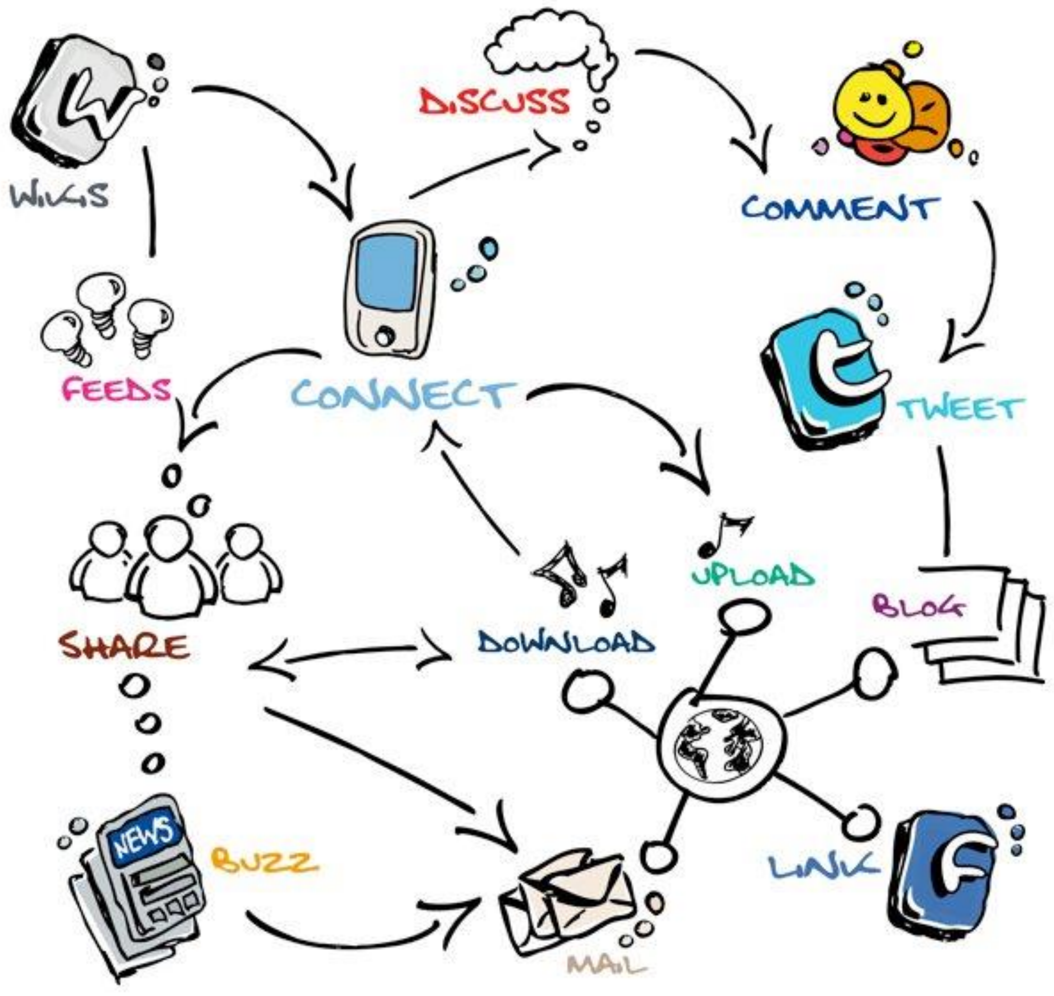


Community Detection in Networks

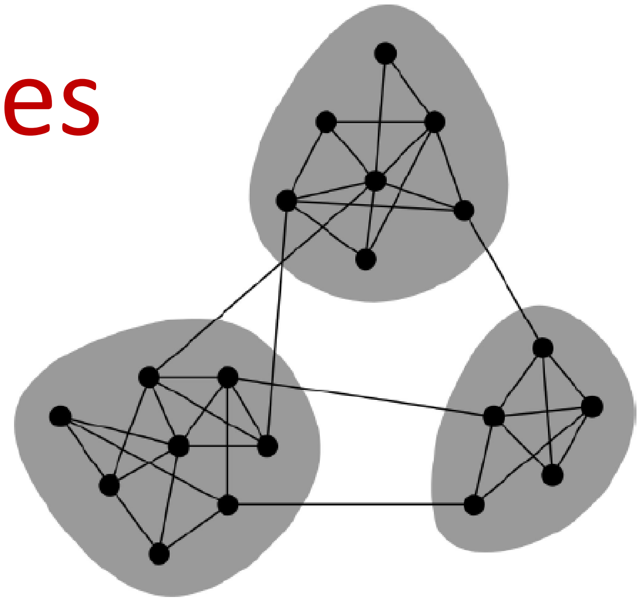
Ahmet Onur Durahim

How the Class Fits Together

Properties	Models	Algorithms
Small diameter, Edge clustering	Small-world model, Erdös-Renyi model	Decentralized search
Scale-free	Preferential attachment, Copying model	PageRank, Hubs and authorities
Strength of weak ties, Core-periphery	Kronecker Graphs	Community detection: Girvan-Newman, Modularity
Densification power law, Shrinking diameters	Microscopic model of evolving networks	Link prediction, Supervised random walks
Complex Graph Structure	Graph Neural Networks	Node Classification Graph Representation Learning
Information virality, Memetracking	Independent cascade model, Game theoretic model	Influence maximization, Outbreak detection, LIM

Network Communities

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

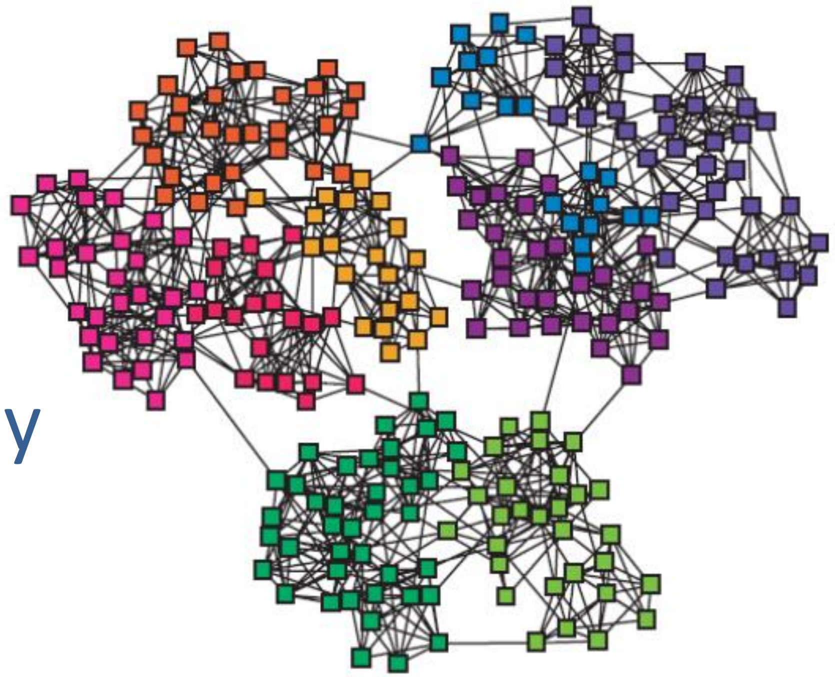


Communities, clusters, groups, modules

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

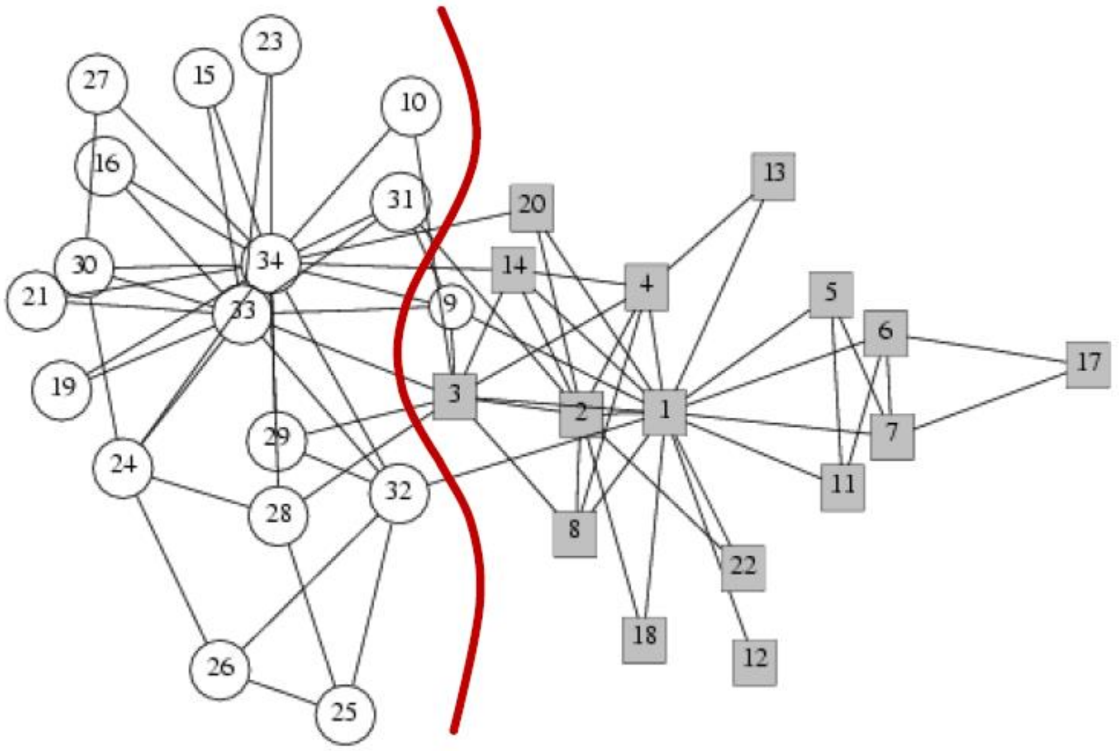
Finding Network Communities

- **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

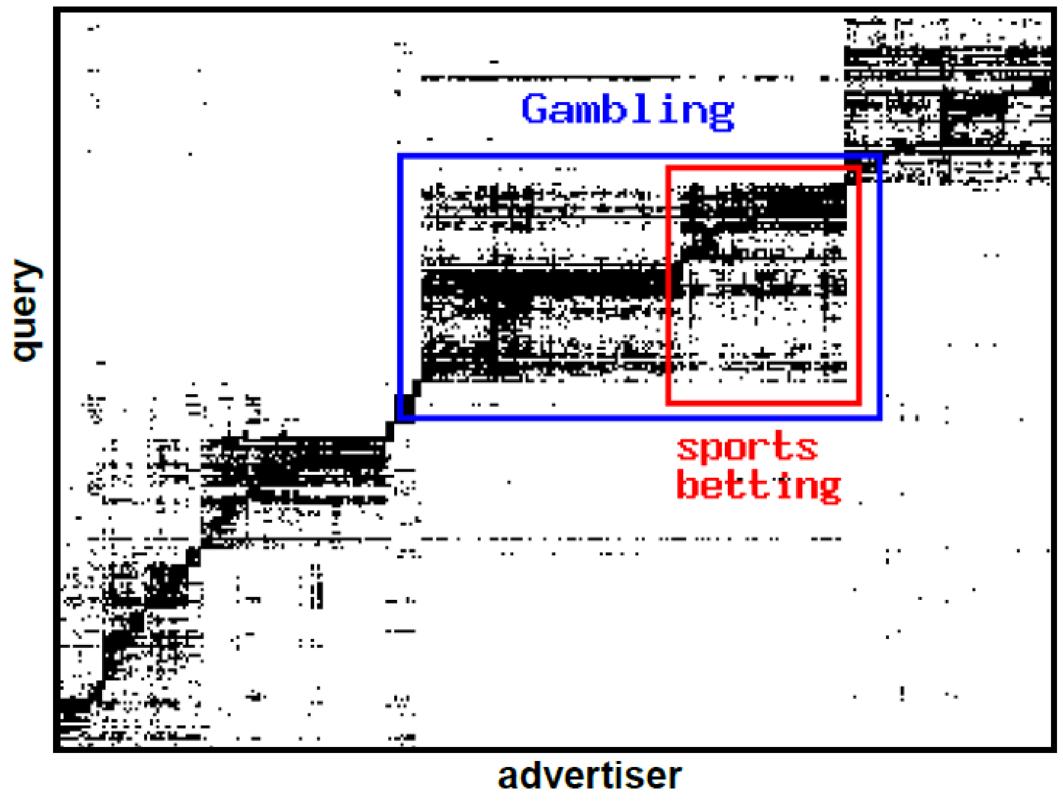
Social Network Data



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

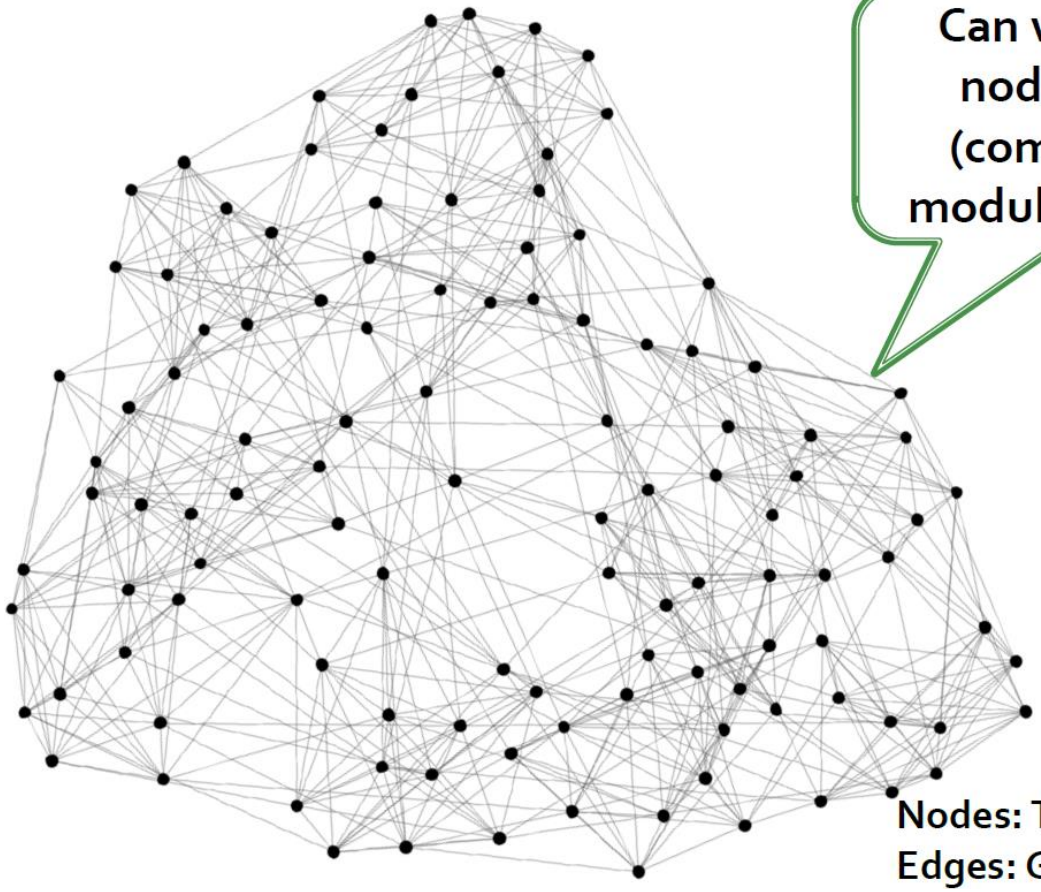
Micro-Markets in Sponsored Search

- **Find micro-markets by partitioning the “Query x Advertiser” graph:**



Web search:
keywords and advertisers bidding with them

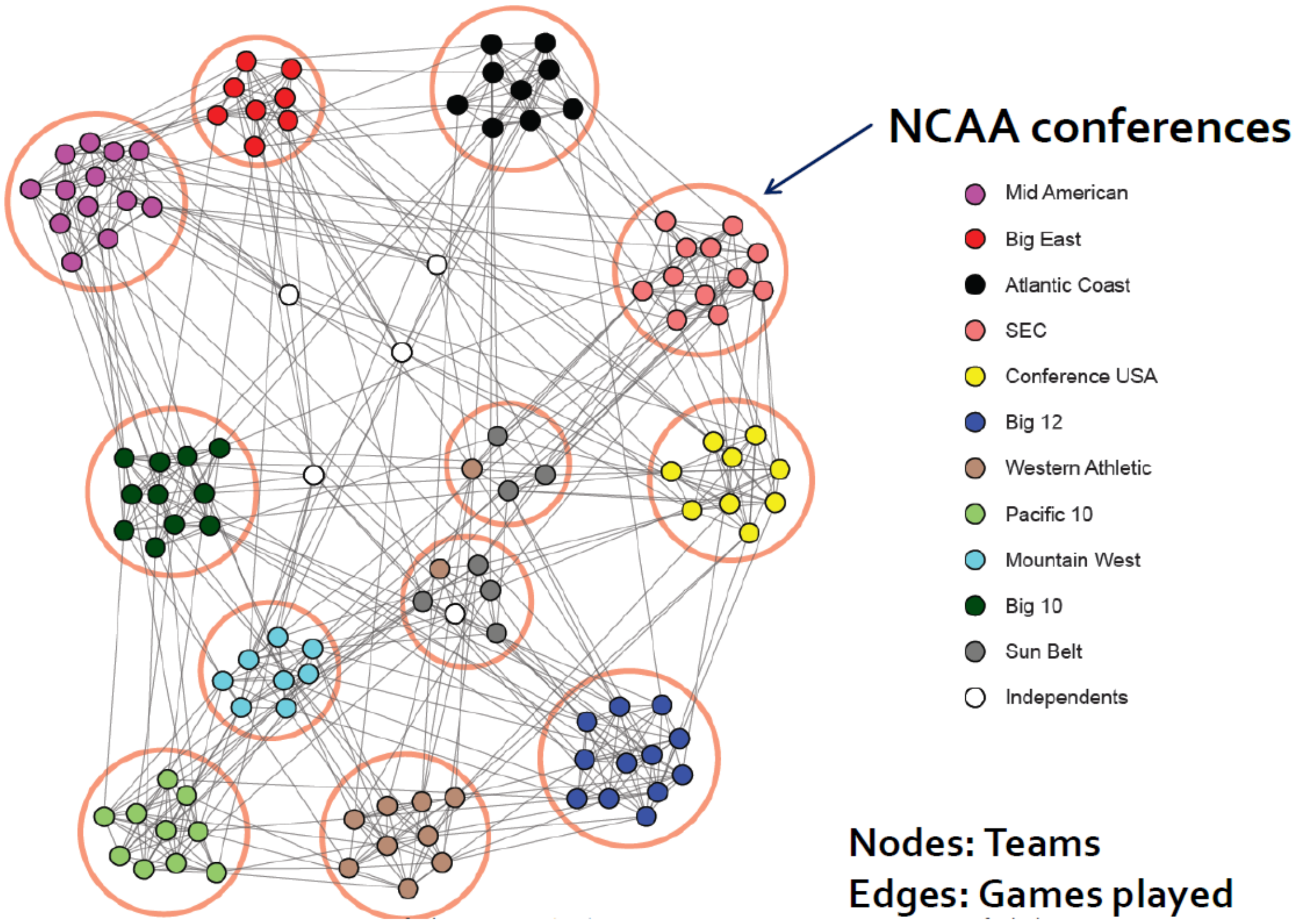
NCAA Football Network



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

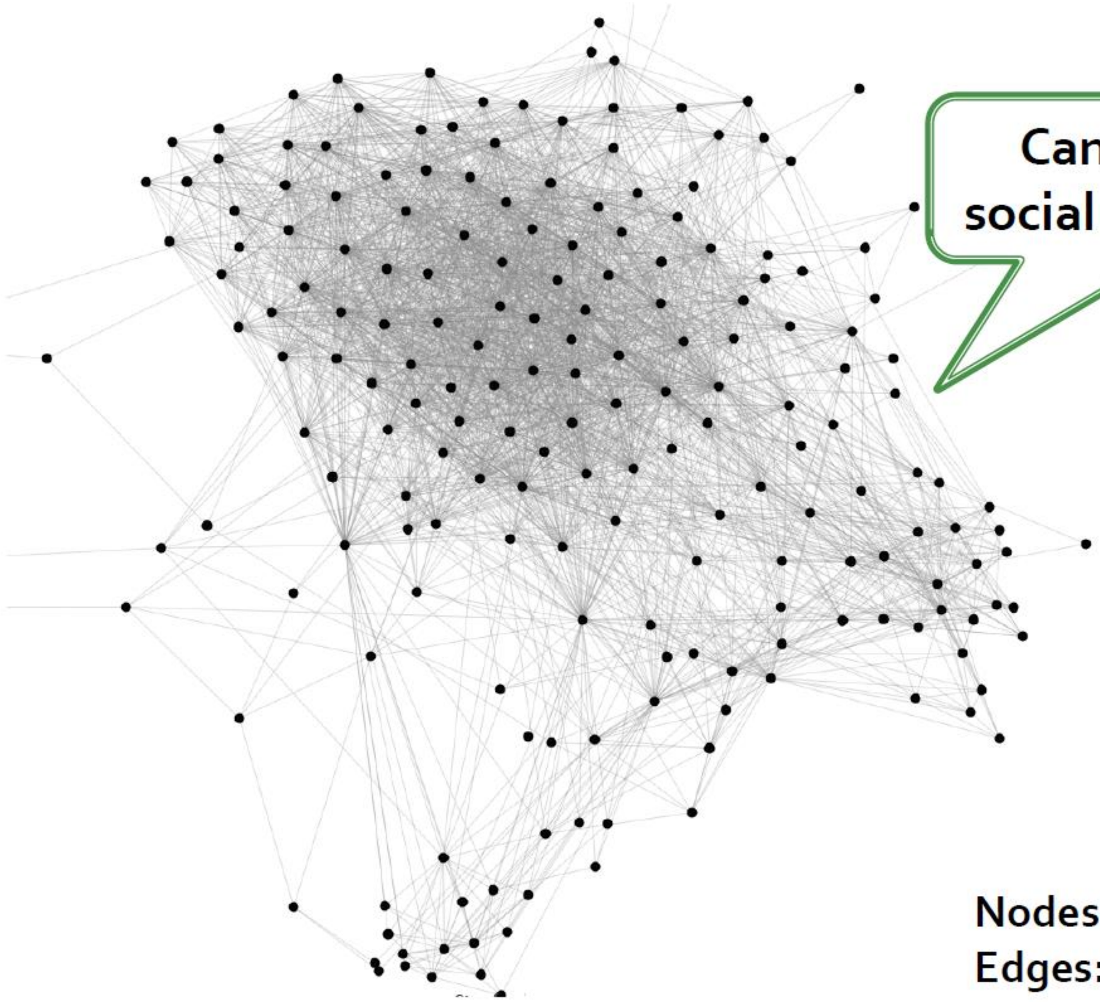
Nodes: Teams
Edges: Games played

NCAA Football Network



Nodes: Teams
Edges: Games played

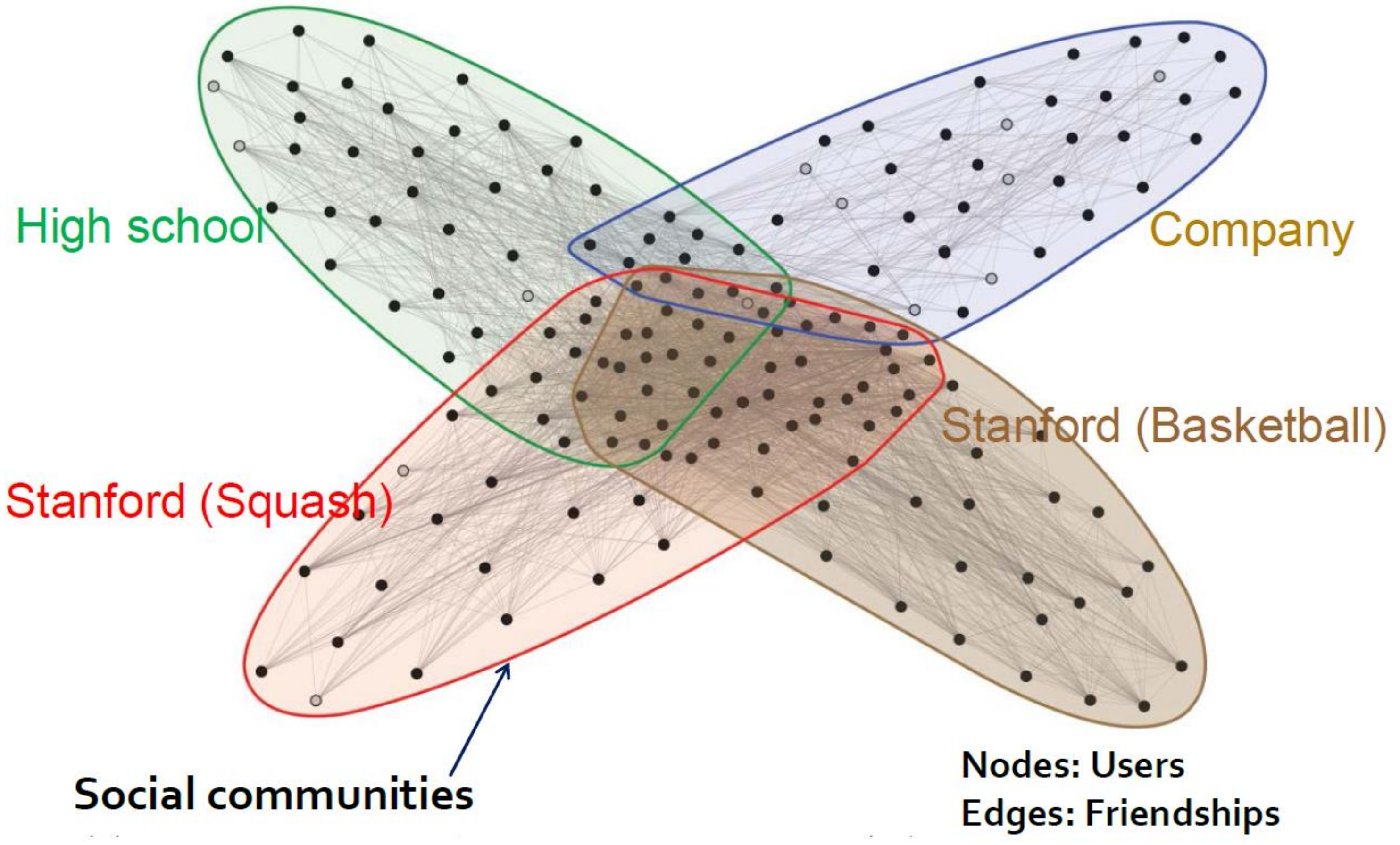
Facebook Ego-network



Can we identify social communities?

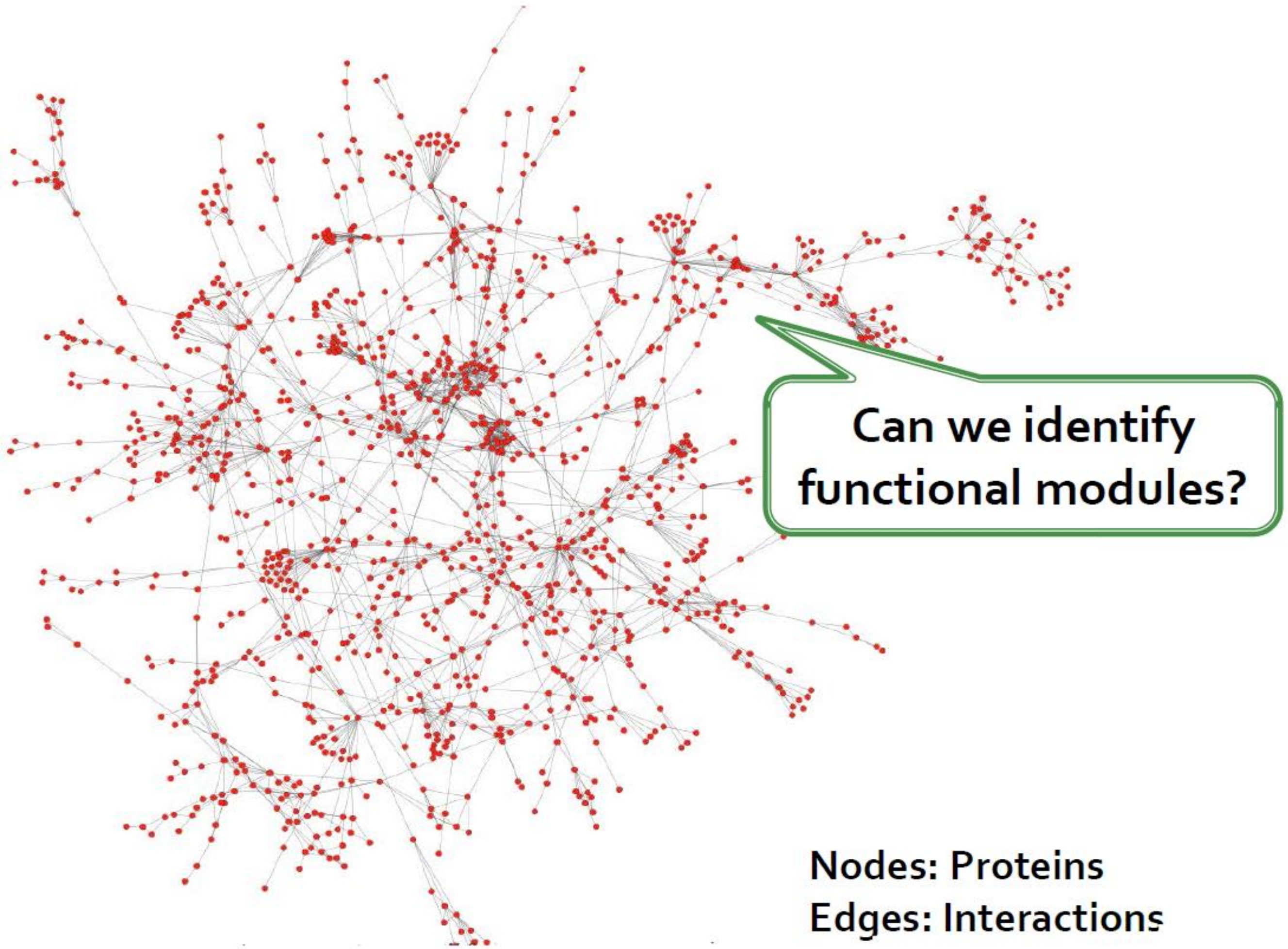
Nodes: Users
Edges: Friendships

Facebook Ego-network



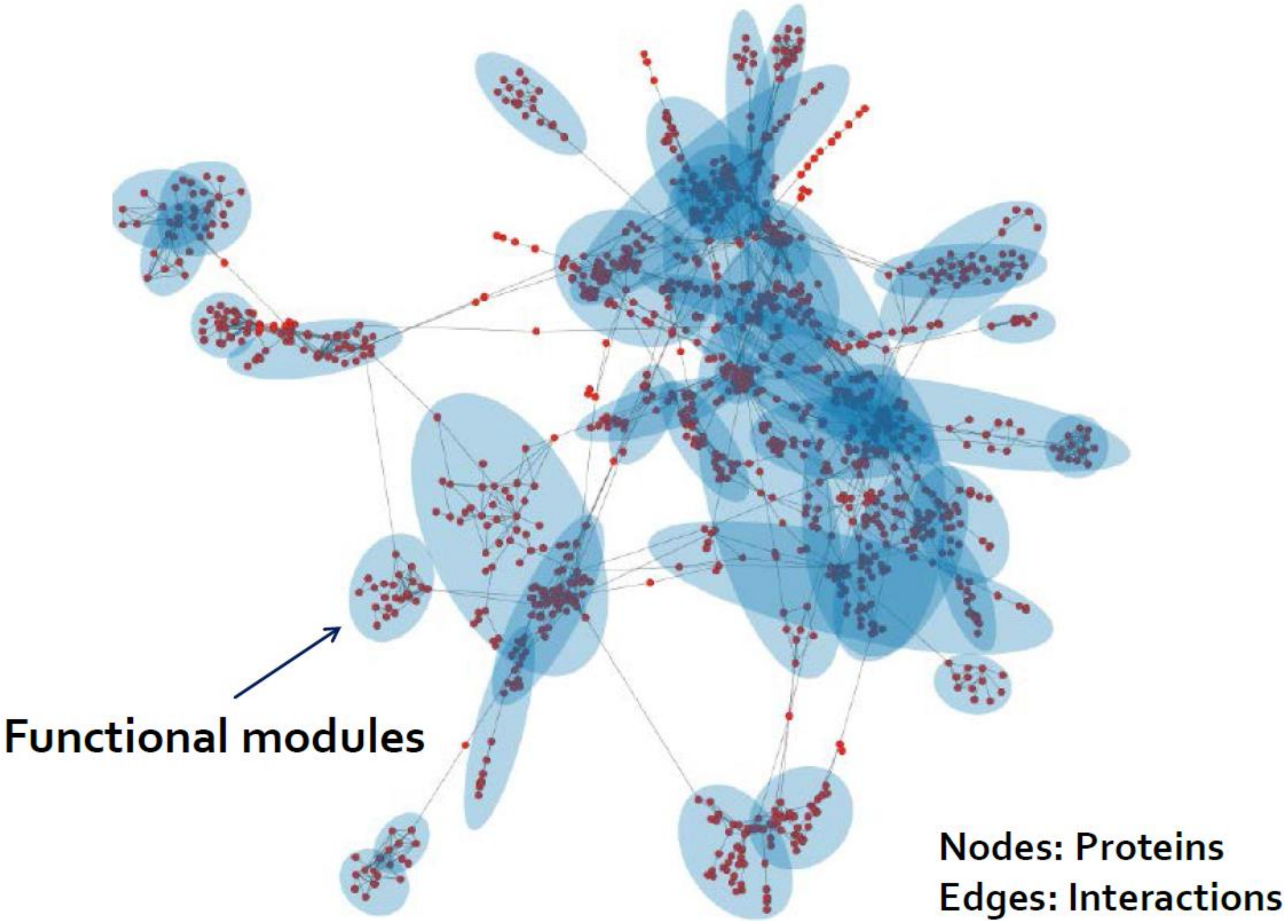
Nodes: Users
Edges: Friendships

Protein-Protein Interactions



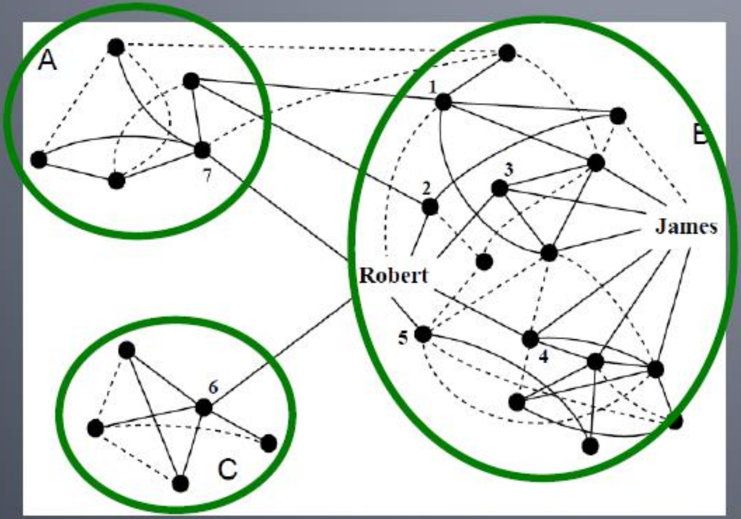
Nodes: Proteins
Edges: Interactions

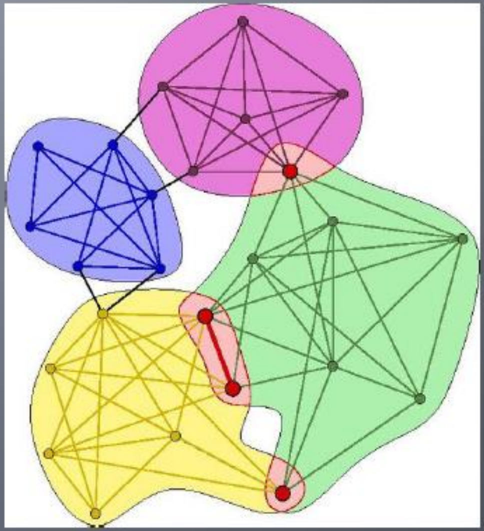
Protein-Protein Interactions



Community Detection

How to find communities?





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

Partitions and Communities

- An algorithm for ***community detection***
  - Choose some ***score function f***
    - takes ***a network A and a partition P of its vertices*** as input
    - returns a ***scalar value (score)*** as output
  - ***Find the partition P that maximizes f***
    - f encodes our beliefs about ***what makes a good partition*** with respect to representing community structure
  - If ***f*** prefers assortative structure
    - assigns higher scores to partitions that place more edges within groups than between them
    - maximizing f over all partitions will yield the partition with the most assortative groups possible under f

$$\boldsymbol{score = f(G, P)}$$

- ***Network partitions (communities):***
  - ***a division of a network into k non-empty groups*** (communities)
  - ***every node v belongs to one and only one group*** (community)

Partitions and Communities

Combinatorial problem:

- Number of ways to divide network of n nodes in 2 groups (bi-partition):

$$\frac{n!}{n_1!n_2!}, \qquad n = n_1 + n_2$$

- Dividing into k non-empty groups (Stirling numbers of the second kind)

$$S(n,k) = \frac{1}{k!}\sum_{j=0}^{n}(-1)^j C_k^j (k-j)^n$$

there are $S(4,2) = 7$ possible partitions of a network with $n = 4$ nodes (indexed as 1,2,3,4) partitioned into $k = 2$ groups:

$$\{1\}\{234\},\ \{2\}\{134\},\ \{3\}\{124\},\ \{4\}\{123\},\ \{12\}\{34\},\ \{13\}\{24\},\ \{14\}\{23\}$$

Partitions and Communities

Combinatorial problem:

- Number of ways to divide network of n nodes in 2 groups (bi-partition):

$$\frac{n!}{n_1!n_2!}, \qquad n = n_1 + n_2$$

- Dividing into k non-empty groups (Stirling numbers of the second kind)

$$S(n,k) = \frac{1}{k!}\sum_{j=0}^{n}(-1)^j C_k^j (k-j)^n$$

$$\left\{ {n \atop k} \right\} = \sum_{j=1}^{k}(-1)^{k-j}\frac{j^{n-1}}{(j-1)!(k-j)!} = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$$

there are $S(4,2) = 7$ possible partitions of a network with $n = 4$ nodes (indexed as 1,2,3,4) partitioned into $k = 2$ groups:

$$\{1\}\{234\},\ \{2\}\{134\},\ \{3\}\{124\},\ \{4\}\{123\},\ \{12\}\{34\},\ \{13\}\{24\},\ \{14\}\{23\}$$

Partitions and Communities

Combinatorial problem:

- Number of ways to divide network of n nodes in 2 groups (bi-partition):

$$\frac{n!}{n_1!n_2!}, \qquad n = n_1 + n_2$$

- Dividing into k non-empty groups (Stirling numbers of the second kind)

$$S(n,k) = \frac{1}{k!}\sum_{j=0}^{n}(-1)^j C_k^j (k-j)^n$$

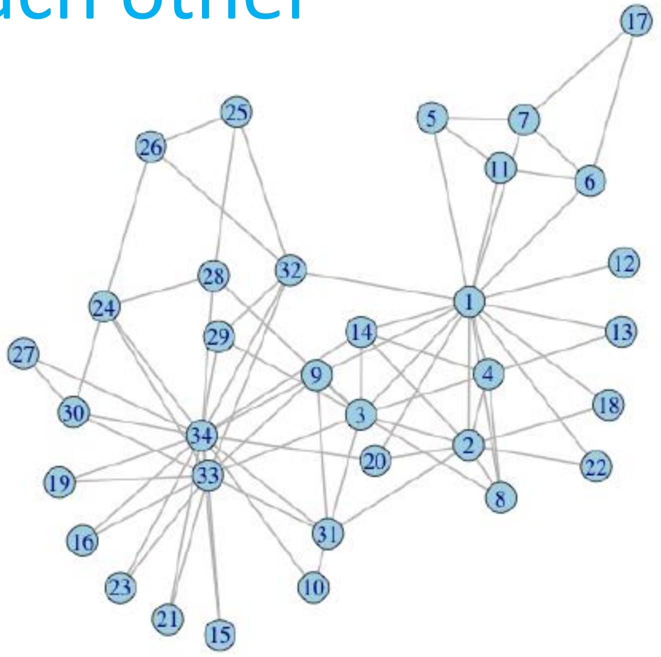
- Number of all possible partitions (n-th Bell number):

$$B_n = \sum_{k=1}^{n} S(n,k)$$

$B_{20} = 5,832,742,205,057$ $\Rightarrow$ **Brute force algorithms won't work**
$\Rightarrow$ **Need Heurisric / Greedy Algorithms**

Network communities

- ***Network communities*** are groups of nodes ***similar*** to each other



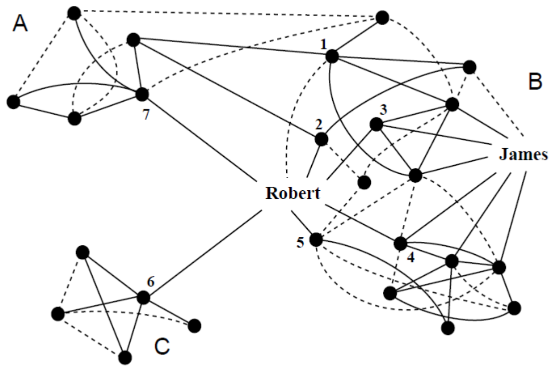
- ***Community detection:*** assignment of nodes to communities
  - Non-overlapping communities (every node belongs to a single group)

Similarity based clustering

Similarity based node clustering

- Define ***similarity measure between nodes based on network structure***
  - *Jaccard similarity*
  - *Cosine similarity*
  - *Pearson correlation*
  - *Euclidean distance (dissimilarity)*
- Calculate similarity between all pairs of nodes in the graph (similarity matrix)
- Group together nodes with high similarities

Clustering and Community Finding

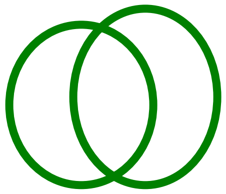


How to extract groups?

Many methods:

No community overlaps

- ***Hierarchical methods:***
  - Top-down / bottom-up
- ***Graph partitioning methods:***
  - Define "edge counting" metric – conductance, expansion, modularity, etc. – and optimize!
- ***Spectral methods:*** **Lecture Notes on Spectral Graph Methods (by Mahoney)**
  - Based on eigenvector decomposition of modified graph adjacency matrix
- ***Clique percolation method:***
  - To extract overlapping communities in networks



Betweenness and Graph Partitioning

- A way of thinking ***networks*** in terms of
  - ***tightly-knit regions*** and ***weaker ties*** that link them together
    - Clustering coefficient and local bridges
  - What we mean by tightly-knit region?
- ***Graph Partitioning:***
  - a method that *takes a network and break it down* into a set of tightly-knit regions, with sparser interconnections between the regions
- What we might ***hope*** from this method?
  - Need a general way to pull (visually observed) groups out of the data, beyond using just visual intuition

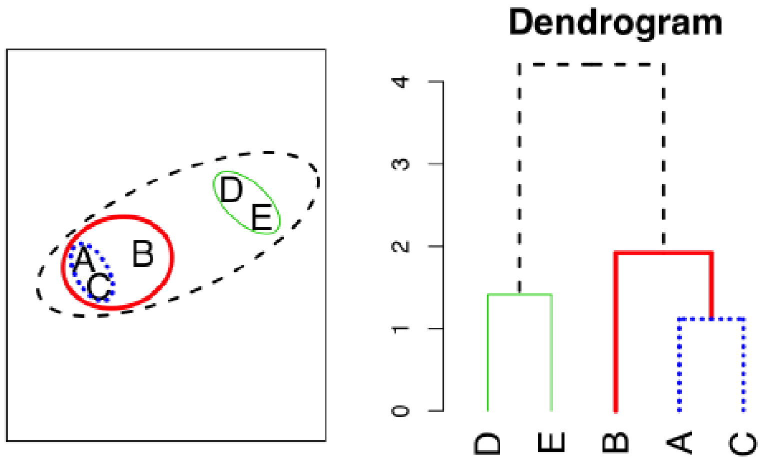
Methods of Graph Partitioning

- **Agglomerative methods** of graph partitioning
  - find nodes that are likely to belong to the same region and merge them together
    - network consists of a large number of ***merged chunks***, each containing the seeds of a ***densely-connected region***
  - process looks for chunks that should be further merged together
  - regions are assembled "bottom-up"
- **Divisive methods** of graph partitioning
  - Identify and remove the "***spanning links***" between ***densely-connected regions***
    - the network begins to fall apart into large pieces
  - spanning links are further identified and removed within these connected components
  - process continues "top-to-bottom"

Hierarchical clustering

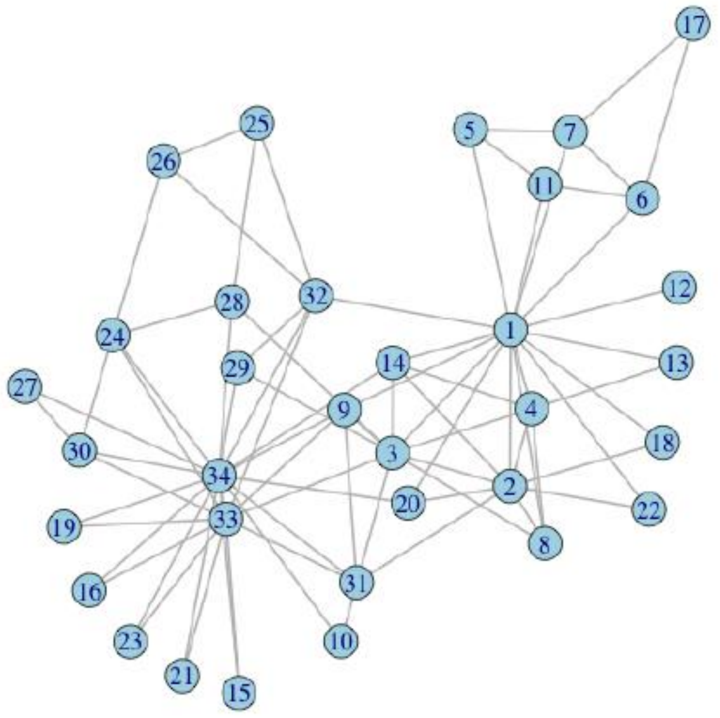
Agglomerative clustering:

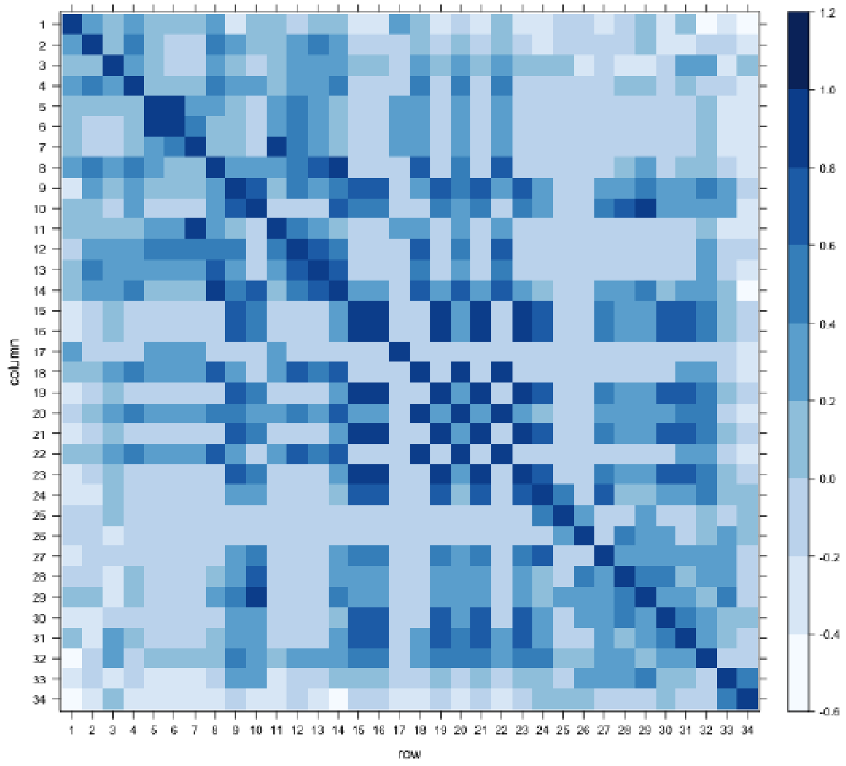
- Assign each node to a group of its own
- Find two groups with the ***highest similarity*** and ***join*** them in a single group
- Calculate ***similarity between groups***:
  - ***single-linkage clustering:*** most similar in the group
  - ***complete-linkage clustering:*** least similar in the group
  - ***average-linkage clustering:*** mean similarity between groups
- Repeat until all joined into single group



Similarity matrix

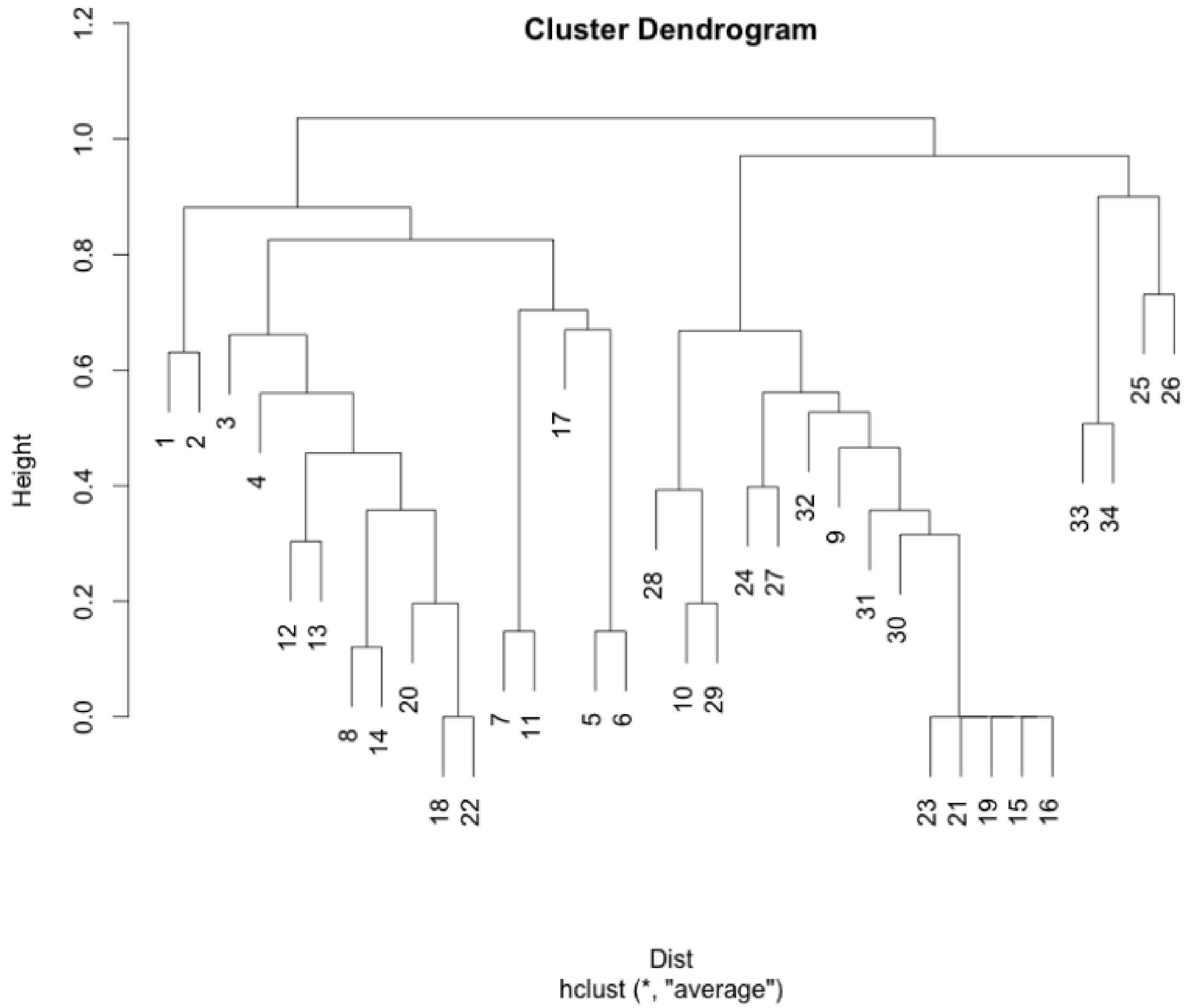
Zachary karate club



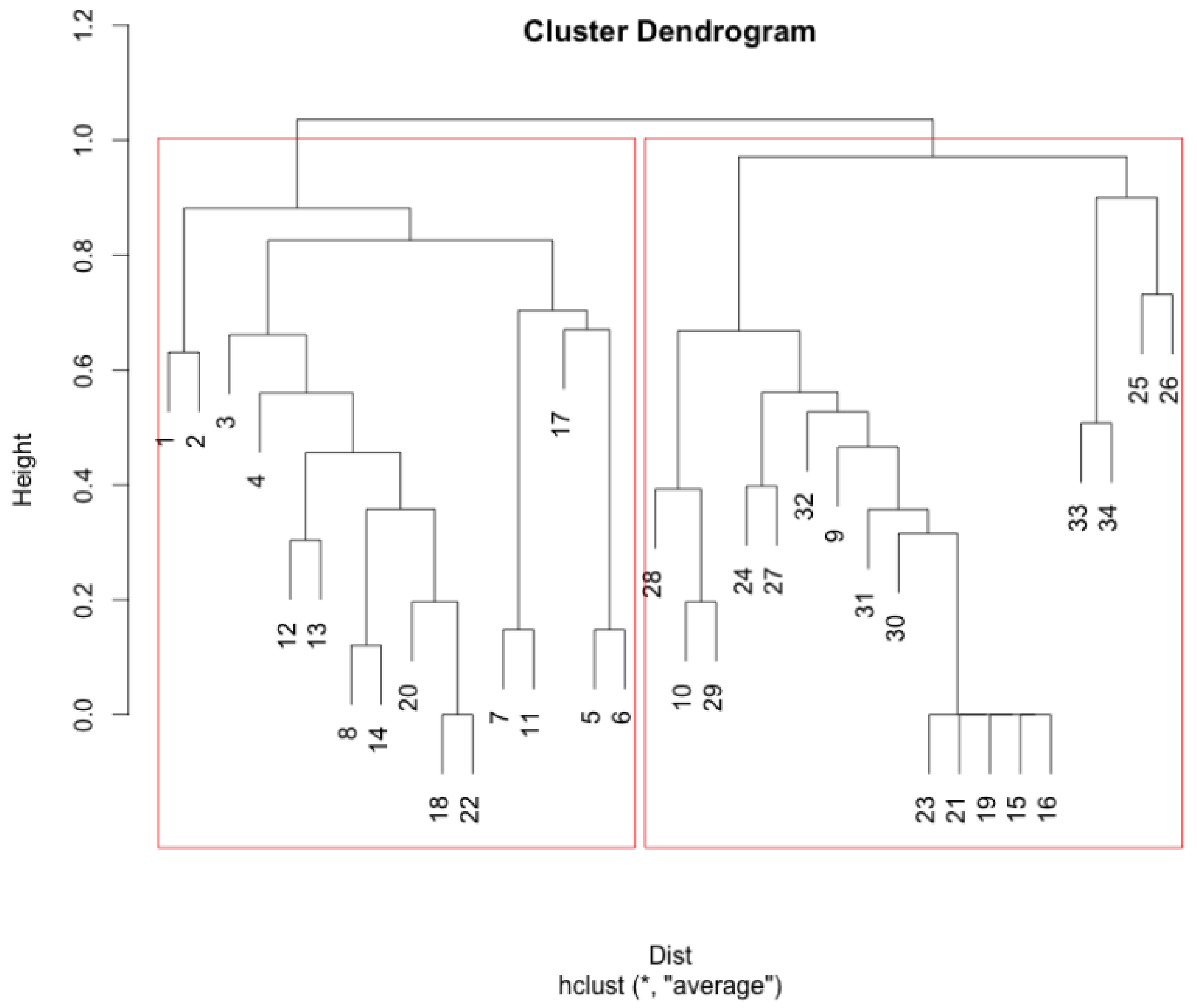


Remember the notions of node equivalences

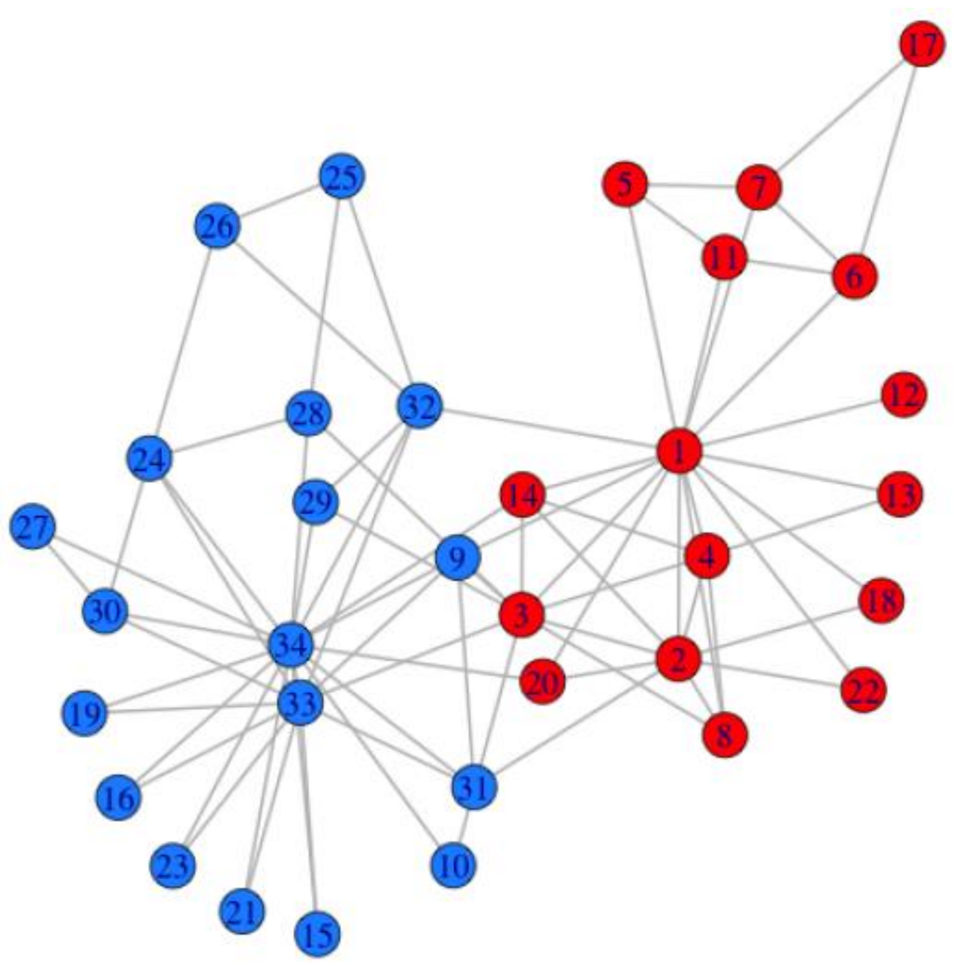
Hierarchical clustering



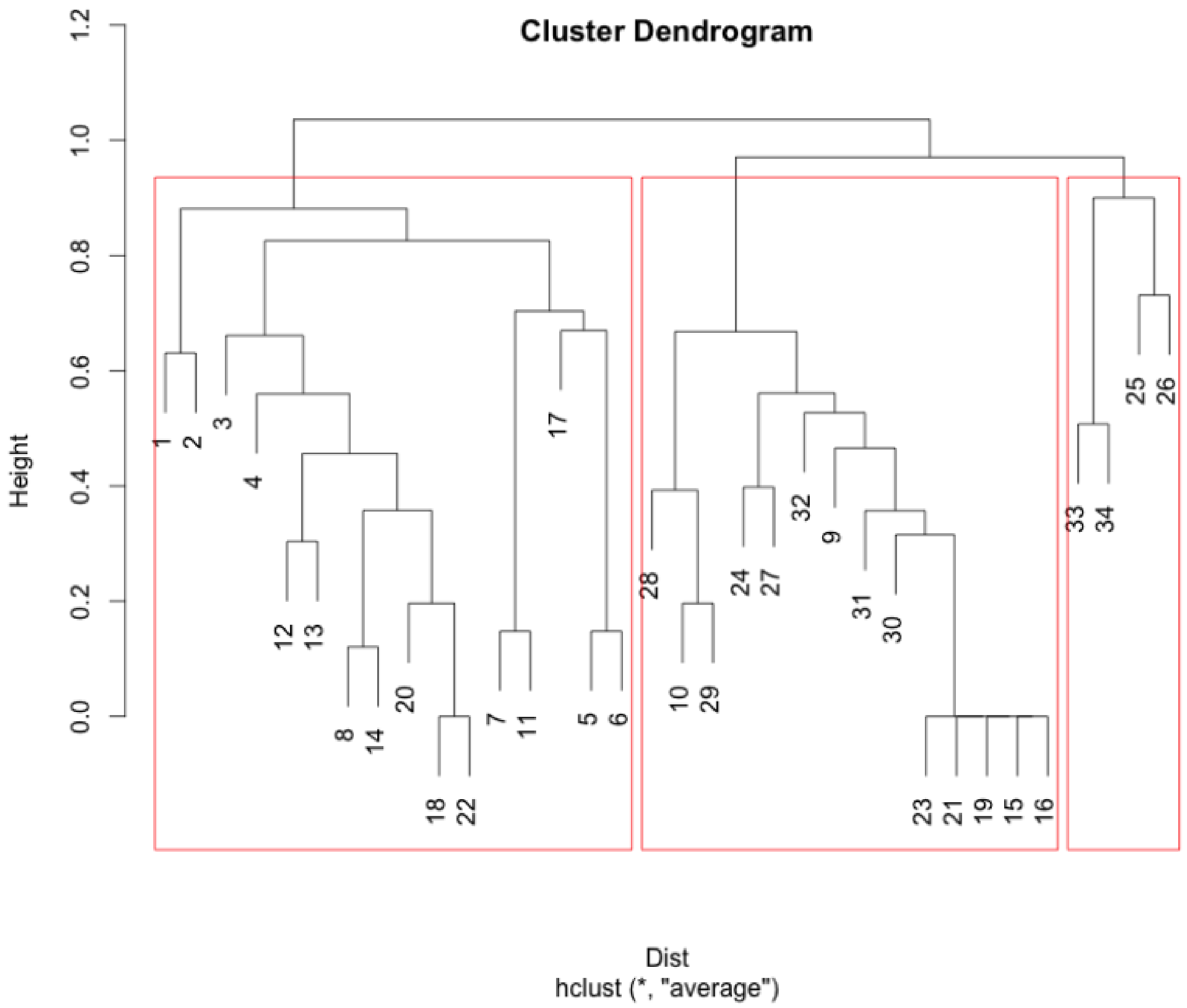
Hierarchical clustering



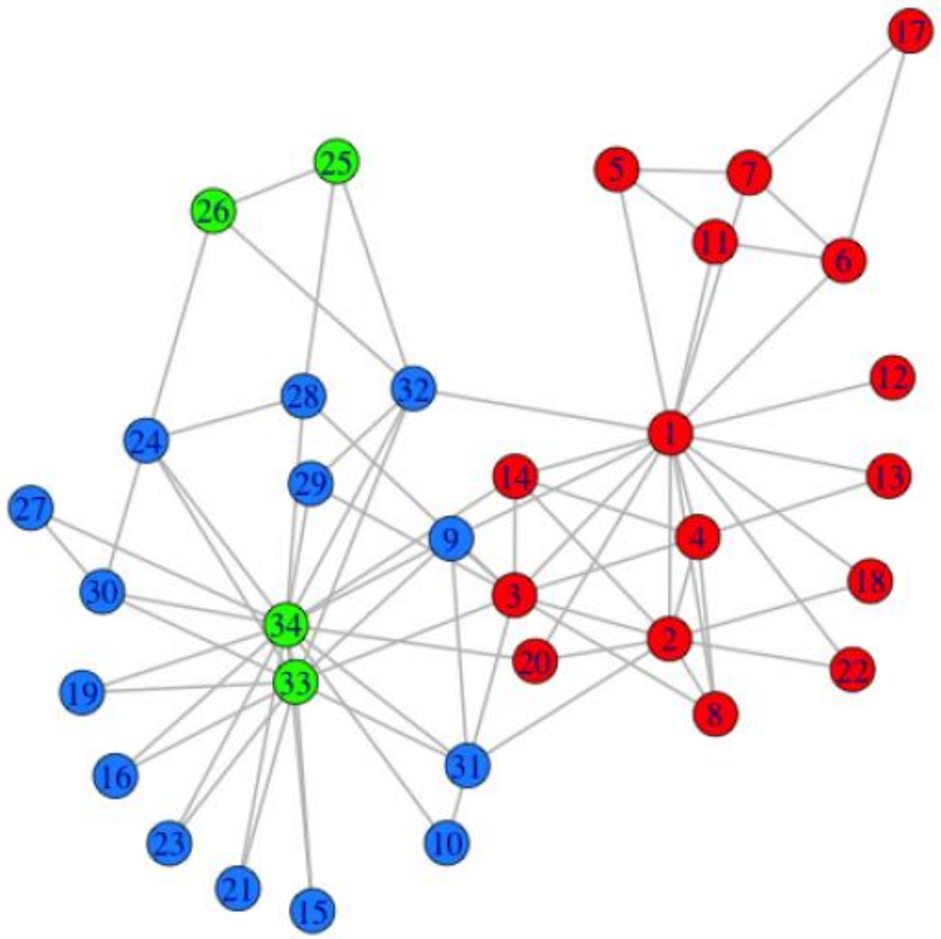
Hierarchical clustering



Hierarchical clustering



Hierarchical clustering



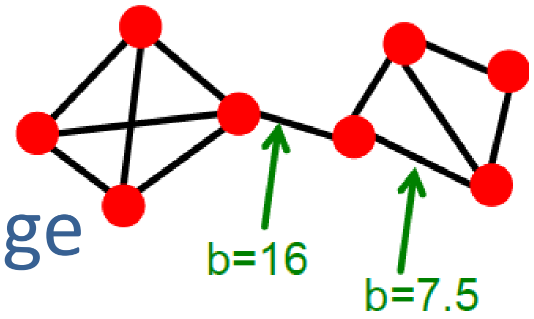
[Girvan-Newman '02]

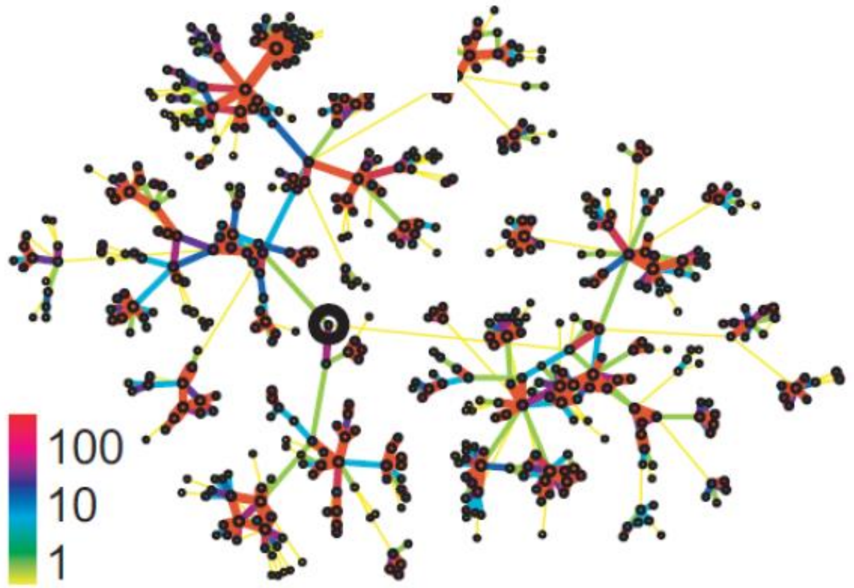
Method 2: ***Girvan-Newman***

- **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

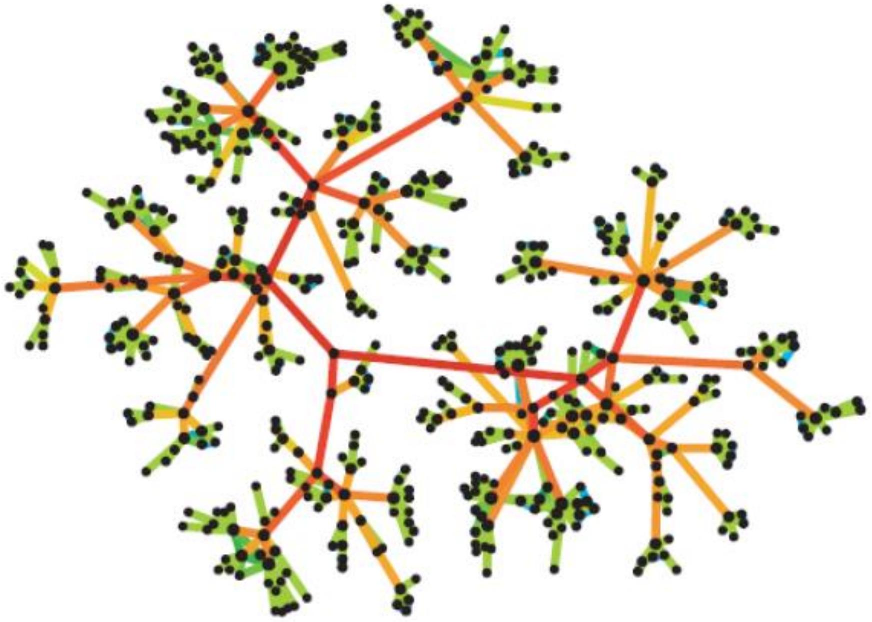
Strength of Weak Ties

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





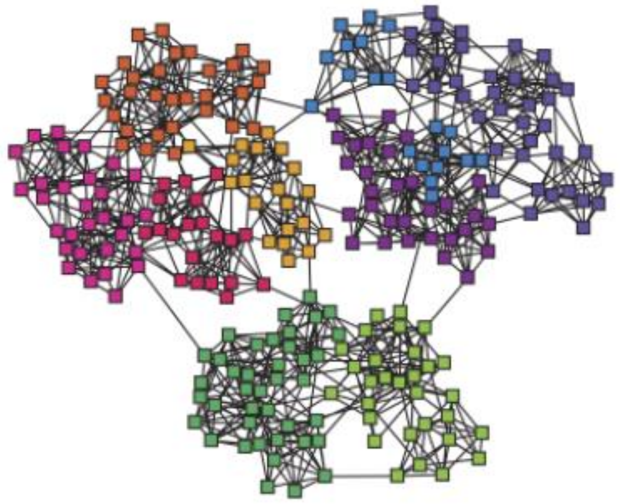
Edge strengths (call volume) in a real network

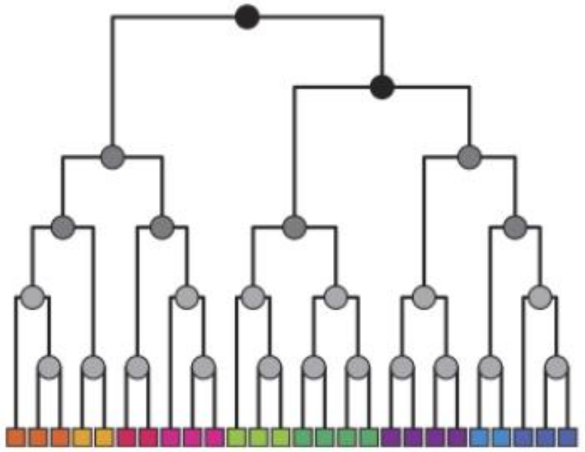


Edge betweenness in a real network

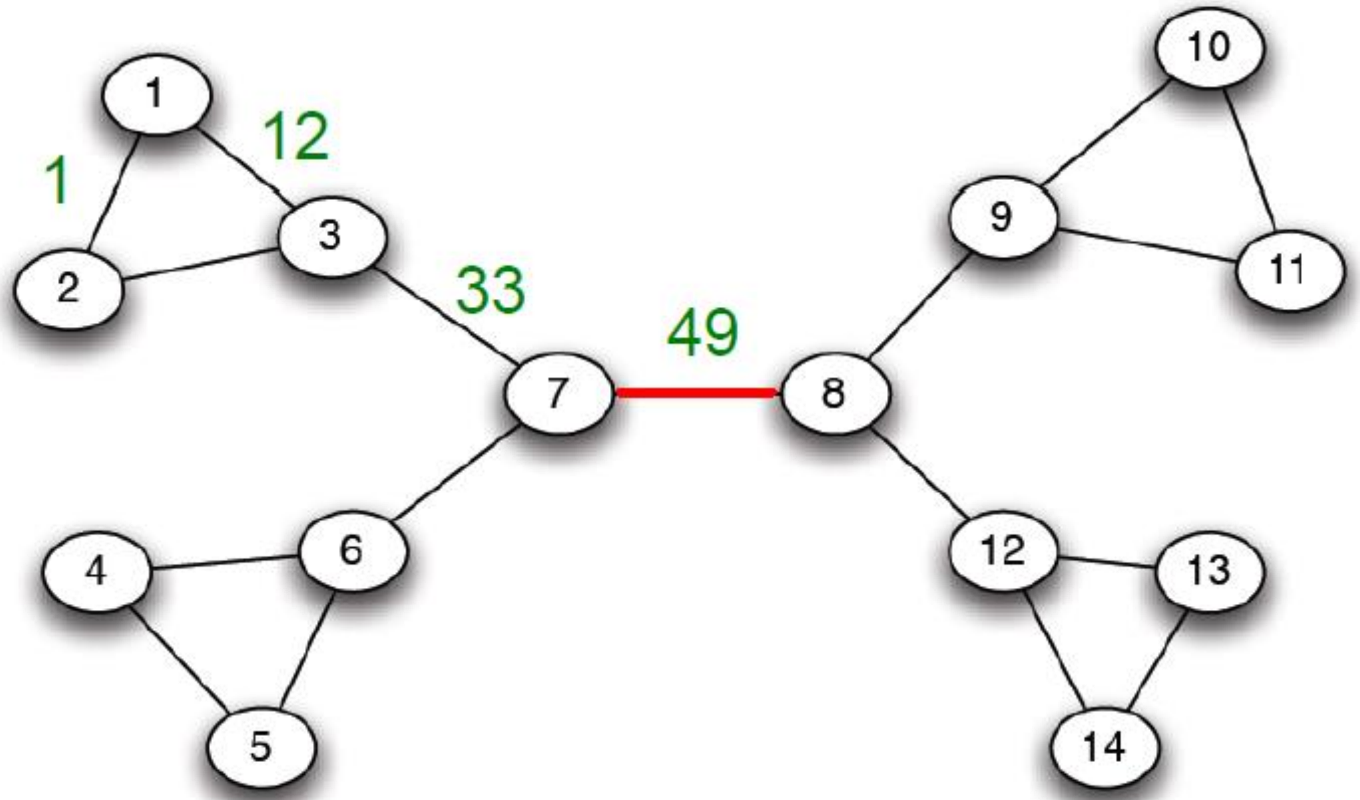
Edge betweenness

- ***Hierarchical algorithm***, dendrogram





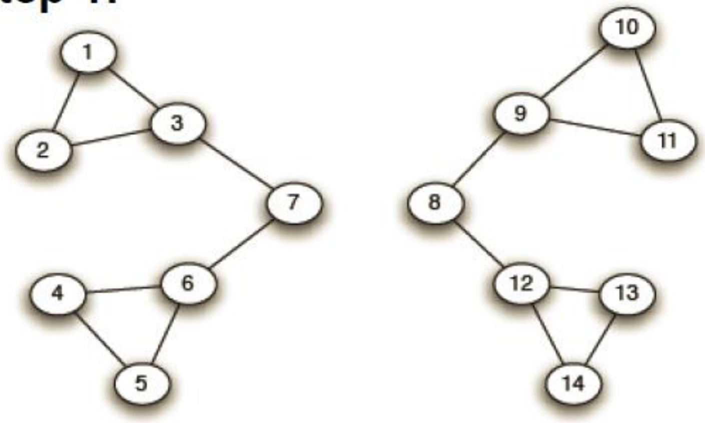
Girvan-Newman: Example



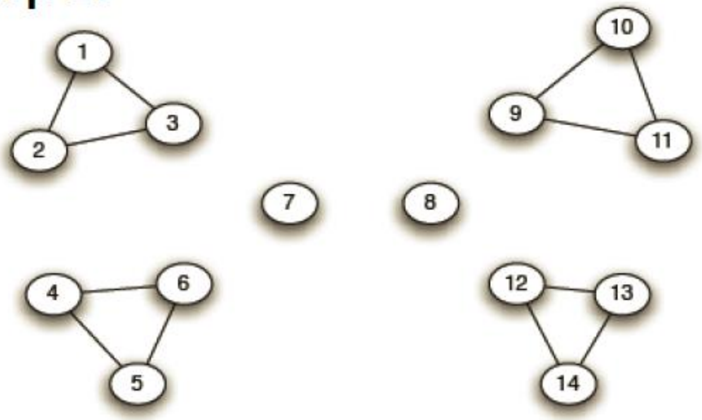
Need to re-compute betweenness at every step

Girvan-Newman: Example

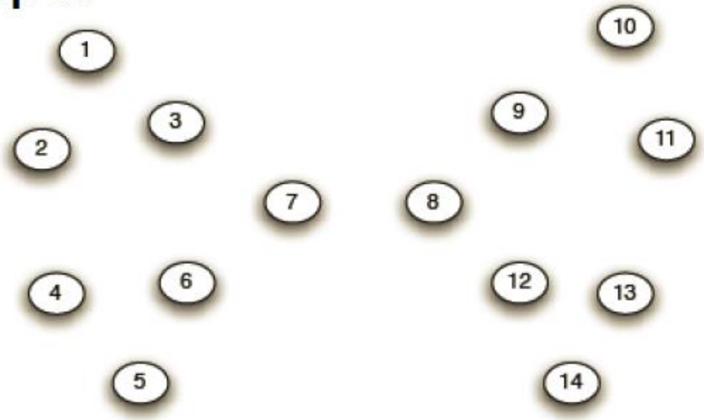
Step 1:



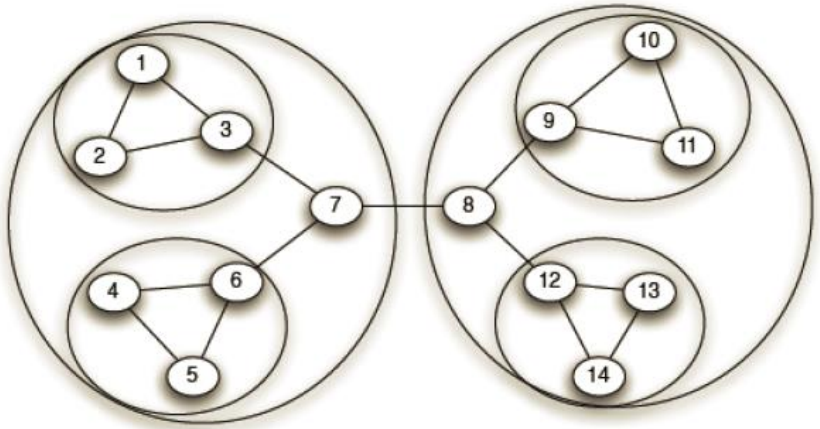
Step 2:



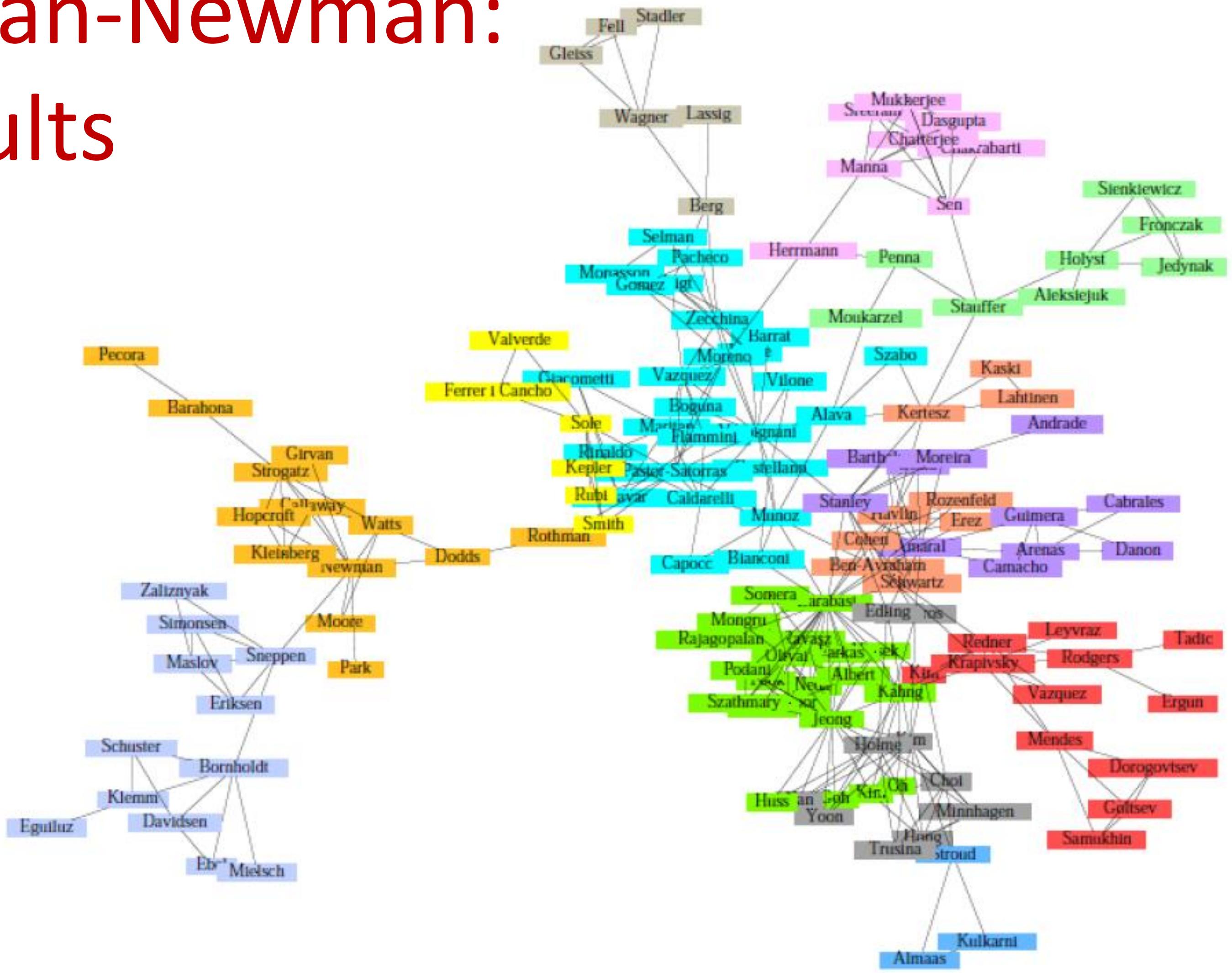
Step 3:



Hierarchical network decomposition:



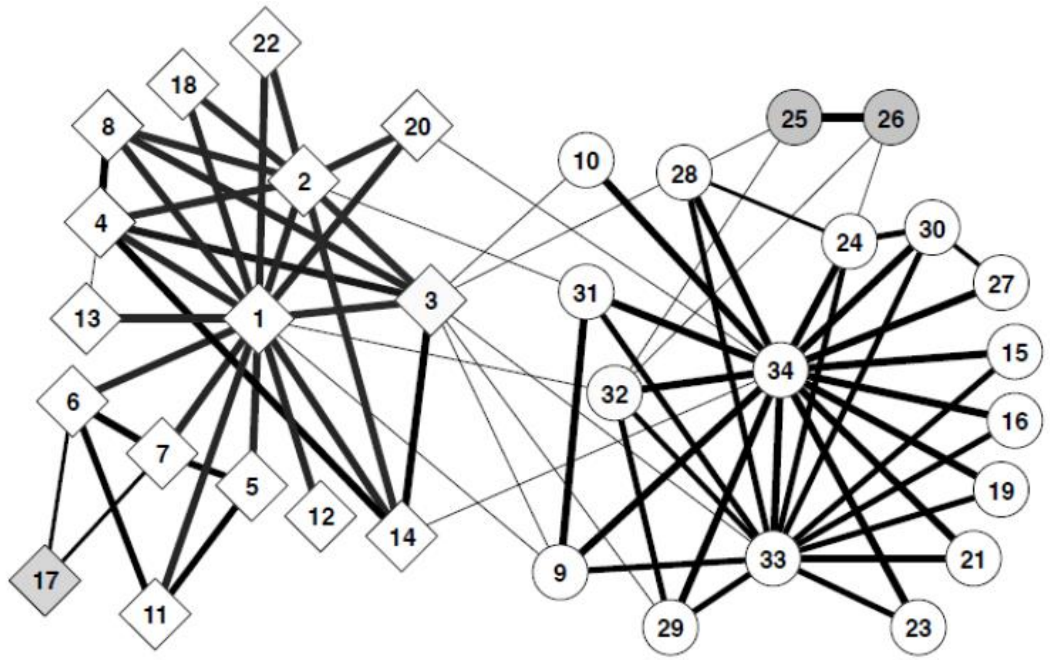
Girvan-Newman: Results

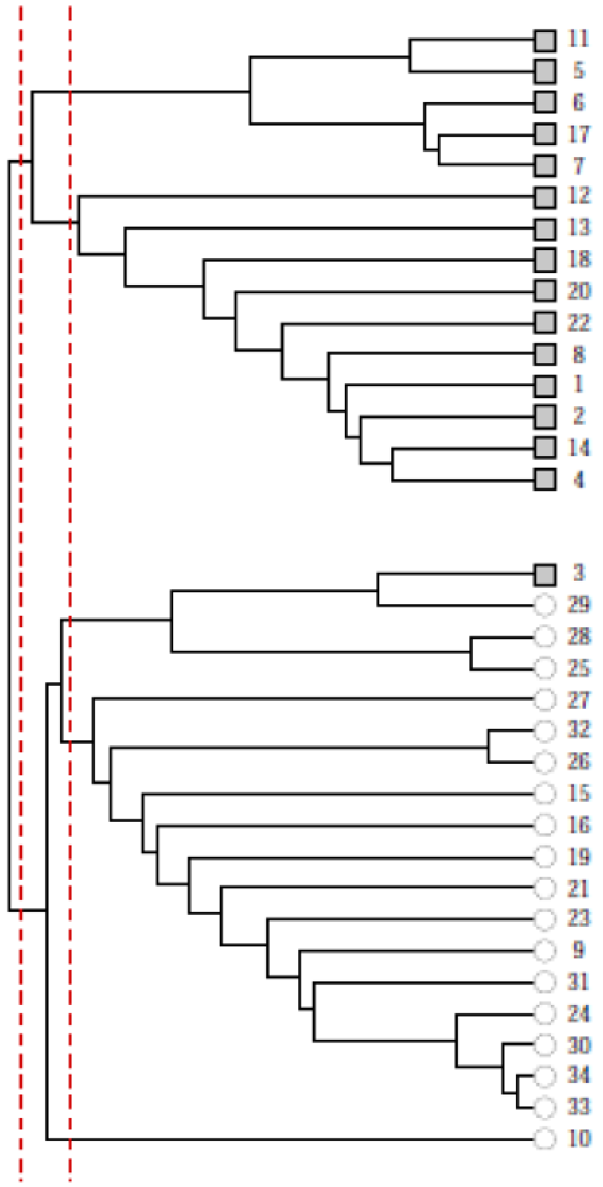


Communities in physics collaborations

Girvan-Newman: Results

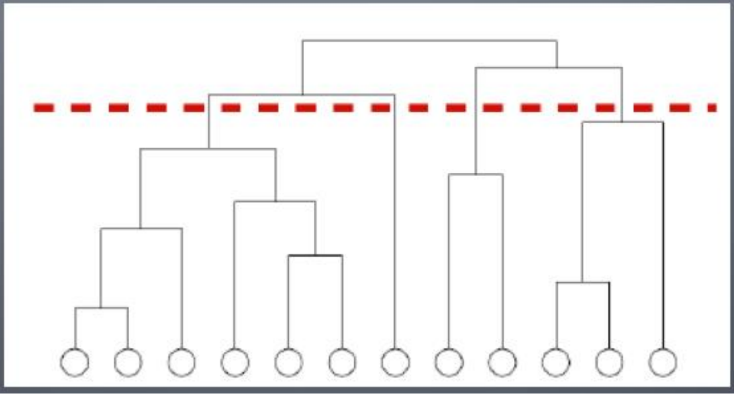
- **Zachary's Karate club:** ***Hierarchical decomposition***





We need to resolve 2 questions

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

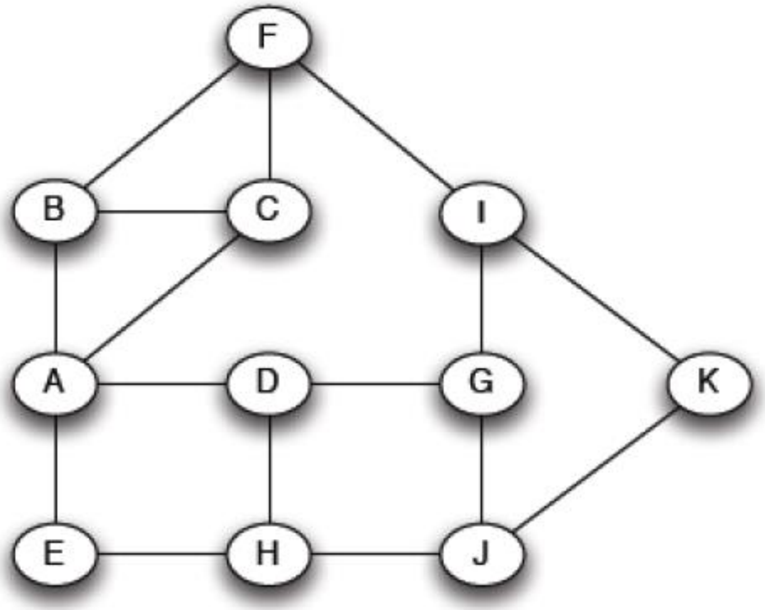


Determine the flow from one node to all other nodes in the graph

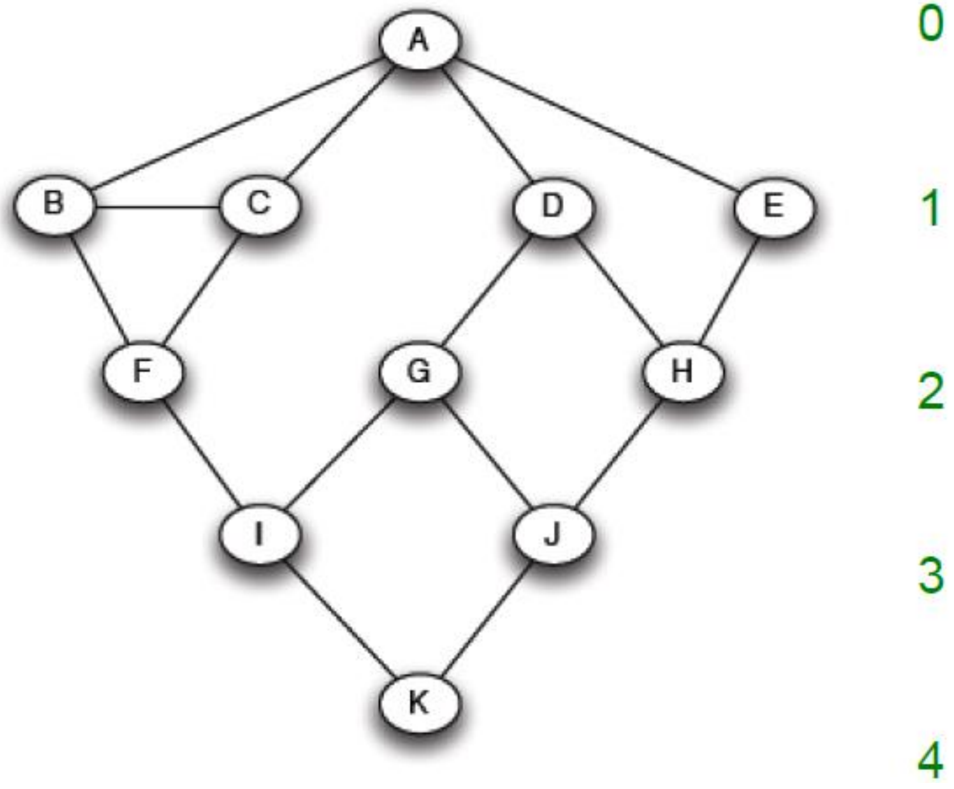
- Perform a breadth-first search of the graph, starting at a node (node A)
- Determine the number of shortest paths from node A to all the other nodes
- Based on these numbers, determine the amount of flow from node A to all other nodes that uses each edge

How to Compute Betweenness?

- Want to compute betweenness of paths starting at node A efficiently

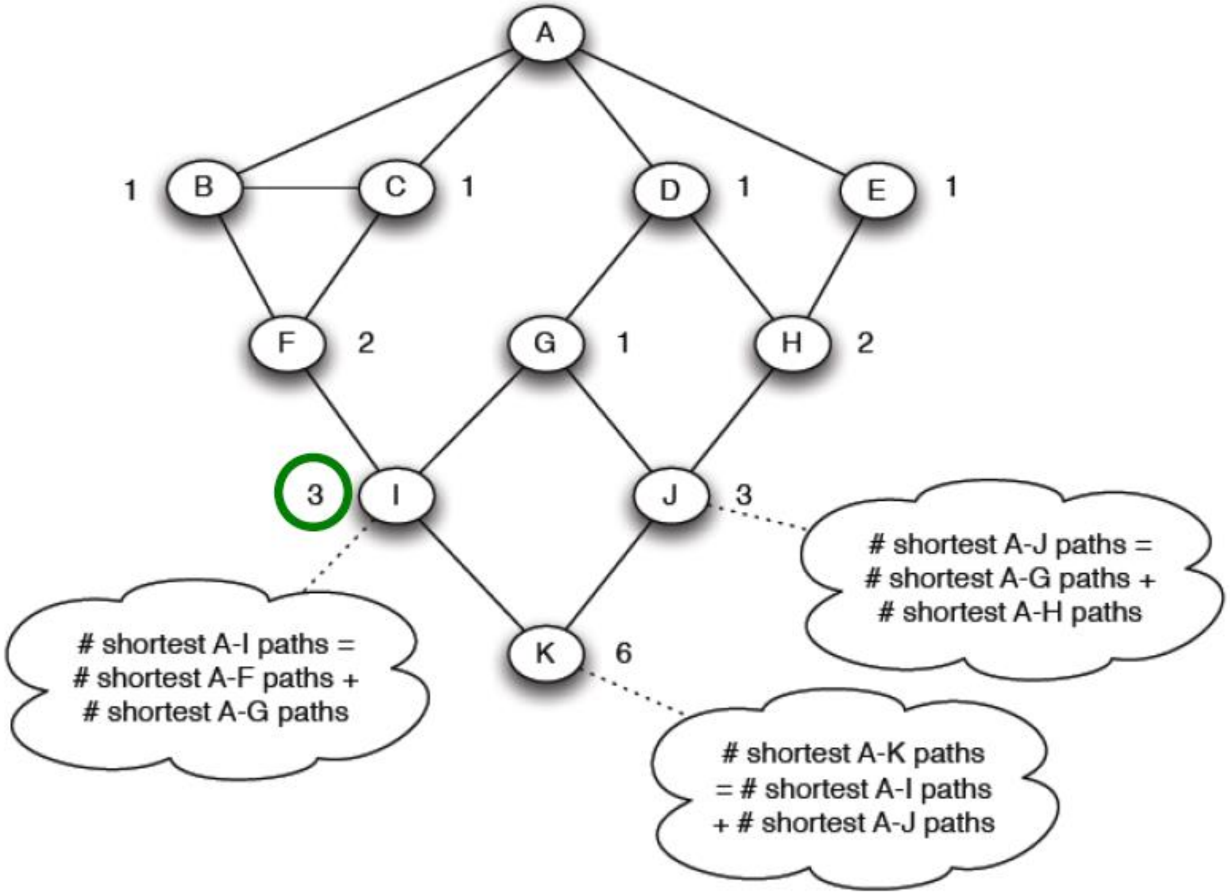


- Breath first search starting from A:



How to Compute Betweenness?

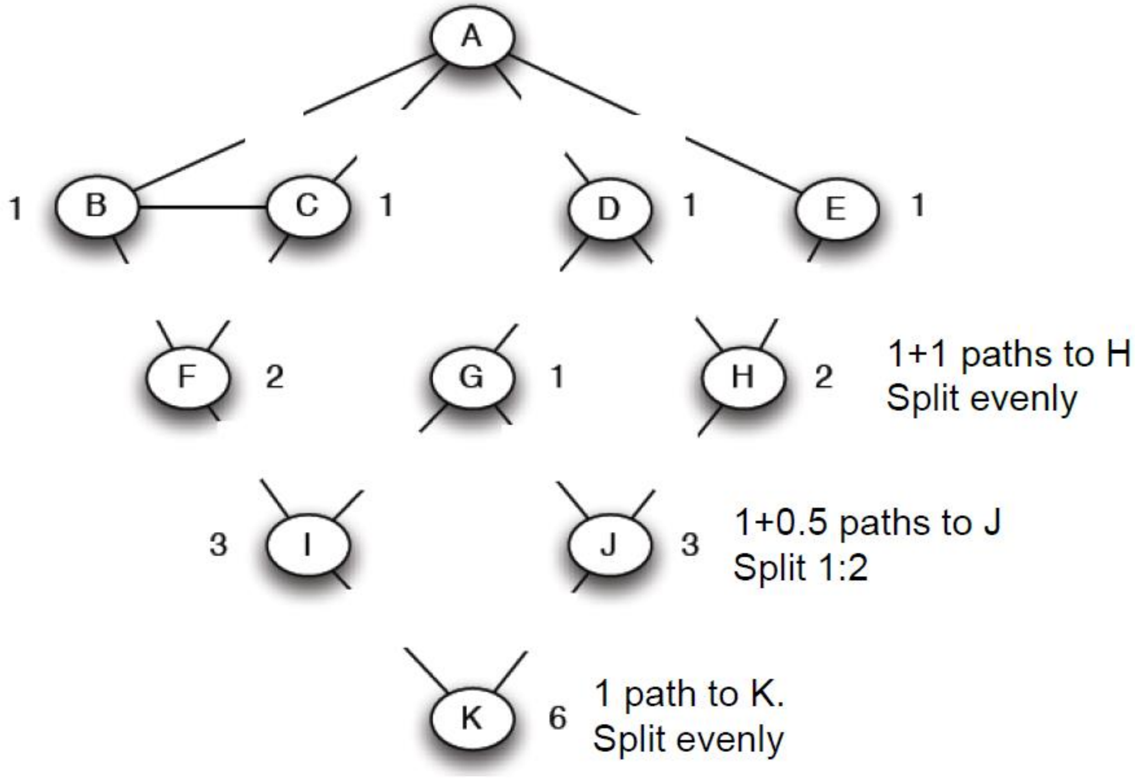
- **Count the number of shortest paths from $\boldsymbol{A}$ to all other nodes of the network:**


A
B 1
C 1
D 1
E 1
F 2
G 1
H 2
3 I
J 3
K 6
# shortest A-I paths = # shortest A-F paths + # shortest A-G paths
# shortest A-J paths = # shortest A-G paths + # shortest A-H paths
# shortest A-K paths = # shortest A-I paths + # shortest A-J paths

How to Compute Betweenness?

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

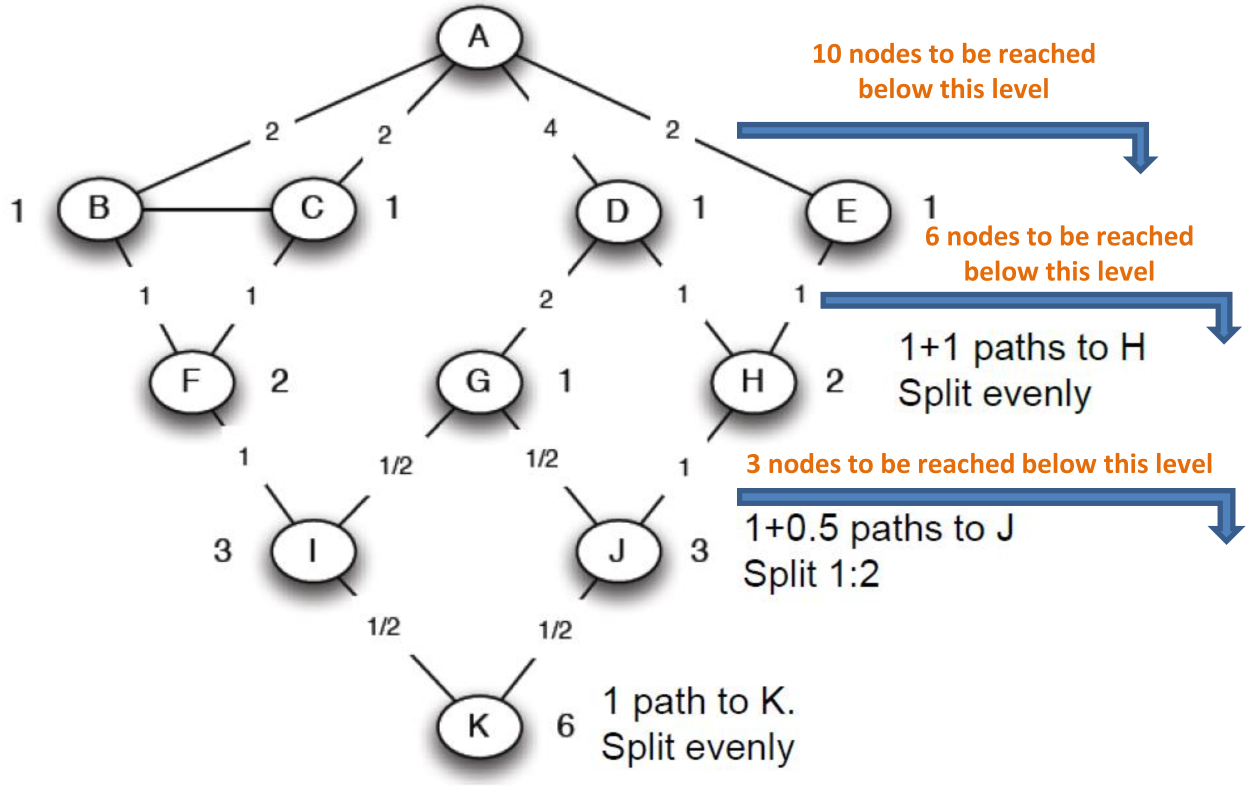
The algorithm:
•Add edge **flows**:
-- node flow = 1+∑child edges
-- split the flow up based on the parent value
• Repeat the BFS procedure for each starting node *U*



How to Compute Betweenness?

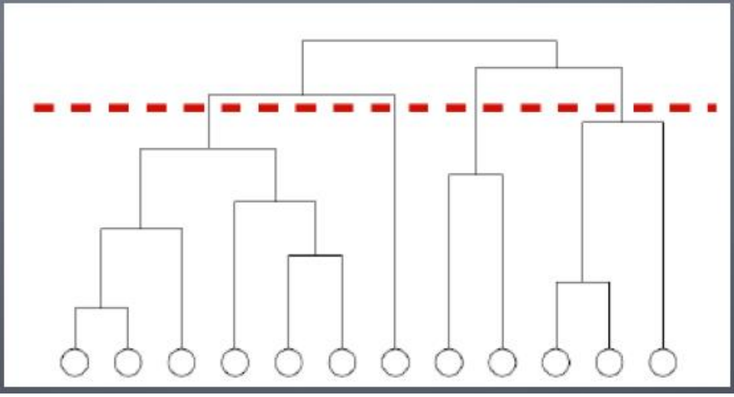
- ***Compute betweenness by working up the tree:*** If there are multiple paths count them fractionally

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


A
B
C
D
E
F
G
H
I
J
K
1
1
1
1
2
1
2
3
3
6
2
2
4
2
1
1
2
1
1
1
1/2
1/2
1
1/2
1/2
10 nodes to be reached below this level
6 nodes to be reached below this level
1+1 paths to H Split evenly
3 nodes to be reached below this level
1+0.5 paths to J Split 1:2
1 path to K. Split evenly

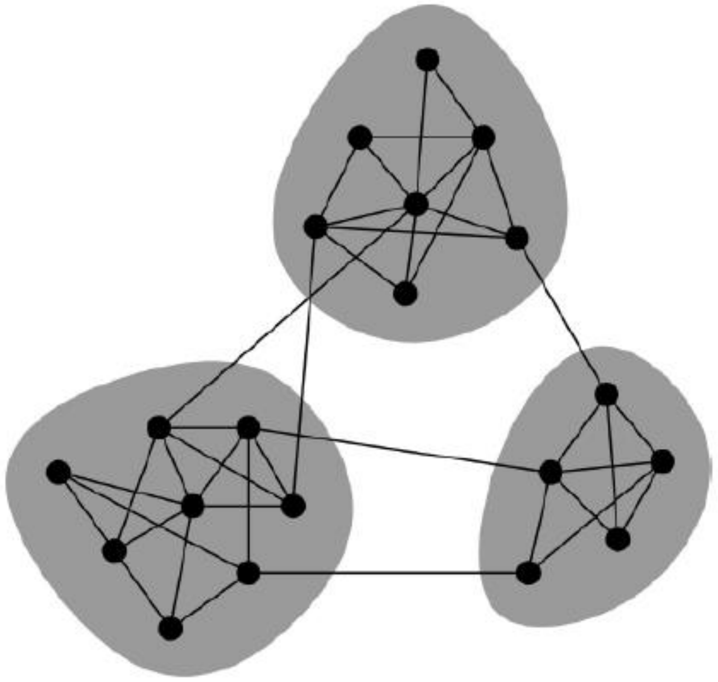
We need to resolve 2 questions

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



Network Communities

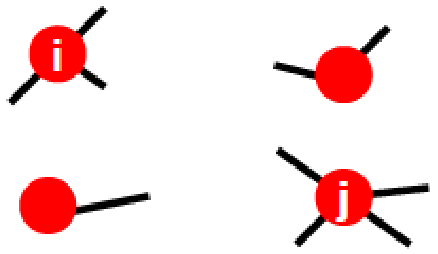
- ***Communities:*** sets of ***tightly connected nodes***
- ***Define:*** **Modularity *Q***
  - A measure of how well a network is partitioned into communities
  - Given a partitioning of the network into groups $s \in S$:



$$Q \propto \sum_{s \in S} [\, (\text{\# edges within group } s) - \underbrace{(\text{expected \# edges within group } s)\,]}_{\text{Need a null model!}}$$

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

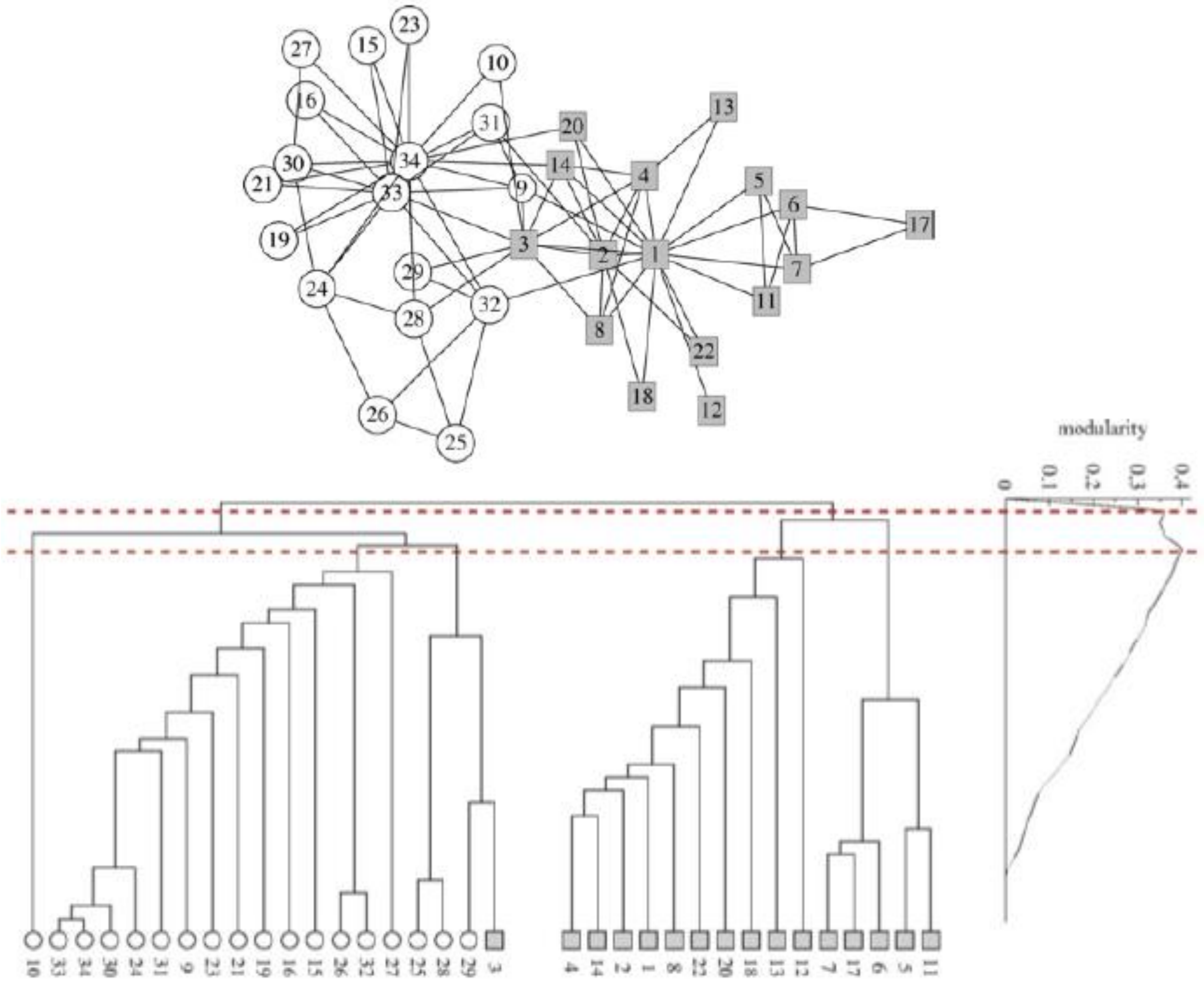
Modularity

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

$A_{ij} = 1$ if $i \rightarrow j$, 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

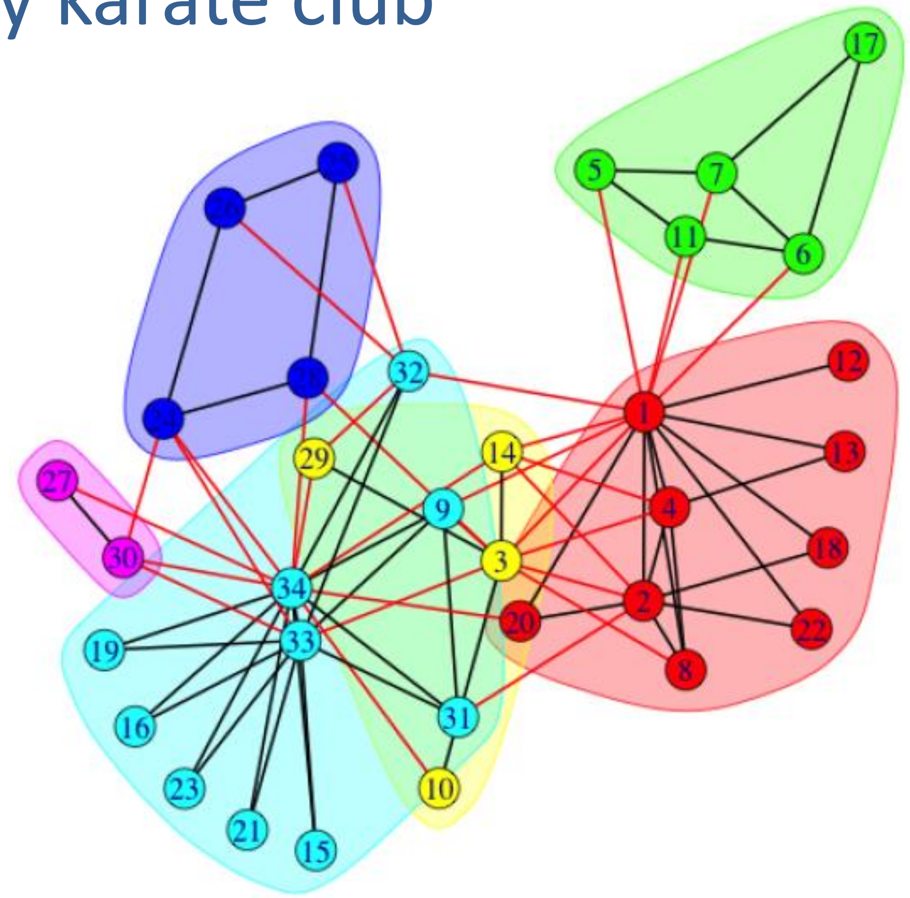
Dendrogram and modularity score



Newman and Girvan, 2004

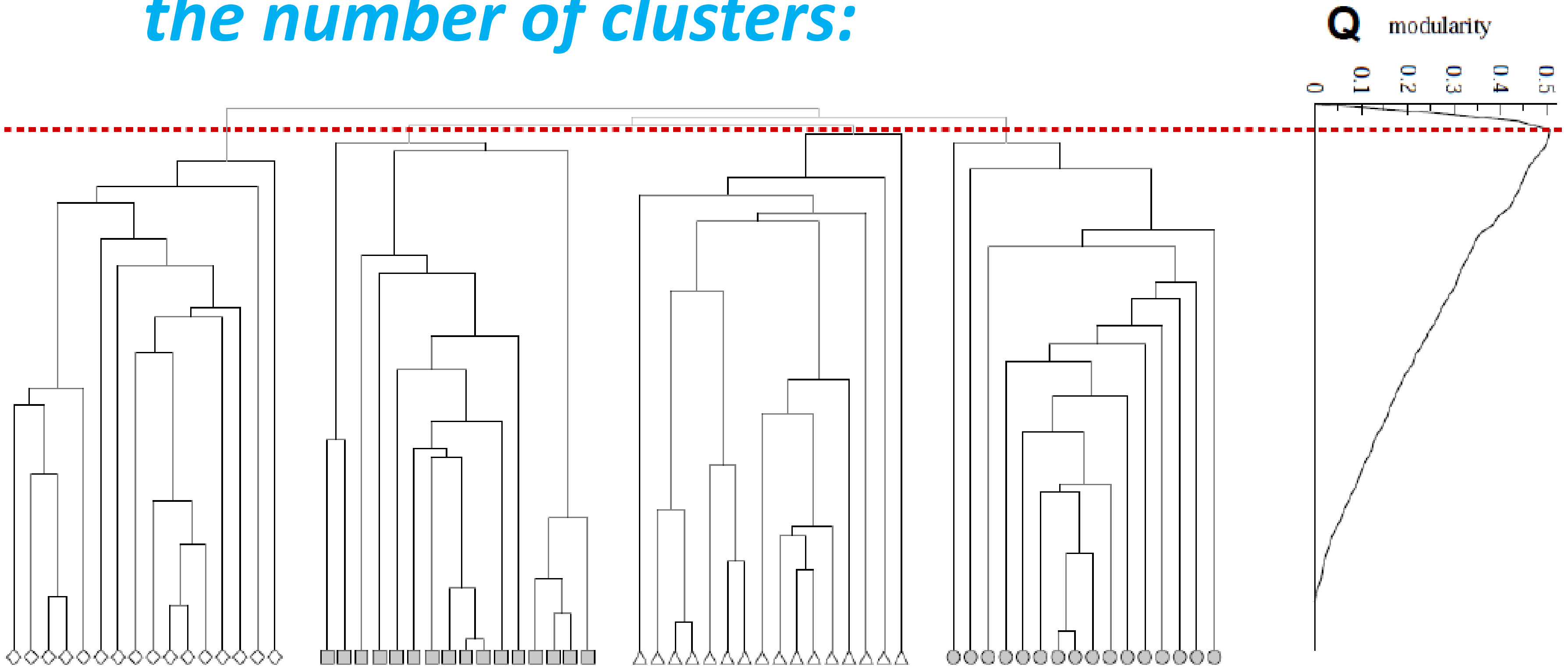
Network communities

- Zachary karate club



Modularity: Number of Clusters

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



Next time: Why not optimize Modularity directly?

Modularity

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

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

- **Modularity $\boldsymbol{Q}$ tells us whether $\boldsymbol{S}$ represents any significant community structure**
  – So, let's find $\boldsymbol{S}$ that maximizes modularity itself!

Method 3: ***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**

$$\frac{s_i s_j + 1}{2} = \begin{matrix} 1\text{.. if } s_i = s_j \\ 0\text{.. else} \end{matrix}$$

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

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

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

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

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

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

NOT RESP.

Quick Review of Linear Algebra

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

- **Symmetric** matrix **A**
  - is Positive Semidefinite: $\boldsymbol{A} = \boldsymbol{U} \cdot \boldsymbol{U}^T$
- Then solutions $\boldsymbol{\lambda}, \boldsymbol{x}$ to equation $\boldsymbol{A} \cdot \boldsymbol{x} = \lambda \cdot \boldsymbol{x}$:
  - **Eigenvectors** $\boldsymbol{x_i}$ ordered by the magnitude of their corresponding **eigenvalues** λ_i $(\lambda_1 \leq \lambda_2 \dots \leq \lambda_n)$
  - $\boldsymbol{x_i}$ 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)
- **Eigendecomposition theorem:** Can rewrite matrix $\boldsymbol{A}$ in terms of its eigenvectors and eigenvalues:

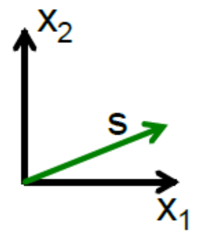
$$\boldsymbol{A} = \sum_i \boldsymbol{x_i} \cdot \lambda_i \cdot \boldsymbol{x_i^T}$$

Modularity Optimization

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

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

- **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 ($\|s\| = 1$)!
  - 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:
  - $s_j = \begin{cases} +1 & if\ x_{n,j} \geq 0 \text{ (j–th coordinate of } x_n \geq 0) \\ -1 & if\ x_{n,j} < 0 \text{ (j–th coordinate of } x_n < 0) \end{cases}$
- **Continue the bisection hierarchically**

Summary: Modularity Optimization

- **Fast Modularity Optimization Algorithm:**
  - Find leading eigenvector $\boldsymbol{x_n}$ of modularity matrix **B**
  - Divide the nodes by the signs of the elements of $\boldsymbol{x_n}$
  - 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 $\boldsymbol{x_n}$? Power method!**
  - Start with random $\boldsymbol{v^{(0)}}$, repeat :
  - When converged ($\boldsymbol{v^{(t)} \approx v^{(t+1)}}$), set $\boldsymbol{x_n = v^{(t)}}$

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

Summary: Modularity based Algorithms

- **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 into an eigenvalue problem

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***

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/