Analyzing Massive Data Sets

Summer Semester 2019

Prof. Dr. Peter Fischer

Institut für Informatik

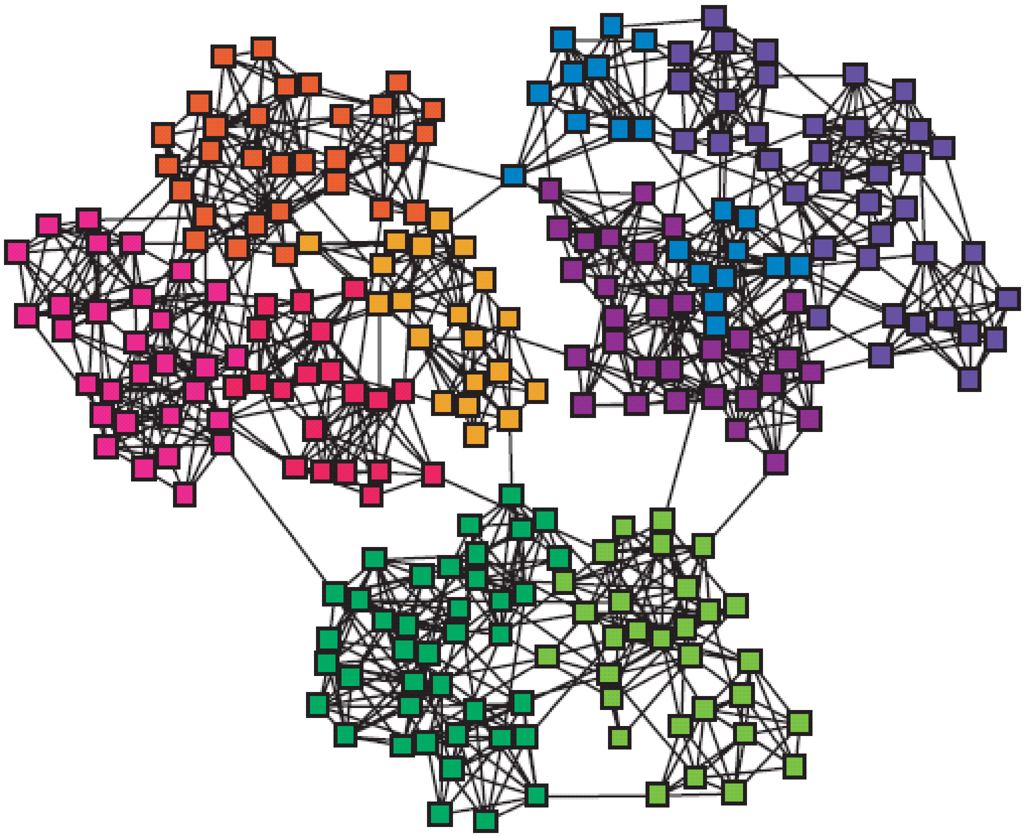
Lehrstuhl für Datenbanken und Informationssysteme

Chapter 8: Graph Structure

Community Detection

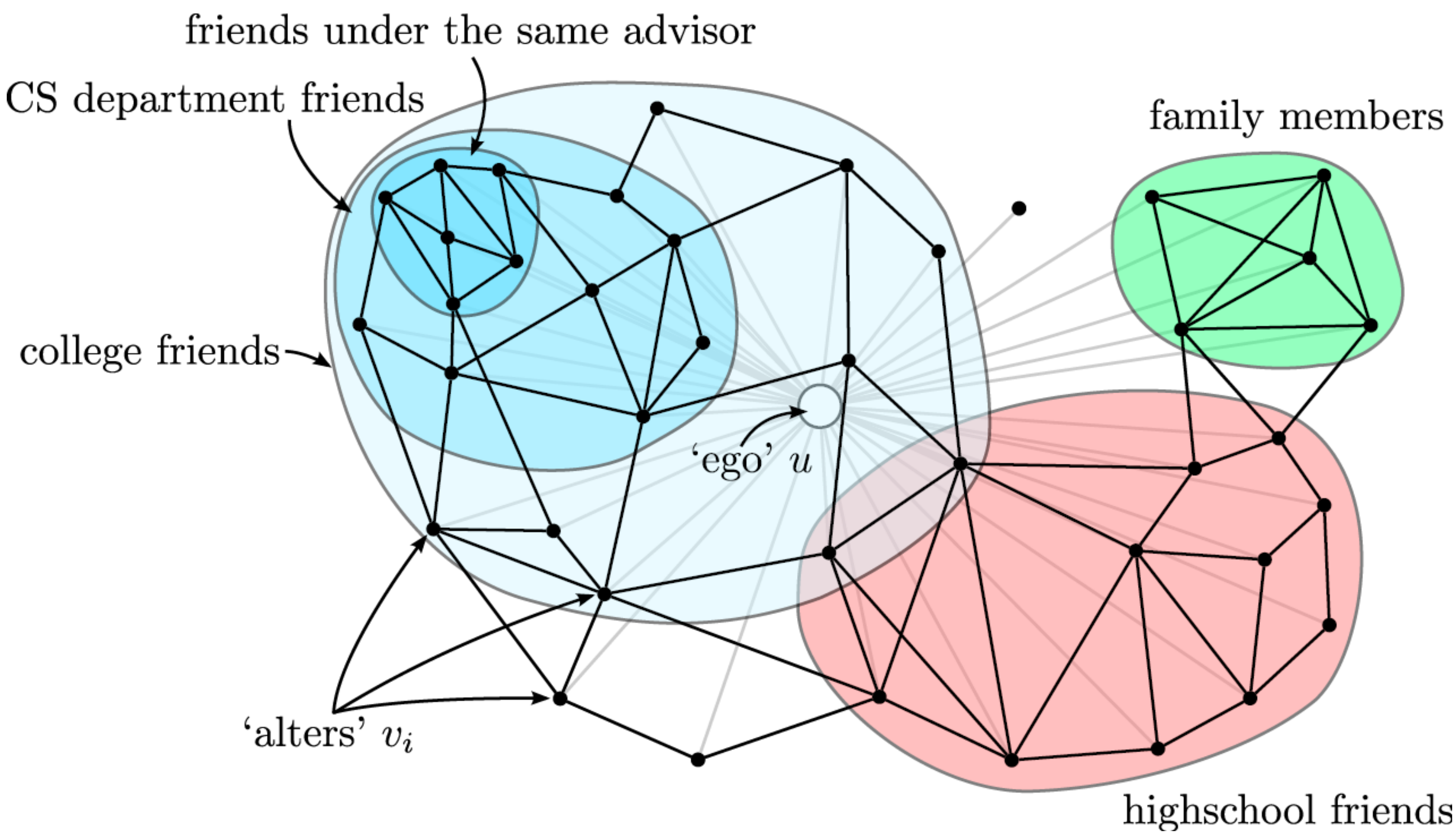
Networks & Communities

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



Twitter & Facebook

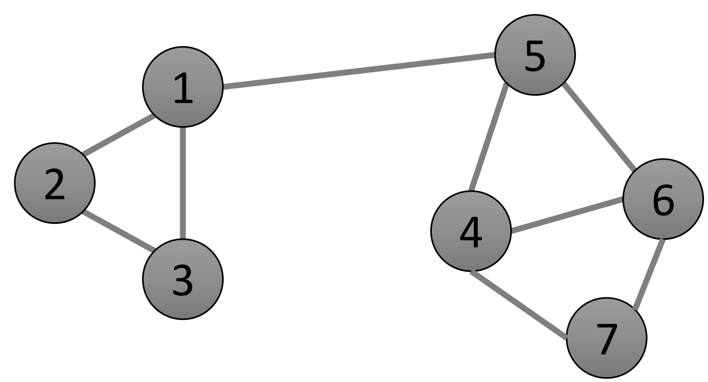
- **Discovering social circles, circles of trust:**



[McAuley, Leskovec: Discovering social circles in ego networks, 2012]

Naive Idea: Standard Clustering on Graphs

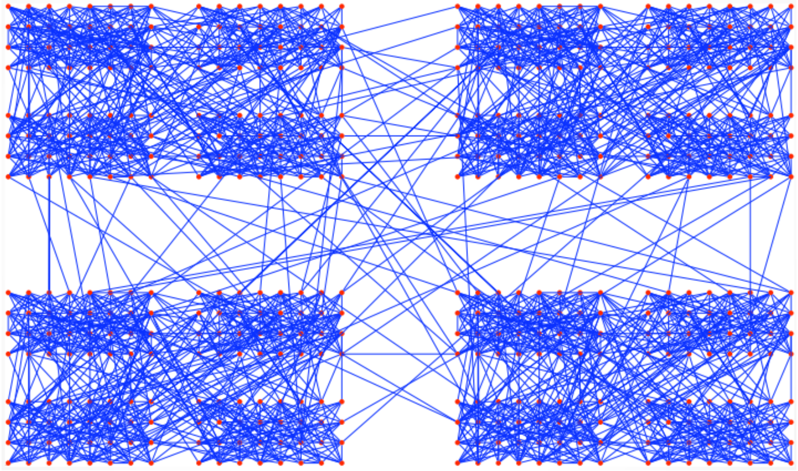
- Why not hierarchical clustering or point assignment (k-means)?
- What would be an appropriate distance function?
  - Edges between nodes?
  - 1 connected, 0 not connected   Similarity, not distance!
  - 0 connected, 1 not connected?
  - 1 connected, inf not connected?
- Violate triangle constraint
  - Maybe 1, 1.5?
- Does not capture the structure of the graph:
  - Random pairs of nodes are combined
  - Eventually, all parts of a connected components are part of the cluster



Background: (Social) Graph Concepts and Observations

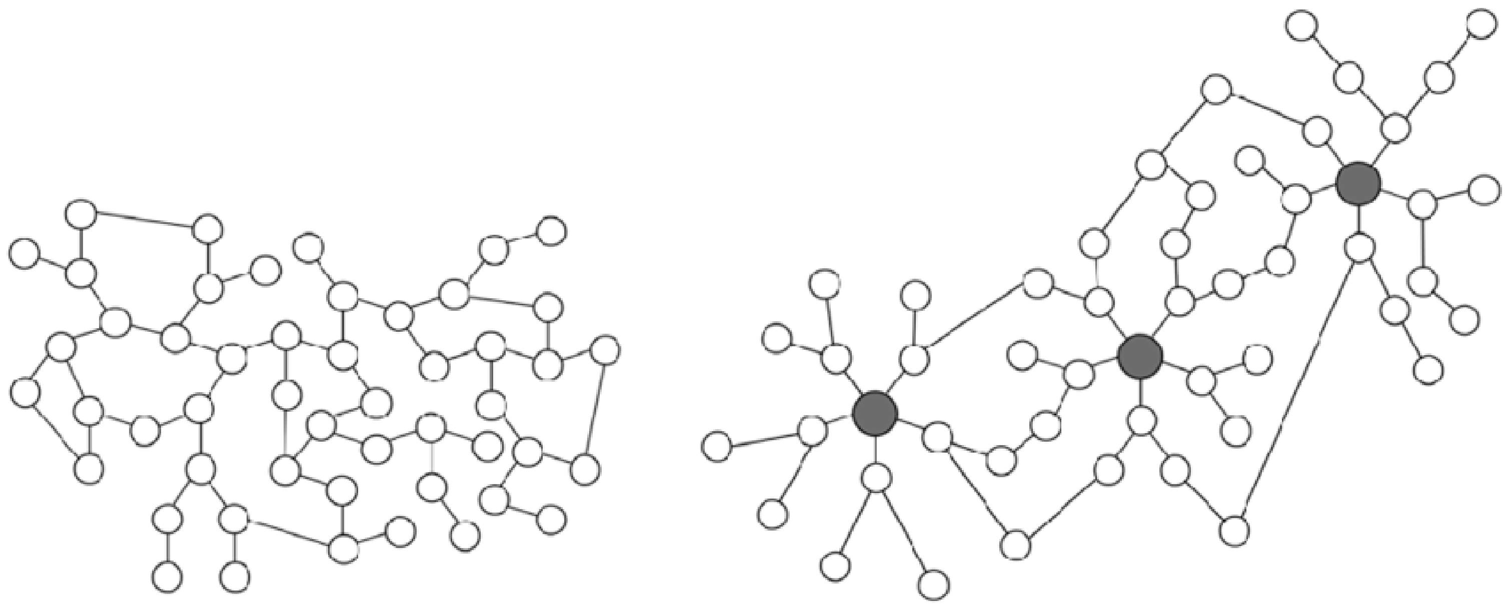
Observations on real-life networks

- Graphs representing real systems are neither regular (lattices), nor random
- The distribution of number of links per node of many real networks is different from what is expected in random networks
- Skewed distributions: The degree distribution is broad, with a tail that often follows a power law
  - Proteins interaction nets: some protein act as hubs, they are highly connected, while most of the others interact only with few other
  - Biological nets: high degree nodes systemically link to nodes with low degree
  - Social nets: nodes with similar degree tend to link each other
- The scale of organization of complex networks shows a hierarchical structure



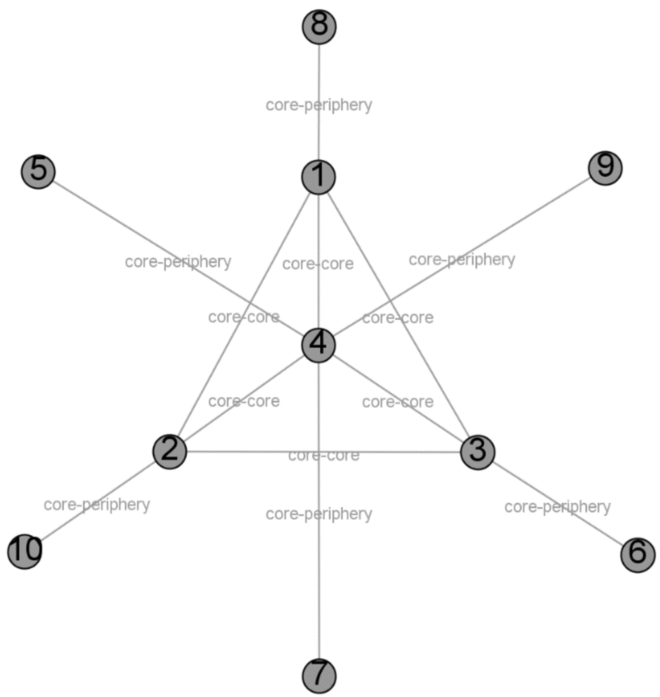
Degree Distributions indicate Graph Type

- Number of links per nodes (aka degrees) played an important role in PageRank and HITS



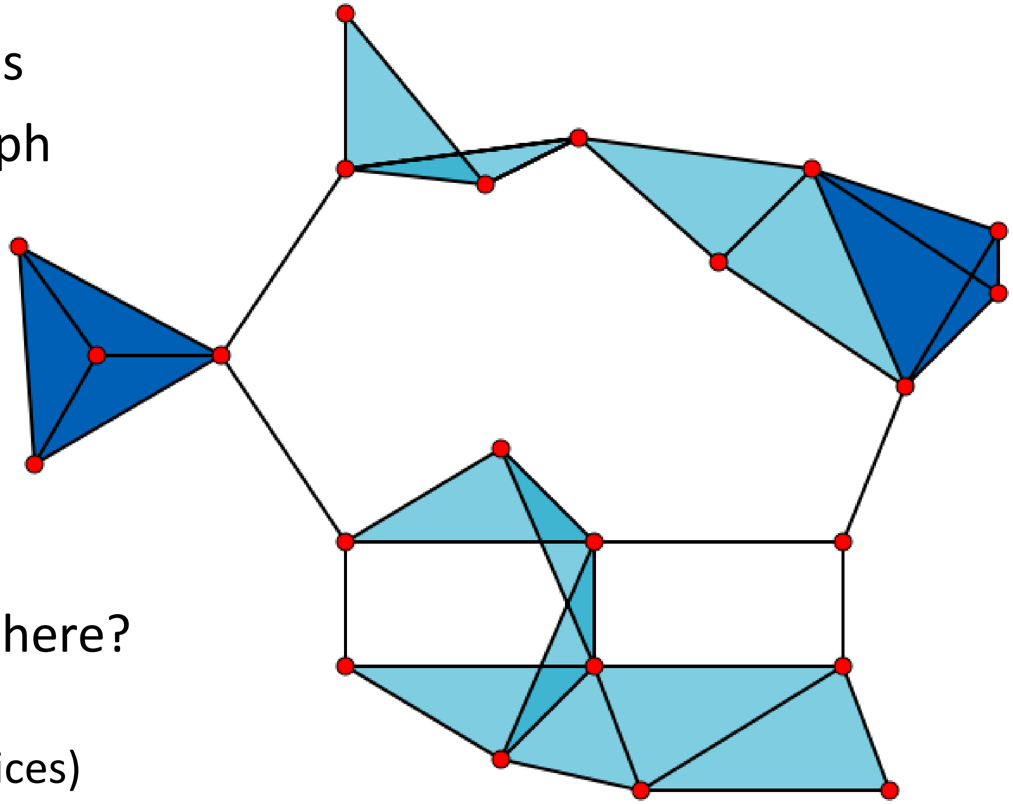
- Degree Distribution is an indicator of the type of graph

Example Structure: Core Periphery



Cliques

- Describe neighborhoods
- Fully connected subgraph



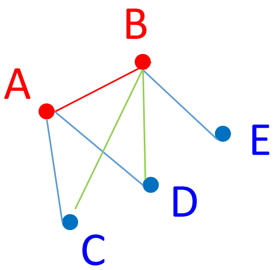
- How many cliques are there?

23 × 1-vertex cliques (vertices)
42 × 2-vertex cliques (edges),
19 × 3-vertex cliques
(light and dark blue triangles)
2 × 4-vertex cliques (dark blue areas)

Clique Relaxations

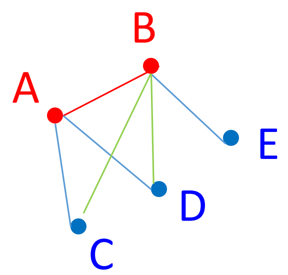
- Problems with cliques:
  - too strict condition
  - vertices are symmetric (wrong assumption for real social networks),
  - cliques are hard to find: NP-complete problem.
- N-clique: subgraph such that the distance between each pair of vertices does not exceed n (variant n-clan)
- K-plex: maximal subgraph such that each vertex is adjacent to all other vertices of the subgraph except at most k of them
- K-core: maximal subgraph such that each vertex is adjacent to at least k other vertices of the subgraph

Triads



- Consider:
  - Two arbitrarily selected individuals A and B and
  - The set S = C,D,E of all persons with ties to either or both of them
- Hypothesis:
  - The stronger the tie between A and B, the larger the proportion of individuals in S to whom they will both be tied.
- Theoretical corroboration:
  - Stronger ties involve larger time commitments - probability of B meeting with some friend of A (who B does not know yet) is increased
  - The stronger a tie connecting two individuals, the more similar they are
- Perform link prediction and recommendation (common underpinning of friend recommender in Facebook or LinkedIn)

Counting Triangles



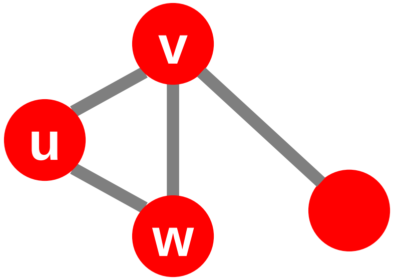
- Most naive solution
  - enumerate all triples of points: ABC, ABC, ABE, BCD, …
  - Check if edge exists
- Cost
  - Notation: G = (V,E) n=|V| m=|E|
  - O(n^3)   n * (n-1) * (n-2)
  - How good is this?
  - How many triangles may exist?
  - Rephrasing in #edges: $O\left(m^{\frac{3}{2}}\right)$: $\frac{Number\ of\ triangles}{Number\ of\ edges}$
  - Best possible solution for all fully connected graph!
- Several approaches exist that achieve $O\left(m^{\frac{3}{2}}\right)$ for sparse graphs
- A matrix-multiplication approach achieves $m^{1.41}$, but very memory-intensive

Node-centric computation

```
foreach v in V
        foreach u,w in adjacency(V)
                if (u,w) in E
                        triangles[v] ++
```

Runtime:

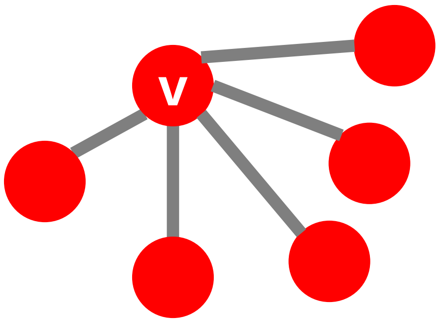
$$\sum_{v \in V} \deg(v)^2$$



- Possibly faster for sparse matrices
- Can be easily parallelized (e.g., per node)
- How many triangles do we get?
- What about skewed graphs – super-popular users?
- Not good enough in worst case: $O(n^2) = O(m^2)$

Node-centric computation (with a twist)

```
foreach v in V
        foreach u,w in adjacency(V)
                if deg(v) < deg(u) && deg(v) < deg(w)
                        if (u,w) in E
                                triangles[v] ++
```



- Only compute triangles for node with smallest degree
- In real-life networks, triplets of high-degree nodes are very infrequent
- Reduced skew also helps for parallelization
- Complexity claim: $O\left(m^{\frac{3}{2}}\right)$

Proof Sketch for Complexity (1)

- For reasoning, split nodes into two degree groups
  1. $\deg(v) > \sqrt{m}$ (big nodes)
  2. $\deg(v) \leq \sqrt{m}$ (small nodes)

- For 1)
  - at most $2\sqrt{m}$ such nodes may exist
    - Per node at least $\sqrt{m}$ degree
    - Num nodes * degree = total edges
    - Thus: $2\sqrt{m}$ * $\sqrt{m}$ = 2m
    - 2m (instead of m): in+out edges per node counted twice
  - There can be at most $\sqrt{m}^3 = m^{\frac{3}{2}}$ triangles
    - At each node, there might be at most a fully connected neighborhood ~ naive enumeration

Proof Sketch for Complexity (2)

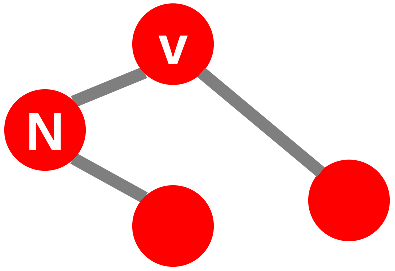
- For 2)
  - Maximize $\sum_{v \in V} \deg(v)^2$ (e.g., using convex optimization)
  - Constraints:
    - $\deg(v) \leq \sqrt{m}$ (by definition)
    - $\sum_{v \in V} \deg(v) \quad < 2m$ (like in 1)
  - Maximum value for: $2\sqrt{m}$ summands, $\sqrt{m}$ degree
  - Thus: $\sum_{v \in V} \deg(v)^2 = 2\sqrt{m} * \sqrt{m}^2 = m^{\frac{3}{2}}$

k-core computation

- Cliques are NP-Hard
- What about k-cores?
- Maximal Subgraphs where deg(n) >= k
- Can be computed in O(m)
- Algorithm determines core degree of every node
- Actual core components need another traversal

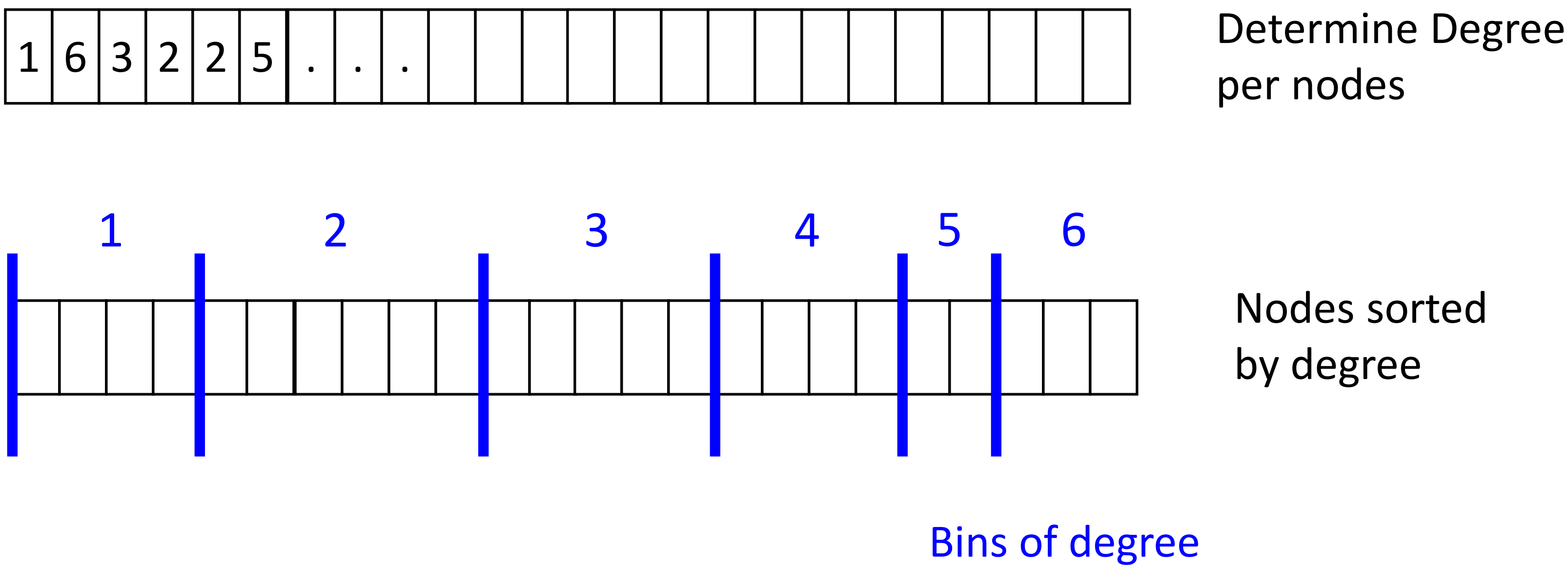
Algorithm Sketch

- Sort all nodes by increasing degree (array+bins)
- Traverse nodes v once
  - For every unvisited neighboring N node with degree > current node, decrease degree by one
  - Shift N down to lower bin

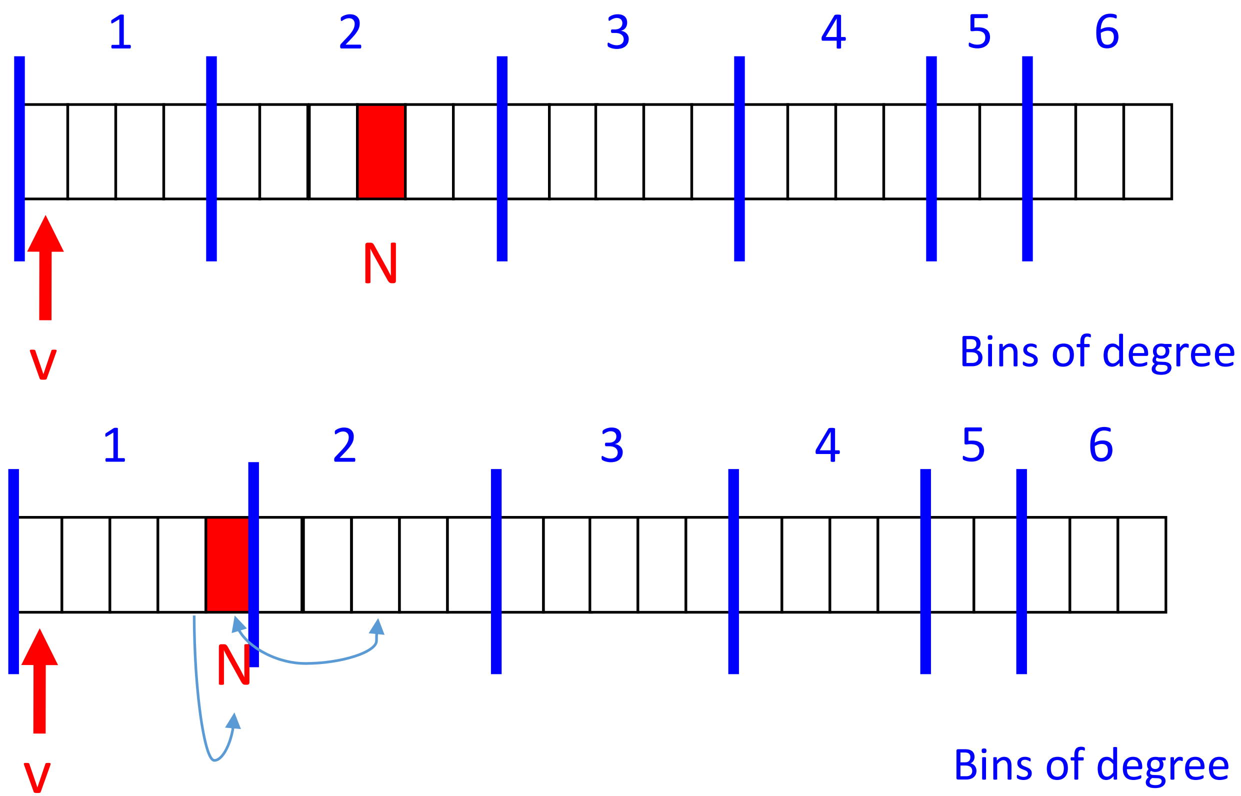


- Idea:
  - Original degree is an upper bound for cores
  - a neighbor with lower degree does not help to maintain the current degree
  - Therefore, it should not count, reducing the effective degree

Describing the computation



Describing the computation



Visit N,
reduce degree

Move to Bin -1

Time Complexity: O(m)

- Compute degrees per node, max degree: max(m,n)
- Sorting into bins: O(m) - Bucketsort
- Traversing all nodes (single pass) O(n)
  - Never need to go back. Why?
- Visiting all neighbors O(m)
- For a connected network m > n-1

Clustering Coefficient

- How close are the neighbors of a node to a clique?
- Count triangles!
- Global definition:

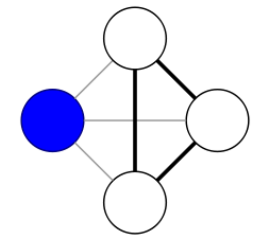
$$C = \frac{number\ of\ closed\ triplets}{number\ of\ all\ triples}$$

- Local definition

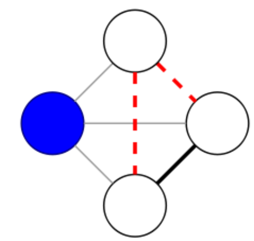
$$C_i = \frac{2|\{e_{jk}: v_j, v_j \in N_i, e_{jk} \in E\}|}{k_i(k_i - 1)}$$

- Global: arithmetic mean of all local coefficents

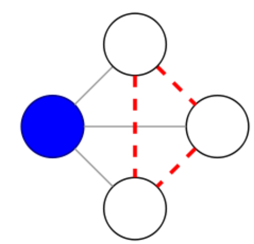
- Random networks
  - Small clustering coefficients
  - Sparsity does not allow much connectivity
- Small worlds networks
  - high clustering coefficients
  - Plenty of neighbors ~ community
  - Yet still many shortcuts



c = 1



c = 1/3



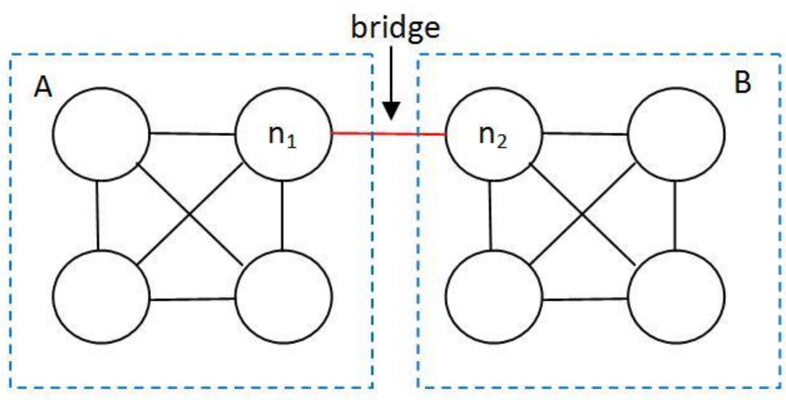
c = 0

Strong and Weak Ties

- Strong and weak ties in social graphs
  - Ties/relationships vary in intensity
  - People who have strong ties tend to share a similar set of acquaintances
  - Ties change over time
  - Nodes (people) have different characteristics
- The strength of an interpersonal tie is a
  - (probably linear) combination of the amount of time
  - The emotional intensity
  - The intimacy
  - The reciprocal services which characterize the tie

Strong and Weak Ties

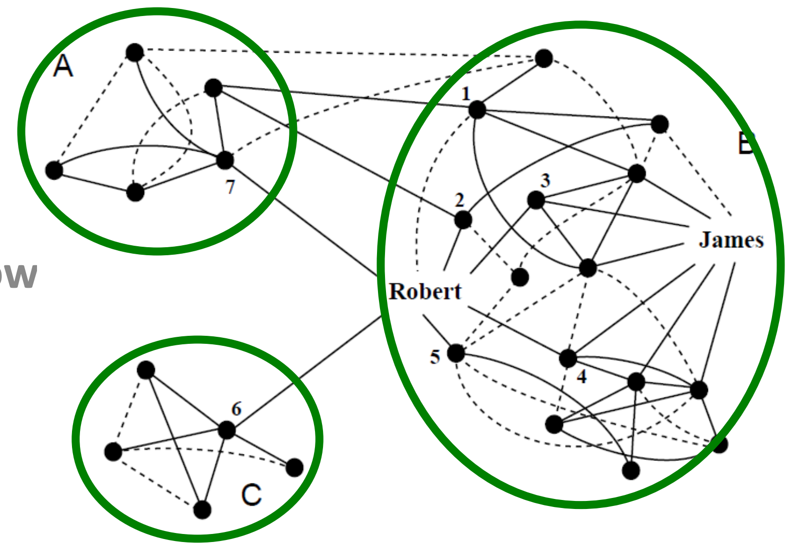
- A bridge is an edge in a network which provides the only path between two points.
  - In social networks, a bridge between A and B provides the only route along which information or influence can flow from any contact of A to any contact of B
  - Bridges are weak ties

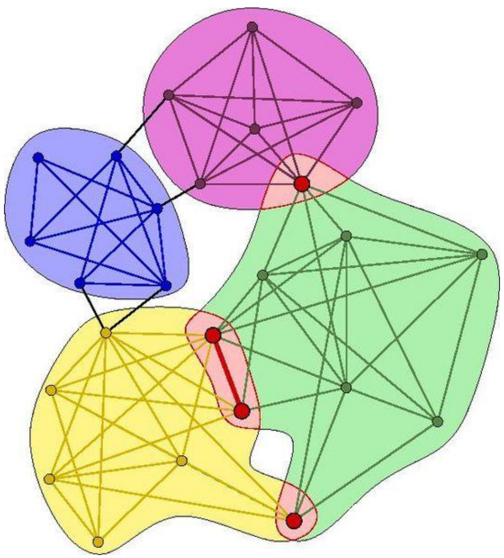


- In real-life networks, more than one connection exists among components -> local bridge

Community Detection

How





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

Defining Communities

- Informally, a community C is a subset of vertices of V such that there are more edges inside the community than edges linking vertices of C with the rest of the graph
- Intra Cluster Density > Inter Cluster Density
- Connectedness is a prerequisite (for every pair of vertices there must exist a path)
- Community Detection makes sense in sparse graphs
- There is no universally accepted definition of community: dependent on individual applications
- Different Approaches:
  - Focus on the subgraph (community): clique, k-core, …
  - Comparison between internal and external cohesion of the subgraph
  - Comparison between subgraph and the whole system

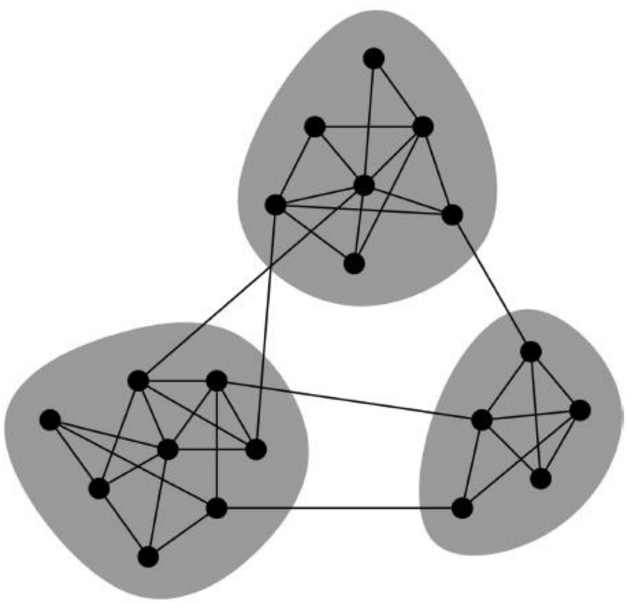
Comparison-based communities

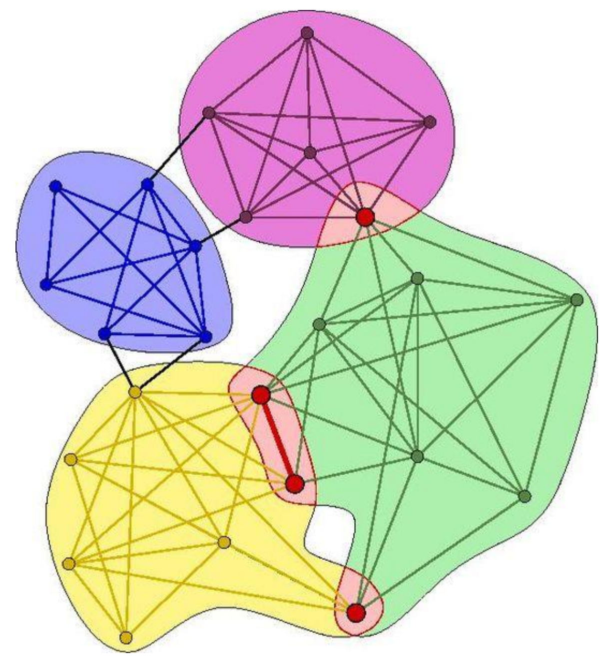
- Comparison between internal and external cohesion of the subgraph
  - Strong community: subgraph such that the internal degree of each vertex is greater than its external degree
    - Problem: condition too strong, unrealistic in practical cases
  - Weak community: subgraph such that the internal degree of the subgraph is greater than its external degree
  - Many other variants exist for strong and weak communities

- Comparison between subgraph and the whole system
  - Null models, i.e. randomized versions of the original graph
  - Most popular null model: random graph with the same expected degree sequence of the original graph

Partitions vs Covers for overlapping and non-overlapping communities

- A partition is a division of a graph into clusters, such that each vertex is assigned to one and only one cluster
- If vertices can belong to two or more clusters simultaneously, then we refer to covers





Methods for Community identification

- Based on vertex similarity
- Graph partitioning
- Based on weak ties
- Based on cliques for overlapping communities
- Spectral clustering

Communities based on Vertex similarity

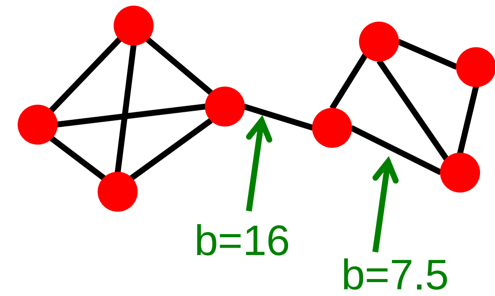
- Clustering Methods
- Communities are subgraphs of vertices ”similar” to each other
- Basic ingredient: measure of similarity between vertices
- Similarity measures essential for methods like hierarchical and spectral clustering
- Two classes of measures:
  - Graphs embedded in euclidean space
  - Graphs not embedded in euclidean space

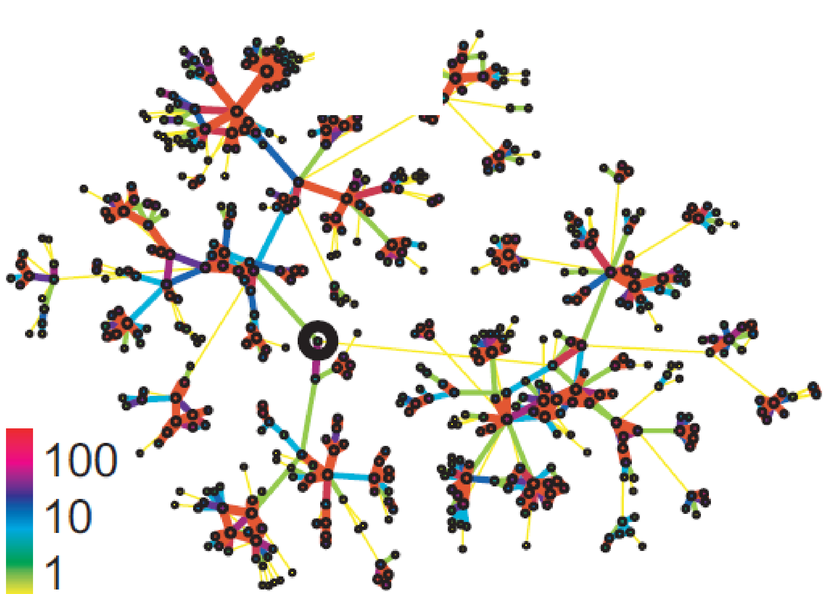
Graph Partitioning

- Graph partitioning
  - Partition the graph in a predefined number of clusters and predefined cluster size, e.g.: Distribute graph over different machines.
  - Normally the cluster size is balanced
  - Usually minimize the cut-edges: edges between different clusters
  - Do not account for the internal structure of the graph
- ... vs Community Detection
  - Goal: Identify structures in the graph
  - Number and size of clusters are not predefined
  - Given the skewed distributions of node degrees, cluster sizes might be highly imbalanced.
  - Many methods to identify clusters (global and local perspective)

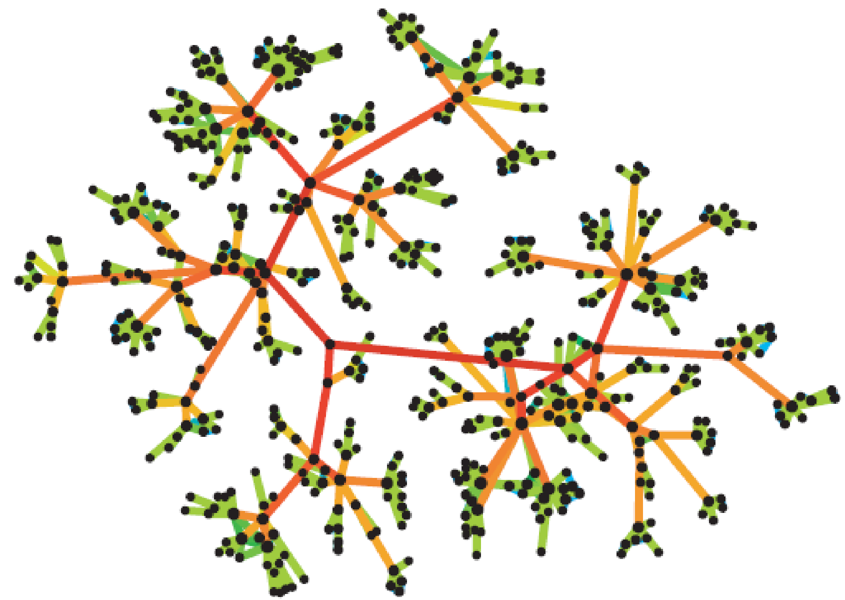
Communities based on Weak Ties

- **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

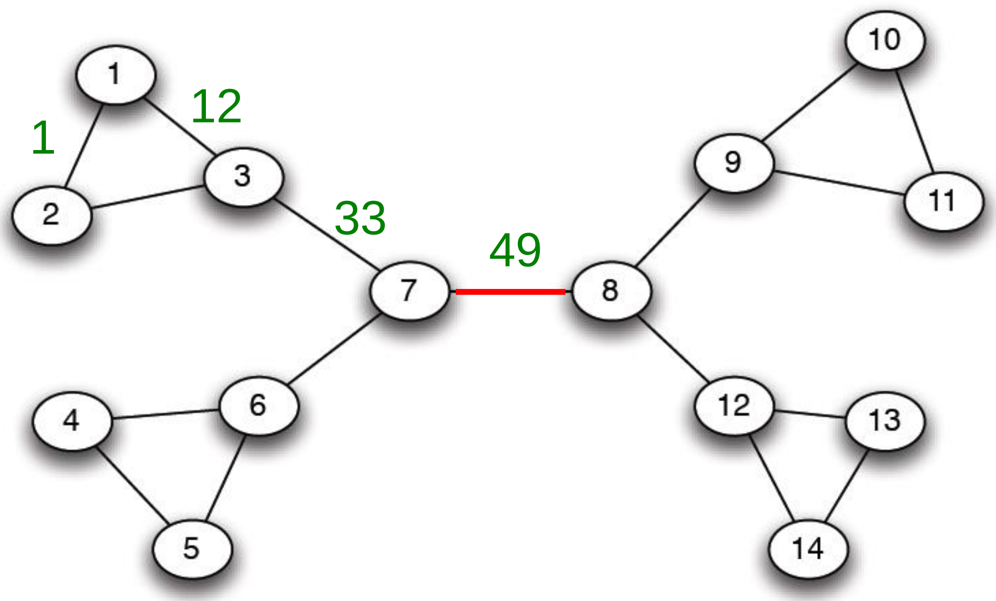
Method 1: Girvan-Newman

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

  Number of shortest paths passing through the edge

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

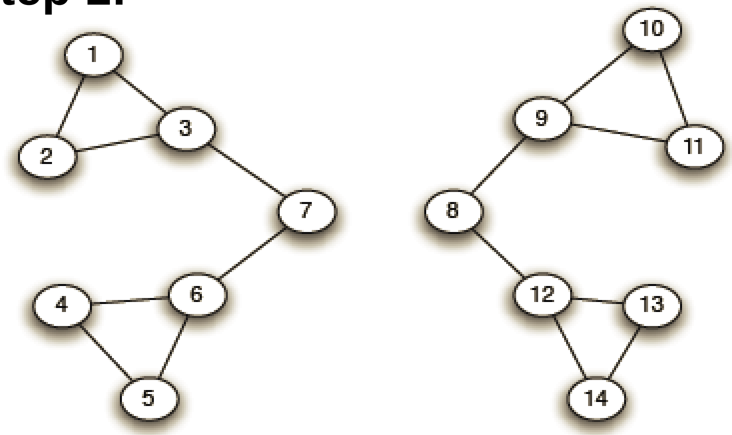
Girvan-Newman: Example



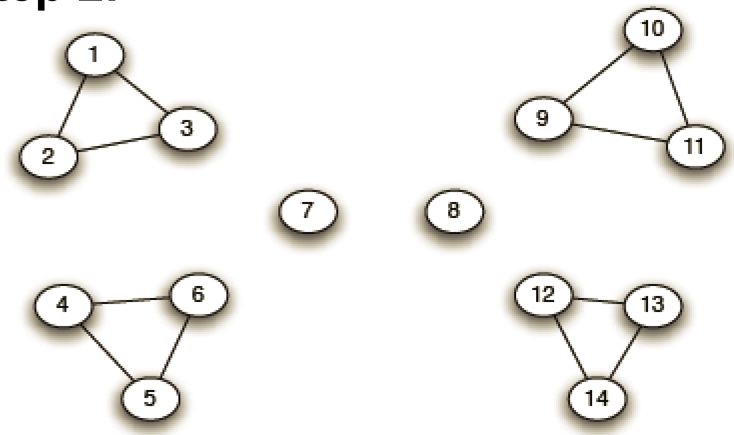
Need to re-compute betweenness at every step

Girvan-Newman: Example

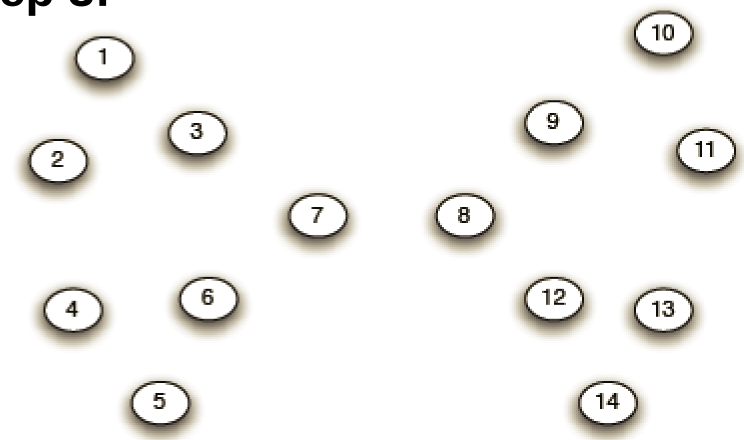
Step 1:



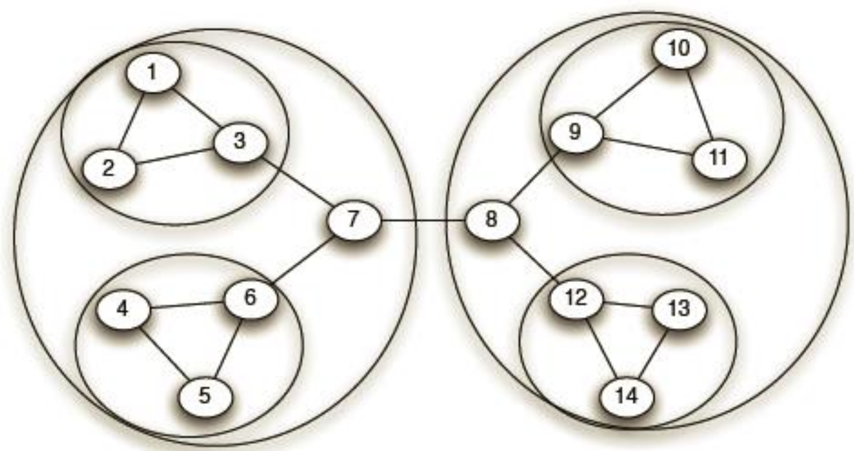
Step 2:



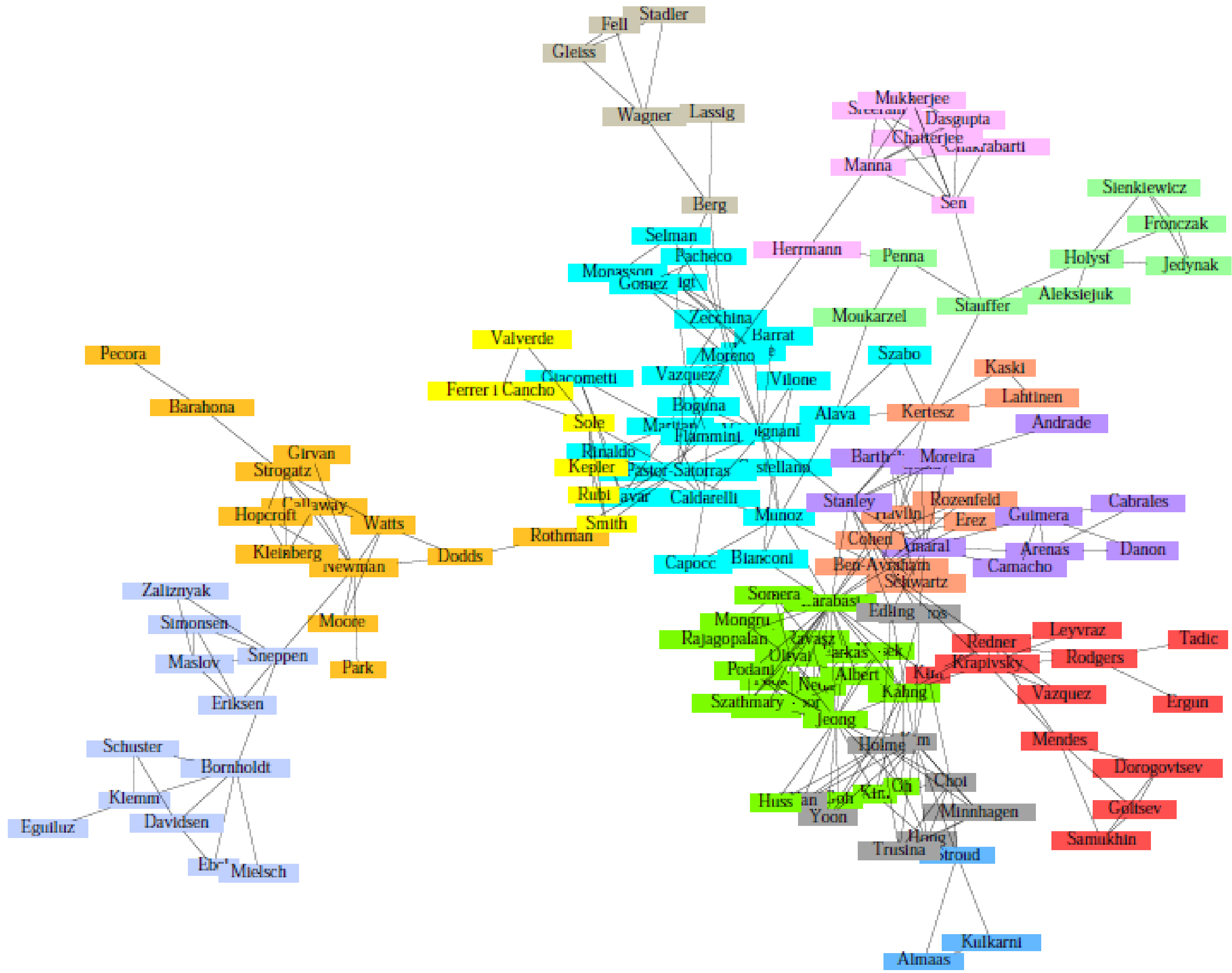
Step 3:



Hierarchical network decomposition:



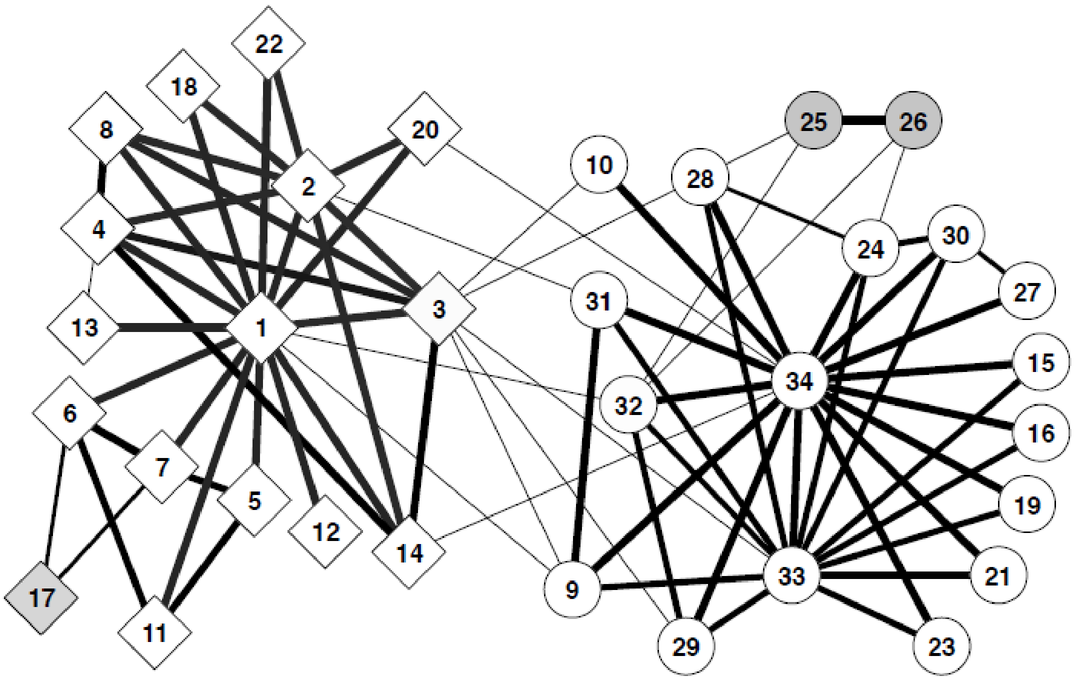
Girvan-Newman: Results

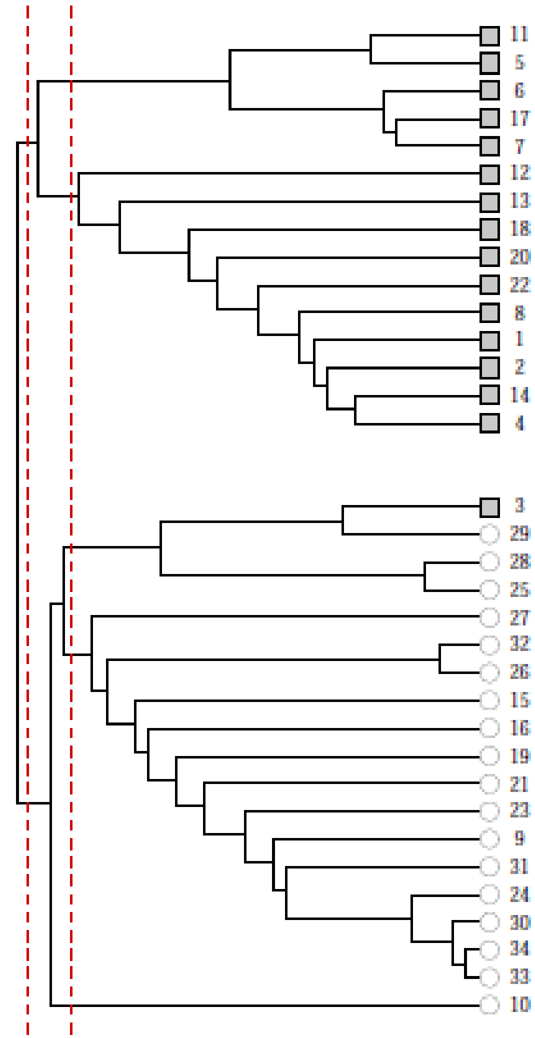


Communities in physics collaborations

Girvan-Newman: Results

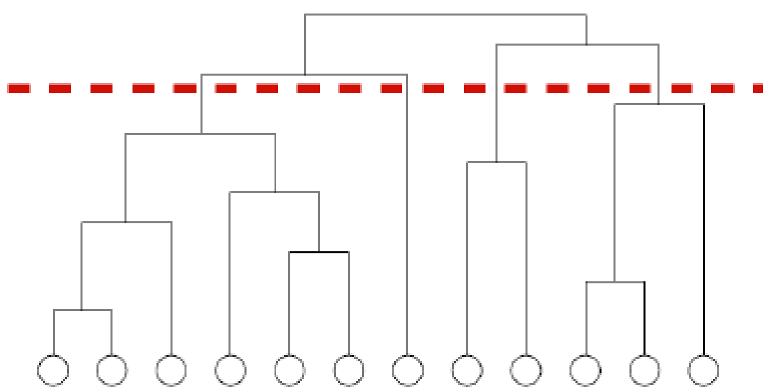
- **Zachary's Karate club:**
  Hierarchical decomposition





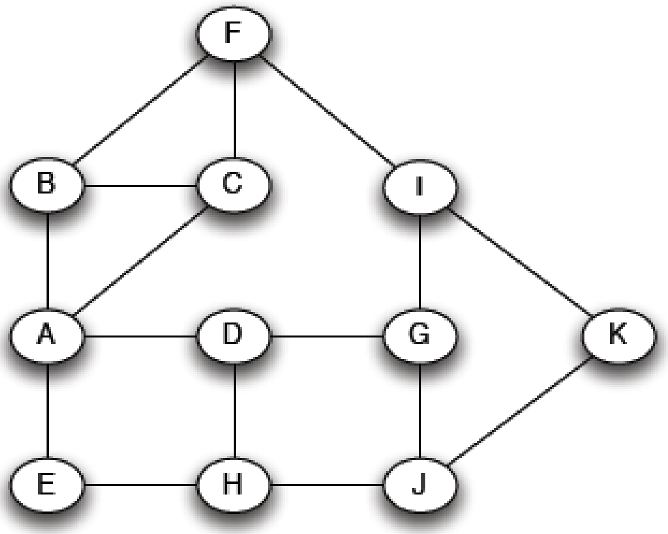
We need to resolve 2 questions

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

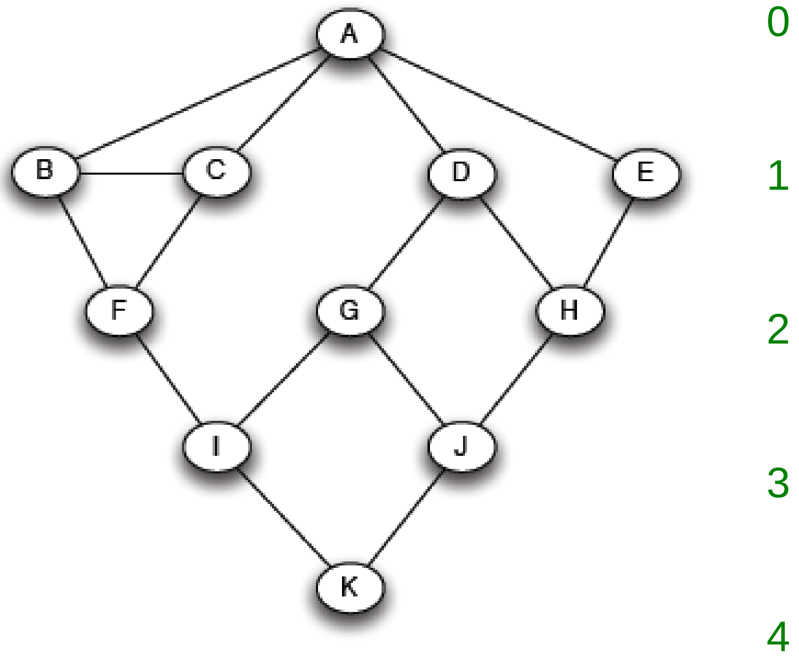


How to Compute Betweenness?

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

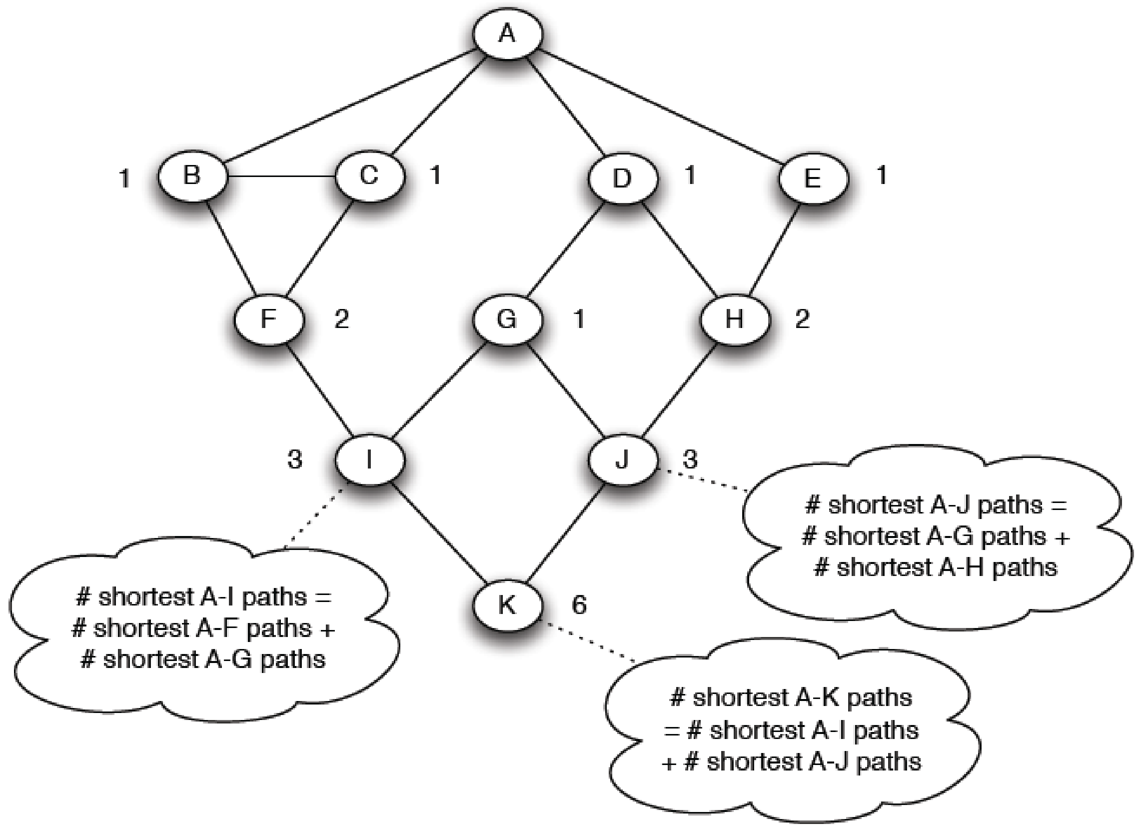


- **Breath first search starting from A:**



How to Compute Betweenness?

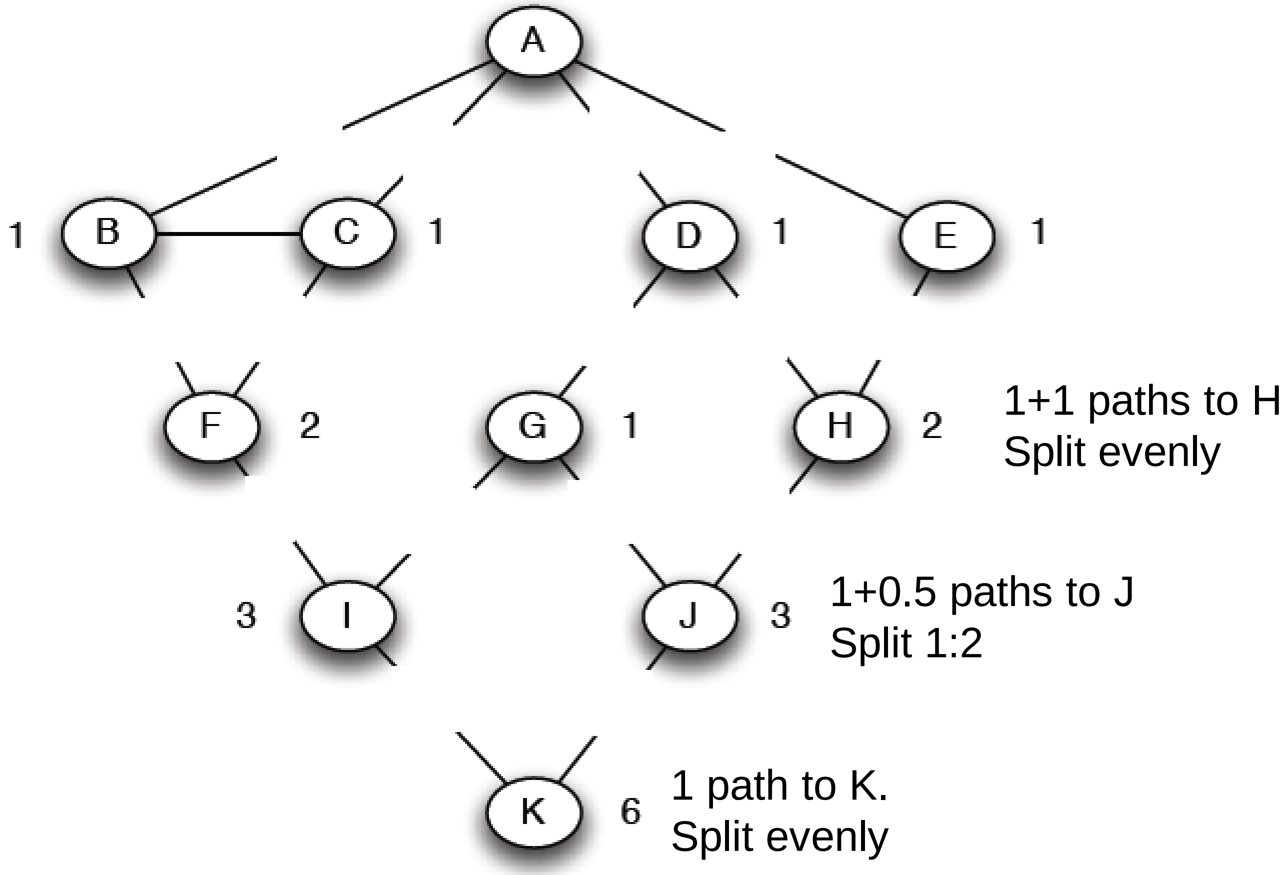
- **Count the number of shortest paths from A to all other nodes of the network:**



How to Compute Betweenness?

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

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



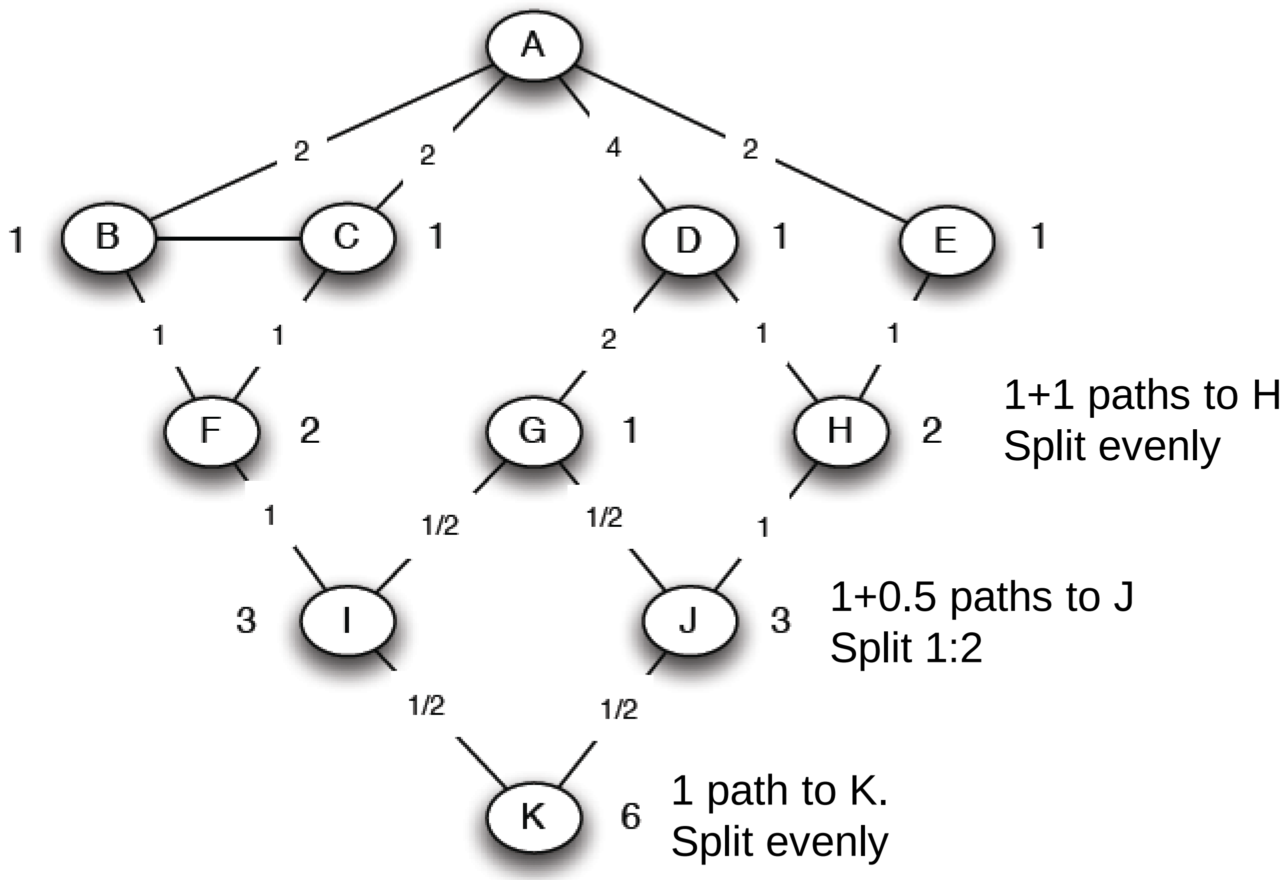
How to Compute Betweenness?

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

The algorithm:

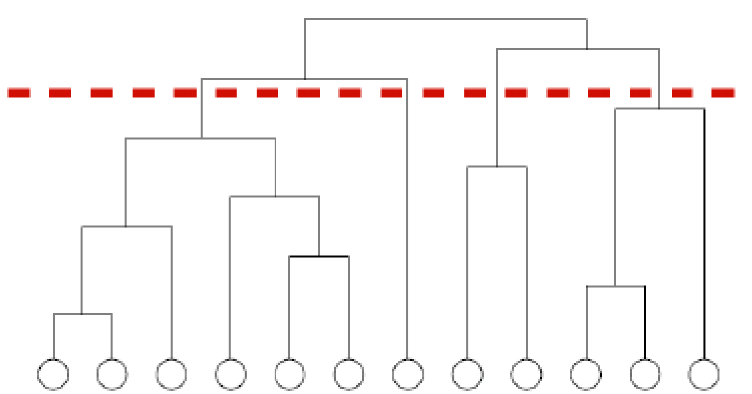
•Add edge **flows**:
  -- node flow = 1+∑child edges
  -- split the flow up based on the parent value

• Repeat the BFS procedure for each starting node *U*



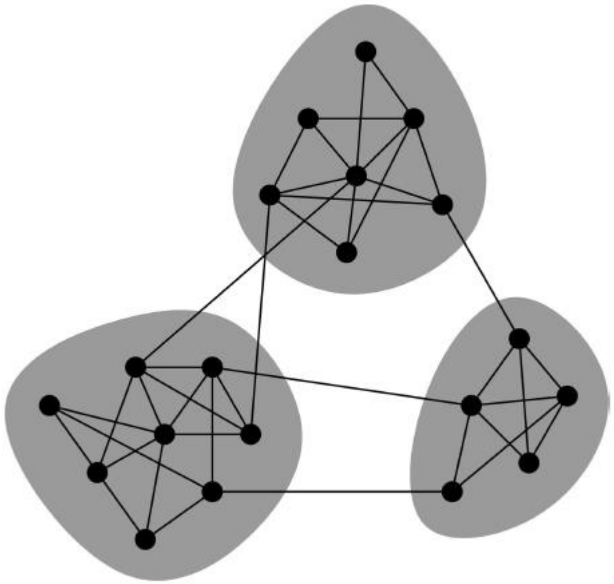
We need to resolve 2 questions

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



Network Communities

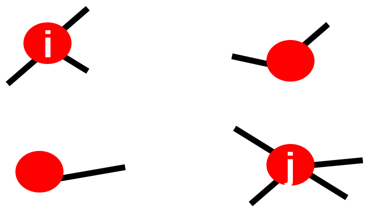
- **Communities:** sets of **tightly connected nodes**
- Define: **Modularity Q**
  - A measure of how well a network is partitioned into communities
  - Given a partitioning of the network into groups $s \in S$:



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

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→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:**