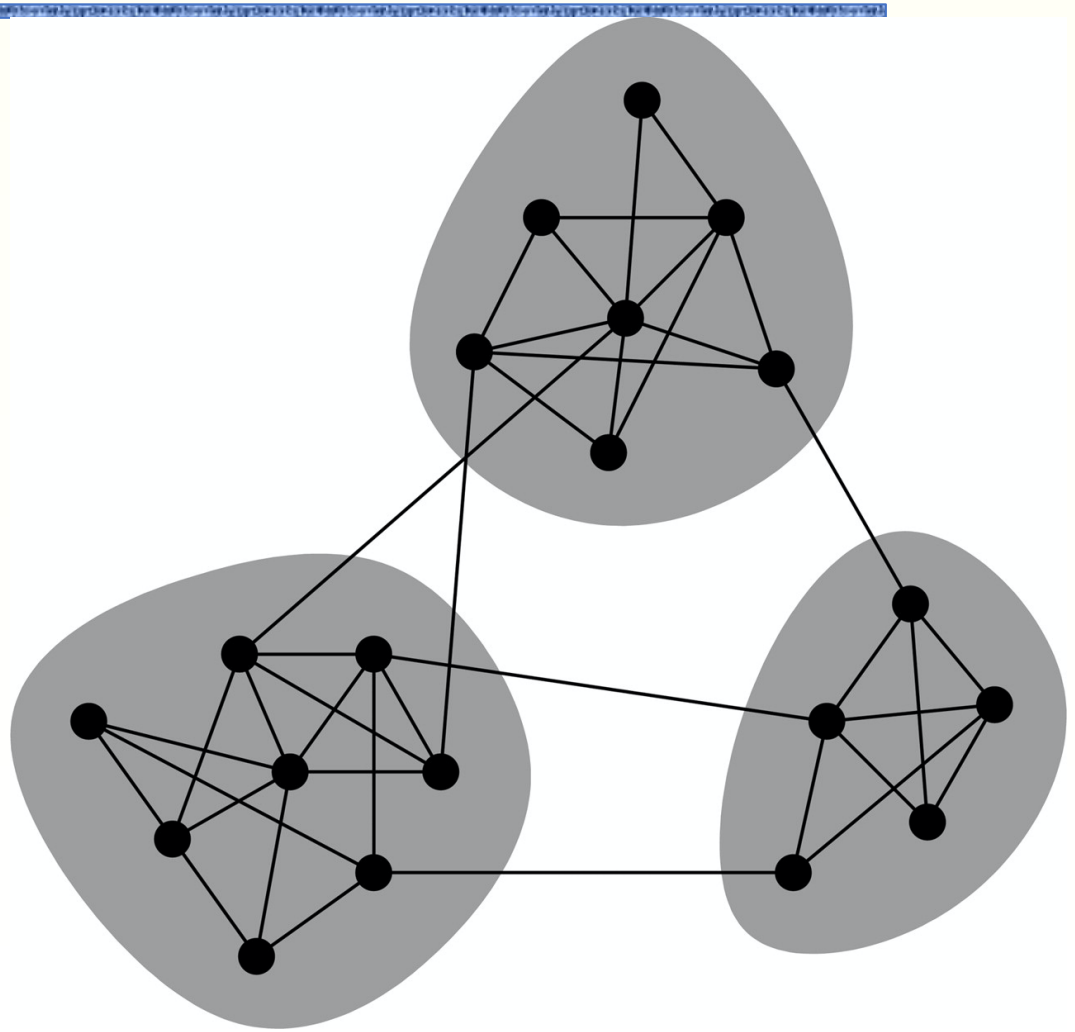
Community Detection (Graph Clustering)

Community Structure

- Clusters = Groups = Modules
  - aka communities
  - (=/= Cliques)
- A community is a tightly knit group of nodes, potentially connected more loosely to other communities.
- Tight / Loose defined in terms of triadic closure
- Many methods for detecting communities
  - Minimum-Cut
  - Edge Betweenness
  - Modularity Maximization
  - Label Propagation



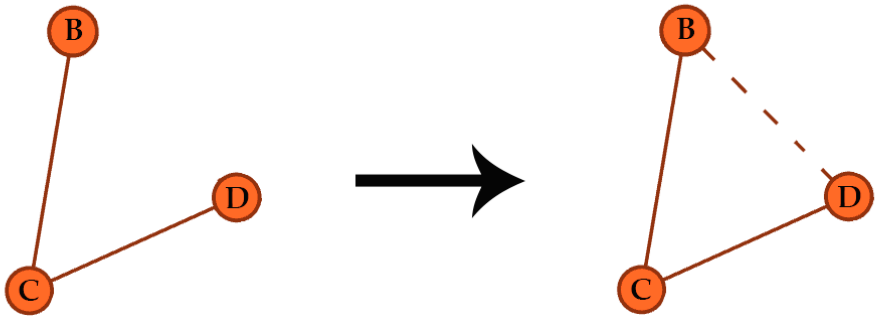
M. E. J. Newman PNAS 2006;103:8577-8582

Triadic Closure

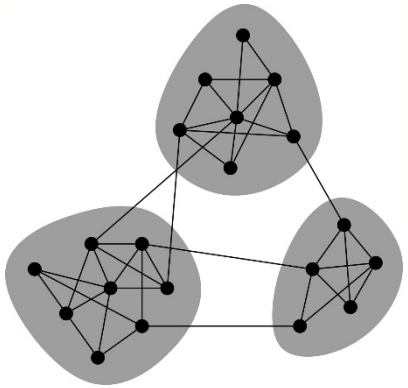
- Higher triadic closure = Higher clustering coefficient

- If C is connected to B and D
  - Are B and D also connected?
- If they are, then this set of nodes forms triadic closure

- Zoom out to a set of nodes
- The more triadic closure within a subgroup of nodes
  - The more likely they belong to a cluster



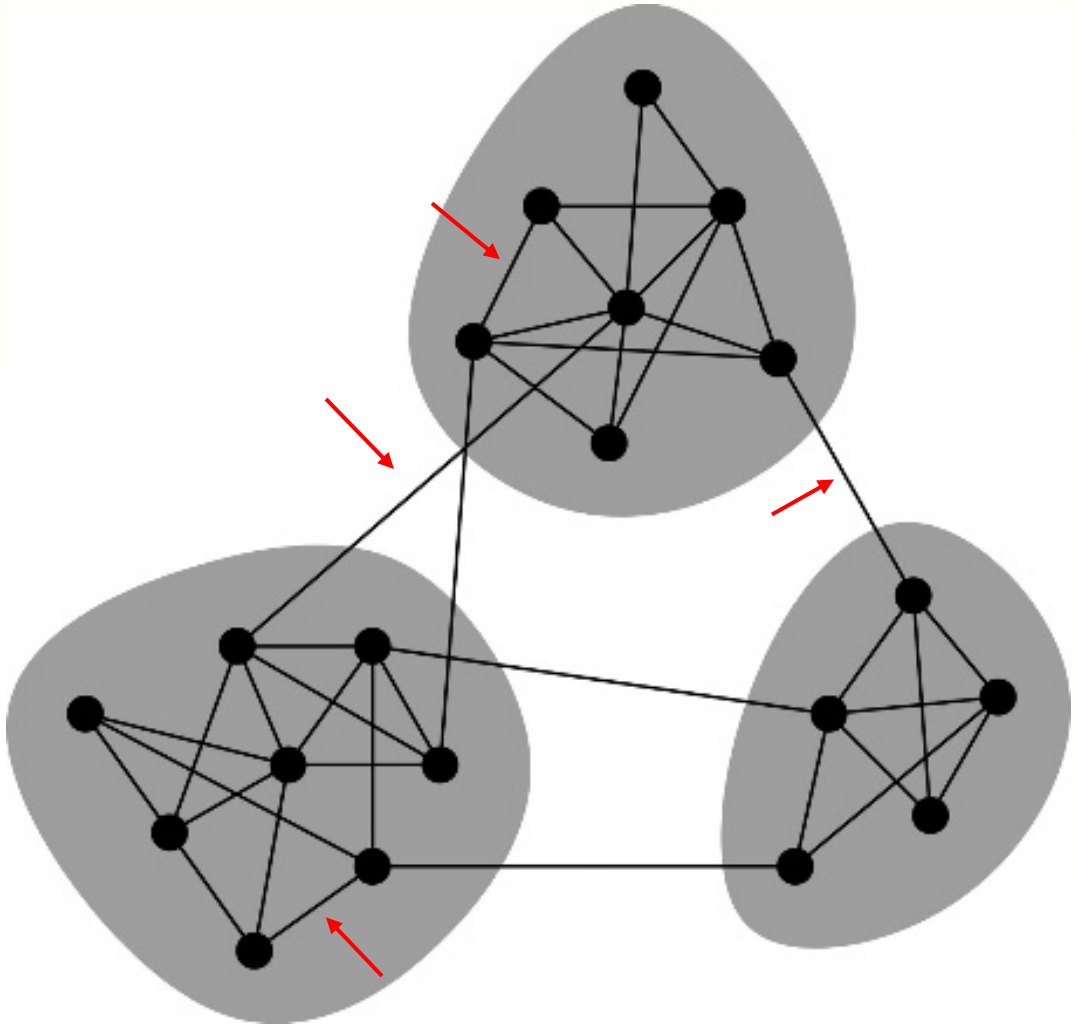
http://mw2015.museumsandtheweb.com/wp-content/uploads/2015/01/espinos-figure6.png



Strong and Weak Ties

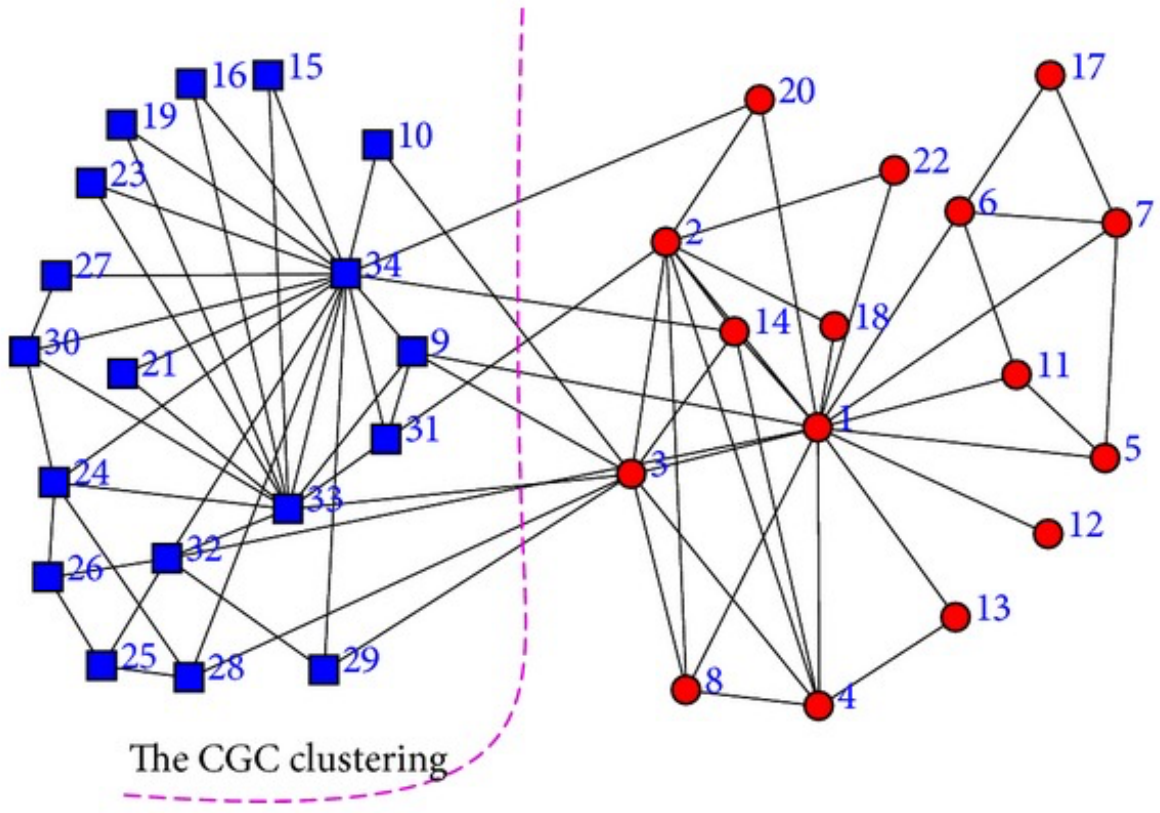
- Weak ties (also called bridges) are those without triadic closure for all or most adjacent connections

- Strong ties have triadic closure on most of their adjacent connections



Why Clustering?

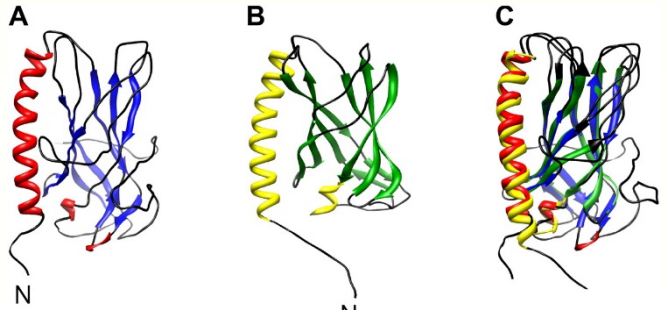
- Most, but not all networks have community structures
- Ideally, clusters correspond to meaningful groups of nodes
  - Biology- function
  - Social – groups (Ex. Karate club)
- A sociologist recorded relationship dynamics in a karate club that eventually fractured into 2 clubs
- Clustering on the initial group breaks apart group A from group B



"Follow the Leader": A Centrality Guided Clustering and Its Application to Social Network Analysis - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/258956117_fig1_The-social-network-of-Zachary's-karate-club-Red-dots-denote-the-supporters-of-instructor [accessed 21 Mar, 2017]

Applications

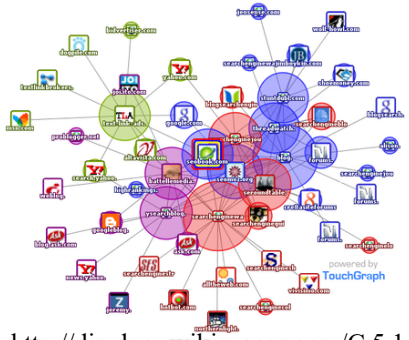
- Sociology/psychology
  - Social network mining
  - Find missing data about a person based on similar people/friends



Caleb Knepper et al. Plant Physiol. 2011;156:286-300

- Bioinformatics
  - Protein homology detection
  - Find proteins that perform similar functions

- Network Security / Criminal Justice
  - Clusters of illicit websites
  - These sites tend to link mostly to each other



http://dis-dpcs.wikispaces.com/C.5.1+-+The+Web+as+a+directed+Graph

Algorithms

1. Minimum-Cut
2. Edge Betweenness
3. Modulatiry Maximization
   - Optimal
   - Approximate: Louvain
4. Label Propagation

Minimum Cut

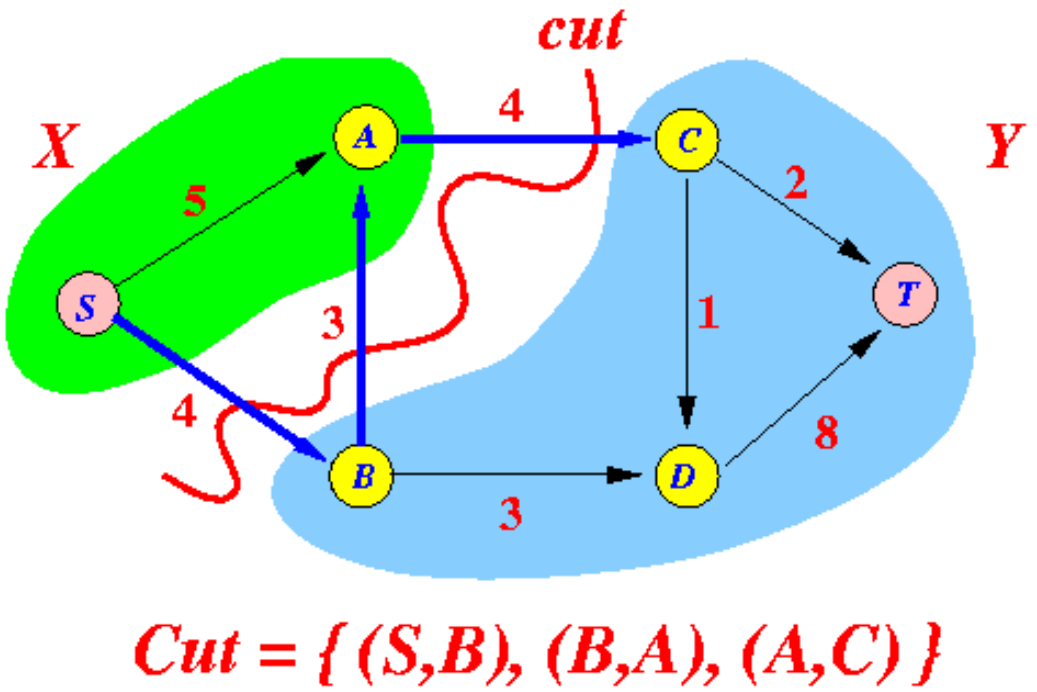
Minimum Cut Problem:

- Find the smallest number of edges that can be removed from a graph to partition it into two segments.

- Min Cut Max Flow Theorem

- Performing iterative cuts will break the graph into a number of clusters

Problems with this approach?



http://www.mathcs.emory.edu/~cheung/Courses/323/Syllabus/NetFlow/max-flow-min-cut.html

Minimum Cut Pro/Con

Benefits:

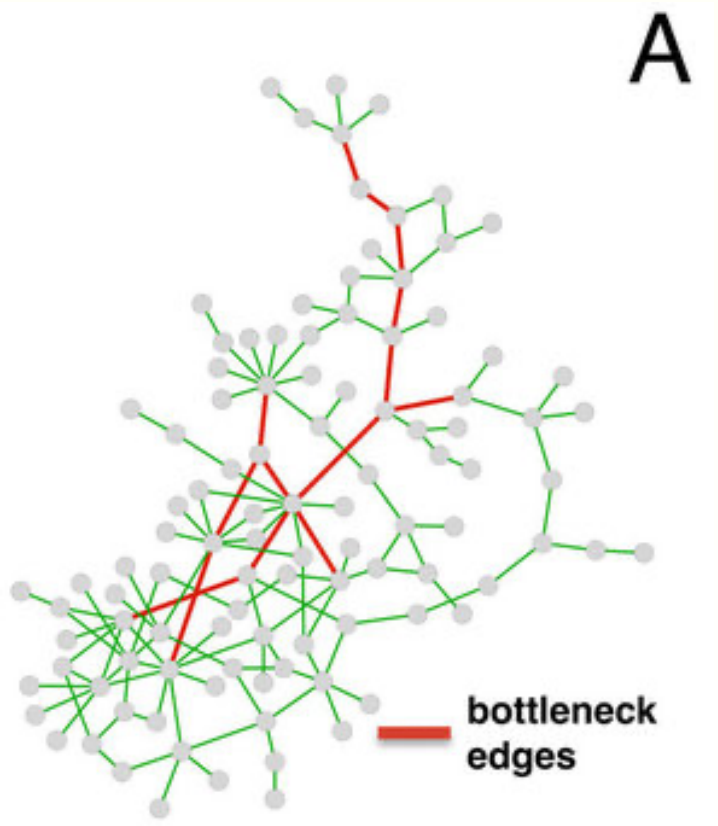
- Works well if number of clusters is known
- Can be relatively fast (close to quadratic time) with careful algorithms

Limitations:

- Better designed for partitioning
- Predetermined how many clusters exist
- In the case where number of clusters is unknown, not clear how many clusters is “enough”
- Will find clusters that do not exist

Betweenness

- Betweenness: For each node/edge in a graph, how many shortest paths is that node/edge on?
- In this case, we care about edges
- Ex. Protein interaction graph
- Intuitively, these edges are important because they connect groups of nodes that are poorly connected otherwise.
- Sound familiar?



S. Wuchty & Peter Uetz Scientific Reports 4, Article number: 7187 (2014)

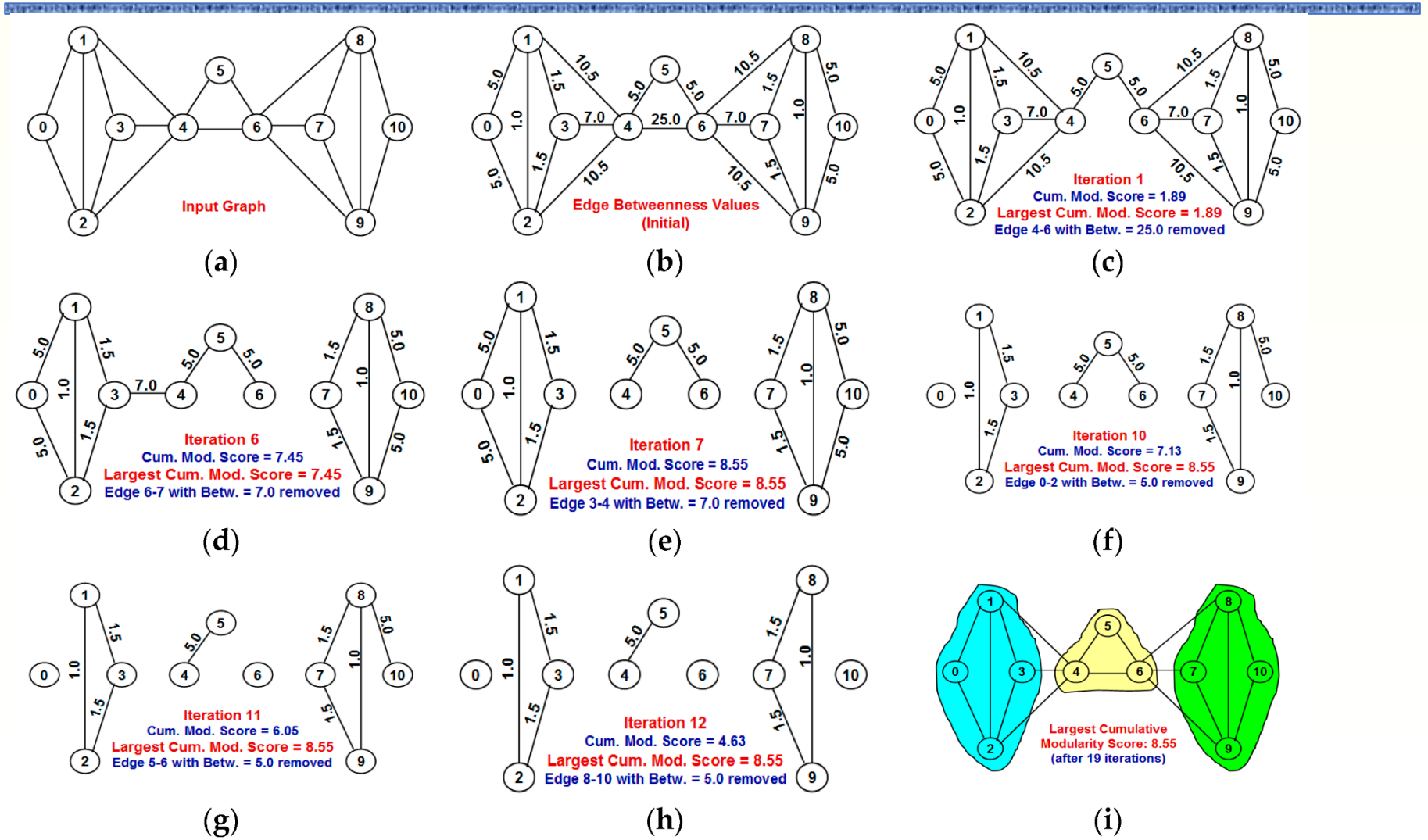
Girvan-Newman Algorithm

- Uses edge betweenness as a measure of weak-ties
- Iteratively removes high betweenness edges to find clusters

Procedure:

Calculate betweenness of all existing edges in the network
While the number of edges in G exceeds zero
    Remove the edge with the highest betweenness
    Recalculate the betweenness of all edges affected by the removal

Girvan-Newman Algorithm



Natarajan Meghanathan, A Greedy Algorithm for Neighborhood Overlap-Based Community Detection, *Algorithms* 2016, *9*(1)

Girvan-Newman Algorithm

Strengths:

- Intuitive
- Number of clusters is based on graph structure (not predefined)

Weaknesses:

- Relatively slow: requires BFS (O(|V|+|E|)) at *every* step to calculate betweenness
- Newer algorithms produce more modular clusters

In igraph this algorithm is **cluster_edge_betweenness**

Modularity

- A measure of how strong a given grouping is on a graph
- Roughly speaking modularity is the ratio of the number of edges within a group over the expected number if edges were rewired randomly

Definition:

- Range [-1,1]
- e_{ii} = % edges in group i
  - e_{ii} =|{(u,v) : u ∈ V_i, v ∈ V_i, (u,v)∈E}| / |E|
- a_i = % edges with at least 1 end in group i
  - a_i = |{(u,v) : u ∈ V_i, (u,v) ∈ E}| / |E|
- Q = $\sum_{i=1}^{k}(e_{ii}-a_i^2)$
  - e_{ii}: probability edge is in group i
  - a_i: probability a random edge would fall into group i

Modularity Maximization

- This implies a new method: Find set of subsets of V(G) to maximize this value
  - Gold standard for clusters

- But… modularity is NP-hard to optimize ☹

- Exact calculation is **cluster_optimal** in igraph
- This is going to be VERY slow, however (obviously)

- Several good approximations exist
  - One popular method is the Louvain algorithm

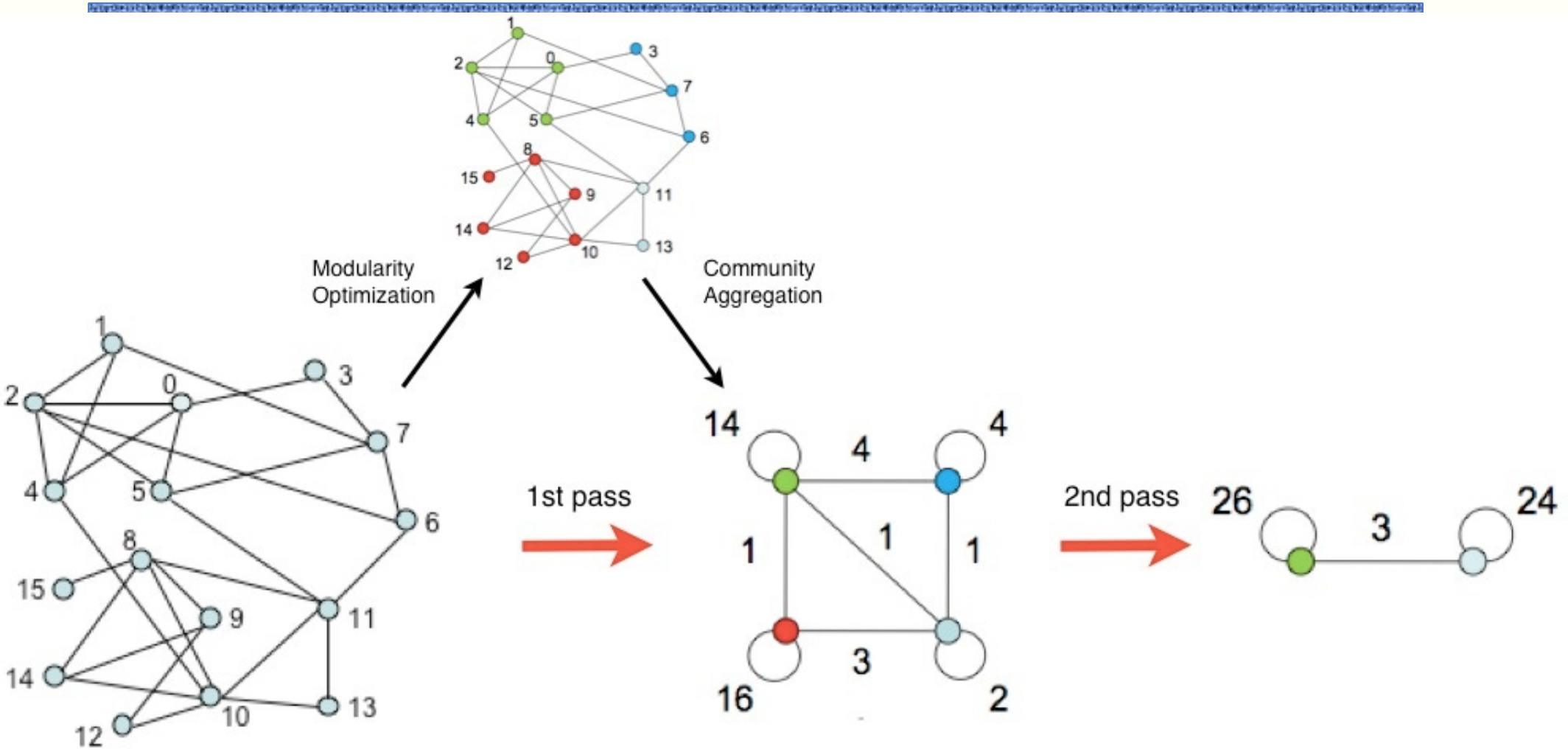
Louvain aka Multilevel Method

- A local greedy optimization of modularity maximization

Procedure:

Initialize modularity to 0
Do While there is a gain in modularity
    Name each node as its own cluster
    While there is a singleton cluster
        Select a singleton node, v
        Add v to the cluster that maximizes modularity
    Generate a new graph, where each cluster is merged to a single node
    Calculate modularity in the original network

Louvain



https://sites.google.com/site/findcommunities/

Louvain

Strengths:

- In practice produces high modularity clusters (near optimal performance in may cases)
- Fast (roughly O(|V| log(|V|))

Weaknesses:

- Approximate method, no guarantee of global maximum of modularity
- Sometimes fails to find smaller clusters

In igraph this algorithm is **cluster_louvain**

Label Propagation

- Labels are shared among neighboring nodes until convergence

Procedure:

Initialize the labels for all nodes in the network.

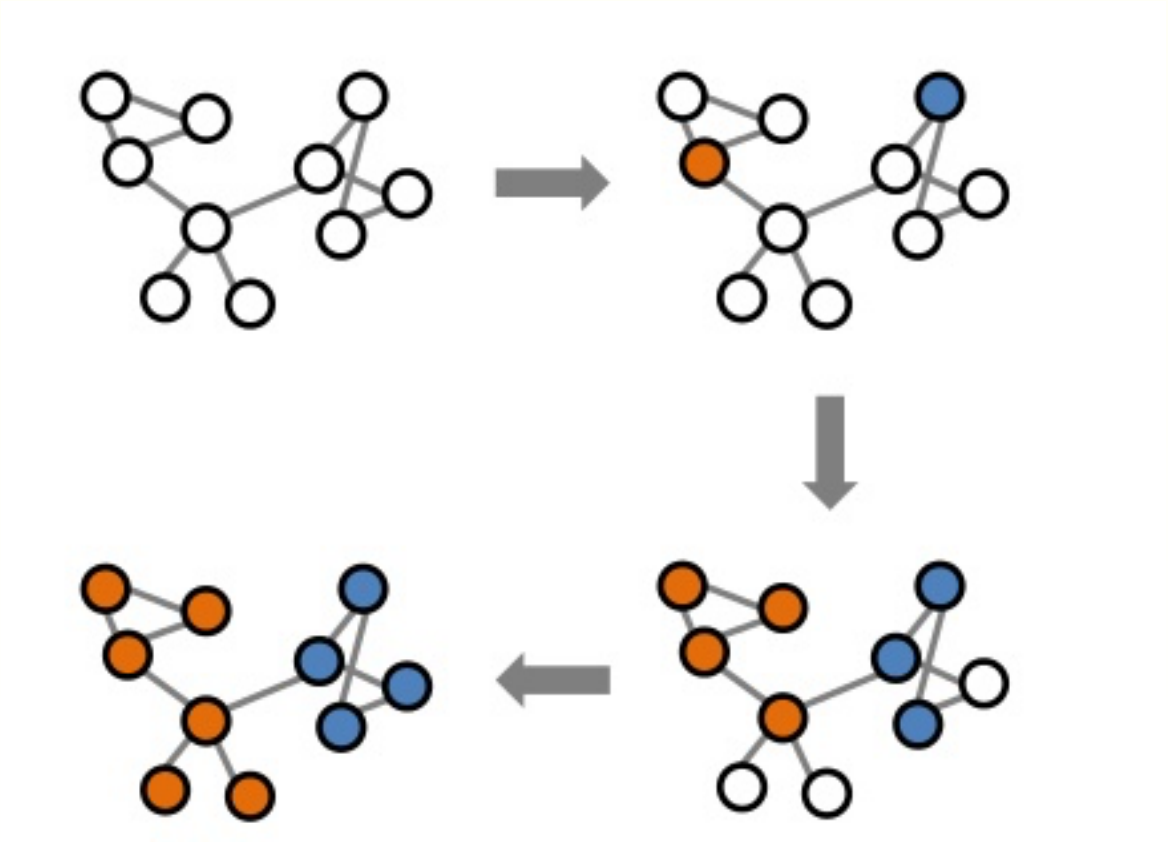
While not all nodes are labelled the same as maximum number of their neighbors

    Arrange the nodes in the network in a random order X.

    For each $x \in X$

        Set the label of x to the label of the largest number of its neighbors.

Label Propagation



https://www.slideshare.net/YelenaMejova/language-of-politics-on-twitter-03-analysis

Label Propagation

Strengths:

- In practice, finds high modularity clusters
- Fast (O(|V| + |E|))

Weaknesses:

- Solutions are not unique (typically the algorithm is run several times and solutions are aggregated).
- Random ordering can mean two clusters are merged into one
- Again, approximate, so no guarantees of optimality

In igraph this algorithm is **cluster_label_prop**

Clustering in Igraph

- These methods and a variety of others are already implemented in igraph
  - **cluster_edge_betweenness**
  - **cluster_optimal**
  - **cluster_louvain**
  - **cluster_label_prop**

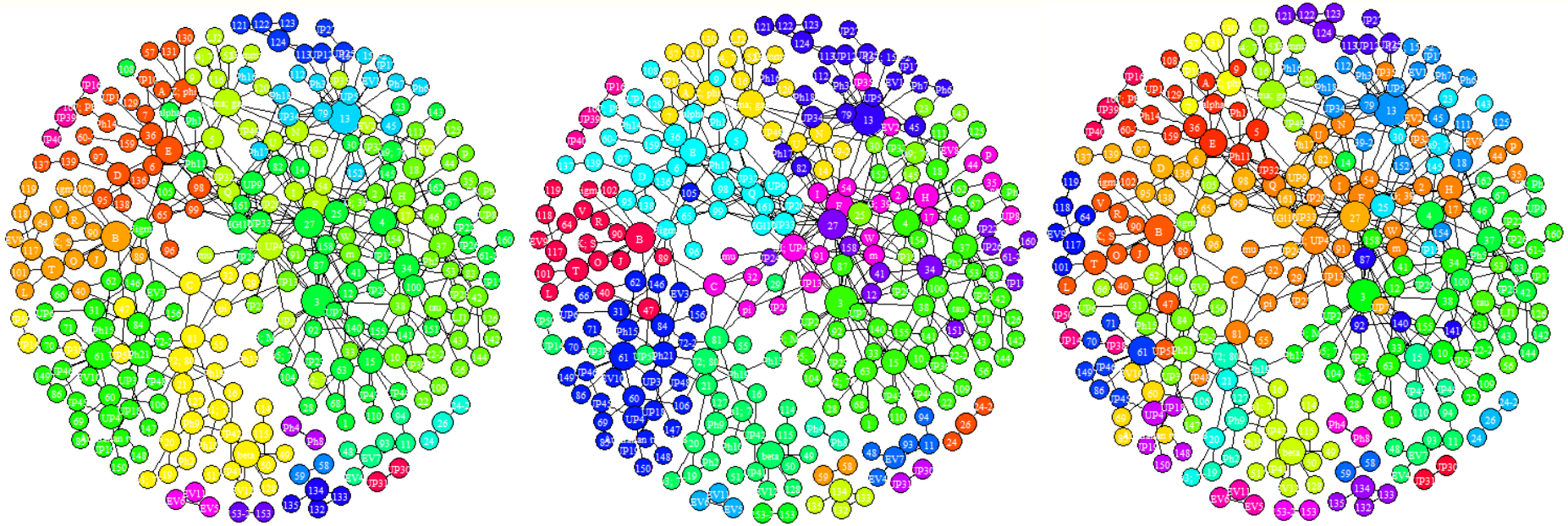
- Easy to measure the modularity of a given clustering
  - Useful when comparing methods for a specific problem

- Can also generate colored visualizations showing clusters

- In the next slide, we give a quick example from our own research

Example from research at SCADS

- Example of a real world network:
  - Protein Homology Graph



	Betweenness	Louvain	Label Propagation
Modularity	0.68	0.71	0.65
Time	0.4438 sec	0.0403 sec	0.0243 sec
# Clusters	18	19	34

Code to generate those plots

```
library(qgraph) library(igraph)
#clusters <- cluster_edge_betweenness(G)$membership
#clusters <- cluster_louvain(G)$membership
#clusters <- cluster_label_prop(G)$membership
colbar <- rainbow(max(clusters)+1)
V(G)$color <- colbar[clusters+1]

e <- get.edgelist(G)
l <- qgraph.layout.fruchtermanreingold(e,vcount=vcount(G),area=9*(vcount(G)^2),repulse.rad=(vcount(G)^3.1))
PageWeights=max(E(G)$weight+1)-E(G)$weight #pagerank uses affinity rather than distance weights on edges

plot(G,
    vertex.frame.color = "black", #set vertex border color
    vertex.label.cex =.6, #set label size
    vertex.label.color = "white", #set label color
    vertex.size=(page.rank(G,weight=PageWeights)$vector+0.016)/max(page.rank(G,weight=PageWeights)$vector)*7.5,
    edge.width = 1,  #set edge thickness
    edge.color = "black", #set edge color
    ylim=c(-.92,.92), #set image borders
    xlim=c(-.9,.9), #set image borders
    layout=l
)
```

Further Readings (References)

- Chapter 3 of Easley and Kleinberg (Strong and Weak Ties)
- Fortunato, Community Detection in Graphs, Physics Reports 486 (2010), 75--174

- Copies of both posted under the Lectures/041620/ folder