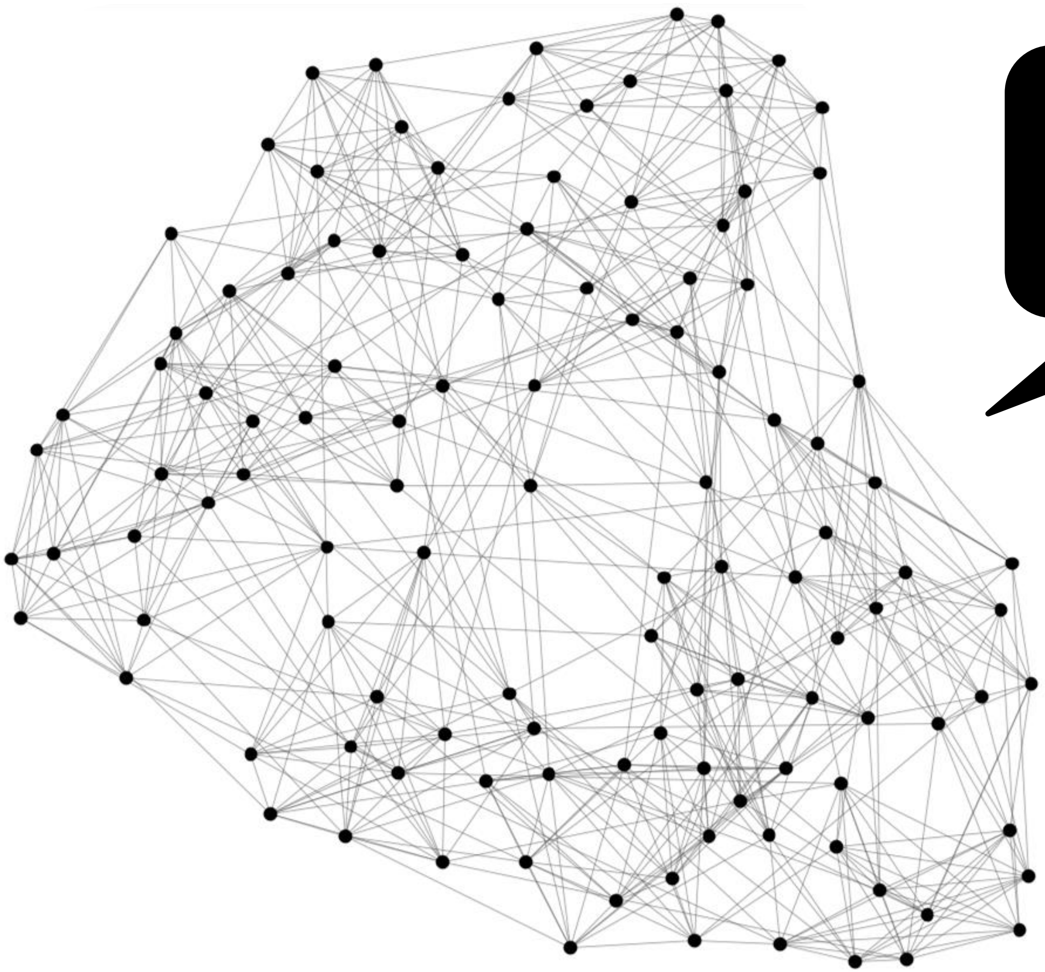
DATA MINING
LECTURE 12

Community detection in graphs

Communities

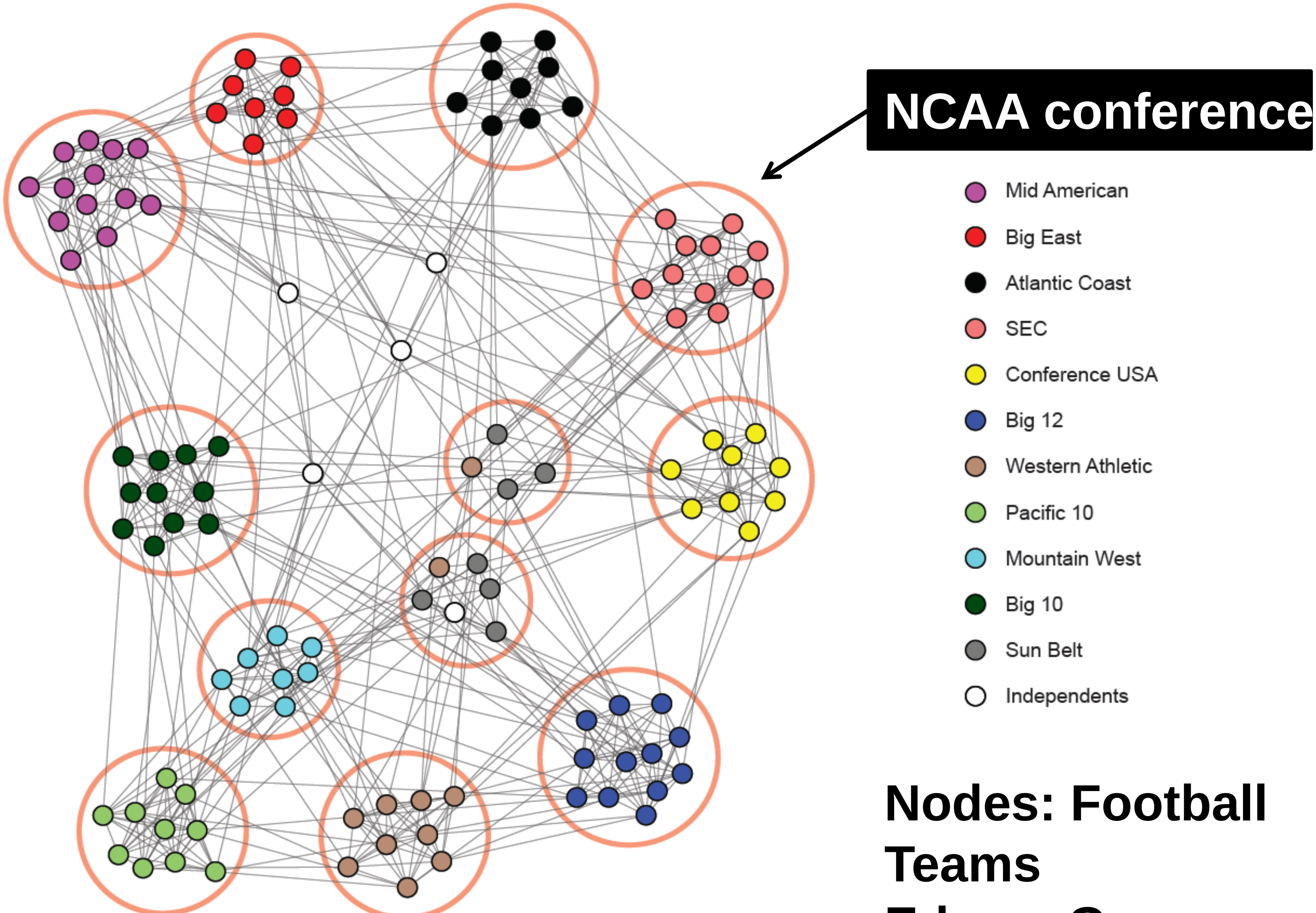
- Real-life graphs are not random
  - E.g., in a social network people pick their friends based on their common interests and activities
- We expect that the nodes in a graph will be organized in communities
  - Groups of vertices which probably share common properties and/or play similar roles within the graph
- How do we find them?
  - Nodes in communities will be densely connected to each other, and sparsely connected with other communities
  - Sounds familiar?

NCAA Football network



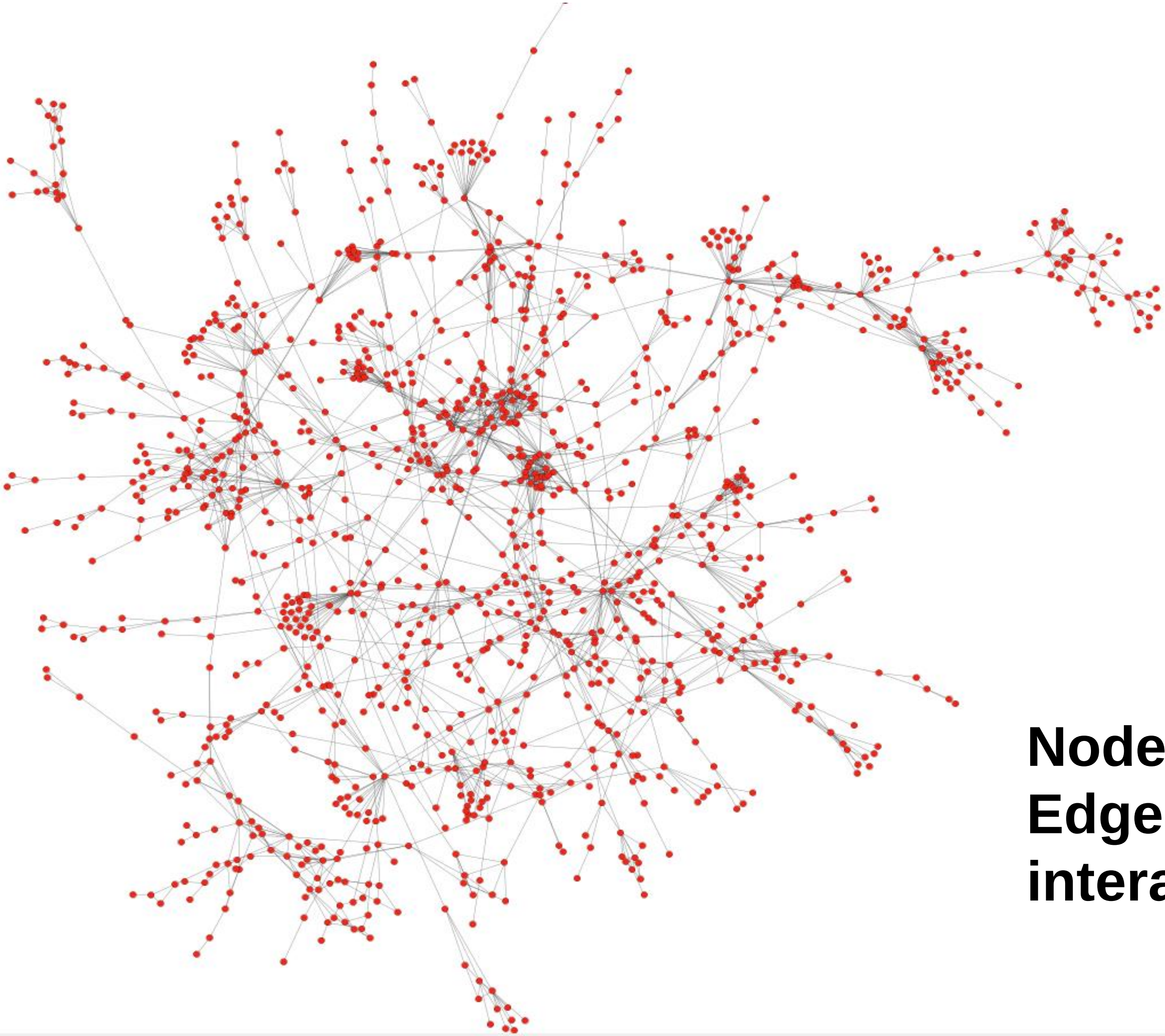
Can we identify node groups? (communities, modules, clusters)

Nodes: Football Teams
Edges: Games played



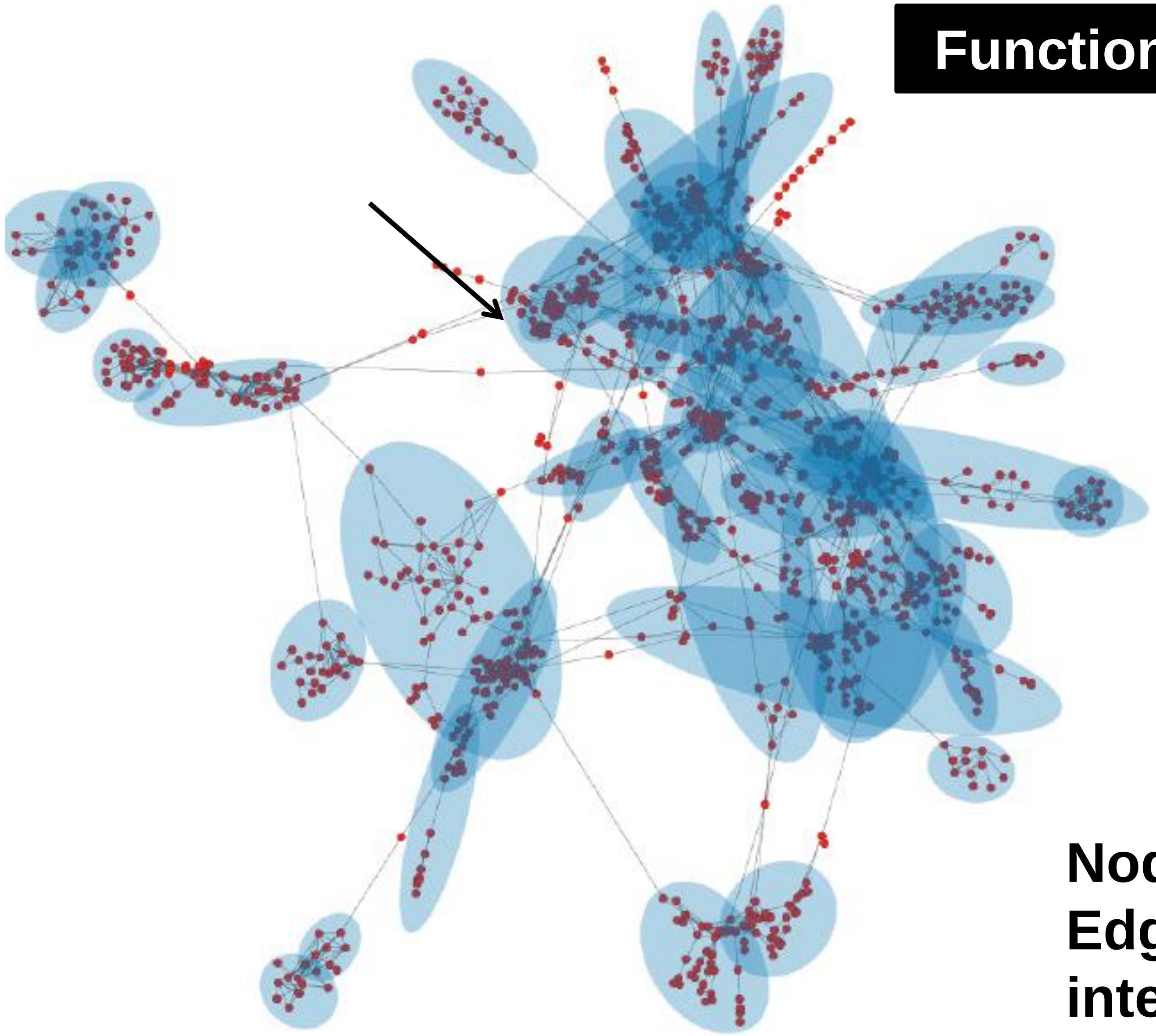
Nodes: Football Teams
Edges: Games played

Protein-Protein interaction networks



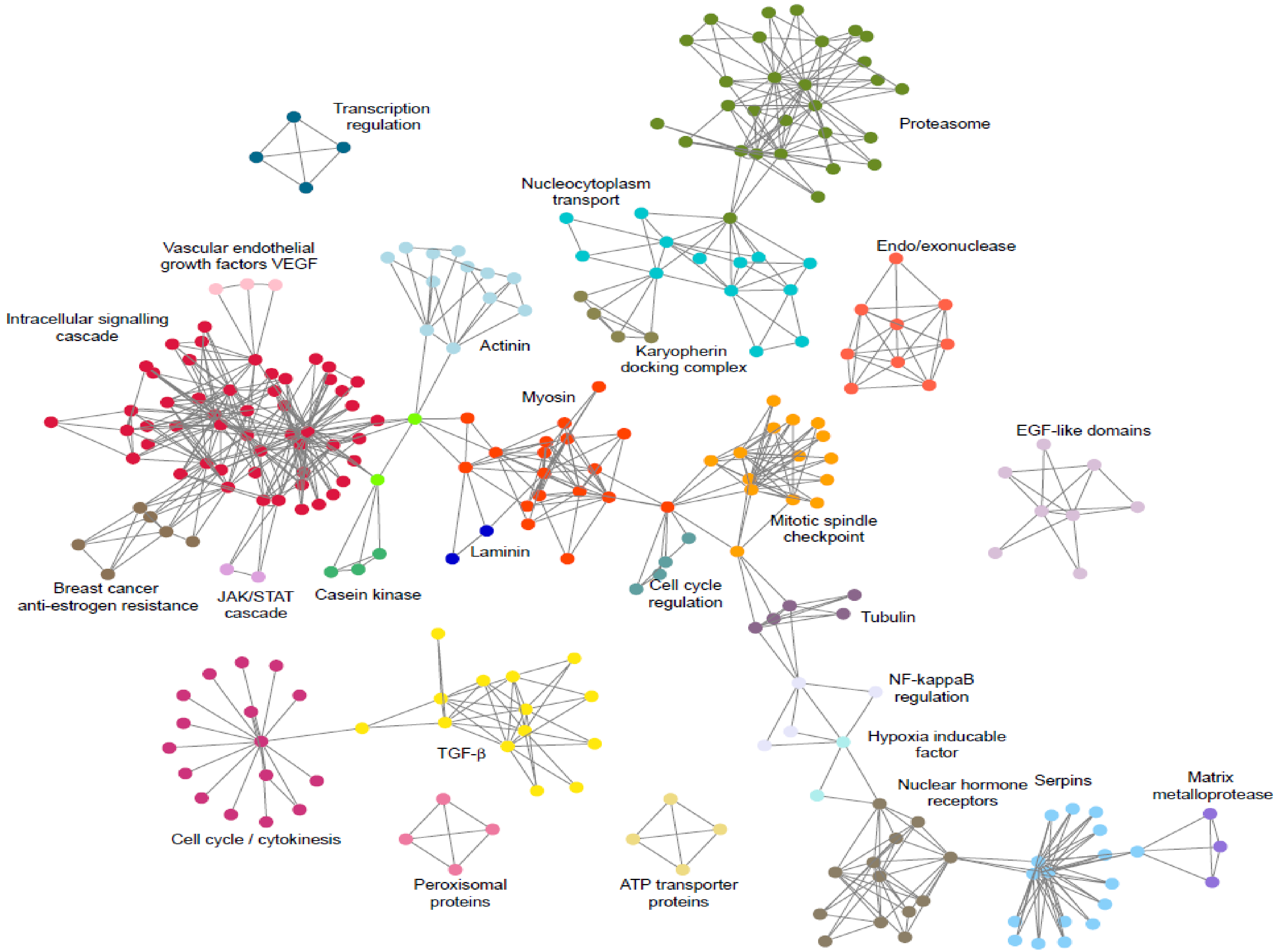
Can we identify functional modules?

Nodes: Proteins
Edges: Physical interactions

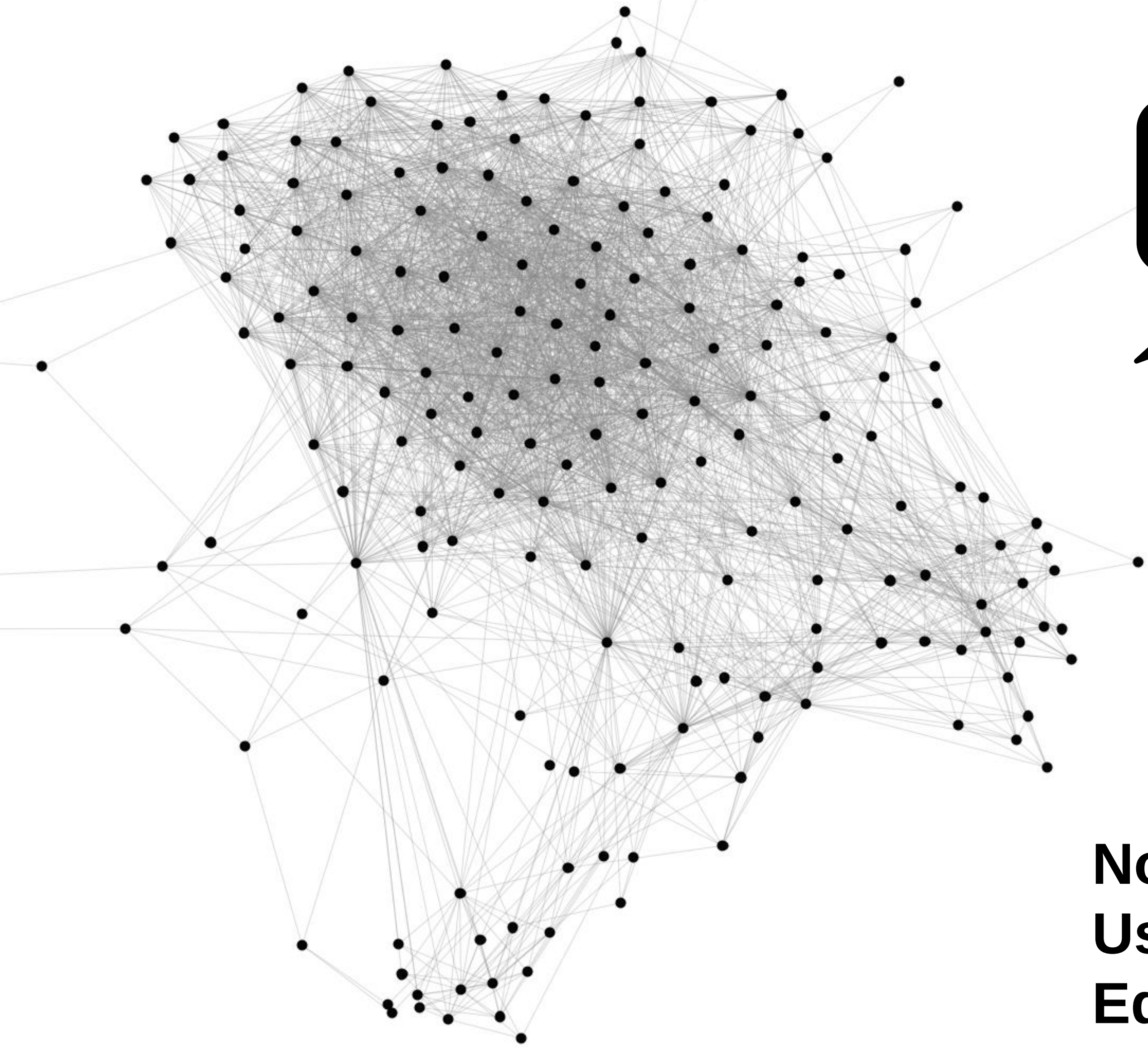


Functional modules

Nodes: Proteins
Edges: Physical interactions

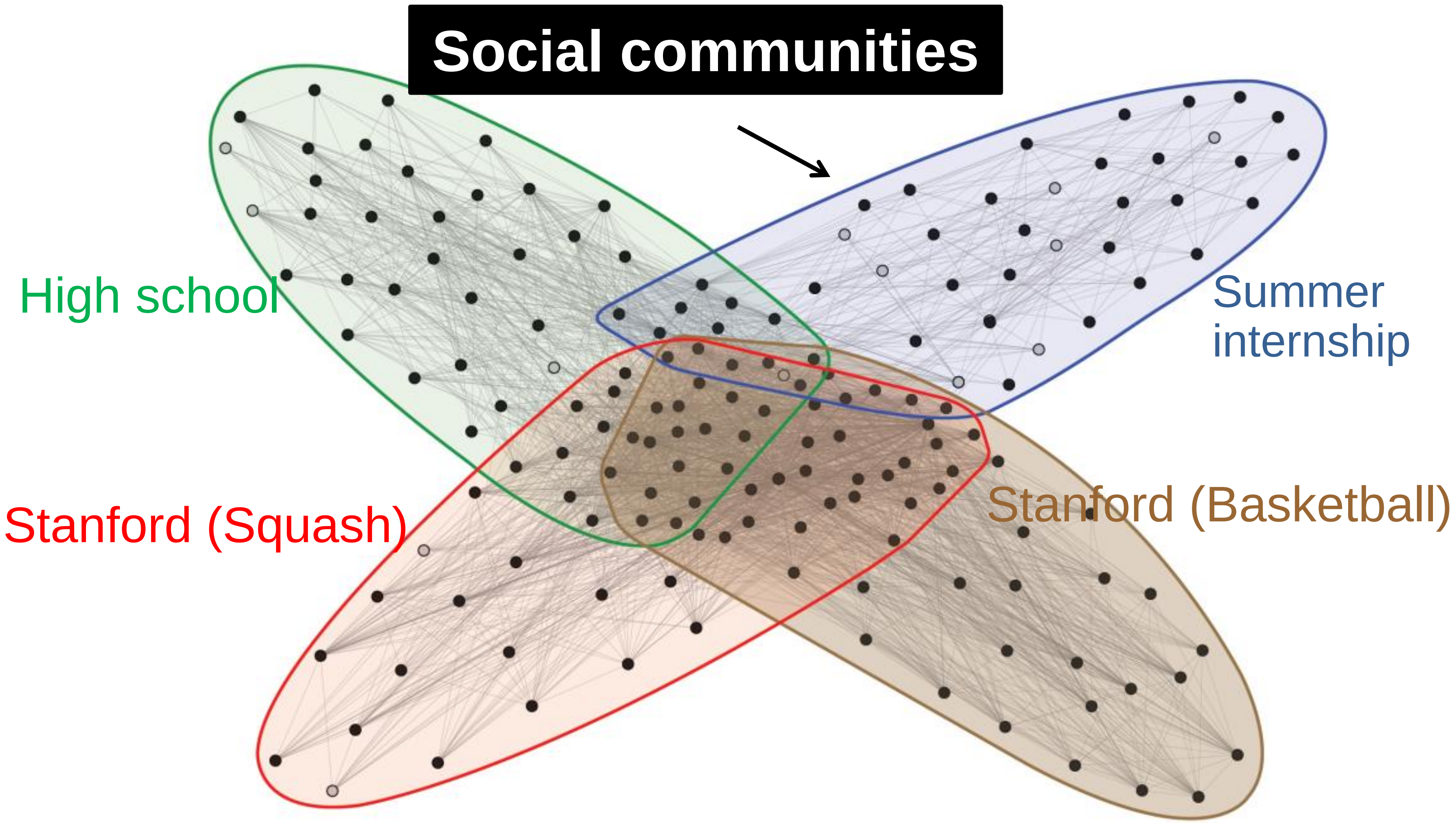

Transcription regulation
Proteasome
Nucleocytoplasm transport
Vascular endothelial growth factors VEGF
Endo/exonuclease
Intracellular signalling cascade
Actinin
Karyopherin docking complex
Myosin
EGF-like domains
Mitotic spindle checkpoint
Laminin
Breast cancer anti-estrogen resistance
JAK/STAT cascade
Casein kinase
Cell cycle regulation
Tubulin
NF-kappaB regulation
Hypoxia inducable factor
TGF-β
Cell cycle / cytokinesis
Nuclear hormone receptors
Serpins
Matrix metalloprotease
Peroxisomal proteins
ATP transporter proteins

Stanford Facebook network



Can we identify social communities?

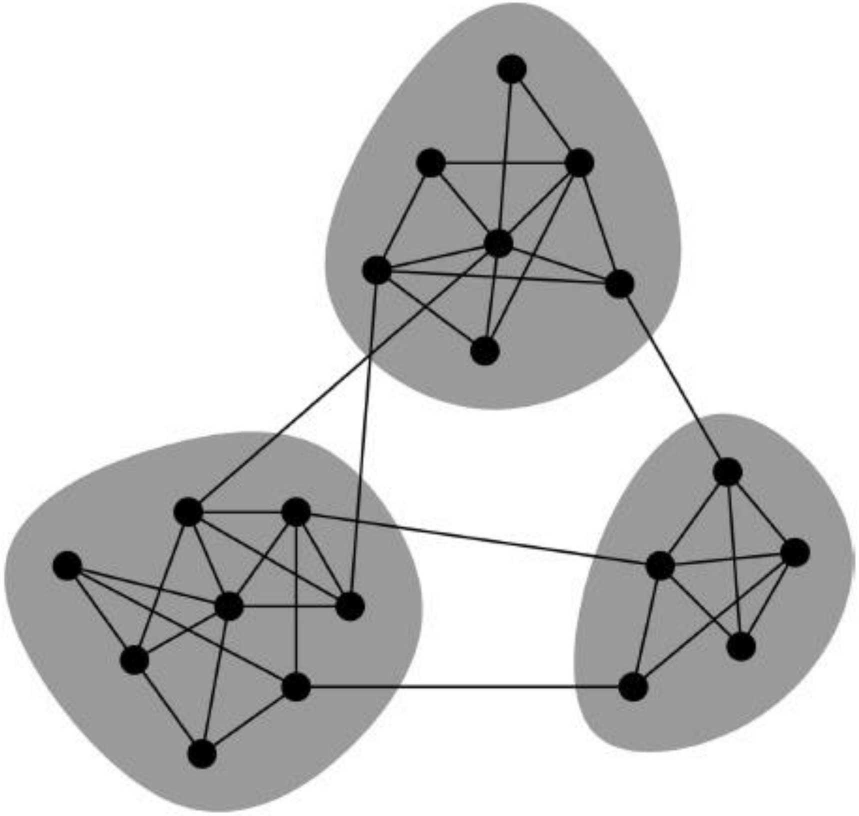
Nodes: Facebook Users
Edges: Friendships

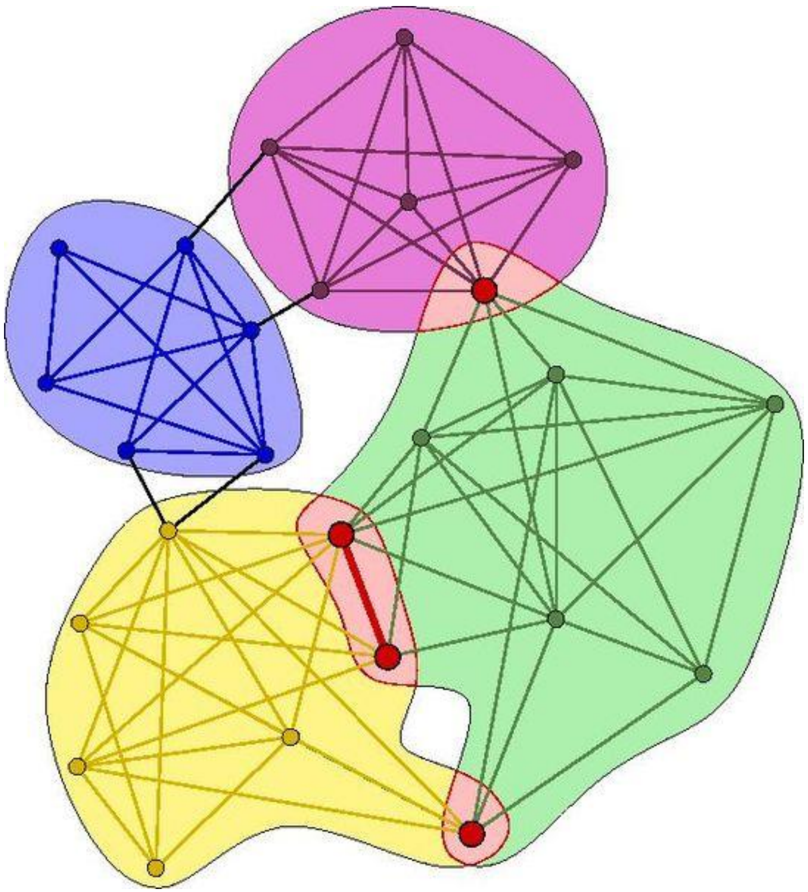


Nodes: Facebook Users
Edges: Friendships

Community types

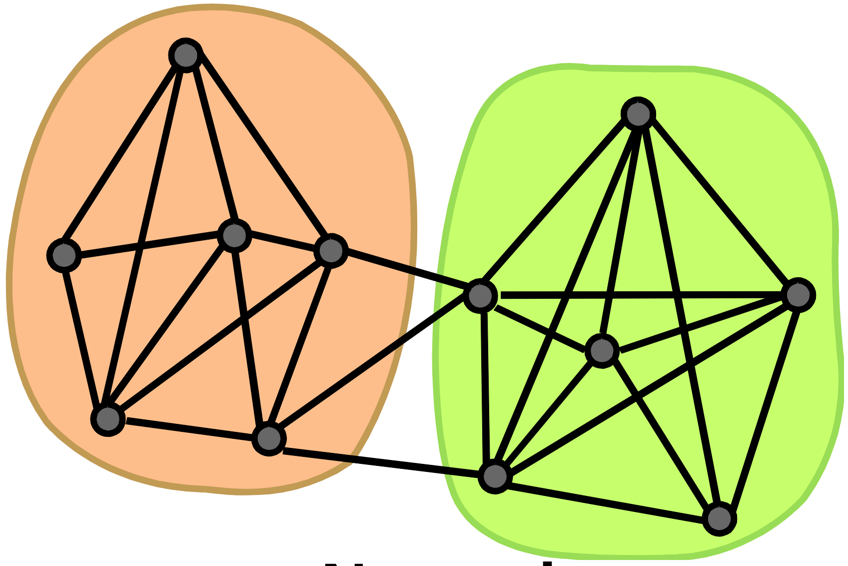
- Overlapping communities vs non-overlapping communities



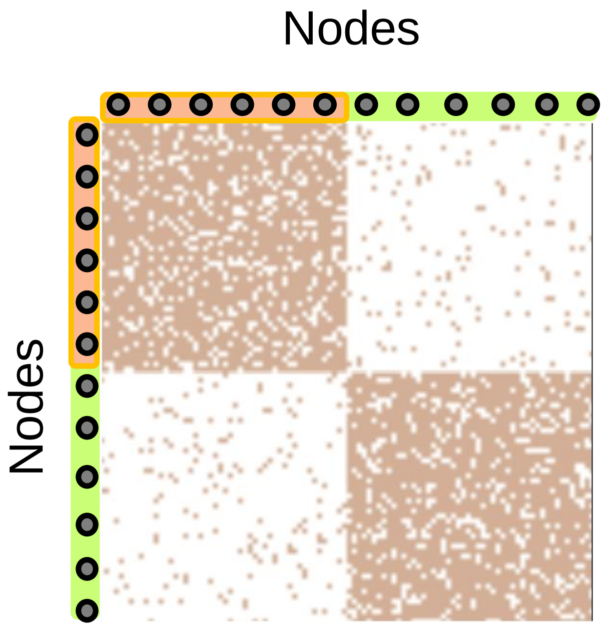


Non-Overlapping communities

- Dense connectivity within the community, sparse across communities

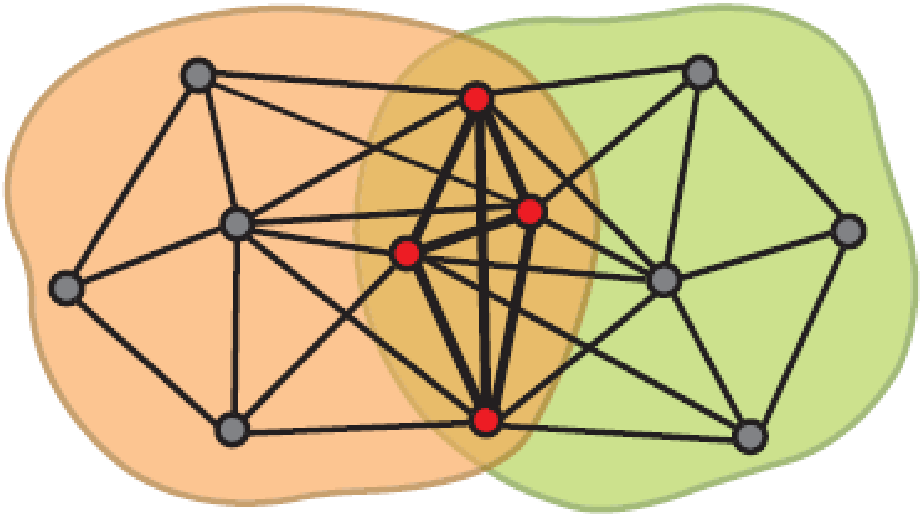


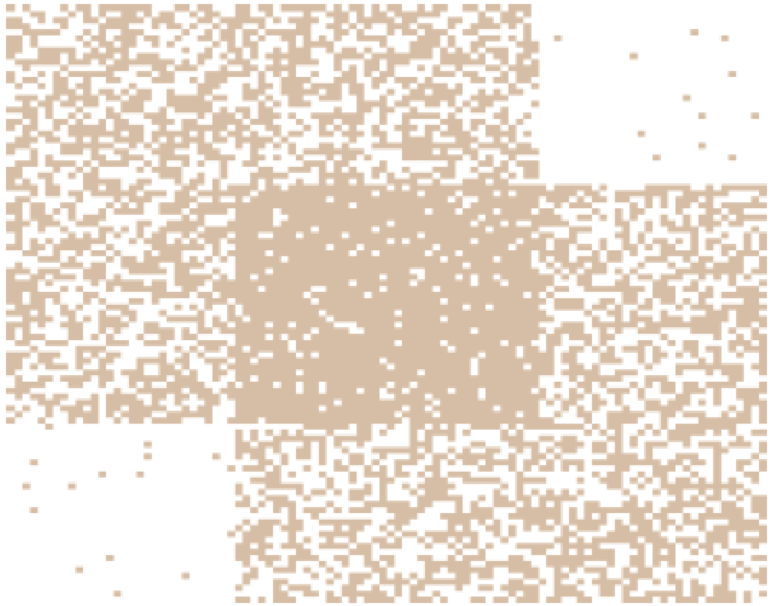
Network



Adjacency matrix

Overlapping communities



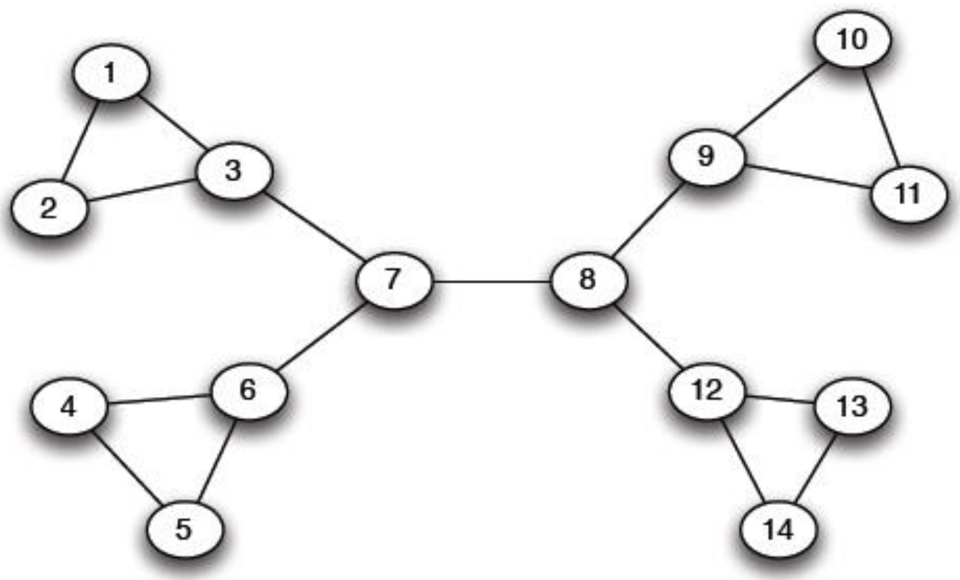


Community detection as clustering

- In many ways community detection is just clustering on graphs.
- We can apply clustering algorithms on the adjacency matrix (e.g., k-means)
- We can define a distance or similarity measure between nodes in the graph and apply other algorithms (e.g., hierarchical clustering)
  - Similarity using jaccard similarity on the neighbors sets
  - Distance using shortest paths or random walks.
- There are also algorithms that are specific to graphs

The Girvan-Newman method

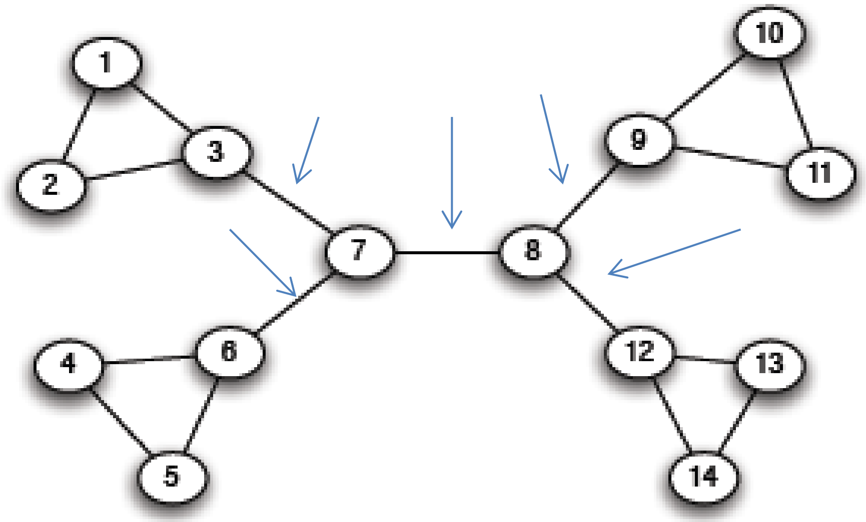
- Hierarchical divisive method
  - Start with the whole graph
  - Find edges whose removal "partitions" the graph
  - Repeat with each subgraph until single vertices



Which edge to remove?

The Girvan-Newman method

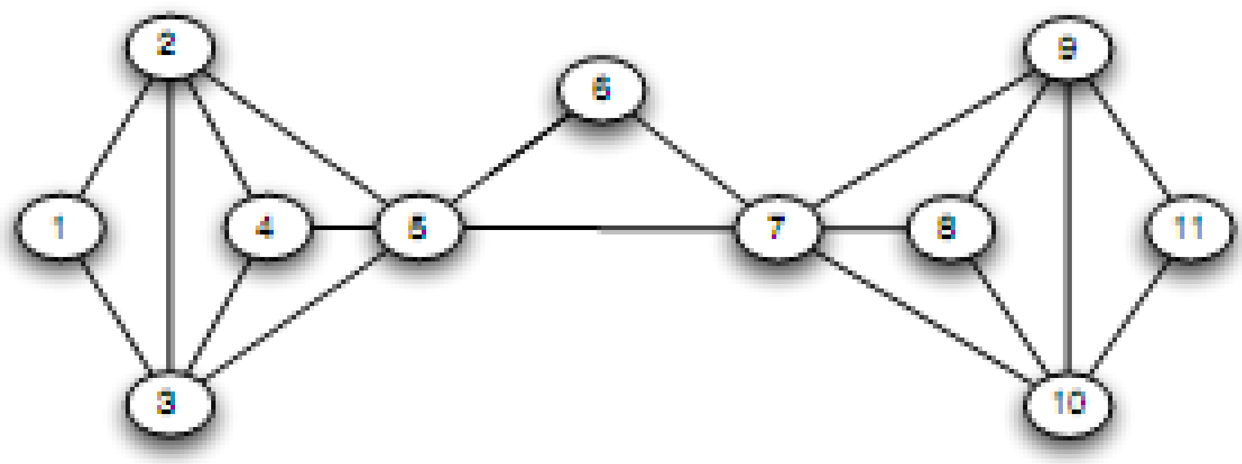
- Select cut-edges (a.k.a. bridge edges): edges that when removed they disconnect the graph



- There may be many of those

The Girvan-Newman method

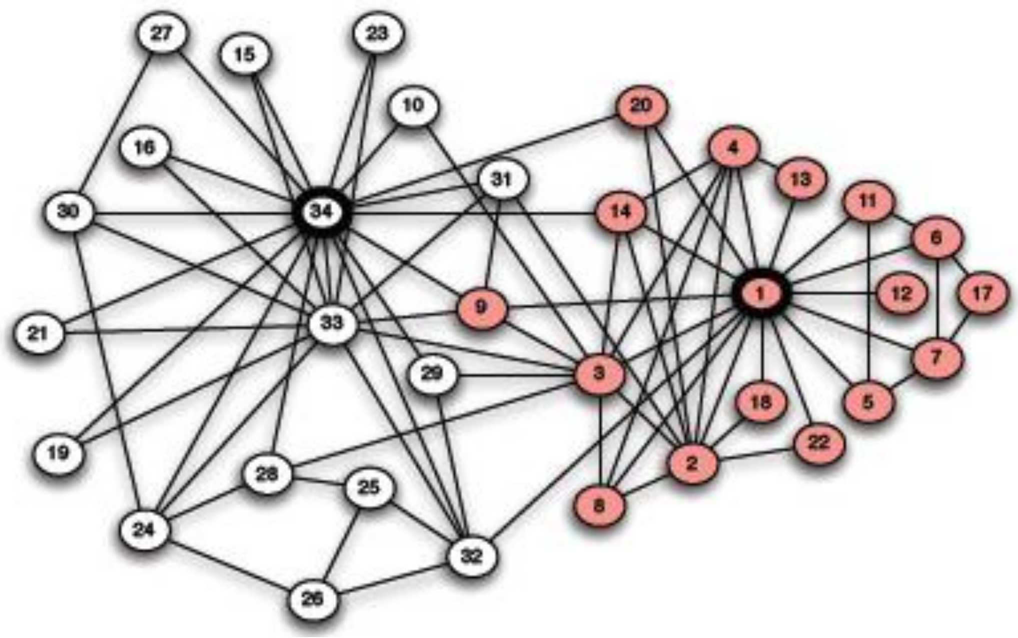
- Select cut-edges (a.k.a. bridge edges): edges that when removed they disconnect the graph



- Or, more often, there may be none

The Girvan-Newman method

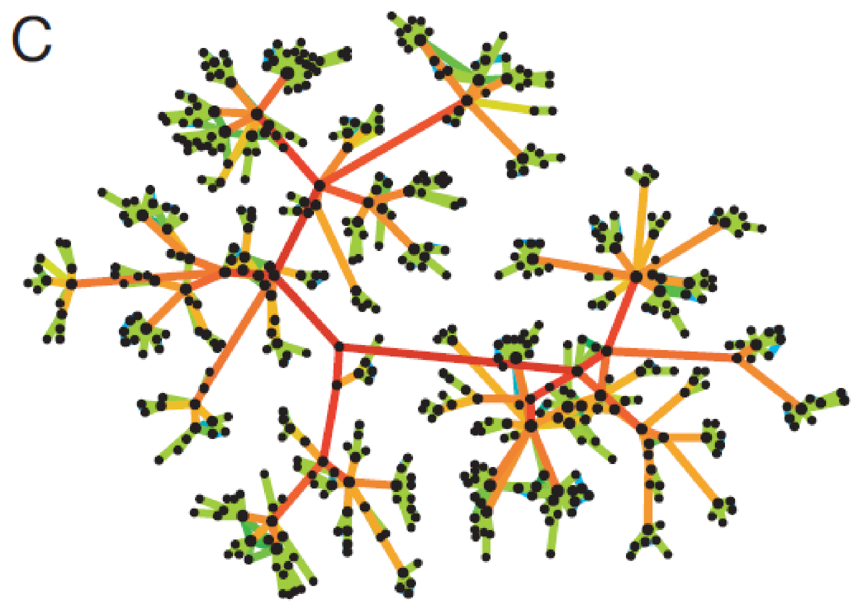
- Select cut-edges (a.k.a. bridge edges): edges that when removed they disconnect the graph



- Or, more often, there may be none

Edge importance

- We need a measure of how important an edge is in keeping the graph connected
- Edge betweenness: Number of shortest paths that pass through the edge



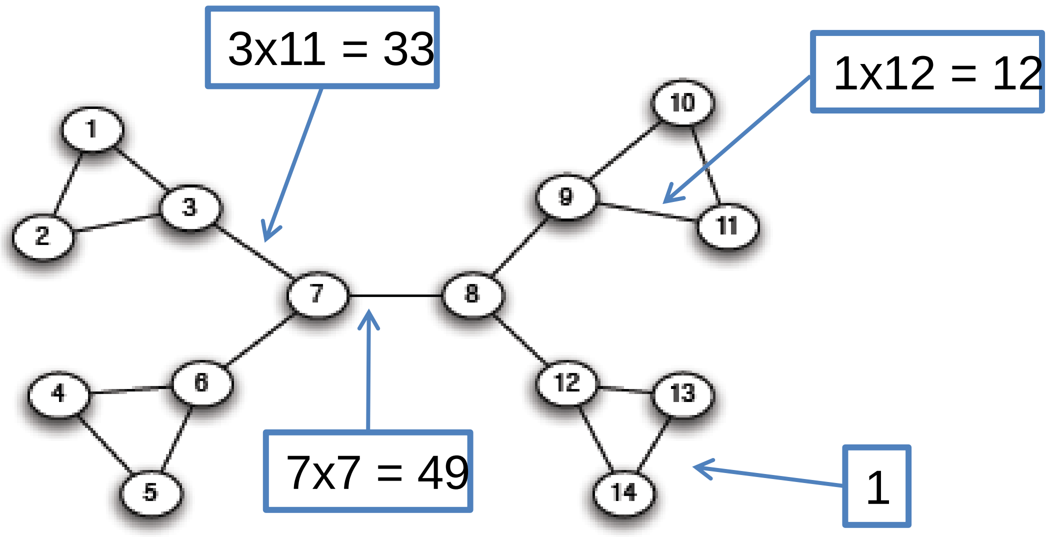
Edge Betweeness

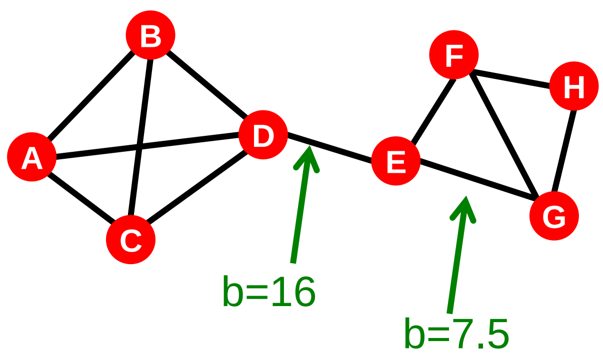
- Betweeness of edge (a, b) ($B(a, b)$):
  - For each pair of nodes x, y compute the number of shortest paths that include (a, b)
  - There may be multiple shortest paths between (x, y) ($SP(x, y)$). Compute the fraction of those that pass through (a, b)
    - Assumes a unit of traffic flow between (x, y)

$$B(a, b) = \sum_{x,y \in V} \frac{|SP(x, y)\ that\ include\ (a, b)|}{|SP(x, y)|}$$

- Betweenness computes the probability of an edge to occur on a randomly chosen shortest path between two randomly chosen nodes.

Examples


3x11 = 33
1x12 = 12
10
1
9
3
11
2
7
8
4
6
12
13
7x7 = 49
1
5
14

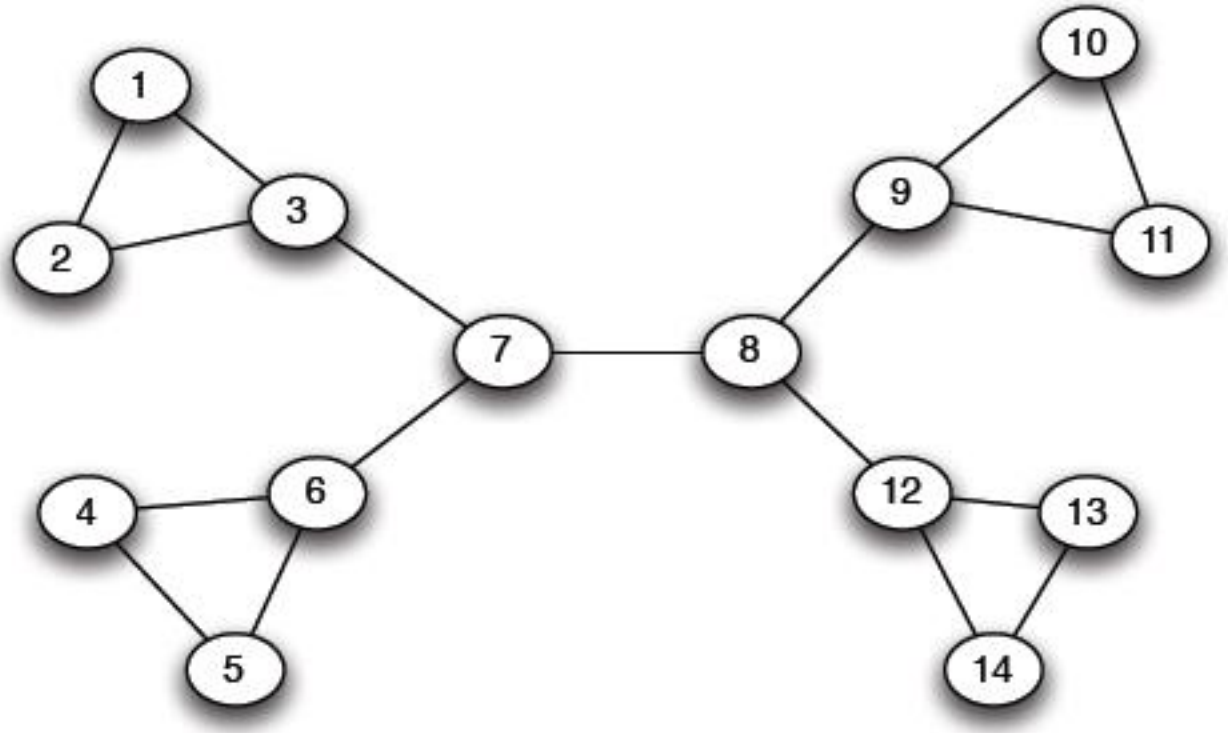

B
F
H
D
A
E
G
C
b=16
b=7.5

The Girvan Newman Algorithm

- Given an undirected unweighted graph:
- Repeat until no edges are left:
  - Compute the edge betweeness for all edges
  - Remove the edge with the highest betweeness

- At each step of the algorithm, the connected components are the communities
- Gives a hierarchical decomposition of the graph into communities

Girvan Newman method: An example

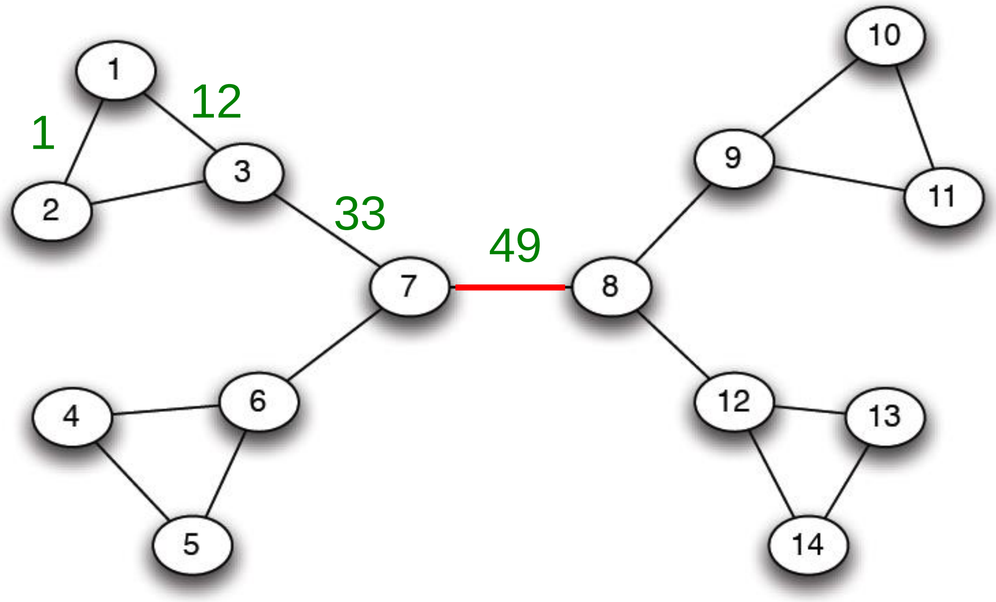


Betweenness(7, 8)= 7x7 = 49

Betweenness(1, 3) = 1X12=12

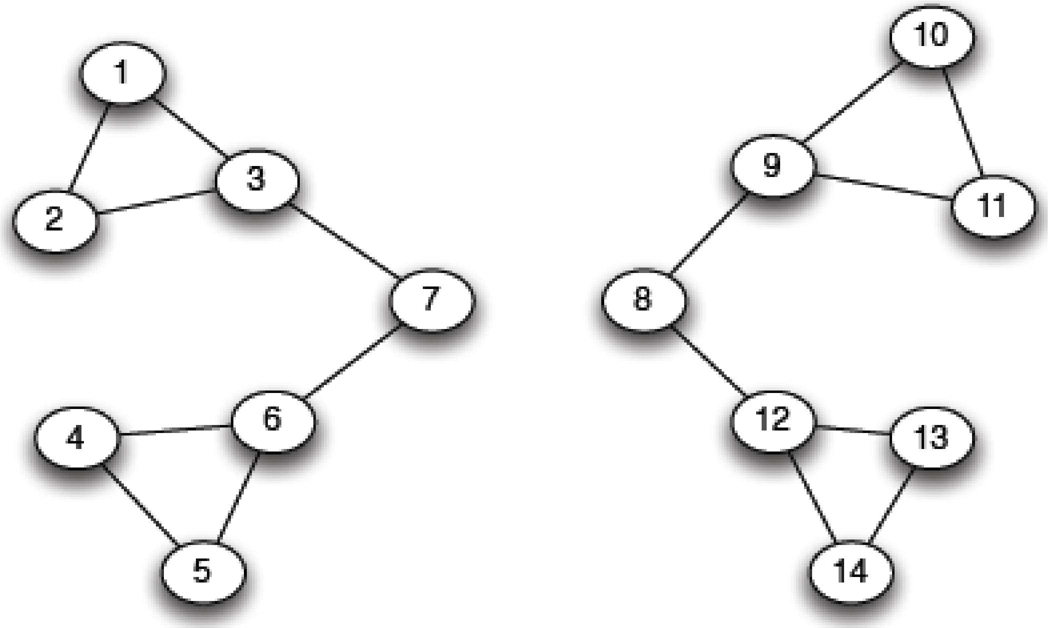
Betweenness(3, 7) = Betweenness(6, 7) =
Betweenness(8, 9) = Betweenness(8, 12)= 3X11=33

Girvan-Newman: Example



Need to re-compute betweenness at every step

Girvan Newman method: An example

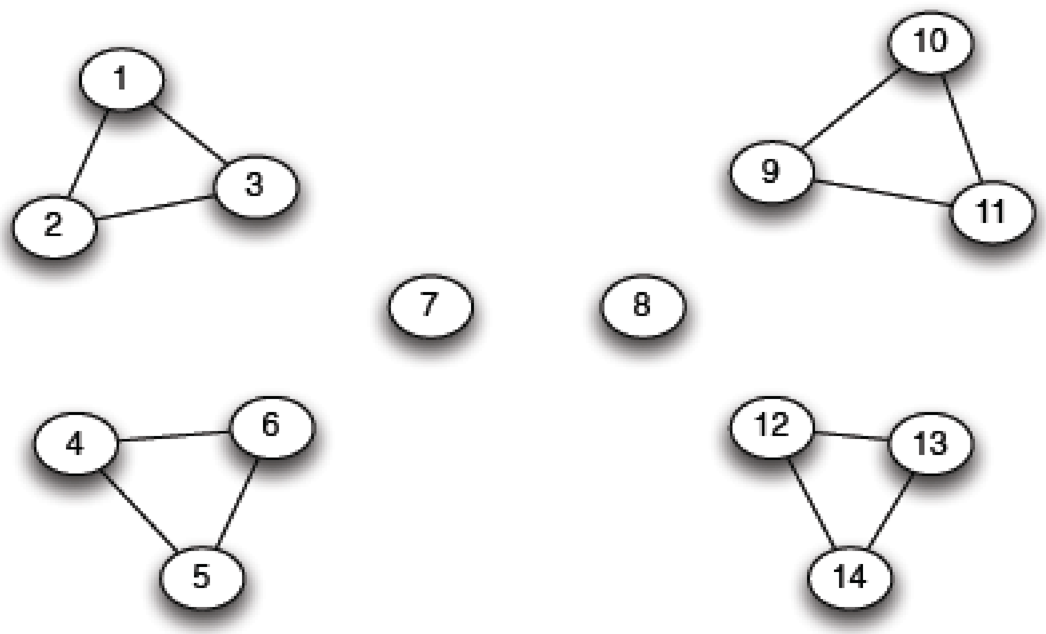


(a) *Step 1*

Betweenness(1, 3) = 1X5=5

Betweenness(3,7) = Betweenness(6,7) =
Betweenness(8,9) = Betweenness(8,12) = 3X4=12

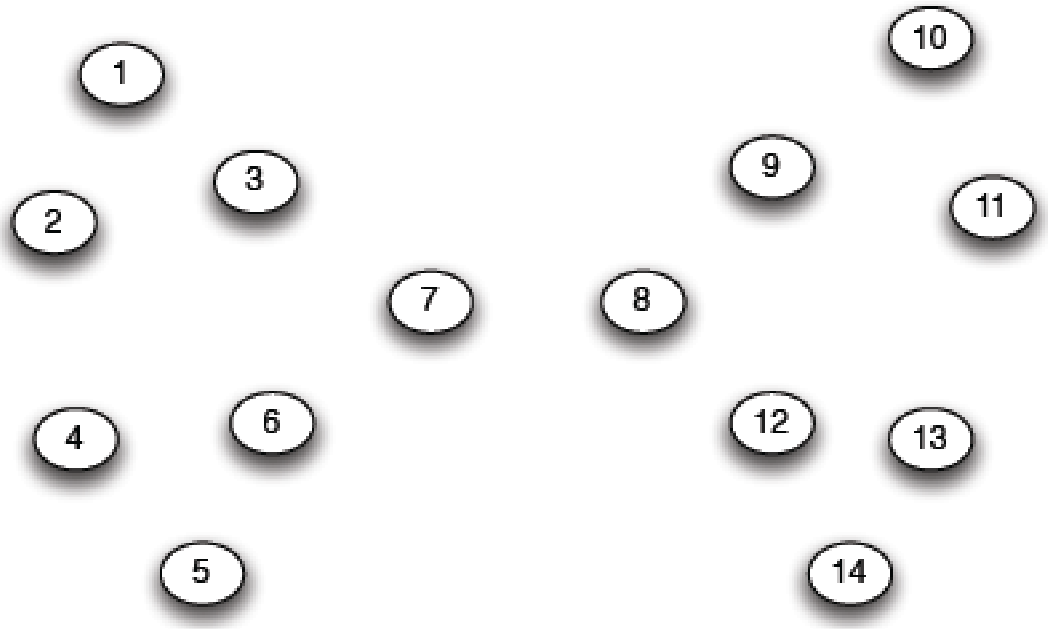
Girvan Newman method: An example



(b) *Step 2*

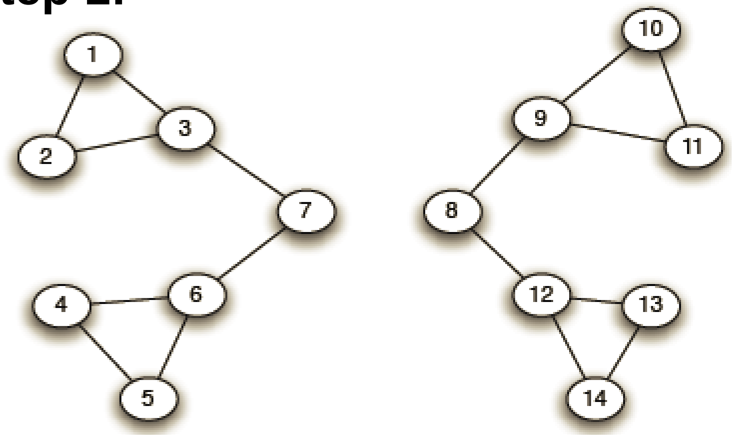
Betweenness of every edge = 1

Girvan Newman method: An example

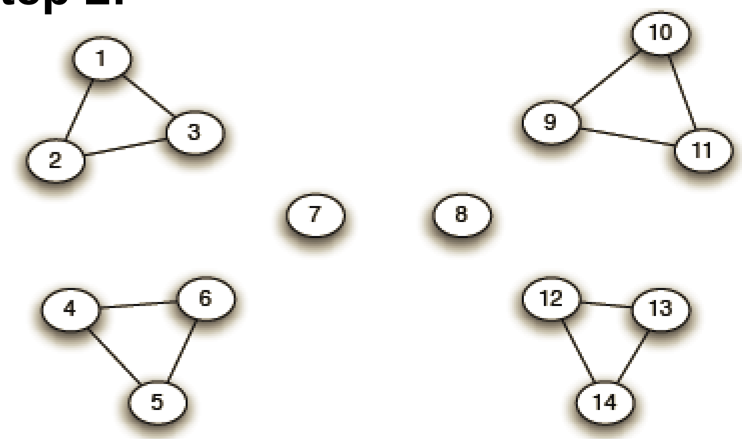


Girvan-Newman: Example

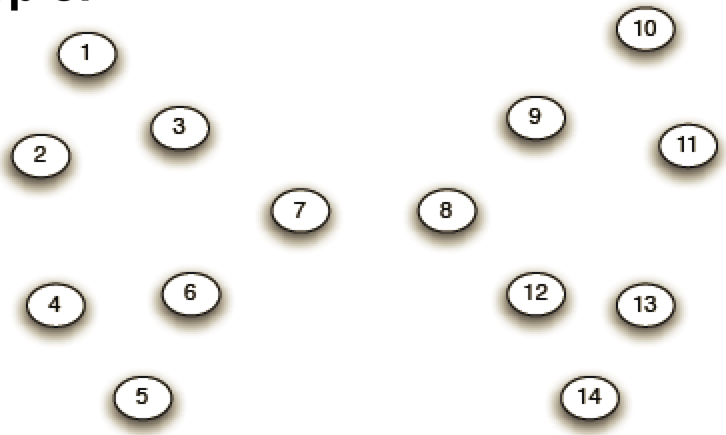
Step 1:



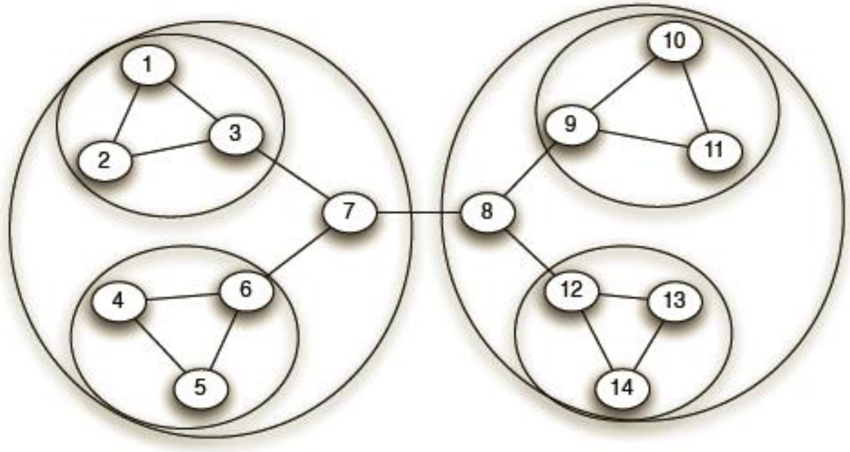
Step 2:



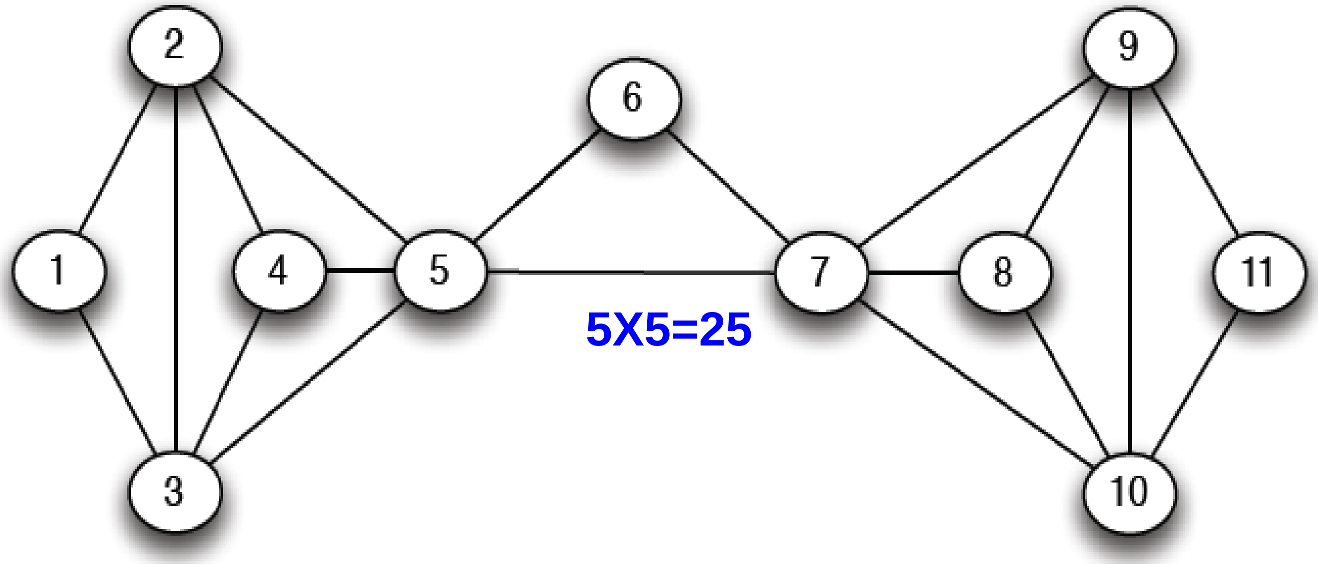
Step 3:



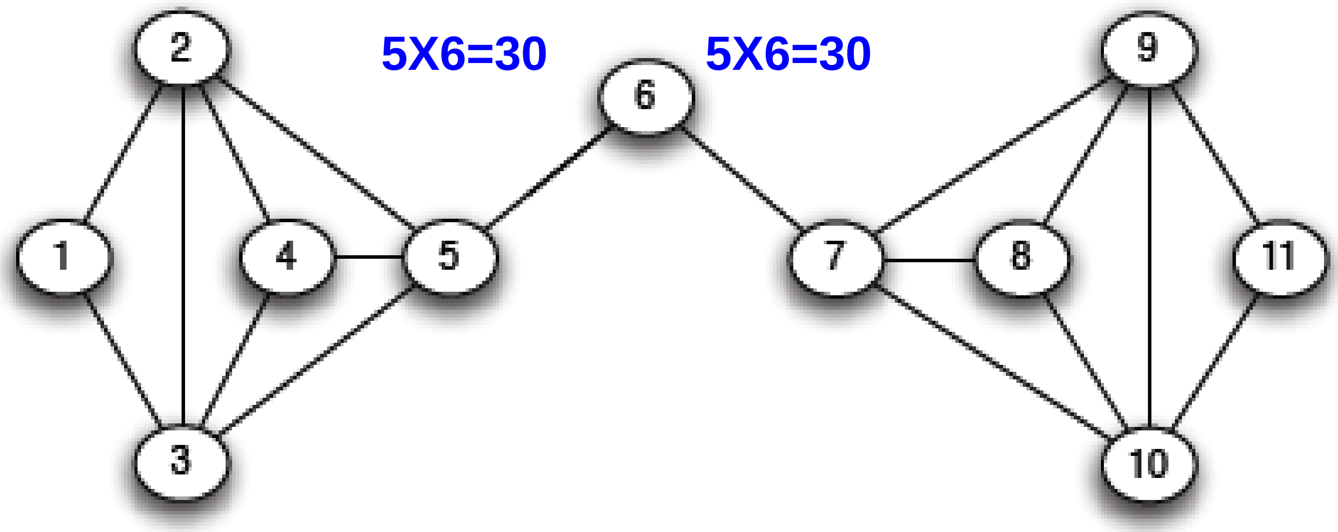
Hierarchical network decomposition:



Another example

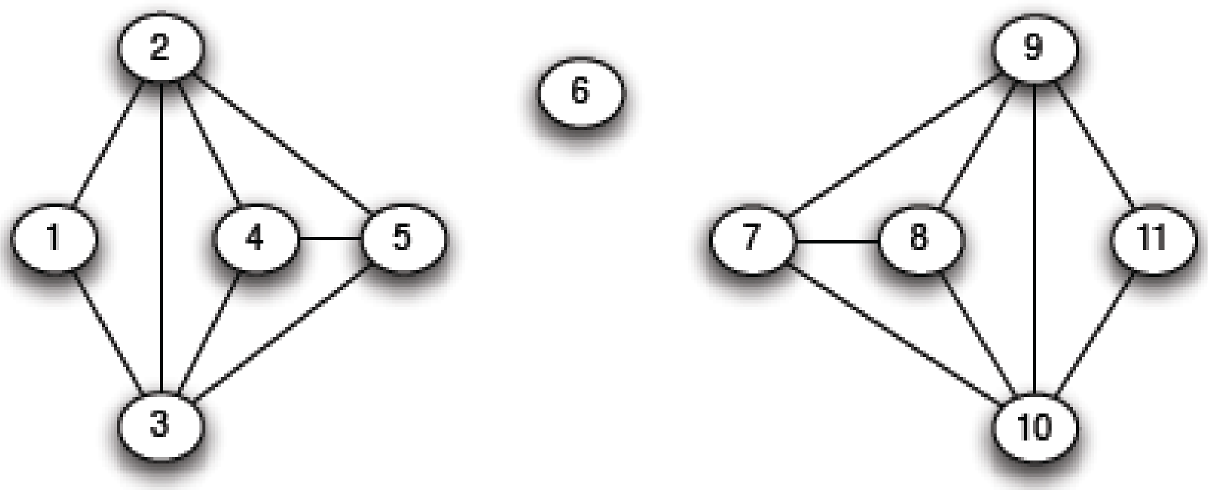


Another example



(a) *Step 1*

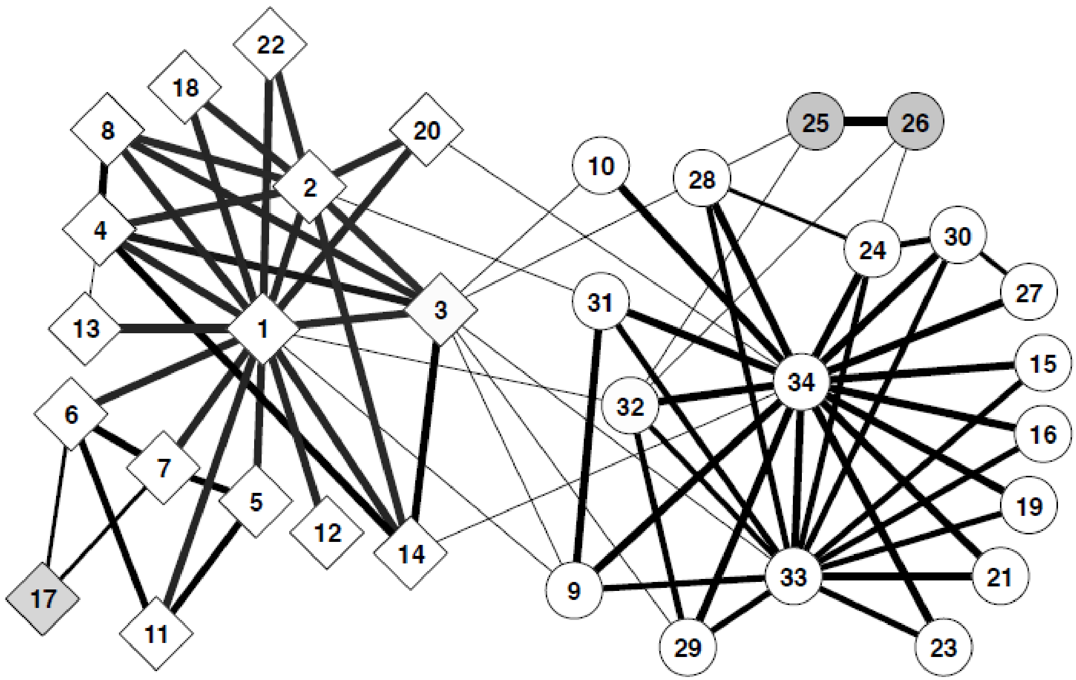
Another example

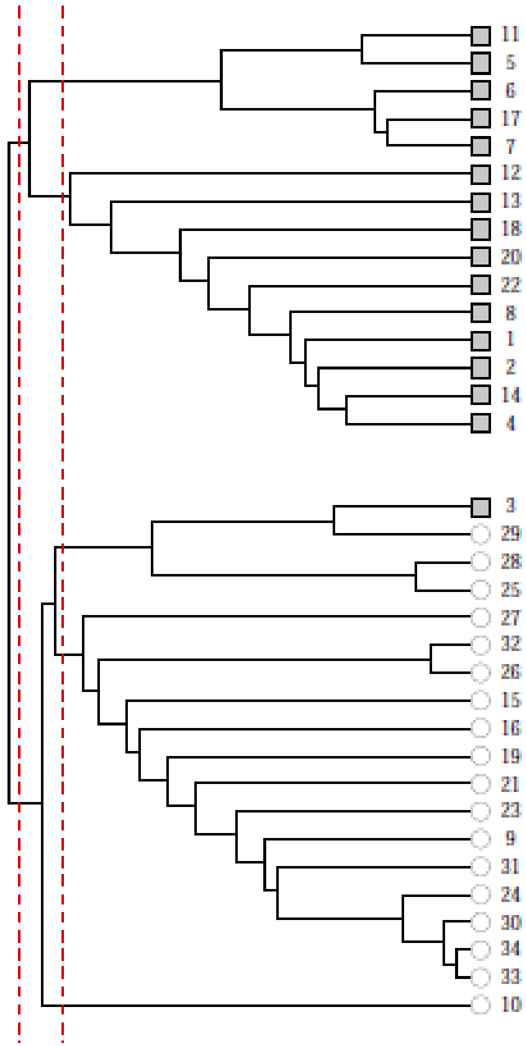


(b) *Step 2*

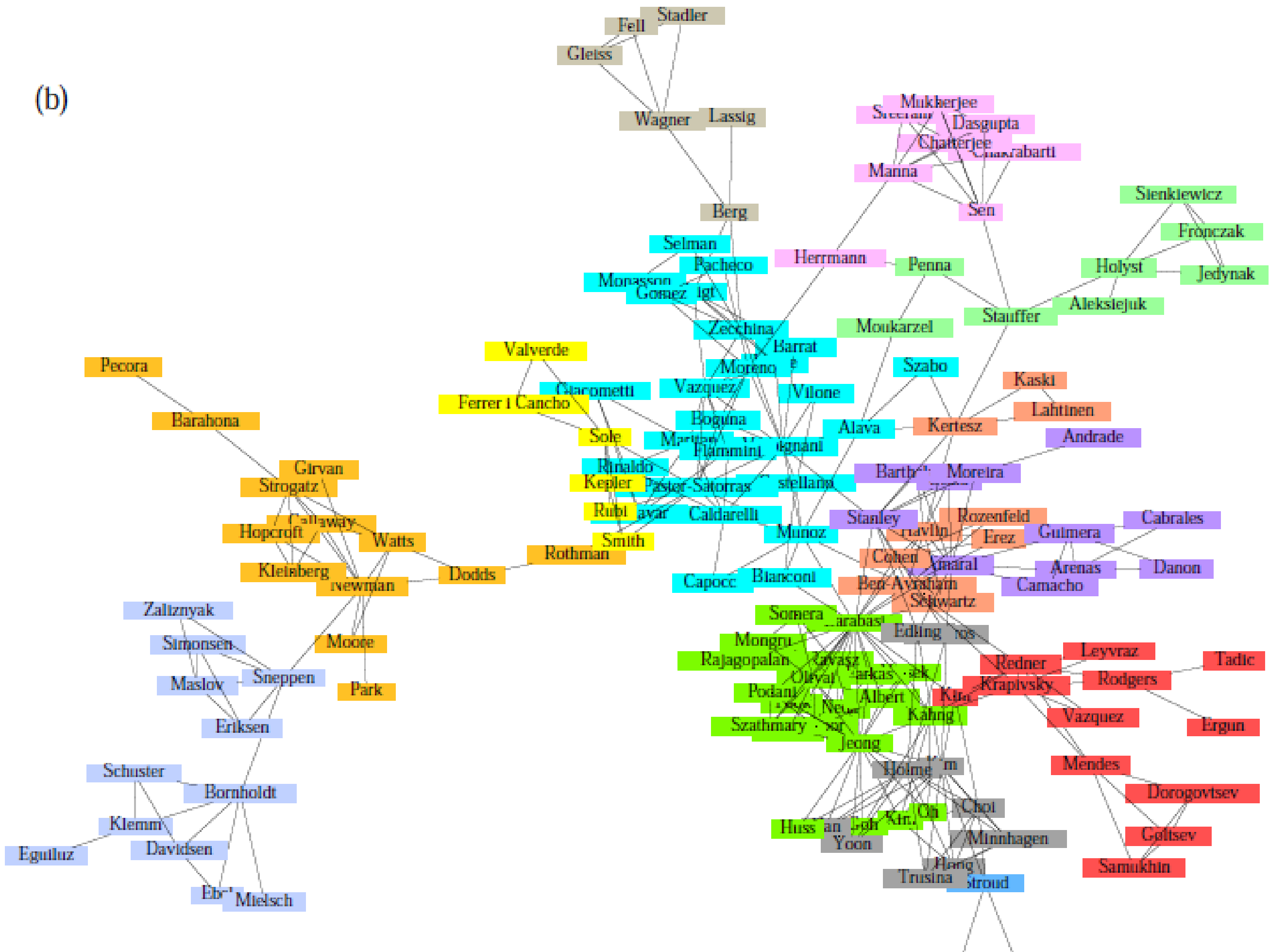
Girvan-Newman: Results

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





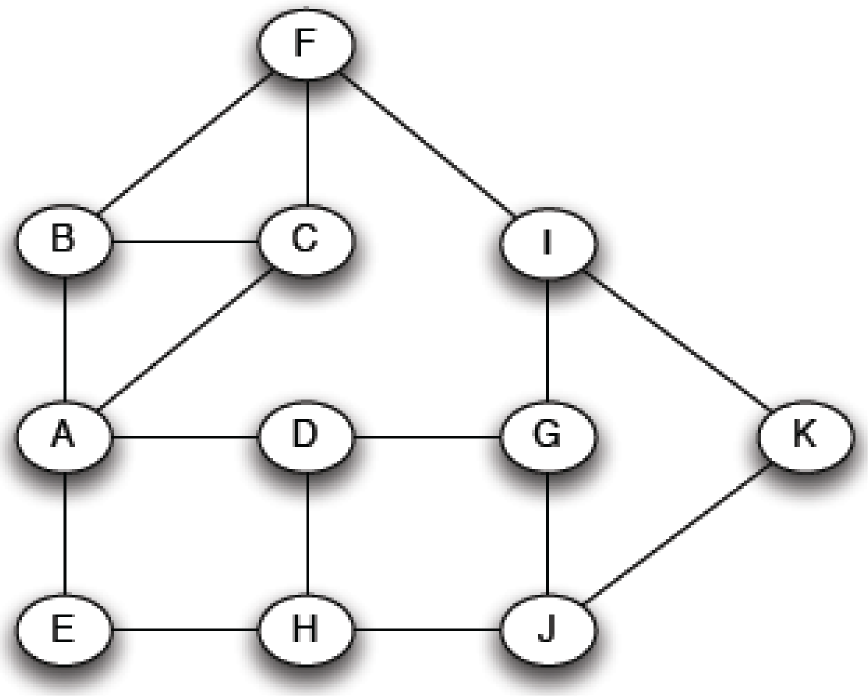
Girvan-Newman: Results



Communities in physics collaborations

How to Compute Betweenness?

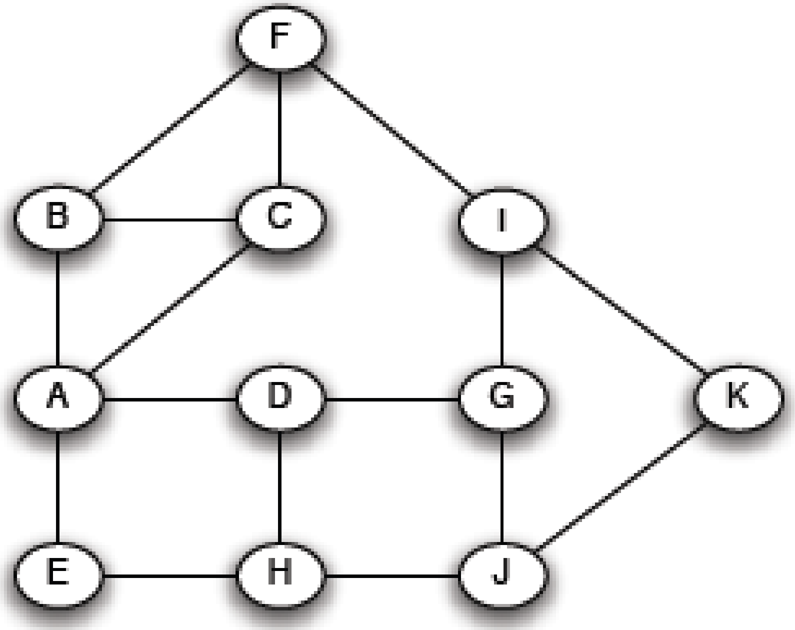
- Want to compute betweenness of paths starting from node A



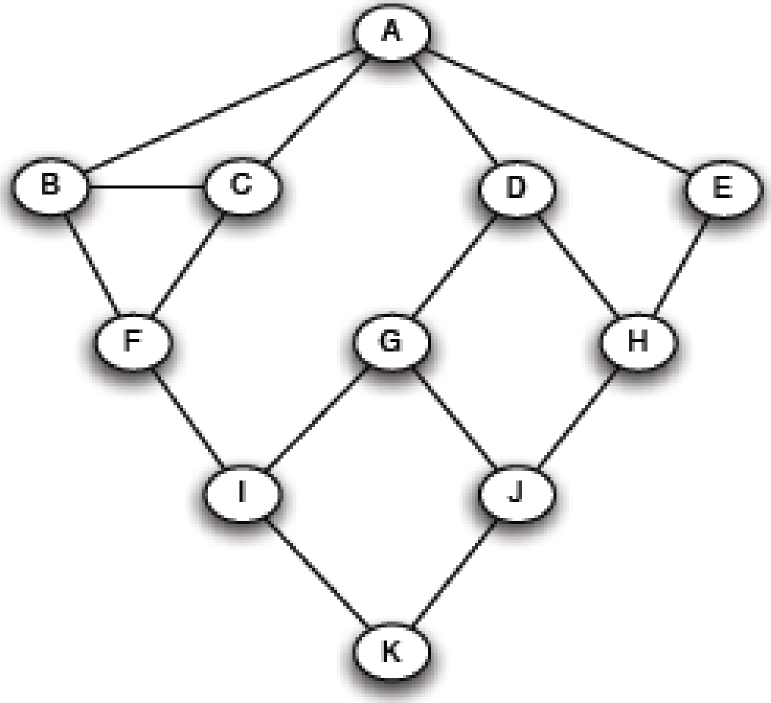
Computing Betweenness

1. Perform a *BFS* starting from A
2. Determine the number of shortest path from A to each other node
3. Based on these numbers, determine the amount of flow from A to all other nodes that uses each edge

Computing Betweenness: step 1



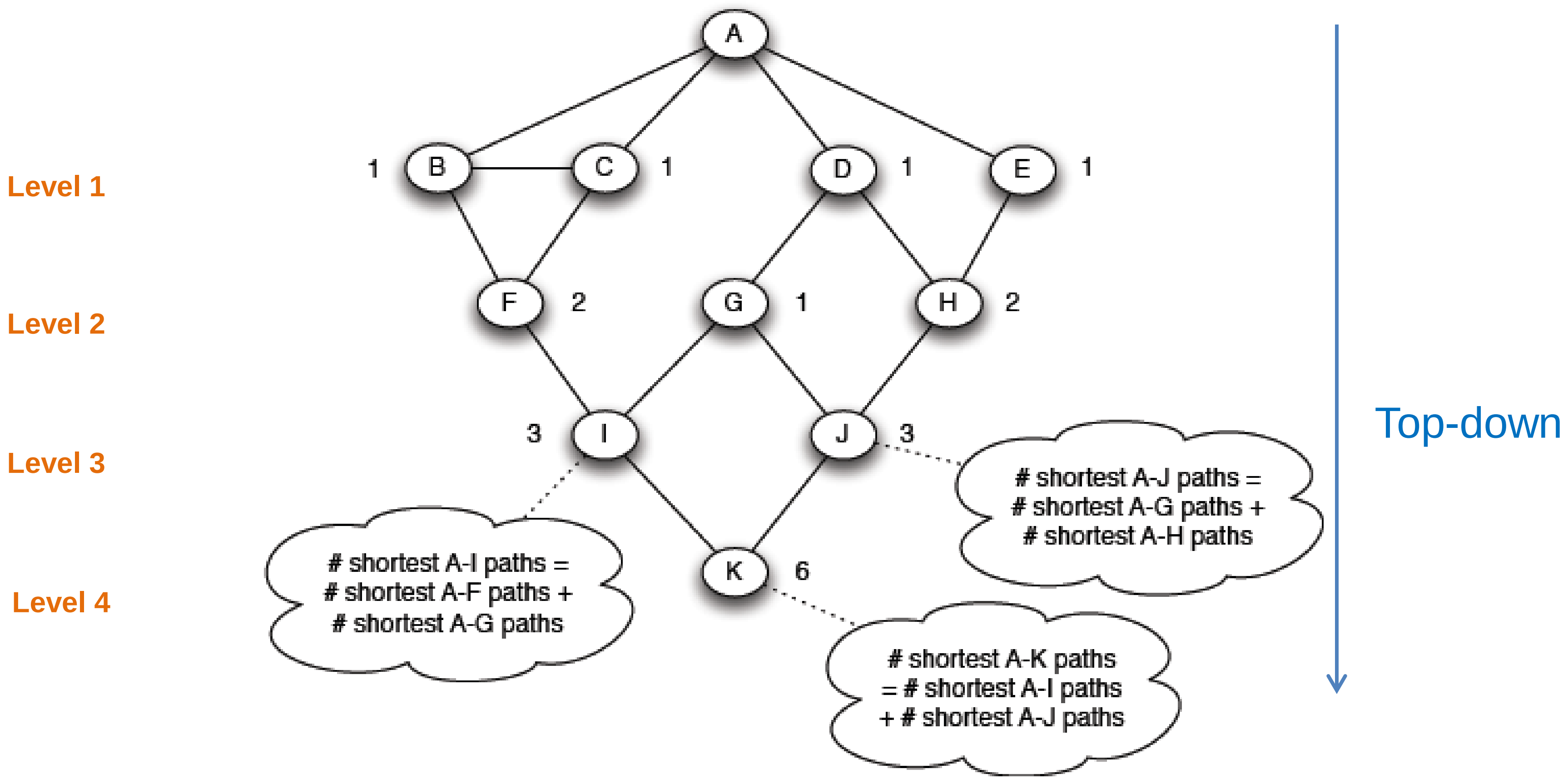
Initial network



BFS from A

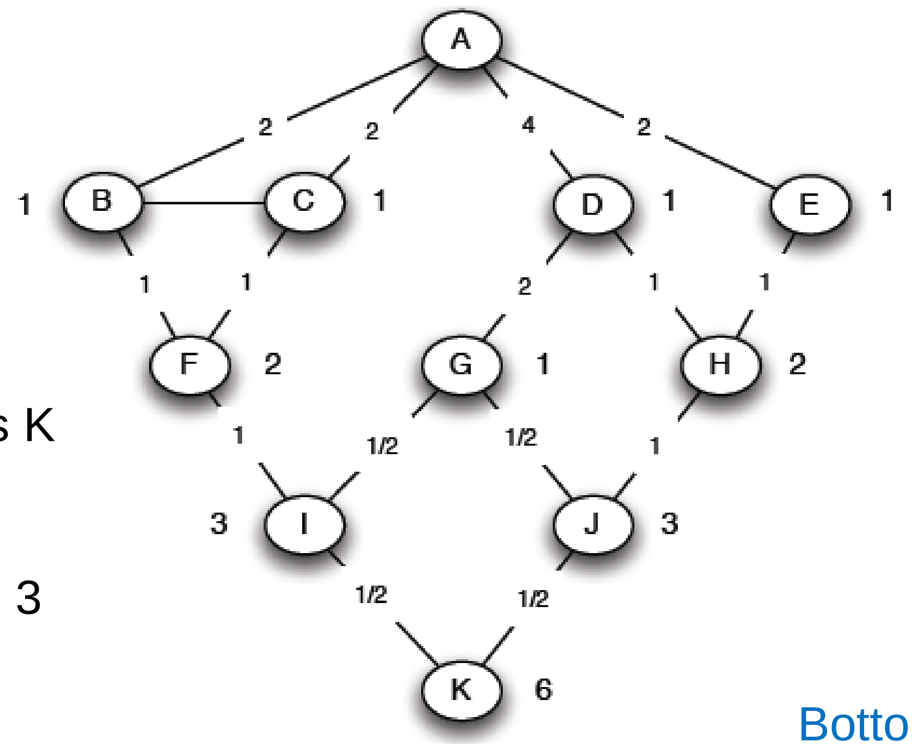
Computing Betweenness: step 2

- Count how many shortest paths from A to a specific node



Computing Betweeness: Step 3

- Compute betweenness by working up the tree:
  - For every node there is a unit of flow destined for that node that it is divided fractionally to the edges that reach that node



Bottom-up

There is a unit of flow to K that reaches K through edges (I,K) and (J,K)

Since there are 3 paths from I to K and 3 from J, each edge gets ½ of the flow: Betweeness ½

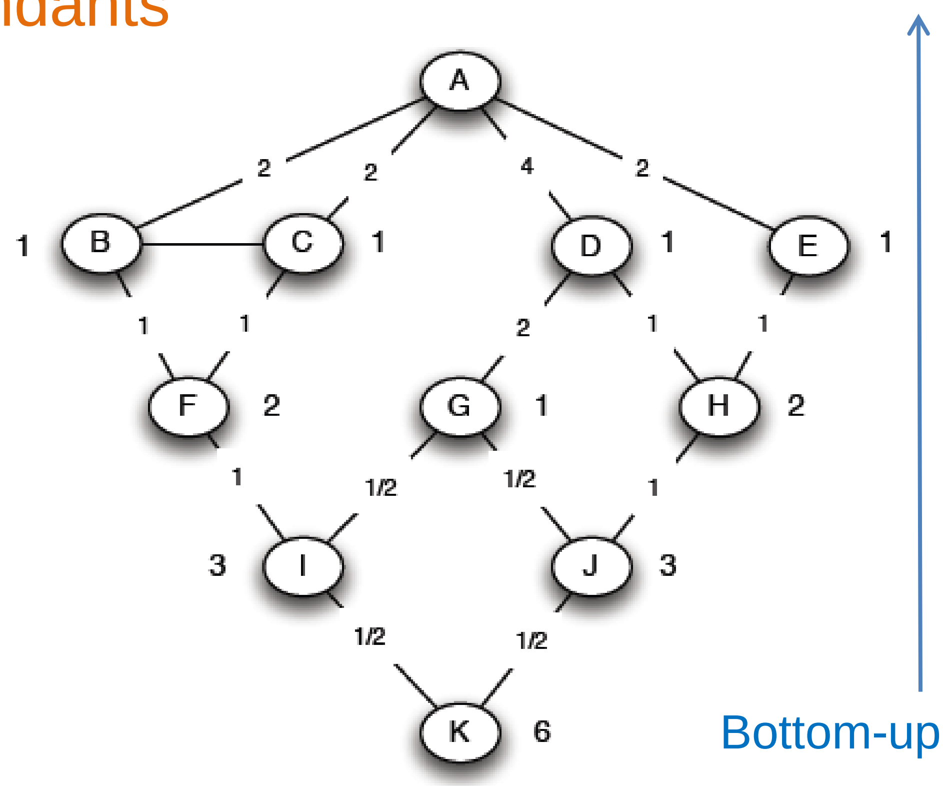
Computing Betweeness: Step 3

- Compute betweenness by working up the tree:
  - If the node has descendants in the BFS DAG, we also need to take into account the flow that passes from that node towards the descendants

For node I, there is a unit of flow to I from A, but also ½ of flow that passes from I towards K (we have computed that as the betweeness of edge (I,K)): Total flow 3/2

There are 2 paths from F to I and 1 path from G to I edge (F,I) gets 2/3 of the total flow: Betweeness 2/3*3/2 = 1

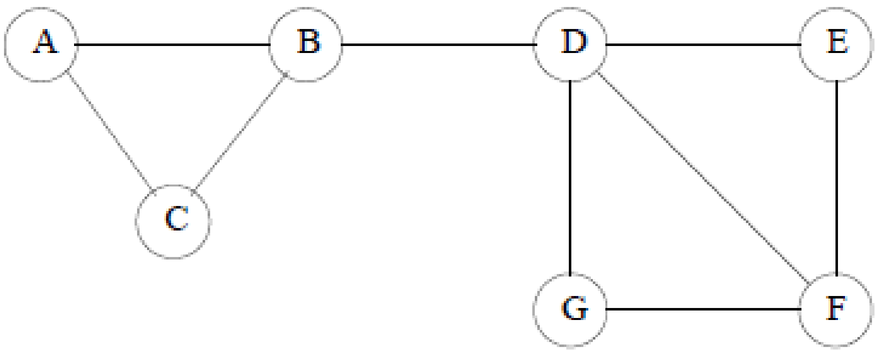
Edge (G,I) gets 1/3 of the total flow: Betweeness 2/3*3/2 = 1

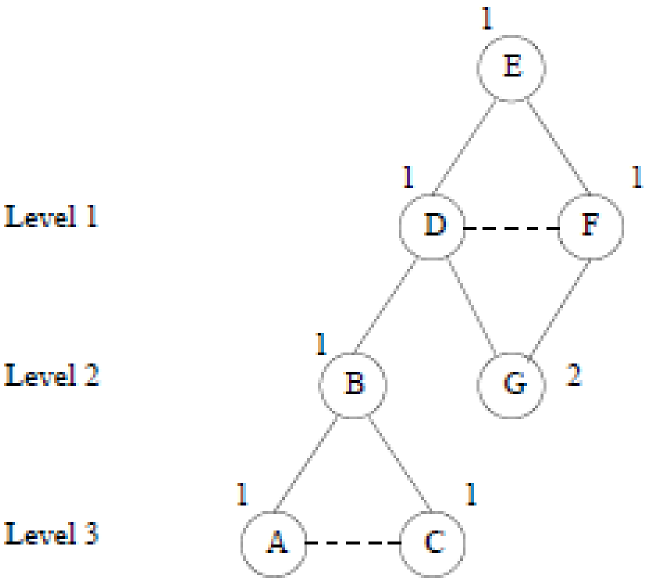


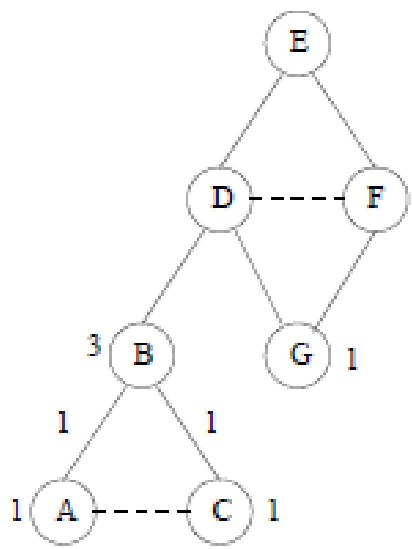
Computing Betweeness

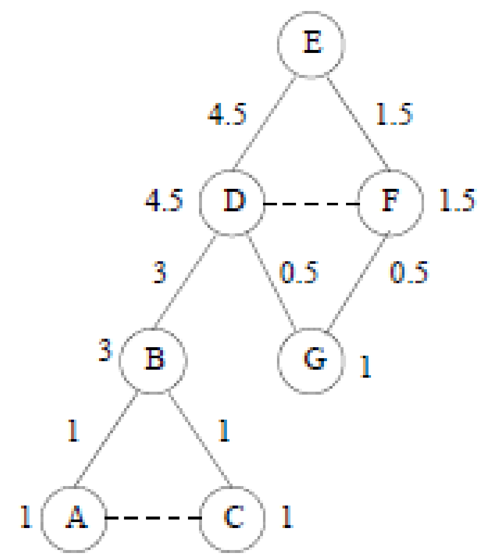
- Repeat the process for all nodes and take the sum

Example

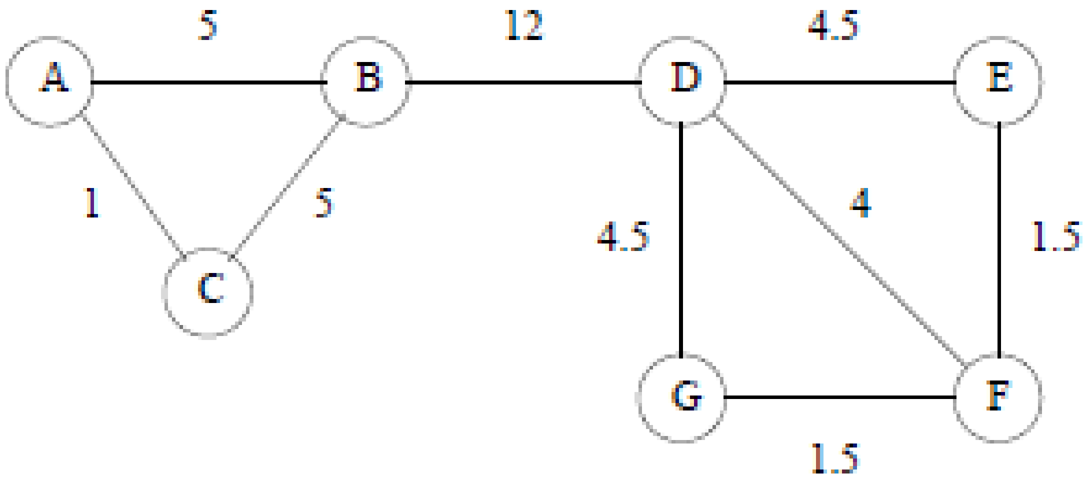

A
B
D
E
C
G
F

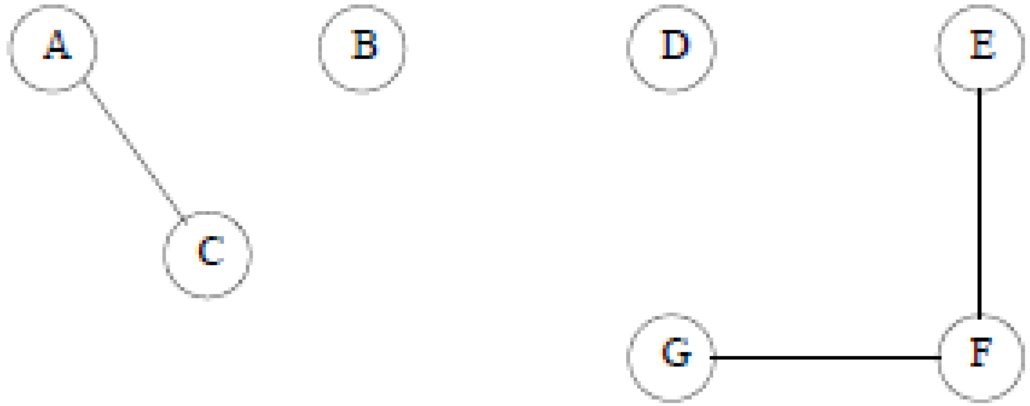

Level 1
Level 2
Level 3
E
D
F
B
G
A
C
1
1
1
1
2
1
1


E
D
F
B
G
A
C
3
1
1
1
1
1


E
D
F
B
G
A
C
4.5
1.5
4.5
1.5
3
0.5
0.5
3
1
1
1
1
1

Example





Computing Betweenness

- Issues
  - Scalability
  - Test for connectivity?
  - Re-compute all paths, or only those affected
  - Parallel computation
  - Sampling