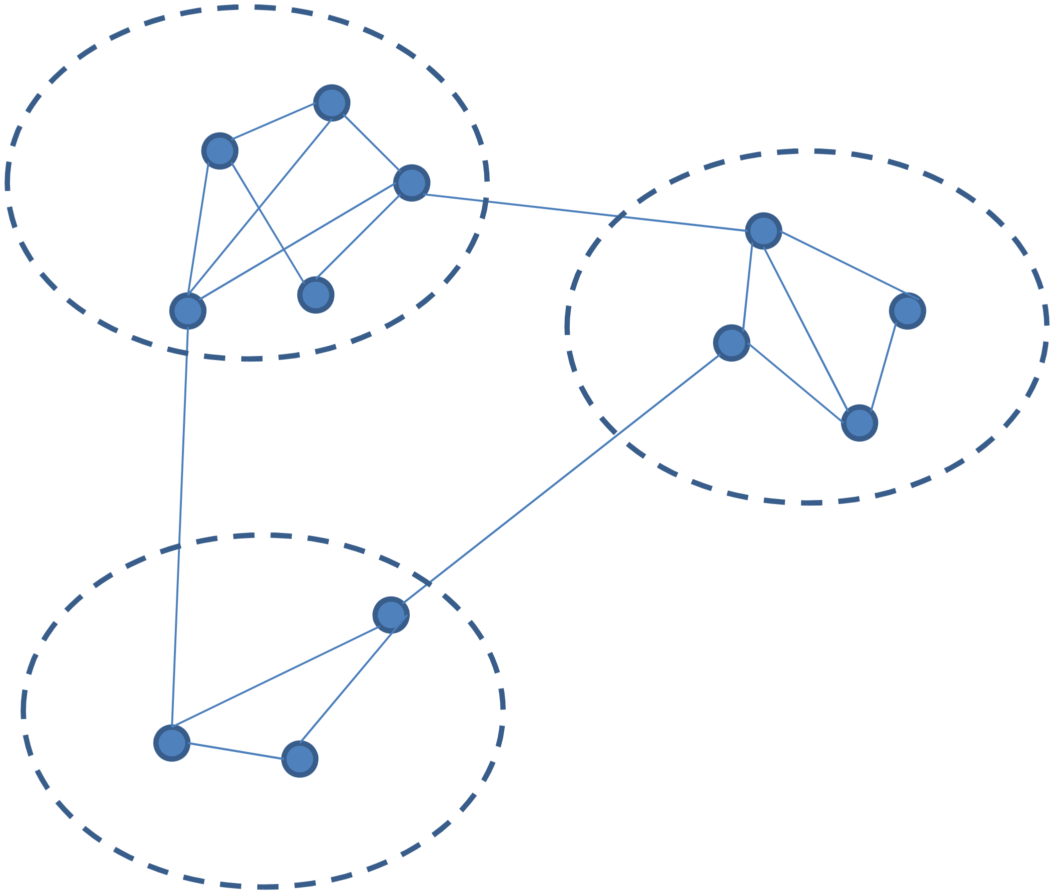
Community Detection

Community Detection

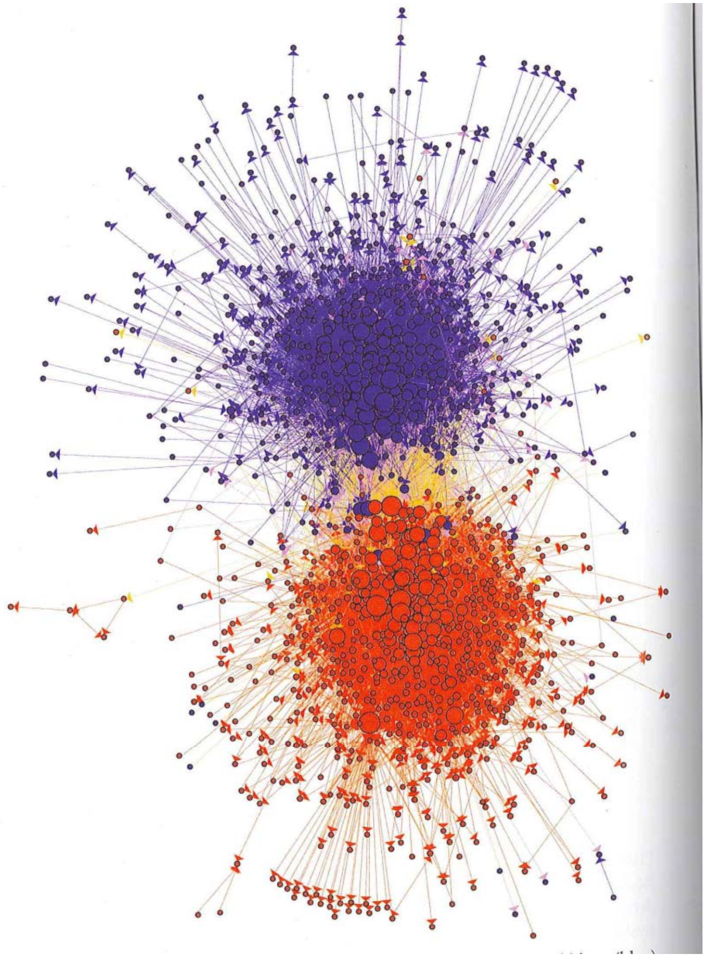


Community Detection: Partitioning of the network to partitions with a lot of edges inside and with a few with other partitions

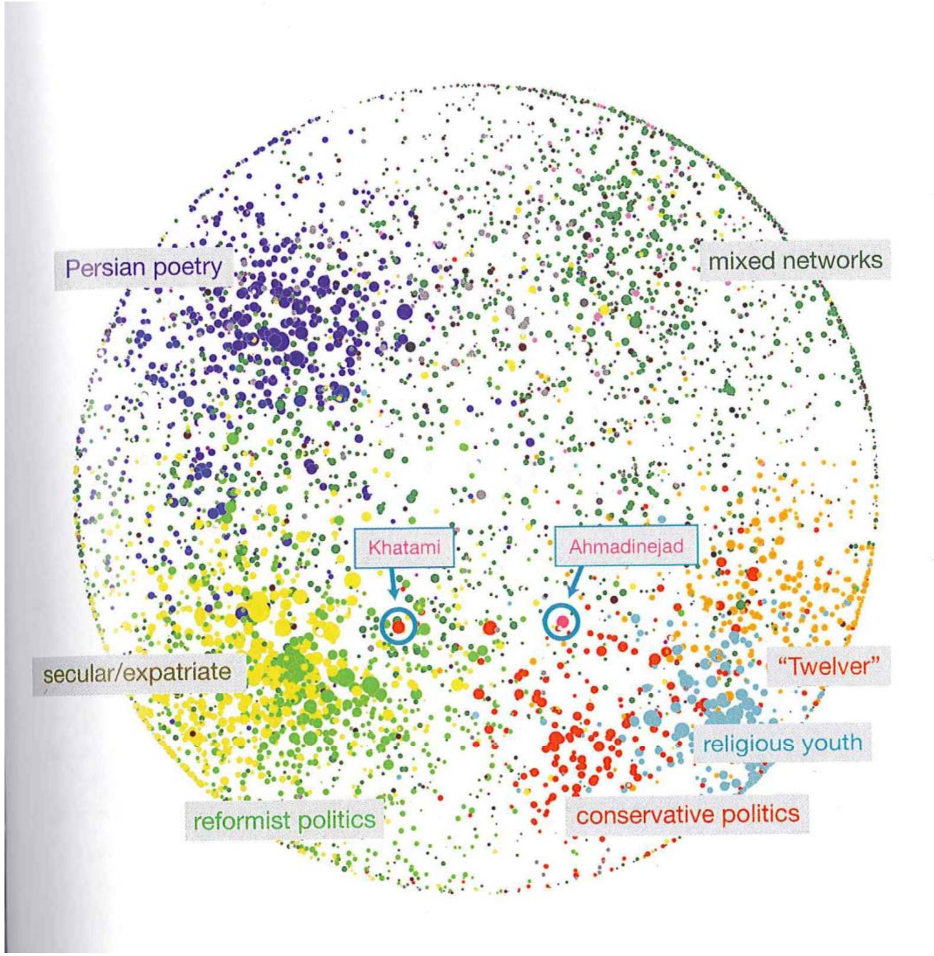
Applications

- Web graph
  - Can help for finding similar pages
- Recommendation systems
  - E.g., can recommend movies according based on friends preferences
- Sociology
  - Who interacts with whom?
  - E.g., Blogosphere
- Communication networks
- Biology

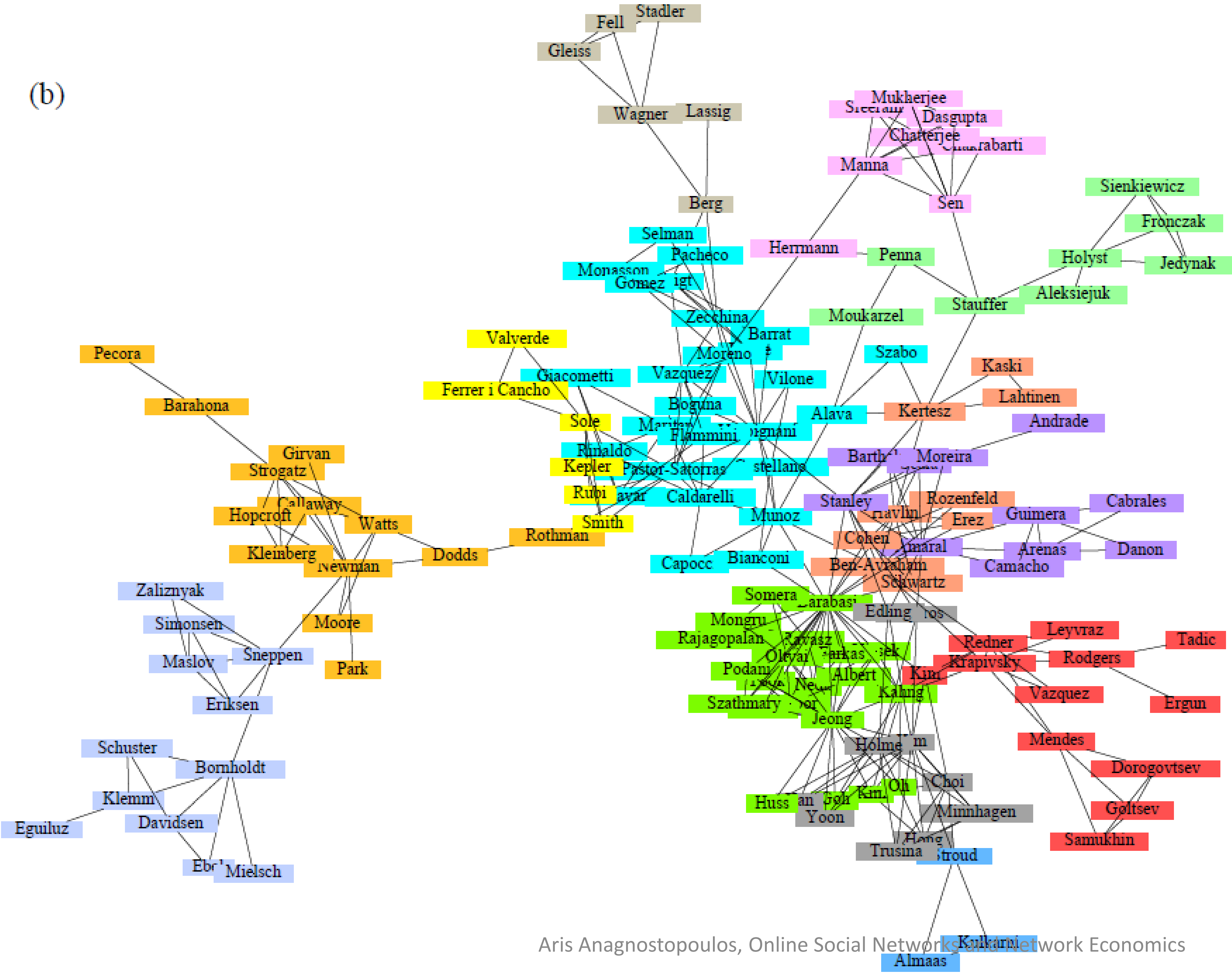
US Politics Blogs



Iranian Blogosphere



(b)


Stadler
Fell
Gleiss
Wagner
Lassig
Berg
Mukherjee
Dasgupta
Chatterjee
Manna
Sen
Sienkiewicz
Fronczak
Holyst
Jedynak
Aleksiejuk
Stauffer
Herrmann
Penna
Moukarzel
Selman
Pacheco
Monasson
Gomez
Zecchina
Barrat
Moreno
Vazquez
Vilone
Szabo
Boguna
Alava
Kertesz
Kaski
Lahtinen
Andrade
Flammini
Moreira
Valverde
Giacometti
Ferrer i Cancho
Sole
Rinaldo
Kepler
Pastor-Satorras
Rubi
Caldarelli
Munoz
Smith
Stanley
Rozenfeld
Erez
Guimera
Cabrales
Cohen
Arenas
Danon
Camacho
Ben-Avraham
Schwartz
Capocci
Bianconi
Pecora
Barahona
Girvan
Strogatz
Hopcroft
Watts
Kleinberg
Dodds
Rothman
Moore
Park
Zaliznyak
Simonsen
Maslov
Sneppen
Eriksen
Schuster
Bornholdt
Klemm
Eguiluz
Davidsen
Mielsch
Somera
Mongru
Rajagopalan
Oltvai
Podani
Albert
Szathmary
Jeong
Kahng
Edling
Redner
Leyvraz
Tadic
Rodgers
Krapivsky
Kim
Vazquez
Ergun
Mendes
Dorogovtsev
Goltsev
Samukhin
Holme
Choi
Oh
Huss
Goh
Yoon
Minnhagen
Hong
Trusina
Stroud
Kulkarni
Almaas

Methods to Discover Communities

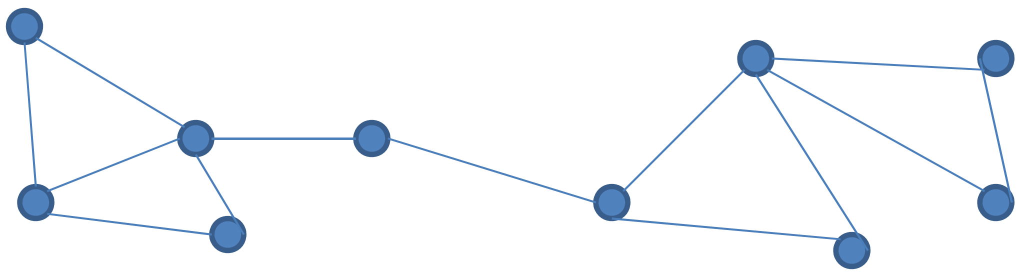
- Graph Partitioning
  - E.g. min-cut
  - Minimizing conductance and variants
- Hierarchical Clustering
  - Agglomerative methods (Bottom-up)
  - Divisive methods (Top-down)
- Partitional Clustering
  - Partition into k clusters so as to optimize some objective function
  - E.g., k-means, k-center, k-median
- Spectral Clustering
  - Based on spectral properties of the adjacency matrix
  - E.g., use Fiedler vector

The Newman-Girvan Algorithm

- A popular divisive method is the algorithm by Newman and Girvan
- Tries to find communities by discovering weak-ties
- Weak ties connect a lot of nodes with a lot of nodes
- This is measured by betweeness

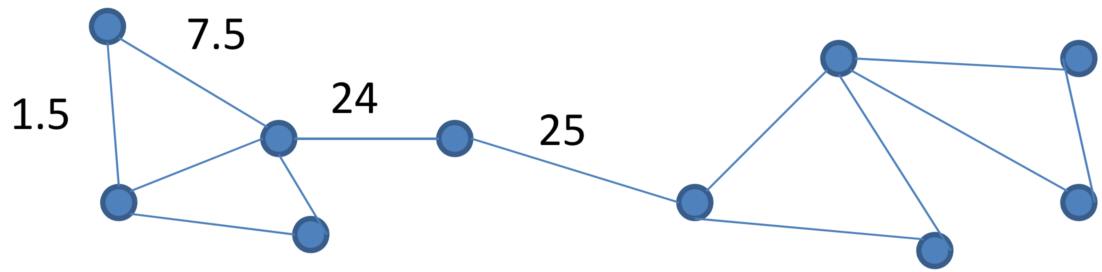
Betweeness

(Shortest-Path) Betweeness: For each edge measures in how many shortest path it belongs



Betweeness

(Shortest-Path) Betweeness: For each edge measures in how many shortest path it belongs



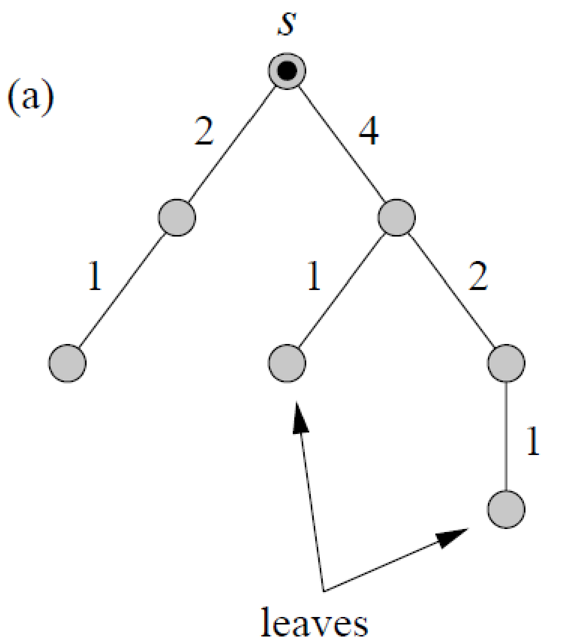
How to Compute Betweeness?

- Straightforward way:
  - For each pair of nodes compute the shortest paths in time O(m).
  - Total time = $O(mn^2)$
- Faster way: O(mn)

Computing Betweeness in O(mn)

For each node s compute shortest path tree using BFS (time=O(m))

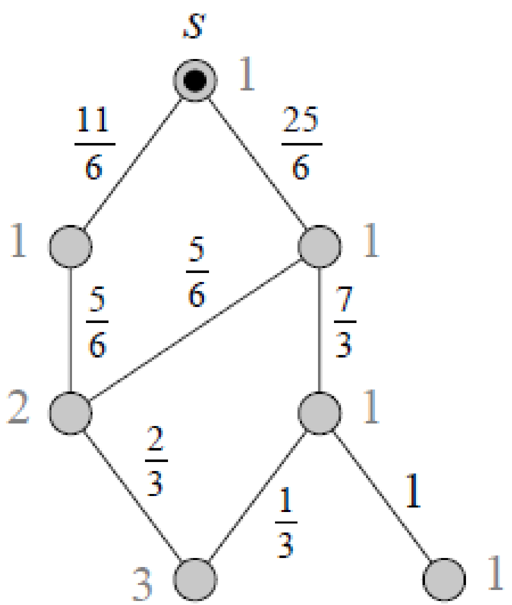
Simple case: Only one shortest path to each node



- Start from the leaves
- Score of edge = 1
- While we have not reached s
  - Go upward
  - Score of edge =
    1 + Sum of score of children

Computing Betweeness in O(mn)

General case: Multiple paths

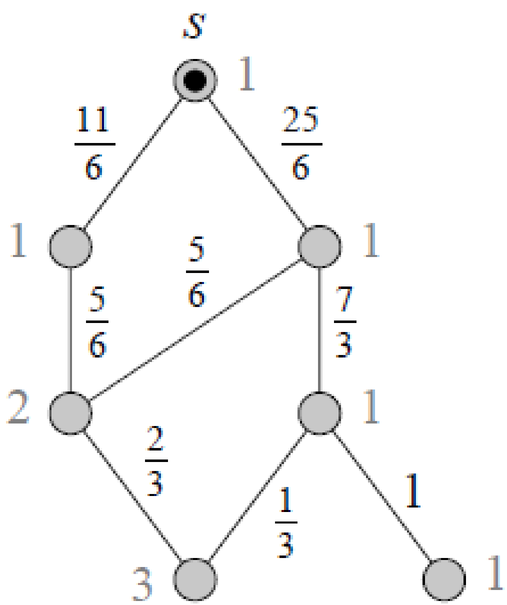


Step 1. Compute # shortest paths

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.

Computing Betweeness in O(mn)

General case: Multiple paths



Step 2. Compute edge score

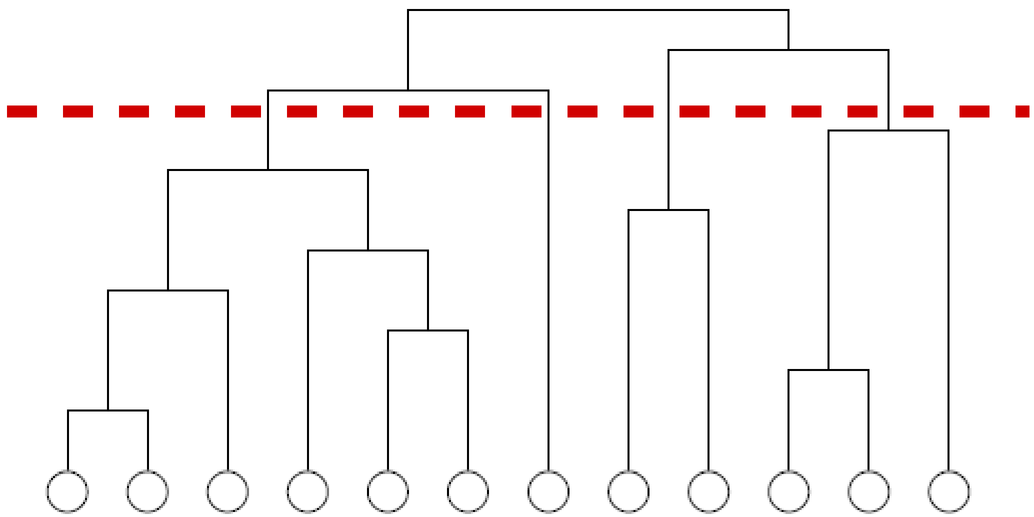
1. Find every "leaf" vertex t, i.e., a vertex such that no paths from s to other vertices go though t.
2. For each vertex i neighboring t assign a score to the edge from t to i of w_i/w_t.
3. Now, starting with the edges that are farthest from the source vertex s—lower down in a diagram such as Fig. 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.

Full Algorithm

- For i = 1 to m
  - For each node s
    - For each edge e compute score(s,e)
  - $betweeneness(e) = \sum_s score(s,e)$
  - Remove edge with highest betweeneness

- Total time $O(m^2 n)$

Result



- At the end we have a dendrogram corresponding to the clustering
- Circles correspond to graph nodes
- As we move up vertices 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?
- What is a good clustering
- Many ways to measure the quality of clusterings (e.g., k-means)
- A popular way for networks is modularity

Modularity

- Modularity Q is a score for clustering.
- Consider a partitioning $V=(V_1, V_2, \ldots, V_k)$

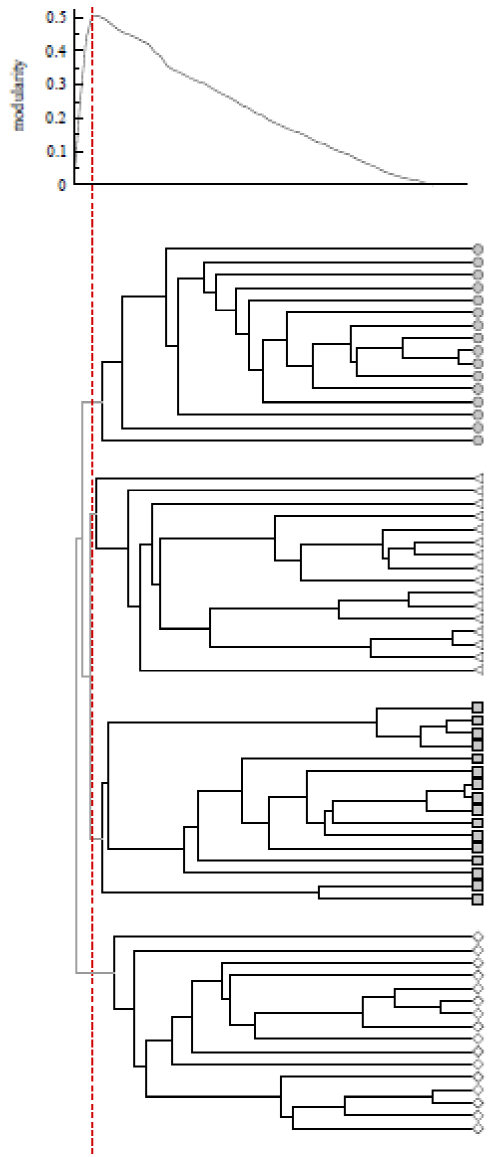
$$Q = \frac{1}{2m} \sum_{i=1}^{k} \sum_{u,v \in V_i} \left(A_{u,v} - \frac{d_u d_v}{2m} \right)$$

where

- m: # edges
- $A_{u,v} = 1$ if $(u,v) \in E$, 0 if not
- d_u: degree of node

- Measures how much the edges fall within a cluster compared with the case that a graph was a random graph

When to Stop



We compute the modularity for every level

We stop at the level when modularity is the highest

Alternative Approaches

- We can use modularity directly and cluster so as to optimize Q
- It is NP-hard
- Heuristics
  - Greedy
  - Connection of modularity with spectral theory

Centrality

- Another question we often have is which nodes are central?

- Many ways we can define central
  - Degree centrality
  - Betweeneness centrality
  - Closeness centrality

Degree Centrality

- With degree centrality we consider central the nodes with high degree:

$$\text{Degree centrality of node } v = \frac{d_v}{n-1}$$

Betweeness Centrality

- Betweeness centrality measures in how many shortest paths a node belongs

$$\text{Absolute betweeness centrality of node } v = \sum_{u,w \in V \setminus \{v\}} \frac{g_{uw}^{v}}{g_{vw}}$$

where

g_{uw}: # shortest paths between u and w

g_{uw}^{v}: # shortest paths between u and w passing through v

$$\text{Betweeness centrality of node } v = \frac{\sum_{u,w \in V \setminus \{v\}} \frac{g_{uw}^{v}}{g_{vw}}}{\left(\frac{n-1}{2}\right)}$$

Closeness Centrality

- With closeness centrality a node is central when its distance to other nodes is small

$$\text{Closeness centrality of node } v = \frac{\dfrac{1}{\sum_{u \in V} d(v,u)}}{\dfrac{1}{n-1}} = \frac{n-1}{\sum_{u \in V} d(v,u)}$$

where

d(v,u): distance between v and u