Mining Social-Network Graphs

Social Networks as Graphs

Clustering of Social-Network Graphs

Direct Discovery of Communities

A. Rajamaran, J. Leskovec, J. D. Ullman

Mining of Massive Datasets

- much information to be gained from the large-scale data from social networks
  - ie. *friends* on Facebook
- techniques for analyzing such networks
  - identifying <u>communities</u> – subsets of nodes with strong connections
  - similar to clustering – overlapping rather than partitioning
  - other properties – similarities of nodes, connectedness of a community, neighborhood sizes of nodes, transitive closure etc.

Social Networks as Graphs

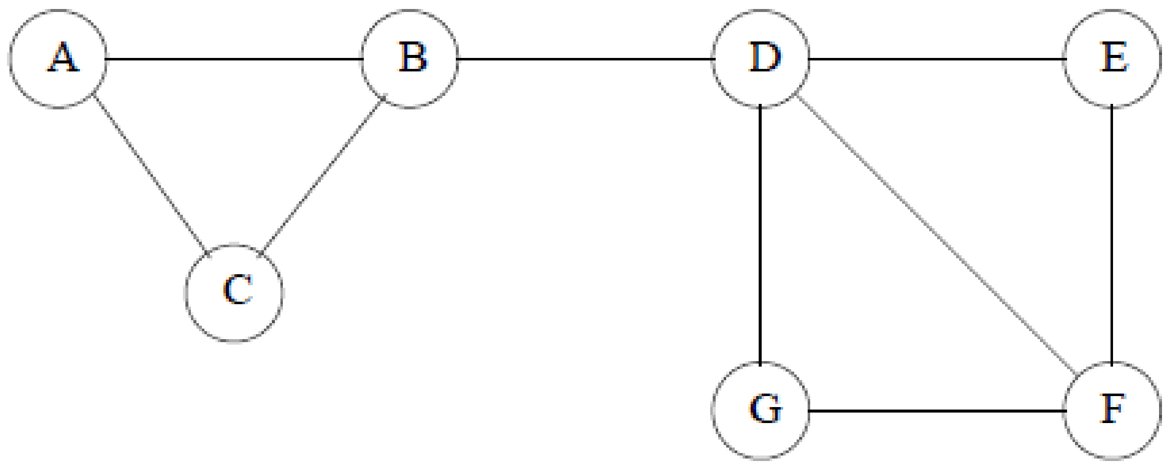
What is a Social Network?

- collection of entities that participate in the network
- at least one relationship between entities
- assumption of nonrandomness or locality.
  - relationships tend to cluster

Social Networks as Graphs

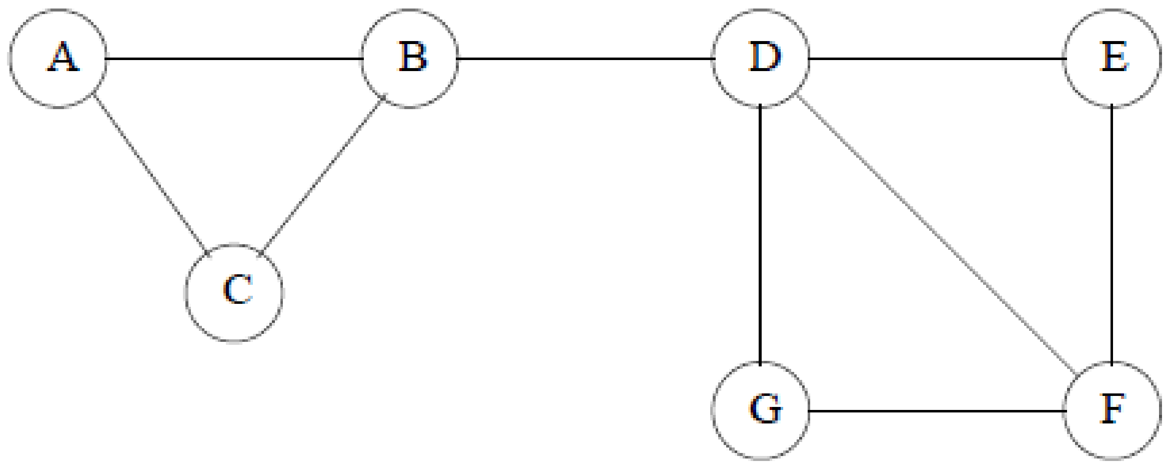
- naturally modeled as graphs - **social graphs**
- **entities** as nodes
- **relationships** as edges
- **degree associated with the relationship** as label of the edge
- undirected or directed

Social Networks as Graphs



Is this graph typical of a social network – does it exhibit locality of relationships?

Social Networks as Graphs



X, Y, Z – nodes of the graph, there are (X, Y) and (X, Z) edges

Probability of an edge (Y, Z)? **7/19 ≈ 0.368**

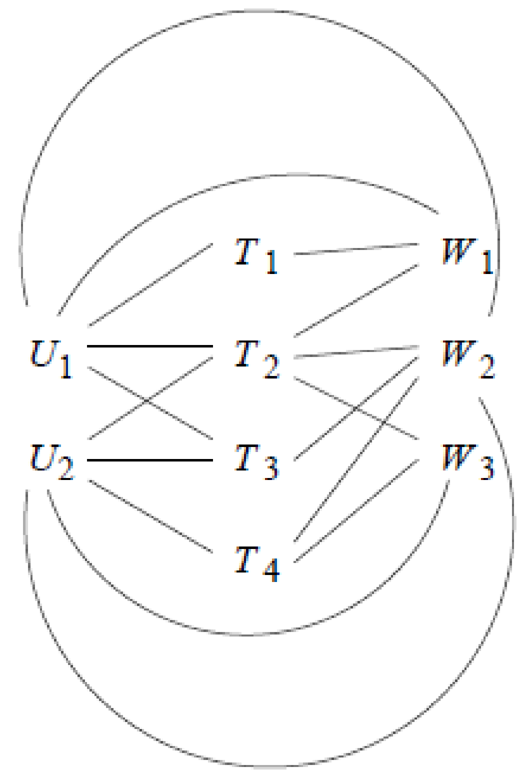
Probability of an edge (Y, Z) given that edges (X, Y) and (X, Z) exists? **9/16 ≈ 0.563** – significantly greater, so network exhibit the locality

Varieties of Social Networks

- telephone networks
- email networks
- collaboration networks
- Wikipedia
- etc.

Graphs With Several Node Types

- sometimes graphs are really formed from two or more types of nodes
  - ie. users, tags and Web pages
- represented by *k*-partite graph



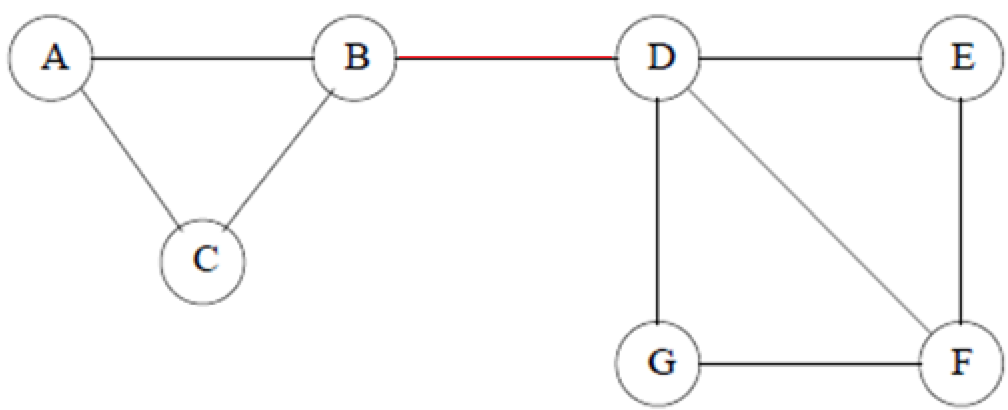
Clustering of Social-Network Graphs

Distance Measures for Social-Network Graphs

- labels of the edges might be usable as a distance measure
- what if edges are unlabeled?
- assumption that having and edge = close, otherwise = distant.
- 0 and 1, 1 and ∞ - but they are not true distance measures!

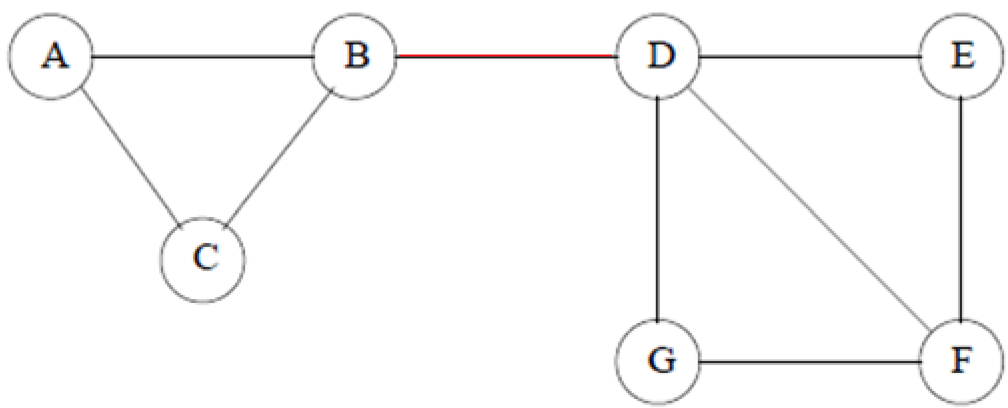
Applying Standard Clustering Methods

- hierarchical (agglomerative) approach:
  - combining some two nodes connected by an edge
  - edges between nodes from two clusters would be chosen **randomly**
  - random choices since distance on each edge is the same



Applying Standard Clustering Methods

- Point-assignment approach:
  - in k-means randomly picked nodes might be in the same cluster
  - randomly chosen and another as far away as possible doesn’t do much better (ie. E and G)
  - even choosing B and F – problem with assignment of D

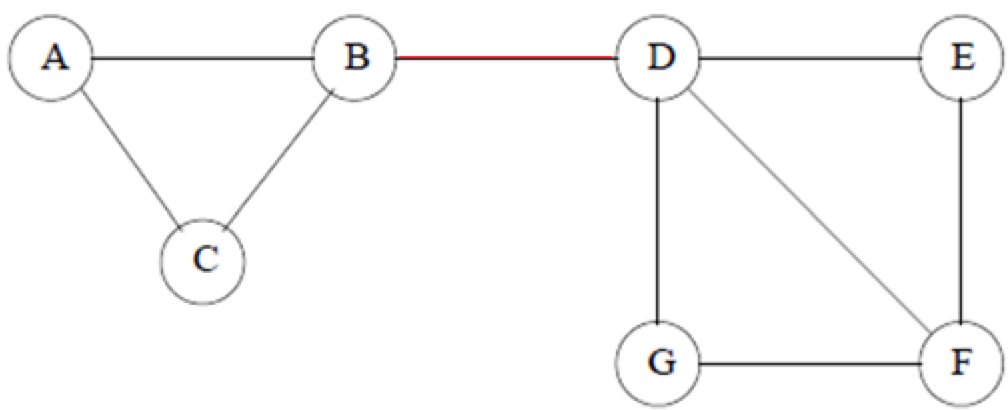


Betweenness

- method based on finding the edges that are **least likely to be inside a community**
- *betweenness* of and edge (*a, b*) is the number of pairs of nodes *x* and *y* such that the edge (*a, b*) lies on the shortest path between *x* and *y*
- high value suggests that (*a, b*) runs between two different communities – *a* and *b* do not belong to the same community

The Girvan-Newman Algorithm

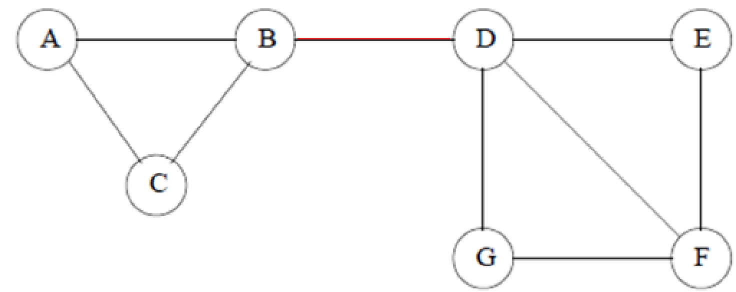
- to exploit betweenness, we need to calculate the number of shortest paths going through each edge
- Girvan-Newman Algorithm visits each node *X* once and computes the number of shortest paths from *X* to other nodes through each of the edges

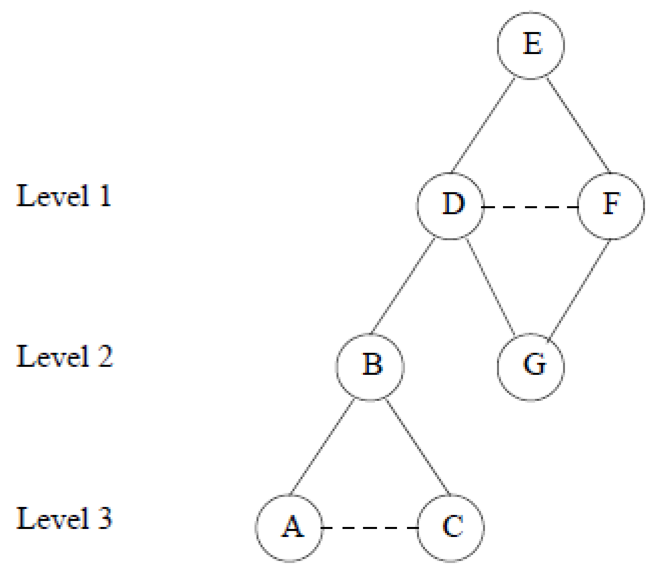


The Girvan-Newman Algorithm

STEP I

- breadth-first search starting at the node *X*
  - level of node is the length of the shortest path from *X*
  - thus edges at the same level can never be part of a shortest path from *X*

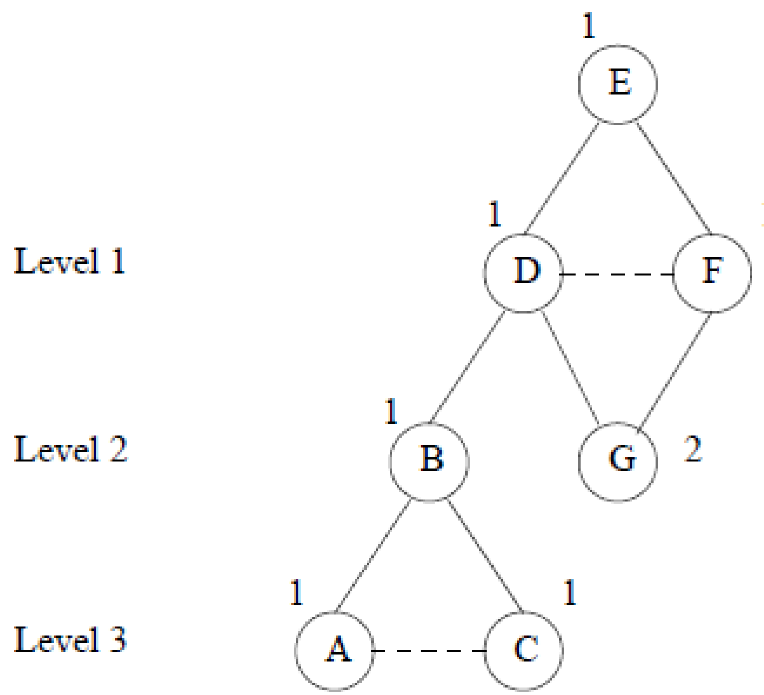




The Girvan-Newman Algorithm

STEP II

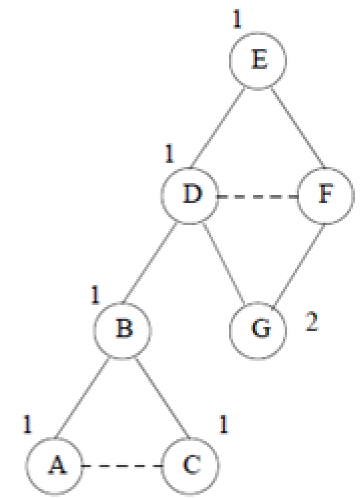
- label each node by the number of shortest paths that reach it from the root
  - the sum of the labels of its parents

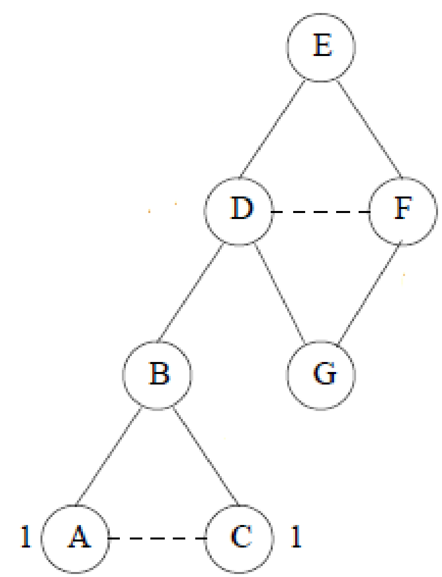


The Girvan-Newman Algorithm

STEP III

- do the calculation starting from the bottom according to rules:
  - leaf gets a credit of 1
  - node that is not a leaf gets a credit equal to 1 plus the sum of the credits of the edges to the level below
  - edge entering node from the level above is given a proportional share of that node

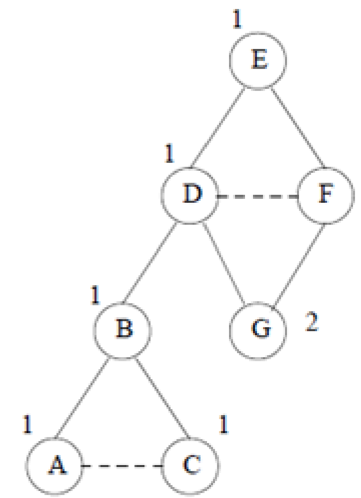


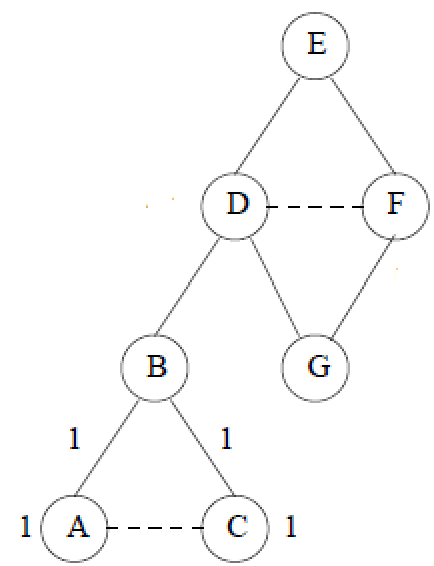


The Girvan-Newman Algorithm

STEP III

- do the calculation starting from the bottom according to rules:
  - leaf gets a credit of 1
  - node that is not a leaf gets a credit equal to 1 plus the sum of the credits of the edges to the level below
  - edge entering node from the level above is given a proportional share of that node

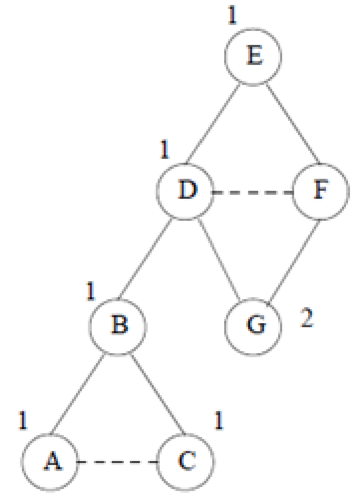


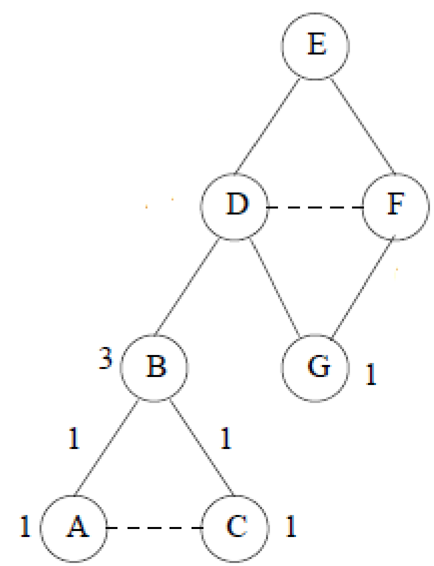


The Girvan-Newman Algorithm

STEP III

- do the calculation starting from the bottom according to rules:
  - leaf gets a credit of 1
  - node that is not a leaf gets a credit equal to 1 plus the sum of the credits of the edges to the level below
  - edge entering node from the level above is given a proportional share of that node

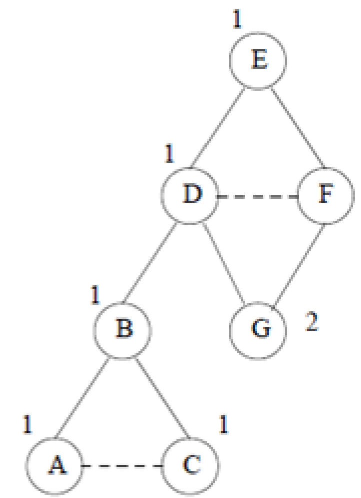


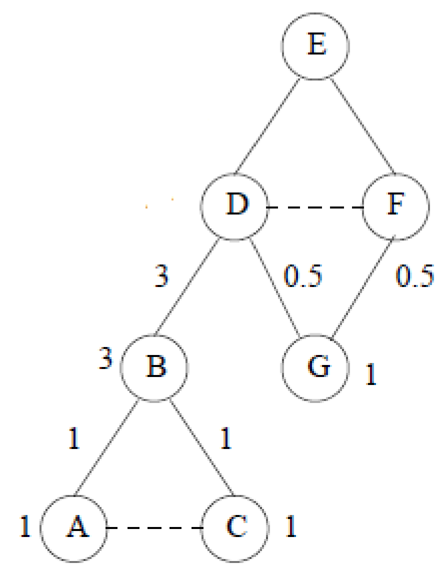


The Girvan-Newman Algorithm

STEP III

- do the calculation starting from the bottom according to rules:
  - leaf gets a credit of 1
  - node that is not a leaf gets a credit equal to 1 plus the sum of the credits of the edges to the level below
  - edge entering node from the level above is given a proportional share of that node

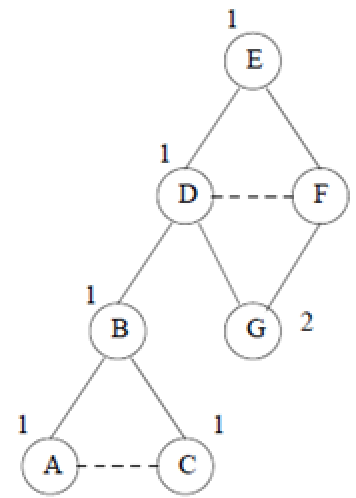


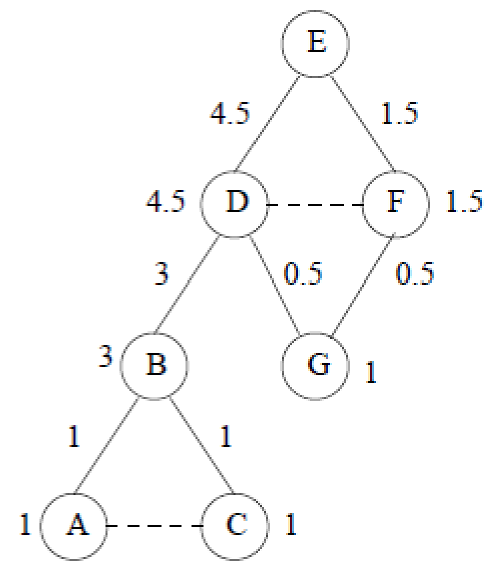


The Girvan-Newman Algorithm

STEP III

- do the calculation starting from the bottom according to rules:
  - leaf gets a credit of 1
  - node that is not a leaf gets a credit equal to 1 plus the sum of the credits of the edges to the level below
  - edge entering node from the level above is given a proportional share of that node

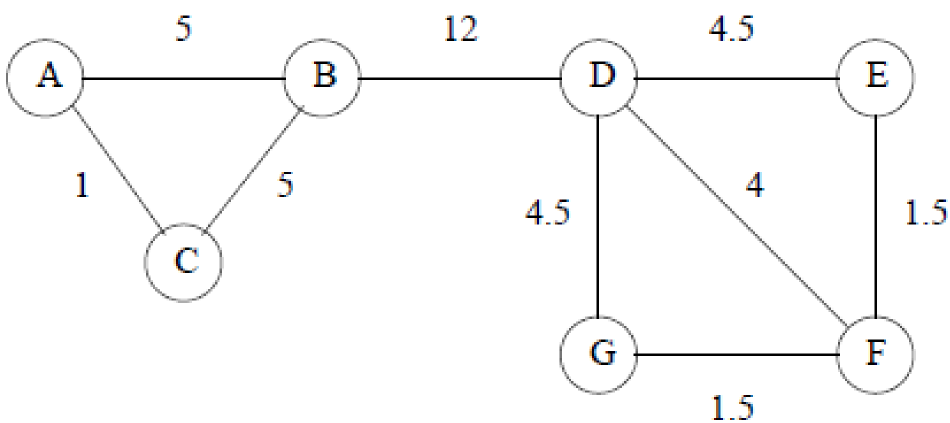


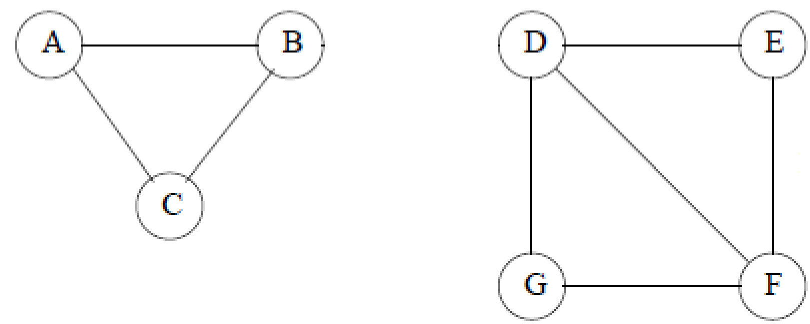


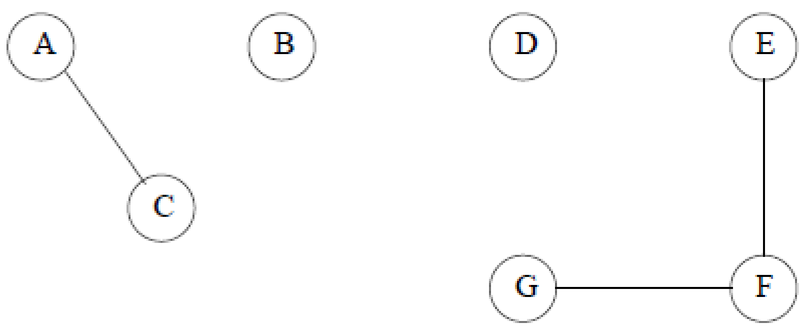
Using Betweenness to Find Communities

- to complete betweenness calculation:
  - repeat these steps for every node as the root and sum the contributions
  - divide by 2, because every shortest path will be discovered twice, once for each endpoints
- betweenness may behave like a distance measure, but is not exactly a distance measure
- ordering edges by betweenness and removing/adding nodes from graph

Using Betweenness to Find Communities







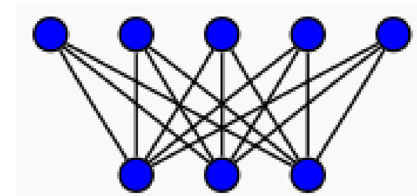
Direct Discovery of Communities

Finding Cliques

- we want to discover communities **directly** by **looking for subsets of the nodes** that have a **relatively large number of edges among them**
- first thought – finding a large clique
- NP-complete problem, even approximating the maximal clique is hard
- it is possible to have a set of nodes with almost all edges between them, yet with relatively small cliques

Complete Bipartite Graphs

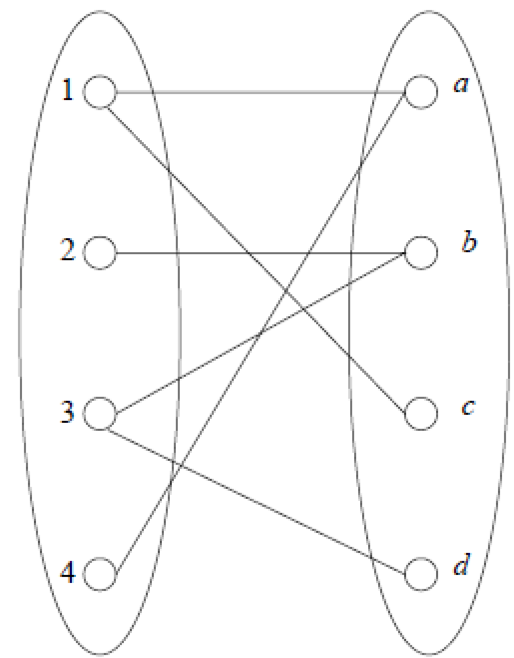
- graph that consists of *s* nodes on one side and *t* nodes on the other side with all *st* possible edges between them



- it is possible to guarantee that a bipartite graph with many edges has a large complete bipartite subgraph (unlike cliques)
- might be regarded as the nucleus of community

Finding Complete Bipartite Subgraphs

- large bipartite graph *G*
- we want to find instances of $K_{s,t}$ within it
- similar to finding frequent itemsets
- „items" on the *left* side ($K_{s,t}$ – *t* nodes there)
- assumption that $t \leq s$
- „baskets" on the *right* side
- member of the basket are the nodes from left side connected to that node
- support threshold *s*
- frequent itemset of size *t* and *s* of the baskets, in which all those items appear, form an instance of $K_{s,t}$



Why Complete Bipartite Graphs Must Exist

- if there is a community with n nodes and average degree d, then this community is guaranteed to have a complete bipartite subgraph $K_{s,t}$ when:

  $$n\binom{d}{t}/\binom{n}{t} \geq s$$

  which approximately is:

  $$n(d/n)^t \geq s$$

Thank you for attention!