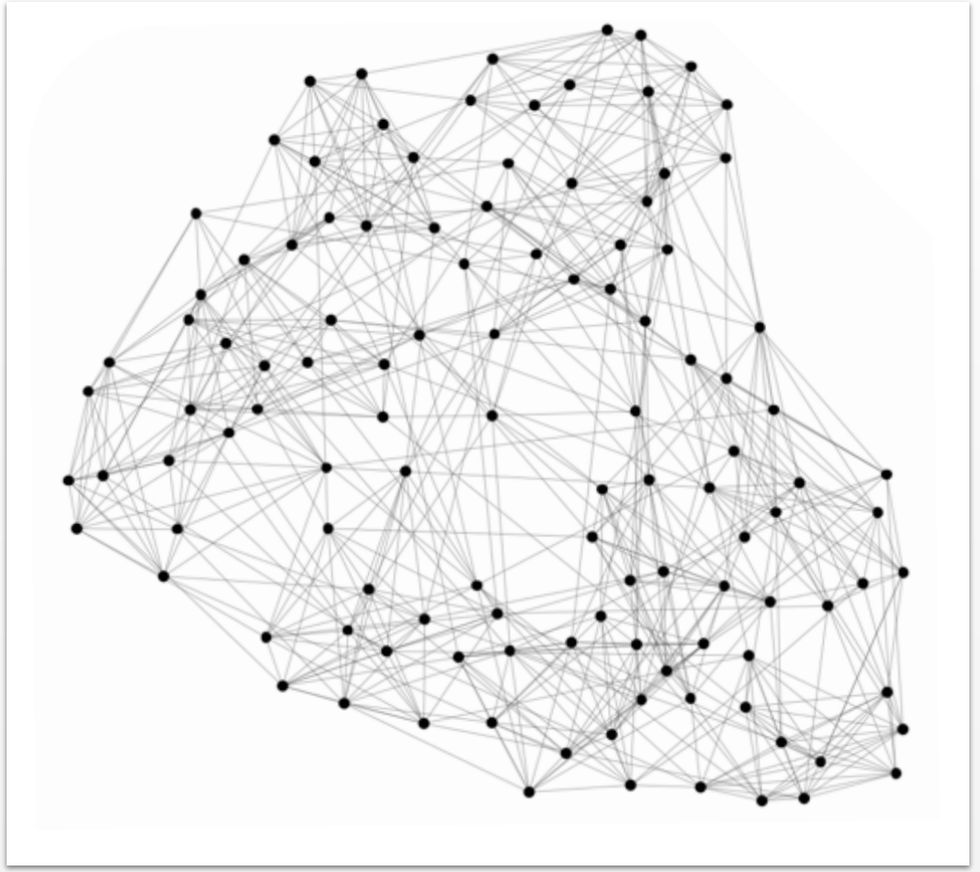
Community Detection

Shantanu Jain

Centrality Measures

Betweenness, Closeness, Degree

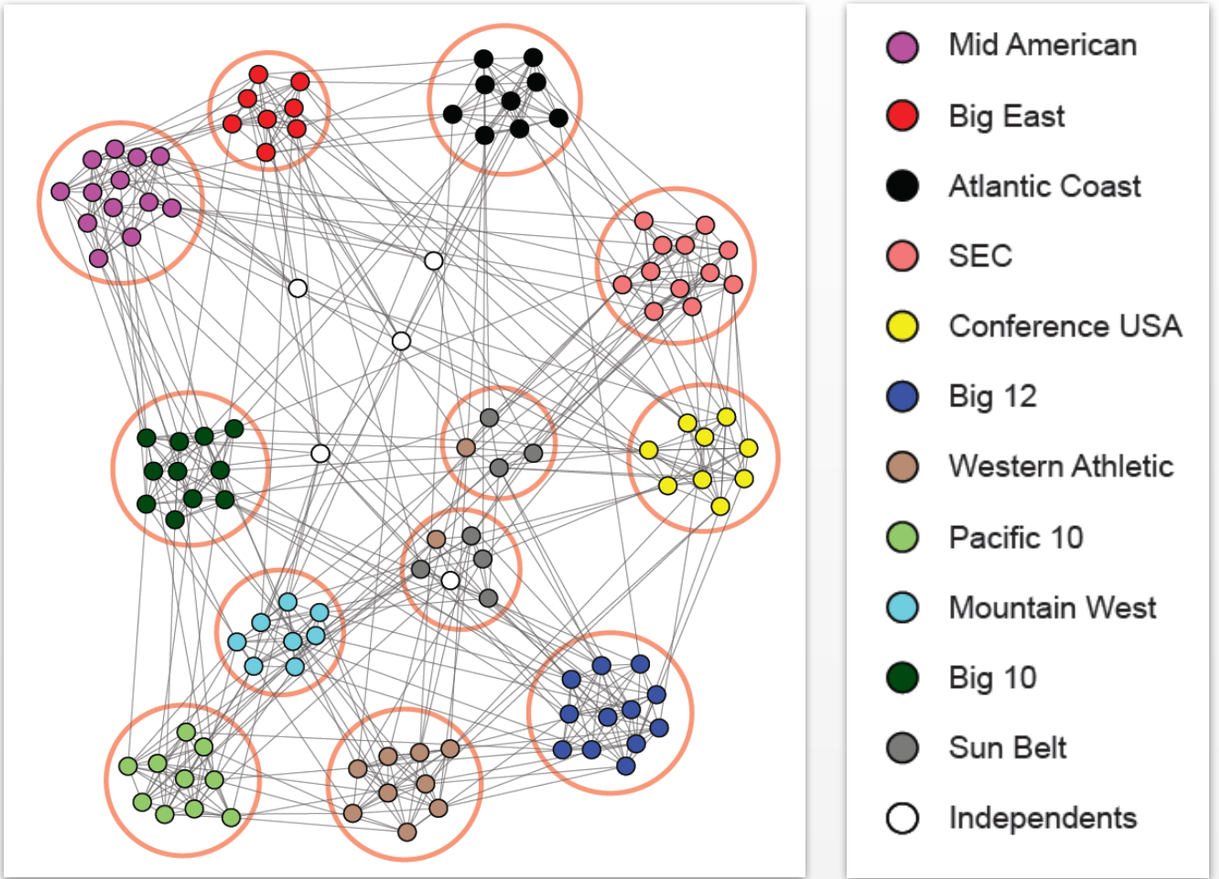
Community Detection



Problem: Can we identify groups of densely connected nodes?

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

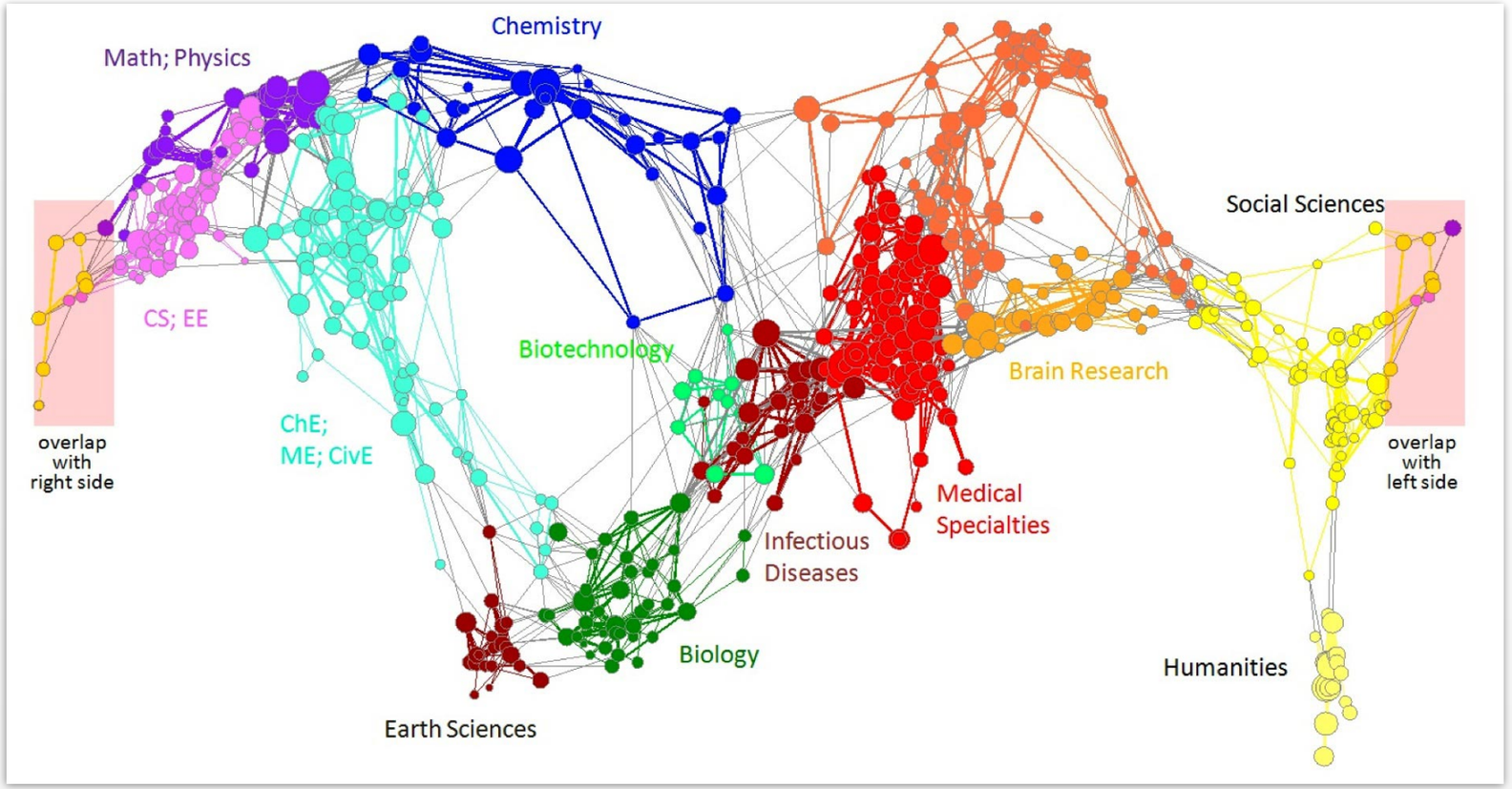
Communities: Football Conferences



Nodes: Football Teams, *Edges:* Matches,
Communities: Conferences

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Communities: Academic Citations

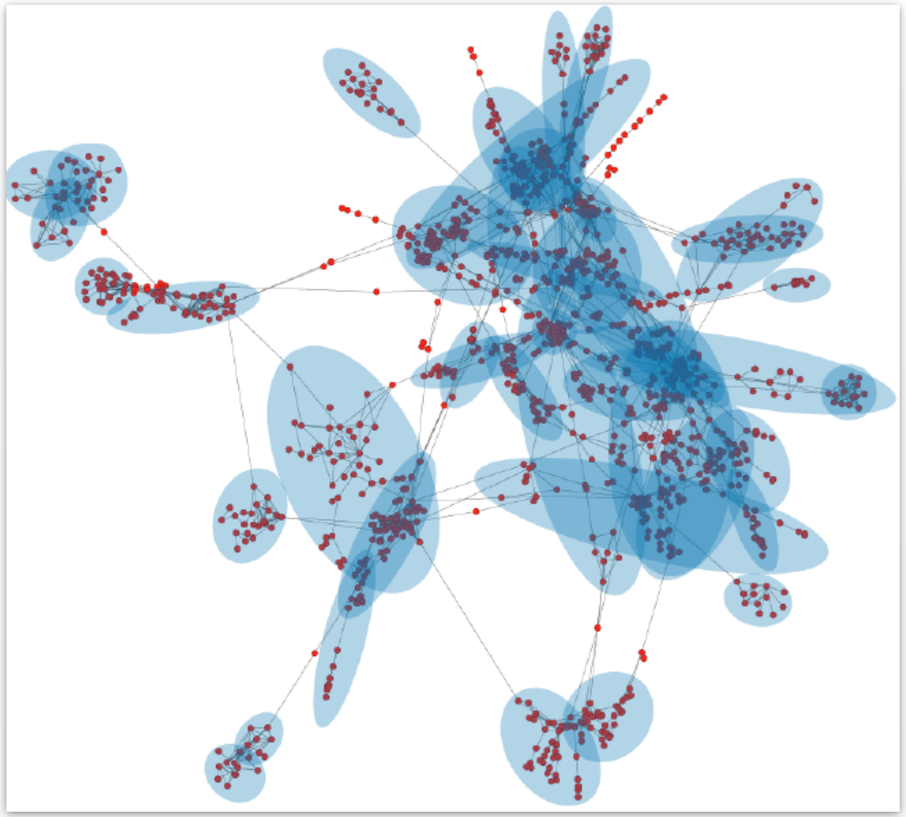


Source: Citation networks and Maps of science [Börner et al., 2012]

Nodes: Journals, *Edges:* Citations,
Communities: Academic Disciplines

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Communities: Protein-Protein Interactions



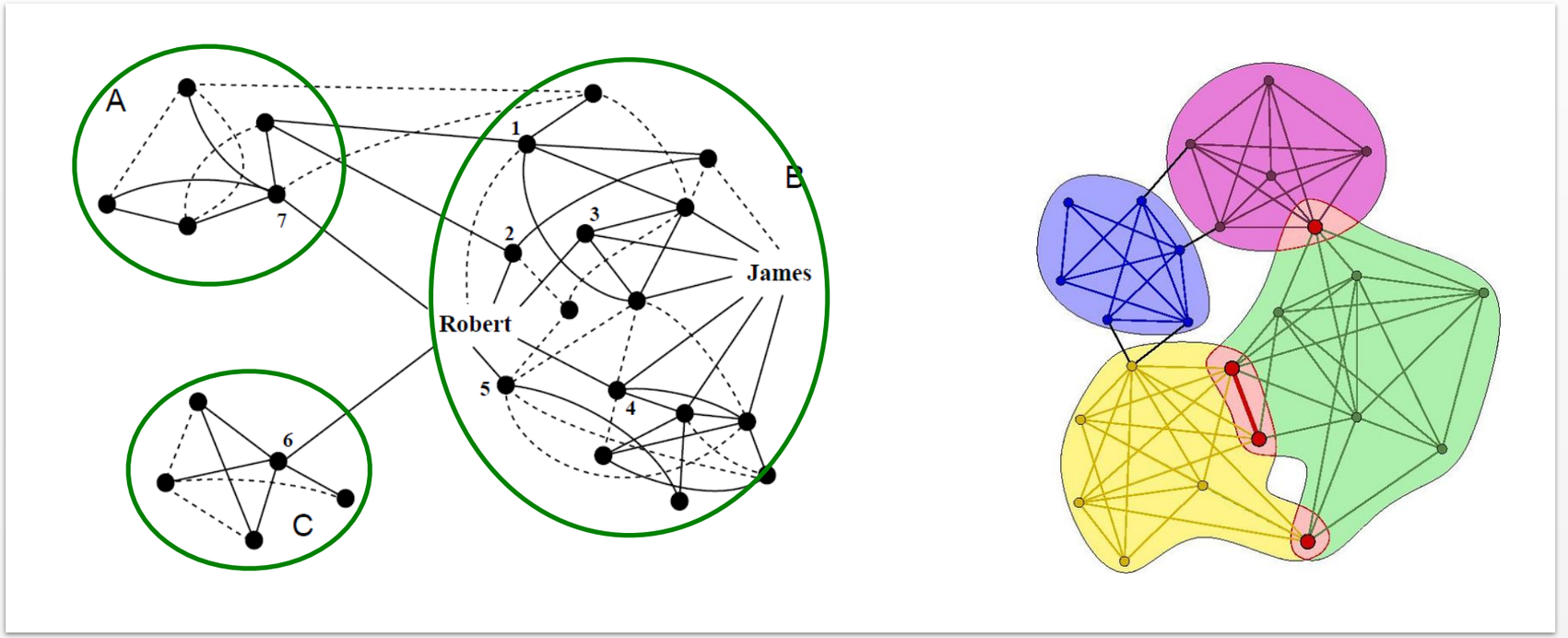
Nodes: Proteins, *Edges:* Physical interactions,
Communities: Functional Modules

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Community Detection

Graph Partitioning

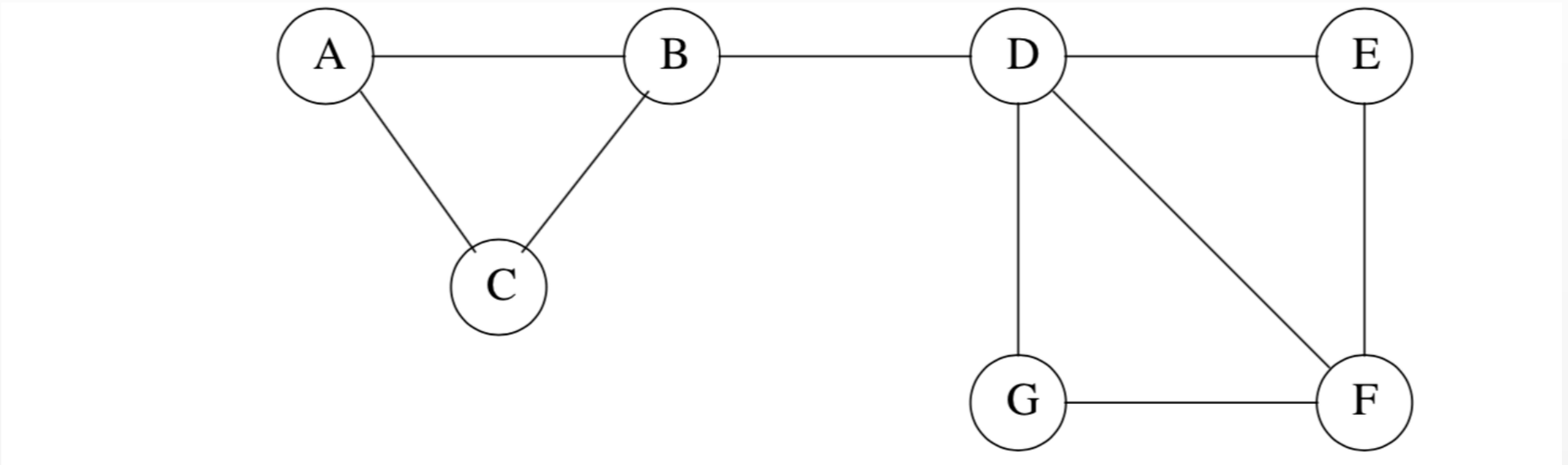
Overlapping Communities



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

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Can we use clustering technique?

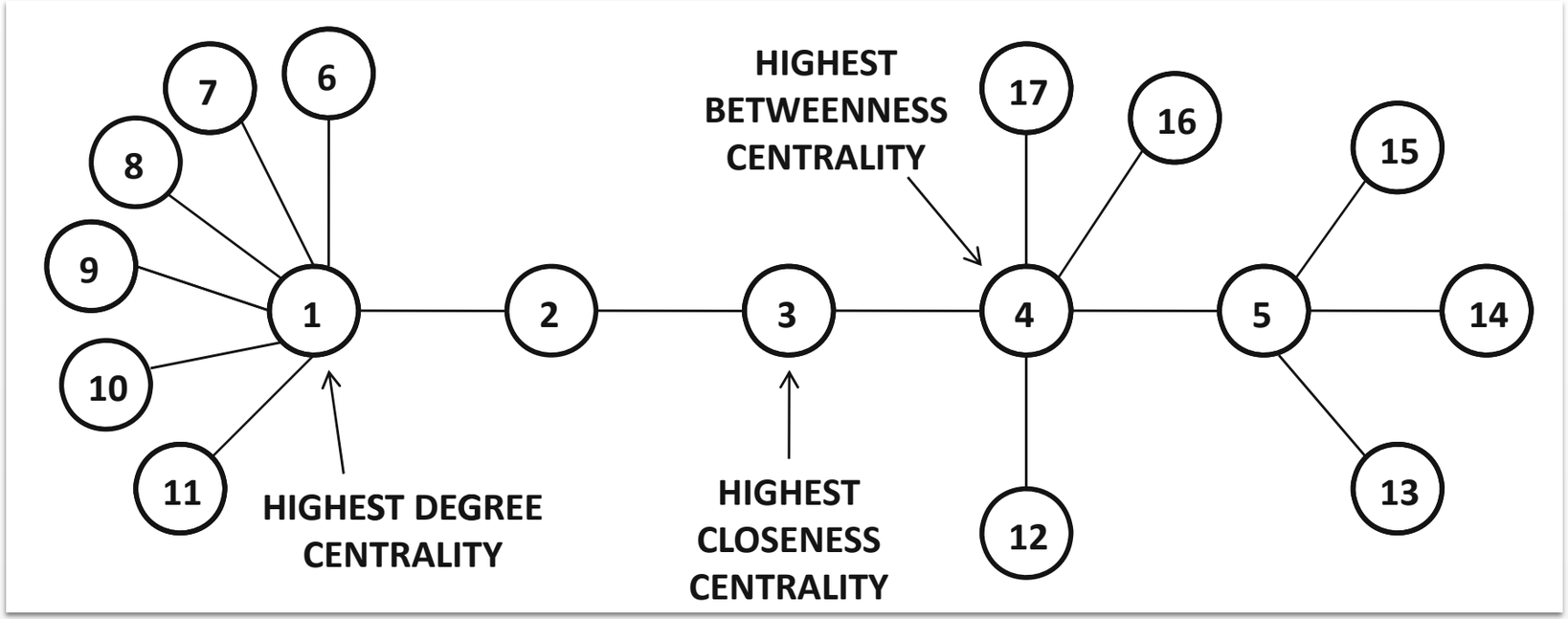


$$d(x,y) = \begin{cases} 1 & (x,y) \text{ is an edge} \\ \infty & (x,y) \text{ is not an edge} \end{cases}$$

Doesn't satisfy triangle inequality.

Since all edges are representing equal distances there is 1/9 chance that B-D will get connected in the first iteration of hierarchical agglomerative clustering, when they clearly belong to different clusters.

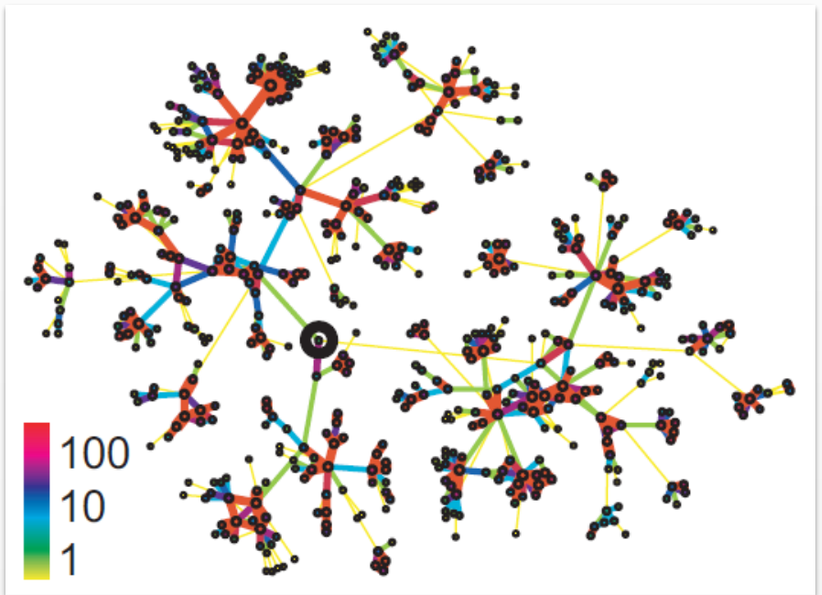
Centrality Measures for nodes



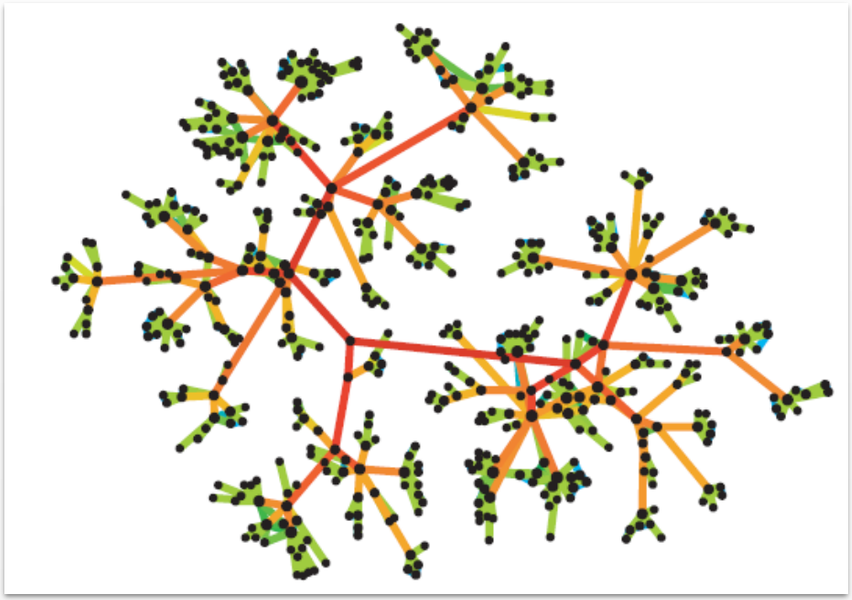
- *Betweenness*: Number of shortest paths passing through the node
- *Closeness*: Average distance to other nodes
- *Degree*: Number of connections to other nodes

Edge betweenness

Edge Strength (call volume)



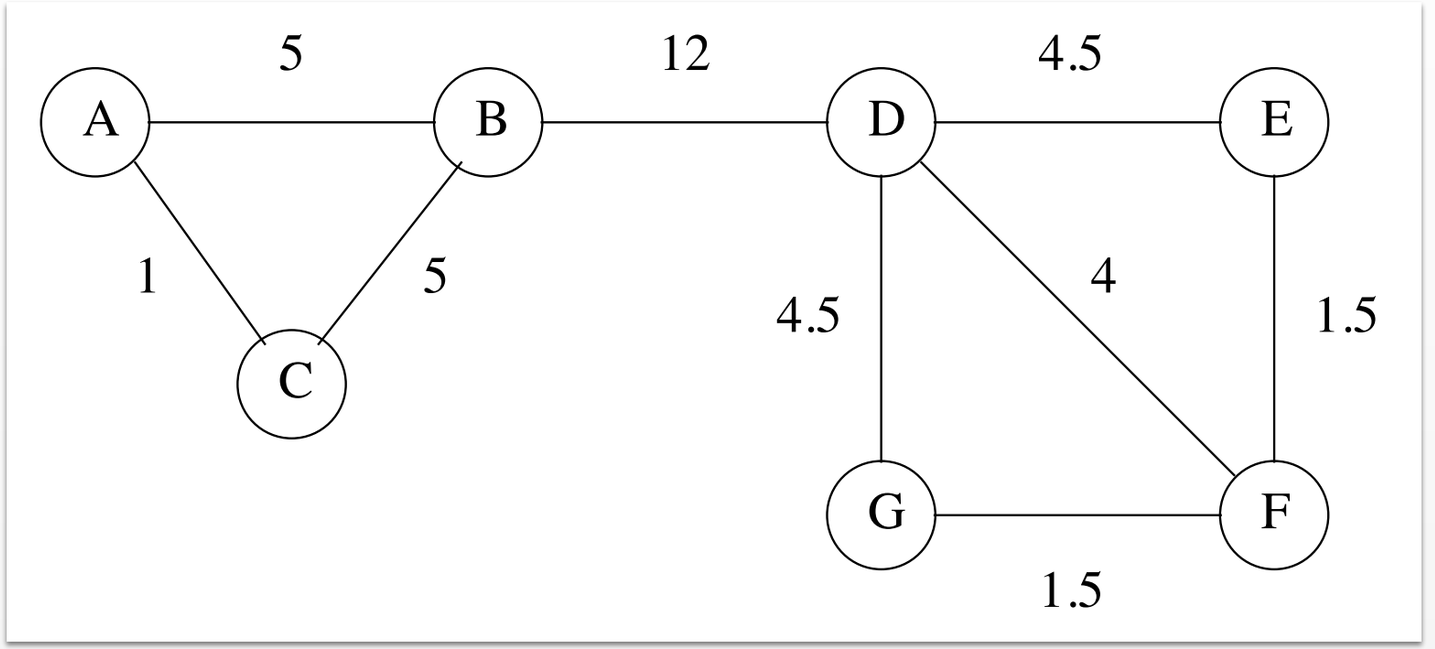
Edge Betweenness



- *Betweenness*: Number of shortest paths passing through a node or edge

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Edge Betweenness



- Count number of shortest paths passing through each edge (*can be done with weighted edges*)
- If there are multiple paths of equal length, then split counts

Community Detection

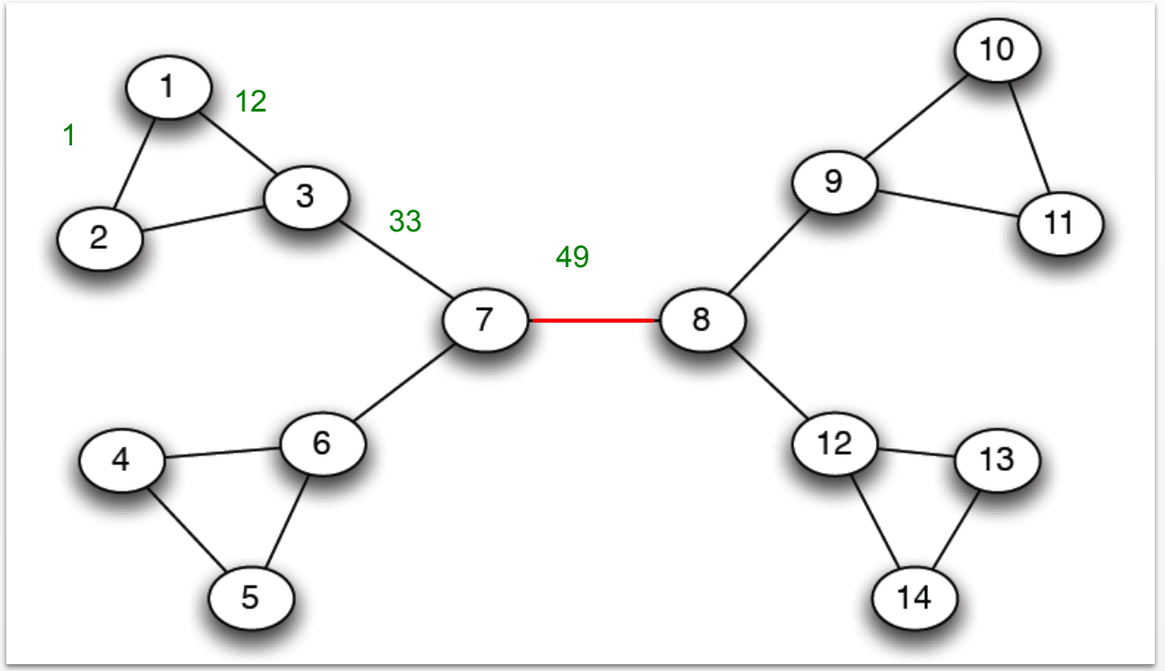
Shantanu Jain

Girvan-Newman Algorithm

Hierarchical divise clustering using the betweenness

Girvan-Newman Algorithm

(hierarchical divisive clustering according to betweenness)



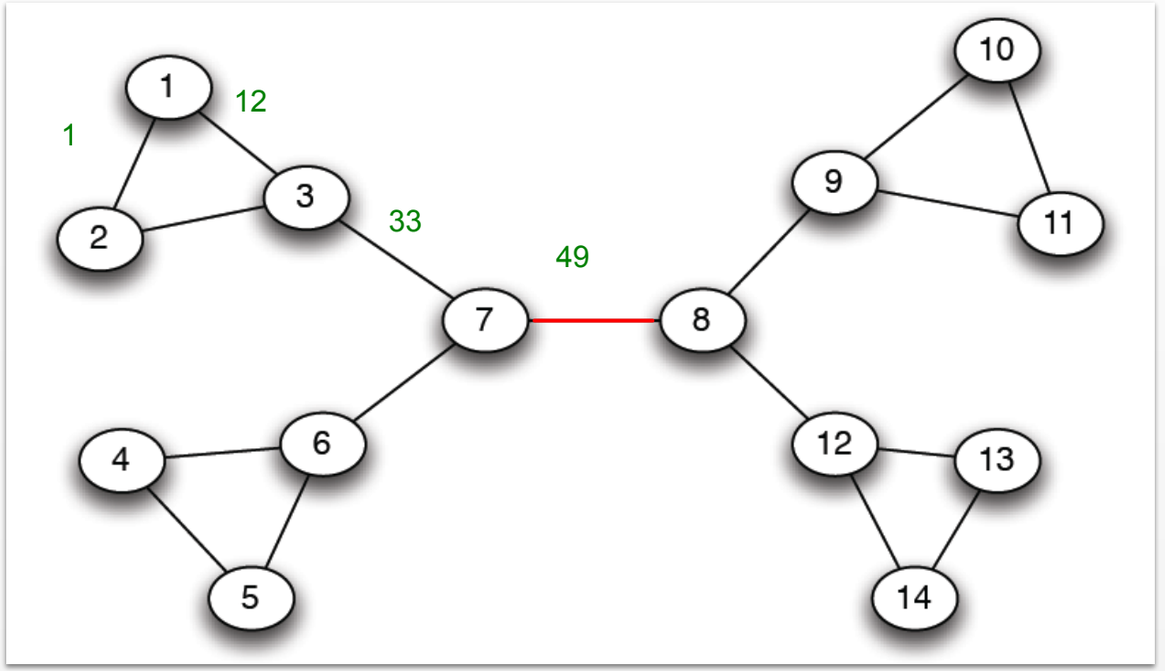
Repeat until k clusters found

1. Calculate betweenness
2. Remove edge(s) with highest betweenness

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Girvan-Newman Algorithm

(hierarchical divisive clustering according to betweenness)



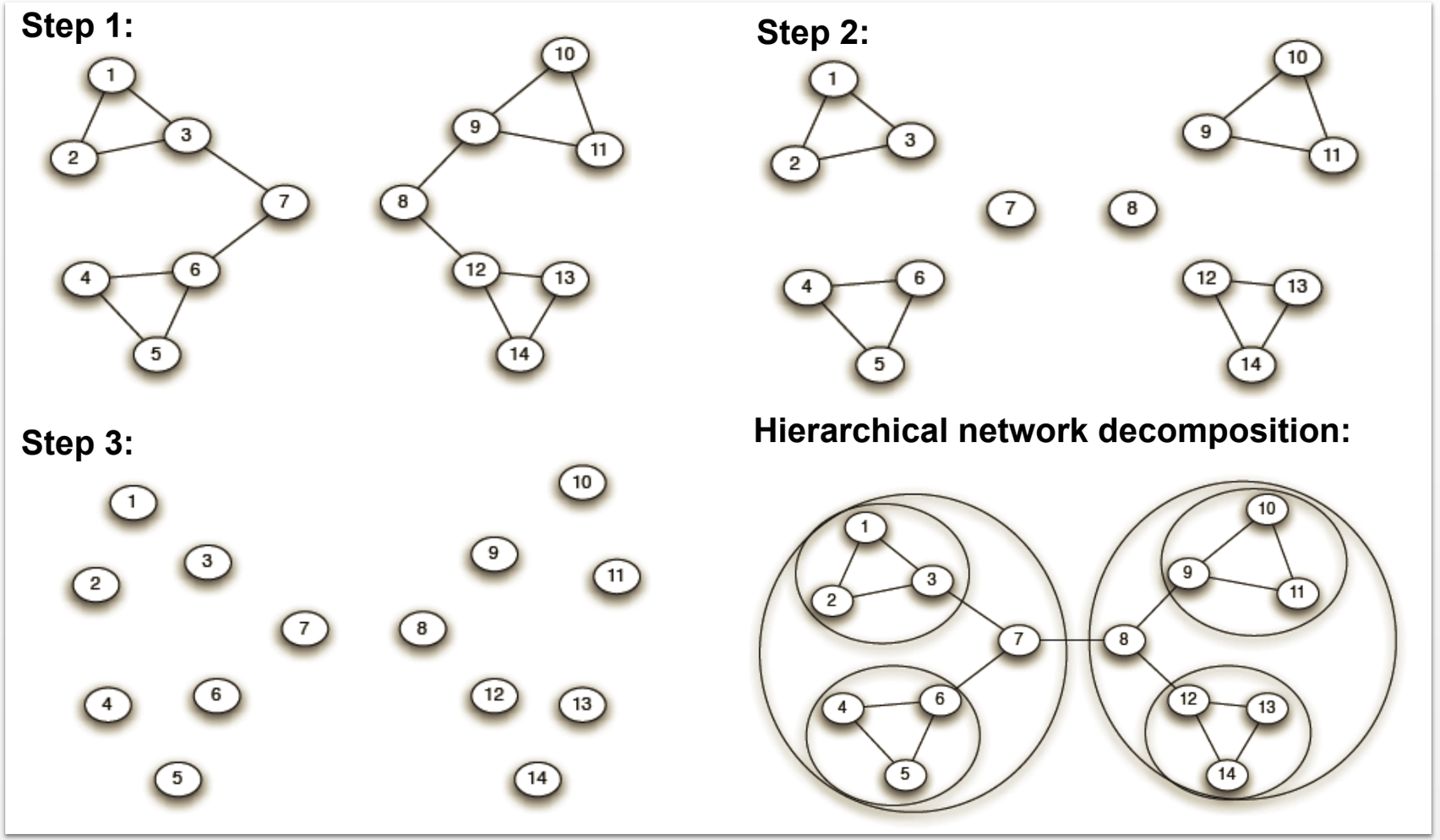
Repeat until k clusters found

1. Calculate betweenness
2. Remove edge(s) with highest betweenness

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

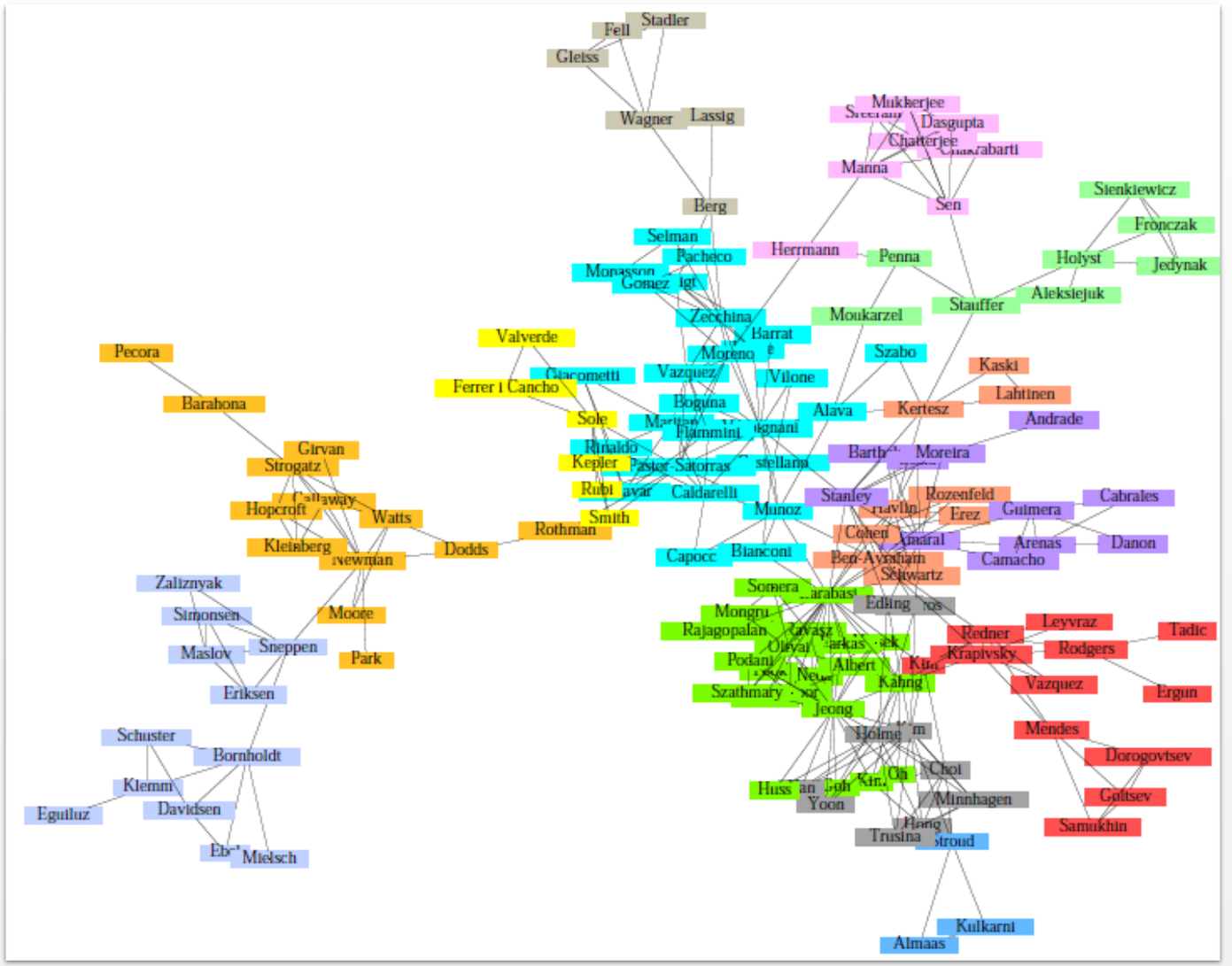
Girvan-Newman Algorithm

(hierarchical divisive clustering according to betweenness)



(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Girvan-Newman: Physics Citations



(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Girvan-Newman

Two problems

1. How can we compute the betweenness for all edges?
2. How can we choose the number of components k?

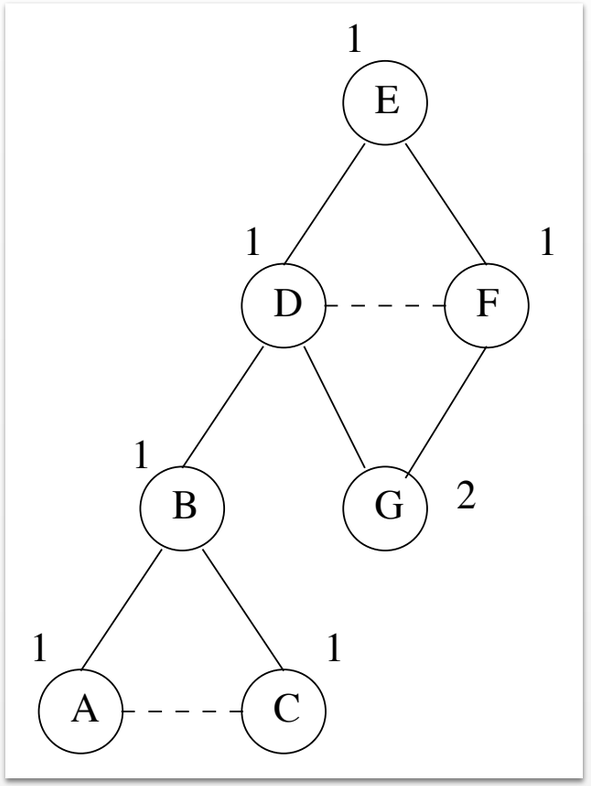
(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Calculating Betweenness

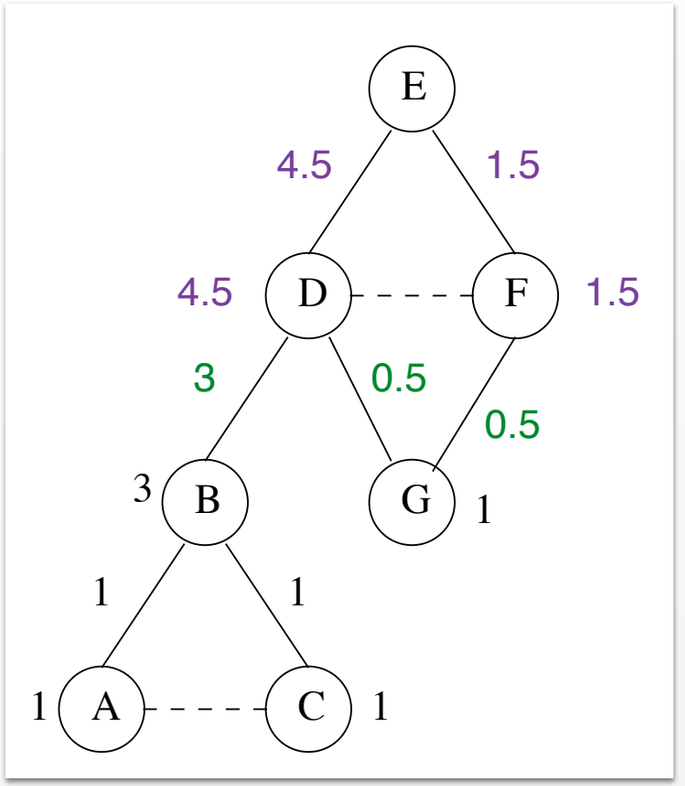
How can we count all shortest paths?

- Loop over nodes in graph
  - Perform breadth-first search to find shortest paths to other nodes
  - Increment counts for edges traversed by shorts paths
- Divide final betweenness by 2 (*since all paths counted twice*)

Counting Shortest Paths



Count number of shortest paths from (E) to each node

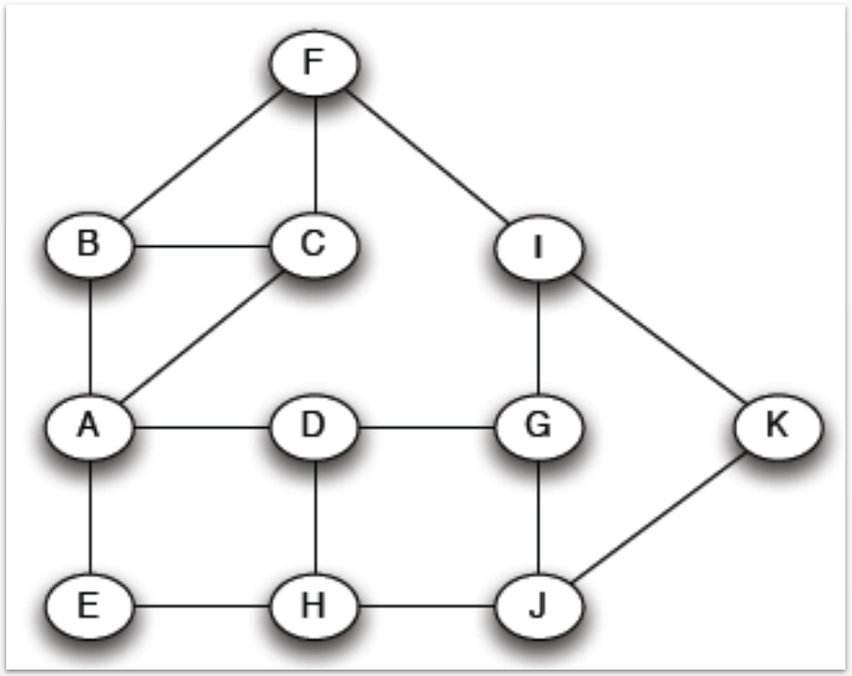


Accumulate credit upwards, dividing across shortest paths

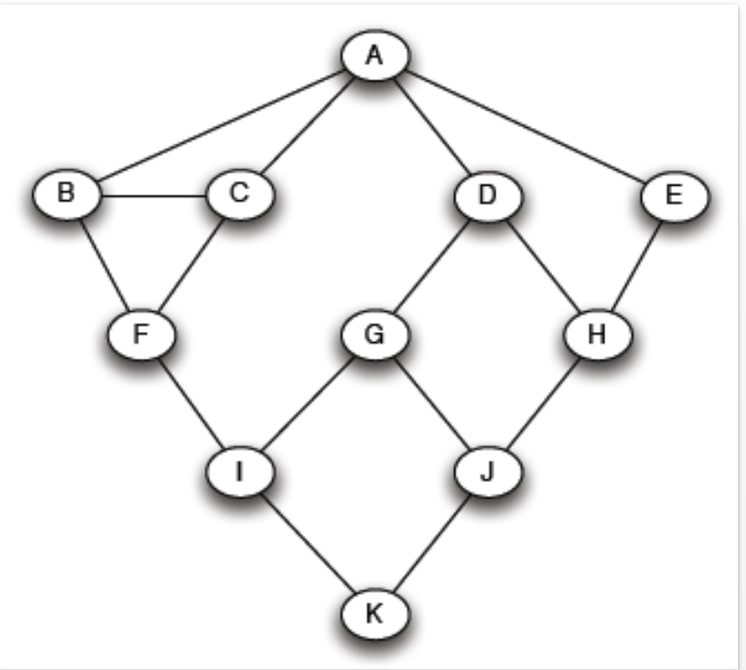
(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Counting Paths: Larger Example

Original Graph

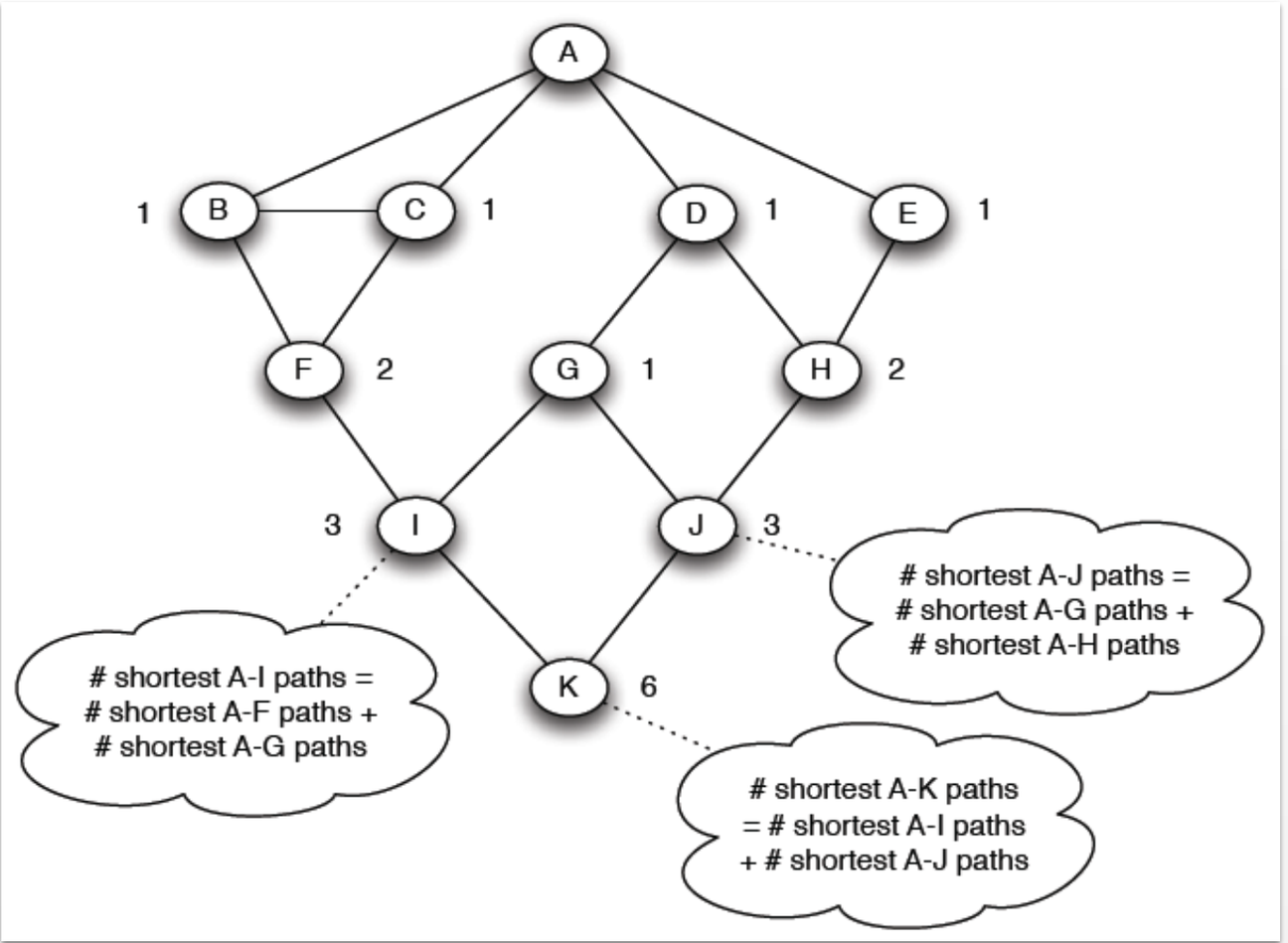


Breadth-first Ordering from A



(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

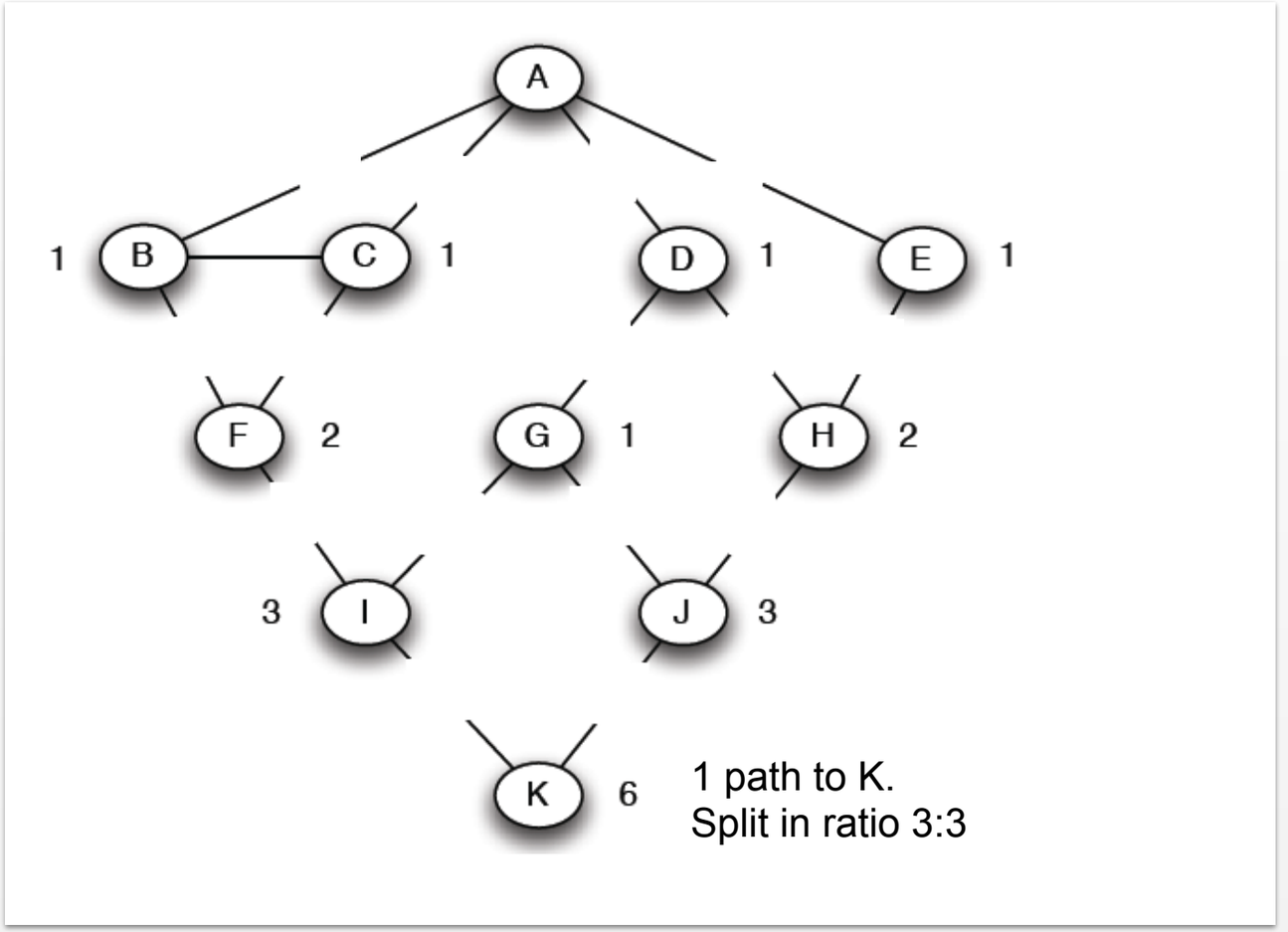
Counting Paths: Larger Example



Step 1. Count number of shortest paths from to each node

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

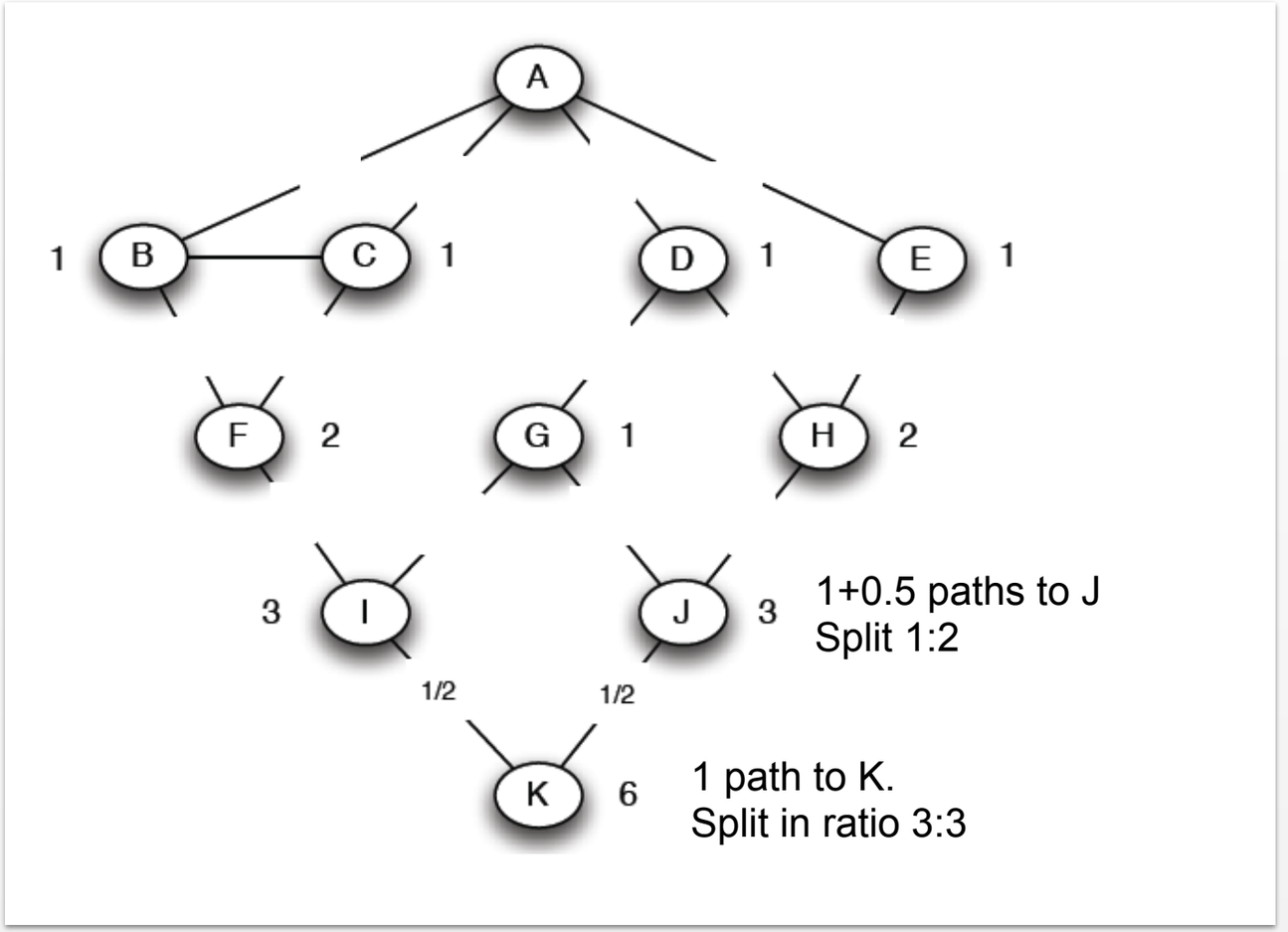
Counting Paths: Larger Example



Step 2. Propagate credit upwards, splitting according to number of paths to parents

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

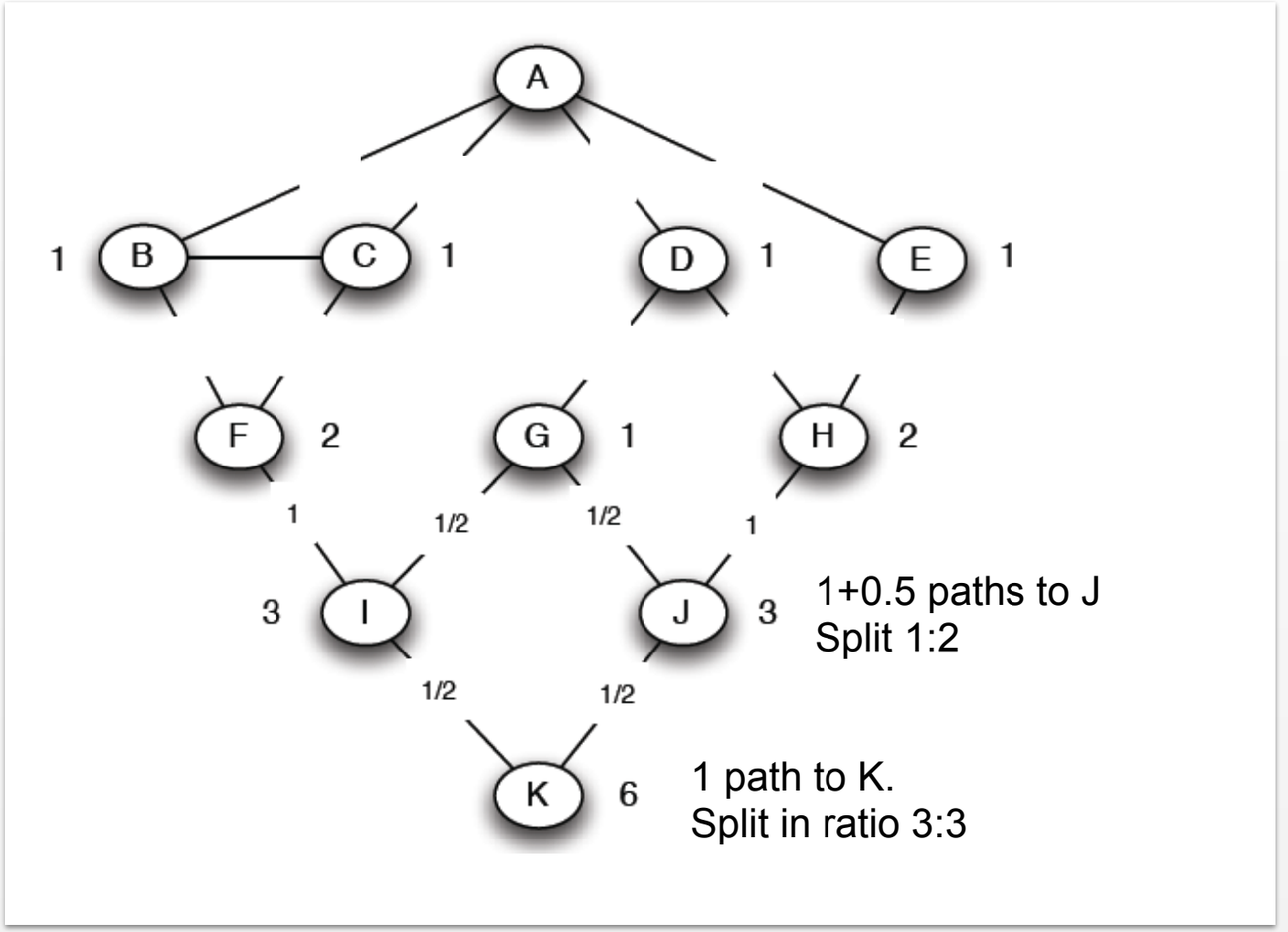
Counting Paths: Larger Example



Step 2. Propagate credit upwards, splitting according to number of paths to parents

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

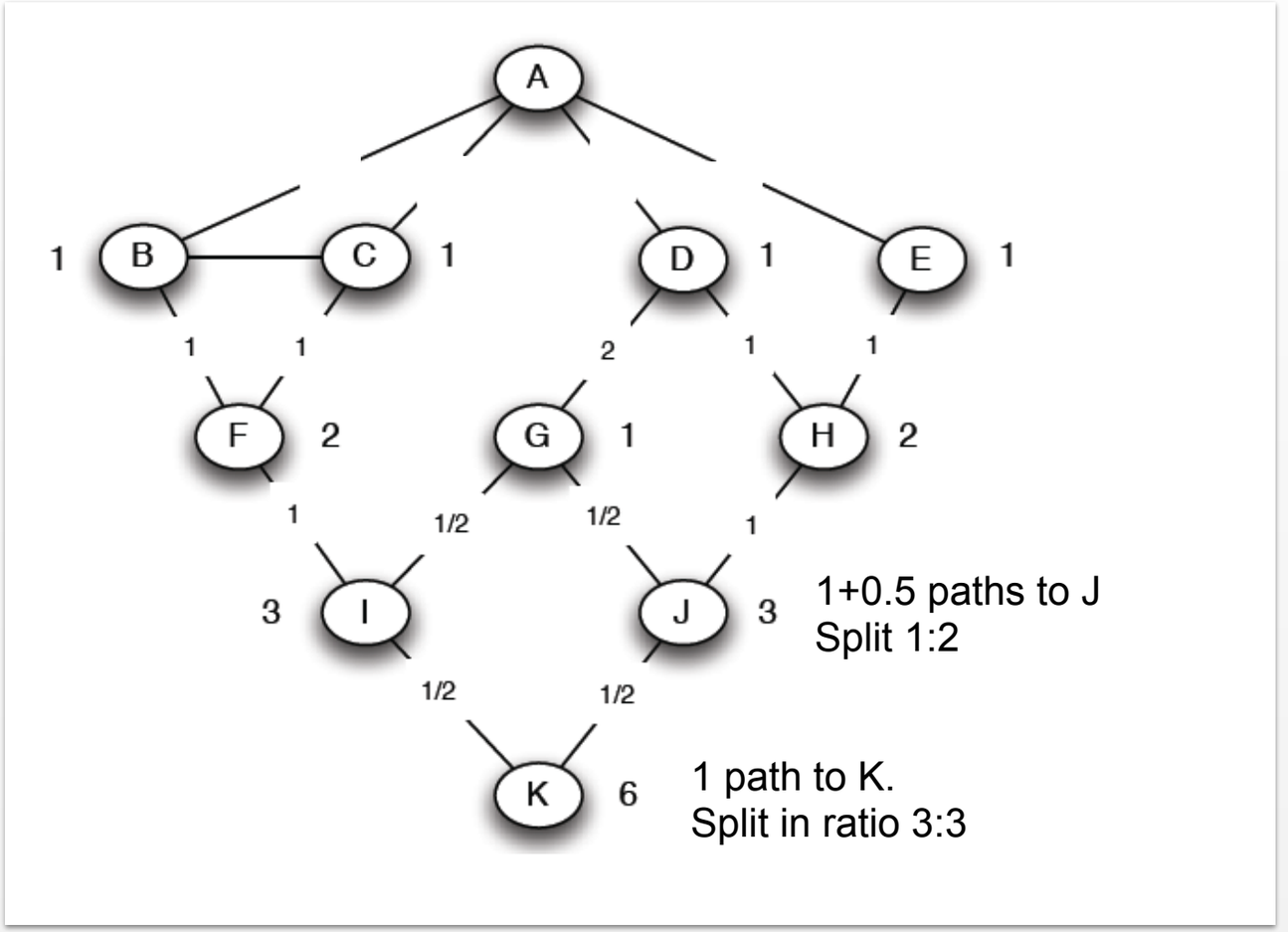
Counting Paths: Larger Example



Step 2. Propagate credit upwards, splitting according to number of paths to parents

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

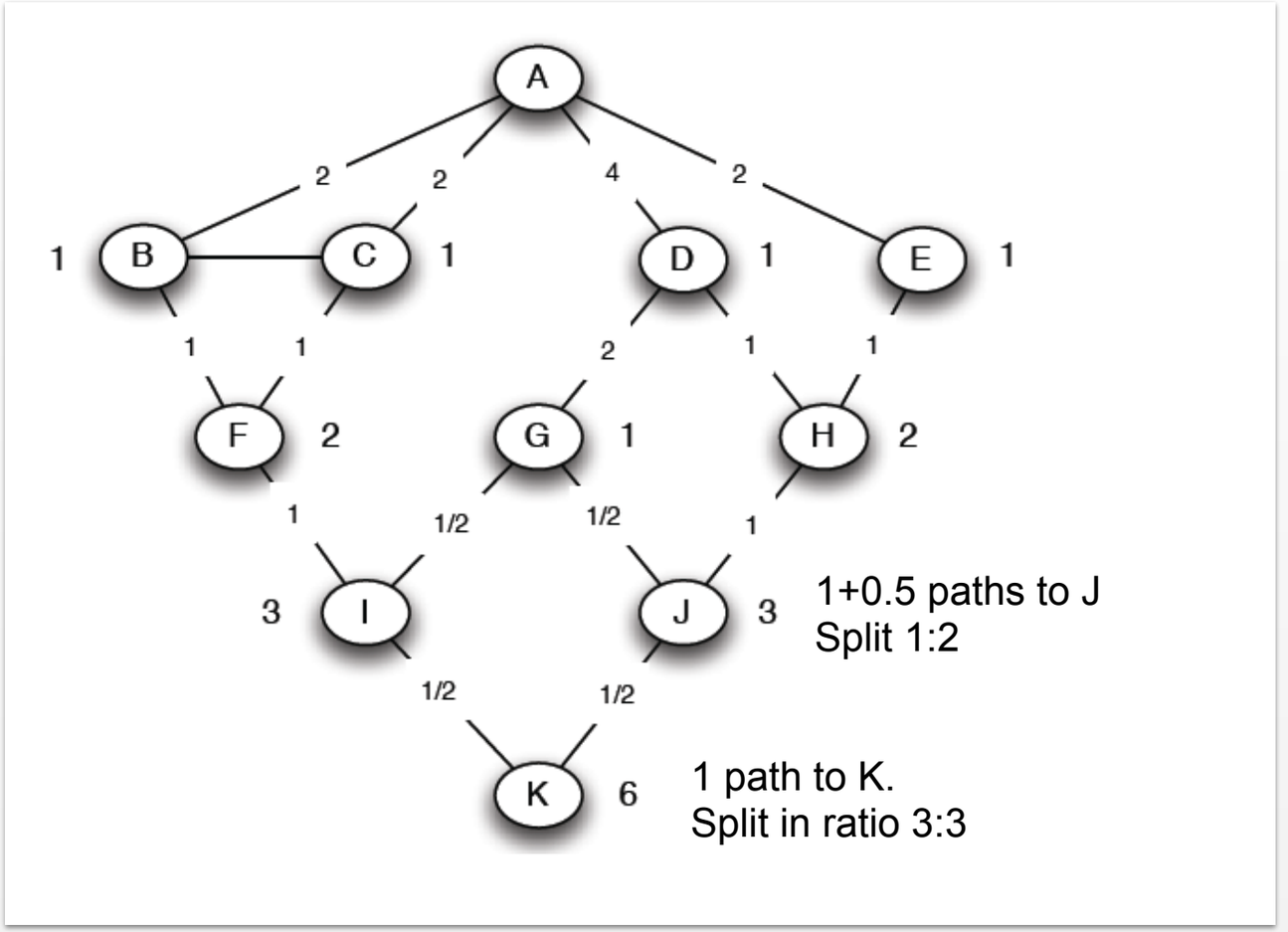
Counting Paths: Larger Example



Step 2. Propagate credit upwards, splitting according to number of paths to parents

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Counting Paths: Larger Example

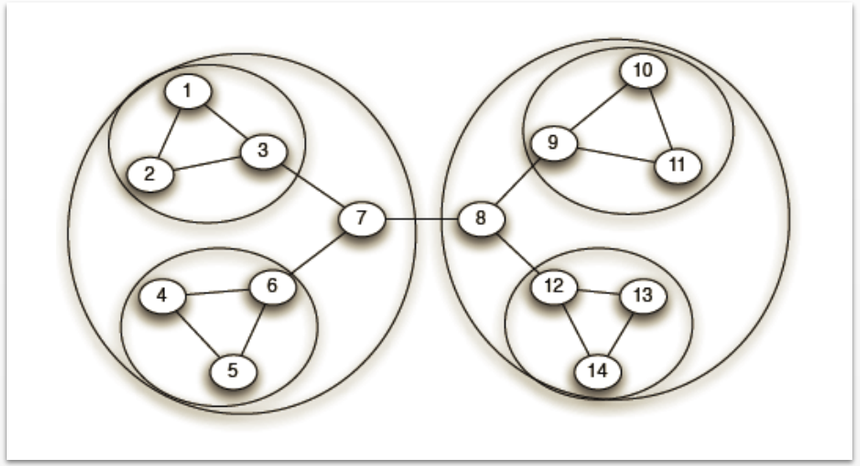


Step 2. Propagate credit upwards, splitting according to number of paths to parents

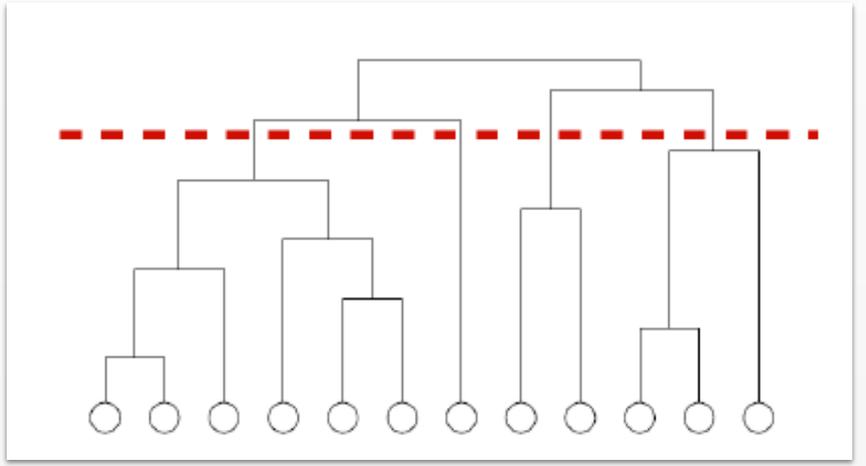
(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Determining the Number of Communities

Hierarchical decomposition



Choosing a cut-off



Analogous problem to deciding on number of clusters in hierarchical clustering

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Modularity

Idea: Compare fraction of edges within module to fraction that would be observed for random connections

$$Q = \frac{1}{2|E|} \sum_{v,w \in V} \left(A_{vw} - \frac{d_v d_w}{2|E|} \right) I[z_v = z_w]$$

Existence of an edge between *v* and *w*

Probability of seeing edge between *v* and *w* at random (given the degree of nodes and number of edges in the graph)

Adjacency Matrix

$$A_{vw} = I[(v,w) \in E]$$

Node Degree

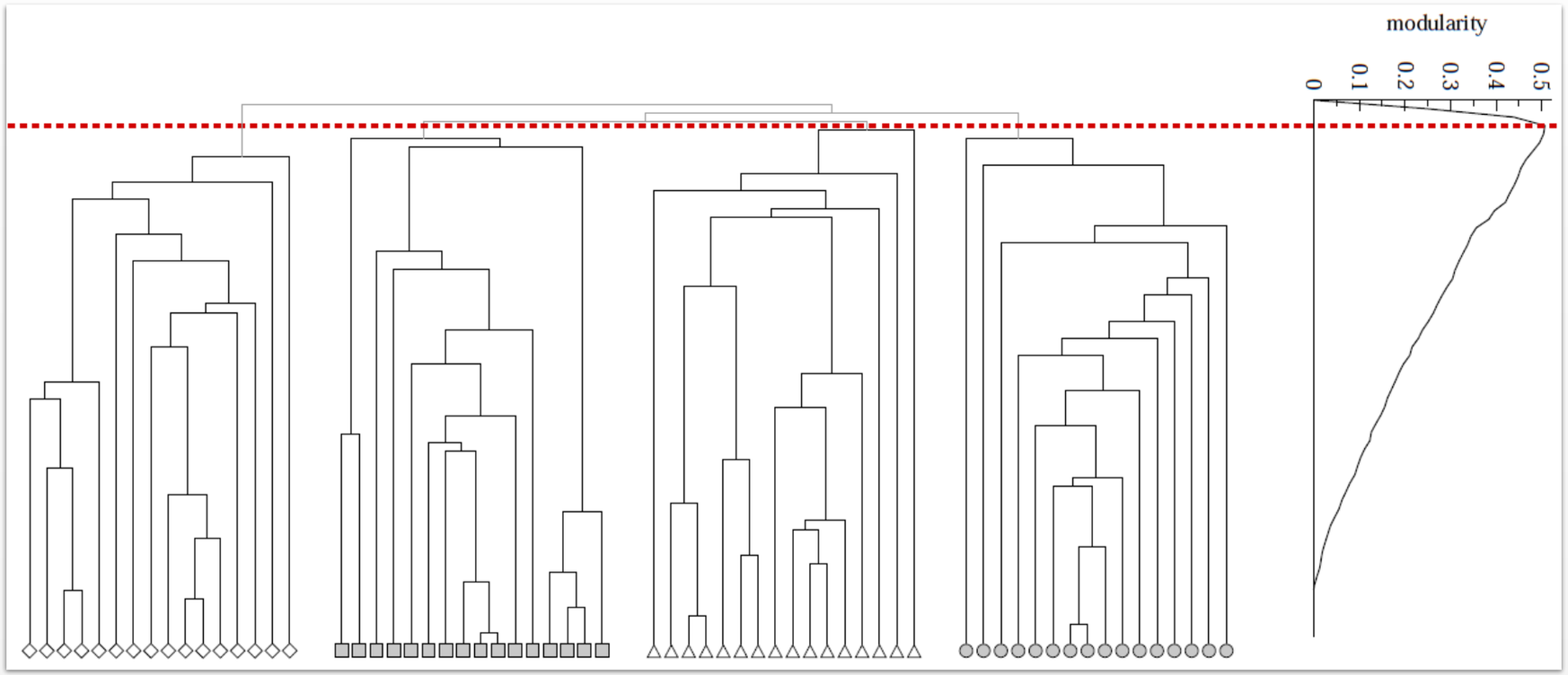
$$d_v = \sum_{w \in V} A_{vw}$$

Community Assignment of a node

$$z_v \in \{1, \ldots, K\}$$

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)

Modularity



Use modularity to optimize connectivity within modules

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)