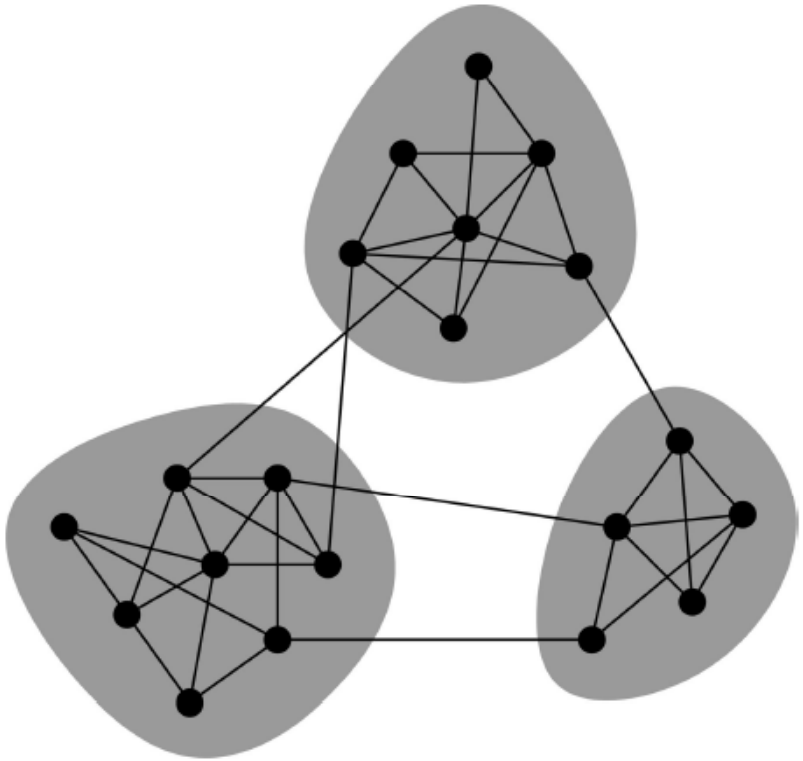
Network Community Detection:

-- Girvan-Newman
-- Modularity optimization
-- Trawling

CS224W: Social and Information Network Analysis
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

Network communities

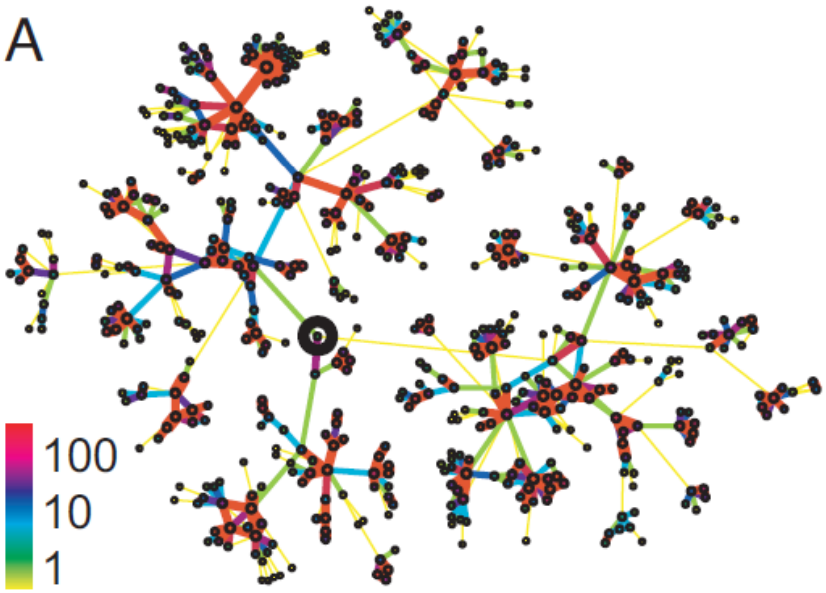
- Networks of **tightly connected groups**
- **Network communities:**
  - Sets of nodes with **lots** of connections **inside** and **few** to **outside** (the rest of the network)



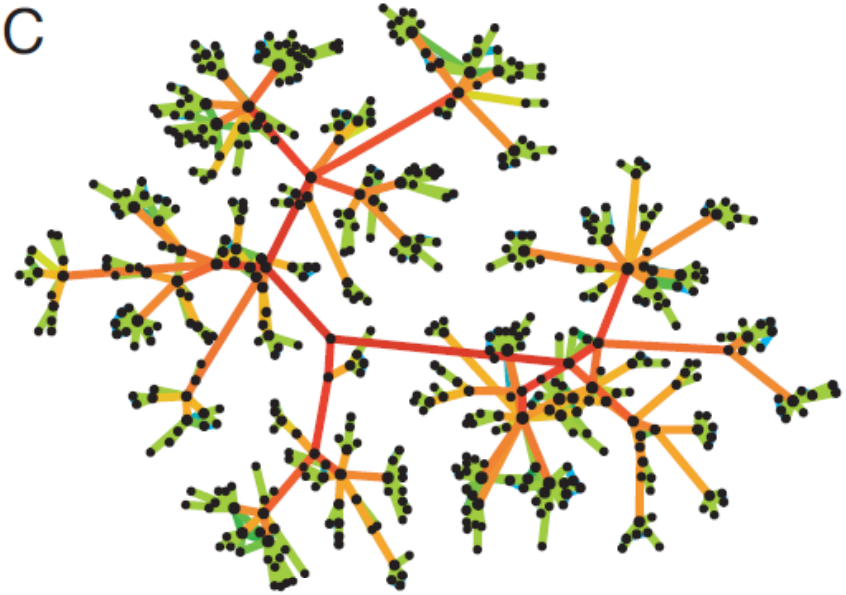
Communities, clusters, groups, modules

[Onnela et al. '07]

Girvan-Newman: Weak ties



Edge strengths (call volume) in real network

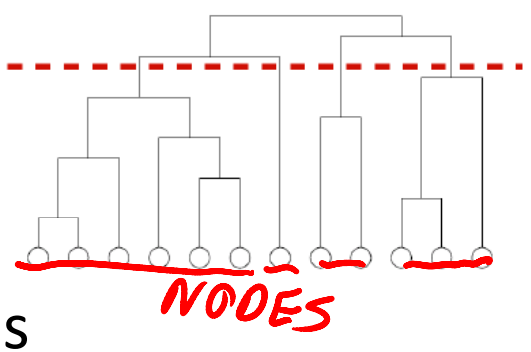


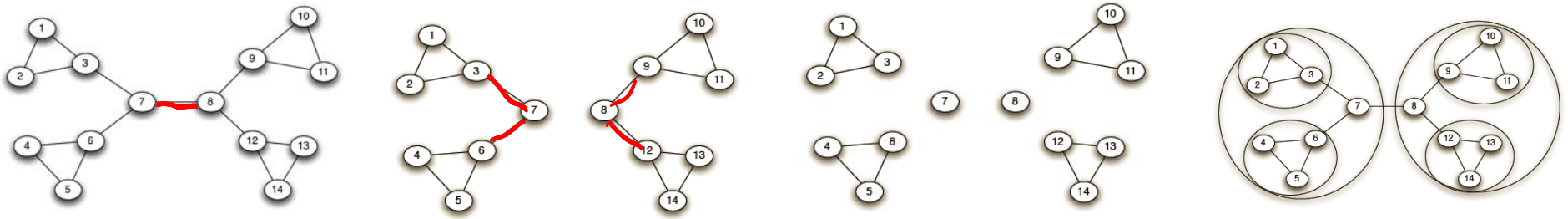
Edge betweenness in real network

[Girvan-Newman PNAS '02]

Method1: Girvan-Newman

- Divisive hierarchical clustering based on edge betweenness:
  - Number of shortest paths passing through the edge
- **Girvan-Newman Algorithm:**
  - 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
- Example:

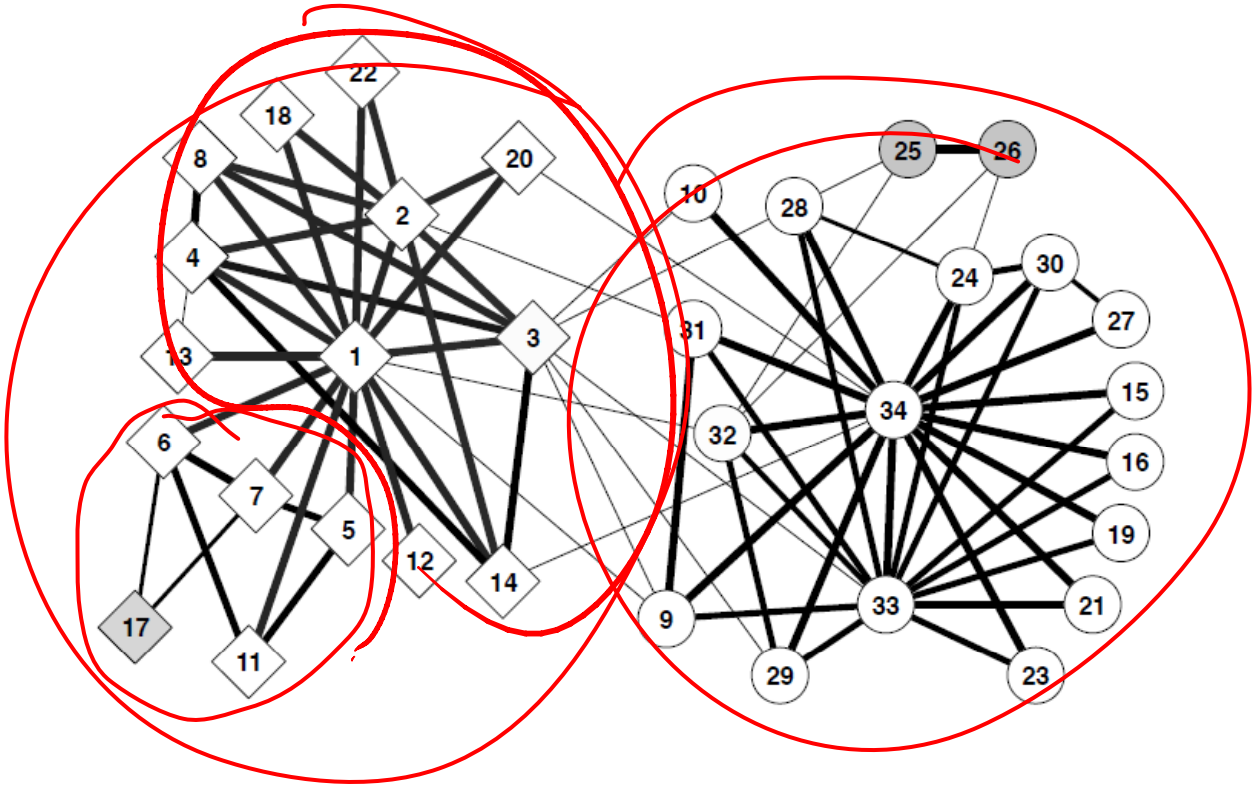


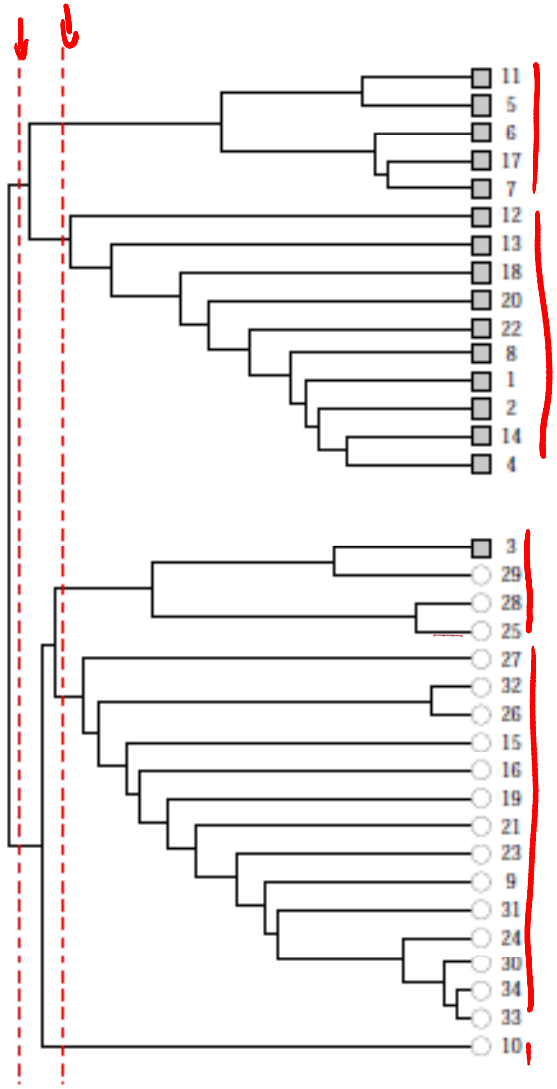


[Newman-Girvan PhysRevE '03]

Girvan-Newman: Example

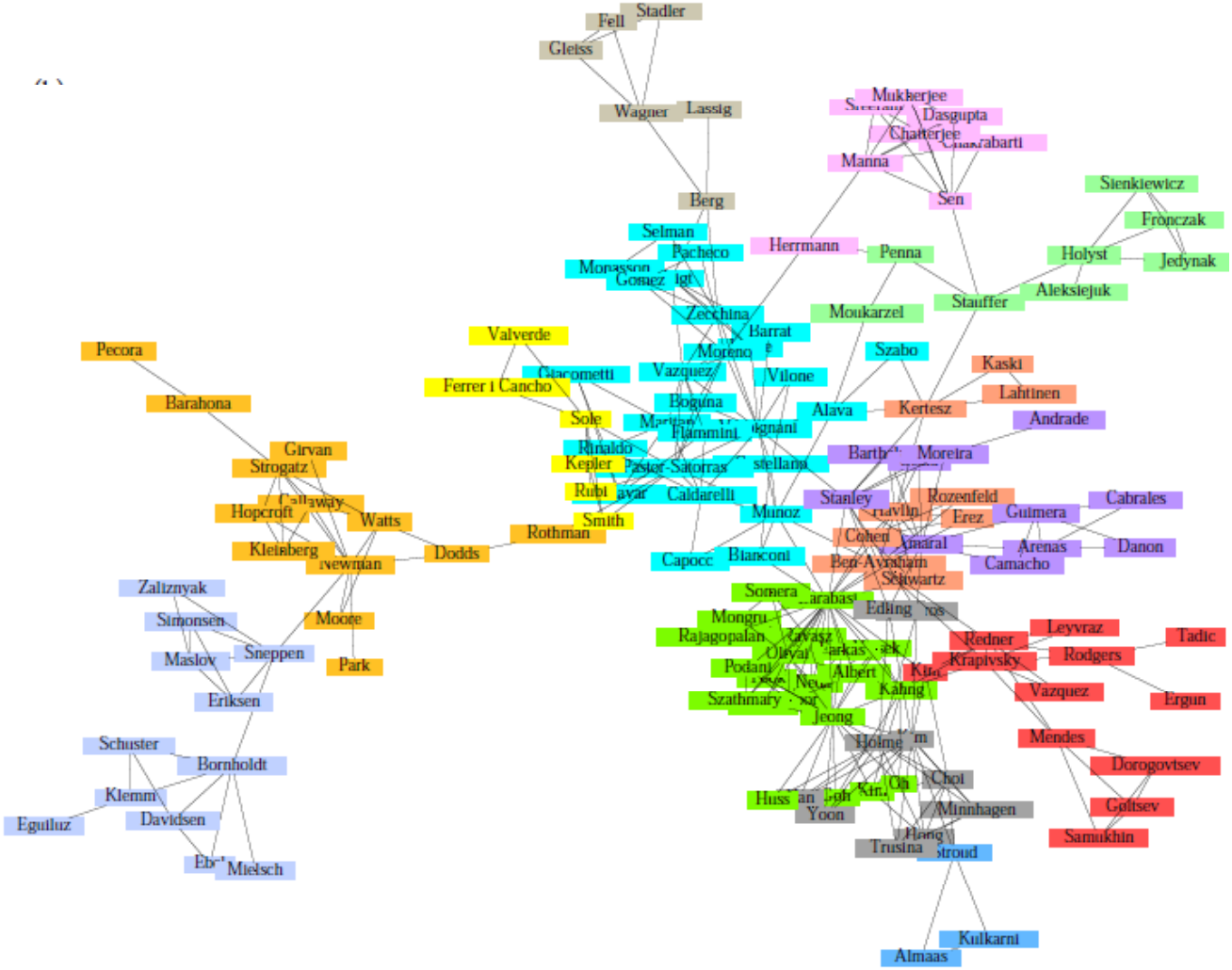
- Zachary's Karate club:
  hierarchical decomposition





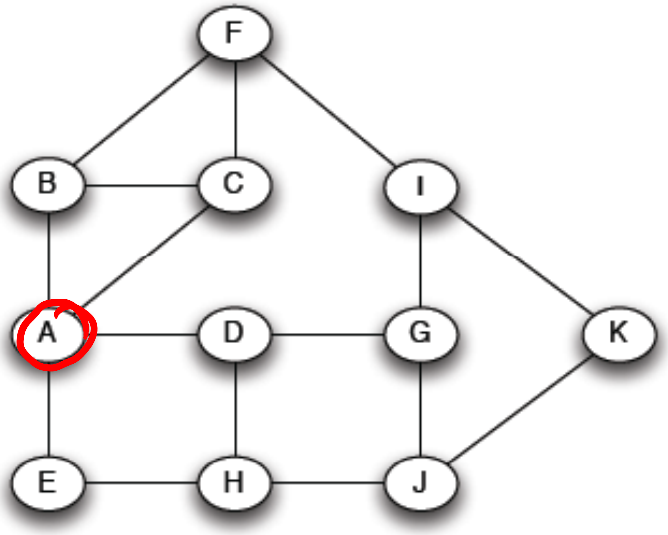
[Newman-Girvan PhysRevE '03]

Girvan-Newman: Example



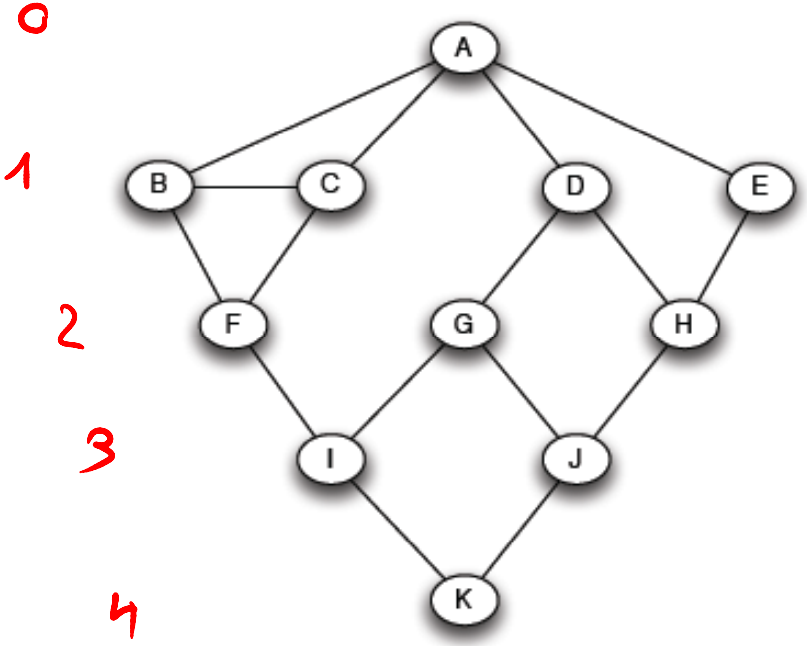
Communities in physics collaborations

How to compute betweenness?



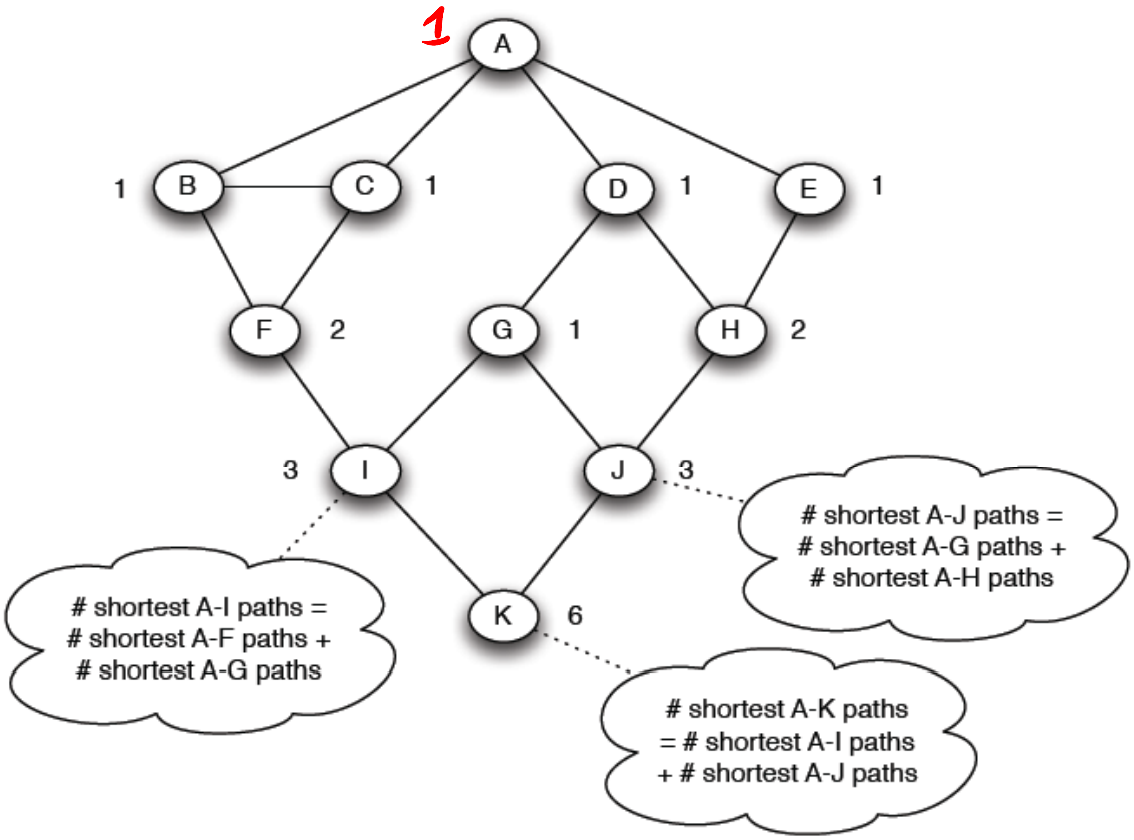
- Want to compute betweenness of paths starting at node A

- Breath first search starting from A:



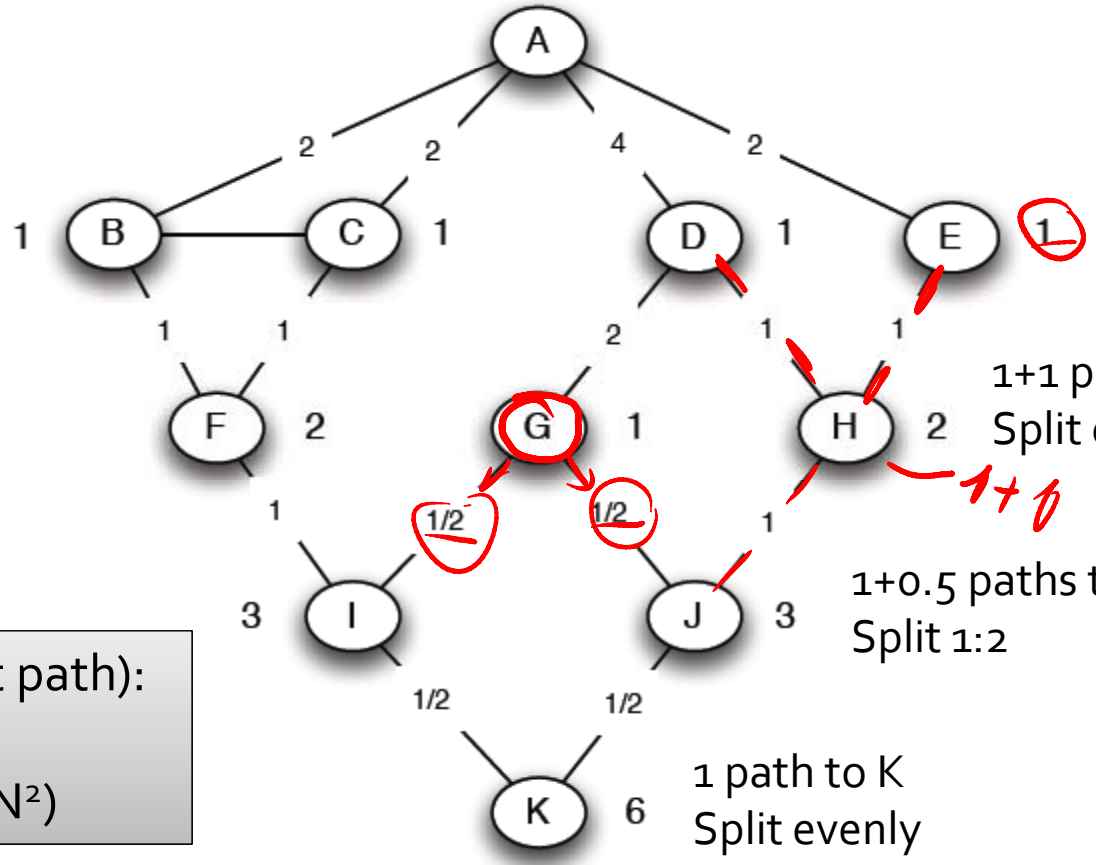
How to compute betweenness?

- 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



- Repeat the BFS procedure for each node of the network
- Add edge scores

- Runtime (all pairs shortest path):
  -- Weighted graphs: $O(N^3)$
  -- Unweighted graphs: $O(N^2)$

How to select the number of clusters?

Define **modularity** to be

Q = (number of edges within groups) – (expected number within groups)

Actual number of edges between i and j is

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge } (i,j), \\ 0 & \text{otherwise.} \end{cases}$$

Expected number of edges between *i* and *j* is

$$\text{Expected number} = \frac{k_i k_j}{2m}.$$

m...number of edges

Modularity: Definition

- Q = (number of edges within groups) – (expected number within groups)
- Then:

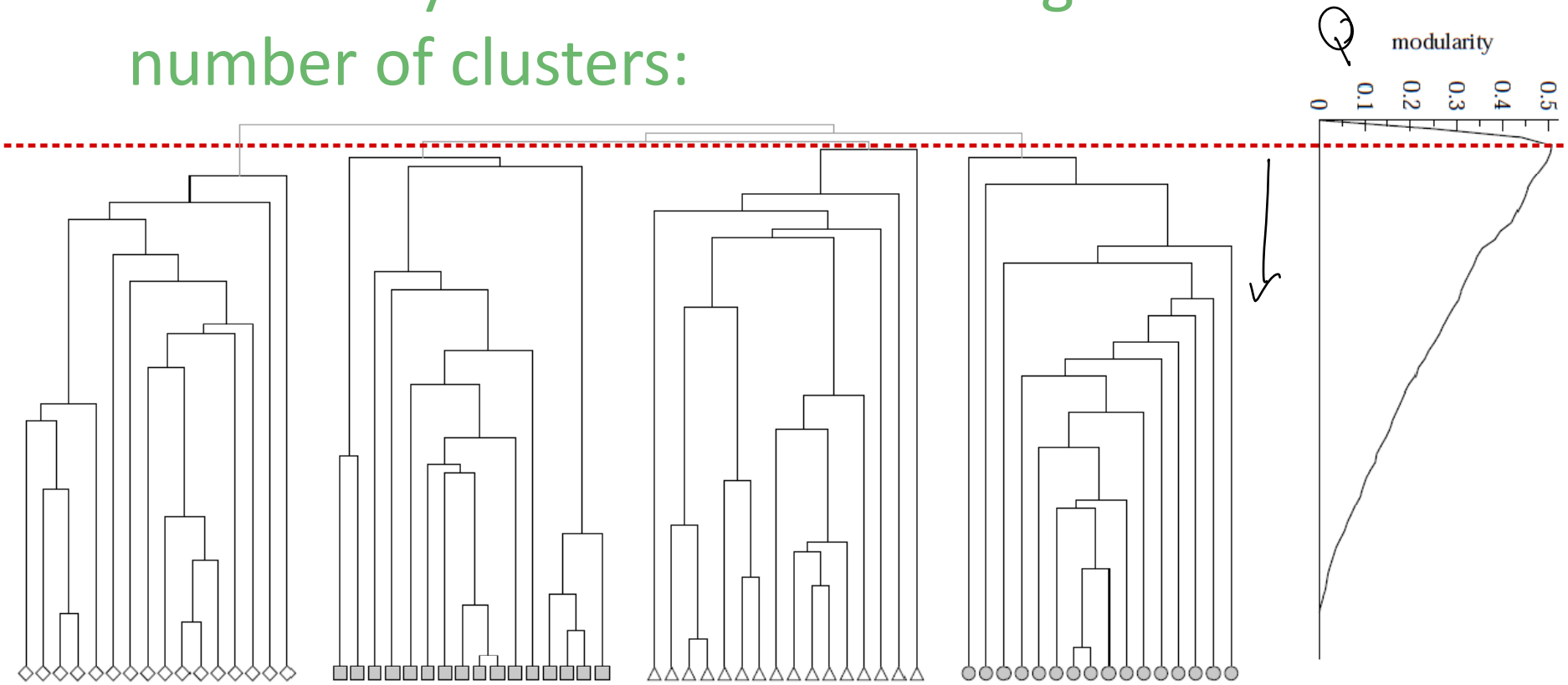
$$Q = \frac{1}{4m}\left[\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j)\right]$$

m … number of edges
A_{ij} … 1 if (i,j) is edge, else 0
k_i … degree of node i
c_i … group id of node i
$\delta(a, b)$ … 1 if a=b, else 0

- Modularity lies in the range [−1,1]
  - It is positive if the number of edges within groups exceeds the expected number
  - 0.3<Q<0.7 means significant community structure

Modularity: Number of clusters

- Modularity is useful for selecting the number of clusters:



Why not optimize modularity directly?

Method2: Modularity optimization

- Consider splitting the graph in two communities
- Modularity Q is: $\sum_{i,j \text{ in same group}} A_{ij} - \frac{k_i k_j}{2m}$
- Or we can write in matrix form as

$$Q = \frac{1}{4m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) s_i s_j = \frac{1}{4m} \mathbf{s}^T \mathbf{B} \mathbf{s}$$

  - **s** … vector of group memberships s_i={+1, -1}
  - *B* … *modularity matrix*

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

Note: each row (column) of B sums to 0

Fast Modularity Optimization

- Task: Find $s \in \{-1,+1\}^n$ that maximizes Q
- Rewrite Q in terms of eigenvalues β_i of B

$$Q = s^{\mathrm{T}}\left[\sum_{i=1}^{n} u_i \beta_i u_i^{\mathrm{T}}\right] s = \sum_i s^{\mathrm{T}} u_i \beta_i u_i^{\mathrm{T}} s = \sum_{i=1}^{n} \left(s^{\mathrm{T}} u_i\right)^2 \beta_i$$

$\beta_1 > \beta_2 > \beta_3 > \ldots$

- To maximize Q, easiest way is to make $\mathbf{s} = \lambda u_1$
  - Assigns all weight in the sum to β_1 (largest eigval)
    - (all other $\mathbf{s}^{\mathrm{T}}\mathbf{u}_i$ terms zero because of orthonormality)
    - Unfortunately, elements of **s** must be ± 1
    - In general, finding optimal **s** is NP-hard

Finding a splitting strategy

$$Q = \sum_{i=1}^{n} \left(s^{\mathrm{T}} u_i\right)^2 \beta_i \approx \left(\sum_{i=1}^{n} s_i u_{1i}\right)^2 \beta_1$$

- Heuristic: try to maximize only the β_1 term

$$s_i = \begin{cases} +1 & \text{if } i\text{th element of } \mathbf{u}_1 \geq 0, \\ -1 & \text{if } i\text{th element of } \mathbf{u}_1 < 0. \end{cases}$$

- Similar in spirit to the spectral partitioning algorithm (we will explore it next time)
- Continue the bisection hierarchically

Fast Modularity Optimization

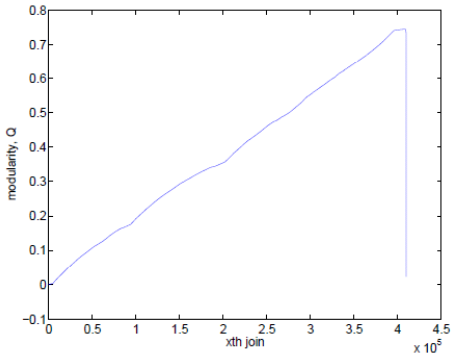
- Fast Modularity Optimization Algorithm:
  - Find leading eigenvector u_1 of modularity matrix B
  - Divide the nodes by the signs of the elements of u_1
  - Repeat hierarchically until:
    - If a proposed split does not cause modularity to increase, declare community indivisible and do not split it
    - If all communities are indivisible, stop
- How to find u_1? Power method!
  - Iterative multiplication, normalization
  - Start with random v, until convergence:

$$v_{k+1} = \frac{Bv_k}{\|Bv_k\|}$$

Even more heuristic approaches

- Also, can combine with other methods:
  - Randomly divide the nodes into two groups
  - Move the node that, if moved, will increase Q the most
  - Repeat for all nodes, with each node only moved once
  - Once complete, find intermediate state with highest Q
  - Start from this state and repeat until Q stops increasing
  - Good results for "fine-tuning" the spectral method

- CNM Algorithm (Clauset-Newman-Moore '04):
  - (1) Separate each vertex solely into n community
  - (2) Calculate ΔQ for all possible community pairs
  - (3) Merge the pair of the largest increase in Q
  - Repeat (2)&(3) until one community remains
  - Cross cut the dendogram where Q is maximum



Comparison to other methods

network	size n	modularity Q GN	CNM	DA	Fast modularity
karate	34	0.401	0.381	0.419	0.419
jazz musicians	198	0.405	0.439	0.445	0.442
metabolic	453	0.403	0.402	0.434	0.435
email	1133	0.532	0.494	0.574	0.572
key signing	10 680	0.816	0.733	0.846	0.855
physicists	27 519	–	0.668	0.679	0.723

GN = Girvan-Newman, $O(n^3)$
CNM = Greedy merging $(n \log^2 n)$
DA = External Optimization $O(n^2 \log^2 n)$

- Issues with modularity:
  - May not find communities with less than $\sqrt{m}$ links
  - NP-hard to optimize exactly [Brandes et al. '07]

Fast Modularity: Example

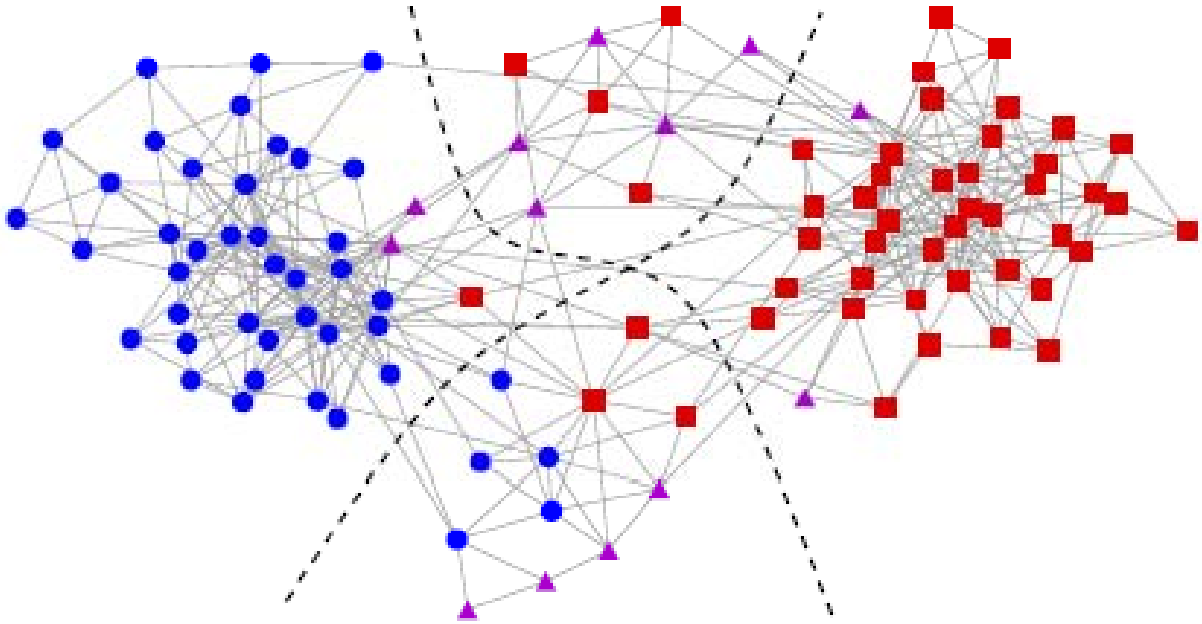


FIG. 3: Krebs' network of books on American politics. Vertices represent books and edges join books frequently purchased by the same readers. Dotted lines divide the four communities found by our algorithm and shapes represent the political alignment of the books: circles (blue) are liberal, squares (red) are conservative, triangles (purple) are centrist or unaligned.

[Kumar et al. '99]

Method3: Trawling

- Searching for small communities in a Web graph
- (1) The signature of a community/discussion in the context of a Web graph

A dense 2-layer graph

Intuition: a bunch of people all talking about the same things

Searching for small communities

- (2) A more well-defined problem:

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

  - Where $K_{s,t} = s$ nodes where each links to the same t other nodes

The Plan: (1), (2) and (3)

- Two points:
  - (1) The signature of a community/discussion
  - (2) Complete bipartite subgraph $K_{s,t}$
    - $K_{s,t}$ = graph on s nodes, each links to the same t other nodes
- Plan:
  - (A) From (2) get back to (1):
    - Via: Any dense enough graph contains a smaller $K_{s,t}$ as a subgraph
  - (B) How do we solve (2) in a giant graph?
    - What similar problems have been solved on a giant non-graph datasets?
    - (3) Frequent itemset enumeration [Agrawal-Srikant '99]

[Agrawa-Srikant '99]

Frequent itemset enumeration

- Marketbasket analysis:
  - What items are bought together in a store?
- Setting:
  - Universe U of n items
  - m subsets of U: $S_1, S_2, \ldots, S_m \subseteq U$
    (S_i is a set of items one person bought)
  - Frequency threshold f
- Goal:
  - Find all subsets T s.t. $T \subseteq S_i$ of $\geq f$ sets S_i
    (items in T were bought together $\geq f$ times)

Frequent itemsets: Example

- Example:
  - Universe of items:
    - U={1,2,3,4,5}
  - Itemsets:
    - S_1={1,3,5}, S_2={2,3,4}, S_3={2,4,5}, S_4={3,4,5}, S_5={1,3,4,5}, S_6={2,3,4,5}
  - Minimum support:
    - f=3
- Algorithm: Build up the lists
  - Insight: for a frequent set of size *k*, all its subsets are also frequent

[Agrawa-Srikant '99]

Example: Apriori algorithm

- U={1,2,3,4,5}, f=3
- S_1={1,3,5}, S_2={2,3,4}, S_3={2,4,5}, S_4={3,4,5}, S_5={1,3,4,5}, S_6={2,3,4,5}

[Agrawa-Srikant '99]

Frequent itemset: Apriori algorithm

- For i = 1,…,k
  - Find all frequent sets of size i by composing sets of size i-1 that differ in 1 element

- Open question:
  - Efficiently find only maximal frequent sets

From Itemsets to bipart graphs $K_{s,t}$

- Claim: (3) (itemsets) solves (2) (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* that occurs in *s* sets S_i
  - Looking for $K_{s,t}$ → set of frequency threshold to *s* and look at layer t – all frequent sets of size *t*.

From $K_{s,t}$ to Communities

- (2) $\Rightarrow$ (1): Informally, every dense enough bipartite graph *G* contains a $K_{s,t}$ subgraph where *s* and *t* depend on size (# of nodes) and density (avg. degree) of *G* [Kovan-Sos-Turan '53]

- Theorem: Let *G=(X,Y,E)*, $|X|=|Y|=n$ with avg. degree:

$$d = s^{1/t} n^{1-1/t} + t$$

then *G* contains $K_{s,t}$ as a subgraph

$K_{s,t}$ and Communities

- For the proof we will need the following fact
  - Recall: $\binom{a}{b} = \frac{a(a-1)\ldots(a-b+1)}{b!}$
    - Let $f(x) = x(x-1)(x-2)\ldots(x-k)$
      Once $x \geq k$, $f(x)$ curves upward (convex)
  - Suppose a setting:
    - $g(y)$ is convex
    - Want to minimize $\sum_i^n g(x_i)$
    - where $\sum_i^n x_i = x$
  - To minimize $\sum_i^n g(x_i)$ make each $x_i = x/n$

Nodes and buckets

- Node *i*, degree d_i:

Potential right-hand sides of $K_{s,t}$ (i.e., all size t subsets of Y_i)

- Put node *i* in buckets for all size *t* subsets of its neighbors

Nodes and buckets

- Notice: As soon as *s* people appear in a bucket we have a $K_{s,t}$
- To how many buckets *i* contributes to?
- What is the total size of all buckets?

Nodes and buckets

- So, total height of all buckets is…

$$\binom{a}{b} = \frac{a(a-1)\ldots(a-b+1)}{b!}$$

And we are done!

- Total height of all buckets:
- How many buckets are there?
- What is the average height of buckets?
- So by pigeonhole principle, there must be a bucket with more than *s* nodes in it.

Summary

- Girvan-Newman:
  - Based on strength of weak ties
  - Remove edge of highest betweenness
- Modularity:
  - Useful to determine the number of clusters
  - Direct approx submodularity optimization
- Trawling (complete bipartite subgraphs):
  - Frequent itemsets and dynamic programming
  - Theorem that complete bipartite subgraphs are embedded in bigger graphs
  - **SCALABLE!!!**