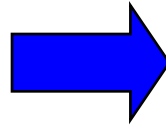
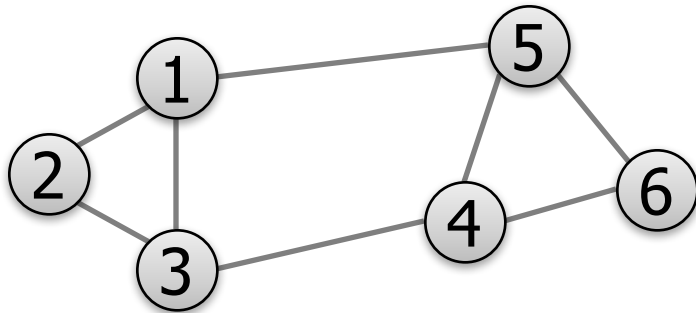
□ Important properties:

- Symmetric matrix
 - because it's an undirected graph
- Eigenvectors are real and orthogonal
 - orthogonal means
 $\text{dot_product}(\text{Eigenvectors}_i, \text{Eigenvectors}_j) = 0$

Matrix Representations

□ Degree matrix (D):

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

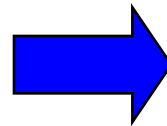
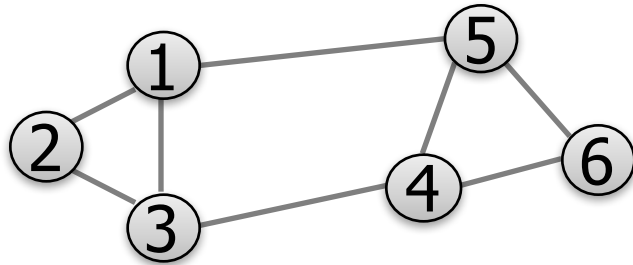


	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

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

□ What is trivial eigenpair?

- $x = (1, \dots, 1)$ then $L \cdot x = 0$ and so $\lambda = \lambda_1 = 0$ (smallest eigenvalue)

$$L = D - A$$

□ Important properties of symmetric matrices:

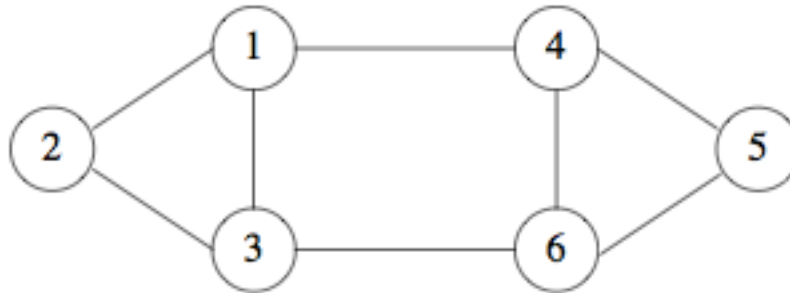
- Eigenvalues are non-negative real numbers
- Eigenvectors are real and orthogonal

$$\mathbf{x}^T \mathbf{1} = \sum_{i=1}^n x_i = 0$$

Partitioning Graphs Using Eigenvalues and Eigenvectors of Laplacian Matrix

- ❑ **Smallest eigenvalue** for every Laplacian matrix is **0**
- ❑ Its corresponding **eigenvector** is **$[1,1,1,\dots,1]$**
- ❑ To find **second-smallest eigenvalue** for symmetric matrix (such as Laplacian): (READ THE PAPER! [Shi-Malik, '97])
 - Second smallest eigenvalue is the minimum of $x^T L x$ where $x = [x_1, x_2, \dots, x_n]$ is a column vector (Rayleigh quotient)
 - Sum of $x_i^2 = 1$
 - x is orthogonal to the eigenvector associated with smallest eigenvalue
- ❑ Value of x that achieves this minimum is the **second eigenvector**
- ❑ This second smallest eigenvector x will have some positive and some negative components (why?)
- ❑ **Partition the graph by taking one set to be the nodes i whose corresponding vector component x_i is positive.**

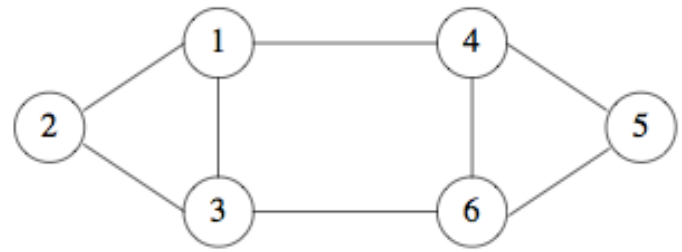
Example 10.19



$$\begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 \\ -1 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

□ Graph and its Laplacian matrix

Example (cont.)



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

- ❑ Use standard math package to find all eigenvalues and eigenvectors
 - (Have not scaled eigenvectors to length 1, but could)
- ❑ Second eigenvector has three positive and three negative components
- ❑ Suggest obvious partitioning of $\{1,2,3\}$ and $\{4,5,6\}$

λ_2 as optimization problem

❑ **Fact:** For symmetric matrix M :

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

❑ **What is the meaning of $\min x^T L x$ on G ?**

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

$$\triangleright = \sum_i D_{ii} x_i^2 - \sum_{(i,j) \in E} 2x_i x_j$$

$$\triangleright = \sum_{(i,j) \in E} \underbrace{(x_i^2 + x_j^2 - 2x_i x_j)}_{(x_i - x_j)^2} = \sum_{(i,j) \in E} (x_i - x_j)^2$$

Node i has degree d_i . So, value x_i^2 needs to be summed up d_i times.
But each edge (i,j) has two endpoints so we need $x_i^2 + x_j^2$

λ_2 as optimization problem (cont'd)

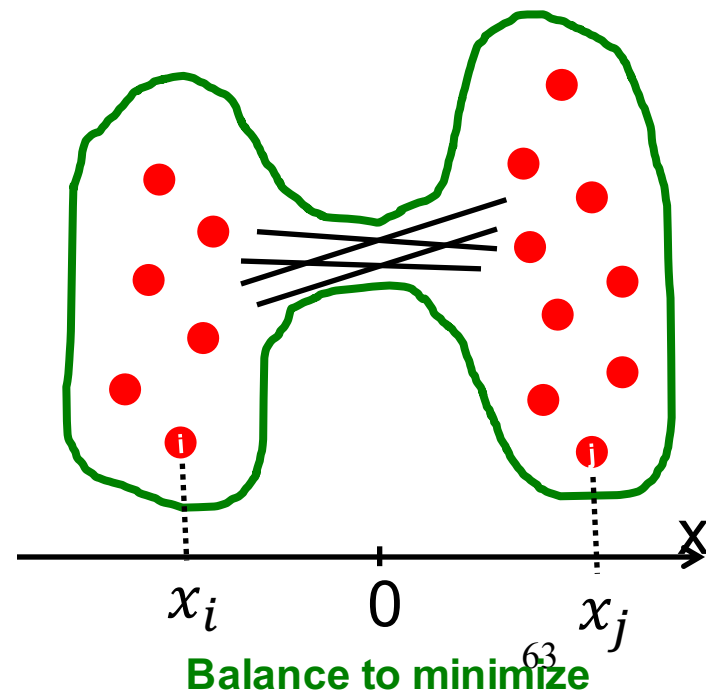
□ What else do we know about x ?

- x is unit vector: $\sum_i x_i^2 = 1$
- x is orthogonal to 1^{st} eigenvector $(1, \dots, 1)$ thus:
 $\sum_i x_i \cdot 1 = \sum_i x_i = 0$

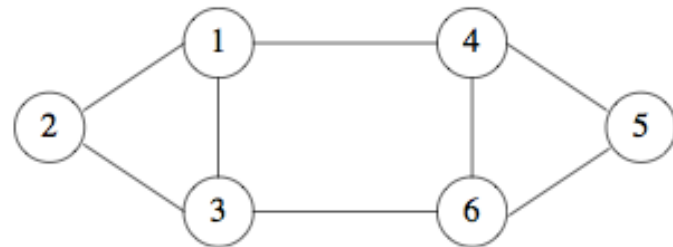
□ Remember:

$$\lambda_2 = \min_{\substack{\text{All labelings} \\ \text{of nodes } i \text{ so} \\ \text{that } \sum x_i = 0}} \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_i x_i^2} = 1$$

We want to assign values x_i to nodes i such that few edges cross 0.
(we want x_i and x_j to subtract each other)



Recall: 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

- ❑ Use standard math package to find all eigenvalues and eigenvectors
 - (Have not scaled eigenvectors to length 1, but could)
- ❑ Second eigenvector has three positive and three negative components
- ❑ Suggest obvious partitioning of $\{1,2,3\}$ and $\{4,5,6\}$

So far...

❑ How to define a “good” partition of a graph?

- Minimize 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

- Naïve approach:
 - Split at **0**

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

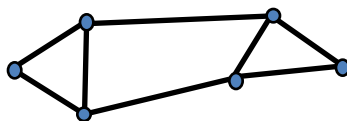
➤ 3) Grouping

- Assign points to two or more clusters, based on the new representation

Spectral Partitioning Algorithm

1) Pre-processing:

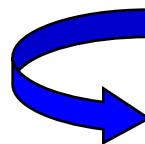
- Build Laplacian matrix L of the graph



	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
- Map vertices to corresponding components of λ_2



$\lambda =$

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

How do we now find the clusters?

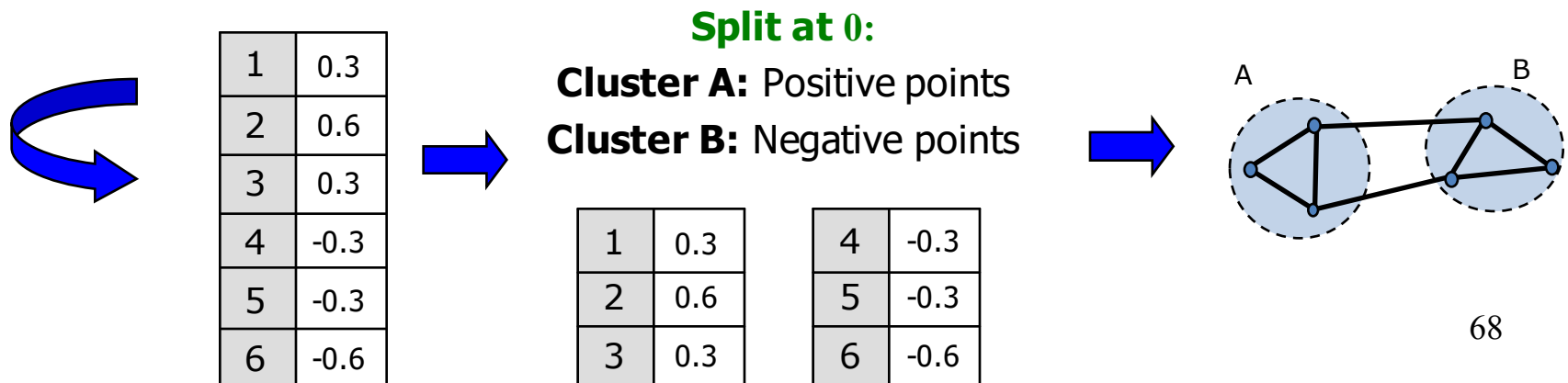
Spectral Partitioning

3) Grouping:

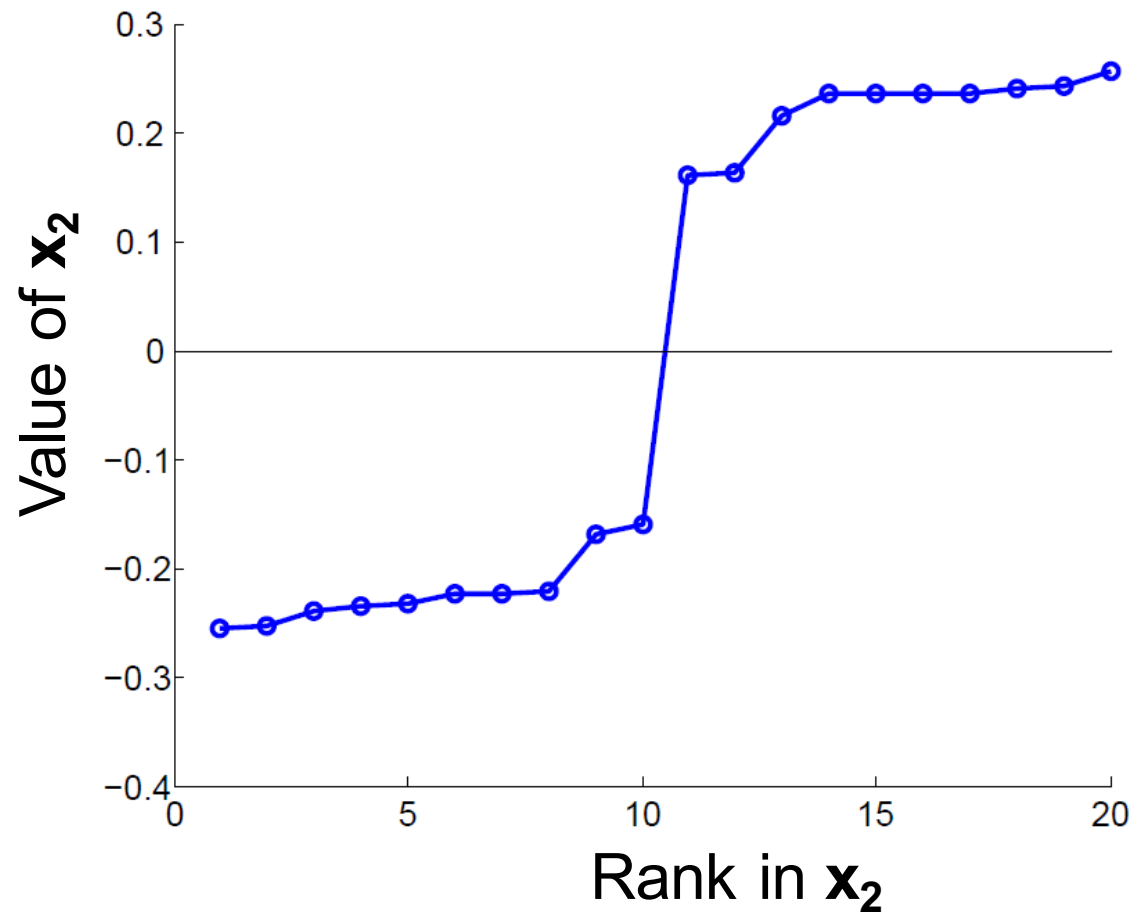
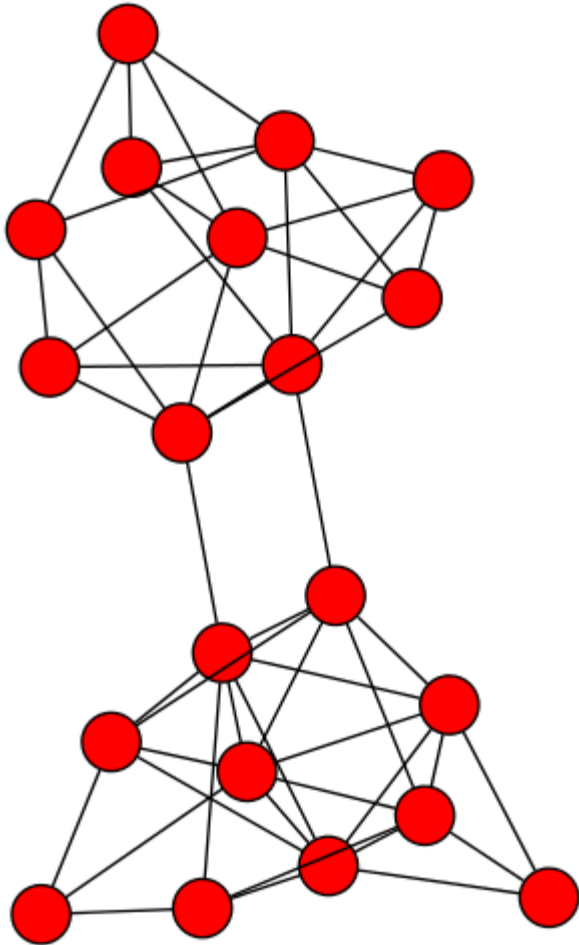
- Sort components of reduced 1-dimensional vector
- Identify clusters by splitting the sorted vector in two

How to choose a splitting point?

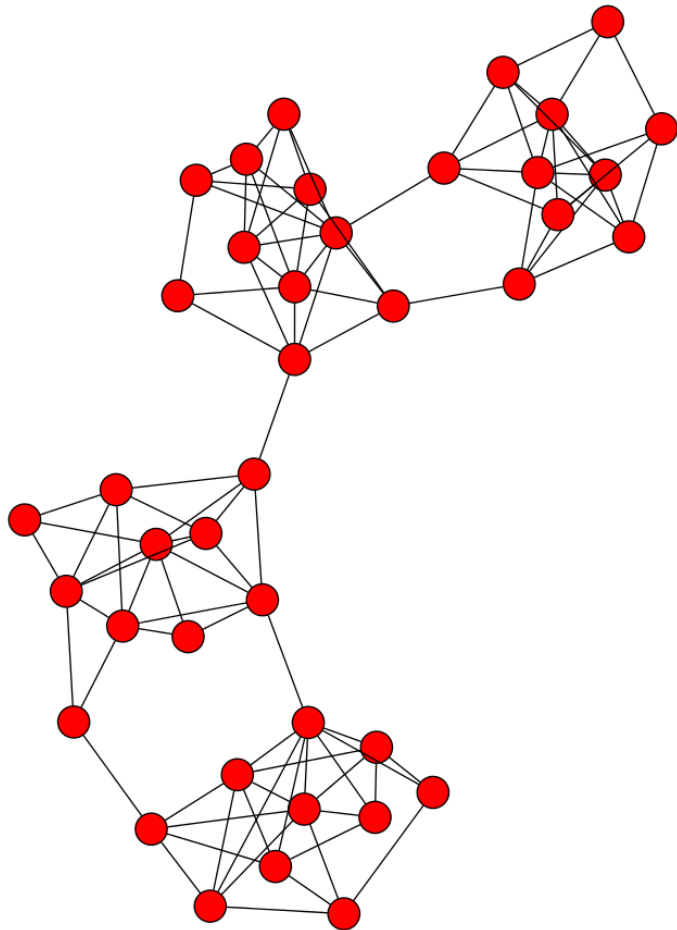
- Naïve approaches:
 - Split at **0** or median value
- More expensive approaches:
 - Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)



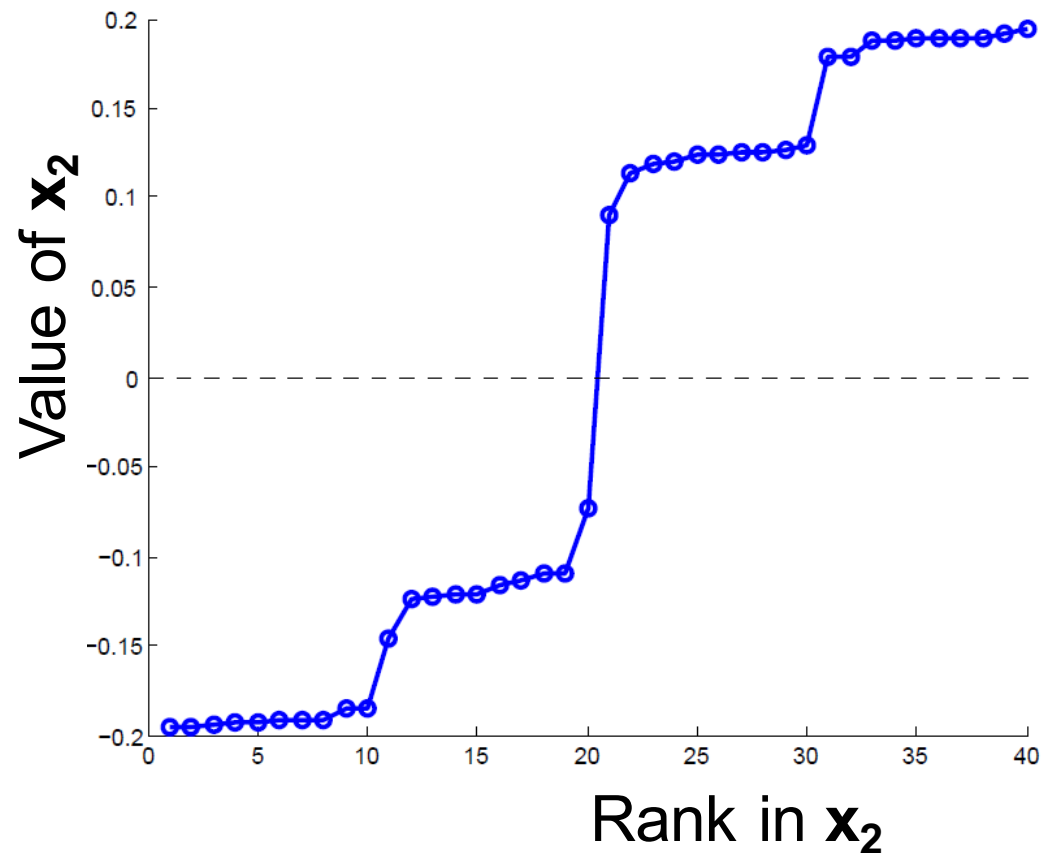
Example: Spectral Partitioning



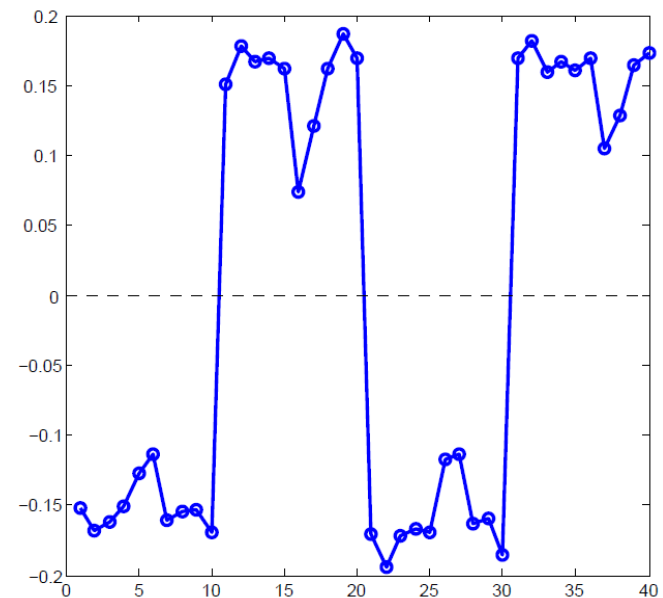
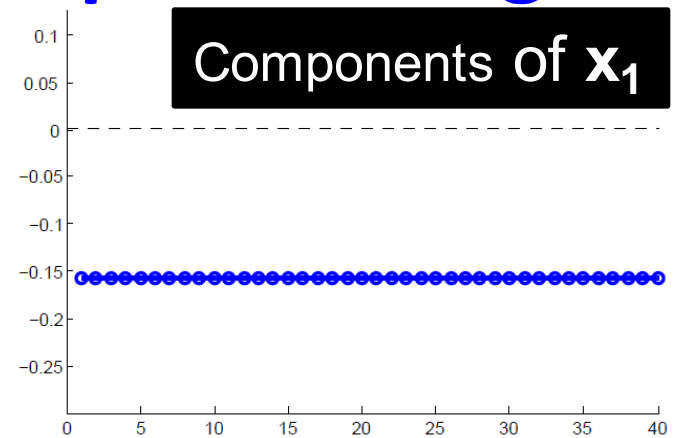
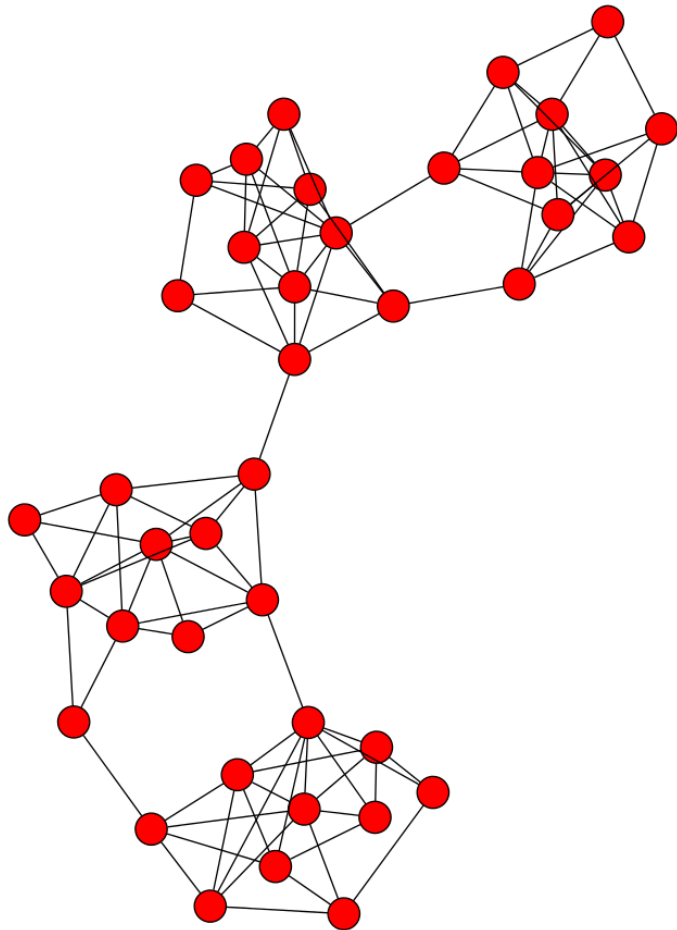
Example: Spectral Partitioning



Components of \mathbf{x}_2



Example: Spectral partitioning



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
- Multiple eigenvectors prevent instability due to information loss
- A preferable approach...

DIRECT DISCOVERY OF COMMUNITIES: TRAWLING

With slide contributions from P. Desikan; <http://www-users.cs.umn.edu/~desikan/>

Web community

- ❑ Groups of individuals who share common interests, together with the web pages most popular among them
- ❑ Web page collections with a shared topic.

Types of Communities

❑ Explicitly- defined

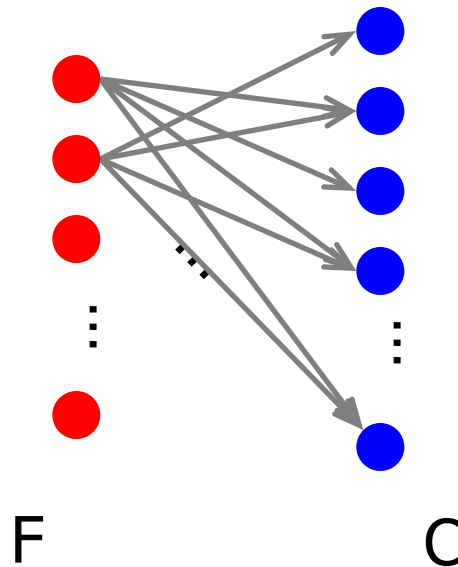
- Communities that manifest themselves as newsgroups or as resource collections on directories such as Yahoo!

❑ Implicitly- defined

- Communities that result from nature of content-creation of the web.

Terms and Definitions (1)

- *Directed Bipartite Graph*: A graph whose nodes set can be partitioned into two sets F and C , and every directed edge in the graph is from a node u in F to a node v in C



Terms and Definitions (2)

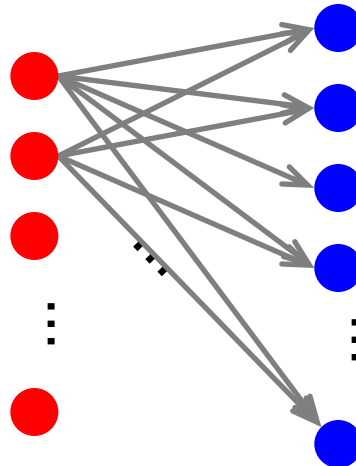
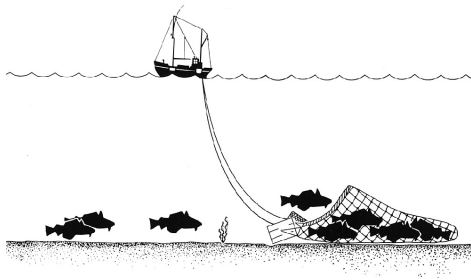
- ❑ *Completed Bipartite Graph*: A bipartite graph that contains all possible edges between a vertex of F and a vertex of C
- ❑ *Core*: A complete bipartite sub-graph with at least i nodes from F and at least j nodes from C
 - In the web world, the i pages that contain the links are referred to as '*fans*' and the j pages that are referenced as '*centers*'

Trawling the Web for Emerging Web Communities

- ❑ *Trawling*: Systematic Enumeration of emerging communities from web crawl
- ❑ Scan through a web crawl and identify all instances of graph structures that are indicative **signatures of communities.**

Trawling

- ❑ Searching for small communities in the Web graph
- ❑ What is the signature of a community / discussion in a Web graph?



Use this to define “topics”:
What the same people on the left talk about on the right

Dense 2-layer

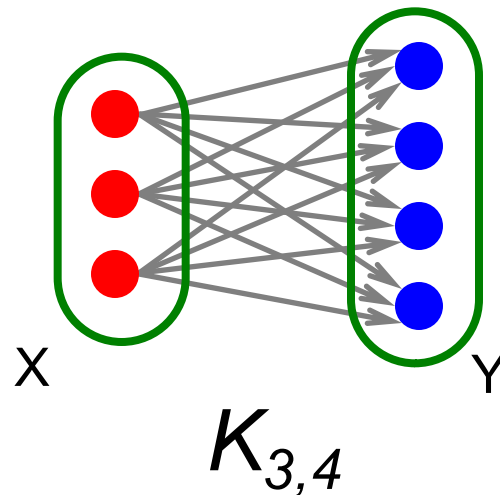
Intuition: Many people all talking about the same things ⁷⁹

Searching for Small Communities

□ A more well-defined problem:

Enumerate complete bipartite subgraphs $K_{s,t}$

- Where $K_{s,t}$: s nodes on the “left” where each links to the same t other nodes on the “right”



$$\begin{aligned} |X| &= s = 3 \\ |Y| &= t = 4 \end{aligned}$$

Fully connected

Frequent Itemset Enumeration

❑ Market basket analysis. Setting:

- **Market:** Universe U of n items
- **Baskets:** m subsets of U : $S_1, S_2, \dots, S_m \subseteq U$
(S_i is a set of items one person bought)
- **Support:** Frequency threshold f

❑ Goal:

- Find all subsets T s.t. $T \subseteq S_i$ of at least f sets S_i
(items in T were bought together at least f times)

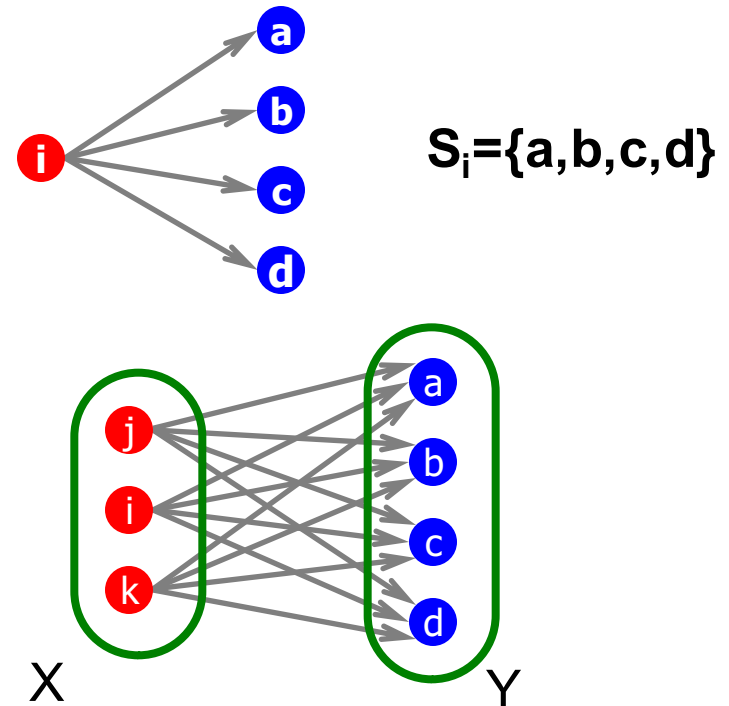
❑ What's the connection between the itemsets and complete bipartite graphs?

From Itemsets to Bipartite $K_{s,t}$

Frequent itemsets = complete bipartite graphs!

□ How?

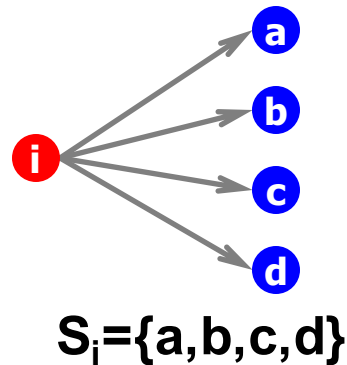
- View each node i as a set S_i of nodes i points to
- $K_{s,t}$ = a set Y of size t (all items) that occurs in s (a basket) sets S_i
- Looking for $K_{s,t} \rightarrow$ set of frequency threshold to s and look at layer t – all frequent sets of size t



s ... minimum support ($|X|=s$)
 t ... itemset size ($|Y|=t$)

From Itemsets to Bipartite $K_{s,t}$

View each node i as a set S_i of nodes i points to

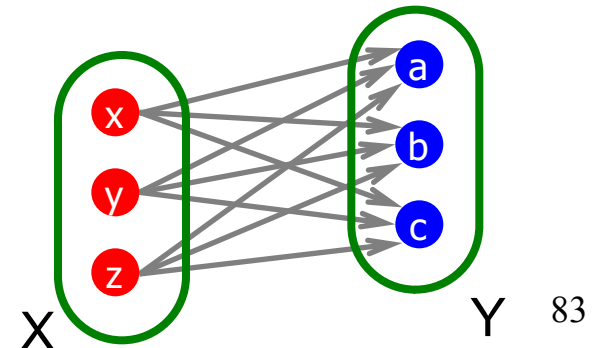
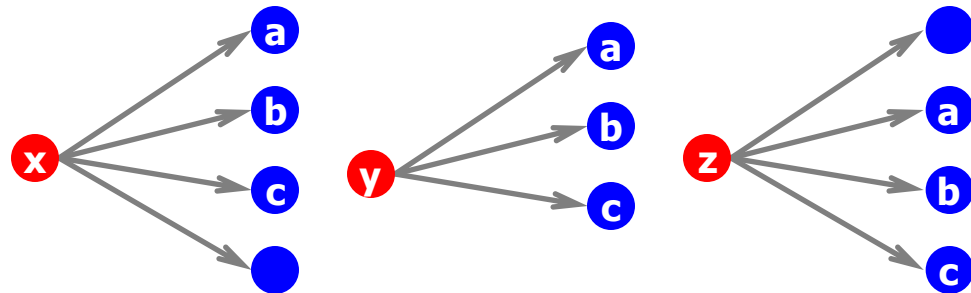


Find frequent itemsets:
 s ... minimum support
 t ... itemset size

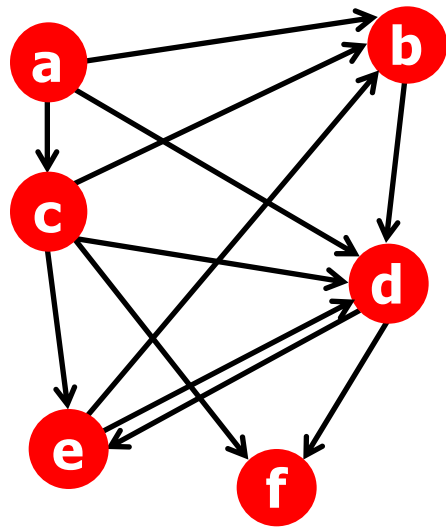
We found $K_{s,t}$!

$K_{s,t}$ = a set Y of size t
that occurs in s sets S_i

Say we find a **frequent itemset** $Y = \{a, b, c\}$ of supp s
So, there are s nodes that link to all of $\{a, b, c\}$:



Example



Itemsets:

$a = \{b, c, d\}$

$b = \{d\}$

$c = \{b, d, e, f\}$

$d = \{e, f\}$

$e = \{b, d\}$

$f = \{\}$

Support threshold $s=2$

➤ $\{b, d\}$: support 3

➤ $\{e, f\}$: support 2

And we just found 2 bipartite subgraphs:

