AALBORG UNIVERSITY

Social Networks: Community Detection

Peter Dolog
dolog@cs.aau.dk
http://people.cs.aau.dk/~dolog

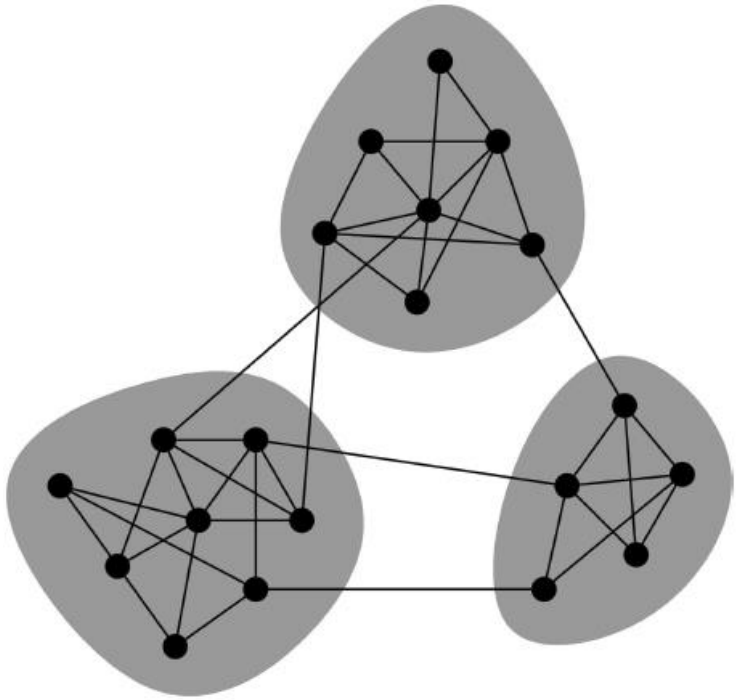
Based on the 'Social Media Mining: An Introduction' book (Chapter 6) by Reza Zafarani, Mohammad Ali Abbasi and Huan Liu; and on ICDM 2013 tutorial by same authors. Also including a few slides from Jure Leskovec's ICML '09 tutorial as well as slides of Bo Thiesson

Social Media Communities

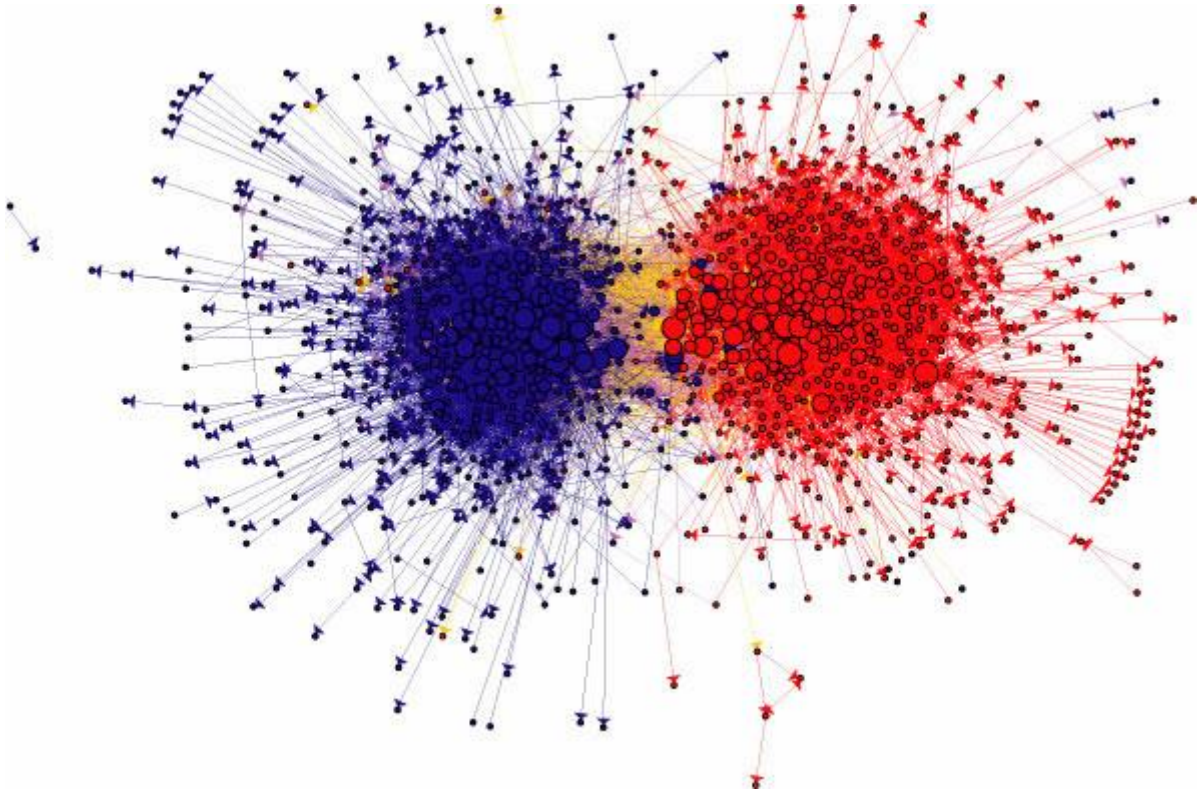
Findings so far suggest that network groups in social media are tightly connected

Network communities:

- Sets of actors with lots of connections/interactions/relations inside a community, and
- few connections to the outside (the rest of the network)



Political blogs

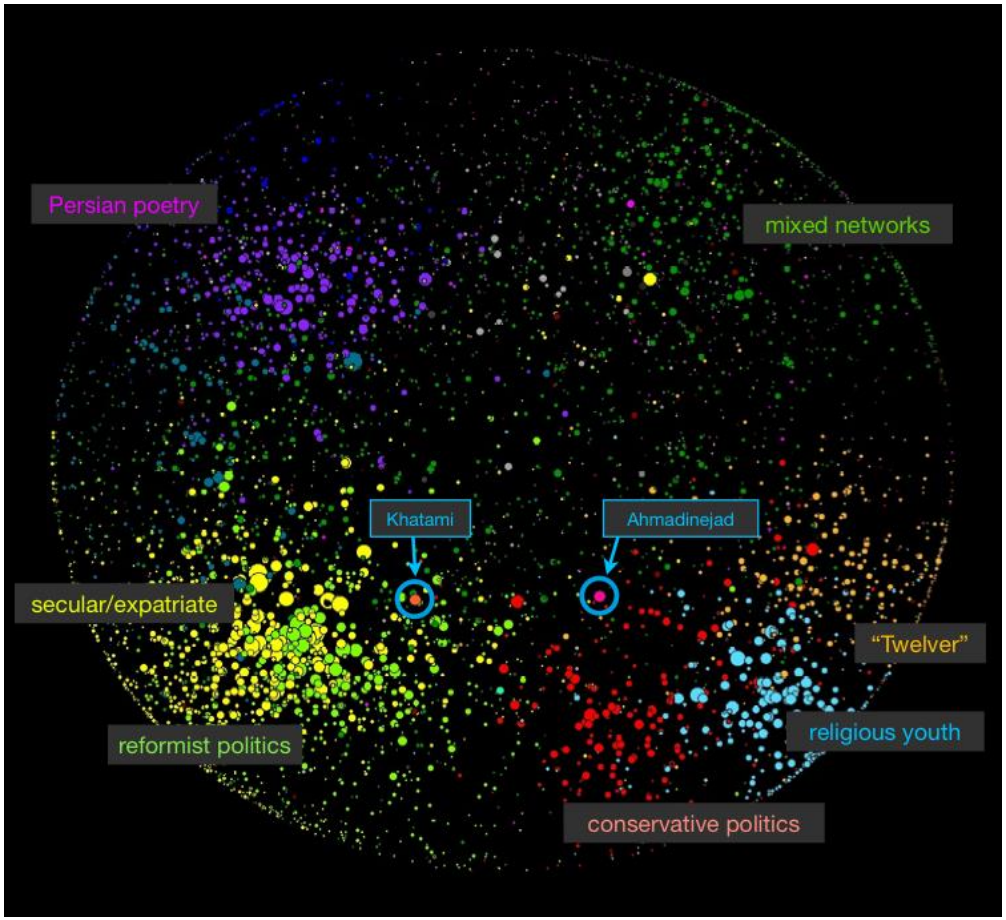


Adamic and Glance, 2005: The political blogosphere and the 2004 U.S. election

Iranian Blogosphere

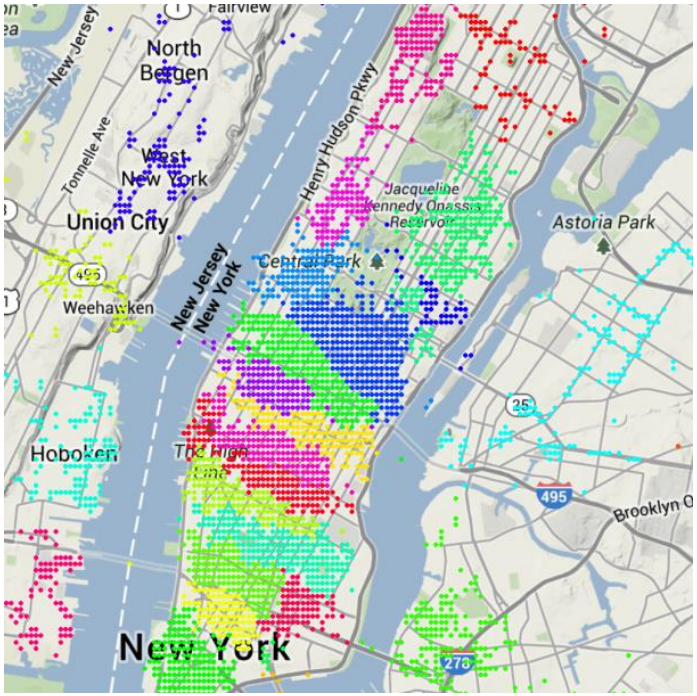
4 major communities:'

- *Secular/Reformist*
- *Conservative/Religious*
- *Persian Poetry and Literature*
- *Mixed Networks*

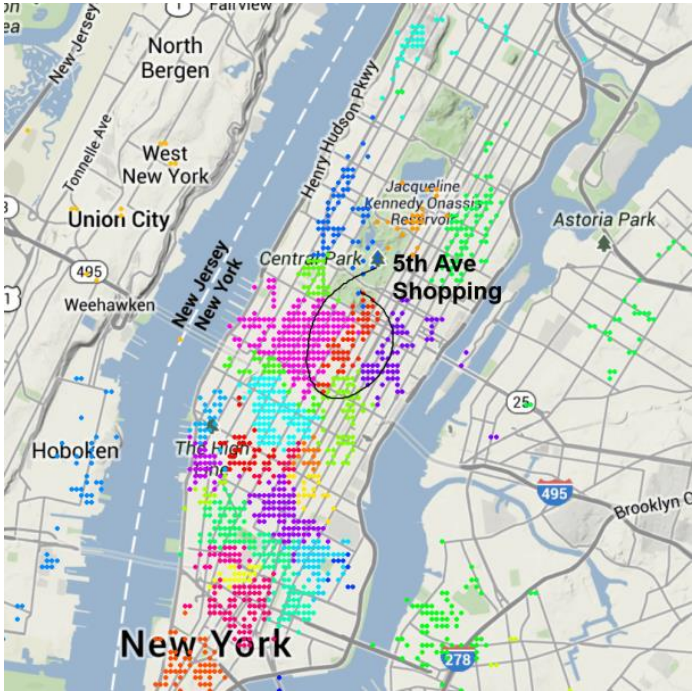


Kelly & Etling, 2008: Mapping Iran's Online Public: Politics and Culture in the Persian Blogosphere

Manhattan neighborhood boundaries



weekday



weekend

Kiciman *et al*, 2014: Discussion Graphs: Putting Social Media Analysis in Context

Social Media Communities

Two types of groups in social media

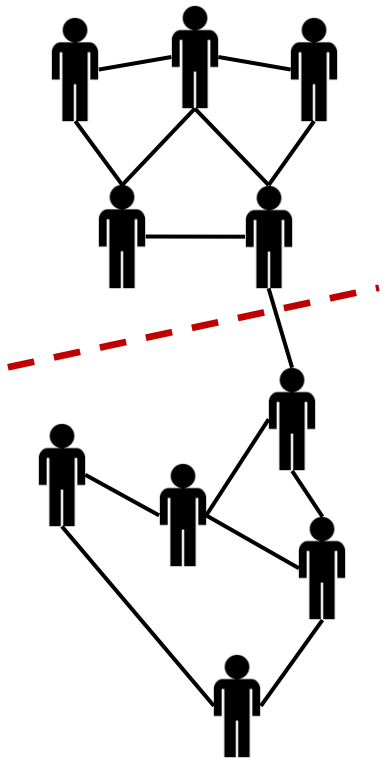
- Explicit Groups: formed by user subscriptions. E.g.,
  - Facebook groups or communities,
  - Yahoo! (mailing list) group,
  - LinkedIn groups and associations
- Implicit Groups: implicitly formed by social interactions or similarities. E.g.,
  - Individuals calling Canada from the United States -> phone operator considers them one community for marketing purposes
  - Political bloggers linking to each other
  - Tweeters with same sentiment towards a product

Community detection:

- Discovering **implicit** communities
- May be constrained by explicit groupings/relations

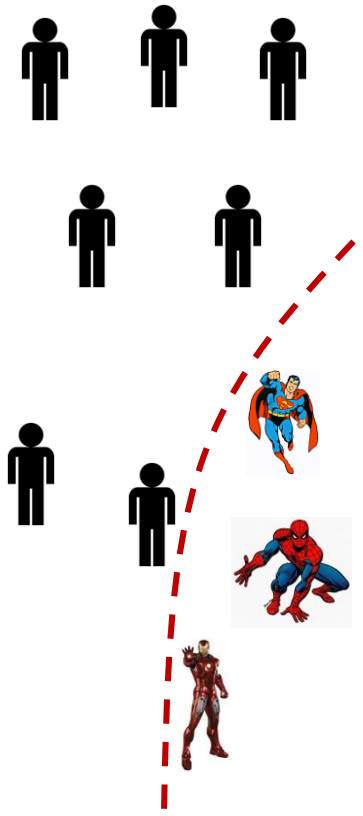
Community detection

- Community Detection is the process of finding clusters ("communities") of actors with
  - strong internal connections/relations, and
  - weak connections/relations between different clusters
- An ideal decomposition of a large graph is into completely disjoint groups of actors ("communities") with no interactions between different communities.
- In practice, the task is to find a partition into communities which are maximally decoupled.



Clustering

- Data points (actors) not explicitly embedded in a network
- Clusters based on similarities between actors
- General goal: find clustering with
  - Large within-cluster similarity
  - Small between-cluster similarity

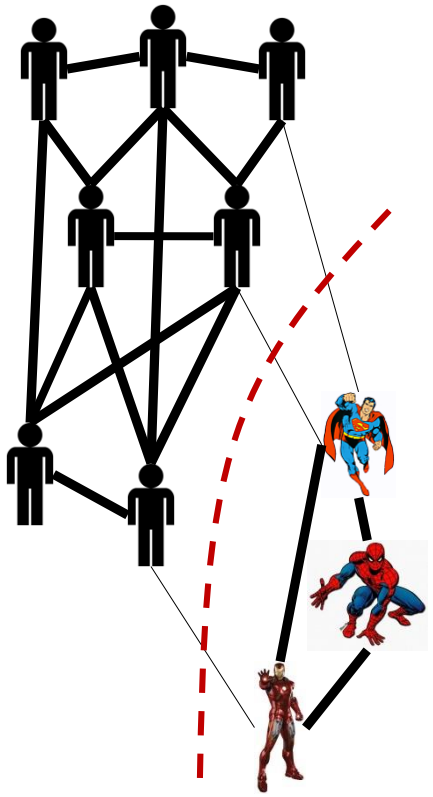


Q: Is this community detection?

Community detection versus clustering

Some argue that community detection and (standard) clustering are different:

- Clustering is based on (pairwise) similarities between nodes' features (i.e. nodes **not** embedded in graphical structure). In book: member-based
- Community detection is based on edges representing connections/relations/interactions (i.e. nodes embedded in graphical structure). In book: group-based



I disagree …it is just a matter of defining the semantic of an edge

- An edge represents the (degree of) similarity between two nodes according to a *chosen similarity measure*

Graph based similarities (structural equivalence)

For sets | | is count

Vertex similarity:

$$Sim_{Vertex}(v_i, v_j) = |N(v_i) \cap N(v_j)|$$

Jaccard similarity

$$Sim_{Jaccard}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$$

Cosine similarity

$$Sim_{Cosine}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{\sqrt{|N(v_i)||N(v_j)|}}$$

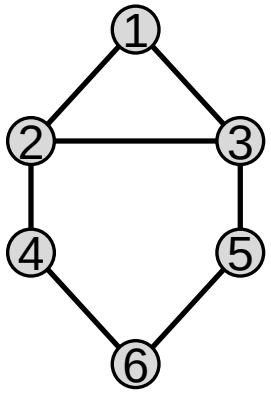
Structural Equivalence: Example

$$Sim_{Vertex}(v_2, v_5) = |\{1,3,4\} \cap \{3,6\}| = 1$$

$$Sim_{Jaccard}(v_2, v_5) = \frac{|\{1,3,4\} \cap \{3,6\}|}{|\{1,3,4,6\}|} = \frac{1}{4}$$

$$Sim_{Cosine}(v_2, v_5) = \frac{|\{1,3,4\} \cap \{3,6\}|}{\sqrt{|\{1,3,4\}||\{3,6\}|}} = \frac{1}{\sqrt{6}}$$

$$= \frac{N(v_2) \cdot N(v_5)}{|N(v_2)||N(v_5)|} = \frac{1}{\sqrt{3}\sqrt{2}} = \frac{1}{\sqrt{6}}$$



Neighborhood bit-vectors

$N(v_2)$	$N(v_5)$
1	0
0	0
1	1
1	0
0	1
0	0

For sets | | is count
For vectors | | is length (sqrt of "sqr-count")

Node based similarities

Let ϕ_i denote a feature vector for node v_i, e.g.:

- A node could be a blog/tweet and ϕ_i its tf-idf vector, or
- A node is a person and ϕ_i the tf-idf vector for all his blog/tweet posts over a period of time
- A neighborhood bit-vector (structure based)
- General demographics

Dot product

Vertex similarity:

$$Sim_{Vertex}(v_i, v_j) = \phi_i \cdot \phi_j$$

Jaccard similarity

$$Sim_{Jaccard}(v_i, v_j) = \frac{\phi_i \cdot \phi_j}{|\phi_i + \phi_j|^2}$$

Cosine similarity

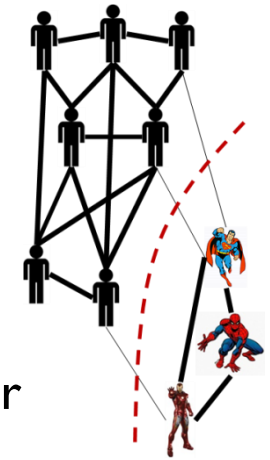
$$Sim_{Cosine}(v_i, v_j) = \frac{\phi_i \cdot \phi_j}{|\phi_i||\phi_j|}$$

For vectors | | is length (sqrt of "sqr-count")

Node based similarities – graph representation

- A weighted edge represents the (degree of) similarity between two nodes according to the *chosen similarity measure*

- **Q**: The graph may now become fully connected; is that a problem?
- **A1**: No (mostly)! – weight defines importance of edge
- **A2**: Could be a computational problem (for the community detection / clustering algorithm)



- Avoid fully connected graph as follows:
  - Most similarities will be very small.
  - ε-neighborhood graph: Only insert edges for similarities greater than ε, or
  - *k*-nearest neighbor graph: For each node, only insert edges to the k most similar nodes.

Word of advise…

How you define similarity is the single most **important** choice in community detection – more important than the actual clustering algorithm!

Methods to discover communities

Simple *local* based clustering

- Cliques (and variations)
- Reachability
- Clique percolation

Global based clustering

- Spectral clustering (balanced)
- Modularity maximization

Hierarchical clustering

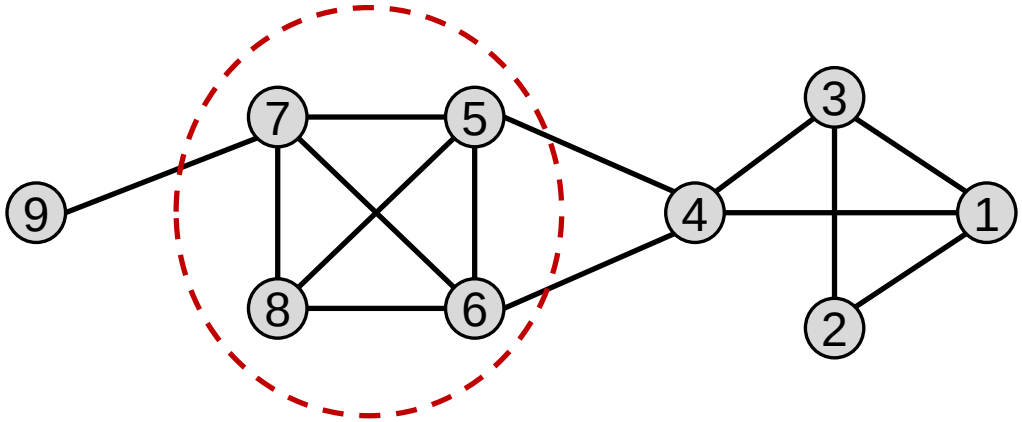
- Agglomerative methods (bottom-up)
- Divisive methods (top-down)
- Girvan-Newman (top-down, edge-betweenness)

Simple *local* structure based clustering

Clique clustering

Clique: a maximum complete subgraph in which all nodes are adjacent to each other. That is

$$d_v = |C| - 1, \text{ for all } v \in C$$



Example: Nodes 5, 6, 7 and 8 form a clique

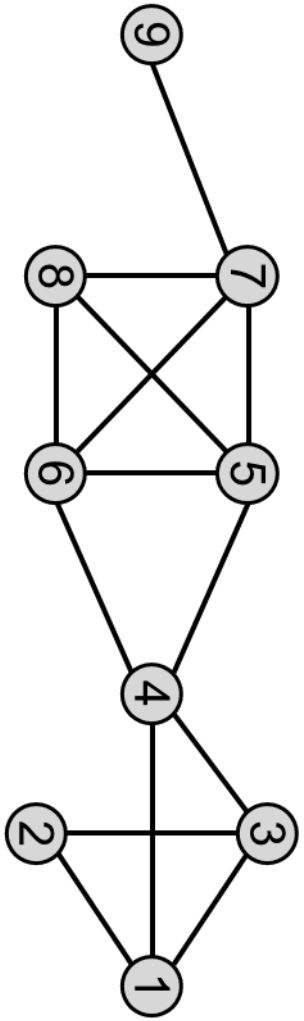
- Finding cliques is computationally expensive (NP-hard)

Brute-Force Clique Identification

(Algorithm 6.1 in SMM book)

Require: Adjacency Matrix A, Vertex v_x

1: **return** Maximal Clique C containing v_x
2: CliqueStack = $\{\{v_x\}\}$, Processed = $\{\}$;
3: **while** CliqueStack not empty **do**
4: C=pop(CliqueStack); push(Processed,C);
5: v_{last} = Last node added to C;
6: $N(v_{last}) = \{v_i | A_{v_{last},v_i} = 1\}$.
7: **for all** $v_{temp} \in N(v_{last})$ **do**
8: **if** $C \bigcup \{v_{temp}\}$ is a clique **then**
9: push(CliqueStack, $C \bigcup \{v_{temp}\}$);
10: **end if**
11: **end for**
12: **end while**
13: Return the largest clique from Processed

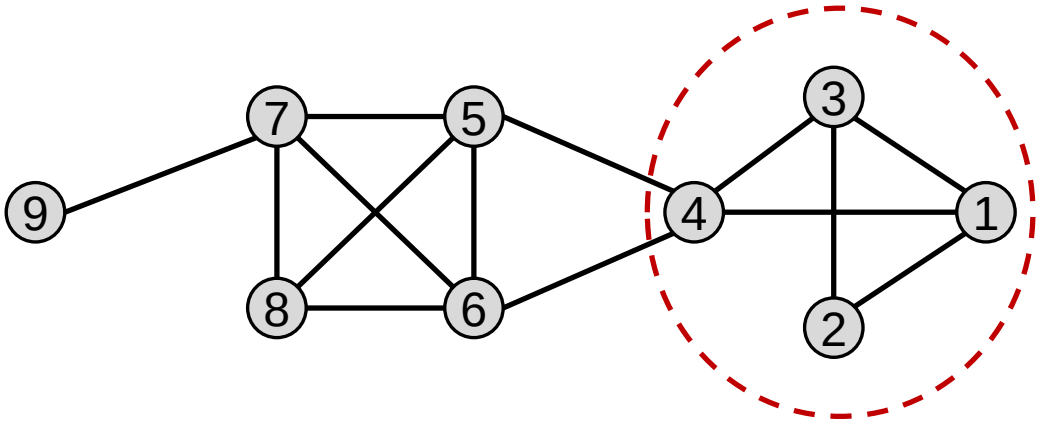


Relaxing cliques

The definition of clique is very strict; often, cliques are relaxed

k-plex: all nodes have a minimum degree not necessarily $|C| - 1$. That is (for $2 \leq k \leq |C| - 1$)

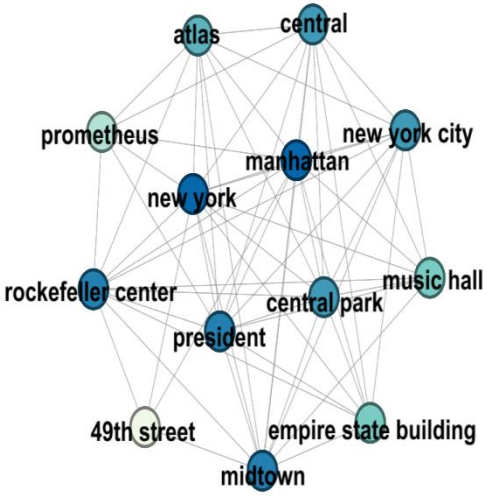
$$d_v \geq |V| - k, \text{ for all } v \in C$$



Example: Nodes 1, 2, 3 and 4 form a 2-plex (of size 4)

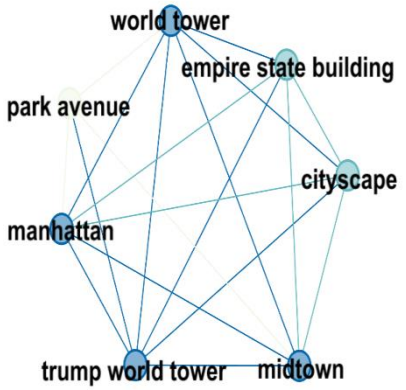
Contextual k-plexes of Tweet discussions

K-plex of
NYC tourist locations



		New York Tourist	Midtown Worker
Gender	Male	49%	63%
	Female	33%	23%
Metroarea	NYC	33%	54%
	Other	67%	46%
Mood	Joviality	56%	49%
	Fear	14%	13%
	Sadness	11%	15%
	Guilt	8%	6%
	Fatigue	3%	6%
	Serenity	3%	4%
	Hostility	2%	4%

K-plex of
NYC "midtown worker"



Clique Percolation Method (CPM)
(Using cliques as seeds)

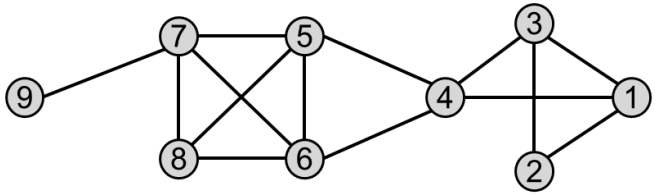
Input

- A parameter k, and a network

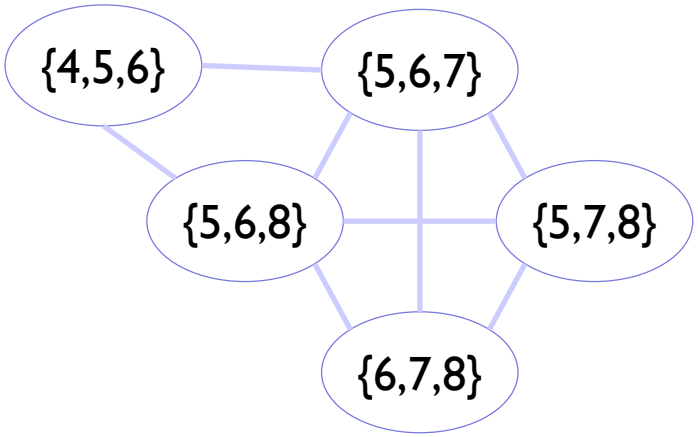
Procedure

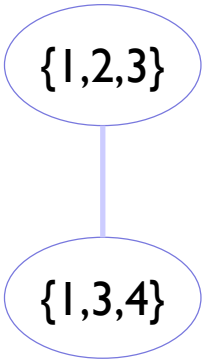
- Find all cliques of size k in the given network
- Construct a clique graph.
  - Two cliques are adjacent if they share $k-1$ nodes
- Each connected components in the clique graph form a community

Clique Percolation Method – 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}



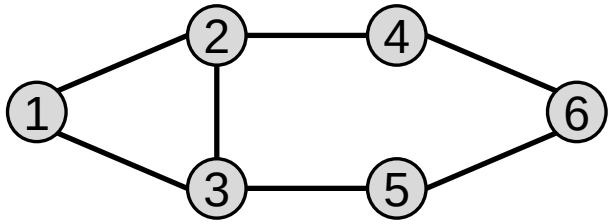


Communities:
{1, 2, 3, 4}
{4, 5, 6, 7, 8}

Reachability clustering

Any node in a group should be reachable in k hops

- k-Clique: a maximal subgraph in which the largest distance between any nodes <= k
- k-Club: it follows the same definition as k-clique with an additional constraint that nodes on the shortest paths should be part of the subgraph
- k-Clan: it is a k-clique where for all shortest paths within the subgraph the distance is equal to or less than k. All k-clans are k-cliques, but not vice versa. k-Clan = k-Clique ∩ k-Club



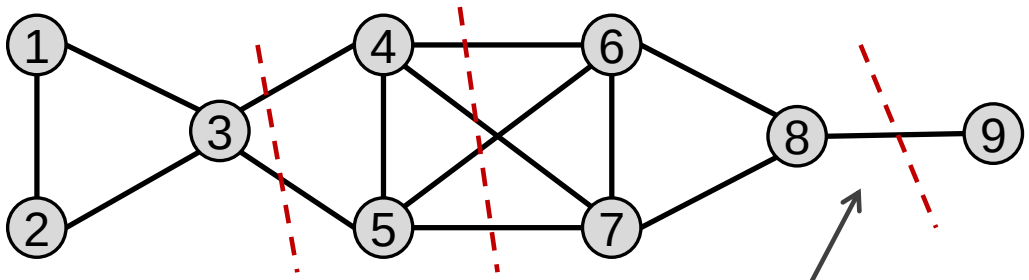
2-cliques: {1, 2, 3, 4, 5}, {2, 3, 4, 5, 6}
2-clubs: {1,2,3,4}, {1, 2, 3, 5}, {2, 3, 4, 5, 6}
2-clans: {2, 3, 4, 5, 6}

Global structure based clustering

Community Detection by Graph Cutting

- Cut: A partition of the nodes in 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
  - Graph $G = (V, E)$ partitioned into two disjoint subgraphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$
  - Objective: minimize

$$\frac{1}{2}\left(cut(G_A, G_B) + cut(G_B, G_A)\right) = \frac{1}{2}\left(\sum_{i \in V_A, j \in V_B}\left(w_{ij} + w_{ji}\right)\right)$$



- Minimum cut often returns an imbalanced partition, with one set being a singleton

RatioCut & NormalizedCut

- Graph $G = (V, E)$ partitioned into two disjoint subgraphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$
- Change the objective function to consider community size
  - $\text{RatioCut}(G_A, G_B) = \frac{1}{2}\left(\frac{cut(G_A,G_B)}{|V_A|} + \frac{cut(G_B,G_A)}{|V_B|}\right)$ ← Node normalization
  - $\text{NormalizedCut}(P) = \frac{1}{2}\left(\frac{cut(G_A,G_B)}{vol(G_A)} + \frac{cut(G_B,G_A)}{vol(G_B)}\right)$ ← Edge normalization
- Solvable by Spectral Clustering (as we will see)

Some math…

Eigenvalue & eigenvector

A (non-zero) vector v and scalar λ are, respectively, an eigenvector and corresponding eigenvalue of square matrix $\boldsymbol{A}$ if and only if

$$\boldsymbol{A}v = \lambda v$$

Eigen decomposition

Let $\boldsymbol{A}$ be a symmetric square matrix. An eigen-decomposition of A is a factorization

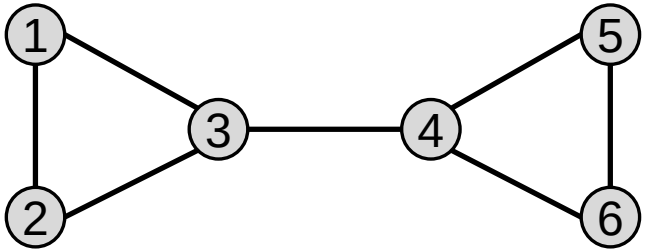
$$\boldsymbol{A} = \boldsymbol{V}^T \boldsymbol{\Lambda} \boldsymbol{V},$$

where $\boldsymbol{V}$ is a square matrix with linear independent eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix whose diagonal elements are the corresponding eigenvalues.

Notice: Standard (open source) algorithms solves the eigen-decomposition!

- E.g., Math.NET Numerics (http://numerics.mathdotnet.com/)
- Combined with Intel Math Kernel Library (MKL), which may boost computational perf. by an order of magnitude (http://numerics.mathdotnet.com/docs/MKL.html)
- Or NumPy: http://www.numpy.org/ see also http://people.duke.edu/~ccc14/sta-663-2016/07_LinearAlgebra2.html

Laplacian



A	1	2	3	4	5	6
1	0	w_{12}	w_{13}	0	0	0
2	w_{21}	0	w_{23}	0	0	0
3	w_{31}	w_{32}	0	w_{34}	0	0
4	0	0	w_{43}	0	w_{45}	w_{46}
5	0	0	0	w_{54}	0	w_{56}
6	0	0	0	w_{64}	w_{65}	0

Weighted degree:

$$d_i = \sum_j w_{ij}$$

Volume (of graph):

$$vol(G) = \sum_i d_i$$

Un-normalized Laplacian:

$$L = D - A$$

D	1	2	3	4	5	6
1	d_1	0	0	0	0	0
2	0	d_2	0	0	0	0
3	0	0	d_3	0	0	0
4	0	0	0	d_4	0	0
5	0	0	0	0	d_5	0
6	0	0	0	0	0	d_6

(Mohar (1991,1997): Let $\boldsymbol{L = V\Lambda V^T}$ be an eigen-decomposition for the Laplacian **L**, then there are as many eigen-vectors with associated eigen-value equal 0 as there are **connected components** in the graph. (And eigen-vector has same value on each element)

Spectral Clustering

- Both ratio cut and normalized cut can be reformulated as

$$\min_{\widehat{X}} Tr(\widehat{X}^T L \widehat{X})$$

$\widehat{X}$ consists of two normalized cluster membership vectors

- $L = \begin{cases} D - A & \text{RatioCut (unnormalized Laplacian)} \\ I - D^{-1/2} A D^{-1/2} & \text{NormalizedCut (normalized Laplacian)} \end{cases}$
- NP-hard
- Spectral relaxation

$$\min_{X \in R^{N \times 2}} Tr(\widehat{X}^T L \widehat{X}),$$

$$\text{s.t. } \widehat{X}^T \widehat{X} = I$$

- Solution: Eigen decomposition -- next slide…

Spectral clustering – in practice

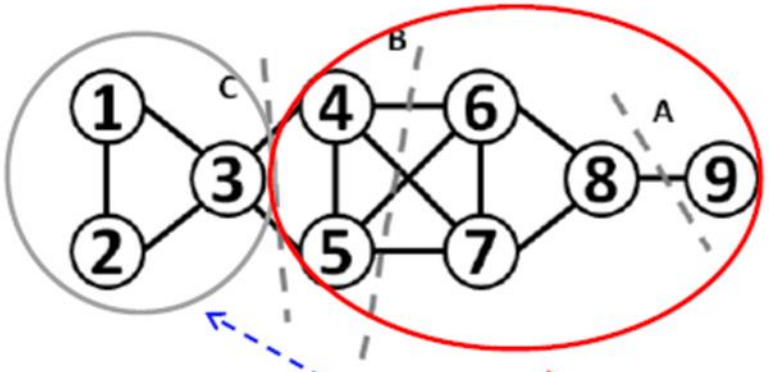
1. Compute the unnormalized Laplacian $\boldsymbol{L}$.
2. Compute the second eigenvector v_2 of $\boldsymbol{L}$ (the one with the second smallest eigenvalue – not 0).
3. The j'th value in eigenvector corresponds to the j'th node
4. Order the nodes according to their eigenvector-values
5. Cut at the largest gap.

Step2 solves $\boldsymbol{L}v = \lambda v$ for RatioCut (for eigen decomposition)

and $\boldsymbol{L}v = \lambda \boldsymbol{D}v$ for NormalizedCut (for generalized eigen decomposition)

Spectral clustering – Example

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



$$D = \text{diag}(2, 2, 4, 4, 4, 4, 4, 3, 1)$$

$$L = D - A = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 & -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Eigenvectors

$$\begin{bmatrix} 0.33 & -0.46 \\ 0.33 & -0.46 \\ 0.33 & -0.26 \\ 0.33 & 1.16 \times 10^{-16} \\ 0.33 & 1.16 \times 10^{-16} \\ 0.33 & 0.13 \\ 0.33 & 0.13 \\ 0.33 & 0.33 \\ 0.33 & 0.59 \end{bmatrix}$$

Modularity

- Intuition: Assuming that real-World networks are far from random, modularity measures how much the edges in a graph falls within clusters as opposed to randomly across clusters

Random edges

- Consider a graph, where edge degree (for each node) is known, but actual edges are unknown

- Q: How likely is it that a *particular* edge from node v_i connects to node v_j?
- A: $$\frac{d_j}{\sum_k d_k}$$

- Q: How likely is it then that *any* edge from node v_i connects to node v_j?
- A: $$r_{ij} = \frac{d_i d_j}{\sum_k d_k}$$

Modularity

- Intuition: Assuming that real-World networks are far from random, modularity measures how likely it is that the edges in a graph defines clusters as opposed to being created at random.

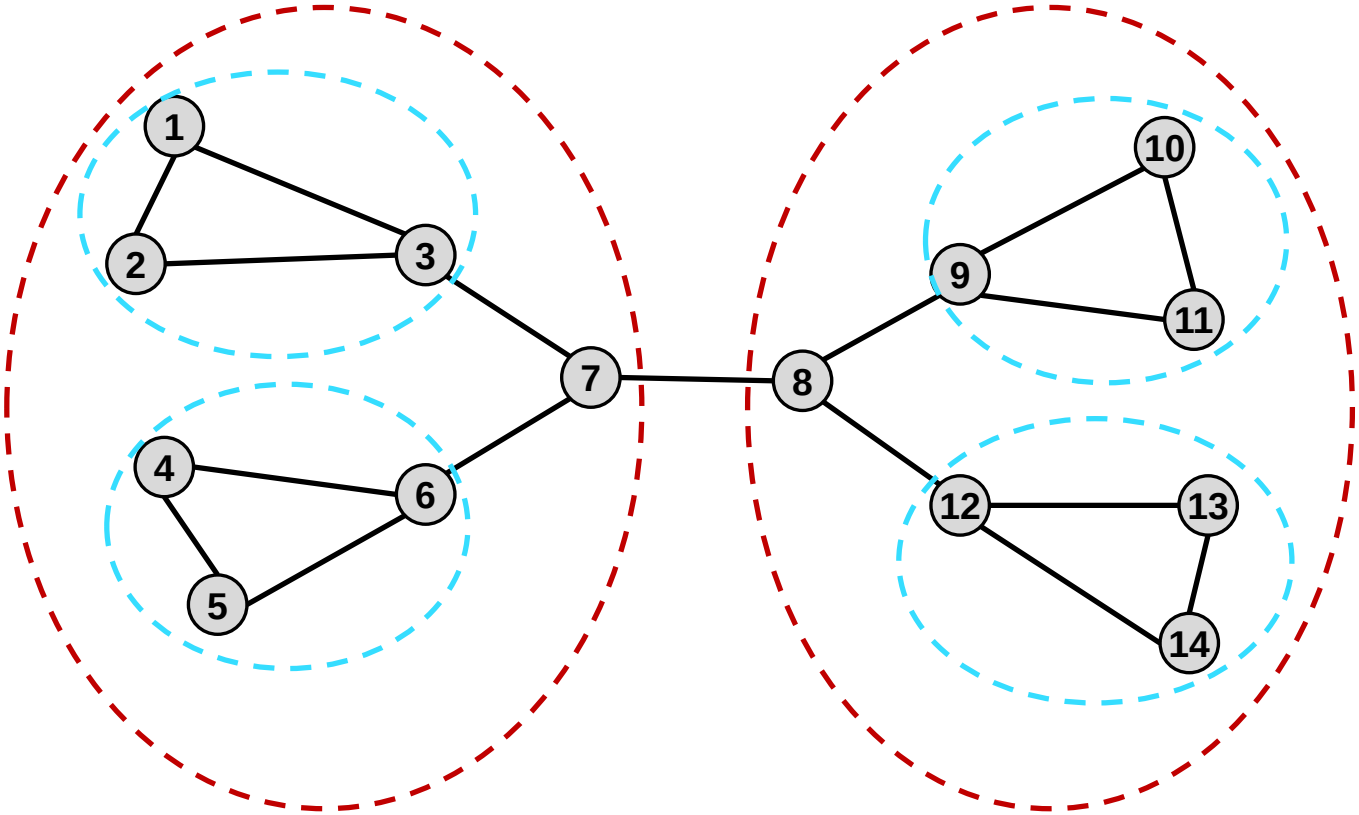
- Consider a graph $G = (V, E)$ partitioned into two disjoint subgraphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$

$$Modularity(G_A, \mathrm{G_B}) = \frac{1}{\sum_k d_k}\left(\sum_{i,j \in V_A} |w_{ij} - r_{ij}| + \sum_{i,j \in V_B} |w_{ij} - r_{ij}|\right)$$

- Very useful as secondary cluster selection criterion (see later)
- Maximization is NP-hard, but has relaxed solution (see book)

Hierarchical clustering

Hierarchically nested communities

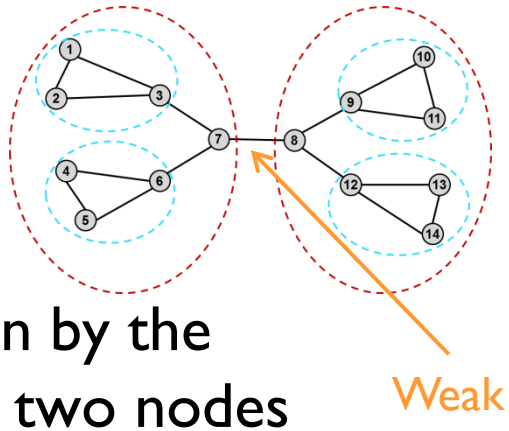


What would be a good measure for cutting edges?

The Girvan-Newman Algorithm

Divisive hierarchical clustering based on the notion of edge betweenness:

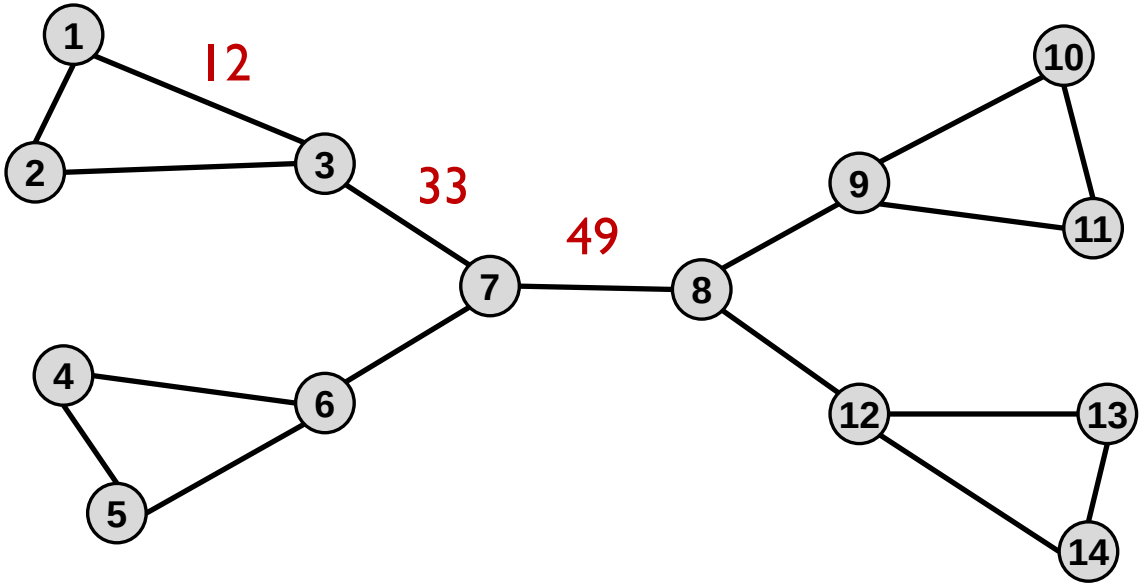
- Tries to find communities by discovering weak ties
- A weak tie is an edge that connect a lot of nodes with a lot of other nodes
- This is measured by betweenness



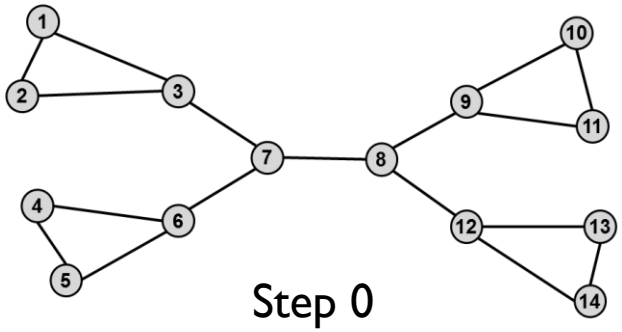
Intuition behind the notion of a weak tie is given by the associated structural similarity measure for the two nodes connected by the edge

Edge Betweenness - Definition

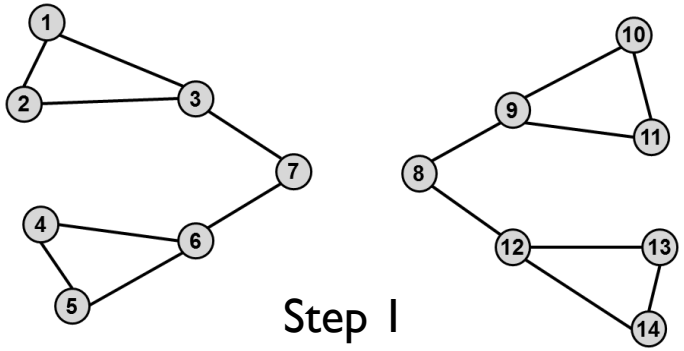
Edge betweenness: For an edge measures how many shortest paths passes through that edge



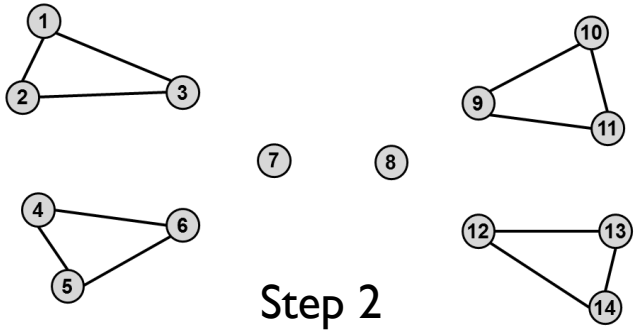
Girvan-Newman Example



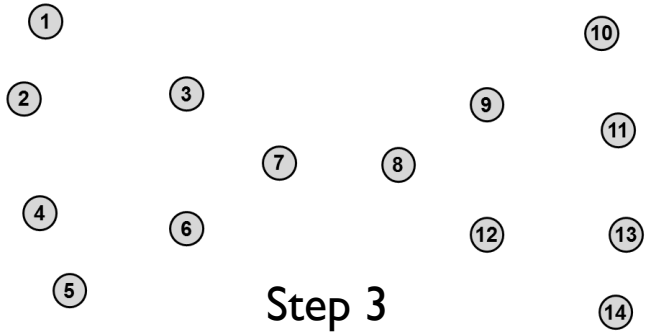
Step 0



Step 1



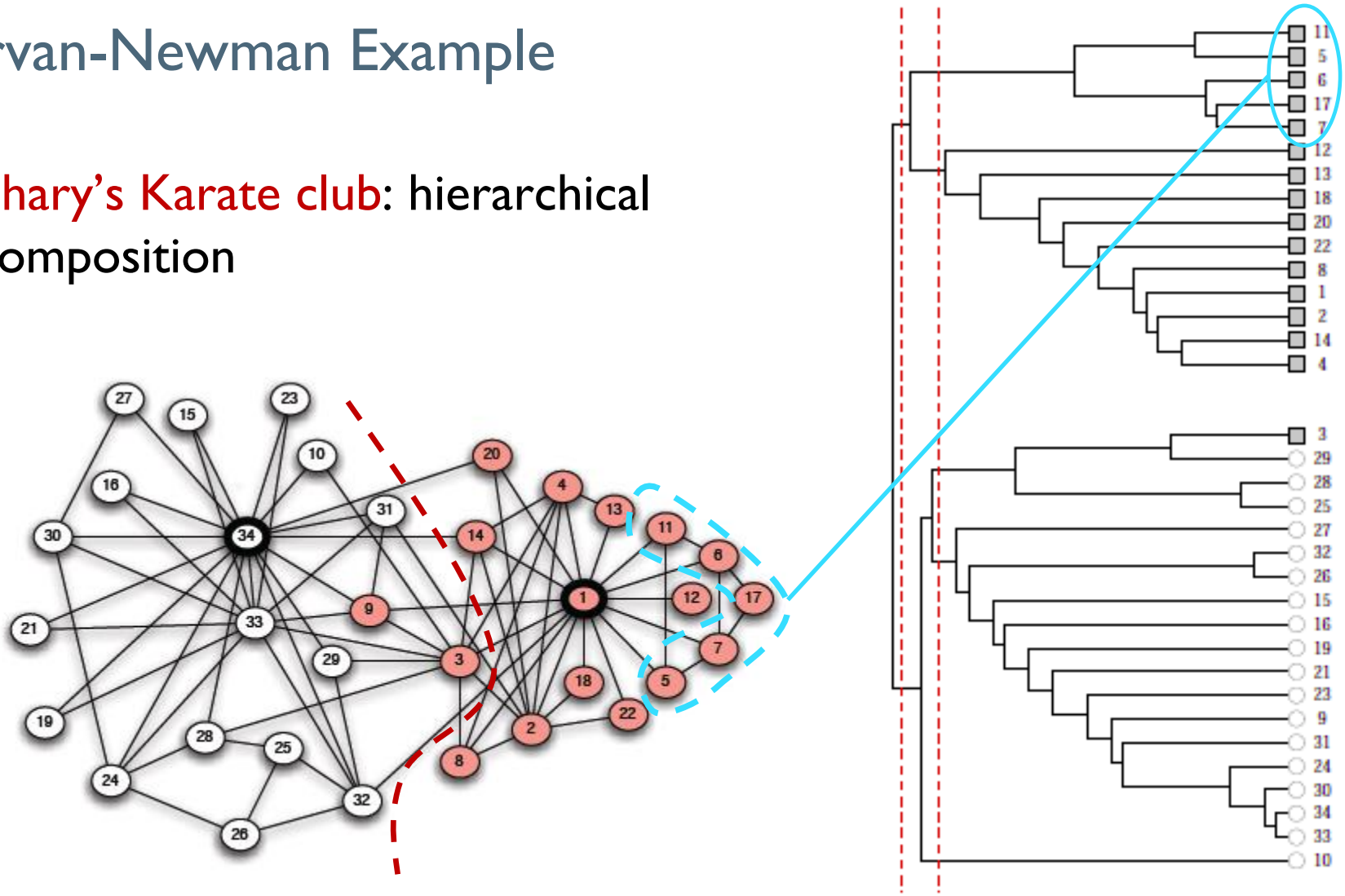
Step 2



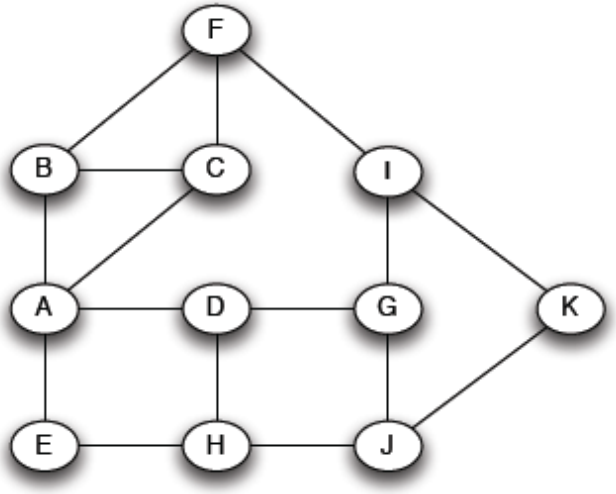
Step 3

Girvan-Newman Example

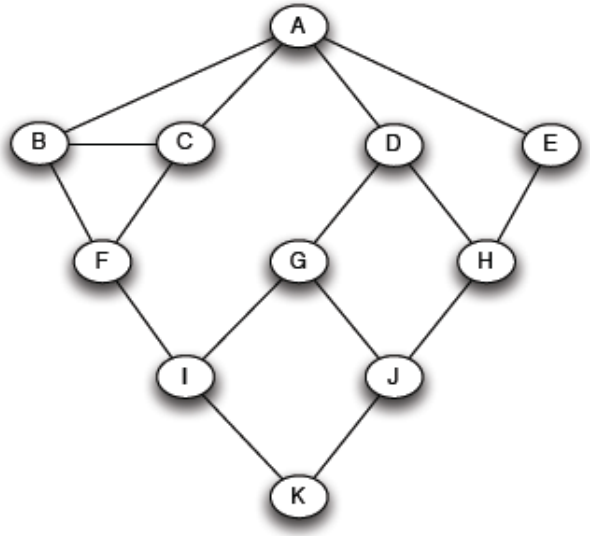
Zachary's Karate club: hierarchical decomposition



How to Compute Betweenness (step 1)

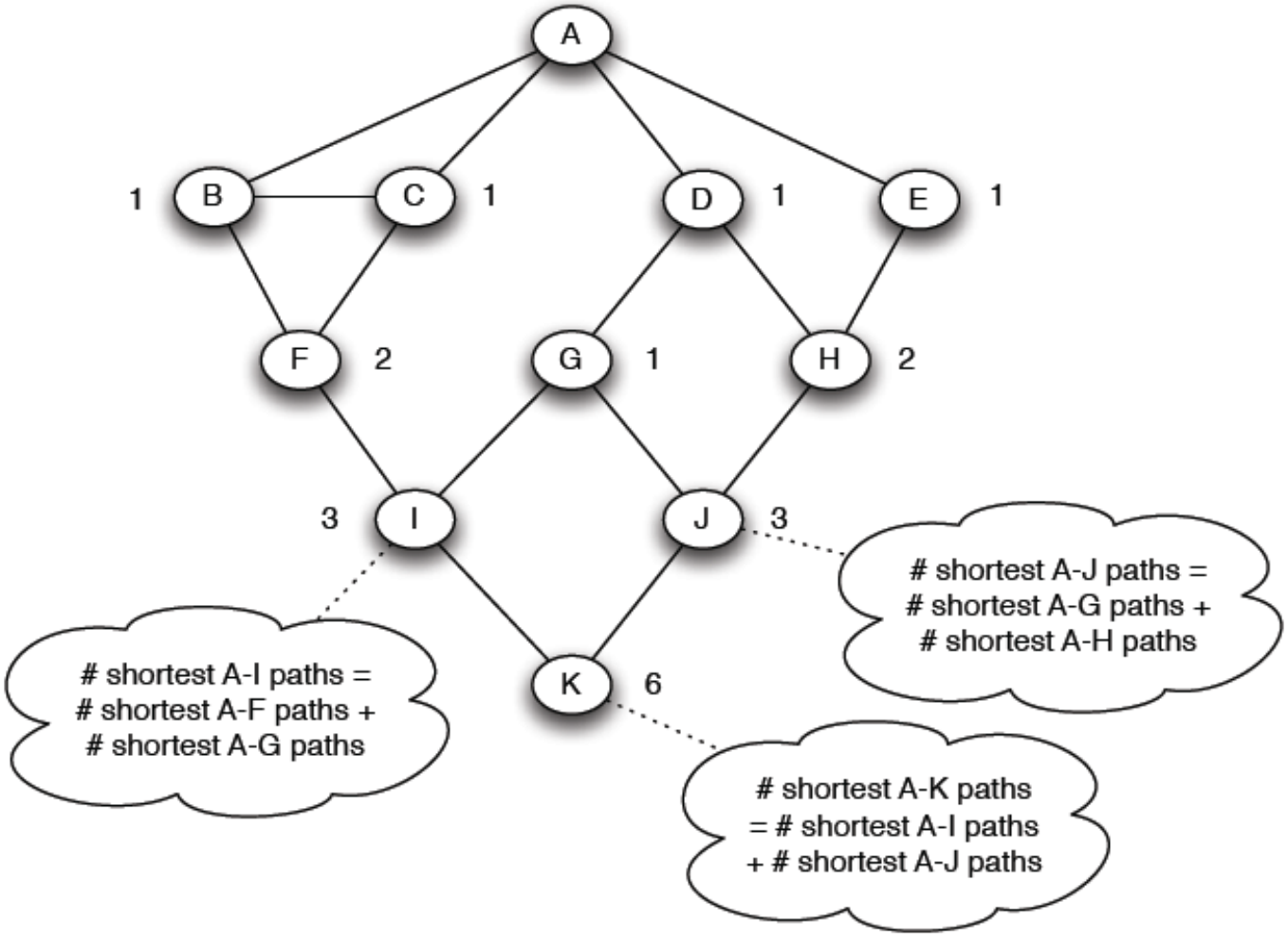


Breath first search starting from A:

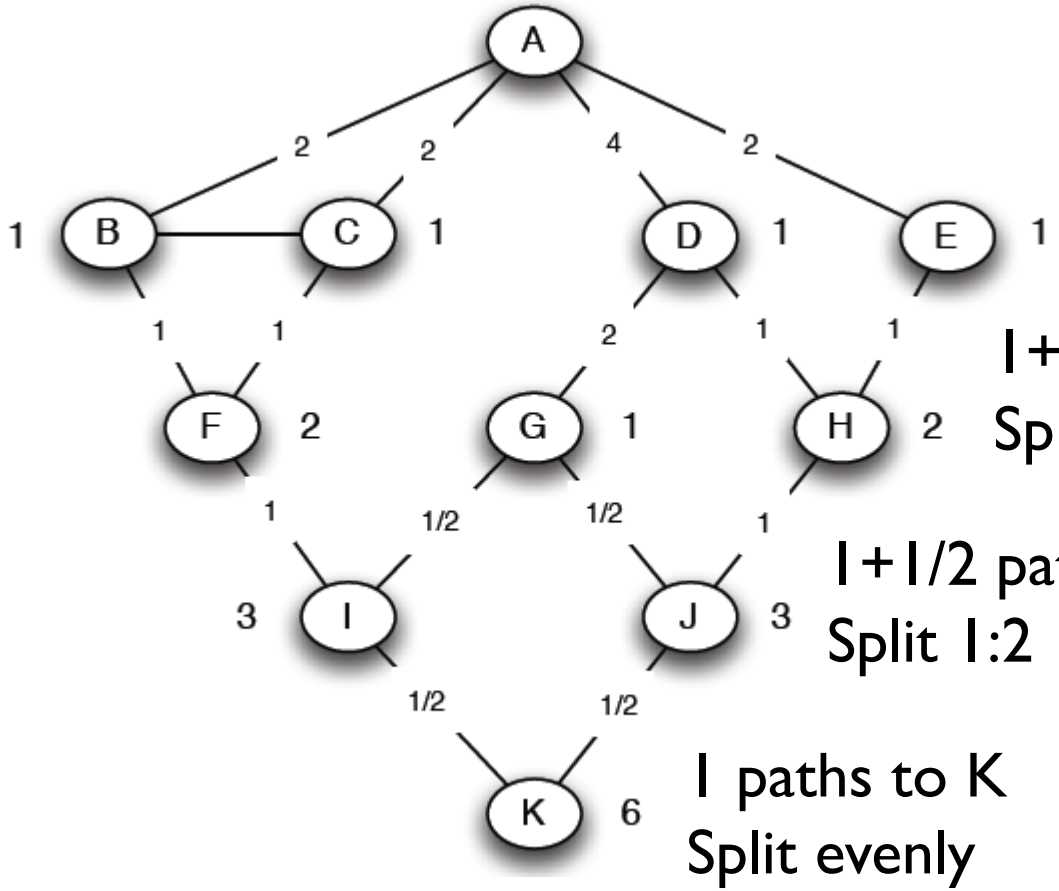


Want to compute betweenness of all paths through G starting at node A

How to Compute Betweenness (step 1 cont.)



How to Compute Betweenness (step 2)



1+1 paths to H
Split evenly

1+1/2 paths to J
Split 1:2

1 paths to K
Split evenly

Steps 1 & 2 details

Step 1. Compute # shortest paths

1. The initial vertex s is given distance $d_s = 0$ and a weight $w_s = 1$.
2. Every vertex i adjacent to s is given distance $d_i = d_s + 1 = 1$, and weight $w_i = w_s = 1$.
3. For each vertex j adjacent to one of *those* vertices i we do one of three things:
   (a) If j has not yet been assigned a distance, it is assigned distance $d_j = d_i + 1$ and weight $w_j = w_i$.
   (b) If j has already been assigned a distance and $d_j = d_i + 1$, then the vertex's weight is increased by w_i, that is $w_j \leftarrow w_j + w_i$.
   (c) If j has already been assigned a distance and $d_j < d_i + 1$, we do nothing.
4. Repeat from step 3 until no vertices remain that have assigned distances but whose neighbors do not have assigned distances

Step 2. Compute edge score

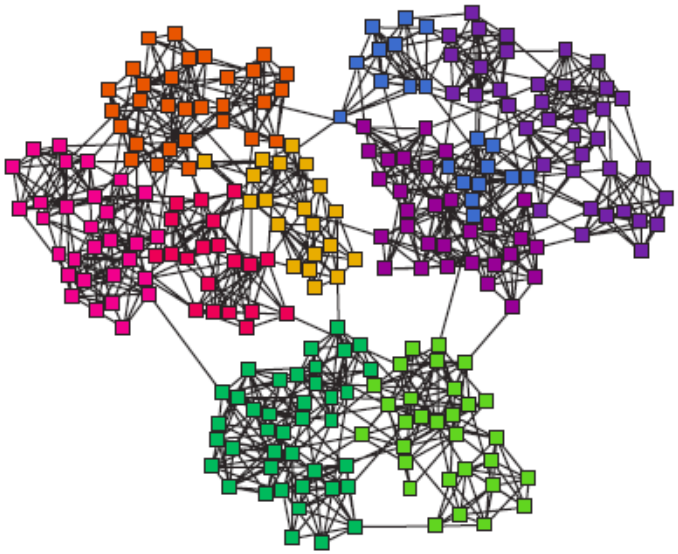
1. Find every "leaf" vertex t, i.e., a vertex such that no paths from s to other vertices go though t.
2. For each vertex i neighboring t assign a score to the edge from t to i of w_i/w_t.
3. Now, starting with the edges that are farthest from the source vertex s—lower down in a diagram such as Fig. 4b—work up towards s. To the edge from vertex i to vertex j, with j being farther from s than i, assign a score that is 1 plus the sum of the scores on the neighboring edges immediately below it (i.e., those with which it shares a common vertex), all multiplied by w_i/w_j.
4. Repeat from step 3 until vertex s is reached.

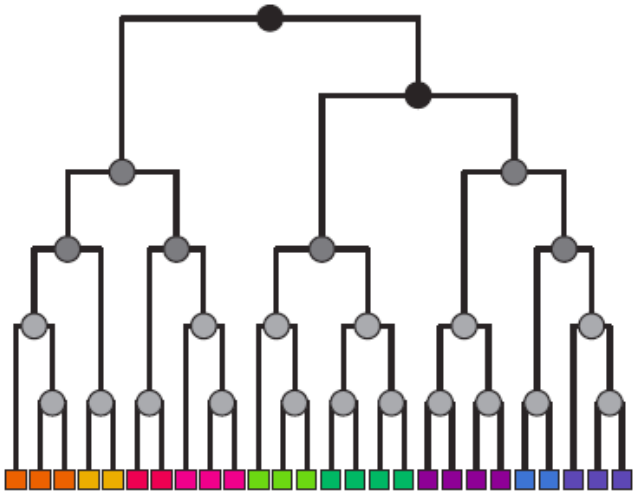
How to Compute Betweenness (full algorithm)

- Repeat steps 1 & 2 for each node v_i in the network
- Add edge scores $s_i(e)$

$$betweenness(e) = \sum_i s_i(e)$$

Hierarchical clustering





At the end we have a dendrogram corresponding to the hierarchical clustering

- As we move closer to the root, nodes join to form larger communities
- Each level corresponds to a clustering

What is a good level?

We have a dendrogram with m levels and each level corresponds to a clustering

- What is a the best level?, i.e.,
- What is a good clustering
- Many ways to measure the quality of a clustering
- A popular measure for networks is modularity

Summary

Clustering = graph based community detection

- …it is just a matter of defining the semantic of an edge

Simple *local* (member) based clustering

- Cliques (and variations)
- Reachability
- Clique percolation

Global (group) based clustering

- Spectral clustering (balanced)
- Modularity maximization

Hierarchical clustering

- Agglomerative methods (bottom-up)
- Divisive methods (top-down)
- Girvan-Newman (top-down, edge-betweenness)