Week 13: Graph Mining (Communities)

Instructor: Daejin Choi (djchoi@inu.ac.kr)

INCHEON NATIONAL UNIVERSITY

* Most slides are used from mmds.org (CS224W) with small updates

Learned so far & Lecture for today

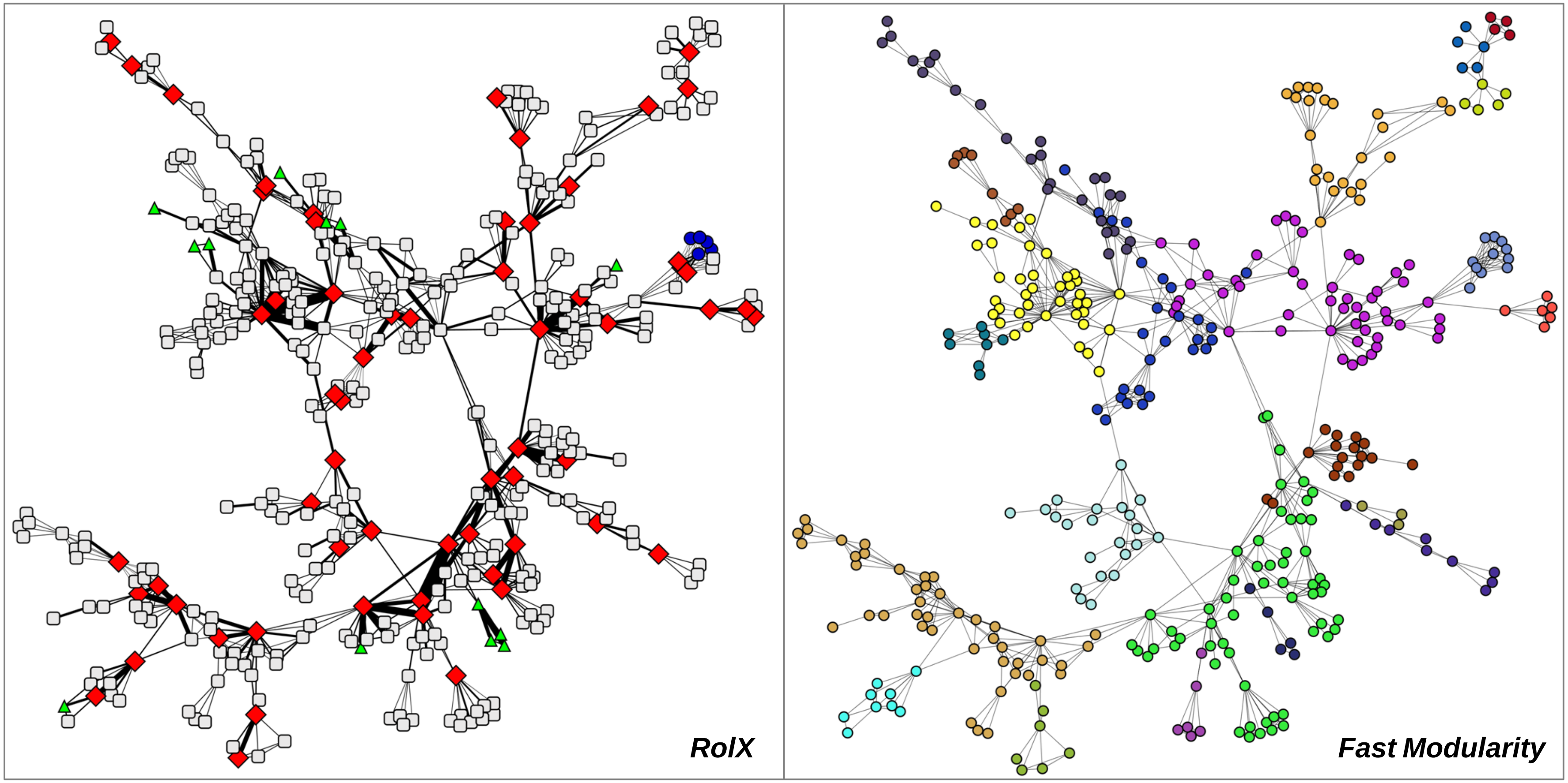
- Understanding Graph Mining!
- Graph Modeling & Applications
- Graph Properties
- Motif Analysis (building blocks)
- Node Analysis (Structural Roles + Centrality)
- **Community Analysis**

Community Analysis: Motivation

Roles and Communities: Example

Past Lecture: Roles

This Lecture: Communities

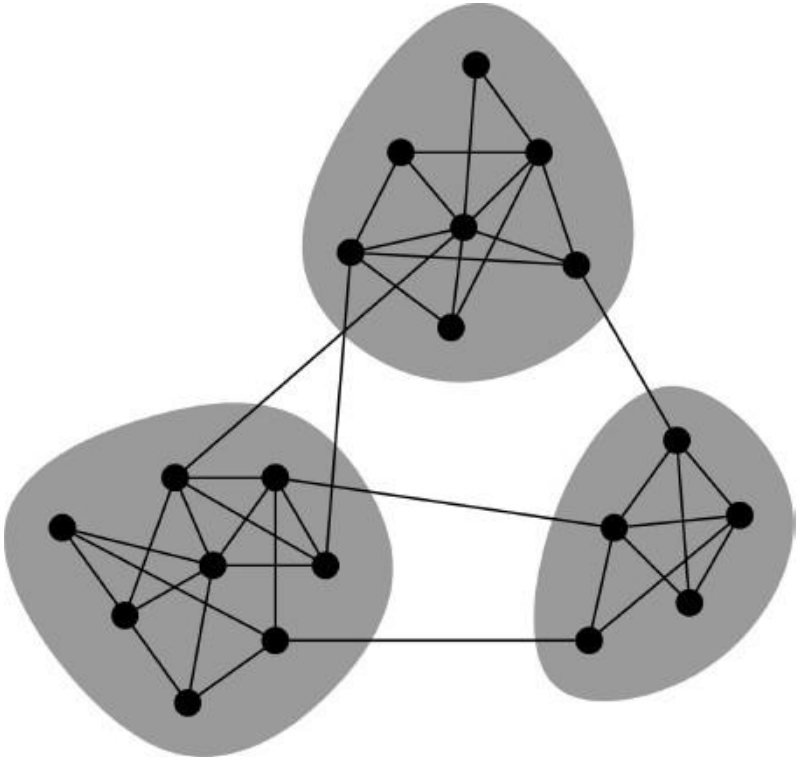


Henderson, *et al.,* KDD 2012

Clauset, *et al.,* Phys. Rev. E 2004

Networks & Communities

- We often think of networks "looking" like this:

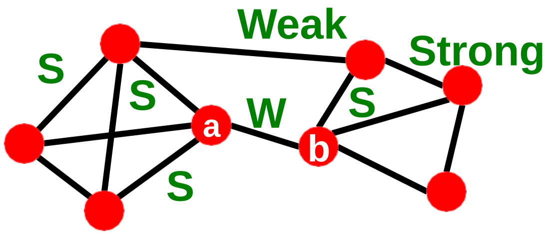


- What led to such a conceptual picture?

Granovetter's Theory

[Granovetter '73]

- Granovetter makes a connection between the social and structural role of an edge
- First point: **Structure**
  - Structurally embedded edges are also socially strong
  - Long-range edges spanning different parts of the network are socially weak
- Second point: **Information**
  - Long-range edges allow you to gather information from different parts of the network and get a job
  - Structurally embedded edges are heavily redundant in terms of information access



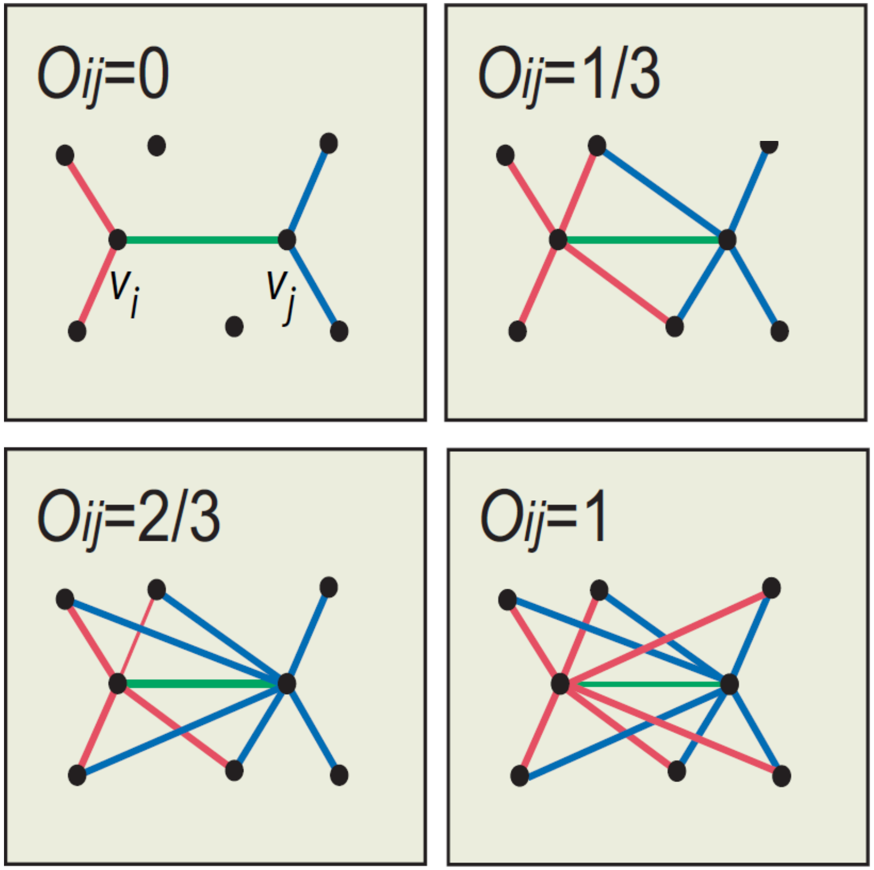
Edge Strength in Real Data

- For many years Granovetter’s theory was not tested
- But, today we have large who-talks-to-whom graphs:
  - Email, Messenger, Cell phones, Facebook
- Onnela et al. 2007:
  - Cell-phone network of 20% of EU country’s population
  - Edge weight: # phone calls

Edge Overlap

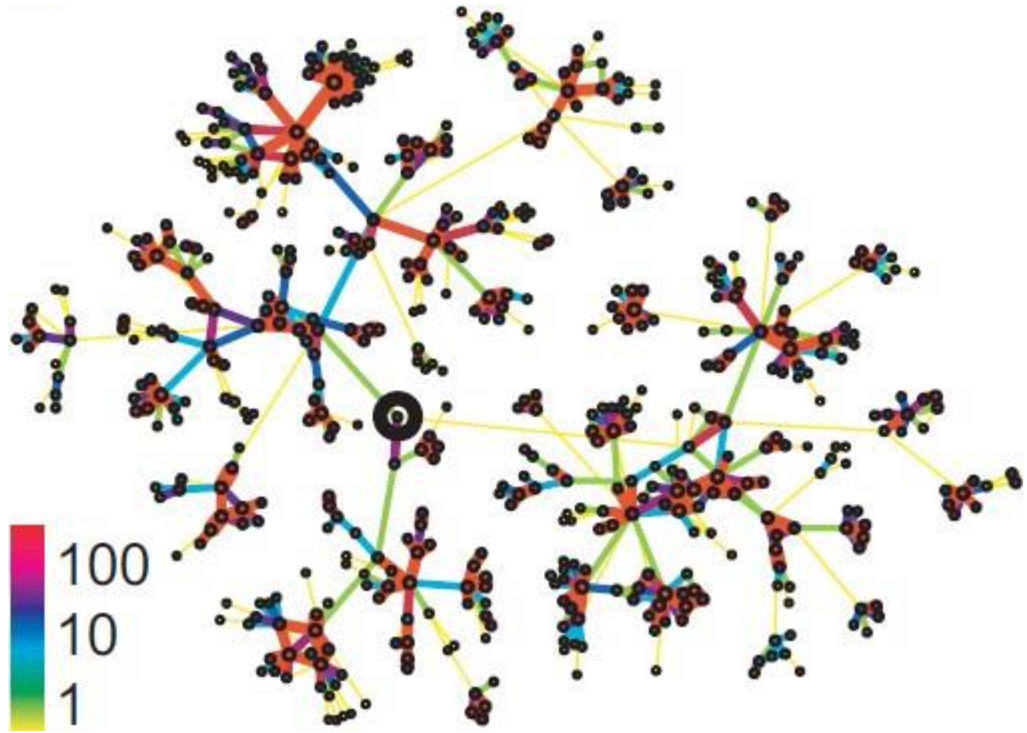
$$O_{ij} = \frac{|(N(i) \cap N(j)) \backslash \{i,j\}|}{|(N(i) \cup N(j)) \backslash \{i,j\}|}$$

$N(i)$ … the set of neighbors of node i



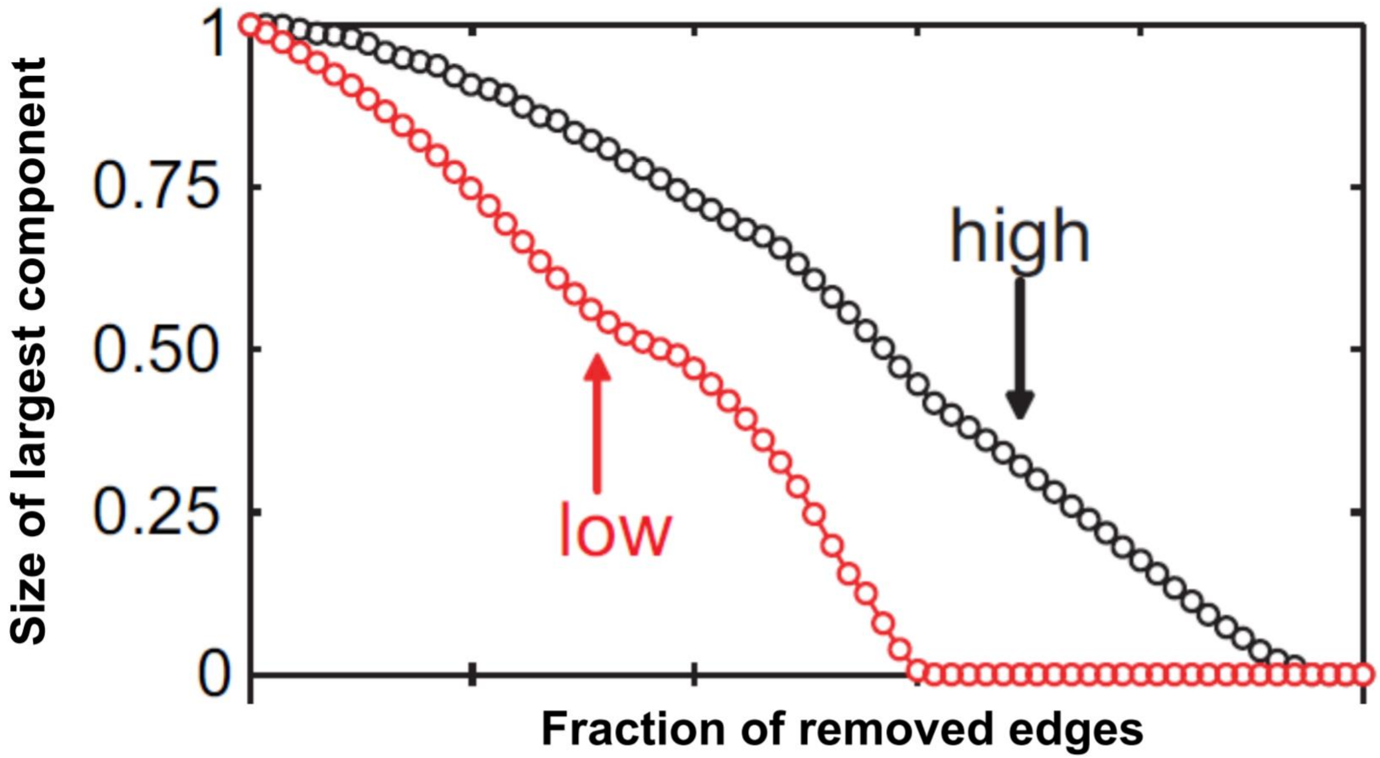
- Note: Overlap = $\mathbf{0}$ when an edge is a local bridge

Real Network, Real Edge Strengths

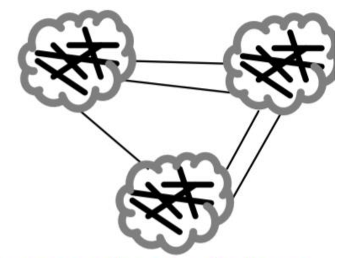


- Real edge strengths in mobile call graph
  - Strong ties are more embedded (have higher overlap)

Link Removed by Overlap



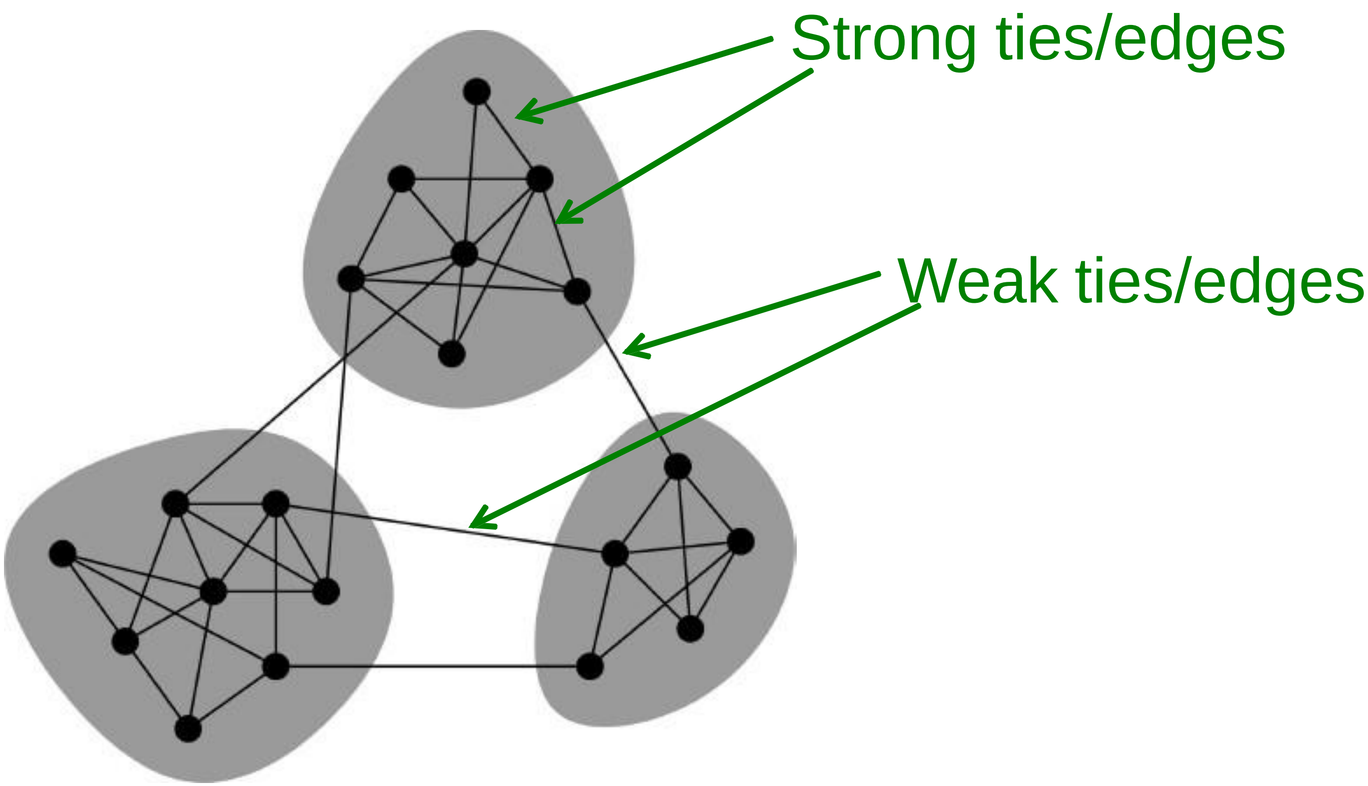
Low disconnects the network sooner



Conceptual picture of network structure

Conceptual Picture of Networks

- Granovetter's theory leads to the following conceptual picture of networks



Lessons from Granovetter's and Onnela's Work

- In real-world, the shape of network (or graph) is **NOT** homogeneous!
  - Homogeneous: all nodes have equal #edges

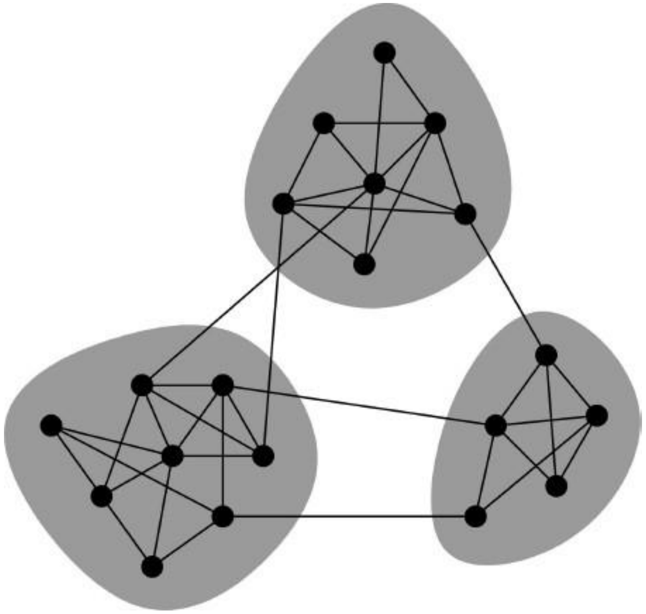
- Heterogeneity (opposite to homogeneity) means:
  - Some edges are skewed to a few nodes
  - Local clusters (=communities) exist

-> It is valuable to analyze network community!

Computing Communities: Overview

Network Communities

- Sets of nodes with lots of internal connections and few external ones (to the rest of the network).

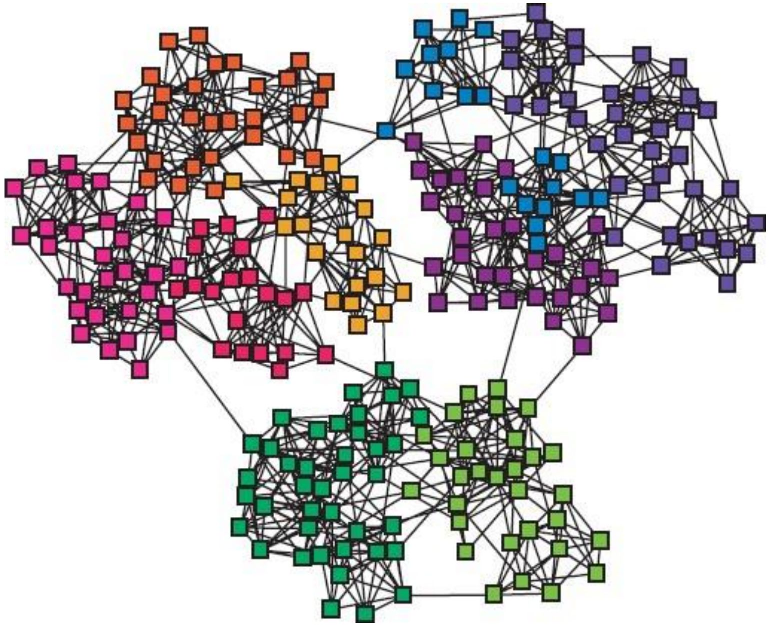


Communities, clusters, groups, modules

Finding Network Communities

- How do we **automatically** find such **densely connected groups** of nodes?

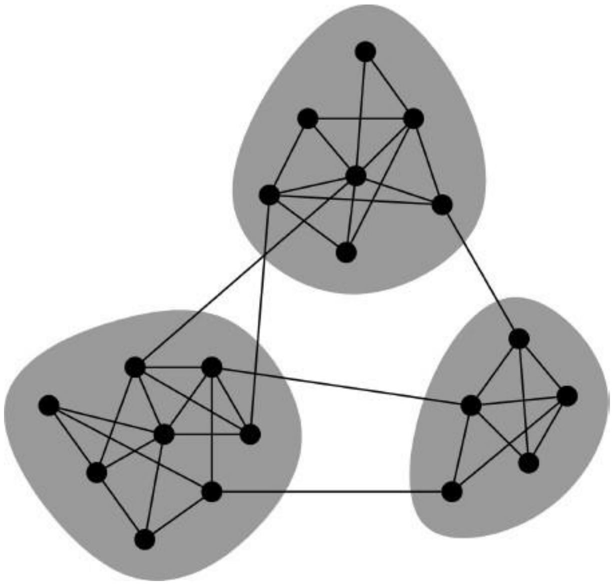
- Ideally such automatically detected clusters would then correspond to real groups



Communities, clusters, groups, modules

Modularity and Network Communities

- Communities: sets of tightly connected nodes

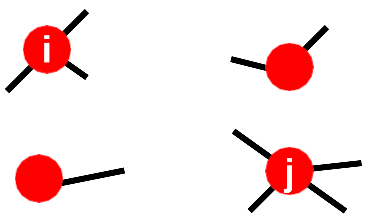


- Define: Modularity $\boldsymbol{Q}$
  - A measure of how well a network is partitioned into communities
  - Given a partitioning of the network into groups disjoint $\boldsymbol{s} \in \boldsymbol{S}$:

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

Null Network Configuration Model

- Given real G on n nodes and m edges, construct **rewired network G'**
  - **Same** degree distribution but **uniformly 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: $k_i \cdot \frac{k_j}{2m} = \frac{k_i k_j}{2m}$
    - The expected number of edges in (multigraph) G':

$$= \frac{1}{2} \sum_{i \in N} \sum_{j \in N} \frac{k_i k_j}{2m} = \frac{1}{2} \cdot \frac{1}{2m} \sum_{i \in N} k_i \left(\sum_{j \in N} k_j \right)$$

$$= \frac{1}{4m} 2m \cdot 2m = \boxed{m}$$ **Same!**

Note:

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

Modularity

- Modularity of partitioning S of graph G:

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

$$Q(G,S) = \underbrace{\frac{1}{2m}}_{\text{Normalizing const.: } -1 \le Q \le 1} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} \left(A_{ij} - \frac{k_i k_j}{2m} \right)$$

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

- Modularity values take range **[−1,1]**
  - It is **positive** if the number of edges within groups **exceeds the expected number**
  - **Q greater than 0.3-0.7 means significant community structure**

Modularity (Cont'd)

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

Equivalently modularity can be written as:

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

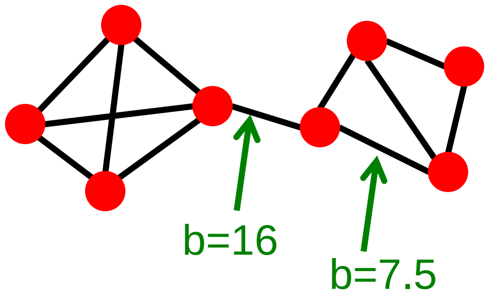
- A_{ij} represents the edge weight between nodes i and j;
- k_i and k_j are the sum of the weights of the edges attached to nodes i and j, respectively;
- $2m$ is the sum of all of the edge weights in the graph;
- c_i and c_j are the communities of the nodes; and
- δ is an indicator function $\delta(c_i, c_j) = 1\ if\ c_i = c_j\ else\ 0$

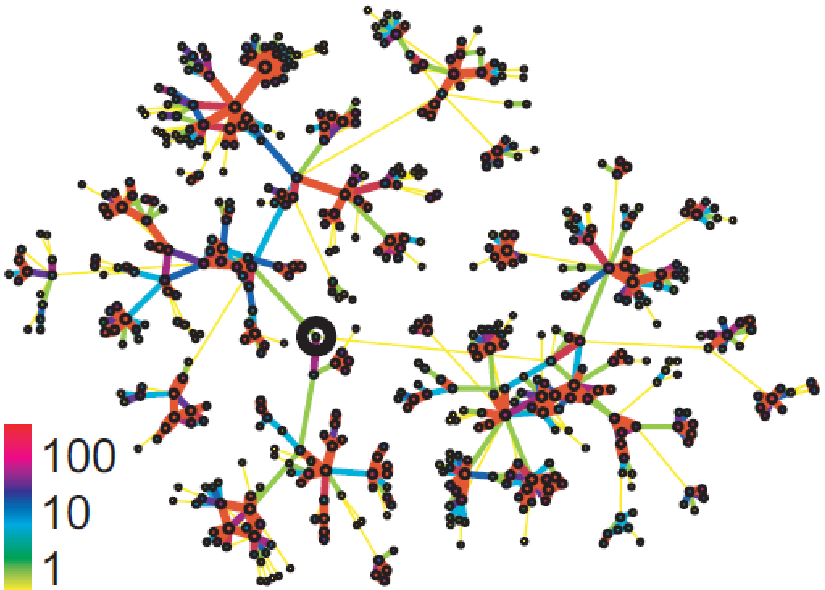
Idea: We can identify communities by maximizing modularity

Computing Communities: Newman Method

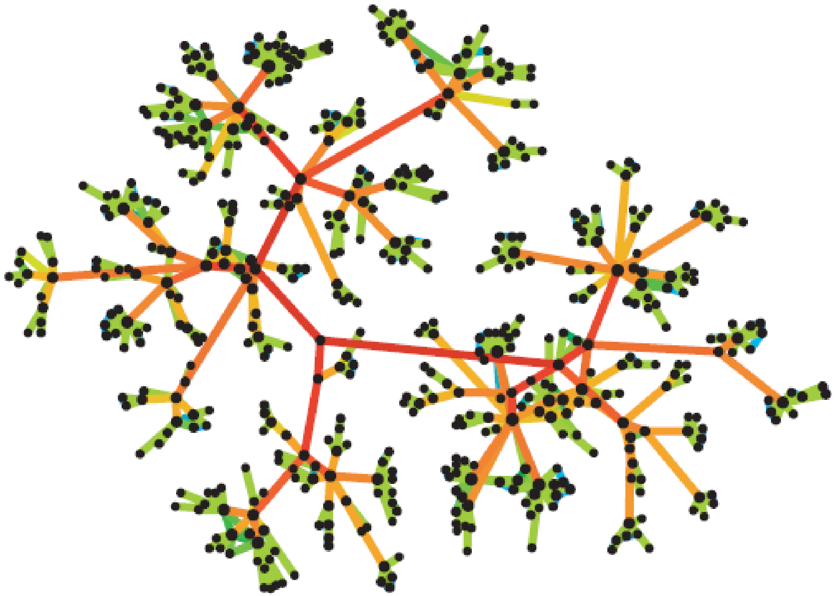
Newman Method: Intuition

- Considering **Edge Betweenness!**
  - #shortest paths passing over the edge





Edge strengths (call volume) in a real network



Edge betweenness in a real network

Newman Method: An Overview

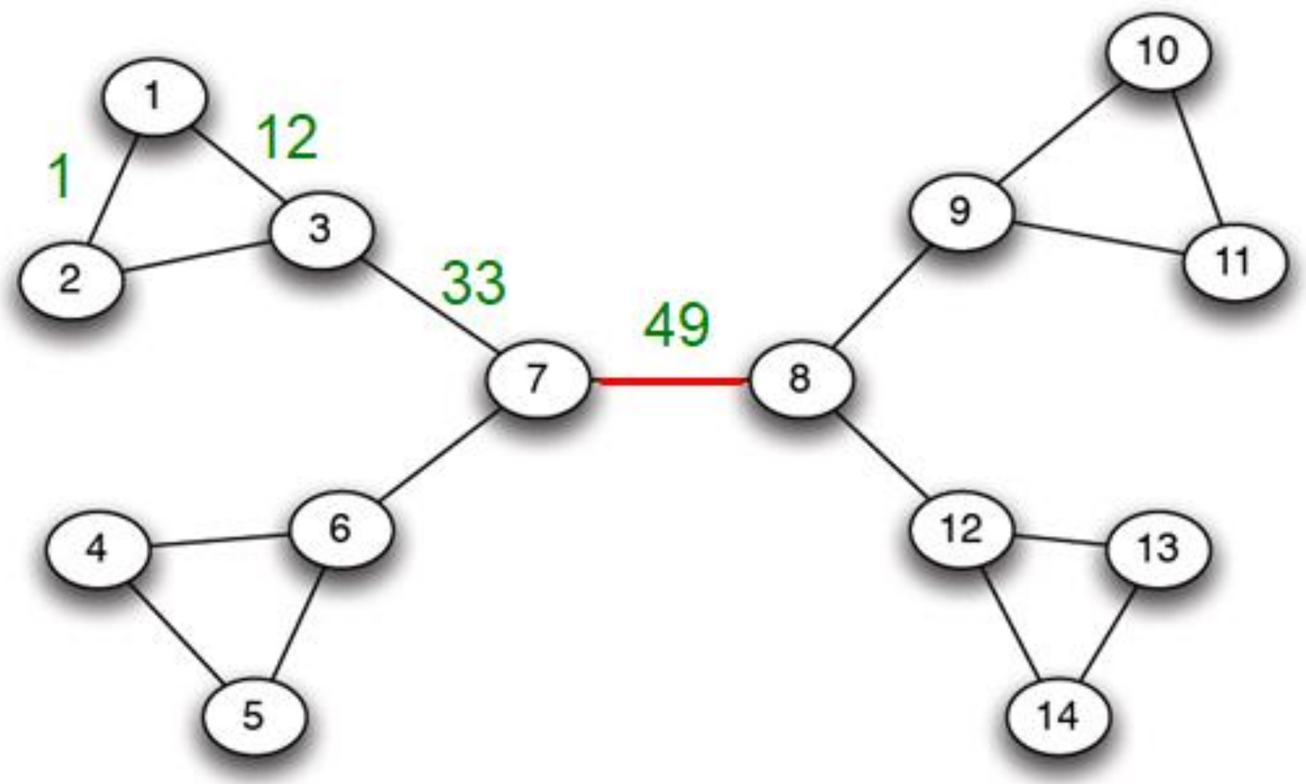
- **Divisive hierarchical clustering** based on the notion of **edge betweenness**

- Works on undirected unweighted networks

- Steps:
  1. Calculate betweenness of edges
  2. Remove edges with highest betweenness
  3. Connected components are communities
  4. Gives a hierarchical decomposition of the network
  5. Repeat 1-5

  → Find the best clusters based on threshold of modularity

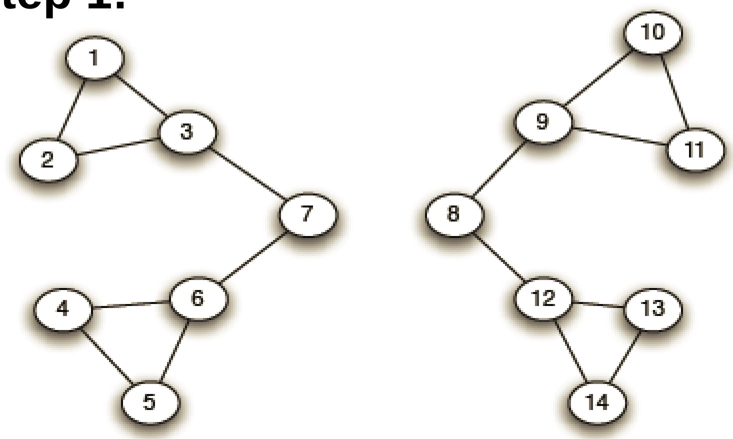
Newman Method: Illustration



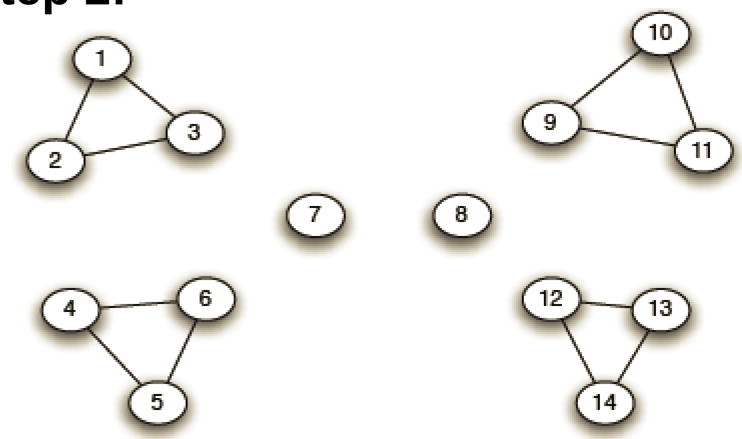
Need to re-compute
betweenness at every step

Newman Method: Illustration (Cont'd)

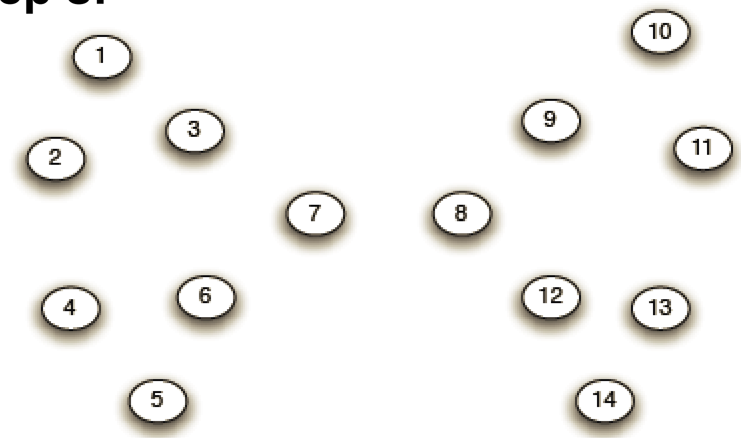
Step 1:



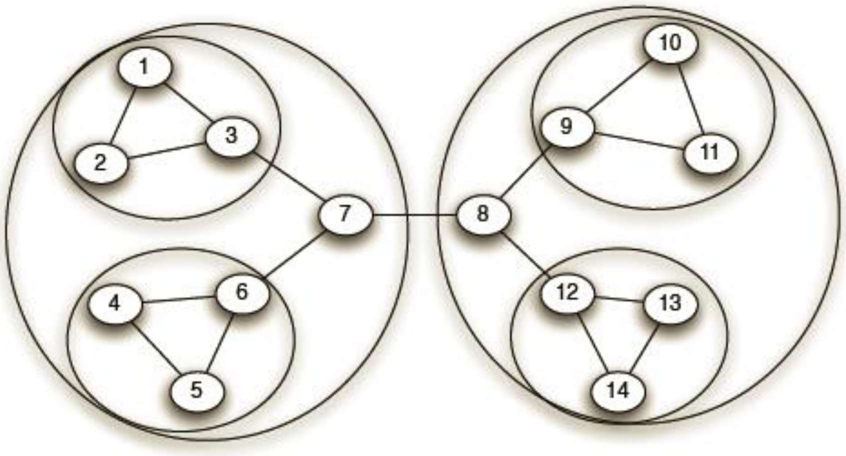
Step 2:



Step 3:

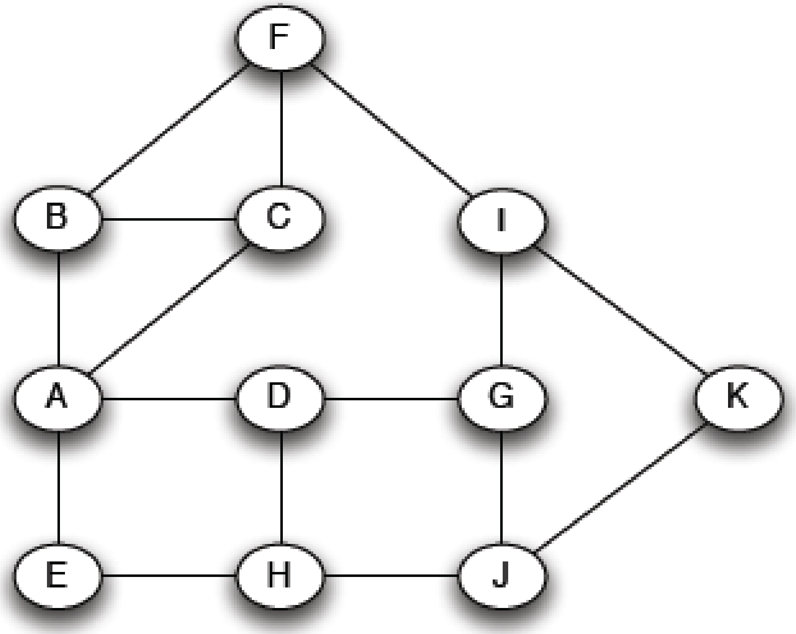


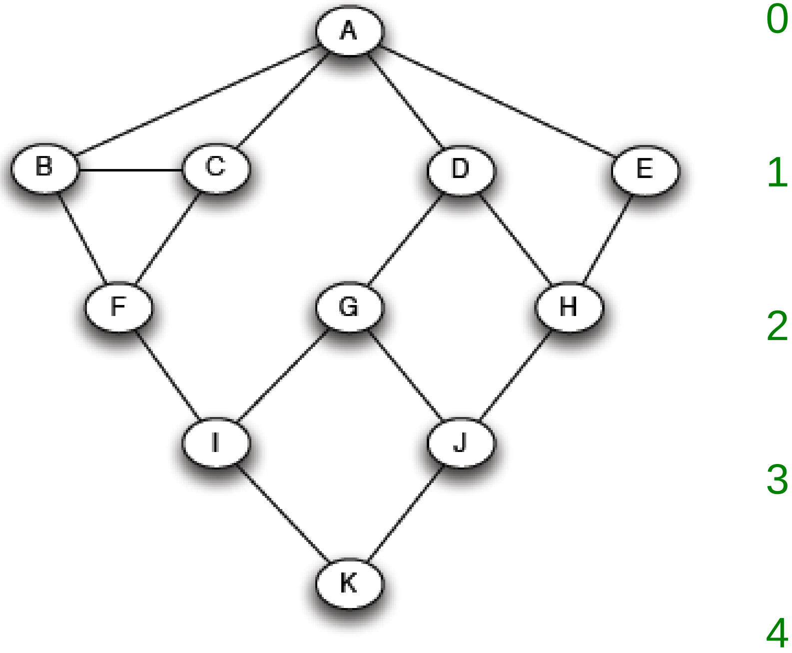
Hierarchical network decomposition:



How to Compute Edge Betweenness? (1/4)

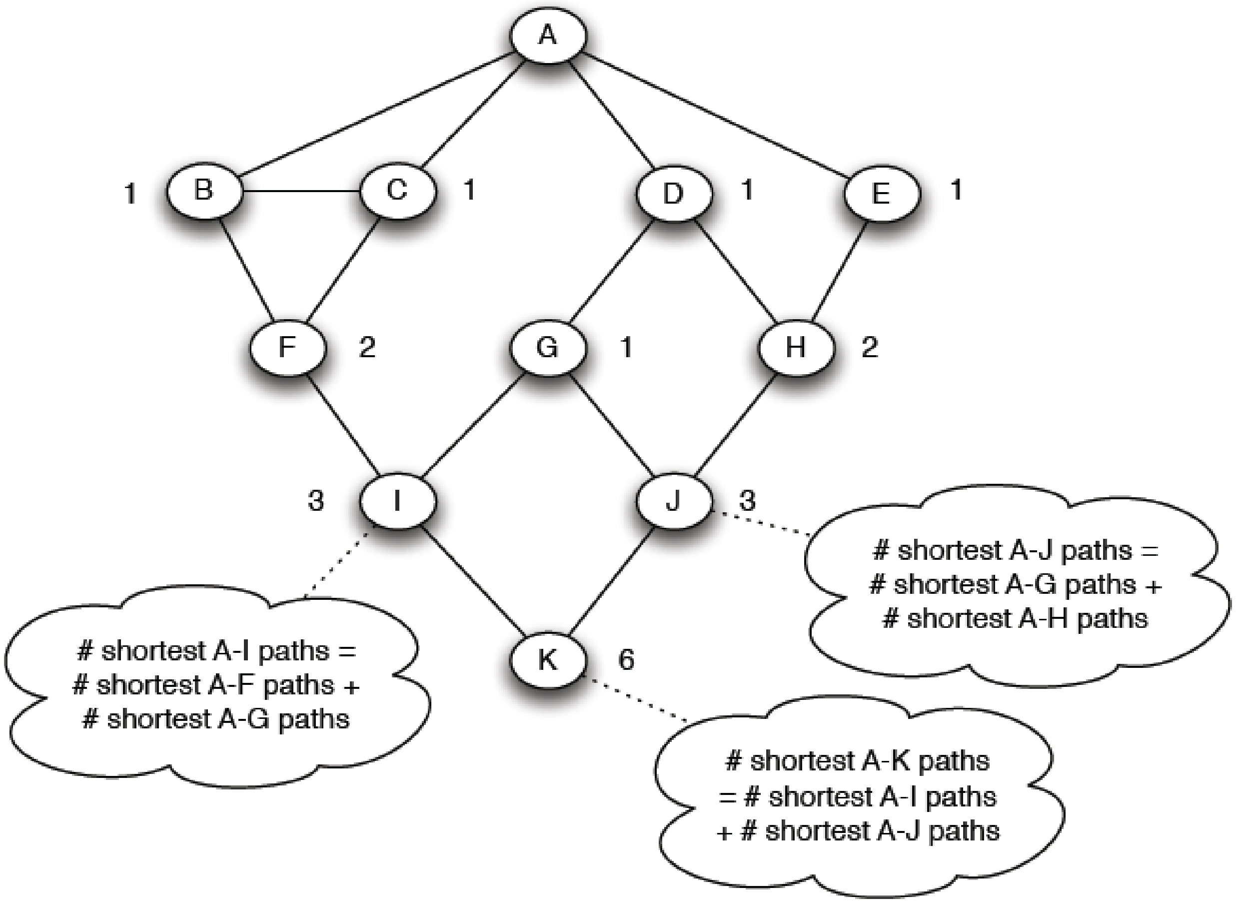
1. Starting from a randomly selected node (e.g., A), conduct breath first search (BFS)





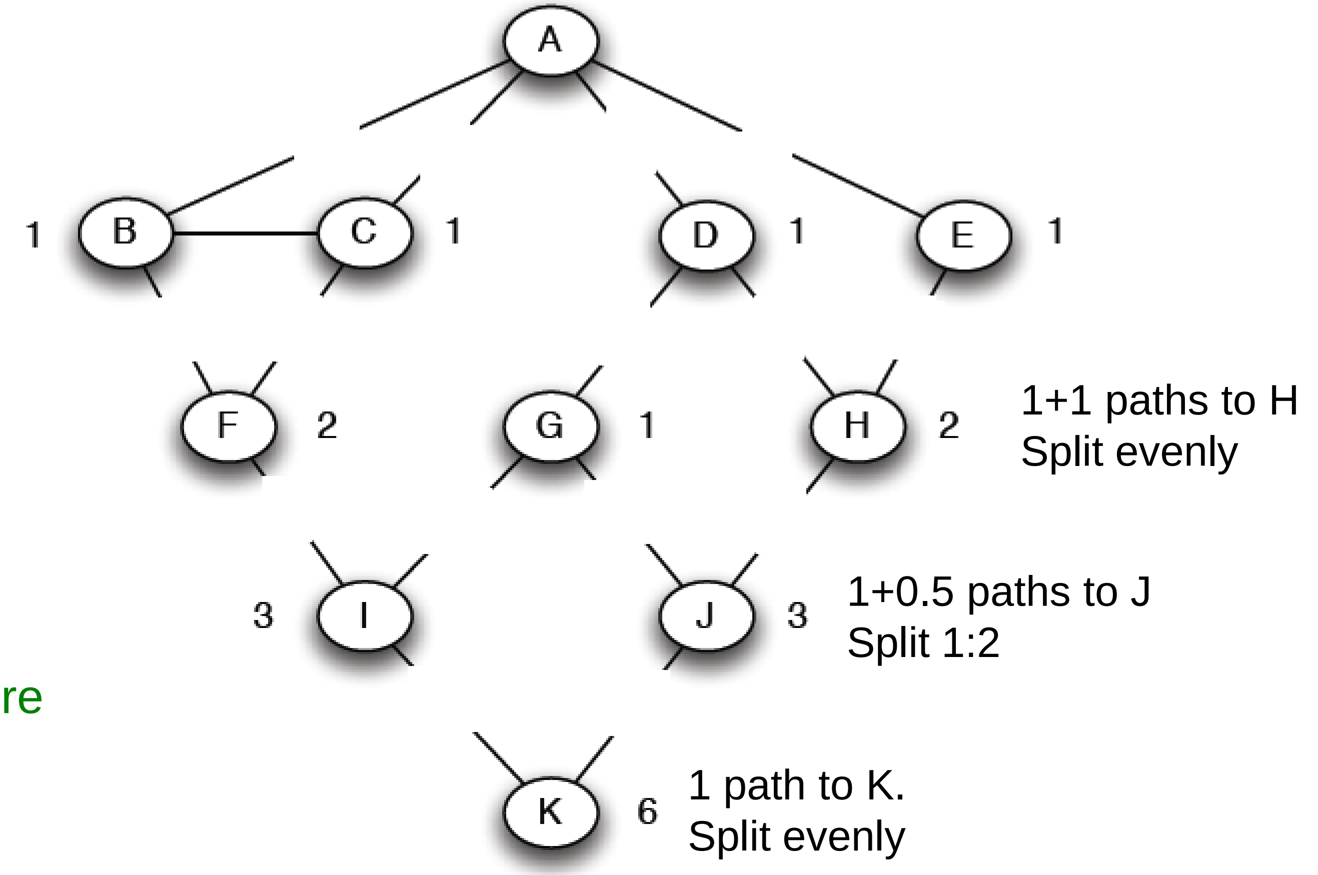
How to Compute Edge Betweenness? (2/4)

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



How to Compute Edge Betweenness? (3/4)

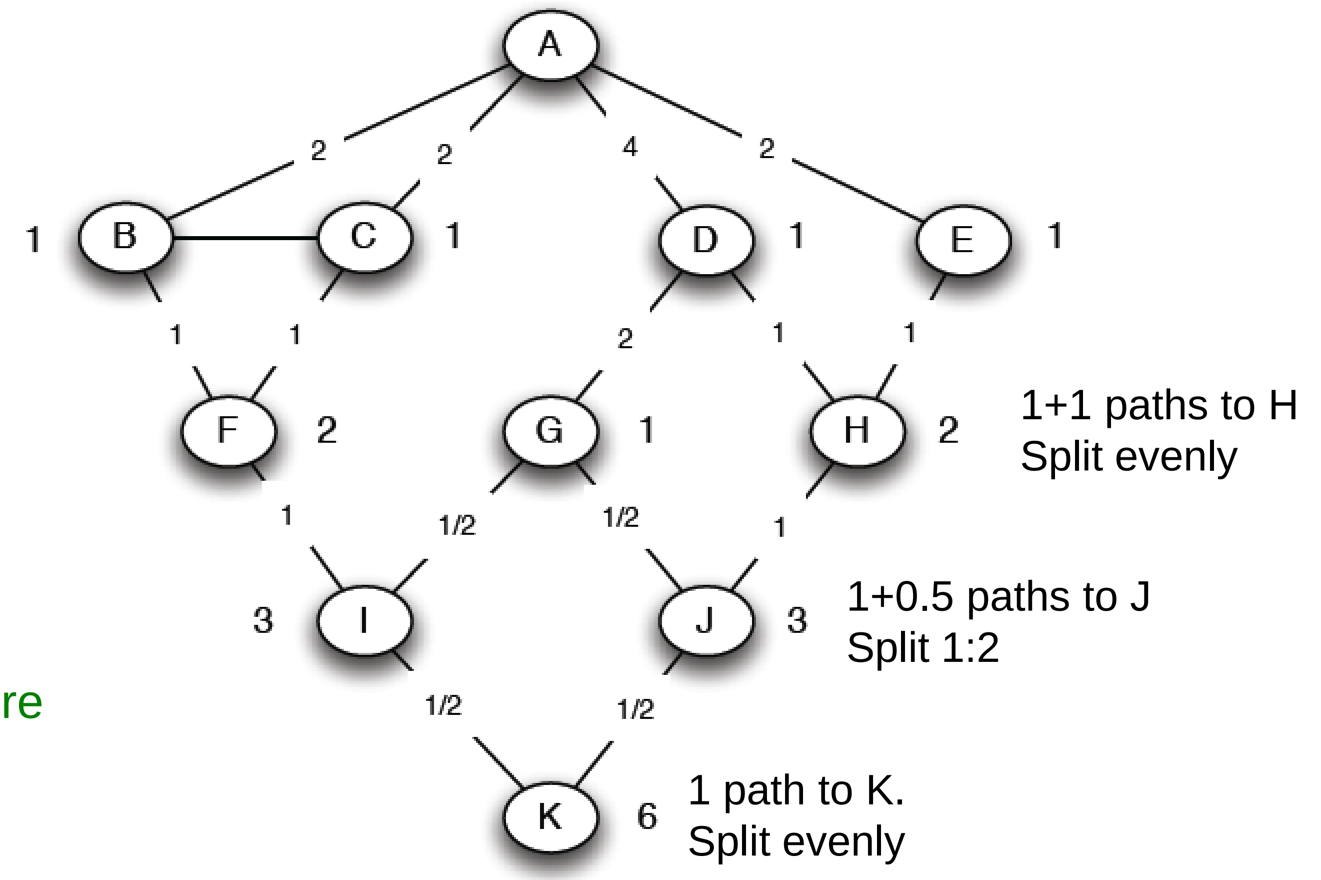
- Compute betweenness by working up the tree from the **lowest layers** of the tree
  - If there are multiple paths, count them **fractionally**



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

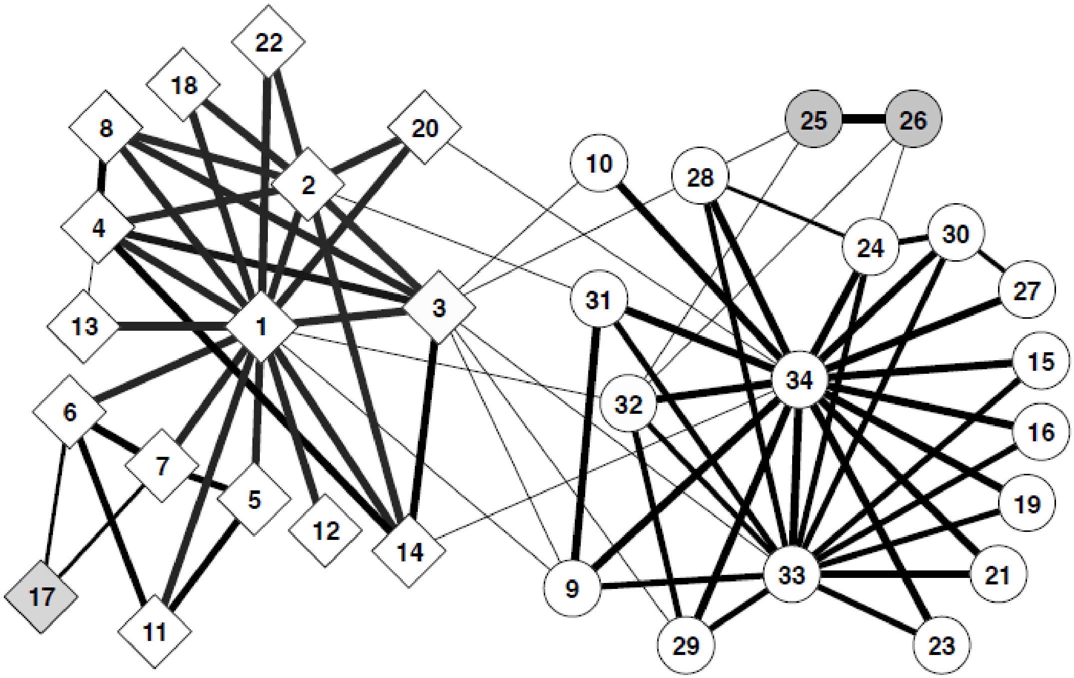
How to Compute Edge Betweenness? (4/4)

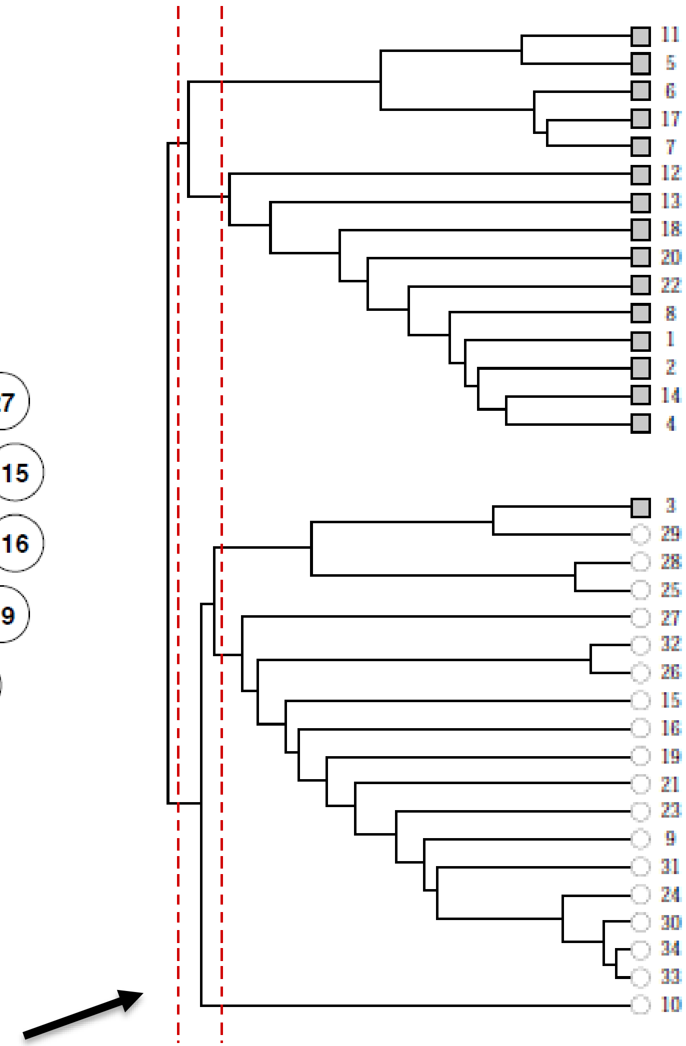
- Compute betweenness by working up the tree from the **lowest layers** of the tree
  - If there are multiple paths, count them **fractionally**



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

Results: Hierarchical network decomposition

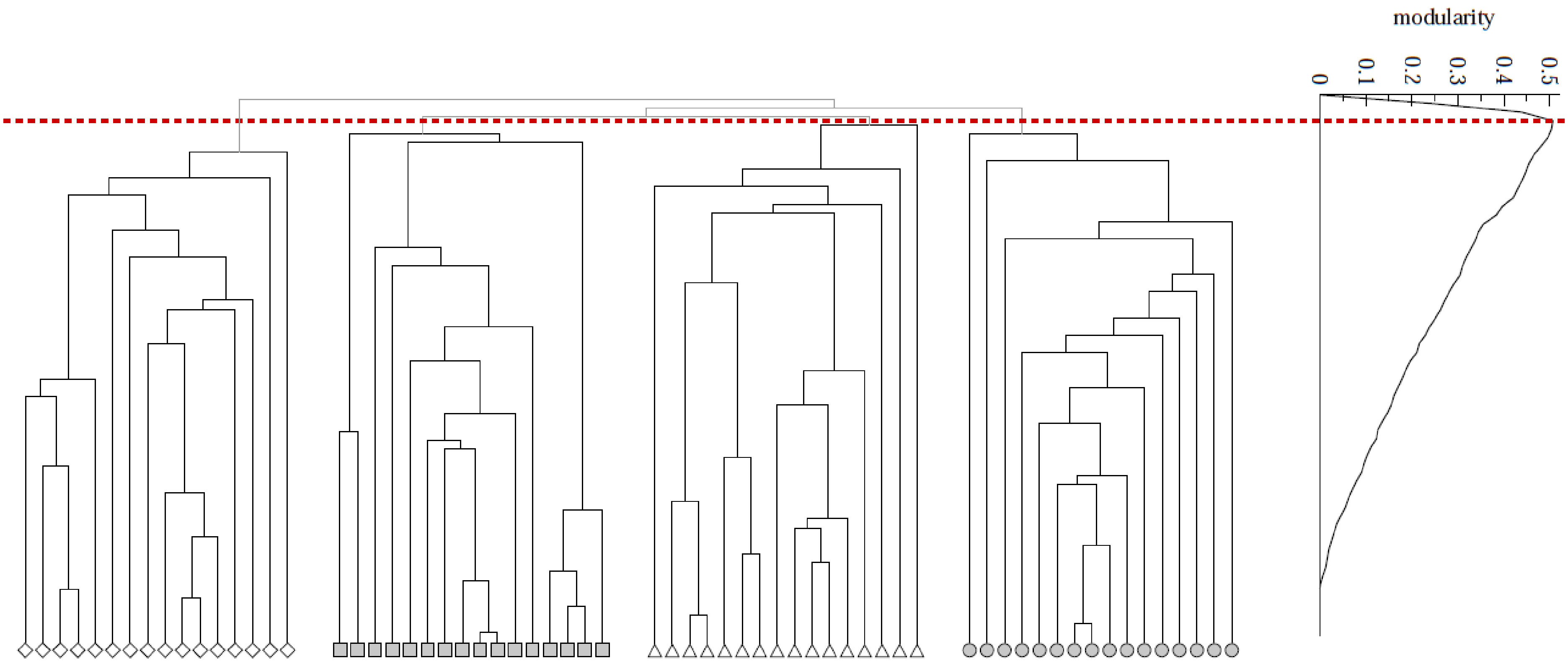




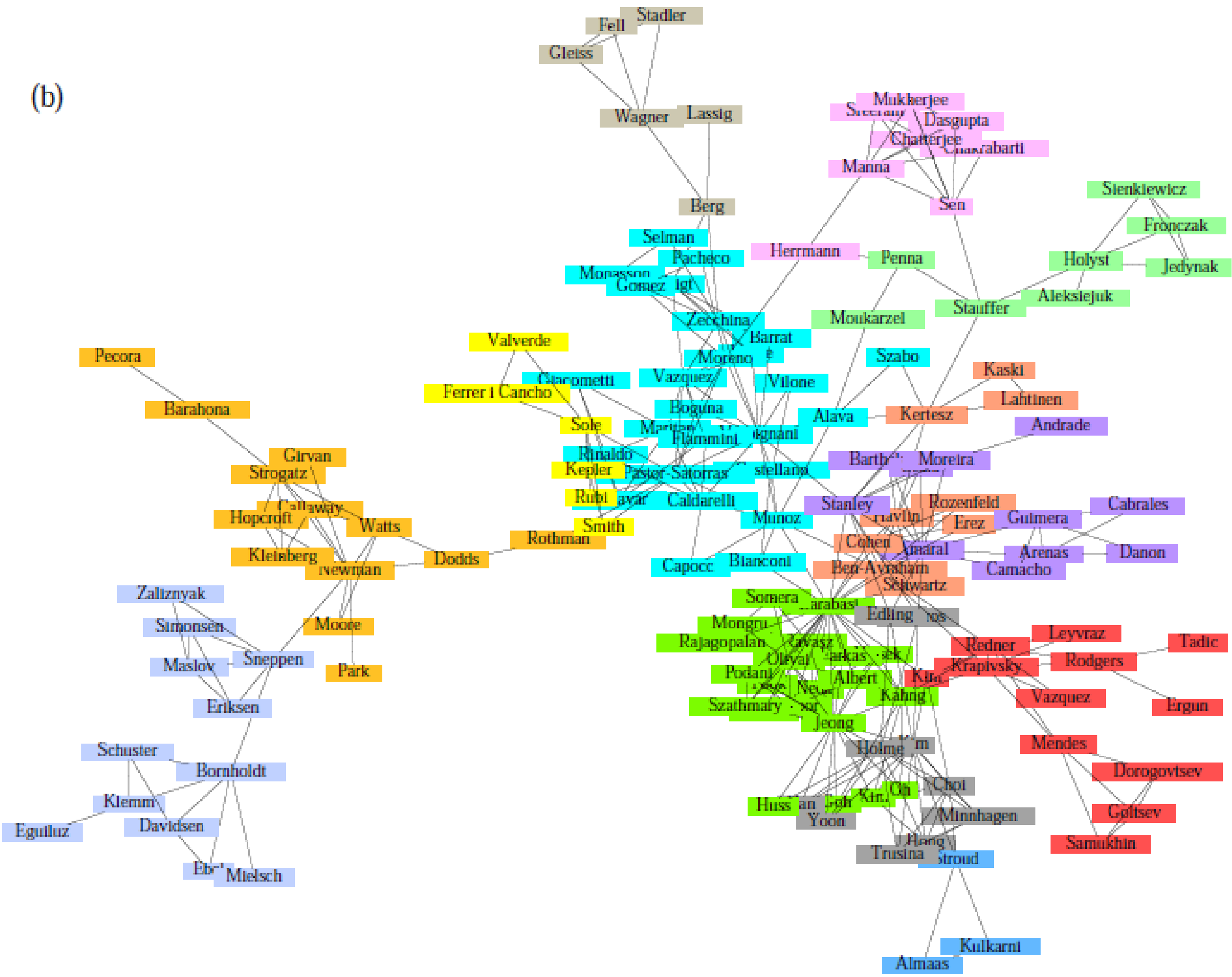
How to find this line? (i.e., What #communities are the best?)

Finally Deciding Clusters

- Compute **modularity** for each steps, then decide clusters with the **highest modularity**!



Newman Method: An Example Graph



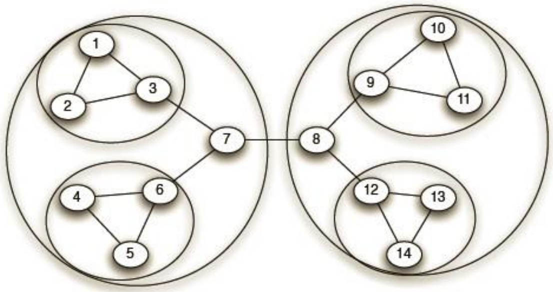
Communities in physics collaborations

Computing Communities: Louvain Algorithm

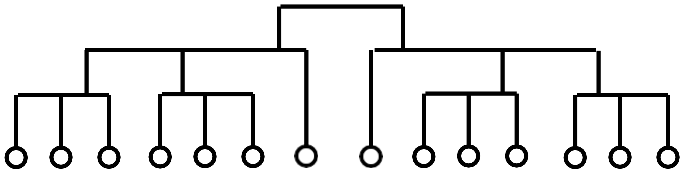
Louvain Algorithm

- **Greedy** algorithm for community detection
  - $O(n \log n)$ run time
- Supports weighted graphs
- Provides hierarchical communities
- Widely used to study large networks because:
  - Fast
  - Rapid convergence
  - High modularity output (i.e., "better communities")

Network and communities:



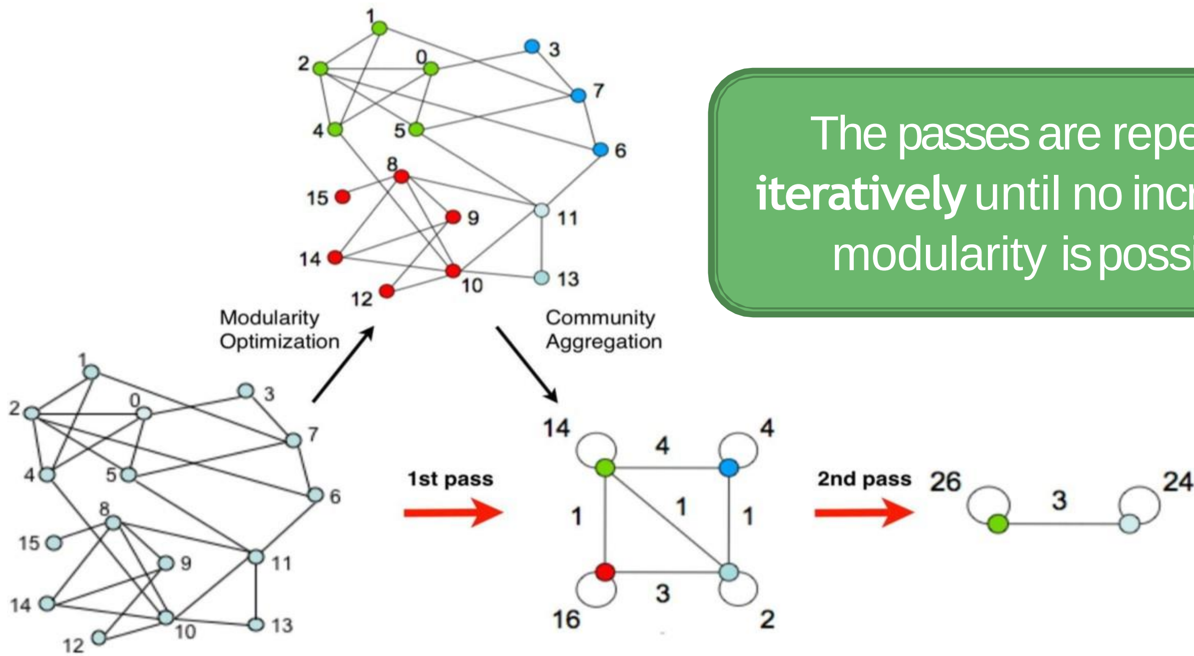
Dendrogram:



"Fast unfolding of communities in large networks" Blondel et al. (2008)

Louvain Algorithm: At High Level

- Louvain algorithm **greedily maximizes** modularity
- Each pass is made of 2 phases:
  - **Phase 1**: Modularity is optimized by allowing only local changes to node-communities memberships
  - **Phase 2**: The identified communities are aggregated into super-nodes to build a new network
  - Goto Phase 1



The passes are repeated **iteratively** until no increase of modularity is possible.

Louvain Algorithm: 1st Phase (Partitioning)

- Put each node in a graph into a distinct community (one node per community)
- For each node i, the algorithm performs two calculations:
  - Compute the modularity delta (ΔQ) when putting node i into the community of some neighbor j
  - Move i to a community of node j that yields the largest gain in ΔQ
- Phase 1 runs until no movement yields a gain

This first phase stops when a local maxima of the modularity is attained, i.e., when no individual node move can improve the modularity.

Note that the output of the algorithm **depends on the order in which the nodes are considered.** Research indicates that the ordering of the nodes does **NOT** have a significant influence on the overall modularity that is obtained.

Louvain Algorithm: Modularity Gain

- What is $\boldsymbol{\Delta Q}$ if we move node $\boldsymbol{i}$ to community $\boldsymbol{C}$?

$$\Delta Q(i \rightarrow C) = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

  - where:

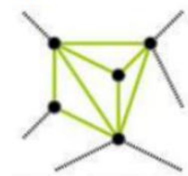
    Σ_{in}… sum of link weights between nodes in C

    Σ_{tot}… sum of all link weights of nodes in C

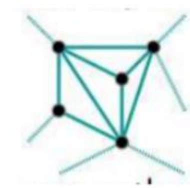
    $k_{i,in}$… sum of link weights between node i and C

    k_i… sum of all link weights (i.e., degree) of node i

Σ_{in}:



Σ_{tot}:

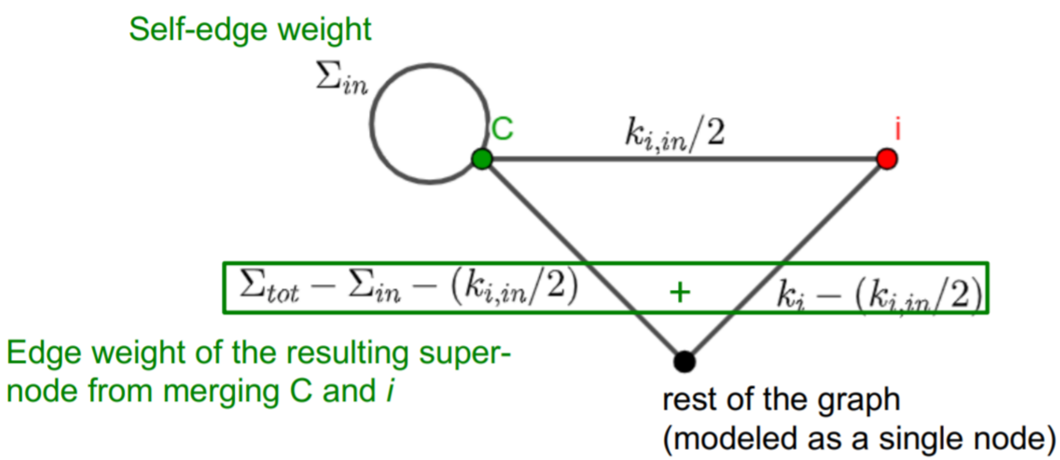


More in detail

Modularity contribution **after merging** node *i*

Modularity contribution **before merging** node *i*

$$\Delta Q(i \rightarrow C) = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m}\right)^2\right] - \left[\underbrace{\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2}_{\text{Modularity of C}} - \underbrace{\left(\frac{k_i}{2m}\right)^2}_{\text{Modularity of } i}\right]$$



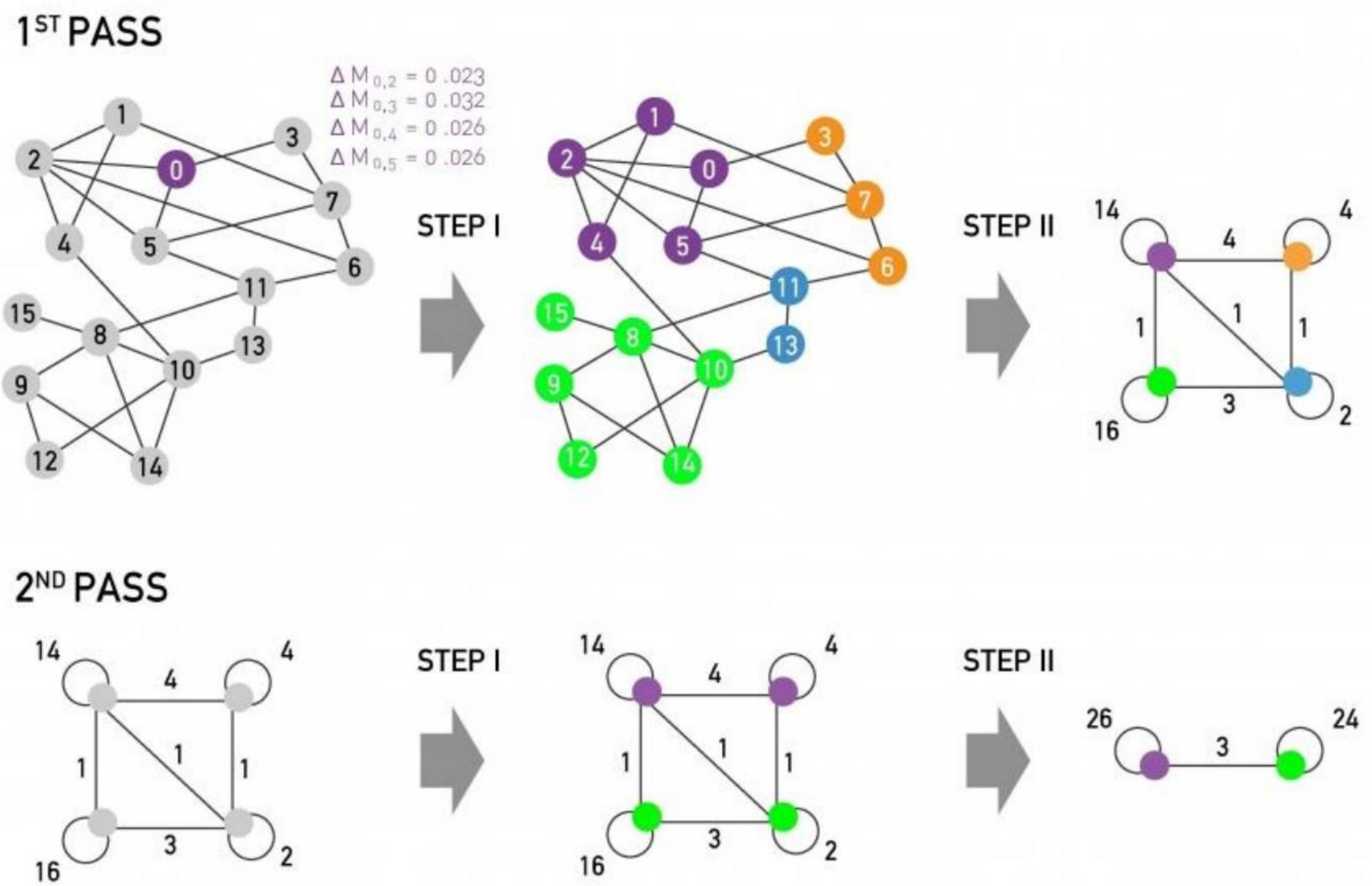
By applying the Modularity definition:

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

Louvain: 2nd Phase (Reconstructing)

- The communities obtained in the first phase are **contracted into super-nodes**, and the network is created accordingly:
  - Super-nodes are connected if there is at least one edge between the nodes of the corresponding communities
  - The weight of the edge between the two super-nodes is the sum of the weights from all edges between their corresponding communities
- Phase 1 is then run on the super-node network

Louvain Algorithm



Summary: Network Communities

- Modularity:
  - Overall quality of the partitioning of a graph into communities
  - Used to determine the number of communities
- Newman method based on edge betweenness
- Louvain modularity maximization:
  - Greedy strategy
  - Great performance, scales to large networks
- There are many other methods!
  - Considering time complexity, graph types, …

The more important thing is
WHAT the communities indicate (mean) in your model!

Thank you!

Instructor: Daejin Choi (djchoi@inu.ac.kr)