Clusters and Communities

Lecture 7

CSCI 4974/6971

22 Sep 2016

Today's Biz

1. Reminders
2. Review
3. Communities
4. Betweenness and Graph Partitioning
5. Label Propagation

Today's Biz

1. **Reminders**
2. Review
3. Communities
4. Betweenness and Graph Partitioning
5. Label Propagation

Reminders

- Project Proposal: due today - expect email this weekend
- Assignment 1: Grades via email tomorrow, solution posted
- Assignment 2: Thursday 29 Sept 16:00
- Project Presentation 1: in class 6 October
- Office hours: Tuesday & Wednesday 14:00-16:00 Lally 317
  - Or email me for other availability
- Class schedule:
  - Social net analysis methods
  - Bio net analysis methods
  - Random networks and usage

Today's Biz

1. Reminders
2. **Review**
3. Communities
4. Betweenness and Graph Partitioning
5. Label Propagation

Quick Review

Strong and weak ties:

- Clustering coefficient - how many of your friends are friends?
- Triadic closure - your friends likely to become friends (more likely if connections are strong ties)
- Bridges - often weak ties, connect disparate parts of the network
- Limits of human social interaction is about 150 strong ties, thousands of weak ties

Quick Review

Network context and evolution:

- Homophily - like attracts like, social connections tend to exist between those who are similar
  - Selective influence - become friends with people similar to yourself
  - Social influence - become more similar to people with whom you are friends
- Affiliation networks - network of people and their affiliations (job, club, etc.)
  - Triadic closure - two mutual friends become friends
  - Focal closure - two people become friends through affiliation
  - Membership Closure - join affiliation with your friend

Quick Review

Distributed triangle counting:

- Can use to calculate clustering coefficient for all vertices
- Data skew is problematic - naive parallelization not effective
  - Explicitly handle data skew
  - Partition data
- This problem and solutions are representable of many real-world graph and analytics

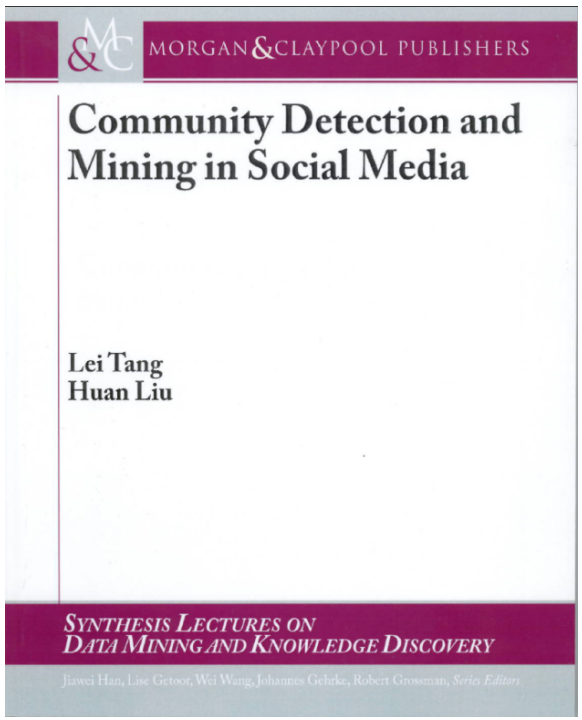
Today's Biz

1. Quick Review
2. Review
3. **Communities**
4. Betweenness and Graph Partitioning
5. Label Propagation

Community Detection and Clustering

Slides from Qiang Yang, UST, HongKong

Community Detection and Graph-based Clustering



Adapted from Chapter 3

Of

Lei Tang and Huan Liu's Book

Slides prepared by Qiang Yang,

UST, HongKong

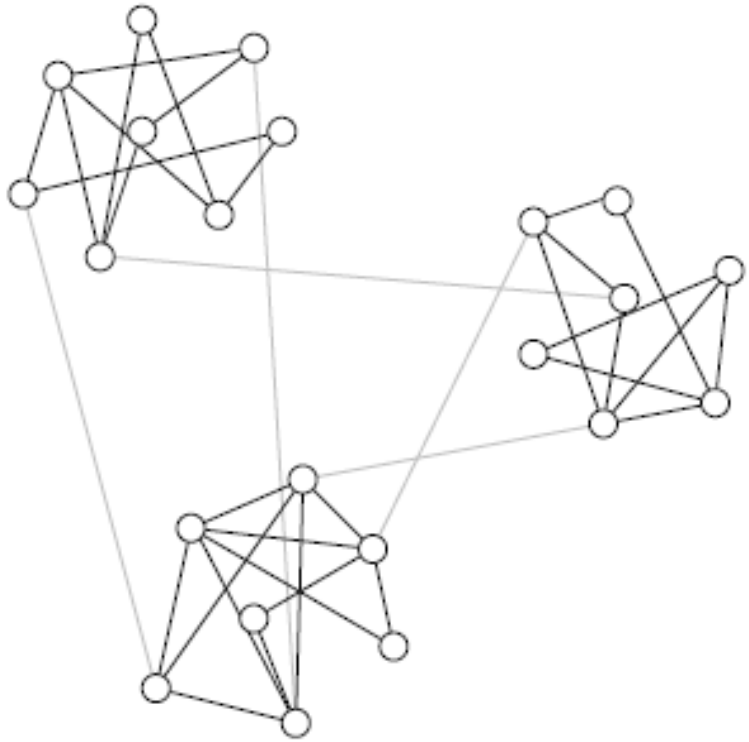


FIG. 1: A schematic representation of a network with community structure. In this network there are three communities of densely connected vertices (circles with solid lines), with a much lower density of connections (gray lines) between them.

Community

- Community: It is formed by individuals such that those within a group interact with each other more frequently than with those outside the group
  - a.k.a. group, cluster, cohesive subgroup, module in different contexts
- Community detection: discovering groups in a network where individuals' group memberships are not explicitly given

- Why communities in social media?
  - Human beings are social
  - Easy-to-use social media allows people to extend their social life in unprecedented ways
  - Difficult to meet friends in the physical world, but much easier to find friend online with similar interests
  - Interactions between nodes can help determine communities

Communities in Social Media

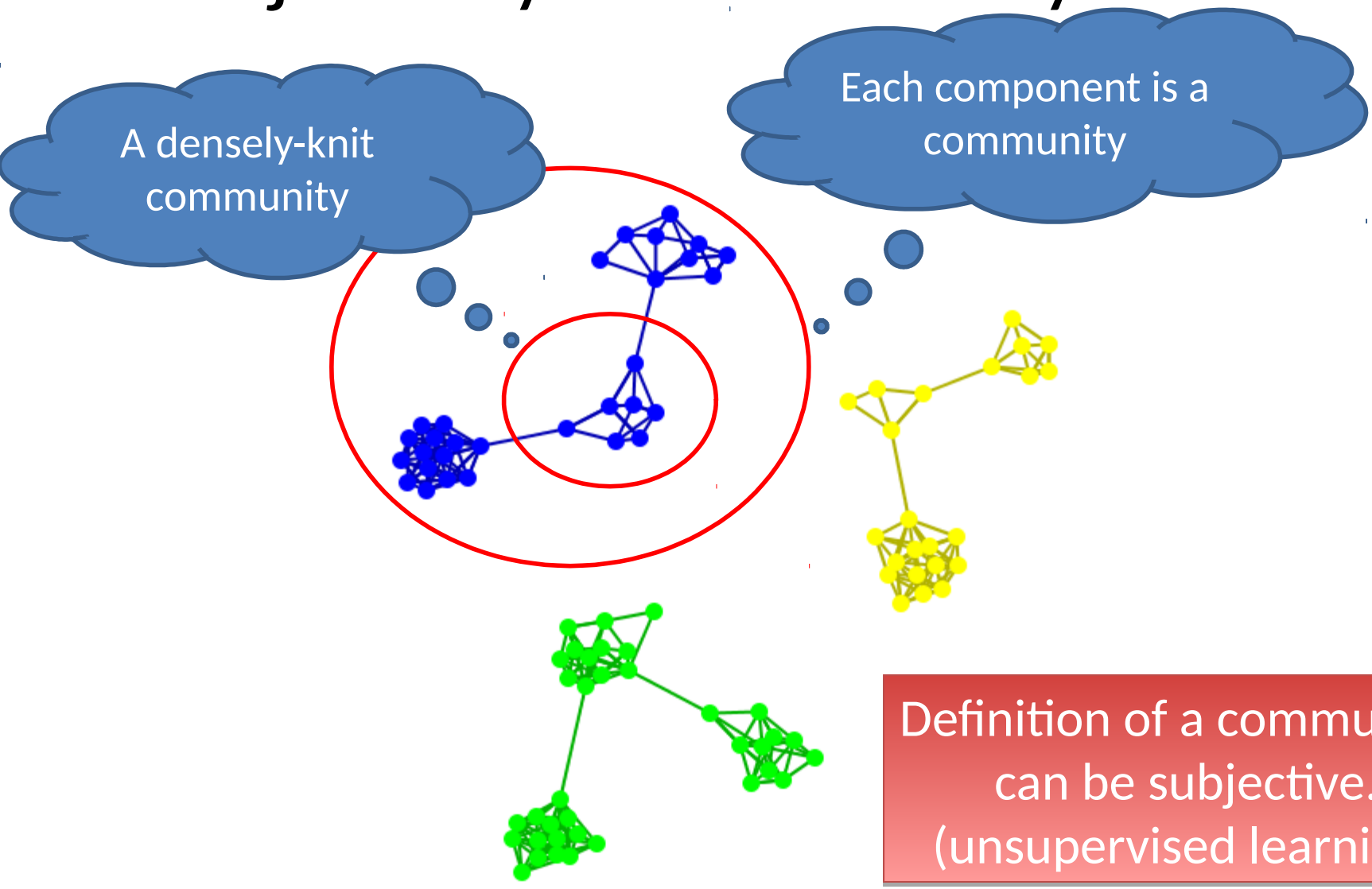
- Two types of groups in social media
  - Explicit Groups: formed by user subscriptions
  - Implicit Groups: implicitly formed by social interactions

- Some social media sites allow people to join groups, is it necessary to extract groups based on network topology?
  - Not all sites provide community platform
  - Not all people want to make effort to join groups
  - Groups can change dynamically
- Network interaction provides rich information about the relationship between users
  - Can complement other kinds of information, e.g. user profile
  - Help network visualization and navigation
  - Provide basic information for other tasks, e.g. recommendation

Note that each of the above three points can be a research topic.

COMMUNITY DETECTION

Subjectivity of Community Definition



Definition of a community can be subjective. (unsupervised learning)

Taxonomy of Community Criteria

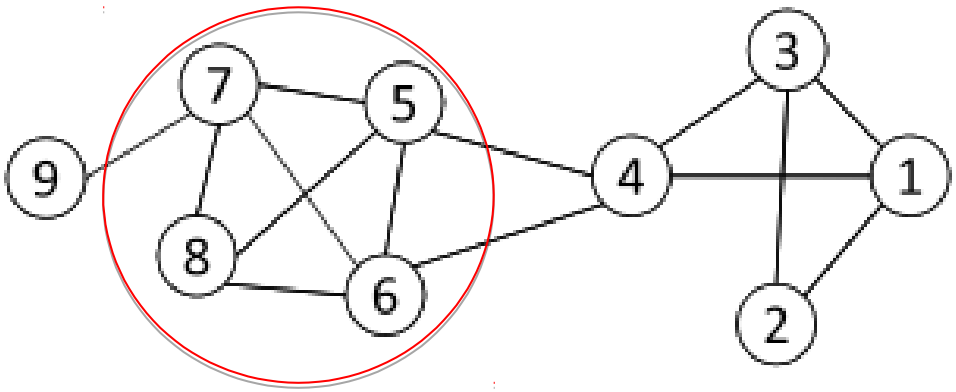
- Criteria vary depending on the tasks
- Roughly, community detection methods can be divided into 4 categories (not exclusive):
- Node-Centric Community
  - Each node in a group satisfies certain properties
- Group-Centric Community
  - Consider the connections within a group as a whole. The group has to satisfy certain properties without zooming into node-level
- **Network-Centric Community**
  - Partition the whole network into several disjoint sets
- Hierarchy-Centric Community
  - Construct a hierarchical structure of communities

Node-Centric Community Detection

- Nodes satisfy different properties
  - Complete Mutuality
    - cliques
  - Reachability of members
    - k-clique, k-clan, k-club
  - Nodal degrees
    - k-plex, k-core
  - Relative frequency of Within-Outside Ties
    - LS sets, Lambda sets
- Commonly used in traditional social network analysis
- Here, we discuss some representative ones

Complete Mutuality: Cliques

- Clique: a maximum complete subgraph in which all nodes are adjacent to each other



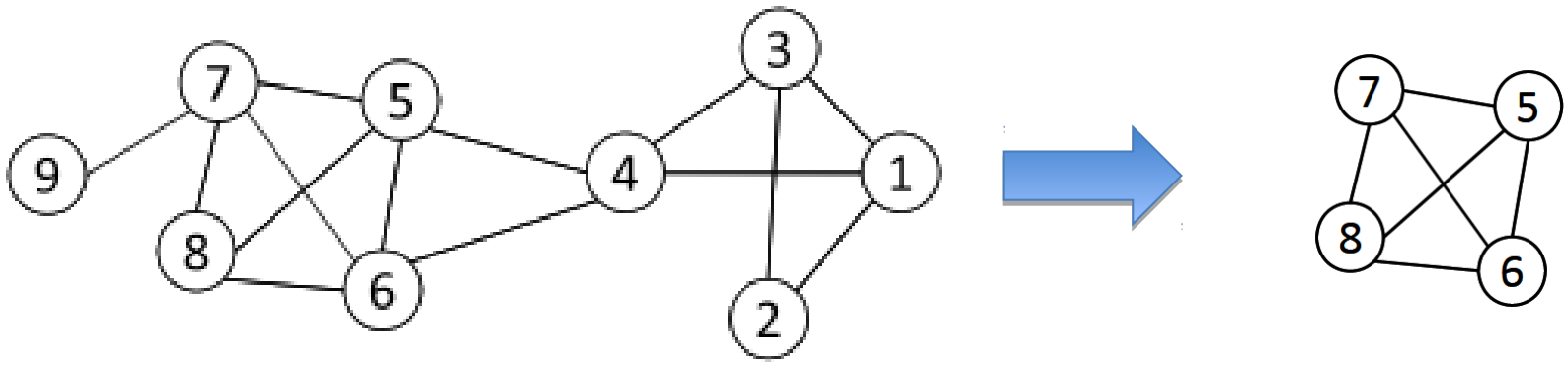
Nodes 5, 6, 7 and 8 form a clique

- NP-hard to find the maximum clique in a network
- Straightforward implementation to find cliques is very expensive in time complexity

Finding the Maximum Clique

- In a clique of size k, each node maintains degree >= k-1
  - Nodes with degree < k-1 will not be included in the maximum clique
- Recursively apply the following pruning procedure
  - Sample a sub-network from the given network, and find a clique in the sub-network, say, by a greedy approach
  - Suppose the clique above is size k, in order to find out a *larger* clique, all nodes with degree <= k-1 should be removed.
- Repeat until the network is small enough
- Many nodes will be pruned as social media networks follow a power law distribution for node degrees

Maximum Clique Example



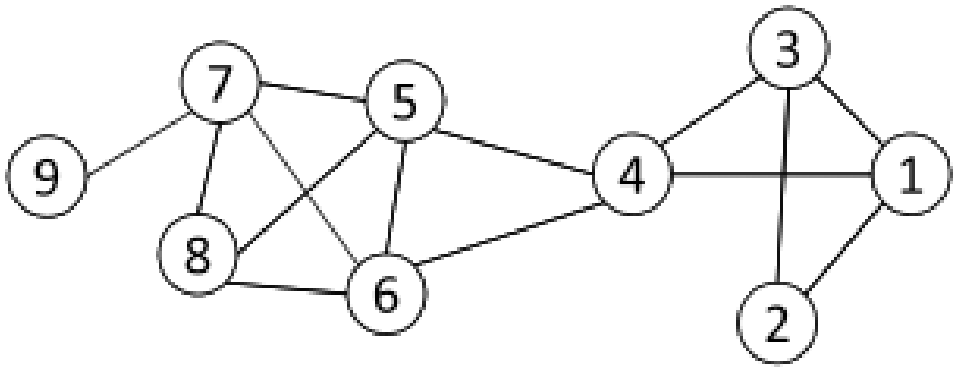
- Suppose we sample a sub-network with nodes {1-9} and find a clique {1, 2, 3} of size 3
- In order to find a clique >3, remove all nodes with degree <=3-1=2
  - Remove nodes 2 and 9
  - Remove nodes 1 and 3
  - Remove node 4

Clique Percolation Method (CPM)

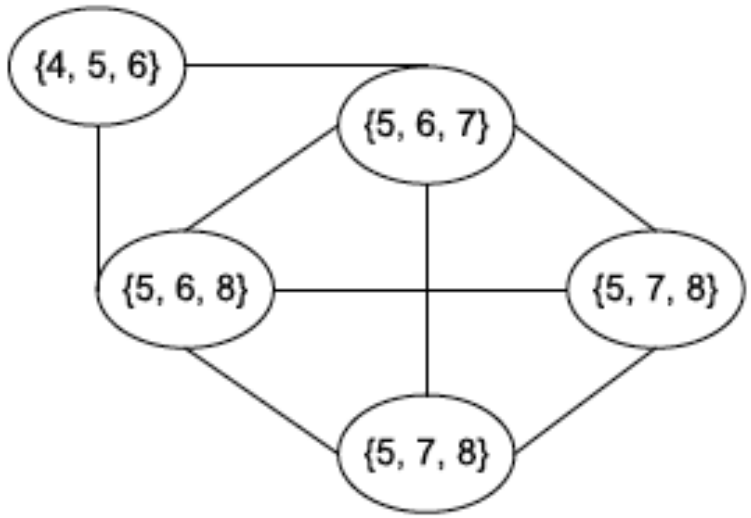
- Clique is a very strict definition, unstable
- Normally use cliques as a core or a seed to find larger communities

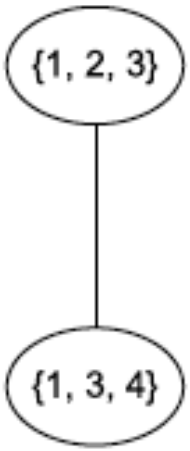
- CPM is such a method to find overlapping communities
  - **Input**
    - A parameter k, and a network
  - **Procedure**
    - Find out all cliques of size k in a given network
    - Construct a clique graph. Two cliques are adjacent if they share k-1 nodes
    - Each connected component in the clique graph forms a community

CPM Example



Cliques of size 3:
{1, 2, 3}, {1, 3, 4}, {4, 5, 6},
{5, 6, 7}, {5, 6, 8}, {5, 7, 8},
{6, 7, 8}

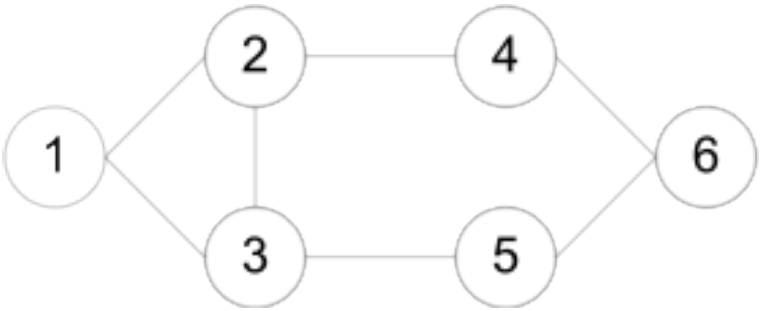




Communities:
{1, 2, 3, 4}
{4, 5, 6, 7, 8}

Reachability : k-clique, k-club

- Any node in a group should be reachable in k hops
- k-clique: a maximal subgraph in which the largest geodesic distance between any two nodes <= k
- k-club: a substructure of diameter <= k



Cliques: {1, 2, 3}
2-cliques: {1, 2, 3, 4, 5}, {2, 3, 4, 5, 6}
2-clubs: {1,2,3,4}, {1, 2, 3, 5}, {2, 3, 4, 5, 6}

- A k-clique might have diameter larger than k in the subgraph
  - E.g. {1, 2, 3, 4, 5}
- Commonly used in traditional SNA
- Often involves combinatorial optimization

Group-Centric Community Detection: Density-Based Groups

- The group-centric criterion requires the whole group to satisfy a certain condition
  - E.g., the group density >= a given threshold
- A subgraph $G_s(V_s, E_s)$ is a $dense$ quasi-clique if

$$\frac{2|E_s|}{|V_s|(|V_s|-1)} \geq \gamma$$

where the denominator is the maximum number of degrees.

- A similar strategy to that of cliques can be used
  - Sample a subgraph, and find a maximal quasi-clique (say, of size)
  - Remove nodes with degree $< \gamma$ n the average degree

$\gamma - dense$

$|V_s|$

$|V_s|\gamma \leq \frac{2|E_s|}{|V_s|-1}$

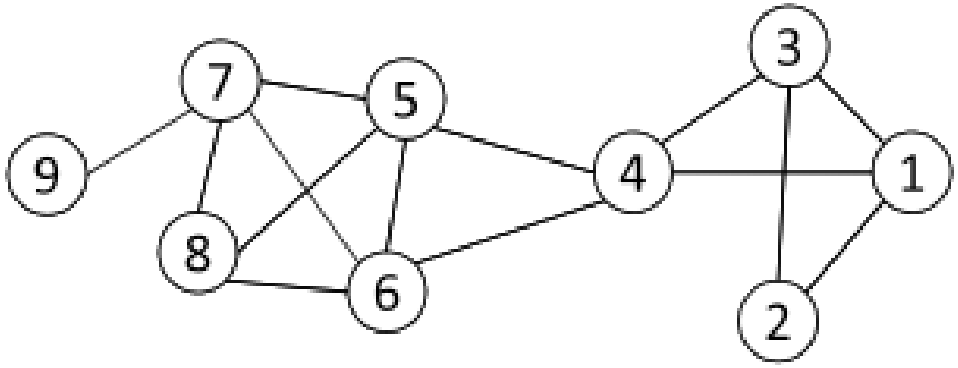
Network-Centric Community Detection

- Network-centric criterion needs to consider the connections within a network globally
- Goal: partition nodes of a network into disjoint sets
- Approaches:
  - **(1) Clustering based on vertex similarity**
  - (2) Latent space models (multi-dimensional scaling)
  - (3) Block model approximation
  - (4) Spectral clustering
  - **(5) Modularity maximization**

Clustering based on Vertex Similarity

- Apply k-means or similarity-based clustering to nodes
- Vertex similarity is defined in terms of the similarity of their neighborhood
- Structural equivalence: two nodes are structurally equivalent iff they are connecting to the same set of actors

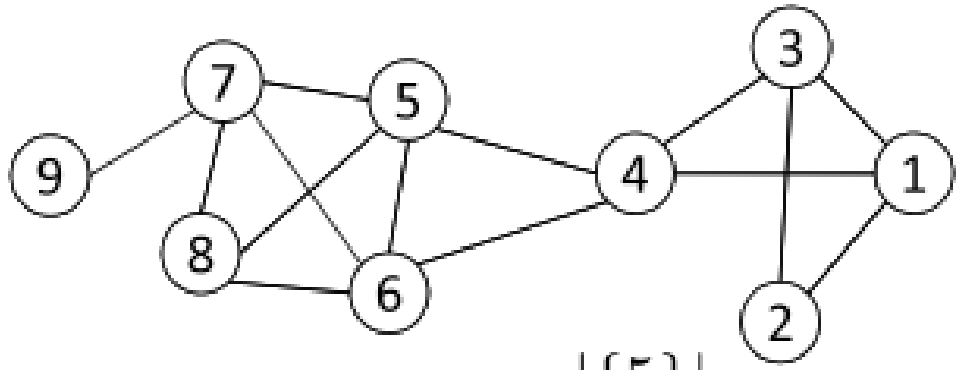
Nodes 1 and 3 are structurally equivalent;
So are nodes 5 and 6.



- Structural equivalence is too strict for practical use.

Vertex Similarity

- Jaccard Similarity $Jaccard(v_i, v_j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$
- Cosine similarity $Cosine(v_i, v_j) = \frac{|N_i \cap N_j|}{\sqrt{|N_i| \cdot |N_j|}}$

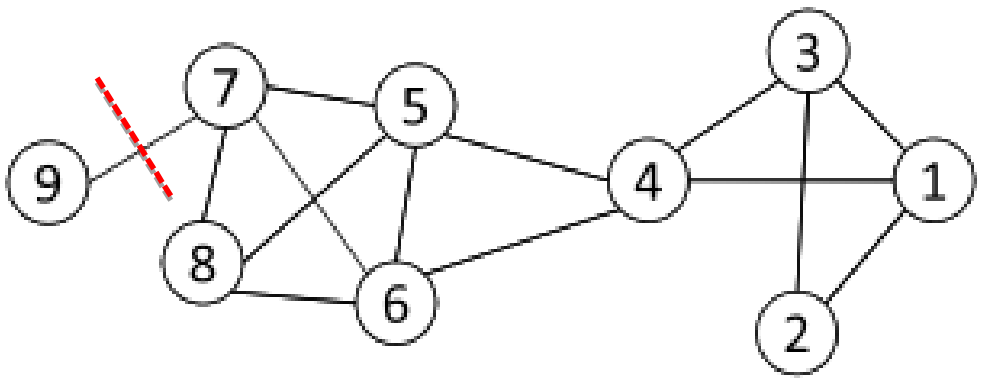


$$Jaccard(4,6) = \frac{|\{5\}|}{|\{1,3,4,5,6,7,8\}|} = \frac{1}{7}$$

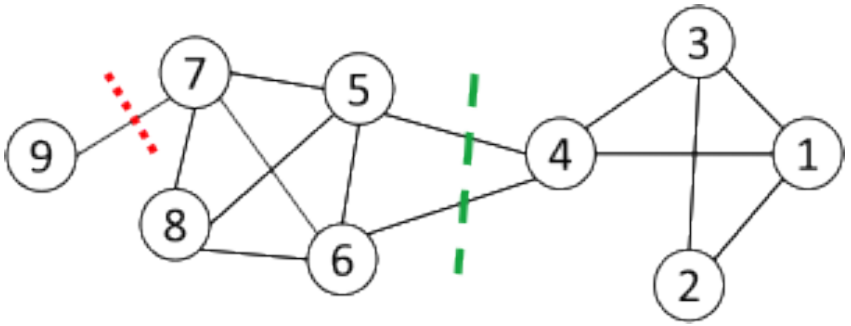
$$cosine(4,6) = \frac{1}{\sqrt{4 \cdot 4}} = \frac{1}{4}$$

Cut

- Most interactions are within group whereas interactions between groups are few
- community detection ✉ minimum cut problem
- Cut: A partition of vertices of a graph into two disjoint sets
- Minimum cut problem: find a graph partition such that the number of edges between the two sets is minimized



Ratio Cut & Normalized Cut



- Minimum cut often returns an imbalanced partition, with one set being a singleton, e.g. node 9
- Change the objective function to consider community size

$$\text{Ratio Cut}(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i, \bar{C}_i)}{|C_i|},$$

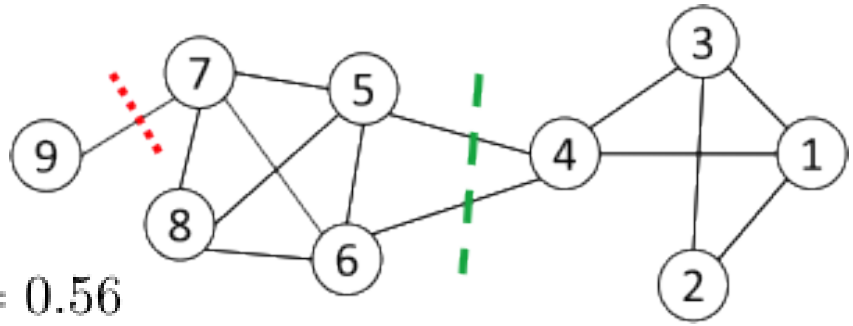
$$\text{Normalized Cut}(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i, \bar{C}_i)}{vol(C_i)}$$

C_i: a community

$|C_i|$: number of nodes in C_i

$vol(C_i)$: sum of degrees in C_i

Ratio Cut & Normalized Cut Example



For partition in red: π_1

$$\text{Ratio Cut}(\pi_1) = \frac{1}{2}\left(\frac{1}{1} + \frac{1}{8}\right) = 9/16 = 0.56$$

$$\text{Normalized Cut}(\pi_1) = \frac{1}{2}\left(\frac{1}{1} + \frac{1}{27}\right) = 14/27 = 0.52$$

For partition in green: π_2

$$\text{Ratio Cut}(\pi_2) = \frac{1}{2}\left(\frac{2}{4} + \frac{2}{5}\right) = 9/20 = 0.45 < \text{Ratio Cut}(\pi_1)$$

$$\text{Normalized Cut}(\pi_2) = \frac{1}{2}\left(\frac{2}{12} + \frac{2}{16}\right) = 7/48 = 0.15 < \text{Normalized Cut}(\pi_1)$$

Both ratio cut and normalized cut prefer a balanced partition

Spectral Clustering

- Both ratio cut and normalized cut can be reformulated as

$$\min_{S \in \{0,1\}^{n \times k}} Tr(S^T \widetilde{L} S)$$

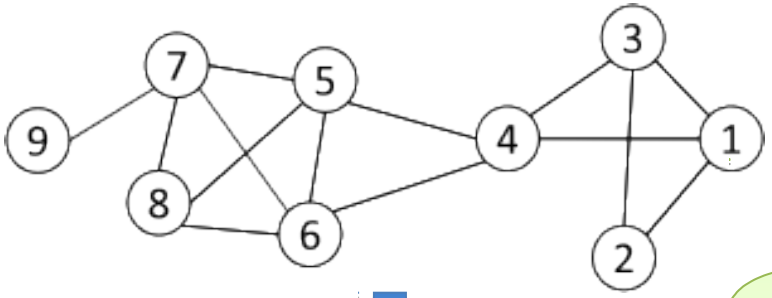
- Where $\widetilde{L} = \begin{cases} D - A & \text{graph Laplacian for ratio cut} \\ I - D^{-1/2} A D^{-1/2} & \text{normalized graph Laplacian} \end{cases}$

$D = diag(d_1, d_2, \cdots, d_n)$ A diagonal matrix of degrees

- Spectral relaxation: $\min_S Tr(S^T \widetilde{L} S) \quad s.t.\ S^T S = I_k$
- Optimal solution: top eigenvectors with the smallest eigenvalues

Reference: http://www.cse.ust.hk/~weikep/notes/clustering.pdf

Spectral Clustering Example



Two communities:
{1, 2, 3, 4} and {5, 6, 7, 8, 9}

The 1st eigenvector means all nodes belong to the same cluster, no use

k-means

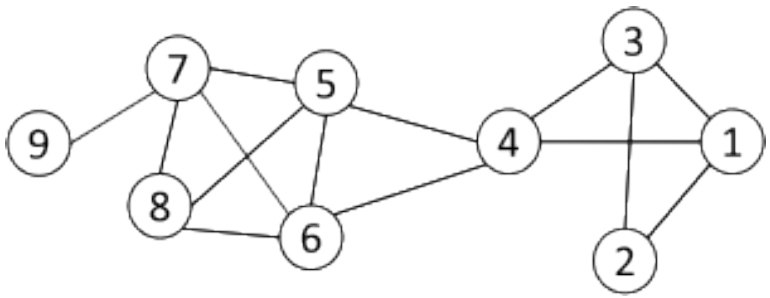
$$D = diag(3, 2, 3, 4, 4, 4, 4, 3, 1)$$

$$\widetilde{L} = D - A = \begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 4 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 4 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix} \Rightarrow S = \begin{bmatrix} 0.33 & -0.38 \\ 0.33 & -0.48 \\ 0.33 & -0.38 \\ 0.33 & -0.12 \\ 0.33 & 0.16 \\ 0.33 & 0.16 \\ 0.33 & 0.30 \\ 0.33 & 0.24 \\ 0.33 & 0.51 \end{bmatrix}$$

Centered matrix

Modularity Maximization

- Modularity measures the strength of a community partition by taking into account the degree distribution
- Given a network with *m* edges, the expected number of edges between two nodes with degrees d_i and d_j is $d_i d_j / 2m$



The expected number of edges between nodes 1 and 2 is
3*2/ (2*14) = 3/14

- Strength of a community: $\sum_{i \in C, j \in C} A_{ij} - d_i d_j / 2m$

Given the degree distribution

- Modularity: $Q = \frac{1}{2m} \sum_{\ell=1}^{k} \sum_{i \in C_\ell, j \in C_\ell} (A_{ij} - d_i d_j / 2m)$
- A larger value indicates a good community structure

Modularity Matrix

Centered matrix

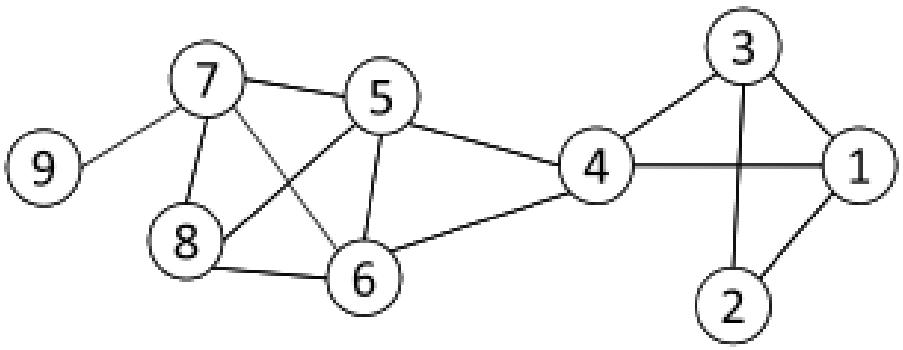
- Modularity matrix: $B = A - \mathbf{d}\mathbf{d}^T/2m \quad (B_{ij} = A_{ij} - d_i d_j/2m)$

- Similar to spectral clustering, Modularity maximization can be reformulated as

$$\max Q = \frac{1}{2m} Tr(S^T B S) \quad s.t.\ S^T S = I_k$$

- Optimal solution: top eigenvectors of the modularity matrix
- Apply k-means to S as a post-processing step to obtain community partition

Modularity Maximization Example



Two Communities:
{1, 2, 3, 4} and {5, 6, 7, 8, 9}

$$B = \begin{bmatrix} -0.32 & 0.79 & 0.68 & 0.57 & -0.43 & -0.43 & -0.43 & -0.32 & -0.11 \\ 0.79 & -0.14 & 0.79 & -0.29 & -0.29 & -0.29 & -0.29 & -0.21 & -0.07 \\ 0.68 & 0.79 & -0.32 & 0.57 & -0.43 & -0.43 & -0.43 & -0.32 & -0.11 \\ 0.57 & -0.29 & 0.57 & -0.57 & 0.43 & 0.43 & -0.57 & -0.43 & -0.14 \\ -0.43 & -0.29 & -0.43 & 0.43 & -0.57 & 0.43 & 0.43 & 0.57 & -0.14 \\ -0.43 & -0.29 & -0.43 & 0.43 & 0.43 & -0.57 & 0.43 & 0.57 & -0.14 \\ -0.43 & -0.29 & -0.43 & -0.57 & 0.43 & 0.43 & -0.57 & 0.57 & 0.86 \\ -0.32 & -0.21 & -0.32 & -0.43 & 0.57 & 0.57 & 0.57 & -0.32 & -0.11 \\ -0.11 & -0.07 & -0.11 & -0.14 & -0.14 & -0.14 & 0.86 & -0.11 & -0.04 \end{bmatrix}$$

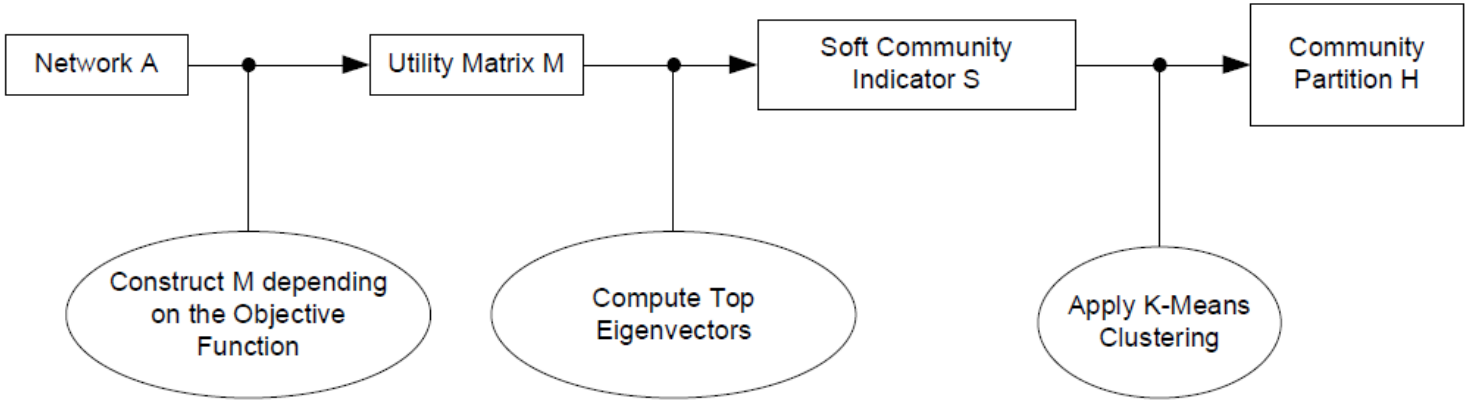
Modularity Matrix

k-means

0.4384	-0.2709
0.3809	0.2671
0.4384	-0.2709
0.1716	0.6063
-0.2861	-0.3487
-0.2861	-0.3487
-0.3754	0.3355
-0.3421	0.1855
-0.1396	-0.1552

A Unified View for Community Partition

- Latent space models, block models, spectral clustering, and modularity maximization can be unified as



$$\text{Utility Matrix } M = \begin{cases} \text{modified proximity matrix } \widetilde{P} & \textit{if latent space models} \\ \text{adjacency matrix } A & \textit{if block models} \\ \text{graph Laplacian } \widetilde{L} & \textit{if spectral clustering} \\ \text{modularity maximization } B & \textit{if modularity maximization} \end{cases}$$

Reference: http://www.cse.ust.hk/~weikep/notes/Script_community_detection.m

Hierarchy-Centric Community Detection

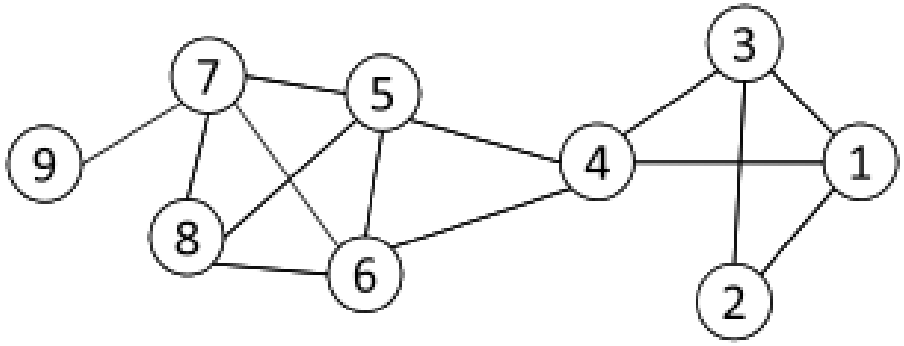
- Goal: build a hierarchical structure of communities based on network topology
- Allow the analysis of a network at different resolutions
- Representative approaches:
  - Divisive Hierarchical Clustering (top-down)
  - Agglomerative Hierarchical clustering (bottom-up)

Divisive Hierarchical Clustering

- Divisive clustering
  - Partition nodes into several sets
  - Each set is further divided into smaller ones
  - Network-centric partition can be applied for the partition
- One particular example: recursively remove the “weakest” tie
  - Find the edge with the least strength
  - Remove the edge and update the corresponding strength of each edge
- Recursively apply the above two steps until a network is decomposed into desired number of connected components.
- Each component forms a community

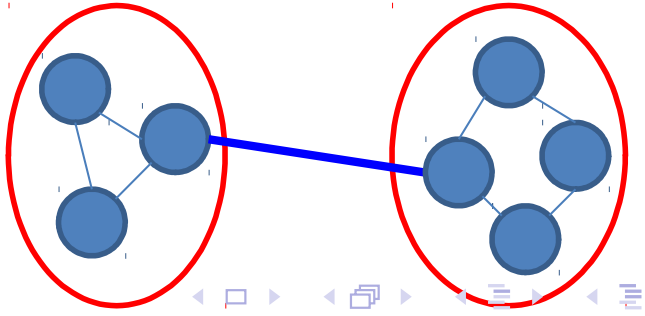
Edge Betweenness

- The strength of a tie can be measured by edge betweenness
- Edge betweenness: the number of shortest paths that pass along with the edge

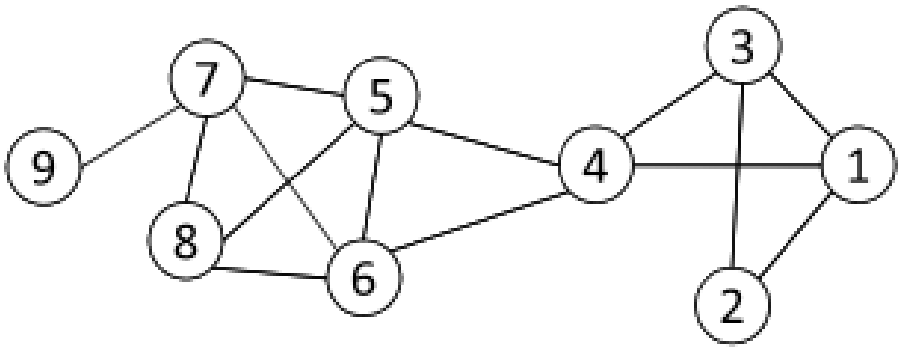


The edge betweenness of e(1, 2) is 4 (=6/2 + 1), as all the shortest paths from 2 to {4, 5, 6, 7, 8, 9} have to either pass e(1, 2) or e(2, 3), and e(1,2) is the shortest path between 1 and 2

- The edge with higher betweenness tends to be the bridge between two communities.



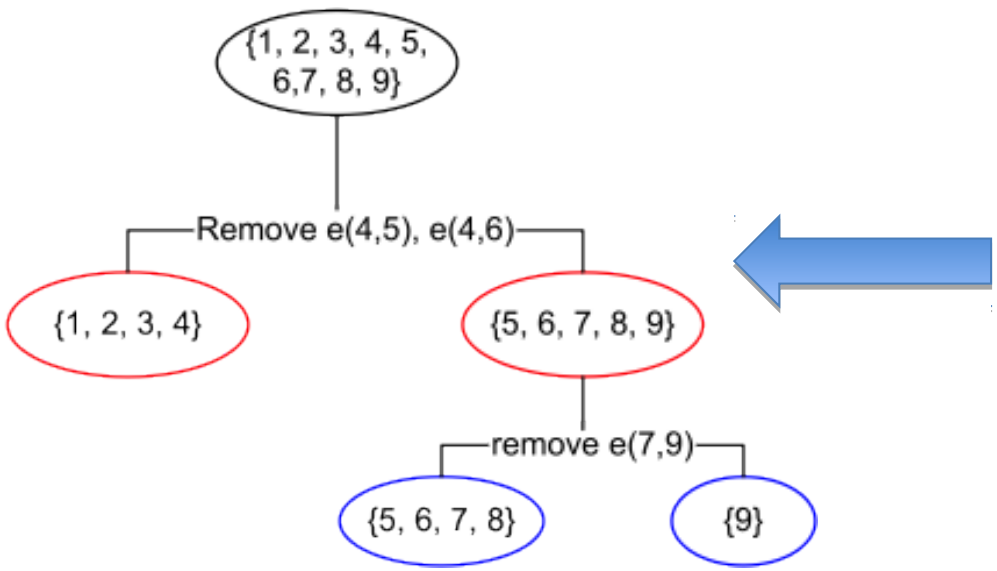
Divisive clustering based on edge betweenness



Initial betweenness value

Table 3.3: Edge Betweenness

	1	2	3	4	5	6	7	8	9
1	0	4	1	9	0	0	0	0	0
2	4	0	4	0	0	0	0	0	0
3	1	4	0	9	0	0	0	0	0
4	9	0	9	0	10	10	0	0	0
5	0	0	0	10	0	1	6	3	0
6	0	0	0	10	1	0	6	3	0
7	0	0	0	0	6	6	0	2	8
8	0	0	0	0	3	3	2	0	0
9	0	0	0	0	0	0	8	0	0



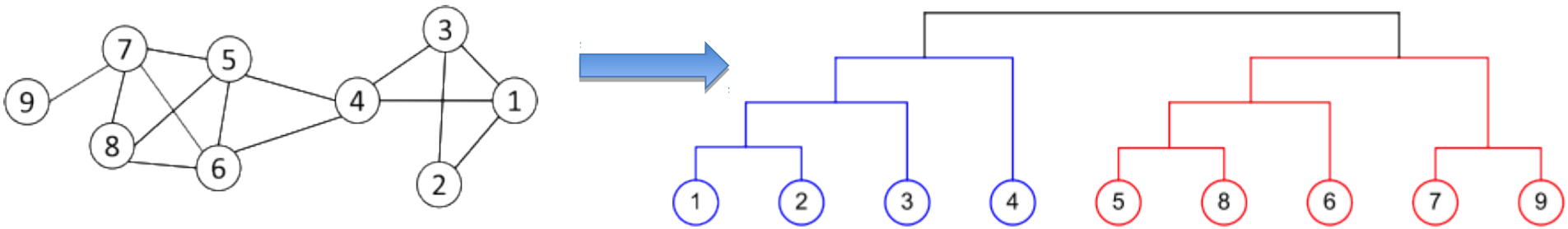
After remove e(4,5), the betweenness of e(4, 6) becomes 20, which is the highest;

After remove e(4,6), the edge e(7,9) has the highest betweenness value 4, and should be removed.

Idea: progressively removing edges with the highest betweenness

Agglomerative Hierarchical Clustering

- Initialize each node as a community
- Merge communities successively into larger communities following a certain criterion
  - E.g., based on modularity increase



Dendrogram according to Agglomerative Clustering based on Modularity

Summary of Community Detection

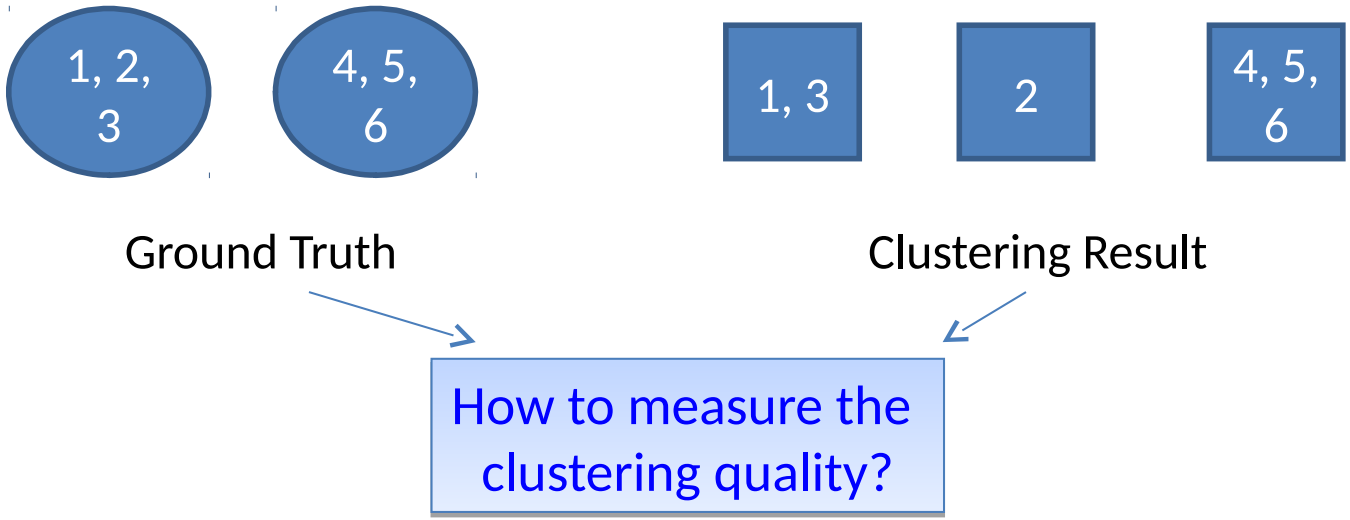
- Node-Centric Community Detection
  - *cliques, k-cliques, k-clubs*
- Group-Centric Community Detection
  - *quasi-cliques*
- Network-Centric Community Detection
  - *Clustering based on vertex similarity*
  - *Latent space models, block models, spectral clustering, modularity maximization*
- Hierarchy-Centric Community Detection
  - *Divisive clustering*
  - *Agglomerative clustering*

COMMUNITY EVALUATION

Evaluating Community Detection (1)

- For groups with clear definitions
  - E.g., Cliques, k-cliques, k-clubs, quasi-cliques
  - Verify whether extracted communities satisfy the definition
- For networks with ground truth information
  - Normalized mutual information
  - Accuracy of pairwise community memberships

Measuring a Clustering Result



- The number of communities after grouping can be different from the ground truth
- No clear community correspondence between clustering result and the ground truth
- Normalized Mutual Information can be used

Normalized Mutual Information

- Entropy: the information contained in a distribution

$$H(X) = \sum_{x \in X} p(x) \log p(x)$$

- Mutual Information: the shared information between two distributions

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x) p_2(y)} \right)$$

- Normalized Mutual Information (between 0 and 1)

$$NMI(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}$$ JMLR03, Strehl or $$NMI(X;Y) = \frac{2I(X;Y)}{H(X)+H(Y)}$$ KDD04, Dhilon

- Consider a partition as a distribution (probability of one node falling into one community), we can compute the matching between the clustering result and the ground truth

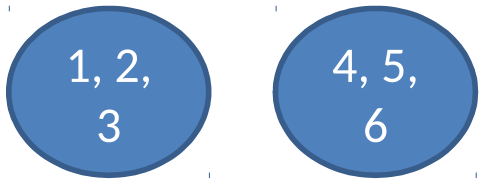
Accuracy of Pairwise Community Memberships

- Consider all the possible pairs of nodes and check whether they reside in the same community
- An error occurs *if*
  - Two nodes belonging to the same community are assigned to different communities after clustering
  - Two nodes belonging to different communities are assigned to the same community
- Construct a contingency table or confusion matrix

		Ground Truth	
		$C(v_i) = C(v_j)$	$C(v_i) \neq C(v_j)$
Clustering Result	$C(v_i) = C(v_j)$	a	b
	$C(v_i) \neq C(v_j)$	c	d

$$accuracy = \frac{a+d}{a+b+c+d} = \frac{a+d}{n(n-1)/2}$$

Accuracy Example



Ground Truth

Clustering Result

		Ground Truth	
		$C(v_i) = C(v_j)$	$C(v_i) != C(v_j)$
Clustering Result	$C(v_i) = C(v_j)$	4	0
	$C(v_i) != C(v_j)$	2	9

Accuracy = (4+9)/ (4+2+9+0) = 13/15

Evaluation using Semantics

- For networks with semantics
  - Networks come with semantic or attribute information of nodes or connections
  - Human subjects can verify whether the extracted communities are coherent
- Evaluation is qualitative
- It is also intuitive and helps understand a community

An *animal* community

A *health* community

Evaluation without Ground Truth

- For networks without ground truth or semantic information
- This is the most common situation
- An option is to resort to cross-validation
  - Extract communities from a (training) network
  - Evaluate the quality of the community structure on a network constructed from a different date or based on a related type of interaction
- Quantitative evaluation functions
  - Modularity (M.Newman. Modularity and community structure in networks. PNAS 06.)
  - Link prediction (the predicted network is compared with the true network)

Today's Biz

1. Quick Review
2. Reminders
3. Social Networks Topics
4. **Betweenness and Graph Partitioning**
5. Label Propagation

Betweenness and Graph Partitioning

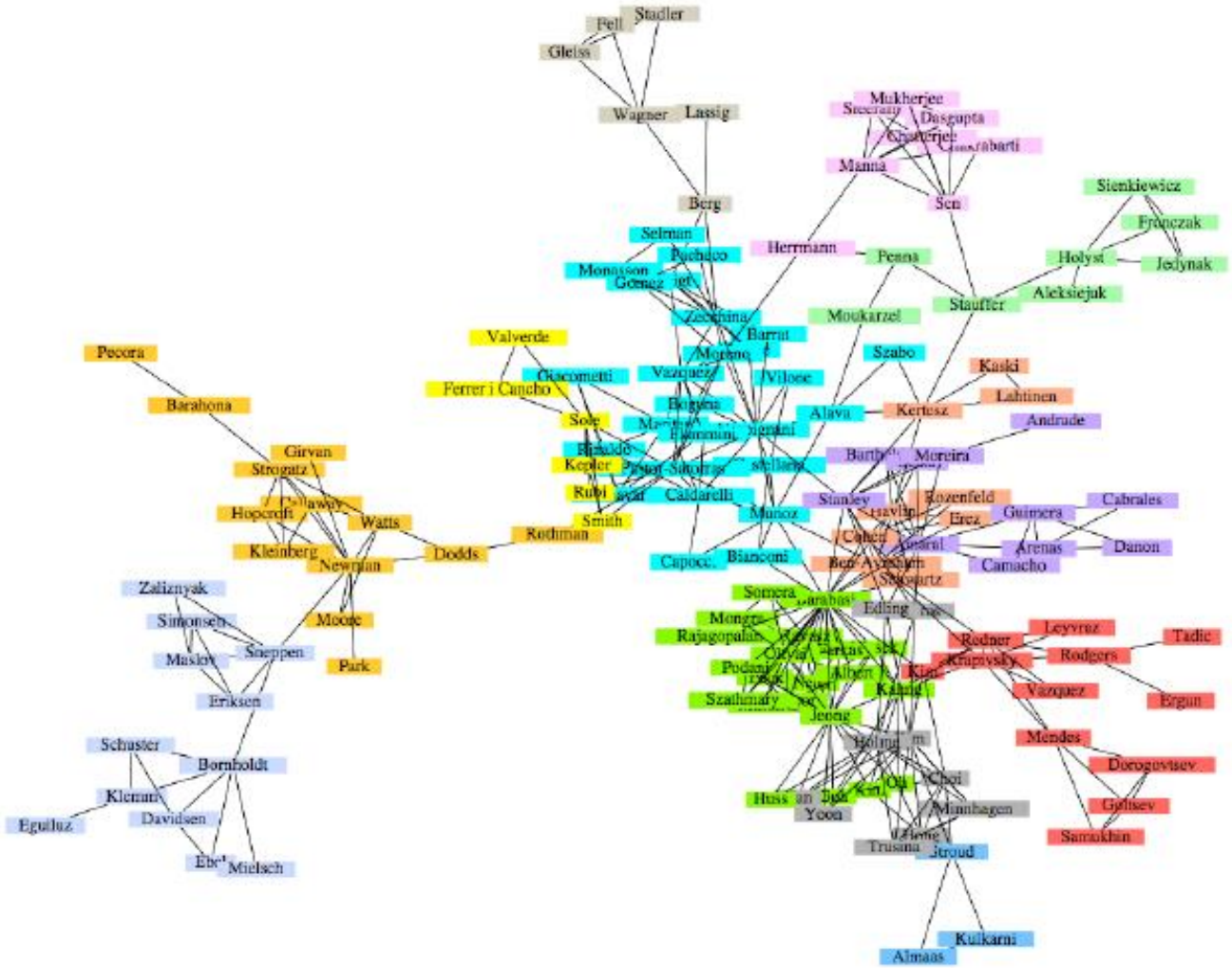
Slides from Alexandros Nanopoulos, Stiftung Universität Hildesheim

Betweenness Measures and Graph Partitioning

Objectives

- Define densely connected regions of a network
- Graph partitioning
  - Algorithm to identify densely connected regions
  - breaking a network into a set of nodes densely connected with each other with edges
  - having sparser interconnections between the regions

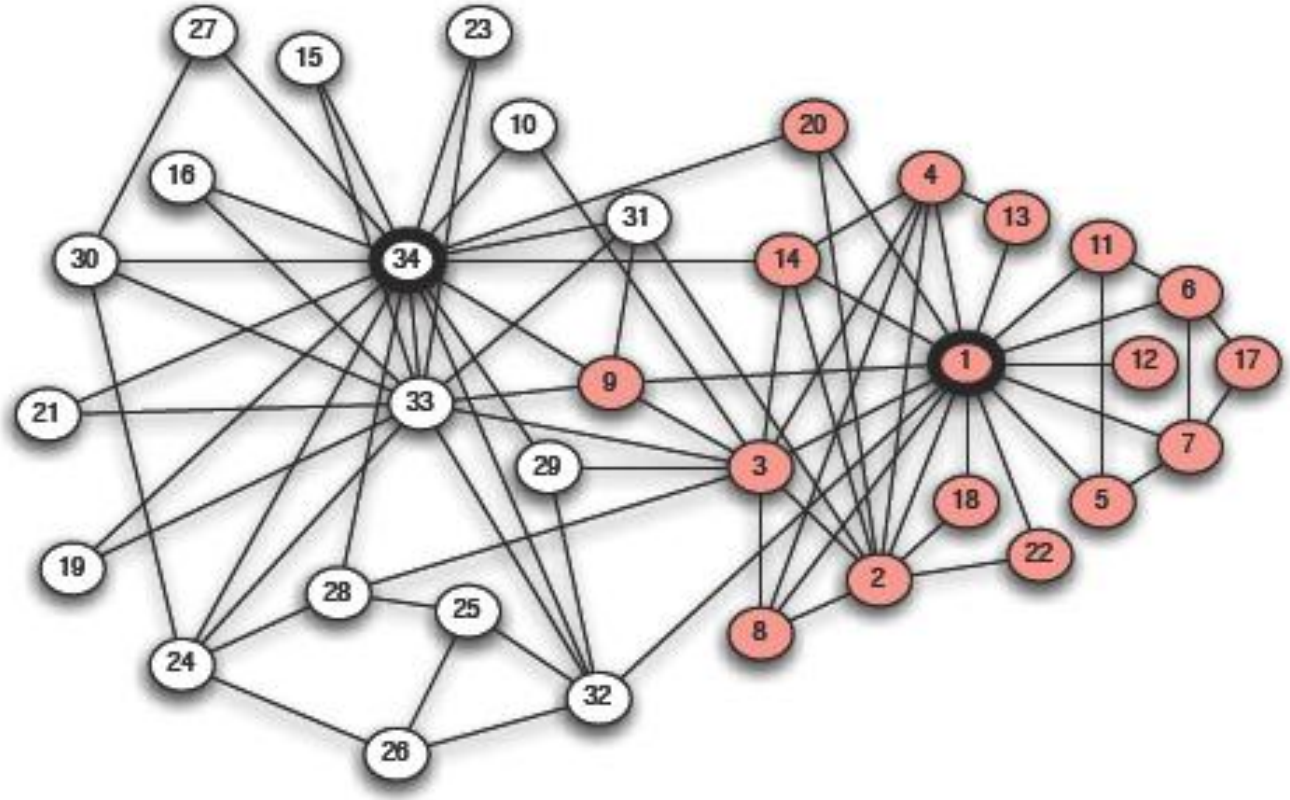
Graph partitioning example



A co-authorships network among a set of physicists

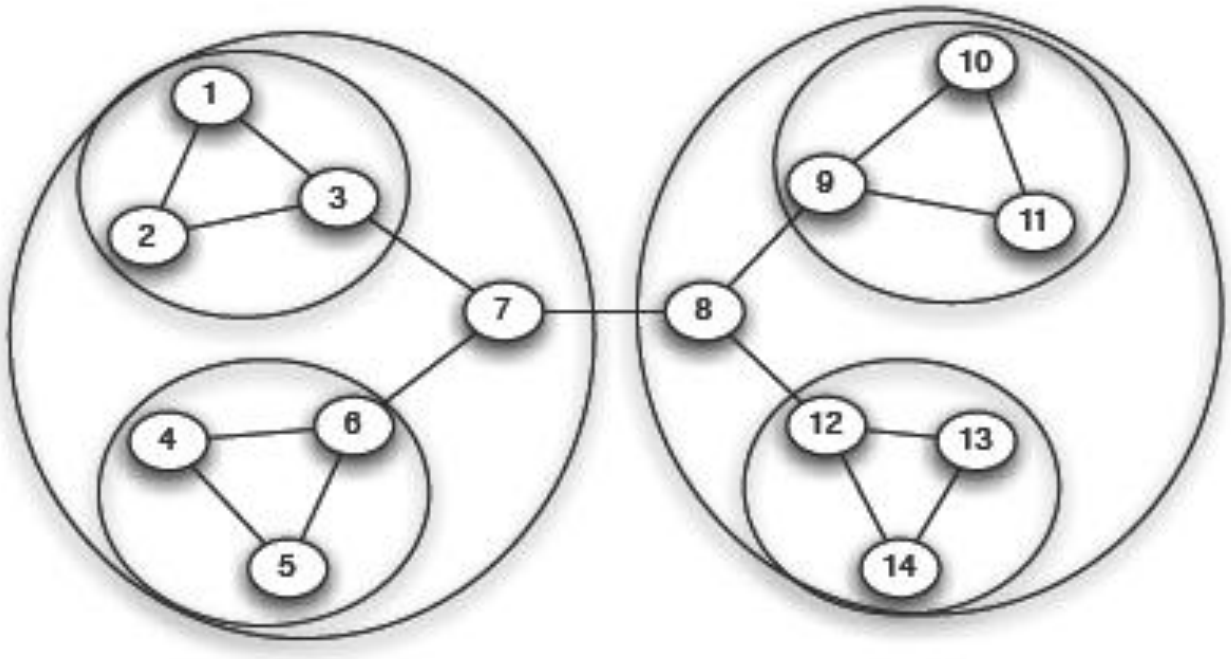
Graph partitioning example

the 2 conflicting groups are still heavily interconnected
Need to look how edges between groups occur at lower “density” than edges within the groups



social network of a karate club

Nesting of regions



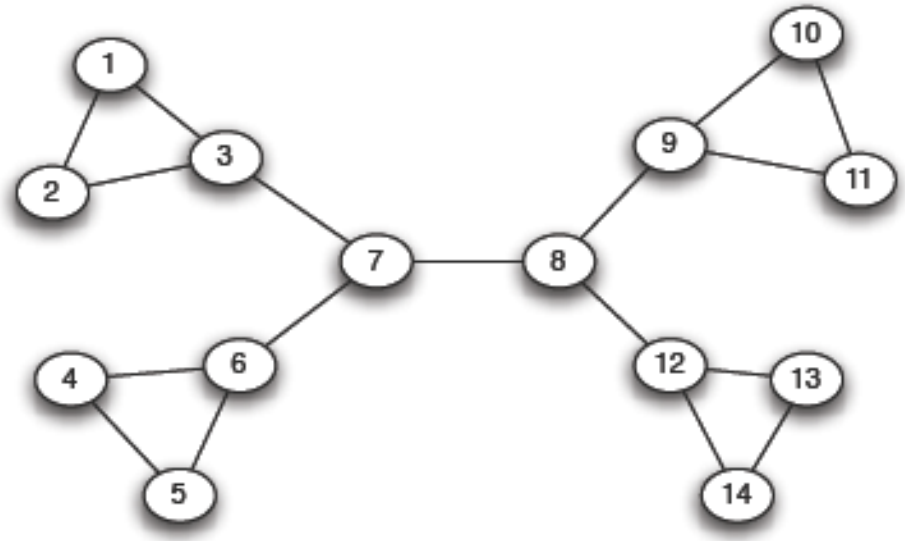
Larger regions containing several smaller

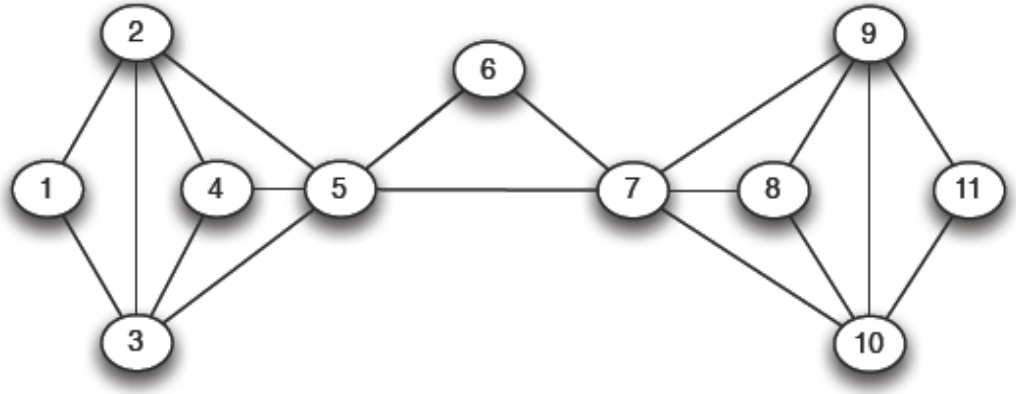
Divisive methods: breaking first at the 7-8 edge, and then the nodes into nodes 7 and 8

Agglomerative methods: merge the 4 triangles and then pairs of triangles (via nodes 7 and 8)

Divisive removal of Bridges

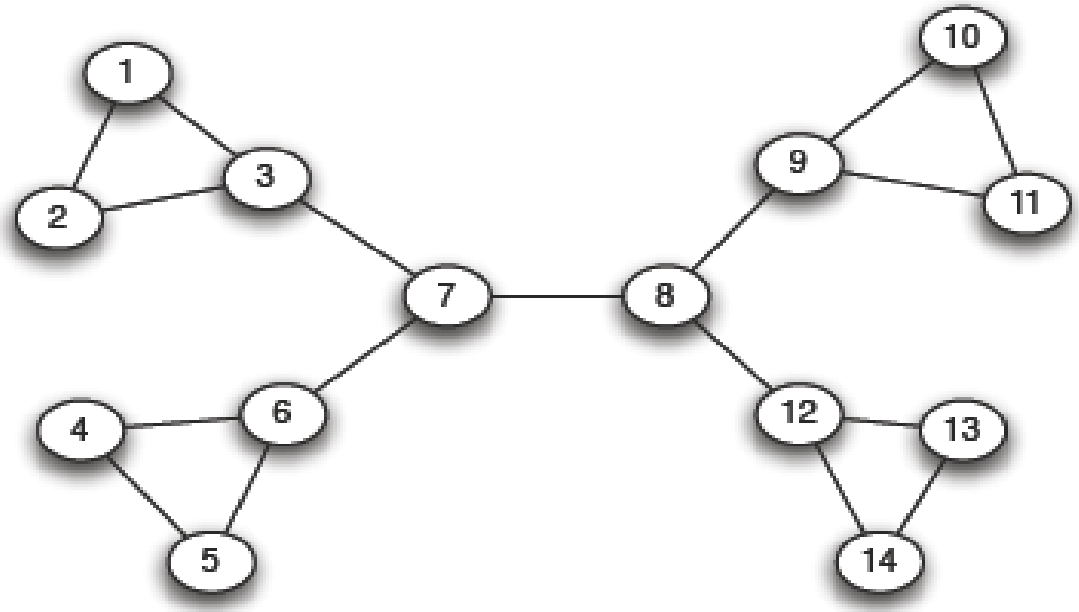
- Simple idea:
  - remove bridges and local bridges
- Problems:
  - which when several? (ex: in fig up 5 bridges)
  - what if none (ex: in fig down nodes 1-5 and 7-11)





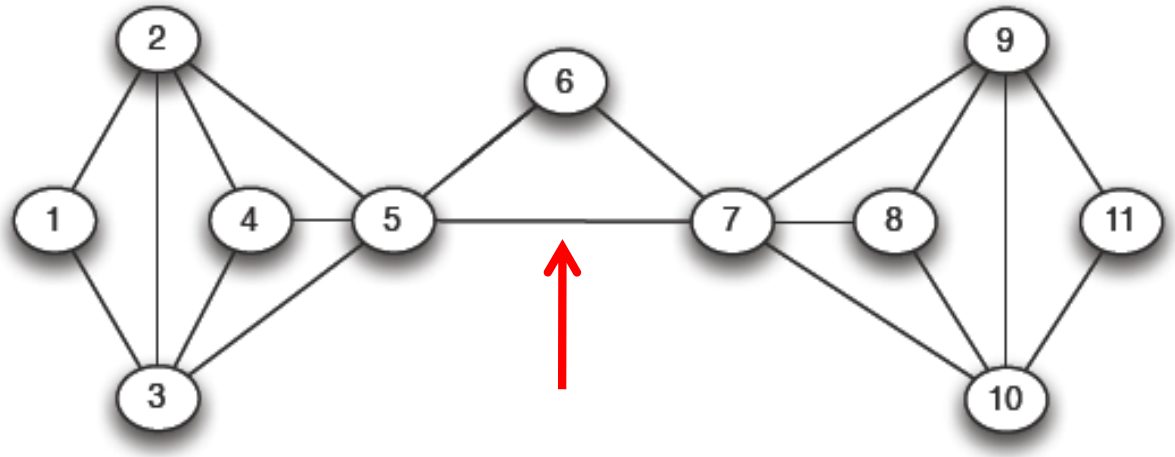
The role of Bridges

- Q: What bridges and local bridges are doing?
- A: They form part of the shortest path between pairs of nodes in different parts of the network



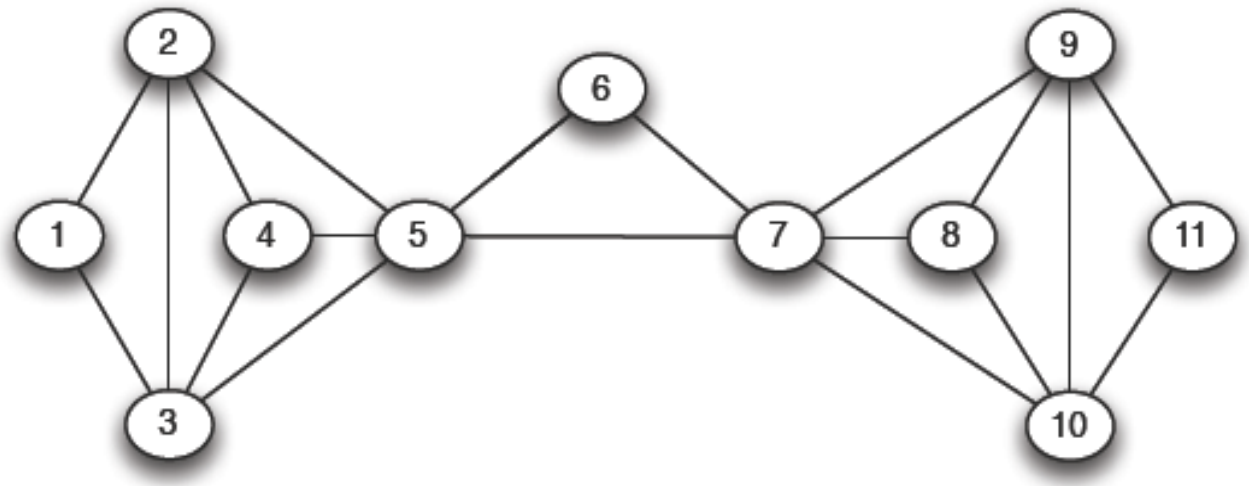
Generalize the Role of Bridges

- Look for the edges that carry the most of "traffic" in a network
  - without the edge, paths between many pairs of nodes may have to be "re-routed" a longer way
  - edges to link different densely-connected regions
  - good candidates for removal in a divisive method
  - generalize the (local) bridges



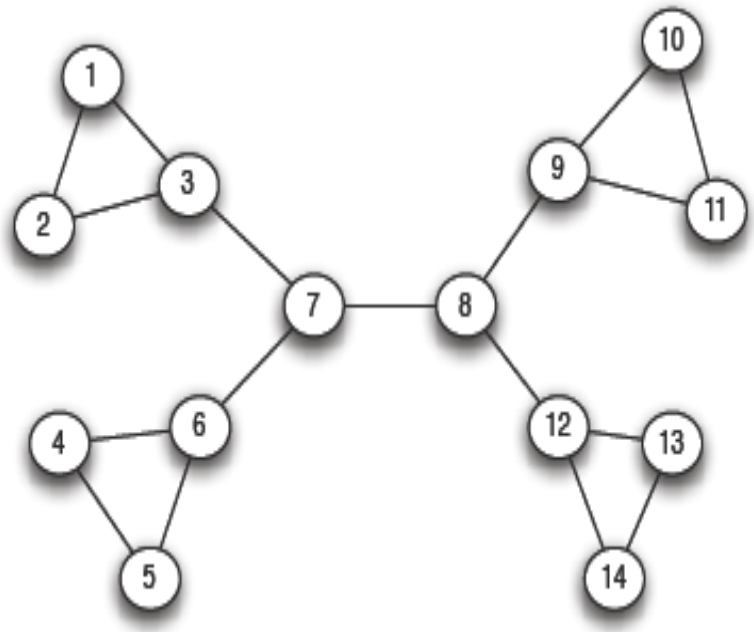
Traffic in a Network

- For nodes A and B connected by a path assume 1 unit of "flow"
  - (If A and B in different connected components, flow = 0)
- Divide flow evenly along all possible shortest paths from A to B
  - if k shortest paths from A and B, then 1/k units of flow pass along each
- Ex: 2 shortest paths from 1 to 5, each with 1/2 units of flow



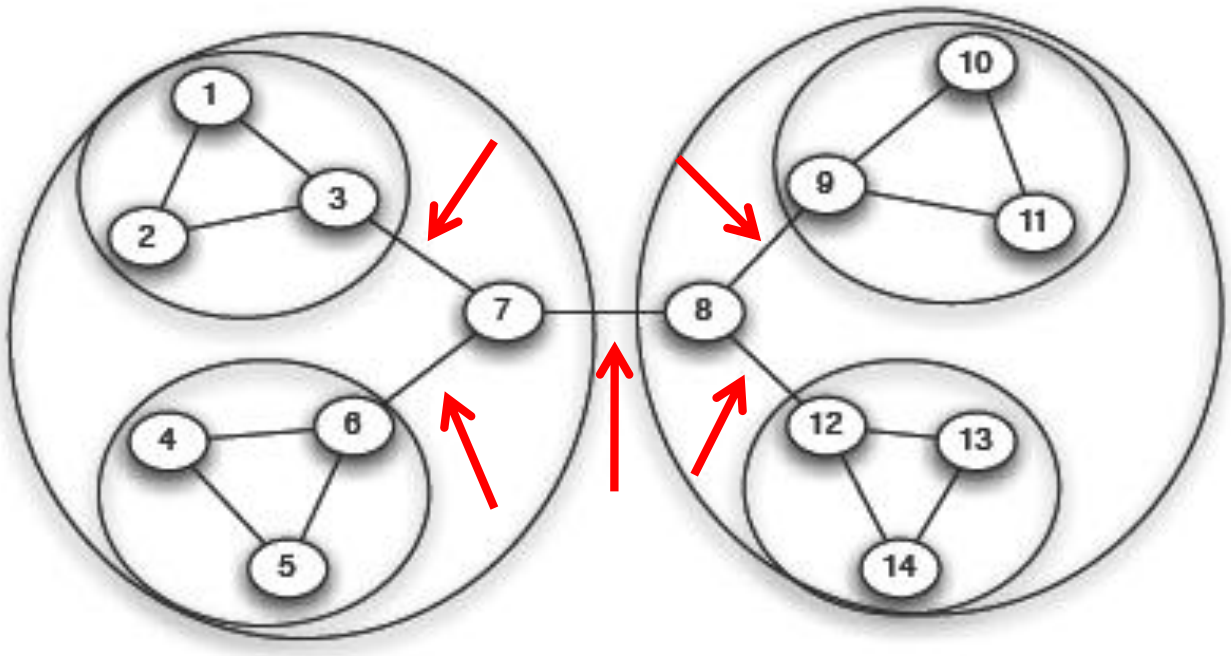
Edge Betweenness

- Betweenness of an edge: the total amount of flow it carries
  - counting flow between all pairs of nodes using this edge
- Ex:
  - Edge **7-8**: each pair of nodes between [1-7] and [8-14]; each pair with traffic = 1; total 7 x 7 = 49
  - Edge **3-7**: each pair of nodes between [1-3] and [4-14]; each pair with traffic = 1; total 3 x 11 = 33
  - Edge **1-3**: each pair of nodes between [1] and [3-14] (not node 2); each pair with traffic = 1; total 1 x 12 = 12
    - similar for edges 2-3, 4-6, 5-6, 9-10, 9-11, 12-13, and 12-14
  - Edge **1-2**: each pair of nodes between [1] and [2] (no other); each pair with traffic = 1; total 1 x 1 = 1
    - similar for edges 4-5, 10-11, and 13-14



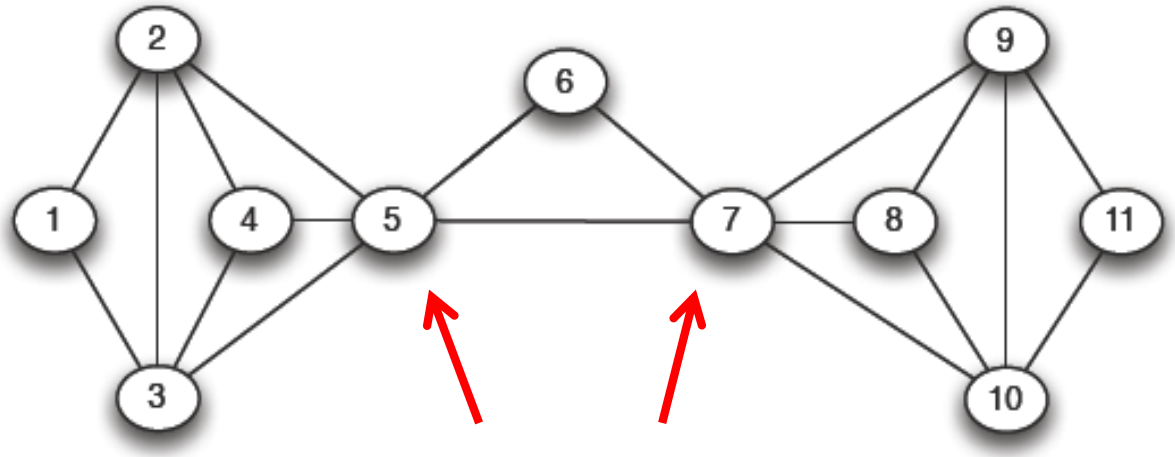
Betweenness for Partitioning

- Divisive: remove edges with high betweenness



Betweenness of Nodes

- Betweenness of a node: total amount of flow that it carries, when a unit of flow between each pair of nodes is divided up evenly over shortest paths (same as for edges)
  - nodes of high betweenness occupy critical roles in the network ("gatekeepers")



Girvan-Newman Partitioning Alg.

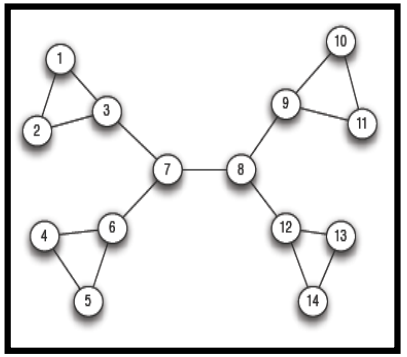
Successively Deleting Edges of High Betweenness

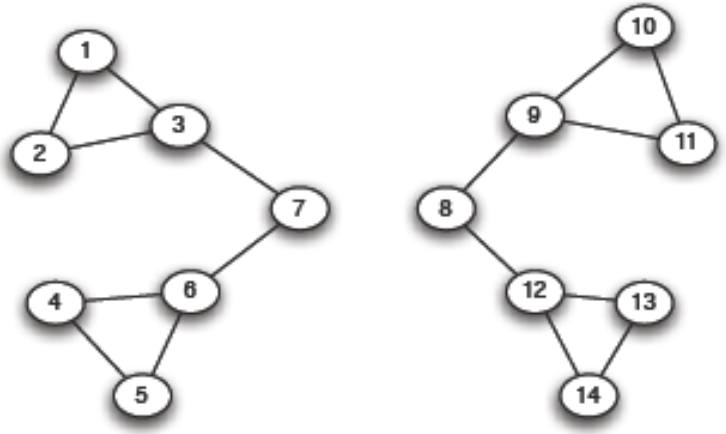
(1) Find the edge of highest betweenness — or multiple edges of highest betweenness, if there is a tie — and remove these edges from the graph. This may cause the graph to separate into multiple components. If so, this is the first level of regions in the partitioning of the graph.

(2) Now recalculate all betweennesses, and again remove the edge or edges of highest betweenness. This may break some of the existing components into smaller components; if so, these are regions nested within the larger regions.

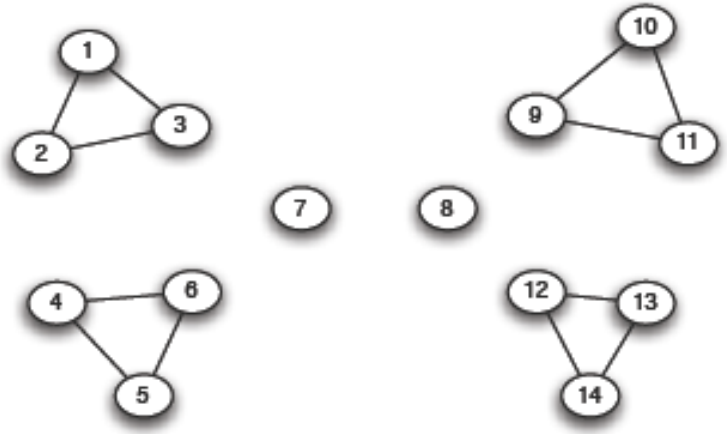
(...) Proceed in this way as long as edges remain in graph, in each step recalculating all betweennesses and removing the edge or edges of highest betweenness.

Example 1

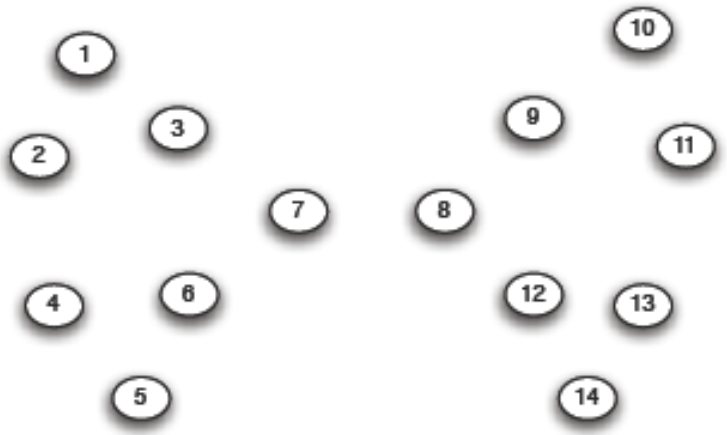




(a) *Step 1*

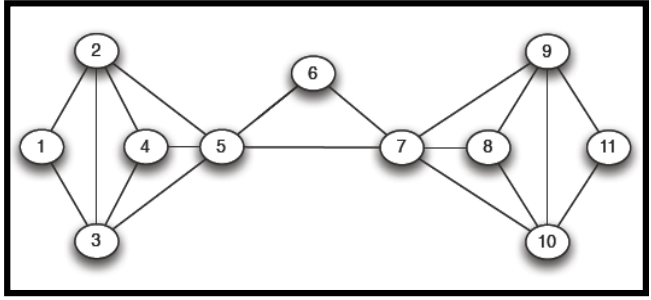


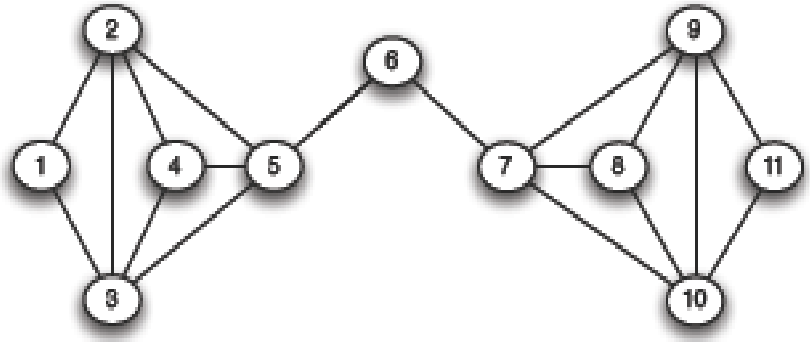
(b) *Step 2*



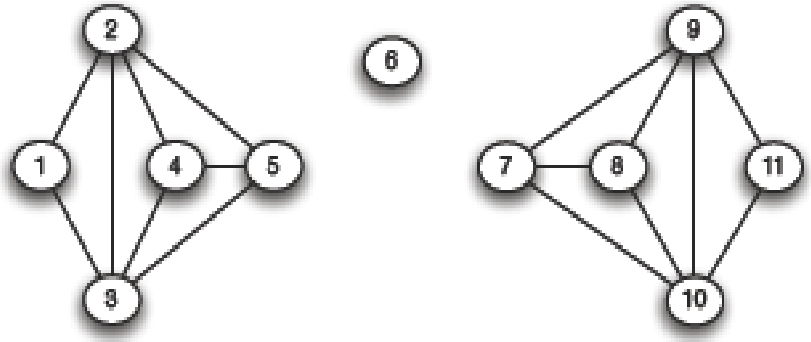
(c) *Step 3*

Example 2

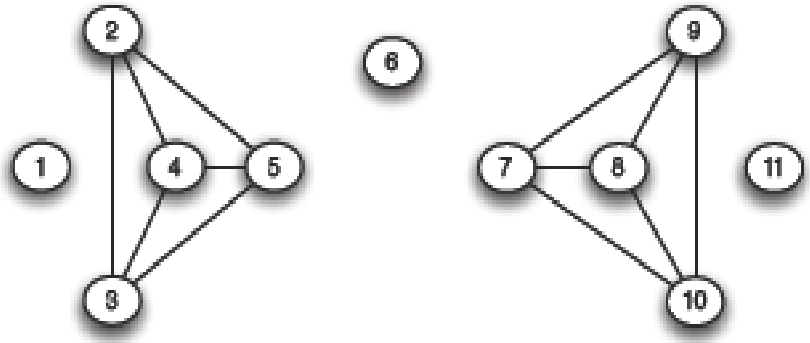




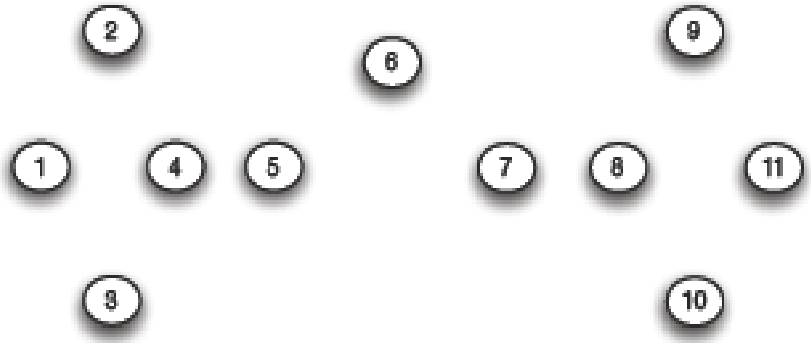
(a) *Step 1*



(b) *Step 2*



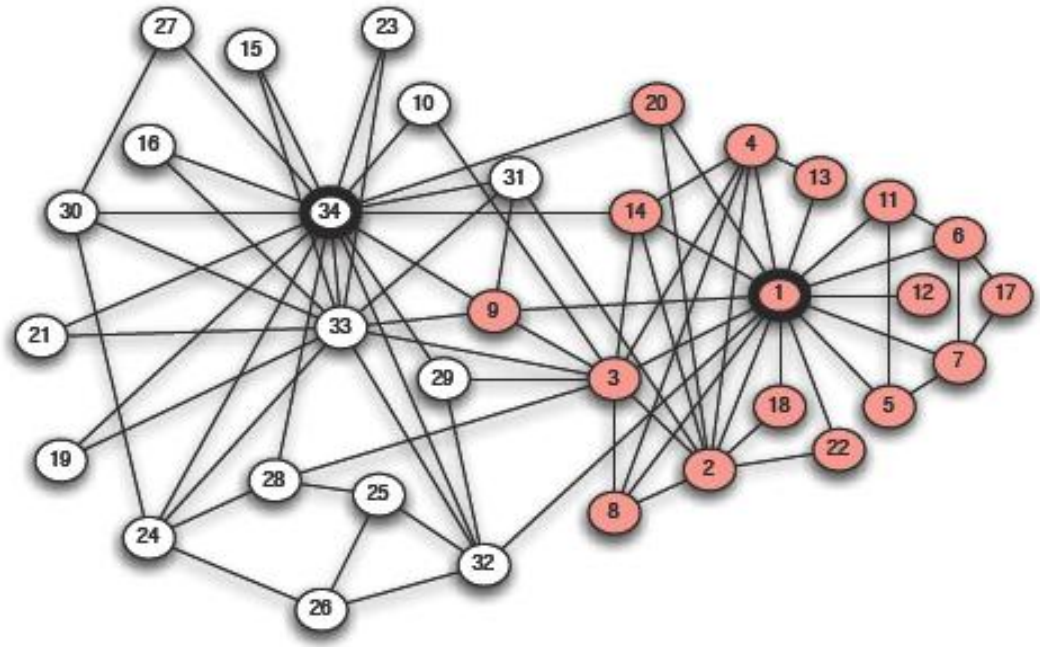
(c) *Step 3*



(d) *Step 4*

Example 3

- Girvan-Newman partitions correctly
  - exception: node 9 assigned to region of 34 (left part)
  - at the time of conflict, node 9 was completing a four-year quest to obtain a black belt, which he could only do with the instructor (node 1)



Partitioning large Social Networks

- In real social network data, partitioning is easier when network is small (at most a few hundred nodes)
- In large networks, nodes become much more "inextricable"
- Open research problem

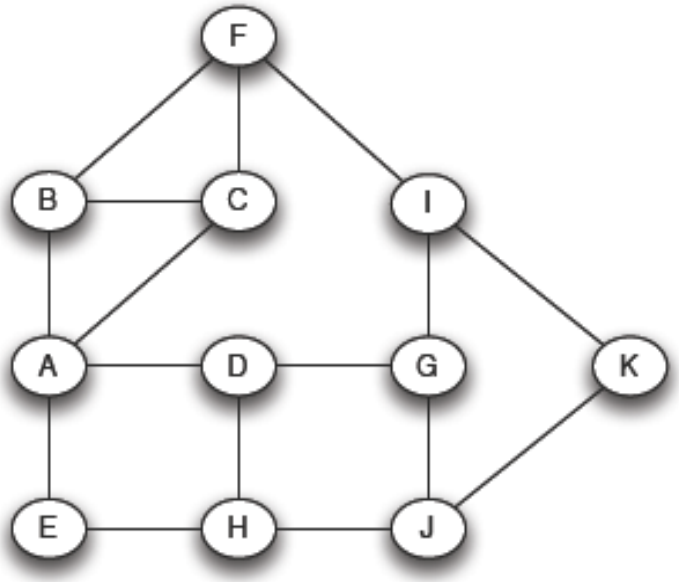
Computing Betweenness Values

- According to definition: consider all the shortest paths between all pairs of nodes
- Computationally expensive
- How to compute betweenness without listing out all such shortest paths?
- Method based on BFS

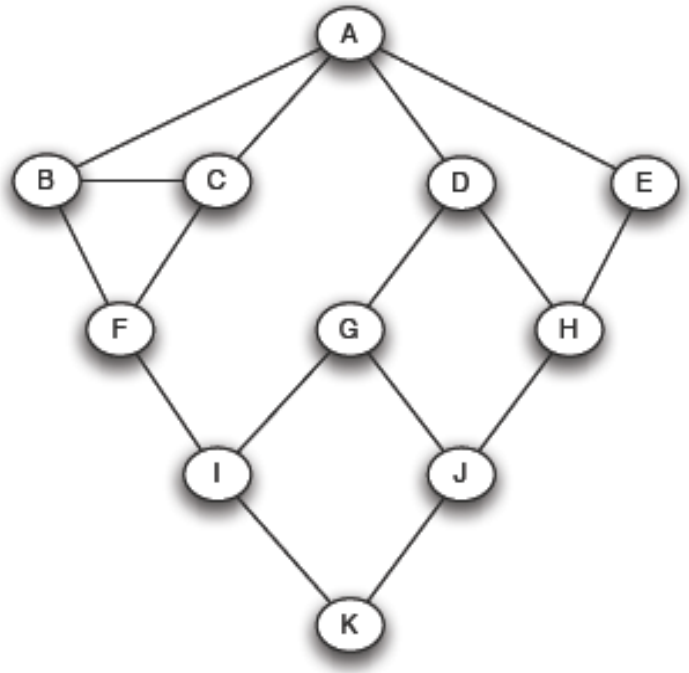
Method

- For each node A:
  1. BFS starting at A
  2. Count the number of shortest paths from A to each other node
  3. Based on this number, determine the amount of flow from A to all other nodes

Step 1: Example



(a) *A sample network*



(b) *Breadth-first search starting at node A*

Layer 1

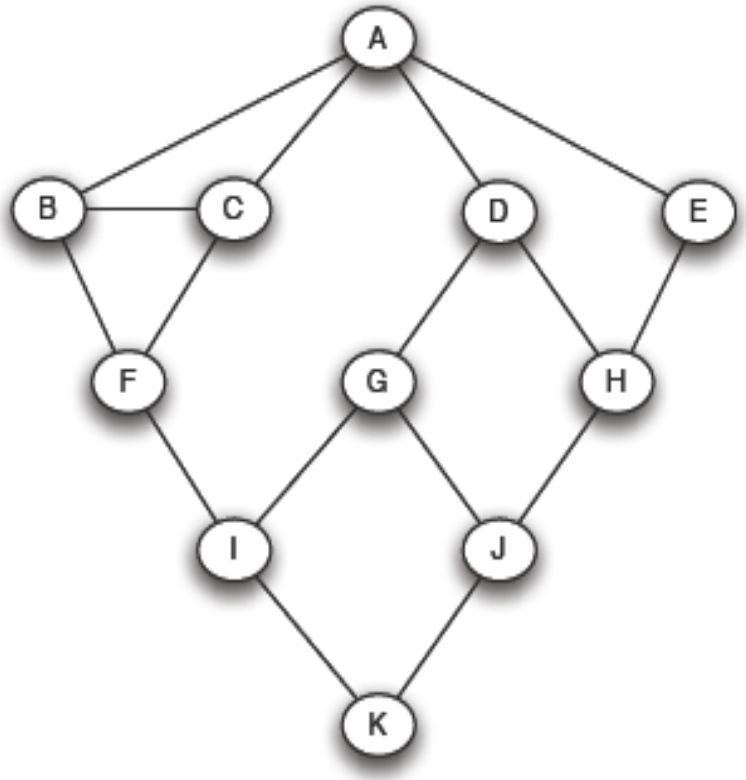
Layer 2

Layer 3

Layer 4

Step 2: Example

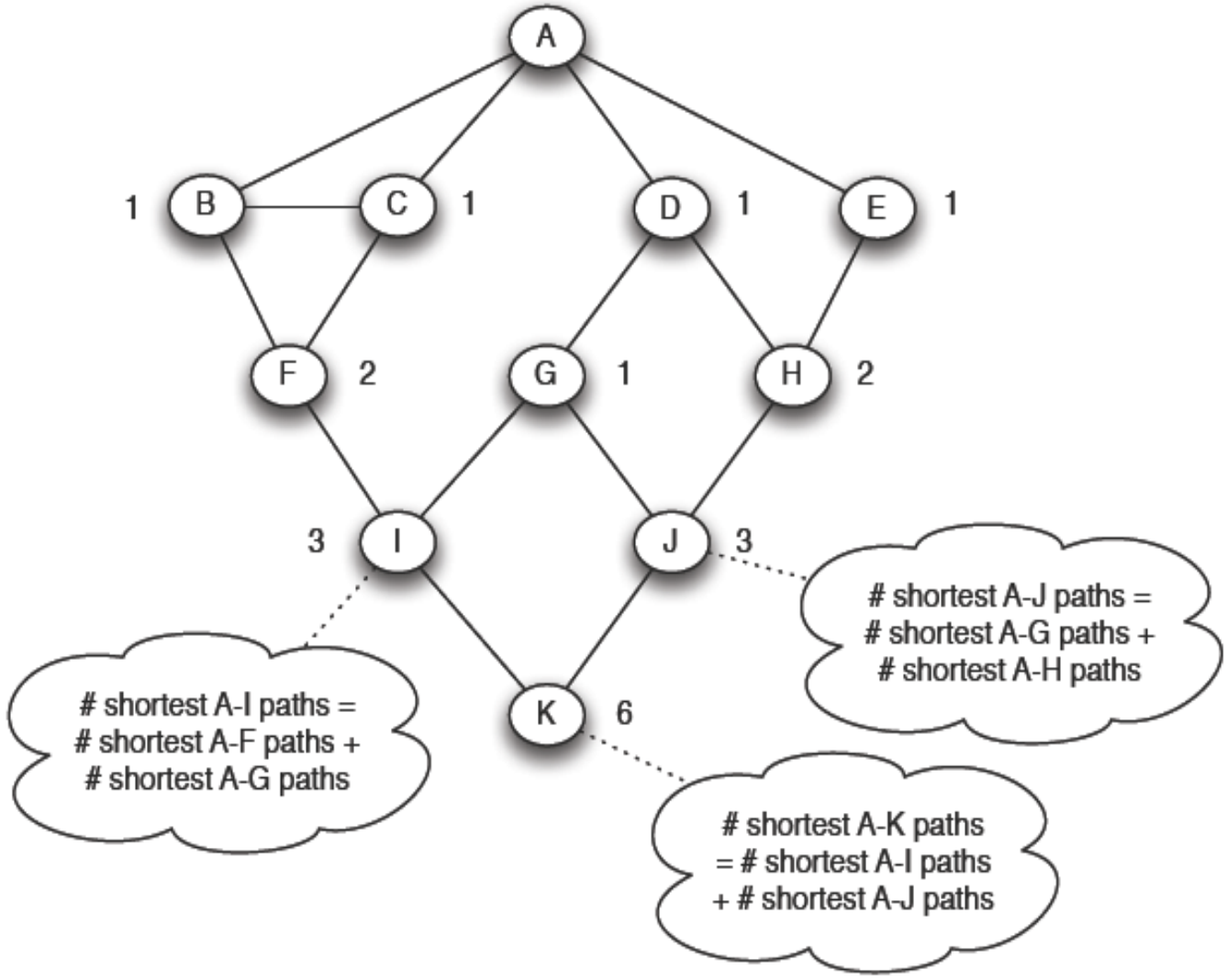
- F and G are above I
- All shortest-paths from A to I must take their last step through either F or G
- To be a shortest path to I, a path must first be a shortest path to one of F or G, and then take this last step to I
- The number of shortest paths from A to I is the number of shortest paths from A to F, plus
- the number of shortest paths from A to G



a node *X is above a node Y in the breadth-first search if X is in the layer immediately preceding Y , and X has an edge to Y*

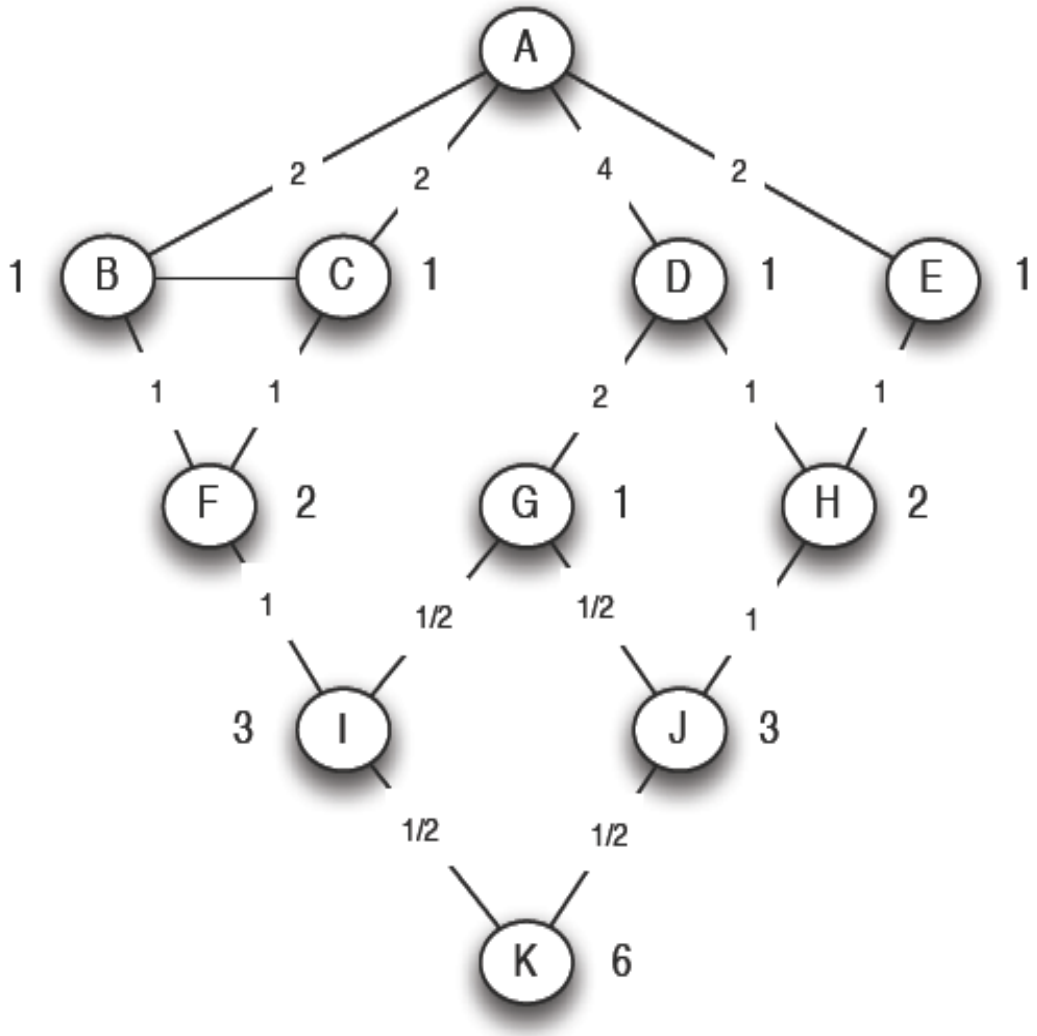
Step 2: Example

- Each node in the first layer has only 1 shortest path from A
- The number of shortest paths to each other node is the sum of the number of shortest paths to all nodes directly above it
- Avoid finding the shortest paths themselves!



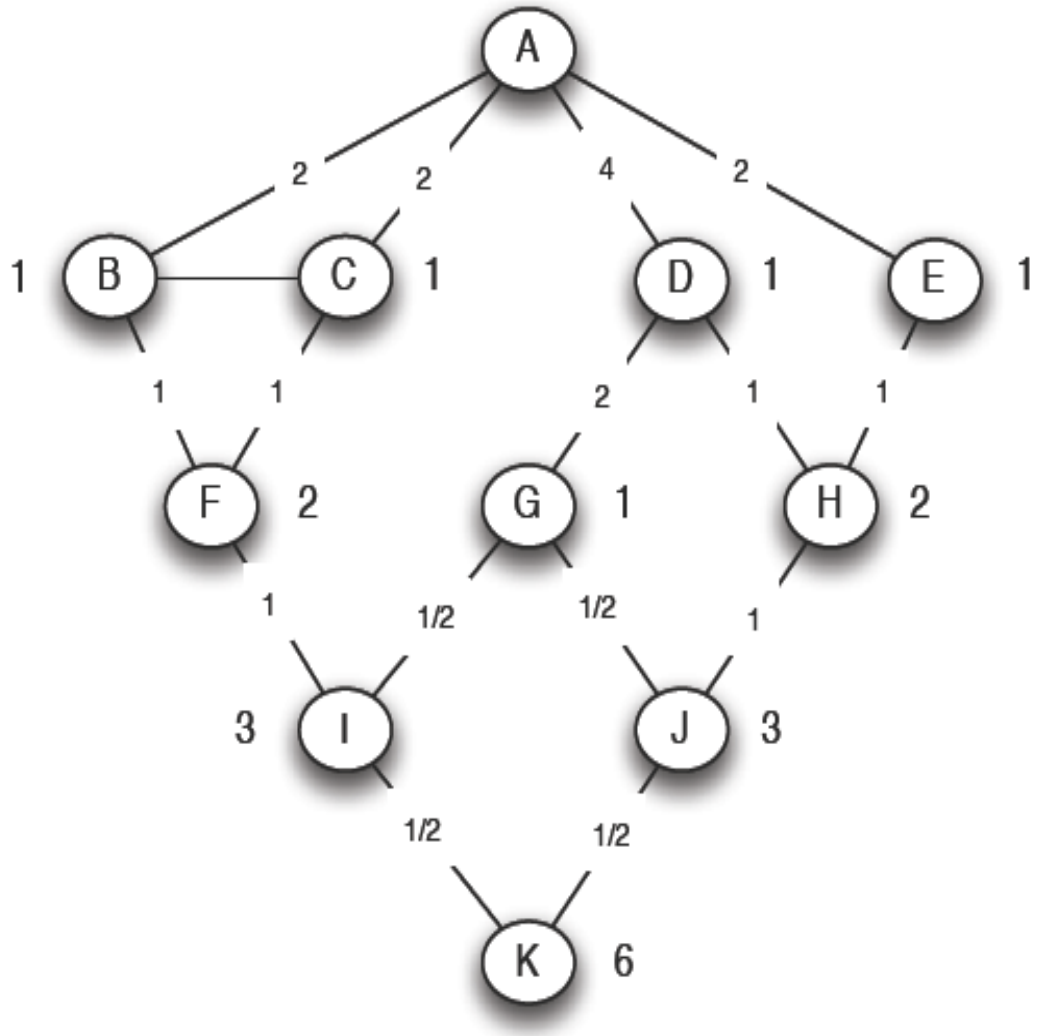
Step 3: Example

- How the flow from A to all other nodes spreads out across the edges?
- Working up from the lowest layers
  - **1** unit of flow arrives at K and an equal number of the shortest paths from A to K come through nodes I and J => **1/2**-unit of flow on each of these edges
  - **3/2** units of flow arriving at I (**1** unit destined for I plus the **1/2** passing through to K). These **3/2** units are divided in proportion **2** to **1** between F and G => **1** unit to F and **1/2** to G



Step 3: Method

- Move bottom up
- At each node X
  - add up all flow arriving from edges directly below X, plus 1 for the flow destined for X itself
  - Divide this up over the edges leading upward from X, in proportion to the number of shortest paths coming through each

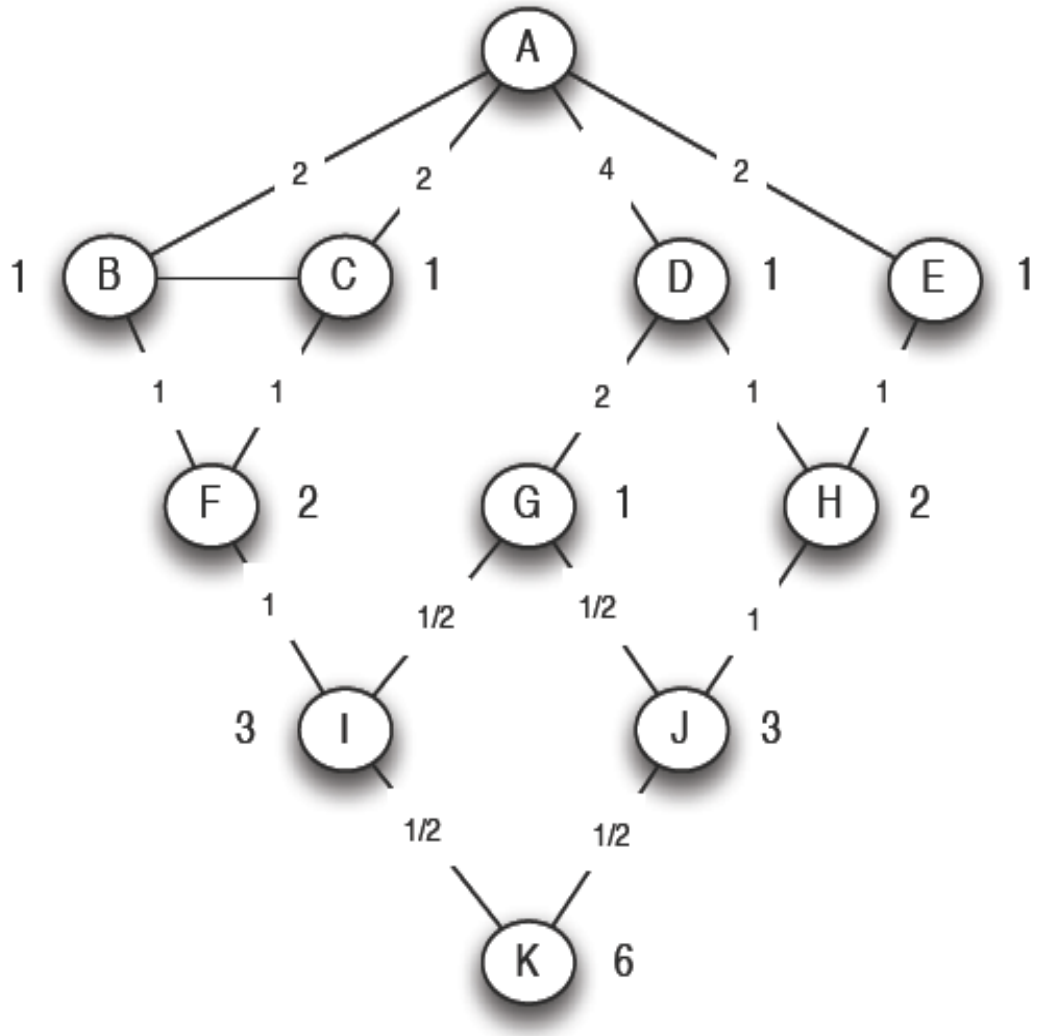


Summary

- Build one BFS structures for each node
- Determine flow values for each edge using the previous procedure and (3 steps)
- Sum up the flow values of each edge in all BFS structures to get its betweenness value
- Notice: we are counting the flow between each pair of nodes X and Y twice (once when BFS from X and once when BFS from Y)
  - at the end we divide everything by two
- Usie these betweenness values to identify the edges of highest betweenness for purposes of removing them in the Girvan-Newman method

Computing Betweennes of Nodes

- Same procedure
- Compute the outgoing (upwords) sum of flow from node
  - or downards sum + 1

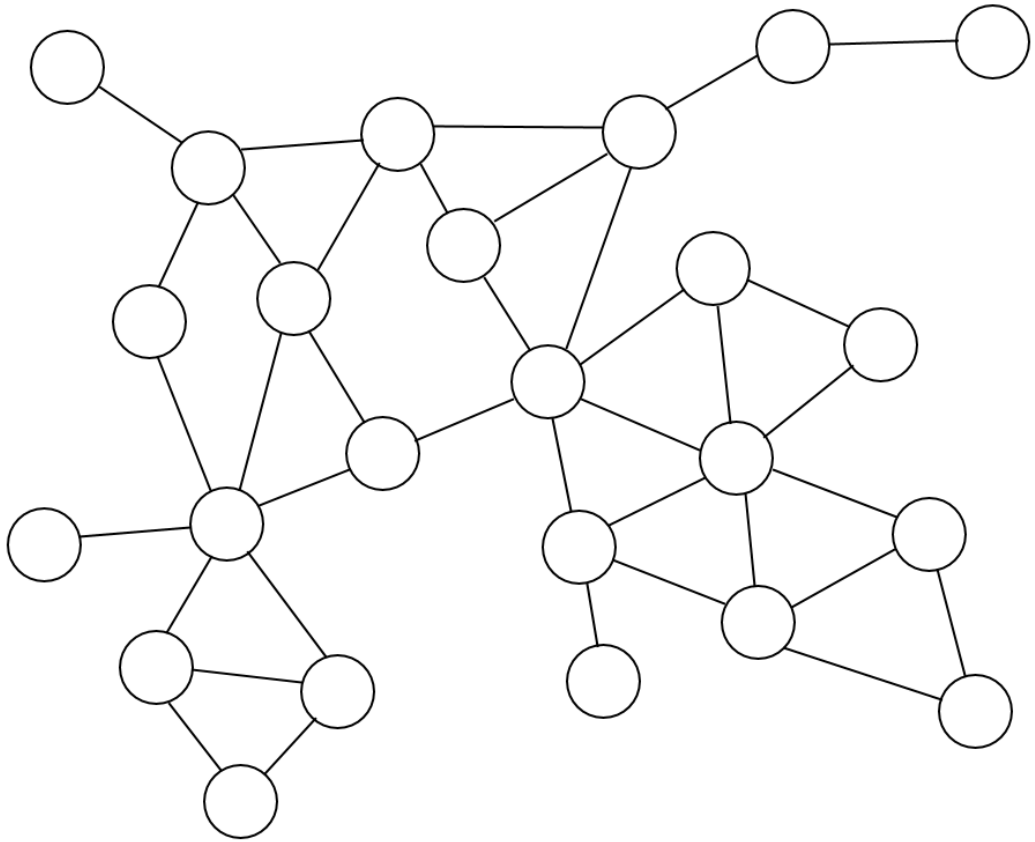


Today's Biz

1. Quick Review
2. Reminders
3. Social Networks Topics
4. Betweenness and Graph Partitioning
5. **Label Propagation**

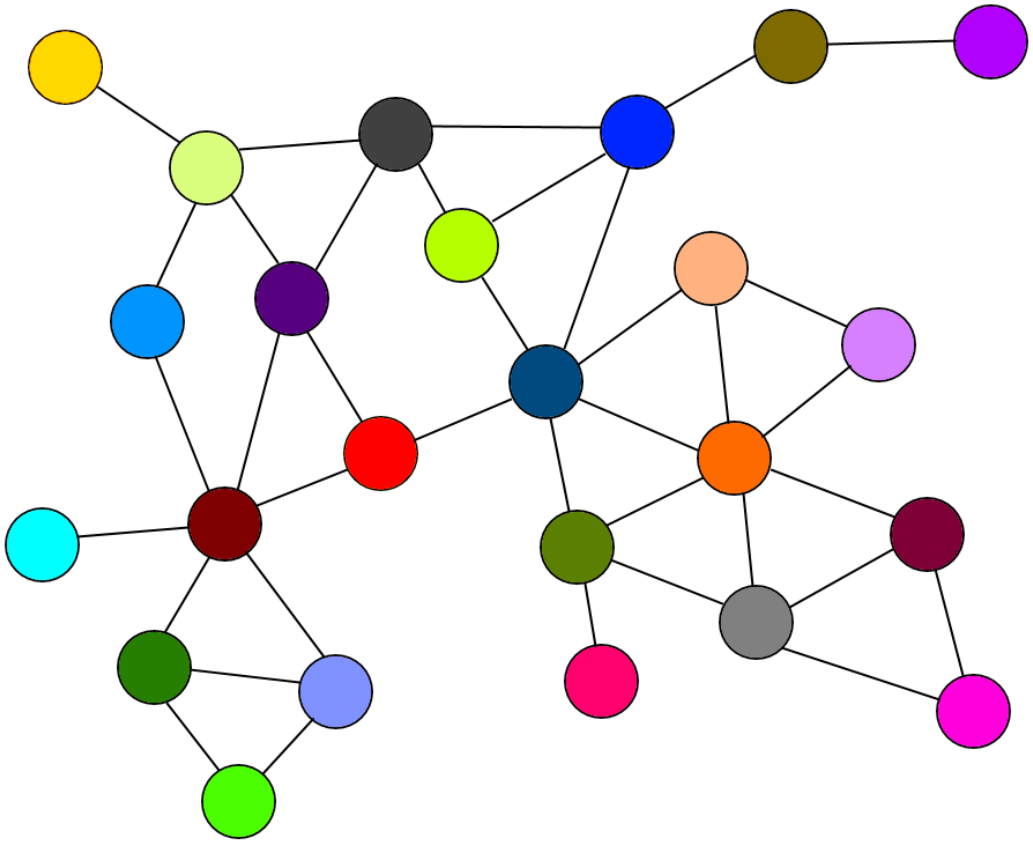
Label Propagation
Algorithm progression

- Randomly label with n labels



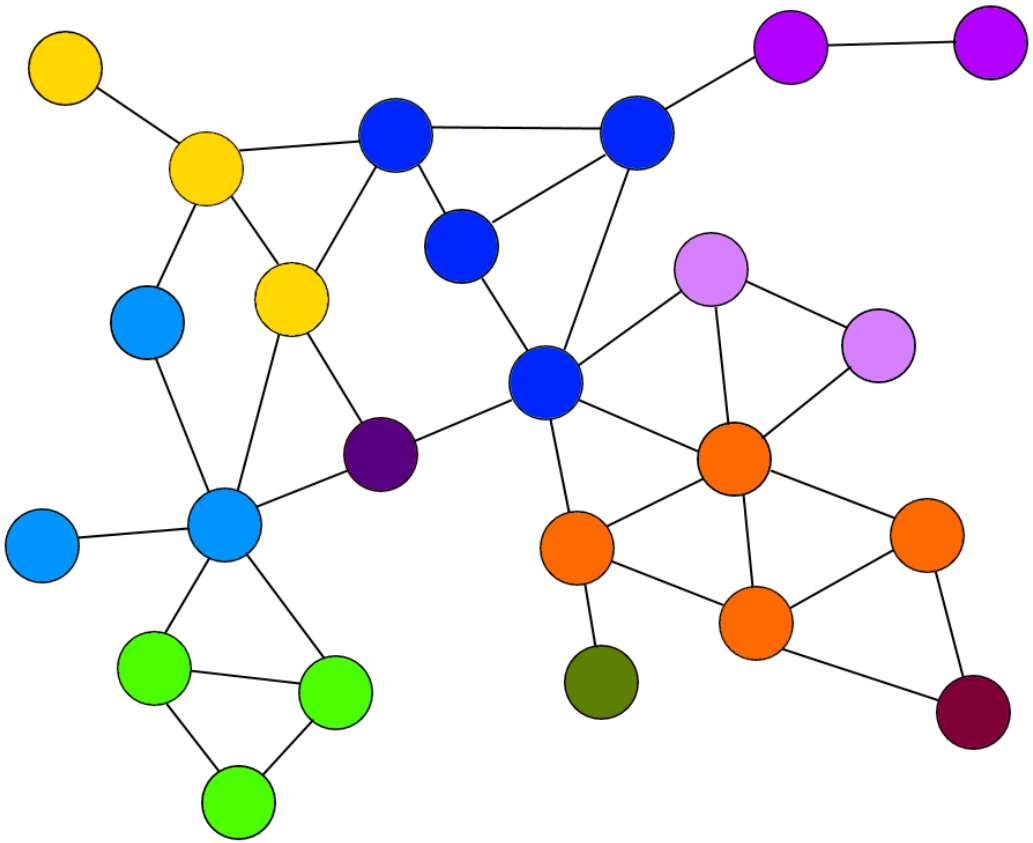
Label Propagation
Algorithm progression

- Randomly label with n labels



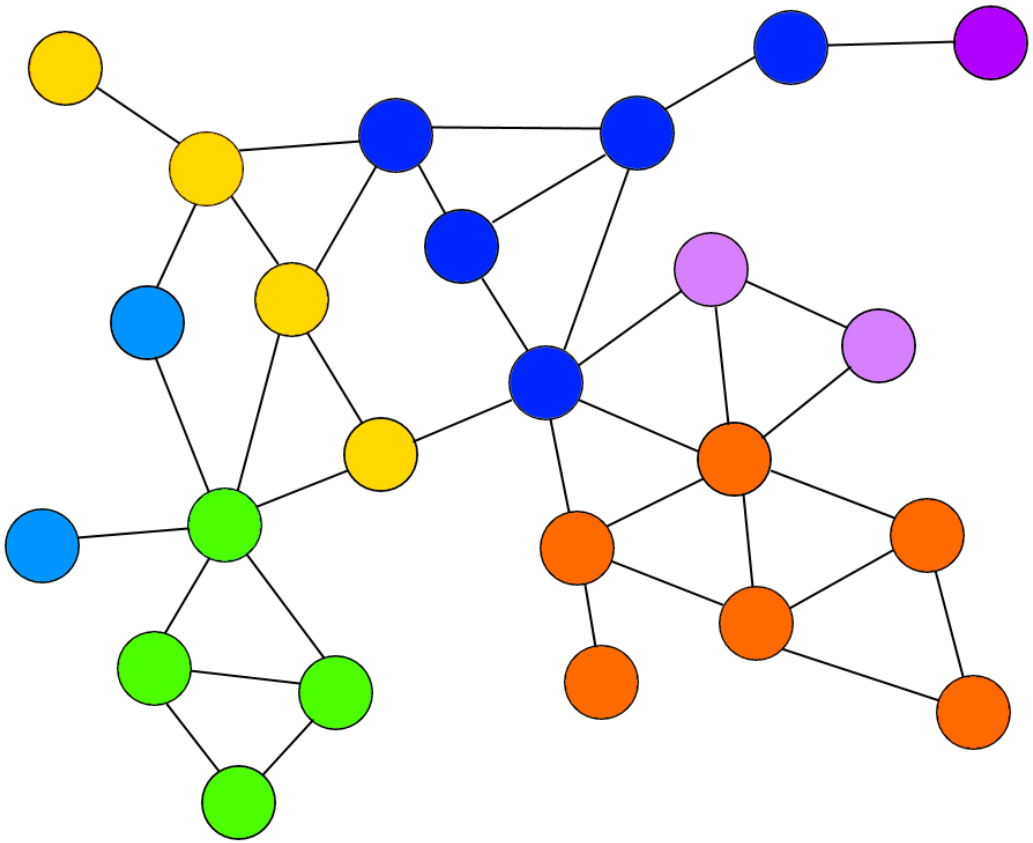
Label Propagation
Algorithm progression

- Randomly label with n labels
- Iteratively update each v with max per-label count over neighbors, ties broken randomly



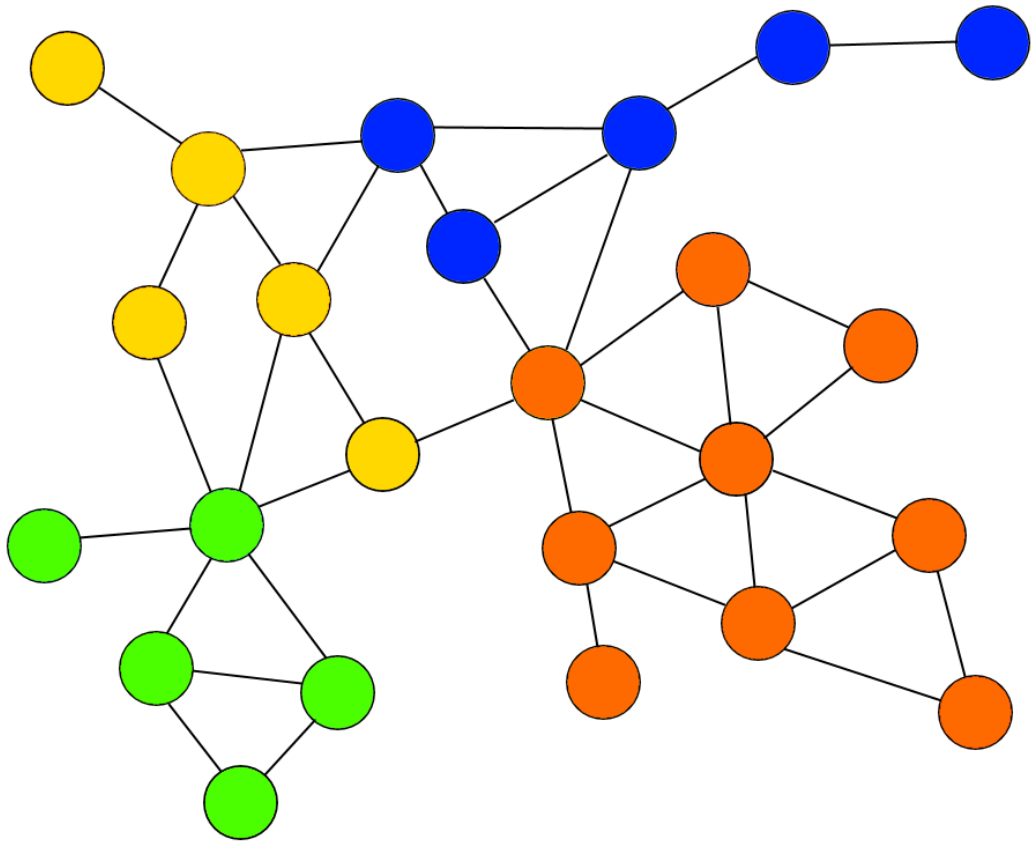
Label Propagation
Algorithm progression

- Randomly label with n labels
- Iteratively update each v with max per-label count over neighbors, ties broken randomly



Label Propagation
Algorithm progression

- Randomly label with n labels
- Iteratively update each v with max per-label count over neighbors, ties broken randomly
- Algorithm completes when no new updates possible



Label Propagation
Overview and observations

- **Label propagation**: initialize a graph with n labels, iteratively assign to each vertex the maximal per-label count over all neighbors to generate clusters, ties broken randomly (Raghavan et al. 2007)
  - Clustering algorithm - dense clusters hold same label
  - Fast - each iteration in $O(n + m)$
  - Naïvely parallel - only per-vertex label updates
  - *Observation*: Possible applications for large-scale small-world graph partitioning

Label Propagation
Blank code and data available on website
(Lecture 7)
www.cs.rpi.edu/~slotag/classes/FA16/index.html