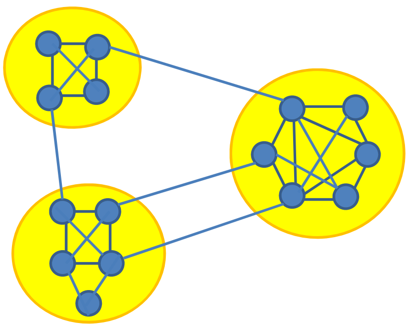
Communities

Community Structure

Communities:

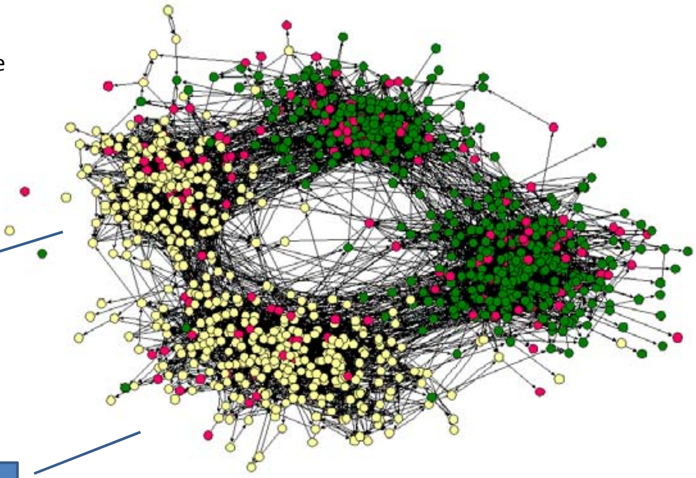
sets of tightly connected nodes

- People with common interests
- Proteins with equal/similar functions
- Web pages in the same topic
- …



School Friendship Network (USA)

<u>Races</u>:
Yellow - White Race
Green - Black Race
Pink - Other

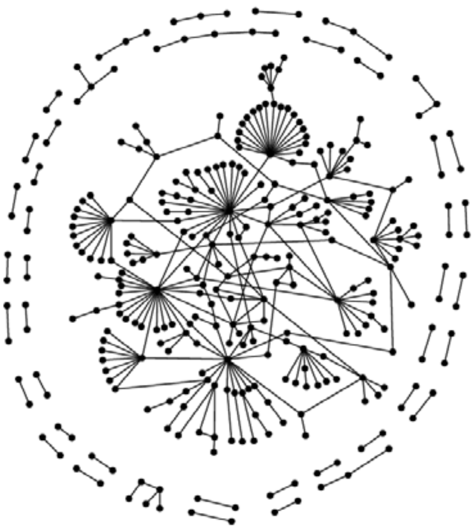


Middle-school

high-school

[J. Moody, *American Journal of Sociology* 2001]

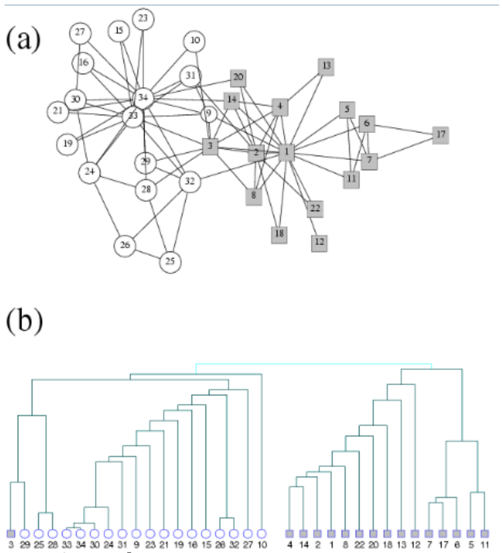
Yeast Protein Network



[Sergei Maslov and Kim Sneppen, *Science* 2002]

Karate Club

- Graph have weights



[W. W. Zachary, *Journal of Anthropological Research* 1977]

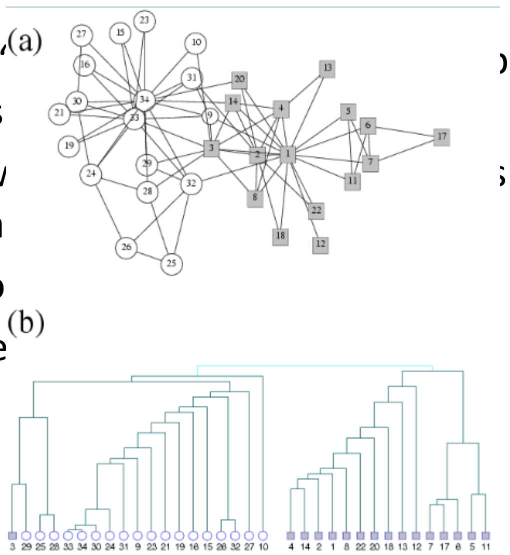
Networks

- Small world
- Long tail degree distribution (power law)
- High clustering
- Clustering coefficient
  - Per node v: The number of edges connecting v's neighbors divided by maximum possible number ($k_v(k_v-1)/2$) $C_i = \frac{2|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i(k_i-1)}$
  - Network: the average over all nodes

Traditional Clustering Methods

Hierarchical clustering

- $W_{i,j}$ – ‘ air of nodes
- Start v s in increa
- Tree o
- A slice structure



Weights

- The number of node independent path between the nodes
- The number of paths (not just node independent) weighted by the length (α^{ℓ})
  W= $\sum_{l=0}^{\infty}(\alpha A)^{l} = [I - \alpha A]^{-1}$
  A = adjacency matrix
  α is small

Adjacency Matrix

For a simple graph: a_{ij} =1 iff *i* and *j* are neighbors

- For undirected graphs A is symmetric
  - Has a complete set of real eigenvalues and an orthogonal eigenvector basis.
- ${A^{n}}_{ij}$ the number of paths of length *n* between *i* and *j*
  - tr(A^3)/6 – the number of triangles in the graph
- The principle eigenvector as a measure of centrality

Isolated Node

- Both weights are small
- Small nodes will not be joined to their natural community and left isolated
- Other pathologies

Edge Betweenness

- Edge betweenness = the number of shortest path passing thru the edge
  - An edge connecting communities will have high edge betweenness
  - Multiple SPs
- Progressively remove edges from the graph

The Algorithm

1) Calculate the betweenness for all edges in the network.
2) Remove the edge with the highest betweenness.
3) Recalculate betweennesses for all edges affected by the removal.
4) Repeat from step 2 until no edges remain.

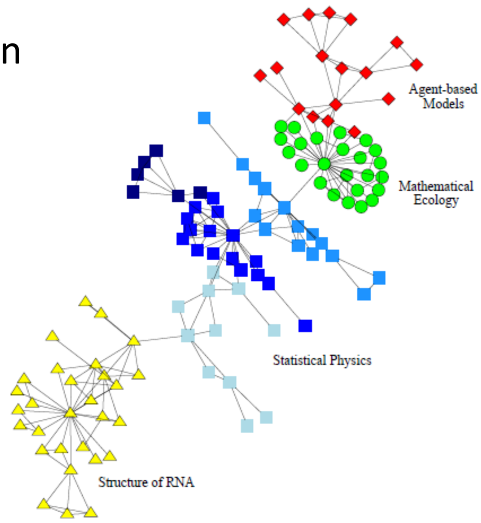
Why? Because if communities are connected by multiple edges, not all will have high betweenness

Complexity of (2) is O(mn), overall O(m^{2}n)

[Givran & Newman, PNAS 2002]

Applying B.C.

- Santa Fe collaboration



K-Medoids

- Can be used for clustering a graph
- Is known to converge to optimal solution
- Complexity:
  - Each iteration requires
    - Running SP from every new medoid
    - Reassigning nodes to clusters
    - Calculating new cluster center (medoid)

Approximating *k*-Medoids (GkM)

- Do not iterate!
- Randomly select node with a condition
  - Minimum distance from previous Medoids
- Node assignment to clusters
  - If roughly equidistance $\Rightarrow$ assign to smaller cluster
- How variable are the results? Not much! Why?
  - Min. distance spread the medoids
  - Large hubs

Other ways to Speed-up

- Graph pruning
  - Remove links with low weights
  - Keep only the top n links